

پایگاه داده‌ها

مؤلفین

مهندس سید ناصر آیت دکتر احمد فراهی

فهرست

فصل ۱ - مفاهیم اولیه سیستم پایگاه داده	۱
هدف کلی	۱
هدف رفتاری	۱
۱-تاریخچه	۲
۱-۱ ذخیره و بازیابی اطلاعات	۴
۲-۱ داده	۴
۳-۱ تعریف داده از دیدگاه ANSI	۴
۴-۱ اطلاع	۵
۵-۱ تعریف اطلاع از دیدگاه ANSI	۵
۶-۱ دانش	۵
۲-تعریف پایگاه داده	۶
۱-۲ تفاوت های بین روش فایلینگ و روش پایگاه داده ها	۷
۱-۱-۲ مراحل کار در روش فایلینگ	۷
۲-۱-۲ مراحل کار در روش پایگاهی	۸
۲-۲ اجزاء پایگاه داده	۱۰
۲-۲-۱ داده ها	۱۱
۲-۲-۲ سخت افزار	۱۲
۱-سخت افزار ذخیره سازی داده ها	۱۲
۲-سخت افزار پردازشگر	۱۲
۳-سخت افزار برقرار کننده ارتباط	۱۲
۲-۲-۳ نرم افزار	۱۳
۱-نرم افزار کاربردی	۱۴
۲-نرم افزار سیستمی	۱۴
۲-۲-۴ کاربر	۱۴
۱-برنامه نویسان کاربردی	۱۵
۲-کاربران واقعی یا نهایی	۱۵
۳-مدیر پایگاه داده	۱۵

تمرینات	۱۶
فصل ۲ - مدل سازی معنایی داده ها	۱۷
هدف کلی	۱۷
هدف رفتاری	۱۷
۱-مدلسازی معنایی داده ها	۱۸
۲-مدلسازی به روش ER	۱۹
۱-۲ نوع موجودیت	۱۹
۱-۱-۲ نمونه موجودیت	۲۰
۲-۱-۲ حالات یک موجودیت	۲۰
موجودیت قوی یا مستقل	۲۱
موجودیت ضعیف یا وابسته	۲۱
۲-۲ صفت	۲۱
رده بندی صفت	۲۱
۱-۲-۲ صفت ساده یا مرکب	۲۲
صفت ساده	۲۲
صفت مرکب	۲۲
۲-۲-۲ صفت تک مقداری یا چند مقداری	۲۲
صفت تک مقداری	۲۲
صفت چند مقداری	۲۳
۳-۲-۲ شناسه	۲۳
مفهوم مقدار هیچ (هیچ مقدار)	۲۳
۴-۲-۲ صفت واقعی یا مشتق	۲۳
صفت واقعی (ذخیره شده)	۲۴
صفت مشتق	۲۴
۳-۲ ارتباط	۲۴
نوع ارتباط	۲۴
۳-نمودار ER	۲۵
۱-۳ درجه نوع ارتباط	۲۸
۲-۳ ماهیت نوع ارتباط (اتصال)	۲۹

۳-۳	حد کاردینالیتی	۳۰
۴-مشکلات روش ER		۳۱
۴-۱ دام حلقه‌ای		۳۱
۴-۲ دام چند شاخه‌ای		۳۲
۴-۳ دام گسل		۳۲
۵-مدل سازی با روش EER		۳۳
۵-۱ تجزیه و ترکیب		۳۳
۵-۲ تخصیص و تعمیم		۳۵
۵-۳ زیر نوع‌های همپوشا و مجزا		۳۶
۵-۴ دسته بندی و وراثت		۳۷
۵-۵ تجمع		۳۸
۶-روش مدل سازی شیء UML		۳۹
۶-۱ مفاهیم اصلی		۳۹
۶-۲ نحوه نمایش مفاهیم		۴۱
۶-۳ خصوصیات کلی روش مدل سازی معنایی داده‌ها		۴۲
تمرینات		۴۴
فصل ۳ - معماری پایگاه داده		۴۵
هدف کلی		۴۵
هدف رفتاری		۴۵
۱-معماری سه سطحی پایگاه داده		۴۶
۲-شرح سطوح سه گانه:		۴۸
۲-۱ (نمای) ادراکی (مفهومی)		۴۸
۲-۲ دید (نمای) خارجی		۴۹
۲-۳ دید (نمای) داخلی		۵۰
۳-سایر اجزاء پایگاه داده‌ها		۵۳
۳-۱ کاربر		۵۳
۳-۲ زبان میزبان		۵۳
۳-۳ زبان داده‌ای فرعی		۵۴
زبان مستقل		۵۵

۵۵.....	زبان ادغام شدنی
۵۶.....	تمرینات
۵۷.....	فصل ۴ - سیستم مدیریت پایگاه داده
۵۷.....	هدف کلی
۵۷.....	هدف رفتاری
۵۸.....	۱-تعریف
۵۹.....	بخش ساختاری
۵۹.....	بخش عملیاتی
۵۹.....	بخش جامعیتی
۶۰.....	۲-رده بندی سیستم های مدیریت پایگاه داده ها
۶۰.....	از نظر مدل داده ای
۶۰.....	از نظر محیط سخت افزاری
۶۰.....	از نظر رده بندی کامپیوتر
۶۰.....	از نظر محیط سیستم عامل
۶۰.....	از نظر نوع معماری سیستم پایگاه داده ها
۶۱.....	از نظر معماری مشتری-خدمتگذار
۶۱.....	از نظر سیستم فایل
۶۱.....	از نظر متدولوژی زبان پایگاهی
۶۱.....	از نظر بهینه سازی پرس و جوها و درخواست های کاربر
۶۱.....	از نظر نوع تراکنش
۶۱.....	از نظر نوع پردازش
۶۱.....	۳-۱-اجزاء سیستم مدیریت پایگاه داده ها
۶۲.....	۳-۱-نمای بیرونی
۶۳.....	۳-۲-نمای درونی
۶۳.....	۳-۲-۱-لایه مدیریت محیط پایگاه داده
۶۴.....	۳-۲-۲-لایه هسته پایگاه داده
۶۴.....	۴-کاتالوگ سیستم و دیکشنری داده ها (متا داده ها)
۶۶.....	۵-پارامترهای شناخت DBMS
۶۷.....	۵-۱ پارامترهای مربوط به توانایی ها و کارایی سیستم

۷۰.....	۲-۵ تسهیلات و جنبه‌های دیگر.....
۷۲.....	۳-۵: مشخصات کلی سیستم.....
۷۲.....	۴-۵: پارامترهای مربوط به معماری پایگاه داده‌ها.....
۷۲.....	۵-۴-۱ سطح داخلی-فیزیکی.....
۷۴.....	۵-۴-۲ سطح ادراکی.....
۷۴.....	۵-۴-۳ سطح خارجی.....
۷۵.....	۵-۵ پارامترهای مربوط به زبان داده‌ای فرعی.....
۷۶.....	۶-محورهای اصلی مقایسه DBMS ها.....
۷۸.....	۷-روش مطالعه سیستم.....
۷۸.....	۸-رویه‌های مستند برای کاربران.....
۷۹.....	۹-هزینه‌ها.....
۸۰.....	۱۰-مدیر پایگاه داده (DBA).....
۸۰.....	۱۰-۱ اصطلاح تیم DBA.....
۸۱.....	۱۰-۲ مسئولیت‌ها.....
۸۱.....	۱۰-۳ وظایف.....
۸۵.....	تمرینات.....
۸۷.....	فصل ۵ - مدل‌ها و ساختارهای داده‌ای پایگاه داده‌ها.....
۸۷.....	هدف کلی.....
۸۷.....	هدف رفتاری.....
۸۸.....	۱-محیط انتزاعی.....
۹۱.....	۲-آشنایی با ساختار داده‌ای سلسله مراتبی.....
۹۱.....	۲-۱ عناصر ساختاری.....
۹۳.....	۲-۲ طراحی پایگاه سلسله مراتبی.....
۹۸.....	۲-۳ عملیات در پایگاه سلسله مراتبی.....
۹۹.....	۲-۴ برخی ویژگی‌های ساختار (و مدل) داده‌ای سلسله مراتبی.....
۱۰۰.....	۳-آشنایی با ساختار داده‌ای شبکه‌ای.....
۱۰۰.....	۳-۱ تعریف ساختار شبکه.....
۱۰۱.....	۳-۲ عناصر ساختاری.....
۱۰۲.....	۳-۳ طراحی پایگاه شبکه‌ای.....

۱۰۲	۳-۳-۱ طرز نمایش ارتباط " یک به چند "
۱۰۴	ارتباط بازگشتی
۱۰۴	۳-۳-۲ طرز نمایش ارتباط " چند به چند "
۱۰۵	۳-۴ عملیات در پایگاه شبکه‌ای
۱۰۶	۳-۵ برخی ویژگی‌های ساختار (و مدل) داده‌ای شبکه‌ای
۱۰۷	تمرینات
۱۰۹	فصل ۶ - پایگاه داده رابطه‌ای
۱۰۹	هدف کلی
۱۰۹	هدف رفتاری
۱۱۰	۱- بانک اطلاعات رابطه‌ای Relational DB
۱۱۰	۲- ساختار داده‌ای رابطه‌ای
۱۱۱	۳- تعریف رابطه
۱۱۱	۳
۱۱۲	۳-۲ تعریف رابطه
۱۱۳	۳-۳ تناظر بین مفاهیم رابطه‌ای و مفاهیم جدولی
۱۱۳	۳-۴ تعریف صوری جدول
۱۱۵	۴- ویژگی‌های رابطه
۱۱۶	۵- انواع کلید در مدل رابطه‌ای
۱۱۶	۵-۱ ابر کلید
۱۱۶	۵-۲ کلید کاندید
۱۱۷	۵-۳ کلید اصلی
۱۱۷	۵-۴ کلید بدیل
۱۱۸	۵-۵ کلید خارجی
۱۱۹	نکاتی در مورد کلید خارجی
۱۲۱	۶- انواع رابطه
۱۲۱	رابطه مبنا
۱۲۱	رابطه مشتق
۱۲۱	مفهوم دید
۱۲۲	۷- قواعد جامعیت در مدل رابطه‌ای

۱۲۳	۱-۷ انواع قواعد جامعیت
۱۲۳	۱-۱-۷ قواعد کاربری
۱۲۴	۲-۱-۷ متا قواعد
۱۲۴	۱-۲-۱-۷ قاعده جامعیت موجودیتی
۱۲۵	۲-۲-۱-۷ قاعده جامعیت ارجاعی
۱۲۵	تمرینات
۱۲۷	فصل ۷ - عملیات در پایگاه رابطه‌ای
۱۲۷	هدف کلی
۱۲۷	هدف رفتاری
۱۲۸	۱- تعریف
۱۲۹	۲- جبر رابطه‌ای (RA) و عملگرهای آن
۱۳۰	عملگرهای اصلی Select (B):
۱۳۰	عملگر اصلی Project (Π):
۱۳۰	الحاق (Join):
۱۳۱	عملگر تغییر نام (ρ):
۱۳۳	دستور (B):
۱۳۴	دستور (Π):
۱۳۵	عملگرهای پیوند (join)
۱۳۵	(X ضرب دکارتی : Cartesian product)
۱۳۵	خواص عملگر پیوند
۱۳۶	کاربردهای ضرب دکارتی (X)
۱۳۶	الحاق طبیعی Natural join (∞):
۱۳۷	$\alpha =$ semi join
۱۳۸	عملگر تقسیم Divide یا Division
۱۴۰	۳- عملگرهای کار بر روی داده‌ها
۱۴۰	۱-۳ عملگر درج (Insert)
۱۴۱	۲-۳ عملگر بهنگام سازی (UPDATE)
۱۴۱	۳-۳ عملگر حذف (DELETE)
۱۴۲	۴- کامل بودن جبر رابطه‌ای

۱۴۲	۵- حساب رابطه‌ای
۱۴۳	۵-۱ حساب رابطه‌ای تاپلی
۱۴۳	۵-۱-۱ شکل کلی عبارت حساب تاپلی
۱۴۴	۵-۱-۲ سور وجودی و سور همگانی
۱۴۴	سور وجودی
۱۴۴	سور همگانی
۱۴۴	۵-۱-۳ عبارت مطمئن
۱۴۵	۵-۲ حساب رابطه میدانی
۱۴۵	۶- بهینه سازی پرس و جوها
۱۴۷	۶-۱ تبدیل Q به Q.O
۱۴۷	۶-۲ سایر قواعد و عملگرهای بهینه سازی
۱۴۸	∞ sec crs
۱۴۸	∞ crs sec
۱۴۹	تمرینات
۱۵۱	فصل ۸- آشنایی با زبان رابطه‌ای SQL
۱۵۱	هدف کلی
۱۵۱	هدف رفتاری
۱۵۲	۱- مقدمه‌ای بر SQL
۱۵۳	۲- دلایل گستردگی SQL
۱۵۴	۳- دستورات تعریف داده‌ها DDL
۱۵۴	۳-۱ تعریف پایگاه داده
۱۵۵	۳-۲ تعریف شما
۱۵۵	۳-۳ جدول
۱۵۵	۳-۳-۱ تعریف جدول
۱۵۶	۳-۳-۲ اصلاح ساختار جدول
۱۵۸	۳-۳-۳ تغییر نوع داده‌های یک ستون جدول
۱۵۸	۳-۳-۴ حذف یک جدول
۱۵۹	۴- دستورات پرس و جو (مشاهده) داده‌ها
۱۶۲	۵- دستورات کار بر روی داده‌ها

۱۶۳	۵- ۱ دستور INSERT
۱۶۴	۵- ۲ دستور UPDATE
۱۶۶	۵- ۳ دستور DELETE
۱۶۷	۶- جستجوهای پیشرفته با عملگر پیوند
۱۶۸	۶- ۱ عملگر پیوند داخلی
۱۷۱	عملگر پیوند طبیعی
۱۷۱	۶- ۳ عملگرهای پیوند خارجی
۱۷۲	۶- ۳- ۱ پیوند خارجی چپ
۱۷۳	۶- ۳- ۲ پیوند خارجی راست
۱۷۵	۶- ۳- ۳ پیوند خارجی کامل
۱۷۶	۷- سایر عملگرها در جستجوهای پیشرفته
۱۷۶	۷- ۱ عملگر گروه بندی
۱۷۷	۷- ۲ عملگر مرتب سازی
۱۷۸	۷- ۳ توابع تجمعی
۱۷۹	۷- ۴ عملگر HAVING
۱۸۰	۷- ۵ عملگر BETWEEN
۱۸۱	۷- ۶ عملگر LIKE
۱۸۲	۸- استفاده از پرس و جوهای تودرتو
۱۸۲	۸- ۱ استفاده از زیرپرسش ها (پرسش های فرعی)
۱۸۳	۸- ۲ عملگر IN
۱۸۴	۸- ۳ عملگر EXISTS
۱۸۴	۹- دستورات کنترل مجوز دسترسی
۱۸۵	۹- ۱ دستور اعطا اختیارات
۱۸۶	۹- ۲ دستور لغو اختیارات
۱۸۶	۱۰- امکانات و ویژگی های SQL2 و SQL3
۱۸۶	۱۰- ۱ امکانات SQL2
۱۸۷	۱۰- ۲ امکانات SQL3
۱۸۸	تمرینات
۱۹۳	فصل ۹ - نرمال سازی

هدف کلی	۱۹۳
هدف رفتاری	۱۹۳
۱-تعریف رابطه نرمال	۱۹۴
۱-۱ دلایل لزوم نرمال بودن رابطه	۱۹۴
۲-۱ معایب رابطه نرمال	۱۹۴
۳-۱ مزایا و معایب رابطه غیر نرمال	۱۹۵
۲- مفاهیم تئوری وابستگی	۱۹۵
۱-۲ انواع وابستگی ها	۱۹۶
۲-۱-۱ وابستگی تابعی	۱۹۶
۲-۱-۲ وابستگی تابعی کامل	۱۹۶
۲-۲ حالت‌های وابستگی	۱۹۷
۳-۲ تعریف $F +$	۱۹۷
روش یافتن $F +$	۱۹۷
۲-۴ نمودار وابستگی‌های تابعی	۱۹۸
۳-صورت‌های نرمال	۱۹۹
۴-فرایند نرمال سازی	۲۰۰
۴-۱ رابطه 1NF	۲۰۱
۴-۲ رابطه 2NF	۲۰۲
۴-۳ رابطه 3NF	۲۰۳
۴-۴ رابطه BCNF	۲۰۴
۴-۵ رابطه 4NF	۲۰۴
۴-۶ رابطه 5NF	۲۰۵
تمرینات	۲۰۷
فصل ۱۰ - معماری‌های مختلف سیستم پایگاه داده	۲۰۹
هدف کلی	۲۰۹
هدف رفتاری	۲۰۹
۱-مقدمه	۲۱۰
۲-انواع معماری	۲۱۰
۲-۱ معماری متمرکز	۲۱۱

۲-۲ معماری نامتمرکز	۲۱۱
۱-۲-۲ معماری سیستم پایگاهی مشتری-خدمتگذار	۲۱۲
۱-۲-۲-۱ تعریف	۲۱۲
۲-۲-۲-۱ معماری پایگاهی مشتری-خدمتگذار	۲۱۳
۲-۲-۲-۳ طرحهای معماری	۲۱۵
۲-۲-۲-۱-۳ از نظر تعداد مشتری و خدمتگذار	۲۱۵
۲-۲-۲-۱-۳ از نظر پیکر بندی سخت افزاری	۲۱۵
معماری حول کامپیوتر بزرگ	۲۱۵
معماری حول شبکه	۲۱۶
۲-۲-۲-۱-۴ مزایای معماری مشتری-خدمتگذار	۲۱۶
۲-۲-۲ معماری سیستم پایگاهی توزیع شده	۲۱۷
۲-۲-۲-۱ تعریف	۲۱۷
۲-۲-۲-۲: پیکربندی سخت افزاری	۲۱۹
۲-۲-۲-۳ قواعد	۲۲۰
۲-۲-۲-۴ مزایا	۲۲۰
۲-۲-۲-۵ معایب	۲۲۱
۳-۲-۲ معماری با پردازش موازی	۲۲۱
۱-۳-۲-۲ طرح کلی معماری	۲۲۲
۲-۳-۲-۲ طرح ها	۲۲۲
۱-۲-۳-۲-۲ معماری با حافظه مشترک	۲۲۳
۲-۲-۳-۲-۲ معماری با دیسک های مشترک	۲۲۴
۳-۲-۳-۲-۲ معماری بی اجزا مشترک	۲۲۵
۴-۲-۳-۲-۲ معماری سلسله مراتبی	۲۲۶
۴-۲-۲ معماری چند پایگاهی	۲۲۶
۵-۲-۲ سیستم پایگاهی همراه	۲۲۷
۱-۵-۲-۲ تعریف	۲۲۷
۲-۵-۲-۲: معماری سیستم پایگاهی همراه	۲۲۸
تمرینات	۲۲۹
ضمیمه ۱ - مجموعه سئوالات	۲۳۱

۲۳۱	تست های سری ۱
۲۳۴	پاسخ تست های سری ۱
۲۳۵	تست های سری ۲
۲۳۸	پاسخ تست های سری ۲
۲۴۰	تست های سری ۳
۲۵۶	پاسخ تست های سری ۳
۲۶۱	تست های سری ۴
۲۶۷	پاسخ تست های سری ۴
۲۶۸	تست های سری ۵
۲۷۶	پاسخ تست های سری ۵
۲۷۹	تست های سری ۶
۲۹۲	پاسخ تست های سری ۶
۲۹۷	تست های سری ۷
۳۰۱	پاسخ تستهای سری ۷
۳۰۲	تست های سری ۸
۳۱۲	پاسخ تست های سری ۸
۳۱۵	تست های سری ۹
۳۱۹	پاسخ تست های سری ۹
۳۲۲	تست های سری ۱۰
۳۳۱	پاسخ تست های سری ۱۰
۳۳۵	ضمیمه ۲ - BNF ۹۹SQL _
۳۹۱	واژه نامه انگلیسی به فارسی
۴۰۷	واژه نامه فارسی به انگلیسی
۴۲۵	منابع و مآخذ

پیشگفتار

این کتاب با توجه به سر فصل تعیین شده برای دانشجویان دانشگاه پیام نور در رشته کامپیوتر تهیه و تنظیم شده است. در تهیه این کتاب سعی بر آن شده است تا مباحثی که برای تدریس درس پایگاه داده‌ها لازم به تدریس است، مطرح گردند. این کتاب مشتمل بر ده فصل می‌باشد.

در ابتدای کتاب لیست سر فصل مطالب قید شده است. در انتهای کتاب مجموعه‌ای از سؤالات تستی به همراه پاسخ نامه ارائه شده است. همچنین به منظور آشنایی بیشتر علاقمندان به ساختار پایه‌ای زبان SQL، در ضمیمه ۲ کتاب متن کامل SQL BNF ارائه شده است.

نظر به لزوم جاگذاری معادل فارسی کلمات تخصصی برای راحتی فهم دانشجویان دو واژه‌نامه به صورت انگلیسی به فارسی و فارسی به انگلیسی در انتهای کتاب آمده است. در صفحه پایانی کتاب لیست منابع و مآخذ نیز برای آگاهی دانشجویان ارائه شده است.

این اثر با دقت نظر فراوان کارشناسان مدیریت تولید مواد و تجهیزات آموزشی مورد ارزیابی قرار گرفت که بدینوسیله از جناب آقای اکبری به نمایندگی از آن عزیزان قدردانی می‌نماییم.

کتاب حاضر بعنوان منبع درسی در دانشگاه پیام نور اعلام شده که بعلت کوتاه بودن زمان امکان رفع کلیه ایرادات تایپی و نگارشی میسر نشد. لذا از کلیه صاحب نظران، اساتید و دانشجویان محترم تقاضا دارد نظرات پیشنهادی خود را در جهت اصلاح ایرادات این اثر، به آدرس پست الکترونیکی مولفین ارسال فرمایند. در پایان از آقای مهندس کامیار آهنگوب که در تنظیم و تدوین کتاب همکاری شایانی داشته‌اند سپاسگزاری می‌کنیم.

احمد فراهی
afaraahi@pnu.ac.ir

ناصر آیت
ayat@pnu.ac.ir

فصل ۱

مفاهیم اولیه سیستم پایگاه داده

هدف کلی

در این فصل ابتدا تاریخچه‌ای در رابطه با بحث پایگاه داده‌ها و ذخیره سازی داده‌ها مطرح و سپس انواع سیستم‌های پایگاه داده نام برده می‌شوند. در ادامه بعضی از مفاهیم کلیدی در پایگاه داده‌ها شرح داده خواهند شد. سپس مفهوم پایگاه داده بصورتی دقیق تر مورد بحث و بررسی قرار خواهد گرفت. با تعریف پایگاه داده‌ها برای درک بهتر موضوع، مراحل کار در مشی فایلینگ و پایگاه داده مورد بحث و بررسی قرار خواهند گرفت و در پایان اجزاء پایگاه داده مورد بحث و بررسی قرار گرفته و جایگاه مدیر پایگاه داده بررسی خواهد شد.

هدف رفتاری

در این فصل عناوین زیر مورد بحث و بررسی قرار می‌گیرند:

- انواع سیستم‌های پایگاه داده
- سیستم مدیریت پایگاه داده^۱
- اصطلاح ذخیره و بازیابی اطلاعات (شامل داده، اطلاع، دانش و...)
- تعریف پایگاه داده
- مراحل کار در روش فایلینگ
- مراحل کار در روش پایگاهی

- اجزاء پایگاه داده
- مفهوم داده، سخت افزار و نرم افزار در پایگاه داده‌ها
- انواع کاربران در پایگاه داده‌ها
- تعریف مدیر پایگاه داده و وظایف آن

۱- تاریخچه

یکی از متداول‌ترین و اصلی‌ترین اصطلاحات در مقوله انفورماتیک اصطلاح پایگاه داده‌ها می‌باشد. اصطلاح سیستم مدیریت پایگاه داده‌ها در معنای عام، یکی از سیستم‌های ذخیره و بازیابی اطلاعات است. پس از طراحی نسل اولیه سیستم‌های ذخیره و بازیابی اطلاعات، برای تقویت مکانیزم‌ها و الگوریتم‌های مرتبط با ذخیره، بازیابی و پردازش داده‌ها و همچنین به منظور تسهیل در انجام امور فوق، سیستم‌های مدیریت پایگاه داده‌ها شکل گرفتند.

تکنولوژی پایگاه داده‌ها از اواسط دهه شصت میلادی و به منظور توسعه سیستم‌های فایلینگ ایجاد شدند. طول دهه هفتاد مدل‌های پایگاه‌های داده سلسله مراتبی و شبکه‌ای توسعه یافته و مورد استفاده زیادی قرار گرفتند. در اوایل دهه هشتاد، شاخه‌ای از آن تکنولوژی به نام سیستم مدیریت پایگاه داده‌های رابطه‌ای مورد توجه بیشتری قرار گرفت و به عنوان تکنولوژی برتر شناخته شد و هم اکنون هم بسیار مورد استفاده قرار می‌گیرد. مهمترین خصیصه سیستم‌های مدیریت پایگاه داده‌ها، مستقل شدن برنامه‌های کاربردی از جنبه‌ها و خصوصیات محیط فیزیکی ذخیره سازی است. از اواسط دهه هشتاد تاکنون، سیستم‌های دیگری هم ایجاد و عرضه شد از جمله:

- سیستم فایلینگ معمولی داده‌ها
- سیستم مدیریت داده‌ها و جستجوها
- سیستم مدیریت پایگاه داده‌ها^۱

به مرور زمان بواسطه افزایش حجم و نوع اطلاعات نیازهای جدیدتری مطرح و

1. Relational Database Management System (RDBMS)

۳ مفاهیم اولیه سیستم پایگاه داده‌ها

پایگاه‌های داده متناسب با آنها نیز ارائه گردید که نمونه‌های آن در ذیل آمده است:

- سیستم مدیریت پایگاه دانش^۱
- سیستم معنایی^۲ مدیریت پایگاه داده‌ها
- سیستم هوشمند^۳ مدیریت پایگاه داده‌ها
- سیستم مدیریت پایگاه داده‌های شیئی گرا^۴
- سیستم مدیریت پایگاه داده‌های زمانمند^۵
- سیستم مدیریت پایگاه داده بلادرنگ^۶
- سیستم داده کاوی^۷
- سیستم مدیریت چند پایگاهی^۸

اما باید به این نکته اشاره کرد که بیشتر این سیستم‌ها در کاربردهای خاصی استفاده می‌شوند. امروزه نوع جدیدی از سیستم‌های مدیریت پایگاه داده‌ها بنام سیستم مدیریت پایگاه داده استنتاجی بصورت آکادمیک در حال طراحی می‌باشد که البته هنوز در محیط‌های تجاری و صنعتی نمودی پیدا نکرده است.

با توجه به آنچه گفته شد شاید بتوان سیستم‌های پایگاهی را به سه نسل زیر

تقسیم کرد:

- سیستم‌های پیش رابطه‌ای^۹
- سیستم‌های رابطه‌ای^{۱۰}
- سیستم‌های پس رابطه‌ای^{۱۱}

-
1. Knowledge Base Management System (KBMS)
 2. Semantic System
 3. Intelligent System
 4. Object Oriented Database Management System
 5. Temporal Database Management System
 6. Real-Time Database Management System
 7. Data Mining System
 8. Multi Database Management System
 9. Pre Relational Systems
 10. Relational Systems
 11. Post Relational Systems

شاید بتوان گفت علت اصلی تفاوت سیستم مدیریت پایگاه داده‌ها با بقیه سیستمها در وجود سیستم حصاری نفوذ ناپذیری به نام سیستم مدیریت پایگاه داده است که هر گونه دستیابی به داده‌ها باید از طریق این سیستم انجام شود. سیستم مدیریت پایگاه داده در واقع انقلابی در بانکهای اطلاعاتی به شمار می‌آید. در همین اواخر دو تحول دیگر هم در تکنولوژی بانک اطلاعاتی پدید آمد:

- طراحی و ایجاد پایگاه‌های داده توزیع شده^۱ تحت شبکه‌های مختلف
- طراحی و ایجاد سیستم‌های مدیریت پایگاه داده برای کامپیوترهای شخصی

برای آشنا شدن ذهن خواننده تعریف چند اصطلاح پایه‌ای در سیستم‌های پایگاه داده در ذیل ارائه می‌گردد. لازم به ذکر است که تعاریف مربوط به اصطلاحات ذکر شده، بعضاً در کتاب‌های مختلف با کمی تغییر نوشته شده‌اند. ولی مفهوم اصلی آنها بسیار نزدیک بهم هستند.

۱-۱ ذخیره و بازیابی اطلاعات

اصطلاح ذخیره و بازیابی اطلاعات در واقع به مجموعه‌ای از الگوریتم‌ها و تکنیکها اطلاق می‌گردد که در طراحی و تولید یک سیستم بکار گرفته می‌شود و به کاربر امکان می‌دهد تا اطلاعات (اسناد، مدارک، متون، اصوات و تصاویر و...) خود را ذخیره سازی، بازیابی و پردازش کند. این گونه داده‌ها ممکن است ساختمانند، نیم ساختمانند و یا حتی ناساخته باشند.

۲-۱ داده

اصطلاح داده در مفهوم کلی عبارت است از نمایش ذخیره شده کلیه موجودیتهای واقعیات و رخدادها که در تصمیم گیری بکار می‌آیند.

۳-۱ تعریف داده از دیدگاه ANSI

استاندارد ANSI برای اصطلاح داده تعاریف زیر را ارائه کرده است:

1. Distributed Data Base

- هر نمایشی که توسط انسان یا یک سیستم مکانیکی خودکار معنایی به آن قابل انتساب باشد.
- نمایش واقعیات، پدیده‌ها، مفاهیم یا شناخته‌ها به طرزی صوری و مناسب برای برقراری ارتباط، تفسیر یا پردازش توسط انسان یا هر دستگاه خودکار.
- به طور کلی می‌توان گفت که داده ارزشهای واقعی هستند که از طریق مشاهده و تحقیق بدست می‌آیند.

۱-۴ اطلاع

هر نوع داده پردازش شده (ساخت یافته) را اطلاع می‌نامند. این تعریف یک تعریف بسیار ساده است که بیانگر تفاوت بین دو اصطلاح داده و اطلاع است. ولی بطور کلی می‌توان گفت اطلاع مجموعه داده‌هایی است که در تصمیم‌گیری بکار می‌روند و اساساً کمیتی است نسبی و وابسته به وضعیت مشخص، زمان مشخص و نیز خود شخص (یا سیستم) تصمیم‌گیرنده. در واقع، شخص تصمیم‌گیرنده با تفسیر داده‌ها در یک وضعیت مشخص اطلاعات لازم برای تصمیم‌گیری را به دست می‌آورد.

۱-۵ تعریف اطلاع از دیدگاه ANSI

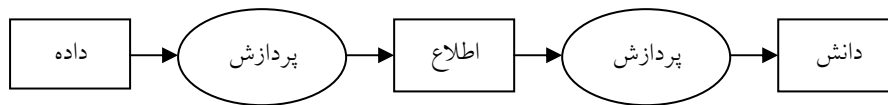
استاندارد ANSI برای اصطلاح اطلاع تعریف زیر را ارائه کرده است:
معنایی که انسان از طریق توافقات و قراردادهای شناخته شده‌ای به داده منتسب می‌کند.

۱-۶ دانش

اصطلاح دانش عبارت است از نمایش نمادین بخش‌هایی از دنیای واقعی. به بیانی دیگر، دانش نوعی شناخت است که از یک مجموعه از اطلاعات، براساس یک مجموعه از قواعد مشخص بدست می‌آید.

نکته ۱: بعضی از تئوریسین‌ها داده را همان مقدار واقعا ذخیره شده و اطلاع را معنای آن می‌دانند، بنابراین اطلاع و داده با هم فرق دارند، اطلاع دارای خاصیت ارتباط دهنده و انتقال دهنده است درحالی‌که داده مجرد این خاصیت را ندارد.

نکته ۲: داده‌ها حالت منفرد و مجزا دارند و لزوماً اطلاعی از آنها بدست نمی‌آید مگر اینکه به نحوی بهم مرتبط شوند و معنایی به آنها منتسب شود و باید دانش را نوعی اطلاع سطح بالاتر دانست در واقع هم اطلاع و هم دانش حاصل عملیاتی روی داده هستند، ولی نوع عملیات لازم برای حصول آنها متفاوت است. با این اوصاف رابطه بین سه مفهوم داده، اطلاع و دانش بصورت زیر قابل نمایش می‌باشد:



شکل ۱-۱ رابطه نمادین بین داده، اطلاع و دانش

با توجه به توضیحاتی که در بالا ارائه شد، اکنون توضیحی دقیقتر از مفهوم پایگاه‌داده را ارائه می‌دهیم.

۲- تعریف پایگاه‌داده

پایگاه‌داده‌ها با توصیفی جامع‌تر، مجموعه‌ای است از داده‌ها که بصورت مجتمع و تا حد ممکن بصورت مرتبط بهم و با کمترین افزونگی ذخیره شده‌اند که این مجموعه تحت مدیریت یک سیستم کنترل متمرکز برای استفاده یک یا چند کاربر قرار گرفته‌اند.

شاید در نگاه اول تعریف ارائه شده در مورد پایگاه‌داده‌ها کمی مبهم به نظر برسد. در تشریح کلی سیستم پایگاه‌داده‌ها می‌توان گفت که یک سیستم پایگاه‌داده مجموعه‌ای از داده‌های بهم وابسته است که از افزونگی بی‌حاصل و مضر مبرا است و برای کاربردهای گوناگون استفاده می‌شود. داده‌ها به گونه‌ای ذخیره شده‌اند که از برنامه‌هایی که آنها را به فرمت می‌گیرند مستقل هستند و راه یافت مشترک کنترل شده برای درج، حذف، تغییر و بازیابی داده‌های موجود استفاده می‌شود و داده‌ها به گونه‌ای ساخت یافته است که پایه‌ای برای توسعه برنامه‌های کاربردی آینده فراهم می‌سازد. این داده‌ها توسط یک سیستم مدیریت پایگاه‌داده‌ها مدیریت می‌شود. با این وصف می‌توان دریافت که هر

مجموعه‌ای از فایلها یا هر مجموعه‌ای از اطلاعات ذخیره شده لزوماً یک پایگاه‌داده‌ها نیست.

با توجه به مطالب فوق می‌توان چنین نتیجه گرفت که برای ایجاد پایگاه‌داده‌ها وجود حداقل یک سیستم مدیریت پایگاه‌داده‌ها به عنوان سیستم واسطه الزامی است. بارزترین برتری یک پایگاه‌داده نسبت به سیستم بانک‌های داده قبلی (سیستم‌های فایلینگ)، سیستم مدیریت پایگاه‌داده می‌باشد. با این وصف لازم است بین اصطلاحاتی مانند بانک داده، بانک اطلاعاتی، پایگاه‌داده و پایگاه اطلاعاتی تفاوت قائل شویم.

در بررسی محیط یک پایگاه‌داده‌ها لازم است به این نکته توجه شود که محیط واحد، مجتمع و مشترک ذخیره سازی لزوماً به این معنا نیست که چنین محیطی از نظر فیزیکی و محل جغرافیایی واحد و یکپارچه است. بلکه محیط پایگاه‌داده‌ها از لحاظ منطقی، یکپارچگی دارد. وضع پایگاه‌داده‌ها در سطح فیزیکی بستگی به معماری سیستم پایگاه‌داده‌ها دارد، پایگاه‌داده‌ها می‌تواند در عین واحد مجتمع و مشترک بودن در سطح منطقی، از نظر فیزیکی نامتمرکز و توزیع شده باشد.

۲-۱ تفاوت‌های بین روش فایلینگ و روش پایگاه‌داده‌ها

با توجه به تعاریف بالا ممکن است این سؤال پیش بیاید که اساساً استفاده از روش فایلینگ در طراحی برنامه‌ها به چه صورت بوده و چه تفاوتی با روش پایگاهی دارد. برای درک بهتر موضوع ابتدا روش فایلینگ برای طراحی برنامه‌ها را بصورت کلی بیان می‌کنیم.

۲-۱-۱ مراحل کار در روش فایلینگ

سیستم مورد درخواست برای مکانیزه کردن انتخاب و مورد بررسی و پردازش قرار می‌گیرد تا مشخصات نیازها تعیین گردد.

- مراحل اولیه لازم برای طراحی و پیاده سازی سیستم مورد نظر انجام می‌گیرد.
- مشخصات سیستم و زیر سیستم‌های احتمالی و وظایف هر کدام از آنها تعیین می‌گردد.

- تعدادی فایل اطلاعاتی برای ذخیره سازی فایلها طراحی می‌گردد. این فایلها معمولاً به صورت منفرد در سطح سیستم طراحی می‌گردند.
- برنامه مورد درخواست با استفاده از یک زبان برنامه نویسی تهیه می‌گردد. برنامه مورد نظر به همراه برنامه‌های مرتبط با فایلها اطلاعاتی، مجموعه نرم‌افزاری سیستم را پوشش می‌دهند.
- یک مجموعه کامل شامل نرم‌افزار و سخت‌افزار و احتمالاً مکانیزمی جهت برقراری ارتباط بین چند سخت‌افزار فراهم می‌گردد.
- مجموعه‌ای از تست‌ها جهت بررسی هر زیر سیستم و در نهایت بررسی کل سیستم اعمال می‌گردد.

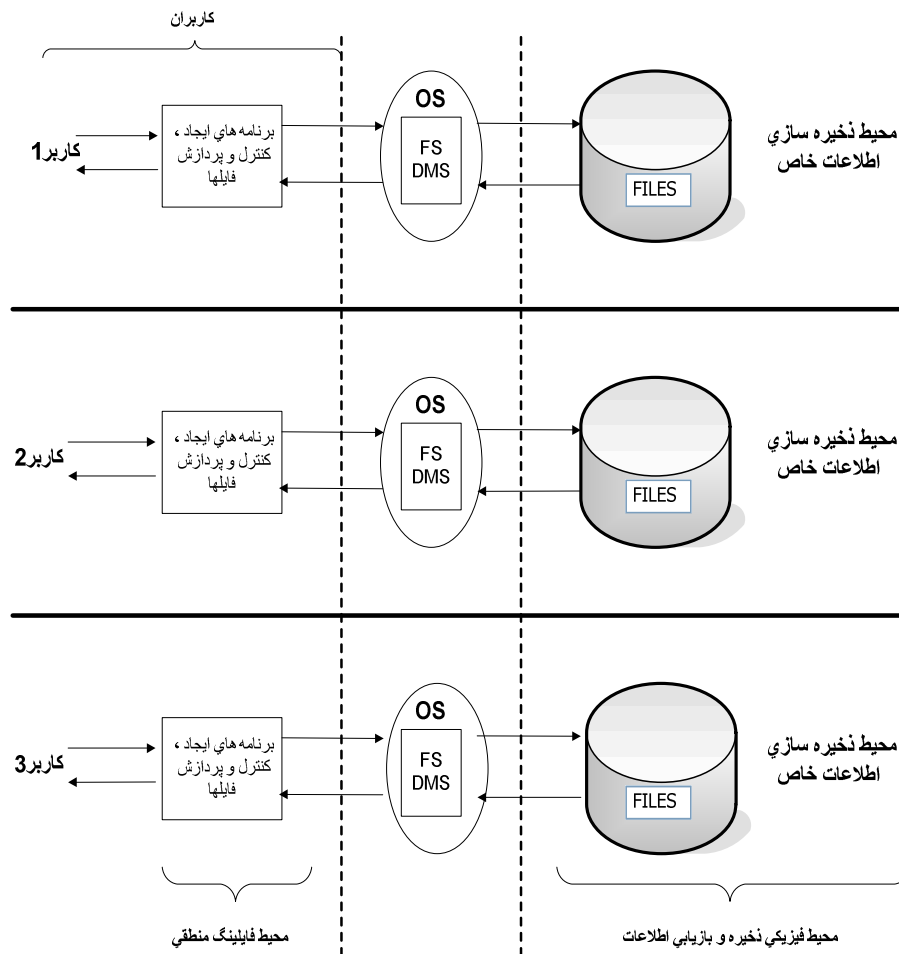
با این وصف به راحتی می‌توان دریافت که در روش فایلینگ، داده‌ها در واقع چند مجموعه مجزا و نامجتمع (از لحاظ منطقی و فیزیکی) و تا حدود زیادی نامرتب با هم و بدون مدیریت متمرکز خواهند بود. نمایی از روش فایلینگ در شکل ۱-۲ آمده است.

۲-۱-۲ مراحل کار در روش پایگاهی

با عنایت به روش کار در سیستم فایلینگ، در ادامه مراحل کار در روش پایگاهی معرفی می‌گردد.

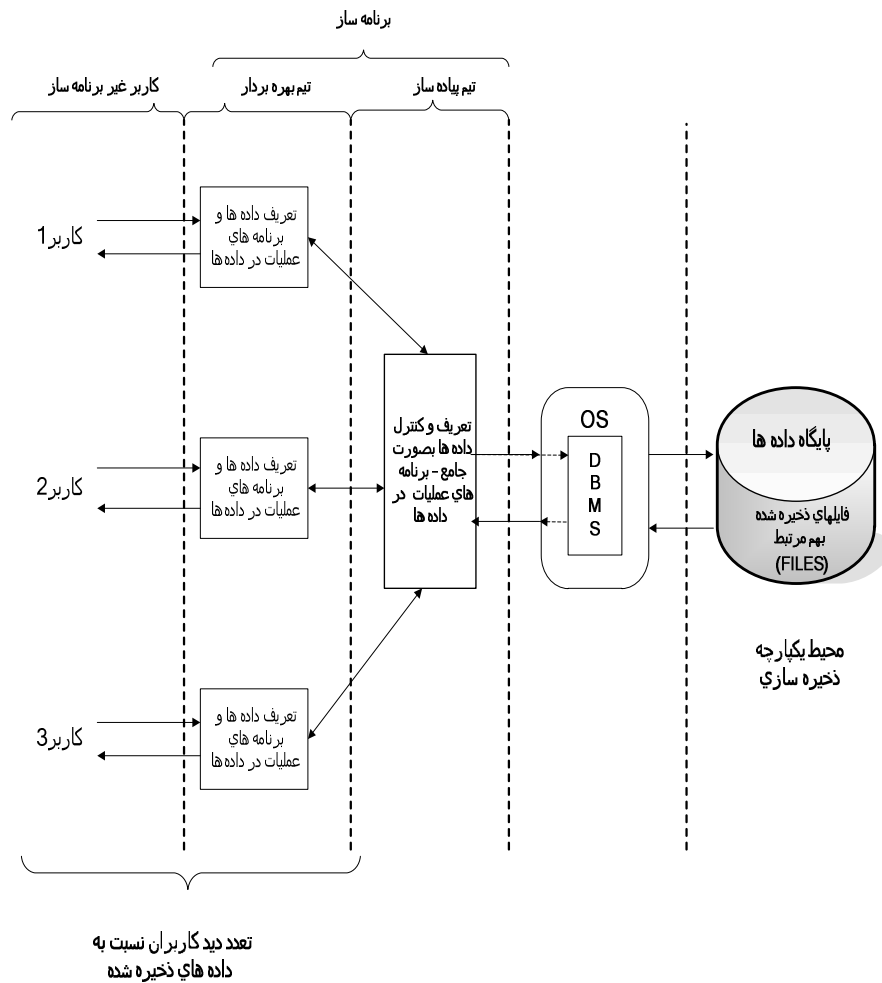
- کلیه نیازهای اطلاعاتی و پردازشی مجموعه مورد نظر بصورت یکپارچه مورد مطالعه و تحلیل قرار می‌گیرد. داده‌های مورد نظر، مدل سازی می‌گردند و مشخصات سیستم و وظایف آن بصورت جامع تعیین می‌گردند.
- یک یا چند پایگاه‌داده بعنوان سیستم مدیریت متمرکز انتخاب می‌گردد. طراحی‌های لازم در سطوح مختلف پایگاه‌داده تعریف و طراحی می‌گردند.
- واسطه‌های مورد نظر برای کاربران واحدهای مختلف (و بعضاً مدیران سیستم) طراحی می‌گردند.
- تست‌های لازم بر روی قسمت‌های مختلف سیستم اعمال می‌گردند.

۹ مفاهیم اولیه سیستم پایگاه داده‌ها



شکل ۱-۲ نمای کلی از روش فایلینگ

با این وصف می‌توان مشاهده کرد که اساس روش فایلینگ بر مبنای ساختار اطلاعاتی یکپارچه طراحی شده است و مبنای کار مدیریت یک پارچه و متمرکز داده‌ها می‌باشد. برای درک بهتر موضوع شمایی از روش پایگاهی در شکل ۱-۳ آمده است.



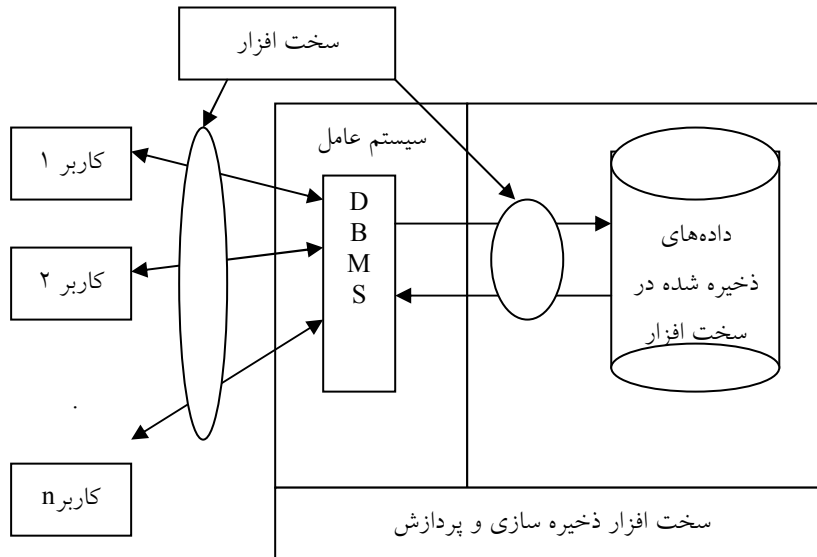
شکل ۳-۱ شمایی از روش پایگاهی

۲-۲ اجزاء پایگاه داده

همانگونه که در شکل زیر مشاهده می شود هر سیستم پایگاه داده از چهار جزء اساسی تشکیل می شود:

- داده ها
- سخت افزار

- نرم افزار
- کاربر



شکل ۱-۴ شمای تصویری ارتباط بین اجزاء پایگاه داده

شکل ۱-۴ یک نمای کلی از اجزاء معرفی شده به همراه نحوه ارتباط بین آنها را نشان می دهد. در ادامه هر یک از اجزاء مذکور مورد بحث و بررسی قرار خواهند گرفت.

۲-۲-۱ داده ها

یک پایگاه داده گنجینه ای از داده ها است که در کل مجتمع شده^۱ و به اشتراک گذاشته شده^۲ است. منظور از مجتمع شدگی، اتحاد چندین فایل داده به صورت مجموعه ای است که آن را به نام پایگاه داده می شناسیم و به همین علت است که تمام آن افزونگی ها که در سیستم پردازش فایلها وجود داشت از میان می رود. منظور از مشترک بودن پایگاه داده ها این است که اطلاعات موجود در پایگاه داده ها بین

1. Integrated

2. Shared

استفاده کنندگان مختلف به اشتراک گذاشته می‌شود. داده‌های ذخیره شده در یک سیستم پایگاهی عبارتند از:

- داده‌های کاربران
- داده‌های سیستمی

۲-۲-۲ سخت‌افزار

یک پایگاه‌داده جهت استقرار به مجموعه مناسبی از تجهیزات سخت‌افزاری نیاز دارد. سخت‌افزارها به سه دسته تقسیم می‌گردند:

- سخت‌افزار ذخیره سازی داده‌ها
- سخت‌افزار پردازشگر
- سخت‌افزار برقرار کننده ارتباط

در ادامه به شرح هر یک از انواع سخت‌افزارها در محیط پایگاه‌داده‌ها خواهیم پرداخت:

۱- سخت‌افزار ذخیره سازی داده‌ها

منظور همان رسانه‌های ذخیره سازی خارجی است ولی باید دانست که رسانه اصلی ذخیره سازی دیسک است، سایر رسانه‌های ذخیره سازی مانند نوار مغناطیسی در محیط پایگاه‌داده‌ها کاربرد دارد ولی نه به عنوان رسانه اصلی، بلکه به صورت رسانه کمکی برای تولید نسخه‌های پشتیبان و فایل‌های ثبت تراکنش‌ها یا فایل‌ها رویدادنگاری.

۲- سخت‌افزار پردازشگر

منظور خود کامپیوتر است. لازم به ذکر است که برای پایگاه‌های داده با معماری خاص و یا بسیار حجیم، از انواع خاصی از کامپیوترها با سخت‌افزارهای خاص و نوع پردازش خاص استفاده می‌گردد.

۳- سخت‌افزار برقرار کننده ارتباط

منظور از سخت‌افزار برقرار کننده ارتباط، سخت‌افزار ارتباطی بین کامپیوتر و دستگاه‌های جانبی و نیز بین کامپیوترهاست. این امکانات به دو دسته تقسیم می‌شوند:

- امکانات محلی: برای ایجاد ارتباط بین کامپیوتر و دستگاه‌های جانبی آن در یک سایت به کار می‌رود.
- امکانات شبکه‌ای: در ایجاد سیستم پایگاه‌داده‌های با معماری نامتمرکز به کار می‌رود.

چنانچه بخواهیم پایگاه‌های داده را بر اساس نگاه ارتباطی دسته بندی کنیم، انواع معماری پایگاه‌داده‌ها به شرح زیر خواهد بود:

- معماری مشتری-خدمت گزار^۱
- معماری متمرکز^۲
- معماری توزیع شده^۳
- معماری با پردازش موازی^۴
- معماری چند پایگاهی^۵
- معماری موبایل^۶

۲-۲-۳ نرم‌افزار

بین داده‌هایی که به صورت فیزیکی روی دستگاه‌های ذخیره سازی مناسب استقرار می‌یابد و پایگاه‌داده‌ها را به وجود می‌آورند و استفاده کنندگان یک لایه نرم‌افزاری قرار می‌گیرد که آن را سیستم مدیریت پایگاه‌داده (DBMS) می‌نامند. تمام تقاضا برای استفاده از اطلاعات پایگاه‌داده از طریق این سیستم سیر می‌شود و بازایی داده‌ها روی سخت‌افزارهایی صورت می‌گیرد. نرم‌افزارها خود به دو دسته تقسیم

1. Client-Server Architecture
2. Centralized Architecture
3. Distributed Architecture
4. Parallel Processing Architecture
5. Multi Database Architecture
6. Mobile Architecture

می‌شوند:

- نرم‌افزار کاربردی
- نرم‌افزار سیستمی

در ادامه به شرح هر یک از انواع نرم‌افزارها خواهیم پرداخت:

۱- نرم‌افزار کاربردی

نرم‌افزاری است که کاربر باید برای تماس با سیستم بانک اطلاعاتی آماده کند. این نرم‌افزار به کمک یک زبان سطح بالا و یک زبان داده‌یی^۱ و برخی تسهیلات نرم‌افزاری برای تماس با بانک ساخته می‌شود.

۲- نرم‌افزار سیستمی

این نوع نرم‌افزار از دو قسمت نرم‌افزار سیستمی خاص بانک که در اینجا به آن DBMS می‌گوییم و نرم‌افزار سیستمی عمومی (سیستم عامل) تشکیل شده است. (DBMS) در یک تعریف ساده، سیستمی است که به کاربران امکان می‌دهد عملیات مورد نظرشان را (مانند تعریف داده‌ها، بازیابی داده‌ها، ذخیره سازی داده‌ها) انجام دهند. DBMS که نرم‌افزاری پیچیده است میهمان یک سیستم عامل است و از امکانات سیستم عامل در انجام وظایفش استفاده می‌کند.

۲-۲-۴ کاربر

هر استفاده کننده از سیستم پایگاه داده‌ها را کاربر می‌گویند. کاربران پایگاه داده را می‌توان به سه گروه اساسی و متفاوت تقسیم نمود:

- برنامه نویسان کاربردی
- کاربران واقعی یا نهایی
- مدیر پایگاه داده‌ها

1. Data Language

۱- برنامه نویسان کاربردی^۱

افرادی هستند که با اطلاعاتی که در مورد پایگاه داده پیدا می‌کنند می‌توانند برنامه‌های مناسبی جهت بروز کردن اطلاعات و یا استفاده از اطلاعات موجود در پایگاه داده تهیه نمایند.

۲- کاربران واقعی یا نهایی^۲

افرادی هستند که با استفاده از امکاناتی که پایگاه داده در اختیار آنها قرار می‌دهد می‌توانند امور مربوط به خود و موسسه و سازمان را انجام دهند.

۳- مدیر پایگاه داده^۳

- مدیر پایگاه داده مسئولیت کنترل متمرکز سازمان بر داده‌های عملیاتی را بر عهده دارد. اگر بخواهیم وظایف DBA را به طور جزئی طراحی کنیم عبارتند از:
- تصمیم‌گیری در مورد داده‌هایی که در پایگاه داده نگهداری می‌شوند.
 - تصمیم‌گیری در مورد ذخیره سازی و روش دستیابی است.
 - ارتباط با کاربران برای حصول اطمینان از برآورده شدن نیازهای آنان.
 - تعریف بررسی‌های مربوط به امنیت^۴ و جامعیت^۵ (صحت عملیات داده‌ها).
 - تعریف استراتژی اخذ نسخه پشتیبانی و ترمیم^۶.
 - نظارت بر عملکرد سیستم و پاسخگویی به نیازهای در حال تغییر.

لازم به ذکر است که امروزه با پیچیده تر شدن پایگاه‌های داده و حجیم تر شدن داده‌ها، معمولاً چند مدیر پایگاه داده (Junior DBA) تحت نظر یک مدیر اصلی پایگاه داده‌ها (Senior DBA) قرار گرفته و هر یک مسئولیت یک قسمت را بر عهده دارند.

1. Application Programmer

2. End User (Real User)

3. Database Administrator

4. Security

5. Integrity

6. Recovery

تمرینات

۱. مفاهیم داده، اطلاع و دانش را توضیح دهید ؟
۲. پایگاه داده را تعریف نمایید ؟
۳. عناصر اساسی یک پایگاه داده را نام ببرید ؟
۴. انواع تجهیزات سخت افزاری مورد نیاز برای یک پایگاه داده را در حالت کلی نام برده و توضیح دهید ؟
۵. انواع کاربران استفاده کننده از یک سیستم پایگاه داده را در حالت کلی نام برده و توضیح دهید ؟

فصل ۲

مدل سازی معنایی داده‌ها

هدف کلی

در این فصل با مفهوم مدل سازی معنایی داده‌ها آشنا خواهیم شد و دو روش مدل سازی موجودیت-ارتباط (ER)^۱ و موجودیت-ارتباط توسعه یافته (ERR)^۲ را مورد بحث و بررسی قرار خواهیم داد. در بحث مدل سازی به روش ER، با مفاهیم موجودیت، صفات و ارتباط آشنا شده و سپس نحوه رسم نمودار را در این روش مورد بررسی قرار خواهیم داد و پس از آن مشکلات روش ER را شرح خواهیم داد. در ادامه مدل سازی به روش ERR را توضیح داده و مطالبی در خصوص نحوه پوشش مباحث شی گرای در این روش را برخواهیم شمرد.

هدف رفتاری

در این فصل عناوین زیر مورد بحث و بررسی قرار می گیرند:

- مدل سازی معنایی داده‌ها
- مدل سازی به روش ER
- نوع موجودیت
- نمونه موجودیت
- حالات یک موجودیت

1. Entity Relationship (ER)

2. Extended (Enhanced) Entity Relationship (EER)

- صفات یک موجودیت
- ارتباط
- نوع ارتباط
- نمودار ER
- درجه و ماهیت نوع ارتباط
- حد کاردینالیته
- مشکلات روش ER
- مدل سازی با روش EER
- تجزیه و ترکیب
- تخصیص و تعمیم
- زیر نوع‌های همپوشا و مجزا
- دسته بندی و وراثت
- تجمع

۱- مدل سازی معنایی داده‌ها^۱

کاربران پایگاه داده به طور معمول با داده‌های ذخیره شده در پایگاه داده سر و کار دارند که اصطلاحاً به آنها داده‌های عملیاتی می‌گویند. یکی از نکات مهم در ذخیره سازی داده‌ها مدل سازی معنایی آنها می‌باشد. یعنی داده‌های ذخیره شدنی در پایگاه داده‌ها ابتدا باید در بالاترین سطح انتزاع مدل سازی معنایی شوند. حال ممکن است این سؤال به ذهن برسد که مدل سازی معنایی به چه معنی است.

مدل سازی معنایی داده‌ها عبارت است از ارائه مدلی از محیط عملیاتی^۲ به کمک مفاهیمی مستقل از موضوعات مربوط به نمایش منطقی و فیزیکی داده‌ها. مدل سازی معنایی را در بعضی از کتب طراحی ادراکی^۳ نیز می‌نامند. برای مدل سازی معنایی روش‌های مختلف وجود دارد. روش‌های کلاسیک رایج عبارتند از:

1. Semantic Data Modeling
2. Operational Environment
3. Conceptual Design

- روش موجودیت-ارتباط
- روش موجودیت-ارتباط توسعه یافته

در این بین روش موجودیت-ارتباط (ER) که از ابتدا به عنوان روش مدل سازی معنایی در پایگاه‌های داده رابطه‌ای مورد استفاده قرار گرفته است بیشتر مورد بررسی قرار خواهد گرفت.

۲- مدلسازی به روش ER

مدل سازی به روش ER یکی از ابزارهای مدل سازی معنایی در پایگاه‌داده است که در سال ۱۹۷۶ توسط آقای Chen در MIT ارائه گردید و به مرور زمان این ابزار پیشرفت کرد. تعریف Chen از بانک اطلاعات عبارت بود از تعدادی پدیده^۱ (موجودیت) دارای صفات^۲ مشخص و ارتباط^۳ بین پدیده‌ها. در این روش، سه مفهوم معنایی^۴ زیر وجود دارد:

- نوع موجودیت^۵
- صفت^۶
- نوع ارتباط^۷

در ادامه هر یک از مفاهیم مذکور را مورد بحث و بررسی قرار خواهیم داد.

۲-۱ نوع موجودیت

نوع موجودیت عبارت است از مفهوم کلی هر آنچه که می‌خواهیم در مورد آن اطلاعی جمع‌آوری کنیم و دانش خود را در موردش افزایش دهیم. گاه به نوع موجودیت، نوع شیئی هم می‌گوییم. لازم بذکر است که تشخیص انواع موجودیت‌ها

1. Entity
2. Attribute
3. Relation
4. Semantic Concept
5. Entity Type
6. Attribute
7. Relationship Type

در یک محیط کاری دشوار می‌باشد. به طور کلی یک نوع موجودیت دارای خصوصیات زیر می‌باشد یا به بیانی دیگر لازم است اطلاعات زیر در مورد هر نوع موجودیتی بدست آید:

- نام موجودیت (یا پدیده)
- معنای مشخص
- مجموعه‌ای از صفات
- مجموعه‌ای از نمونه‌ها
- حالت کنش‌گری یا کنش‌پذیری
- عدم وابستگی و یا وابستگی به یک نوع دیگر

برای مثال در یک سیستم اطلاعات اتومبیل انواع موجودیت‌ها می‌توانند شامل موجودیت اتومبیل، کشور سازنده (یا مصرف کننده) و... باشند. در یک سیستم آموزشی انواع موجودیت‌ها می‌توانند شامل موجودیت‌های درس، دانشجو، استاد، کلاس و... باشند.

۲-۱-۱ نمونه موجودیت^۱

تمام نمونه‌های مشخص (در مواردی متمایز) هر نوع موجودیت از یک محیط مشخص، مجموعه‌ای به نام مجموعه نمونه‌های^۲ آن موجودیت را تشکیل می‌دهند. هر نوع موجودیت خود می‌تواند دارای نمونه‌های مختلفی باشد که این نمونه‌ها از مشخصات نوع خود تبعیت می‌کنند. برای مثال موجودیت اتومبیل می‌تواند دارای نمونه‌هایی مانند پیکان، پژو، سمند، بنز و... باشد که هر کدام از اینها در واقع یک نمونه از موجودیت اتومبیل هستند.

۲-۱-۲ حالات یک موجودیت

یکی از نکات بسیار مهم در تعیین موجودیت‌ها تعیین مستقل یا وابسته بودن موجودیت‌ها می‌باشد. یک موجودیت ممکن است به دو صورت قوی (مستقل) یا

1. Entity Instance

2. Instances Set

ضعیف (وابسته) باشد. در ادامه به شرح حالات مذکور خواهیم پرداخت:

موجودیت قوی یا مستقل^۱

موجودیتی است که مستقل از هر نوع موجودیت دیگر و به خودی خود در یک محیط مشخص مطرح باشد. این نوع موجودیت وابستگی خاصی به سایر موجودیت‌های محیط عملیاتی ندارد.

موجودیت ضعیف یا وابسته^۲

موجودیت ضعیف موجودیتی است که وجودش وابسته به یک نوع موجودیت دیگر (موجودیت قوی) است. لازم بذکر است که اگر موجودیت قوی از مدل معنایی حذف گردد، وجود موجودیت ضعیف بی معنا بوده و موجودیت ضعیف نیز حذف می‌گردد.

نکته: مستقل بودن یا وابسته بودن موجودیت‌ها در محیط عملیاتی که می‌خواهیم برای آن پایگاه داده‌ای طراحی کنیم، تعیین می‌گردد و این موضوع ارتباطی به وابستگی و یا استقلال موجودیت در دنیای واقعی ندارد.

۲-۲ صفت

هر نوع موجودیت شامل مجموعه‌ای از صفات (مشخصات) مربوط به آن موجودیت است که حالت یا وضع آن موجودیت را توصیف می‌کند. این صفات خود دارای رده‌بندی‌های مختلفی هستند که در ذیل آمده است:

رده بندی صفت

صفات یک موجودیت بر حسب مفهوم آنها به دسته‌های زیر تقسیم می‌گردند:

- ساده^۳ یا مرکب^۴
- تک مقداری^۱ یا چند مقداری^۲

1. Strong
2. Weak
3. Single
4. Composite

- شناسه یا ناشناسه
- هیچ مقدار پذیر^۲ یا هیچ مقدار ناپذیر
- ذخیره شده (واقعی^۴ یا مبنا) یا مشتق^۵

در ادامه هر یک از رده بندی صفات (بر اساس آنچه که در بالا آمده است) مورد بحث و بررسی قرار می‌گیرند.

۲-۲-۱ صفت ساده یا مرکب

صفت ساده

صفتی است که مقدار آن تجزیه نشدنی می‌باشد، به این معنا که اگر مقدار آنرا به اجزائی تجزیه کنیم، مقادیر هر جزء فاقد معنا می‌باشد. مانند اسم درخت.

صفت مرکب

صفتی که از چند صفت ساده تشکیل شده و تجزیه شدنی می‌باشد. مانند صفت آدرس که می‌تواند شامل نام کشور، استان، شهر، منطقه و... باشد. لازم به ذکر است که می‌توان بر حسب نوع نیاز بجای استفاده از یک صفت مرکب، صفت مذکور را به چندین صفت ساده شکست.

۲-۲-۲ صفت تک مقداری یا چند مقداری

صفت تک مقداری

صفت تک مقداری صفتی است که حداکثر یک مقدار از میدان مقادیر را برای یک نمونه از یک نوع موجودیت می‌گیرد. به بیانی دیگر مقدار آن صفت برای نوع موجودیت مورد نظر، یک مقدار مشخص از میدان مقادیر مربوط به آن صفت می‌باشد. مثلاً برای هر شخص یک کد ملی وجود دارد.

1. Single Valued

2. Multi Valued

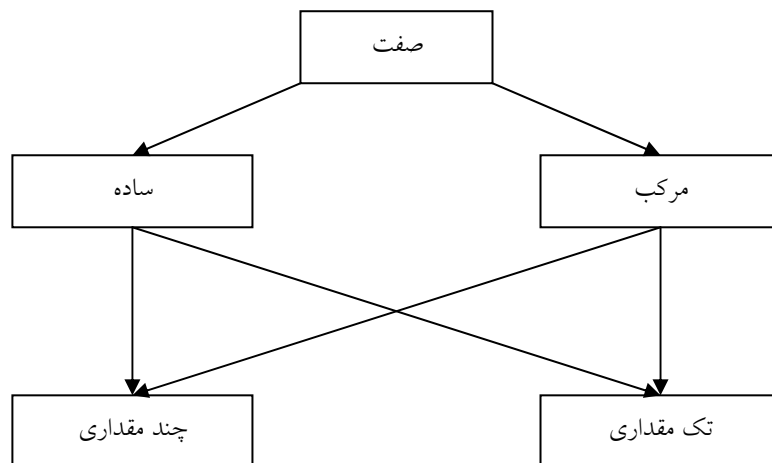
3. Null Value

4. Real

5. Derived

صفت چند مقداری

صفتی است که بیش از یک مقدار از میدان مقادیر را برای حداقل یک نمونه از نوع موجودیت در بر می‌گیرد. مانند صفت مدرک تحصیلی برای یک شخص که ممکن است چند مقداری باشد، لیسانس، فوق لیسانس و....



شکل ۱-۲ نمودار ارتباط دهنده رده‌های مختلف صفات

۳-۲-۲ شناسه

صفت شناسه موجودیت، صفتی است که باید یکتایی مقدار داشته باشد و حتی الامکان طول مقادیرش کوتاه باشد.

مفهوم مقدار هیچ (هیچ مقدار)

این مفهوم از مفاهیم مدل رابطه‌ای است، مقدار هیچ یعنی مقدار ناشناخته، مقدار تعریف نشده. ممکن است مقدار یک صفت برای برخی از نمونه‌های یک نوع موجودیت، ناشناخته باشد. لازم به ذکر است که صفت شناسه موجودیت نمی‌تواند هیچ مقدار پذیر باشد.

۲-۲-۴ صفت واقعی یا مشتق

صفت واقعی (ذخیره شده)

صفت واقعی آن صفتی است که مقادیرش در پایگاه داده‌ها ذخیره شده باشد. باید توجه داشت که چنانچه صفت بعنوان شناسه نباشد، می‌تواند مقدار هیچ را نیز داشته باشد.

صفت مشتق

صفتی است که مقادیرش در پایگاه داده‌ها ذخیره شده نباشد. این صفت وجود خارجی ندارد. ولی از روی دیگر صفات قابل محاسبه است. مانند سن افراد که از روی تاریخ تولد قابل محاسبه است. هر صفت جنبه‌های زیر را دارد:

- نام
- معنا
- میدان (دامنه) مقادیر
- نوع مقدار
- طول مقدار (صریح یا صفتی)
- یک یا چند محدودیت ناظر به صفت

۲-۳ ارتباط^۱

یکی از مفاهیم بسیار مهم در مدل سازی معنایی داده‌ها مفهوم ارتباط یا بستگی است، به همین منظور به تعریف نوع ارتباط می‌پردازیم:

نوع ارتباط

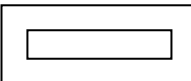
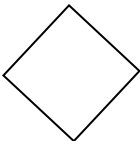
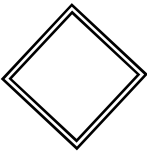
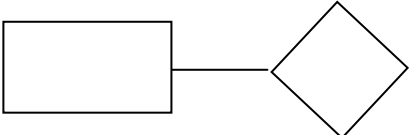
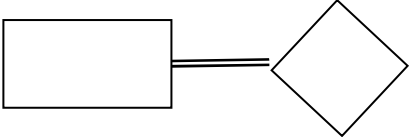
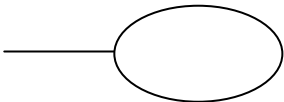
نوع ارتباط عبارت است از تعامل^۲ بین n نوع موجودیت ($n \geq 1$) و ماهیتا نوعی بستگی بین انواع موجودیت‌هاست. به تعبیری دیگر عملی است که بین انواع موجودیت‌ها جاری بوده، هست یا خواهد بود. هر نوع ارتباط یک معنای مشخص داشته و با یک نام بیان می‌شود.

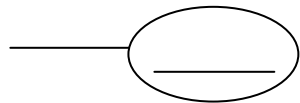
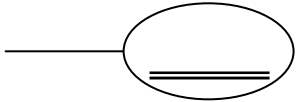
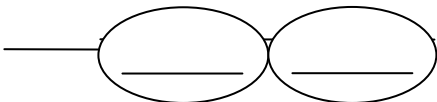
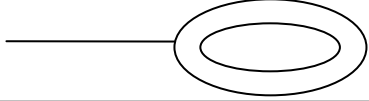
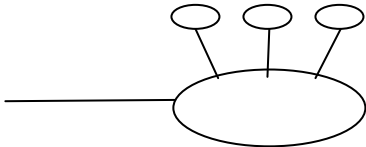
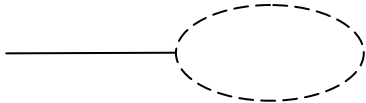
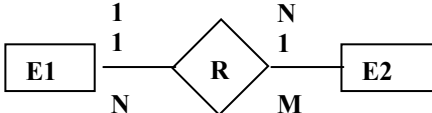
1. Association

2. Interaction

۳- نمودار ER

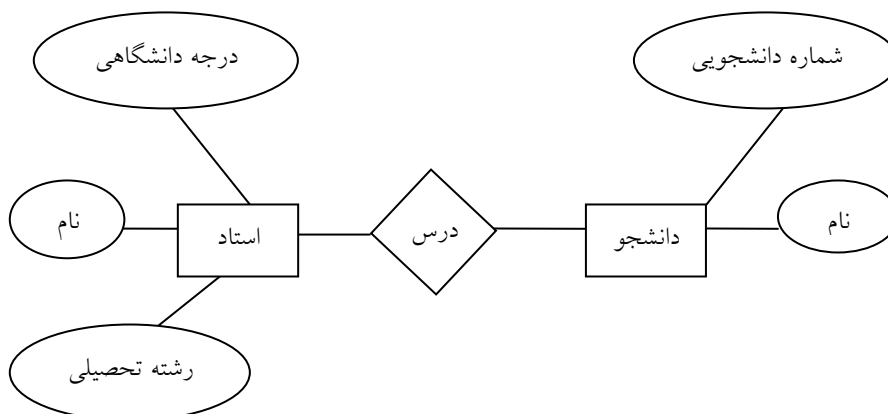
نمودار ER در واقع نموداری است که در آن سه مفهوم اساسی مدل ER یعنی موجودیت، صفت و نوع ارتباط نمایش داده می‌شوند. در واقع نمودار ER اولین طرح پایگاه داده‌ها و مدل کلی آن در بالاترین سطح انتزاع می‌باشد.

نماد در مدل ER	مفهوم نماد در مدل ER
	نوع موجودیت: پدیده‌های موجود که وجود خارجی دارند (Entity) را با مستطیل نمایش می‌دهند.
	نوع موجودیت ضعیف (وابسته)
	نوع ارتباط (Relationship): عامل ارتباط موجودیت‌ها را با لوزی نمایش می‌دهند.
	نوع ارتباط موجودیت ضعیف با قوی
	مشارکت نوع موجودیت در نوع ارتباط
	مشارکت الزامی
	صفت: صفتهای هر موجودیت را توسط اشکال بیضی به آن متصل می‌نماییم.

	صفت شناسه اول
	صفت شناسه دوم (در صورت وجود)
	صفت شناسه مرکب (مثلا درختی)
	صفت چند مقداری
	صفت مرکب
	صفت مشتق (مجازی یا محاسبه شدنی)
	چندی ارتباط

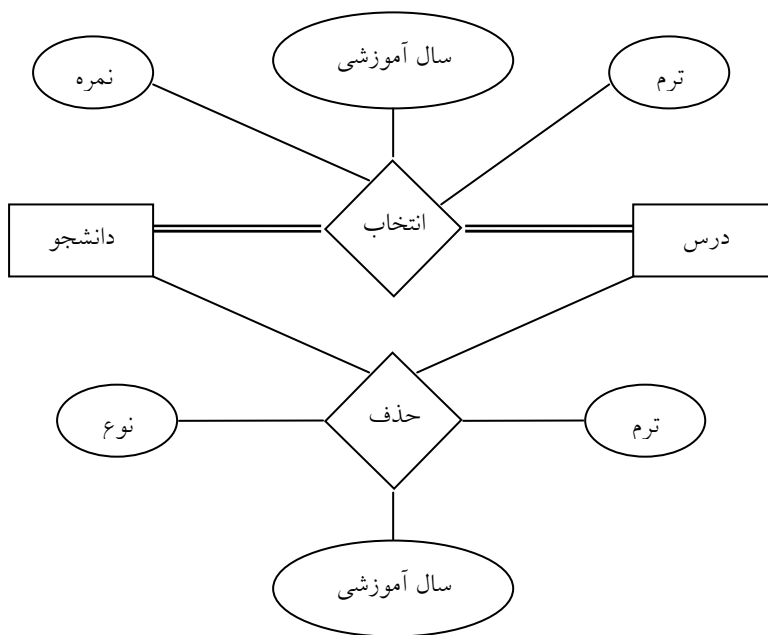
شکل ۲-۲ نمادهای رسم نمودار ER

هر نمودار ER پاسخگوی مجموعه مشخصی از نیازهای کاربران است و بدیهی است که این نمودار با تغییر و رشد نیازهای کاربران تغییر یافته و توسعه داده می‌شود. برای رسم این نوع نمودار نیاز به نمادهایی است که این نمادها در جدول ذیل آمده است. لازم به ذکر است که تمامی نمادهای مورد استفاده برای رسم نمودار ER در رسم نمودار EER نیز معتبر بوده و مورد استفاده قرار می‌گیرند. برای درک بهتر نحوه استفاده از نمادها در رسم نمودار ER، به نمودار زیر که بیانگر وضعیت یک سیستم دانشجویی است توجه نمایید:



شکل ۳-۲ یک نمونه از نمودار ER برای نمایش سیستم دانشجویی

یکی از نکات مهم در رسم نمودار ER، معنای ارتباط بین عناصر است. بدین صورت که معنای ارتباط ((انتخاب)) با معنای ارتباط ((حذف)) فرق دارد. در ادامه نمونه‌ای از معنای ارتباط بین موجودیت‌ها ترسیم شده است.



شکل ۴-۲ نمونه‌ای از معنای ارتباط بین عناصر

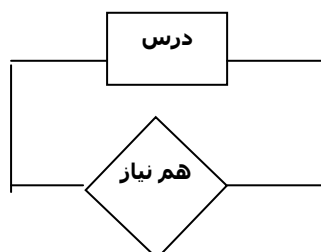
۳-۱ درجه نوع ارتباط

درجه ارتباط بستگی به تعداد موجودیتها مرتبط به هم دارد. به بیانی دیگر تعداد شرکت کنندگان در یک نوع ارتباط را درجه آن ارتباط می‌گویند.

درجه		تعداد موجودیت‌های شرکت کننده در ارتباط
فارسی	لاتین	
یگانی	Unary	۱
دوگانی	Binary	۲
سه گانی	Ternary	۳
...	...	...
چندگانی	n-ary	N

در ادامه هر یک از درجات نوع ارتباط بصورت شماتیک رسم شده و توضیح داده می‌شوند:

یگانی^۱ زمانی که یک نوع ارتباط بین یک نوع موجودیت و خودش برقرار باشد.

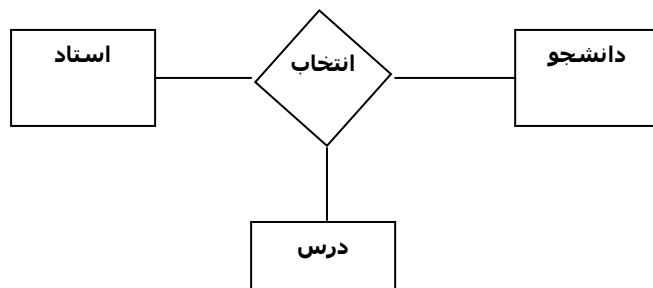


دوگانی^۲ ارتباط بین دو موجودیت

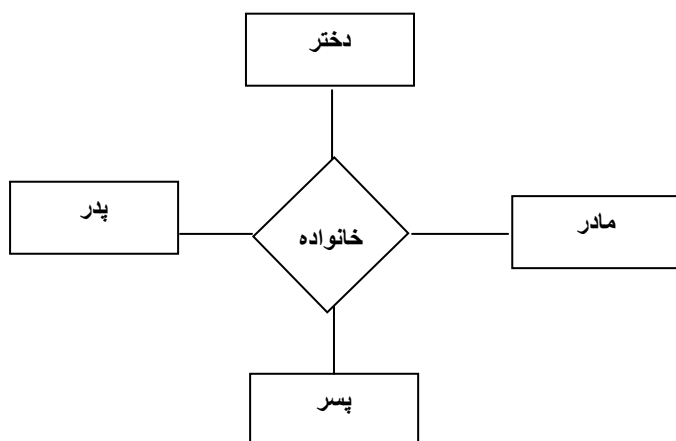


1. Unary
2. Binary

سه گانی^۱: ارتباط بین سه موجودیت



چند گانی: ارتباط بین چند موجودیت



توجه: چون معمولاً مادر و پدر و خواهر و برادر یک عضو محسوب می‌شوند، این نوع ارتباط منطقاً درست نیست. شکل مذکور فقط بعنوان نمونه است.

۲-۳ ماهیت نوع ارتباط (اتصال)

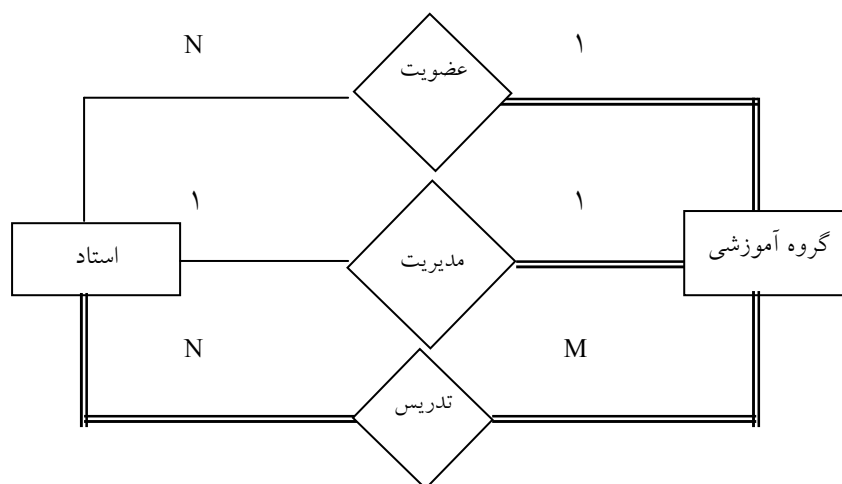
اگر دو موجودیت A و B را در نظر بگیریم این دو موجودیت به یکی از سه حالت زیر با هم ارتباط دارند.

- یک به یک^۲: یک نمونه از موجودیت A حداکثر با یک نمونه از موجودیت B ارتباط دارد و بر عکس. به اختصار 1:1 نمایش می‌دهند.

1. Ternary

2. One To One

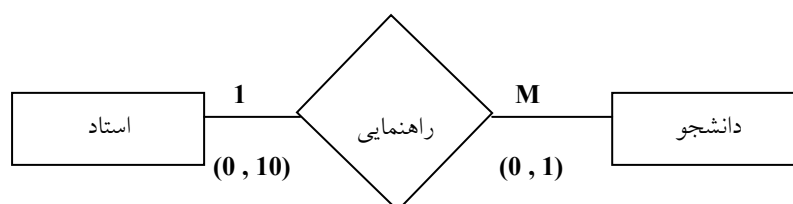
- یک به چند^۱: یک نمونه از A با n نمونه از B ارتباط دارد ولی یک نمونه از B حداکثر با یک نمونه از A ارتباط دارد ($n > 1$). به اختصار 1: N نمایش می‌دهند.
- چند به چند^۲: یک نمونه از A با n نمونه از B ارتباط دارد و بر عکس ($n > 1$). به اختصار M: N نمایش می‌دهند.



شکل ۵-۲ نمایش چندی ارتباط

۳-۳ حد کاردینالیتی^۳

حد کاردینالیتی، حداقل و حداکثر ارتباط بین دو موجودیت را می‌رساند. برای درک بهتر این مفهوم مثال زیر ارائه می‌گردد:



شکل ۶-۲ نمونه‌ای از نمایش حد کاردینالیتی

1. One To Many
2. Many To Many
3. Cardinality

عبارات ذکر شده در بالا و پایین ارتباط بین عناصر، هر یک دارای مفاهیمی هستند. در ادامه دو عبارت (0, 10) و (0, 1) مورد بررسی قرار گرفته‌اند:

- (۰۱۰) بدین معنی است که یک استاد ممکن است حداکثر ۱۰ دانشجو را راهنمایی کند و یا هیچ دانشجویی برای راهنمایی نداشته باشد (این استاد پروژه ارائه نکرده است).
- (۰۱۰) بدین معنی است که یک دانشجو حداکثر می‌تواند یک استاد راهنما داشته باشد و اگر پروژه اخذ نکرده باشد هیچ استاد راهنمایی ندارد.

۴- مشکلات روش ER

در نتیجه درک نادرست و تفسیر ناصحیح از معنای بعضی ارتباطات در مدل سازی داده‌ها مشکلاتی موسوم به دامهای پیوندی^۱ نمایان می‌گردد. این دامهای پیوندی عبارتند از:

- دام حلقه‌ای^۲
- دام چند شاخه (چتری)^۳
- دام گسل (شکاف)^۴

در ادامه برای درک بهتر مفهوم دام‌ها، هر یک از انواع دام‌ها در روش ER با ذکر مثال شرح داده خواهند شد.

۴-۱ دام حلقه‌ای

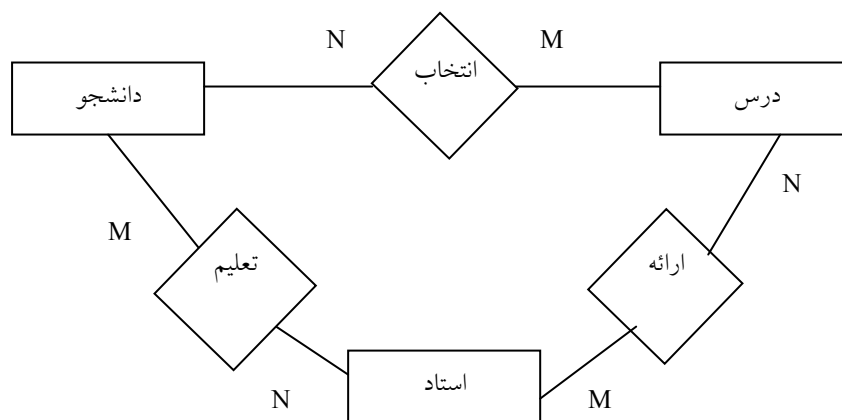
این دام وقتی ایجاد می‌شود که با داشتن مثلاً سه ارتباط دو موجودیتی، وجود یک ارتباط سه موجودیتی را نتیجه‌گیری کنیم، ولی این استنتاج درست نباشد.

1. Connection Traps

2. Loop Trap

3. Fan Trap

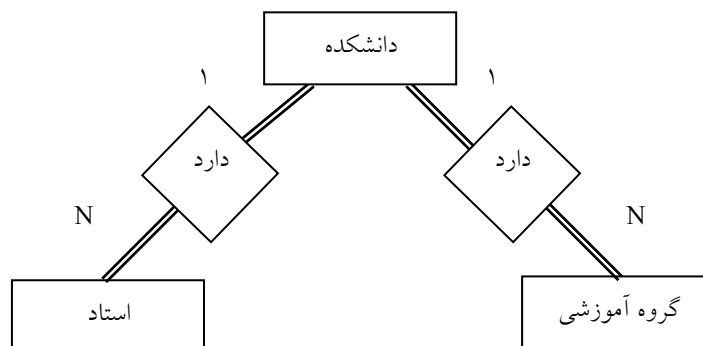
4. Chasm Trap



شکل ۷-۲ نمونه‌ای از دام حلقه‌ای

۲-۴ دام چند شاخه‌ای

این نوع دام وقتی ایجاد می‌شود که بین یک نوع موجودیت E و موجودیت‌های F و G ارتباط 1:N با مشارکت الزامی وجود داشته باشد، ولی ارتباط بین F و G دیده نشده باشد. در این صورت نمی‌توان وجود ارتباط بین F و G را بدست آورد.



شکل ۸-۲ نمونه‌ای از دام شاخه‌ای

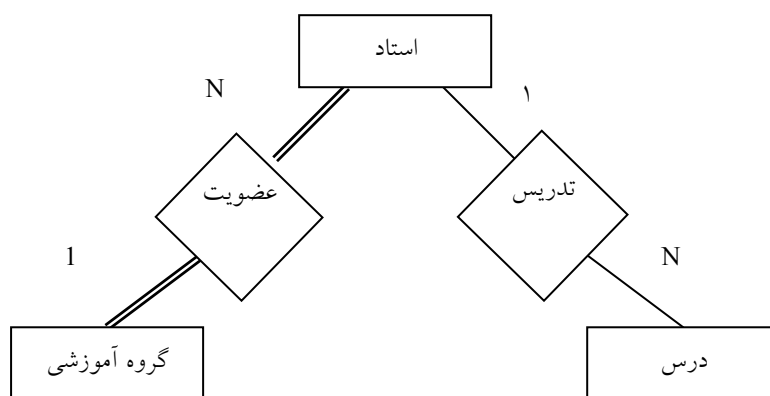
۳-۴ دام گسل

این نوع دام وقتی ایجاد می‌شود که بین دو نوع موجودیت E و F یک ارتباط 1:N و مشارکت الزامی وجود داشته باشد، ولی F با نوع موجودیت G ارتباط 1:N با مشارکت

غیر الزامی داشته باشد. در این شرایط نمی توان تمام اطلاع‌های دو موجودیتی بین E و G را بدست آورد. اگر چنین فرضی در نظر گرفته شود، دچار دام گسل شده‌ایم.

۵- مدل سازی با روش EER

همانطور که می‌دانیم روش ER دارای نقاط ضعفی بود. این نقاط ضعف بیشتر زمانی نمایان می‌شد که می‌خواستیم یک سیستم شیء گرا را مدل سازی نماییم. بطور کلی نقاط ضعف روش ER که در روش EER رفع شده‌اند به شرح ذیل بودند.



شکل ۲-۹ نمونه‌ای از دام گسل

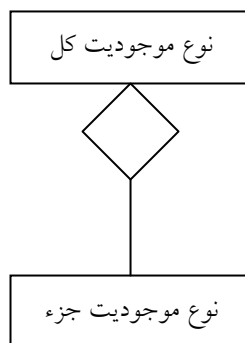
۵-۱ تجزیه^۱ و ترکیب^۲

فرایند تقسیم یک شیء کل به اجزاء تشکیل دهنده آن را تجزیه گویند. بدیهی است که در فرایند تجزیه، شیء کل و اجزاء آن هر یک دارای صفات، ساختار و رفتار خاص خود می‌باشند.

ترکیب، عکس عمل تجزیه است که در آن با داشتن تعدادی نوع موجودیت، یک نوع موجودیت جدید را ایجاد می‌کنیم.

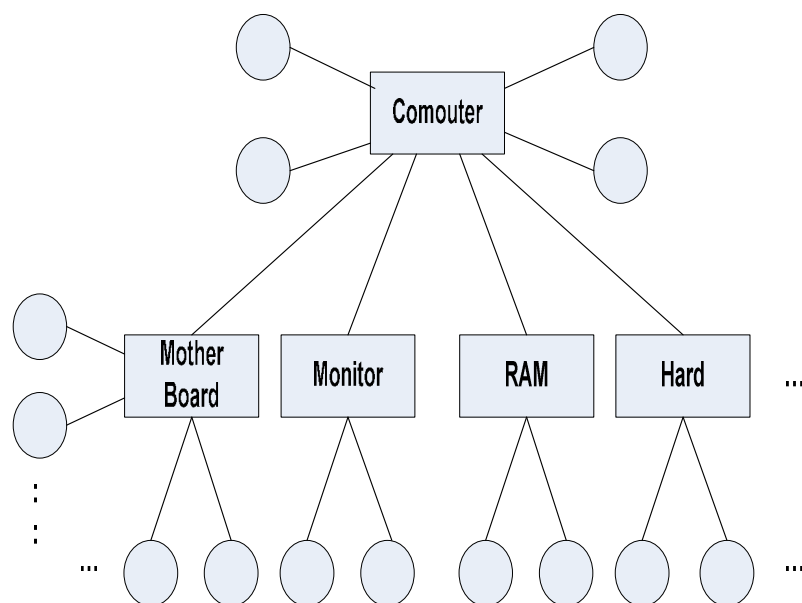
1. Decomposition

2. Composition



شکل ۲-۱۰ نماد ارتباط "جزئی است از..."

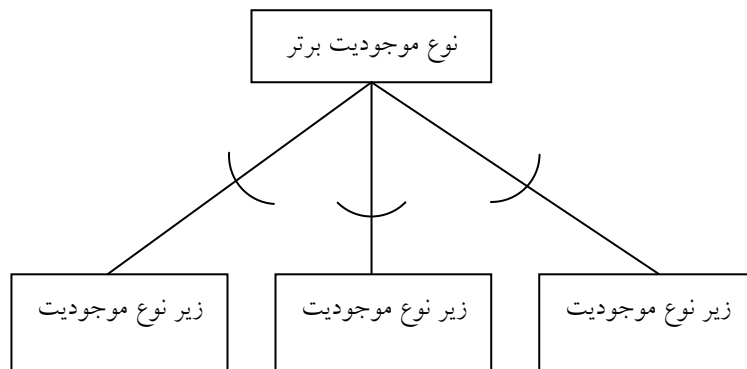
در روش ERR ارتباط بین شیء کل و اشیاء جزء را ارتباط "جزئی است از..." گویند. نماد مورد استفاده برای نمایش ارتباط نوع موجودیت کل و نوع موجودیت جزء به شکل زیر می‌باشد:



شکل ۲-۱۱ نمونه‌ای از تجزیه و ترکیب موجودیت‌ها

۲-۵ تخصیص^۱ و تعمیم^۲

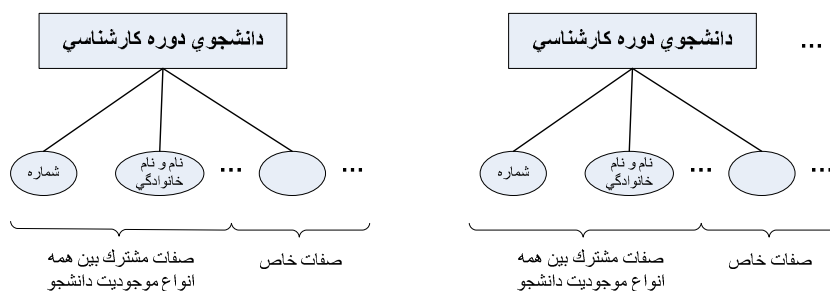
تخصیص فرایندی است که طی آن نمونه‌های یک نوع موجودیت برتر^۳ (زیر نوع) را بر اساس یک یا چند صفت خاصه آن موجودیت برتر تشخیص می‌دهیم. لازم به ذکر است که یک نوع موجودیت می‌تواند دارای یک یا چند زیر نوع موجودیت^۴ نیز باشد. ارتباط بین موجودیت برتر و زیر نوع‌های آن را ارتباط "**گونه‌ای است** / **از...**" می‌نامیم. نماد مورد استفاده برای نمایش این نوع ارتباط در شکل زیر آمده است.



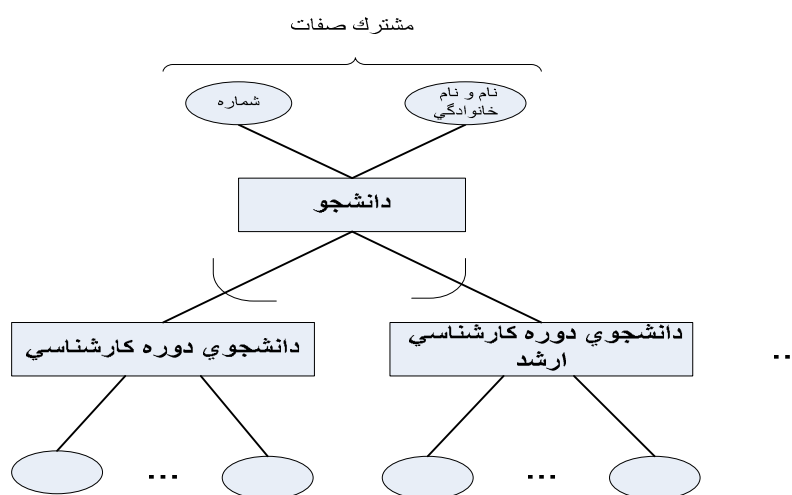
شکل ۲-۱۲ نماد ارتباط "**گونه‌ای است از...**" (تخصیص)

تعمیم عکس عمل تخصیص است که در آن با داشتن زیر نوع‌های یک نوع موجودیت و تعیین صفات مشترک بین آنها، یک مجموعه صفات را برای نوع موجودیت برتر در نظر می‌گیریم.

-
1. Specialization
 2. Generalization
 3. Super Entity Type
 4. Sub Entity Type



شکل ۲-۱۳ (الف) نمونه‌ای از حالت تعمیم در سیستم‌های دانشجویی

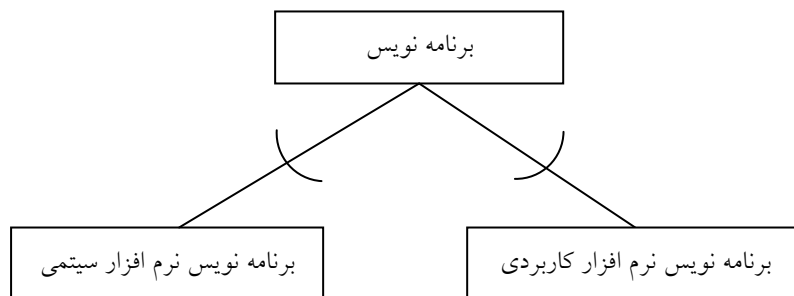


شکل ۲-۱۳ (ب) نمونه‌ای از حالت تعمیم در سیستم‌های دانشجویی

۳-۵ زیر نوع‌های همپوشا^۱ و مجزا^۲

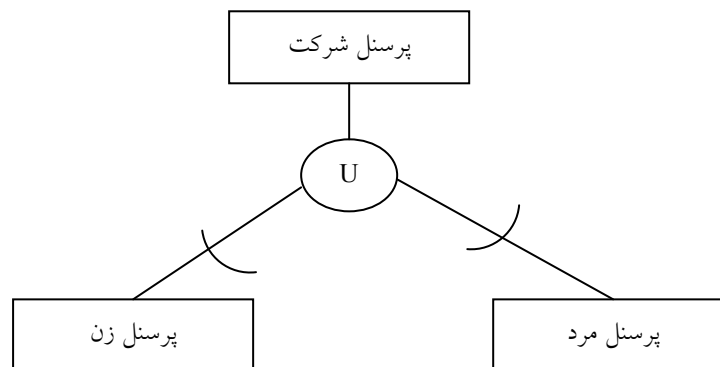
نمونه‌های یک نوع موجودیت برتر بر اساس صفات خاصه مشترک و مجزا در دسته‌های خاص خود دسته بندی می‌شوند. حال ممکن است یک نمونه موجودیت در دو دسته قابل دسته بندی باشد در چنین شرایطی به این نوع موجودیت‌ها، موجودیت‌های همپوشا (مشترک) گویند.

-
1. Overlap
 2. Disjoint



شکل ۲-۱۴ نمونه‌ای از زیر نوع‌های همپوشا

در عین حال ممکن است یک نوع موجودیت فقط در یک دسته بندی از زیر نوع موجودیت‌ها قابل دسته بندی باشد. به این زیر نوع موجودیت‌ها، موجودیت‌های مجزا می‌گویند.



شکل ۲-۱۵ نمونه‌ای از زیر نوع‌های مجزا

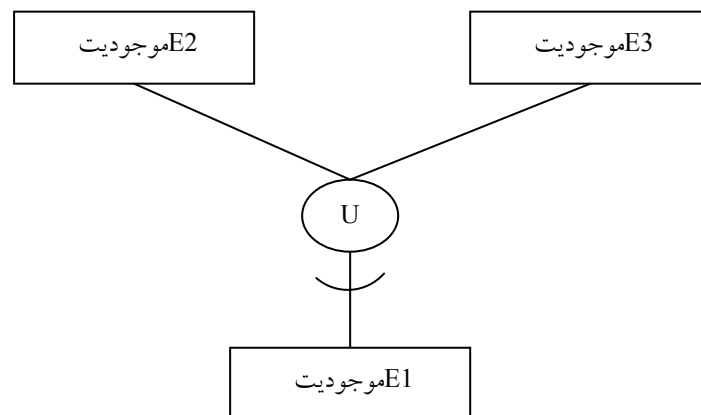
۵-۴ دسته بندی^۱ و وراثت^۲

یک زیر نوع موجودیت می‌تواند زیر نوع بیش از یک نوع موجودیت برتر باشد. که در این شرایط بعضی از خواص خود را از یک نوع موجودیت برتر و بعضی دیگر

1. Categorization

2. Inheritance

از صفات خاصه خود را از یک نوع موجودیت برتر دیگر به ارث می‌برد. این موجودیت‌های برتر می‌توانند از یک نوع باشند که در این شرایط دارای شناسه‌های یکسان هستند. ولی در شرایطی که موجودیت‌های برتر از یک نوع نباشند، در واقع وراثت چندگانه^۱ رخ داده است. به این زیر نوع‌ها در اصطلاح دسته^۲ (طبقه) می‌گویند و در بعضی از کتاب‌ها به این زیر نوع اصطلاحاً نوع اجتماع^۳ می‌گویند. شکل زیر نحوه نمایش وراثت چندگانه را نشان می‌دهد:

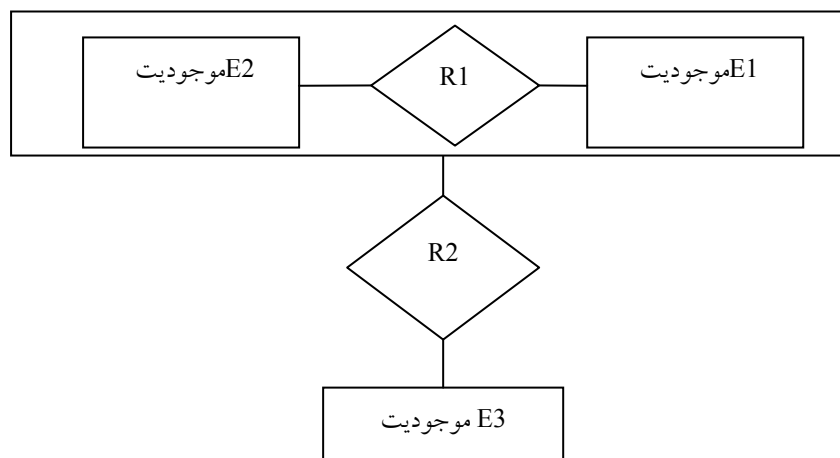


شکل ۱۶-۲ شمای نمایش وراثت چندگانه و دسته بندی

۵-۵-۵ تجمع^۴

تجمع بدین معنا است که یک نوع موجودیت جدید را بر اساس دو یا چند موجودیت مرتبط با یکدیگر، به صورت یکپارچه در یک نوع موجودیت واحد ارائه نماییم. بدیهی است که این نوع موجودیت واحد خود می‌تواند با نوع موجودیت‌های دیگر نیز در ارتباط باشد. شکل زیر نحوه نمایش تجمع را نشان می‌دهد:

-
1. Multiple Inheritance
 2. Category
 3. Union Type
 4. Aggregation



شکل ۲-۱۷ نمایش تجمع

۶-روش مدل سازی شیء UML^۱

با مطرح شدن الگوهای شیء گرایی که هر موجودیت در جهان واقع را شامل داده‌های آن موجودیت و عملکردهای مرتبط با آن توصیف می‌کرد، طراحان سیستم روش‌های ER و EER را ناکارآمد تشخیص داده و در جستجوی تعریف یک متودولوژی جدید بودند تا بتوانند عملکرد هر موجودیت را نیز به نمایش در آورند. از اینرو گروه OMG^۲ روشی استاندارد بنام UML را برای ایجاد سیستم‌های کاربردی معرفی کردند و این روش را در ایجاد سیستم‌های کاربردی مورد استفاده قرار دادند. مهمترین خصوصیت این روش، ایجاد امکانی برای ایجاد و نمایش اشیاء جهان واقع بصورت تصویری است. یکی از نکات بسیار مهم در استفاده موثر از روش UML، وجود دانش کافی در مفاهیم شیء گرایی است. به بیانی دیگر بدون دانستن مفاهیم شیء گرایی، امکان استفاده موثر و بجا از روش UML میسر نخواهد شد.

۶-۱ مفاهیم اصلی

اساس روش UML بر مبنای نمودار (دیاگرام) می‌باشد. این روش از نمودارهای

1. Object Modeling

2. Object Management Group

برای نمایش مدل سازی و طراحی نرم افزار استفاده می کند. چند نمونه از مهمترین نمودارها در این روش به شرح زیر می باشند:

- نمودار کلاس^۱
- نمودار چرخه حیات موجودیت^۲
- نمودار مورد استفاده^۳
- نمودار فعالیت^۴
- نمودار پیاده سازی^۵

مهمترین نمودار در این روش نمودار کلاس می باشد. این نمودار مجموعه ای از موجودیت ها و عملیات (پردازش) مرتبط با موجودیت ها را مدل سازی می کند. عملیات در واقع رفتار شیء^۶ را نشان می دهد و به بیانی دیگر رویداد^۷ های مرتبط با شیء را به نمایش در می آورد. مفاهیم اصلی در مدلسازی با این روش عبارتند از:

- کلاس
- صفت
- بستگی^۸

در روش UML دو گونه ارتباط بین رده ها وجود دارد که این دو نوع ارتباط عبارتند از

- بستگی
- تجميع

که در این بین مفهوم بستگی همان ارتباط بین کلاس ها است. تجميع نیز عبارت است از ارتباط بین یک شیء کل و شیء های جزء تشکیل دهنده آن. در ادامه برای درک بهتر تناظر بین مفاهیم در دو روش UML و EER جدول زیر ارائه می گردد:

-
1. Class Diagram
 2. Entity Life Cycle Diagram
 3. Use Case Diagram
 4. Activity Diagram
 5. Implementation Diagram
 6. Object Behavior
 7. Event
 8. Association

مفهوم در EER	مفهوم در UML
نوع موجودیت	کلاس
نمونه موجودیت	شیء
صفت	صفت
ارتباط	بستگی
نمونه ارتباط	پیوند ^۱
ارتباط بازگشتی	بستگی انعکاسی ^۲
نوع موجودیت ضعیف	بستگی مقید ^۳
صفت مرکب	میدان ساختمان ^۴
صفت ارتباط	صفت پیوند
درجه ارتباط	چندی بستگی ^۵

۶-۲ نحوه نمایش مفاهیم

کلاس: برای نمایش یک کلاس از مربع یا مستطیل استفاده می‌شود. در این حالت مربع به سه قسمت مجزا تقسیم شده و در این قسمت‌ها موارد زیر نوشته می‌شوند

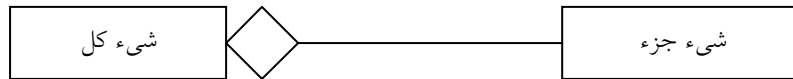
- نام کلاس
- نام صفات کلاس
- نام پردازشهای کلاس

نام کلاس
نام صفات کلاس
نام پردازشهای کلاس

بستگی: بستگی بین دو کلاس بصورت یک خط متصل کننده دو کلاس نمایش داده می‌شود و نام ارتباط روی خط نوشته می‌شود.

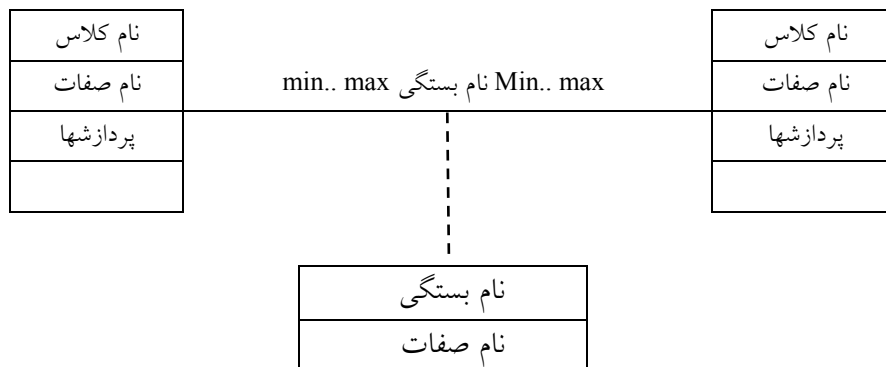
-
1. Link
 2. Reflexive Association
 3. Qualified Association
 4. Structural Domain
 5. Multi Association

تجمع: تجمع بصورت زیر نمایش داده می‌شود



چندی ارتباط: چندی ارتباط بصورت min ... max نوشته می‌شود
صفت چند مقداری: صفت چند مقداری به صورت یک کلاس جداگانه نشان داده می‌شود، ولی قسمت پردازش را همراه ندارد.
میدان: نام میدان معمولاً بعد از نام صفت نوشته می‌شود و بین این دو نام از علامت: استفاده می‌شود بصورت زیر
 نام میدان: نام صفت

صفت پیوند: صفت پیوند در یک مربع یا مستطیل نوشته می‌شود و با خط چین به خط نشان دهنده بستگی متصل می‌شود. نام پیوند و نام صفات پیوند در دو قسمت این مربع نوشته می‌شوند. شکل زیر نحوه نمایش صفت پیوند را نشان می‌دهد



۳-۶ خصوصیات کلی روش مدل سازی معنایی داده‌ها

هر روش مدل سازی معنایی داده‌ها باید حداقل دارای خصوصیات زیر باشد

- گویایی
- سادگی مفاهیم

- ایجاز
- گسترش پذیری
- صوری بودن
- قابلیت نمایش نموداری
- جامع بودن مفاهیم
- قابلیت نمایش ساختار حالت و رفتار نوع موجودیت

باید به این نکته توجه داشت که بعضی از خصوصیات ذکر شده مانند تجمیع و ایجاز ممکن است با یکدیگر مغایرت داشته باشند. در این صورت وجود یکی از این خصوصیات کافی است.

تمرینات

۱. مدل سازی معنایی داده را توضیح دهید ؟
۲. انواع موجودیت‌ها را نام برده و توضیح دهید ؟
۳. صفت را تعریف کرده و رده‌های مختلف آن را نام ببرید ؟
۴. حالات مختلف ارتباط بین دو موجودیت را نام برده و توضیح دهید ؟
۵. مشکل دام حلقه‌ای در روش ER را توضیح دهید ؟
۶. مشکل دام چند شاخه‌ای را در روش ER توضیح دهید ؟
۷. مشکل دام گسل را در روش ER شرح دهید ؟
۸. مفاهیم تجزیه و ترکیب در روش EER را شرح دهید ؟
۹. مفاهیم تخصیص و ترمیم در روش EER را شرح دهید ؟
۱۰. مفاهیم وراثت و دسته بندی در روش EER را شرح دهید ؟
۱۱. مفهوم تجمع در روش EER را شرح دهید ؟

فصل ۳

معماری پایگاه داده

هدف کلی

در این فصل معماری سه سطحی پایگاه داده مورد بحث و بررسی قرار خواهد گرفت. در این راستا سطوح خارجی، ادراکی و داخلی بصورتی دقیق تر مورد ارزیابی قرار خواهند گرفت. در ادامه با سایر اجزاء پایگاه داده آشنا خواهیم شد و زبان های مختلف مورد استفاده درون پایگاه داده نیز بررسی خواهند شد.

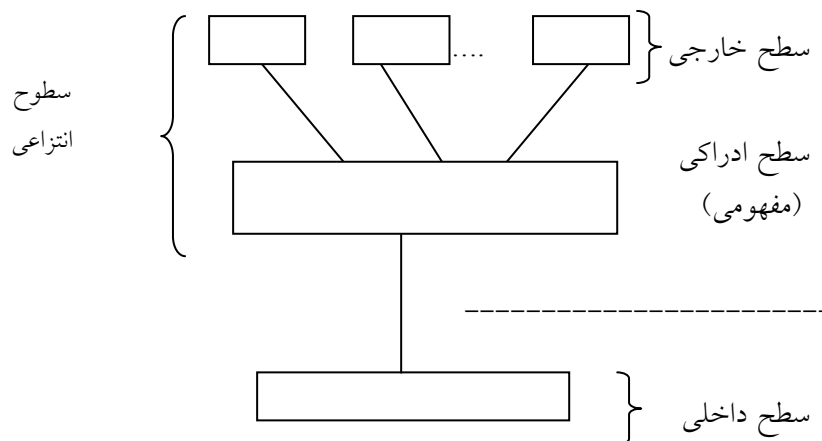
هدف رفتاری

در این فصل عناوین زیر مورد بحث و بررسی قرار می گیرند:

- معماری سه سطحی پایگاه داده
- دید (نمای) ادراکی (مفهومی)
- دید (نمای) خارجی
- دید (نمای) داخلی
- کاربر
- زبان میزبان
- زبان داده ای فرعی
- انواع زبان داده ای

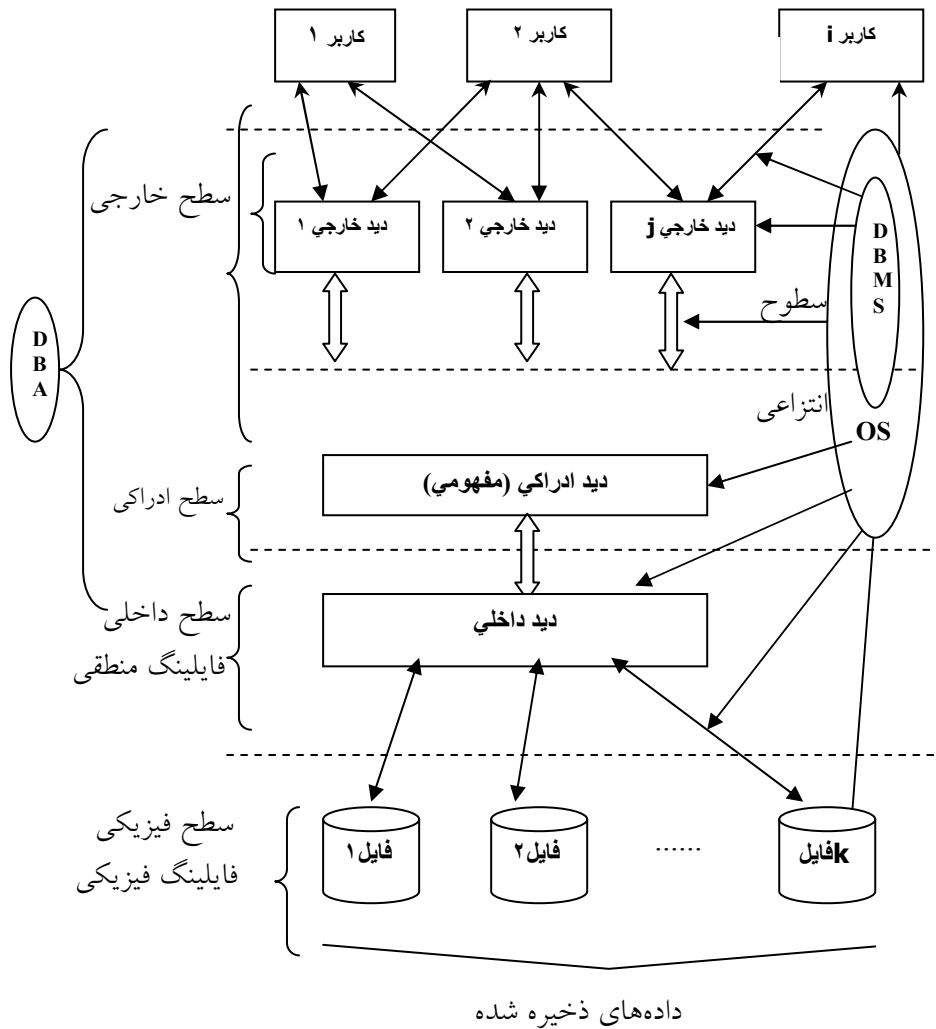
۱- معماری سه سطحی پایگاه داده

از آنجا که در چگونگی معماری "پایگاه داده‌ها" حداقل در سالهای آغازین ایجاد این تکنولوژی، بین کارشناسان اتفاق نظر نبود، لذا ANSI/SPARC با توجه به اهمیت محیط انتزاعی، معماری پایگاه داده‌ها را (به عنوان یک معماری استاندارد) پیشنهاد کرد. این معماری به معماری سه سطحی موسوم است. باید توجه داشت که این سه سطح، اساساً سه سطح "تعریف داده‌ها" هستند: دو سطح خارجی و ادراکی در محیطی انتزاعی هستند و سطح سوم که همان سطح داخلی است در محیط فایلینگ منطقی قرار گرفته است. شکل ۱-۳ نمای ساده‌ای از معماری سه سطحی را نمایش می‌دهد.



شکل ۱-۳ نمای ساده شده معماری سه سطحی

واضح است که سطح فیزیکی یک لایه پایین تر از سطح داخلی قرار می‌گیرد. در بعضی از پایگاه‌های داده، سطح فیزیکی نیز جزئی از معماری محسوب می‌گردد. نمای کاملتر معماری پیشنهادی ANSI برای پایگاه داده در شکل ۲-۳ نشان داده می‌شود.



شکل ۲-۳ معماری پایگاه داده‌ها

در این معماری علاوه بر سه سطح، اجزای دیگری هم دیده می‌شود که در واقع جزء "سیستم پایگاه داده‌ها" هستند. در اینجا سه سطح و نیز اجزاء دیگر را نام می‌بریم:

- کاربر
- زبان میزبان

- زبان داده‌ای فرعی
- دید خارجی
- دید ادراکی
- دید داخلی
- فایل‌های فیزیکی
- سیستم مدیریت پایگاه داده‌ها
- مدیر پایگاه داده‌ها

در ادامه برای درک بهتر موضوع، سطوح سه گانه مورد بررسی قرار می‌گیرند:

۲- شرح سطوح سه گانه

همانطور که در سمت چپ تصویر بالا مشاهده می‌کنید سه سطح (دید) مختلف در معماری پایگاه داده‌ها ارائه شده است. این سطوح عبارتند از:

- دید (نمای) ادراکی (مفهومی)
- دید (نمای) خارجی
- دید (نمای) داخلی

۱-۲ دید (نمای) ادراکی (مفهومی)

- دید یا نمای ادراکی در واقع همان دید طراح پایگاه داده‌ها نسبت به داده‌های ذخیره شدنی در پایگاه داده است.
- این دید یک دید جامع (سرتاسری) بوده و تمام نیازهای کاربران در محیط عملیاتی را در بر می‌گیرد.
- این دید در یک محیط انتزاعی مطرح است: بنابراین مبتنی است بر یک ساختار داده‌ای مشخص (از یک مدل داده‌ای که انتزاع لازم را تامین می‌کند).
- این دید، با استفاده از عناصر ساختاری اساسی همان ساختار داده‌ای، طراحی می‌شود.

- این دید باید (پس از طراحی طبعا) توصیف شود. به وصف یا شرح دید ادراکی، شمای ادراکی می‌گوییم. شمای ادراکی نوعی "برنامه" است حاوی دستورات "تعریف داده‌ها" و "کنترل داده‌ها" (و نه دستورات عملیات در داده‌ها). سطح ادراکی در واقع همین شمای ادراکی است.
- شمای ادراکی به سیستم داده می‌شود و در کاتالوگ سیستم نگهداری می‌شود.

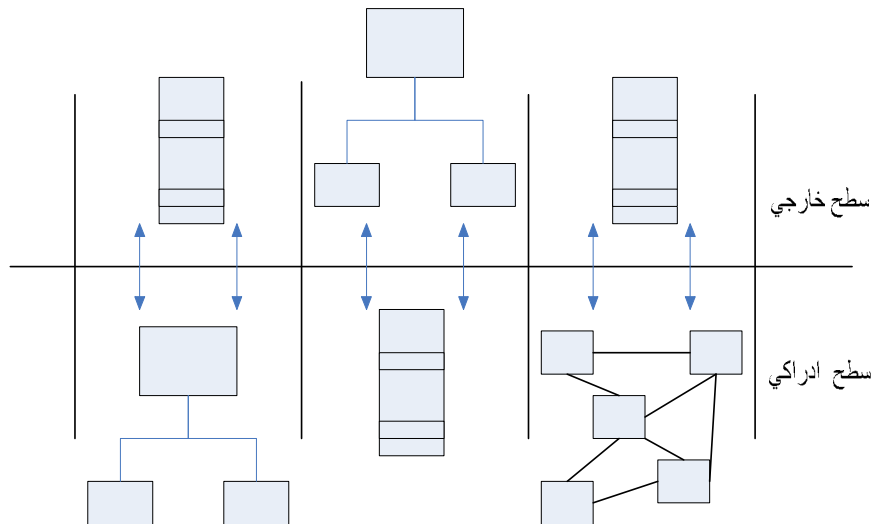
به بیانی ساده، دید ادراکی همان تعریف جدول می‌باشد. برای مثال فرض کنید می‌خواهید یک موجودیت مثل دانشجو را تعریف کنید. تعریف جدول مربوطه به همراه فیلدها (صفات خاصه) آن موجودیت در یک پایگاه داده، در بحث دید ادراکی دسته بندی می‌شود.

۲-۲ دید (نمای) خارجی

دید یا نمای خارجی در کتب مختلف با عبارات مختلفی توضیح داده شده است. در ادامه چند تعریف از دید خارجی مطرح شده است:

- مفهوم دید یا نمای خارجی در واقع همان مفهومی است که در تحلیل و طراحی یک سیستم بکار برده می‌شود. در ادامه تعاریف و نکاتی در مورد دید خارجی آمده است:
- دید کاربر خاص است نسبت به داده‌های ذخیره شده در پایگاه داده‌ها.
- این دید جزیی است و نه جامع: نشان دهنده "محدوده‌ای" از پایگاه داده‌ها که به نیازهای اطلاعاتی یک کاربر خاص پاسخ می‌دهد.
- این دید هم در سطح انتزاعی مطرح است: بنابراین مبتنی است بر یک ساختار داده‌ای مشخص و معمولاً همان ساختار داده‌ای که دید ادراکی بر اساس آن طراحی و تعریف می‌شود.
- این دید روی دید ادراکی طراحی و تعریف می‌شود (به همین دلیل به جدولهای سطح ادراکی، جدولهای مبنا یا پایه می‌گویند).
- به وصف یا تعریف دید خارجی، شمای خارجی می‌گوییم: نوعی "برنامه" که کاربر سطح خارجی می‌نویسد حاوی دستورات "تعریف داده‌ها" و گاه

- "کنترل داده‌ها" در همان سطح خارجی. شمای خارجی هم به سیستم داده می‌شود و در کاتالوگ آن نگهداری می‌شود.
- به تعریف مجموعه دیدهای خارجی کاربر، سطح خارجی گفته می‌شود.



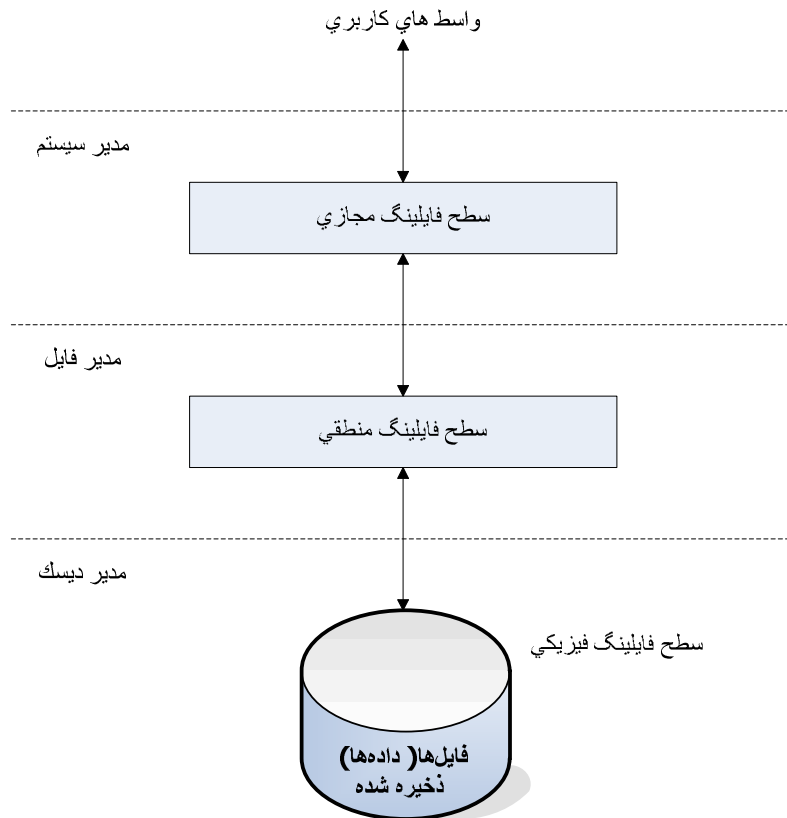
شکل ۳-۳ شمایی از نایکسانی ساختار داده‌ای در سطوح ادراکی و خارجی

نکته: حداقل از لحاظ نظری می‌توان گفت که ساختار (مدل) داده‌ای در دو سطح انتزاعی یعنی سطح خارجی و سطح ادراکی می‌تواند یکسان نباشد. با این وصف، بدیهی است که یک نرم‌افزار واسطه برای تبدیل دو ساختار به یکدیگر لازم است. شکل زیر تفاوت بین ساختارهای داده‌ای دو دید را مشخص می‌کند. عناوین ذکر شده در سطح ادراکی، در مورد ساختارهای داده‌ای است که در فصل بعد مورد بحث و بررسی قرار خواهند گرفت.

۳-۲ دید (نمای) داخلی

سطح داخلی، همان سطح فایلینگ منطقی پایگاه داده‌هاست. معنایش این است که DBMS به جنبه‌های فایلینگ فیزیکی پایگاه داده‌ها نمی‌پردازد. این کار بر عهده "سیستم فایل فیزیکی" در سیستم عامل است. در واقع، DBMS در محیط فایلینگ

منطقی عمل می‌کند، کم و بیش شبیه برنامه فایل پرداز که در سطح منطقی و نه فیزیکی، کار می‌کند.

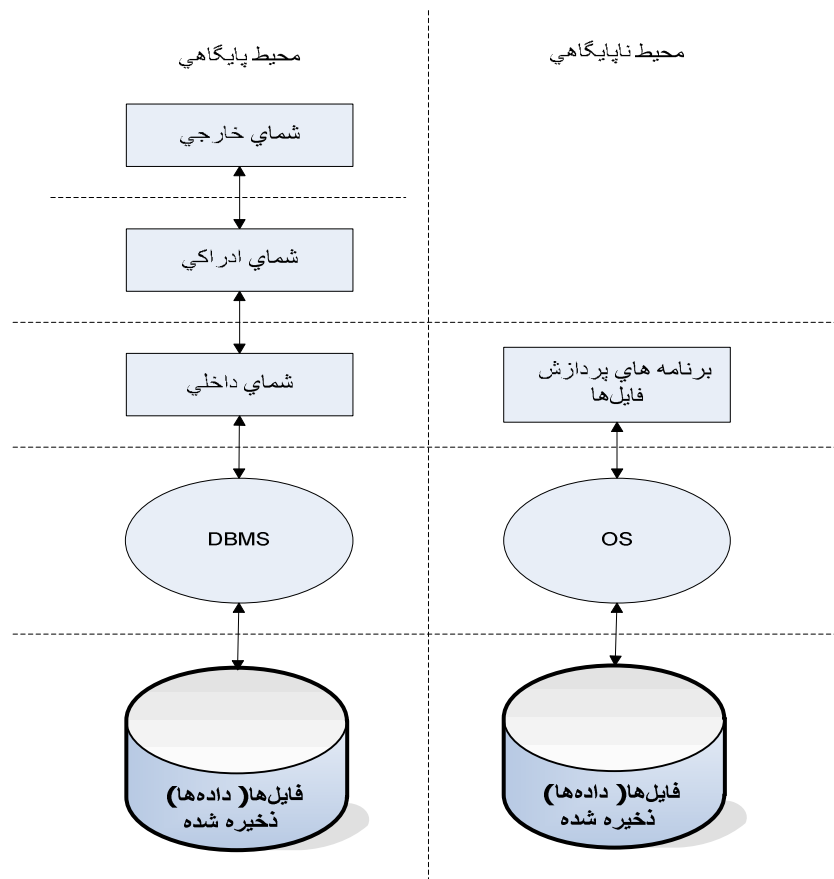


شکل ۳-۴ طرح شماتیک از ذخیره سازی فایلها در محیط DBMS

در ادامه توضیحاتی در خصوص دید داخلی ارائه شده است:

- دید DBMS (و نیز طراح پایگاه داده‌ها) است در سطح پایین تر از سطح ادراکی، نسبت به کل داده‌های ذخیره ش‌دنی در پایگاه داده‌ها.
- این دید در سطح فایلینگ منطقی (و گاه مجازی) پایگاه داده‌ها مطرح است.

- این دید مبتنی است بر یک (و گاه بیش از یک) ساختار فایل که معمولاً با نظر و دخالت طراح پایگاه داده‌ها طراحی می‌شود. این طراحی، اصطلاحاً به طراحی فیزیکی موسوم است.
- سطح داخلی پایگاه داده‌ها، در واقع سطحی است که در آن فایل‌های منطقی پایگاه داده‌ها تعریف می‌شود.
- به شرح یا تعریف دید داخلی، شمای داخلی گفته می‌شود: نوعی "برنامه" که توسط خود DBMS (و تا حدی با دخالت طراح پایگاه داده‌ها) تولید می‌شود و همانطور که گفتیم شرح یا توصیف فایلینگ منطقی پایگاه است که در واقع همان سطح داخلی است.



شکل ۵-۳ شمایی از تفاوت سطوح معماری بین محیط پایگاهی و ناپایگاهی

نکته: در بعضی از سیستم‌های پایگاهی، DBMS کل فضای پایگاه داده‌ها را به صورت مجموعه‌ای از مجموعه صفحات می‌بیند، یعنی نوعی نمای مجازی از داده‌های ذخیره شده در پایگاه دارد. این نمای مجازی بالاتر از سطح فایلینگ منطقی قرار می‌گیرد. در واقع بین سطح نمای مجازی و سطح فایلینگ فیزیکی، یک سطح فایلینگ منطقی واسط است. این سه سطح در شکل ۳-۴ نمایش داده شده است.

نکته: با توجه به دیدهای سه گانه ارائه شده در پایگاه داده، اکنون تفاوت سطوح معماری بین محیط پایگاهی و ناپایگاهی بهتر نمایان می‌گردد. شکل زیر این تفاوت را نمایان می‌کند.

۳- سایر اجزاء پایگاه داده‌ها

بغیر از سه دید اصلی ارائه شده در بالا، اجزاء دیگری نیز در پایگاه داده وجود دارند که عبارتند از:

- کاربر
- زبان میزبان
- زبان داده‌ای فرعی

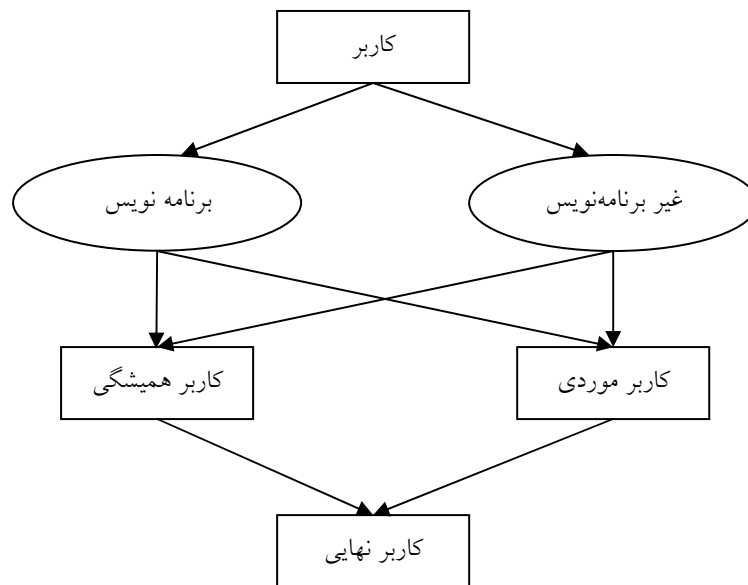
۳-۱ کاربر

در معنای عام، هر استفاده کننده از پایگاه داده‌ها را کاربر گوئیم. لازم بذکر است که در بعضی از کتب کاربران پایگاه داده‌ها بر حسب نوع کار دسته بندی می‌گردند، در اینجا کلمه کاربر بصورت عام استفاده شده است و از ذکر انواع کاربر خودداری شده است.

۳-۲ زبان میزبان

یکی از زبانهای برنامه سازی می‌باشد که قادر به برقراری ارتباط و انتقال دستورات بین خود و پایگاه داده باشد. واضح است که هرچه تعداد زبانهای میزبان مورد پذیرش DBMS بیشتر باشد، موارد ذیل تامین می‌گردد:

- تنوع کاربردها امکان پذیر می باشد
- تنوع کاربران تامین می گردد
- انعطاف پذیری سیستم بیشتر می گردد.



شکل ۳-۶ شمایی از انواع کاربران و ارتباط بین آنها

۳-۳ زبان داده‌ای فرعی

این زبان از سه دسته دستور تشکیل شده است:

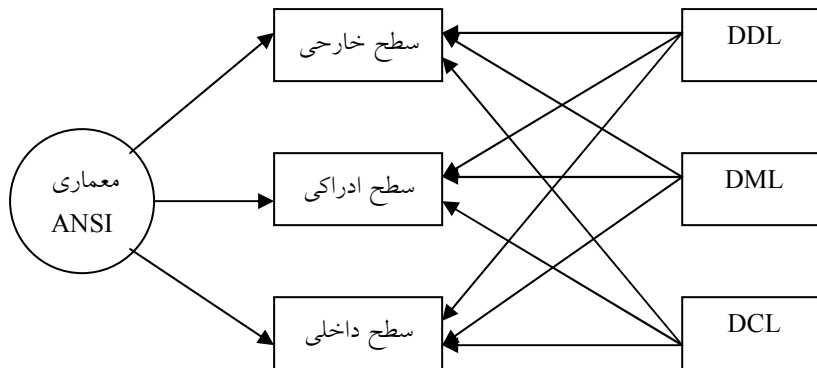
- دستورات تعریف داده‌ها^۱
- دستورات عملیات روی داده‌ها^۲
- دستورات کنترل داده‌ها^۳

این نوع دستورها باید برای هر یک از سطوح معماری پایگاه داده وجود داشته باشند. این موضوع در شکل ۳-۷ نشان داده شده است. زبان داده‌ای فرعی از نظر نیاز به

1. Data Definition Language (DDL)
 2. Data Manipulation Language (DML)
 3. Data Control Language (DCL)

زبان میزبان یا عدم نیاز به آن، به دو رده تقسیم می‌شود:

- مستقل یا خود کفا
- ادغام شدنی (ادغام شده)



شکل ۳-۷ ارتباط بین گروه‌های دستوری مختلف و سطوح معماری پایگاه داده

زبان مستقل

زبانی است که نیاز به زبان میزبان ندارد و خود به صورت تعاملی (اندر کنشی) استفاده می‌شود و آن را I.DSL گوئیم.

زبان ادغام شدنی

زبانی است که دستورهایش (به نحوی) در متن برنامه‌ای به زبان میزبان بکار می‌روند و مستقلاً قابل استفاده نیستند و آن را E.DSL گوئیم.

نکته: زبان داده‌ای فرعی ممکن است هم مستقل و هم ادغام شدنی باشد (I/E.DSL).

تمرینات

۱. دیدهای سه گانه ارائه شده در معماری پایگاه داده‌ها را نام ببرید ؟
۲. دید (نمای) ادارکی را شرح دهید ؟
۳. دید (نمای) خارجی را شرح دهید ؟
۴. دید (نمای) داخلی را شرح دهید ؟
۵. انواع دستورات زبان داده‌ای فرعی را نام ببرید ؟
۶. زبان مستقل به چه معنی است ؟
۷. زبان ادغام شدنی را شرح دهید ؟

فصل ۴

سیستم مدیریت پایگاه داده

هدف کلی

در این فصل ابتدا سیستم مدیریت پایگاه داده‌ها تعریف و سه بخش اصلی آن مورد بررسی قرار خواهند گرفت. سپس سیستم‌های مدیریت پایگاه داده‌ها از جهات مختلف مورد دسته بندی قرار خواهند گرفت. در ادامه اجزاء سیستم مدیریت پایگاه داده معرفی شده و کاتالوگ یا دیکشنری پایگاه داده‌ها توضیح داده خواهد شد. سپس پارامترهای مختلف مورد نیاز جهت شناخت DBMS مطرح شده و محورهای اصلی مقایسه DBMS شرح داده خواهند شد. در پایان نیز جایگاه مدیر پایگاه داده تشریح شده و تیم مدیریت پایگاه داده شرح داده خواهد شد.

هدف رفتاری

در این فصل عناوین زیر مورد بحث و بررسی قرار می گیرند:

- سیستم مدیریت پایگاه داده‌ها
- بخش ساختاری
- بخش عملیاتی
- بخش جامعیتی
- رده بندی سیستم‌های مدیریت پایگاه داده‌ها
- اجزاء سیستم مدیریت پایگاه داده‌ها
- نمای بیرونی پایگاه داده‌ها

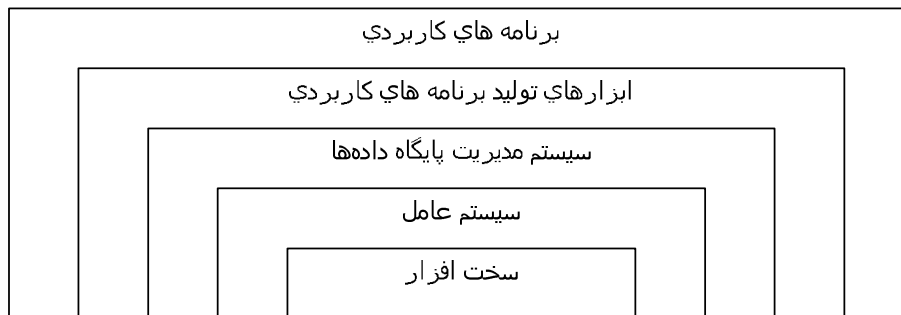
- نمای درونی پایگاه داده‌ها
- لایه مدیریت محیط پایگاه داده
- لایه هسته پایگاه داده
- کاتالوگ سیستم و دیکشنری داده‌ها: متا داده‌ها
- پارامترهای شناخت DBMS
- پارامترهای مربوط به معماری پایگاه داده‌ها
- سطح داخلی - فیزیکی
- سطح ادراکی
- سطح خارجی
- پارامترهای مربوط به زبان داده‌ای فرعی
- محورهای اصلی مقایسه DBMS‌ها
- مدیر پایگاه داده (DBA)
- تیم DBA، وظایف و مسئولیت‌های آن

۱- تعریف

سیستم مدیریت پایگاه داده‌ها یکی از انواع نرم‌افزارهای واسطه بین محیط فیزیکی ذخیره و بازیابی اطلاعات و محیط منطقی برنامه سازی است این نرم‌افزار به کاربر برنامه ساز امکان می‌دهد تا:

- پایگاه داده‌های خود را تعریف و ایجاد کند.
- در پایگاه داده‌های خود عملیات انجام دهد.
- روی پایگاه داده‌های خود تا حدی کنترل داشته باشد.

بدیهی است که یک سیستم مدیریت پایگاه داده‌ها در محیط یک سیستم عامل عمل می‌کند. شمای ساده‌ای از جایگاه این نرم‌افزار در یک سیستم کامپوتری در شکل زیر نشان داده شده است:



شکل ۴-۱ جایگاه DBMS در یک سیستم کامپیوتری

لازم به ذکر است که یک سیستم مدیریت پایگاه داده واقعی باید امکان انجام این سه فعالیت اساسی را قبل از هر چیز در محیط انتزاعی به کاربر بدهد. به بیانی دیگر می توان گفت که پایگاه داده از سه بخش اصلی به شرح ذیل تشکیل شده است:

- بخش ساختاری
- بخش عملیاتی
- بخش جامعیتی

بخش ساختاری

شامل عناصر ساختاری مدل است که همان ساختار داده ای اصلی و مفاهیم مرتبط با آن است.

بخش عملیاتی

مجموعه امکاناتی است که به وسیله آنها عملیات مورد نظر کاربر، از جمله بازیابی و ذخیره سازی، در کادر ساختار داده ای و مبتنی بر عنصر ساختار اساسی آن انجام می شود.

بخش جامعیتی

از مجموعه ای از قواعد جامعیتی تشکیل شده است که با استفاده از آن سیستم مدیریت پایگاه داده می تواند صحت، دقت و سازگاری داده ها را در هر لحظه از حیات پایگاه داده ها، کنترل و تضمین کند.

۲- رده بندی سیستم‌های مدیریت پایگاه داده‌ها

سیستم‌های مدیریت پایگاه داده را می‌توان به چندین صورت دسته‌بندی نمود. این دسته‌بندی‌ها در ذیل آمده‌اند:

از نظر مدل داده‌ای

- سیستم رابطه‌ای
- سیستم سلسله مراتبی
- سیستم شبکه‌ای

از نظر محیط سخت‌افزاری

- قابلیت اجرا بر روی یک سخت‌افزار خاص
- قابلیت اجرا بر روی سخت‌افزارهای متنوع

از نظر رده بندی کامپیوتر

- قابل اجرا بر روی کامپیوترهای بزرگ^۱ و خیلی بزرگ^۲
- قابل اجرا بر روی کامپیوترهای متوسط
- قابل اجرا بر روی کامپیوترهای شخصی
- قابل اجرا بر روی انواع کامپیوترها

از نظر محیط سیستم عامل

- وابسته به یک نوع خاص از سیستم عامل
- عدم وابستگی به سیستم عامل و قابل اجرا بر روی چند سیستم عامل

از نظر نوع معماری سیستم پایگاه داده‌ها

- دارای معماری پایگاه داده متمرکز
- دارای معماری پایگاه داده نا متمرکز

1. Mainframe

2. Super Computer

از نظر معماری مشتری - خدمتگذار^۱

- توانایی در ارائه معماری یک خدمتگذار - چند مشتری
- توانایی در ارائه معماری چند خدمتگذار - چند مشتری

از نظر سیستم فایل

- مستقل از سیستم فایلینگ مربوط سیستم عامل
- وابسته به سیستم فایلینگ سیستم عامل

از نظر متدولوژی زبان پایگاهی

- بدون متدولوژی شیئی گرایي
- دارای متدولوژی شیئی گرایي

از نظر بهینه سازی پرس و جوها و درخواستهای کاربر

- دارای بهینه ساز متعارف
- دارای بهینه ساز مبتنی بر قاعده، معنایی و...

از نظر نوع تراکنش

- پذیرنده تراکنشهای ساده و تک سطحی
- پذیرنده تراکنشهای با مدل پیشرفته (مثلا تودرتو، زنجیره ای و...)

از نظر نوع پردازش

- با قابلیت پردازش بی درنگ
- فاقد این قابلیت

۳- اجزاء سیستم مدیریت پایگاه داده ها

این نرم افزار، مثل هر نرم افزار نیمه بنیادی دیگر از واحدهایی تشکیل شده است. تعداد واحدها و حجم هر واحد نرم افزاری بستگی به توانایی سیستم در انجام کارها

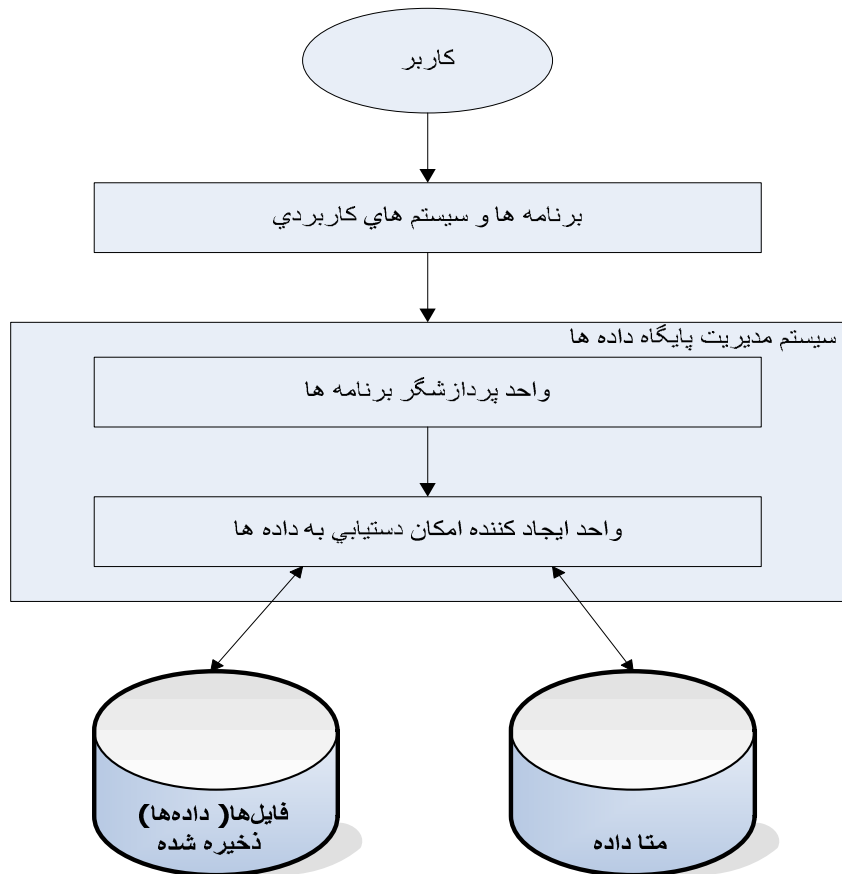
1. Client-Server Architecture

و ارائه خدمات به کاربران دارد. در ادامه واحدهای اصلی سیستم مدیریت پایگاه داده‌ها معرفی خواهیم کرد.

۳-۱ نمای بیرونی

نمای بیرونی، از واحدهای اصلی زیر تشکیل شده است:

- واحد پردازشگر "پرسش‌ها"
- واحد پردازشگر "برنامه‌های کاربردی"
- واحد ایجاد و مدیریت داده‌های ذخیره شده.



شکل ۴-۲ نمای بیرونی سیستم مدیریت پایگاه داده

در اینجا لازم است به این نکته توجه شود که پرسشها نیز در واقع نوعی از برنامه کاربردی و یا در مواردی قسمتی از برنامه کاربردی هستند. تفاوت بین این دو در این نکته است که برنامه‌های کاربردی با یک زبان برنامه نویسی تهیه می‌شوند که به طریقی با پایگاه داده ارتباط برقرار می‌کند و دستورات مورد نیاز برای کار با پایگاه داده درون زبان انتخاب شده می‌باشد که توسط کامپایلر زبان برای پایگاه داده قابل فهم می‌شود. ولی پرسشها با استفاده از دستوراتی که برای پایگاه داده قابل فهم می‌باشد، تهیه می‌شوند و در واقع به نوعی وابستگی کامل به پایگاه داده مورد استفاده دارد. شکل ۴-۲ نمای بیرونی سیستم مدیریت پایگاه داده را بصورتی ساده بیان می‌کند.

۳-۲ نمای درونی

این نرم افزار (نمای درونی) از سه لایه تشکیل شده است که هر لایه وظایف خاص خود را بر عهده دارد. لازم به ذکر است که در کتابهای مختلف ممکن است وظایف هر یک از این لایه‌ها متفاوت باشد. در ادامه لایه‌های این قسمت به همراه وظایف هر یک شرح داده شده است:

- لایه مدیریت محیط پایگاه داده‌ها^۱
- لایه هسته^۲ (سیستم کنترل یا موتور پایگاه داده‌ها)
- لایه تسهیلات نرم افزاری^۳ (ابزارها)

لایه مربوط به تسهیلات نرم افزاری معمولاً در سیستم‌های مدیریت پایگاه داده‌ها متفاوت می‌باشد و توانایی‌ها و خدمات قابل ارائه در این لایه بعضاً بسیار متنوع و یا بسیار ساده است. لذا از بحث پیرامون این لایه اجتناب می‌گردد.

۳-۲-۱ لایه مدیریت محیط پایگاه داده

لایه مدیریت محیط پایگاه داده وظایف زیر را بر عهده دارد:

1. Database Environment Management Layer

2. Engine Layer

3. Software Facilities

- کنترل جامعیت^۱ پایگاه داده
- کنترل ترمیم^۲ (بازسازی) پایگاه داده
- ایمنی و حفاظت پایگاه داده^۳
- تولید نسخه‌های پشتیبان^۴
- تولید فایل‌های ثبت تراکنش^۵ها

۳-۲-۲ لایه هسته پایگاه داده

لایه هسته در واقع لایه اصلی دریافت کننده درخواست‌ها و انجام عملیات مرتبط با آن را بر عهده دارد. مهمترین وظایف این واحد به شرح زیر می‌باشد:

- دریافت درخواست‌ها و پیش کامپایل
- کامپایل درخواست‌ها و بهینه سازی پرس و جو
- مدیریت فایلینگ منطقی و فضای دیسک
- مدیریت بافر (حافظه نهان)
- مدیریت و نظارت بر زمان اجرا دستورات
- مدیریت همزمانی تراکنش‌ها
- مدیریت انتقال داده‌ها
- مدیریت کاتالوگ سیستم

۴- کاتالوگ سیستم و دیکشنری داده‌ها (متا داده‌ها)

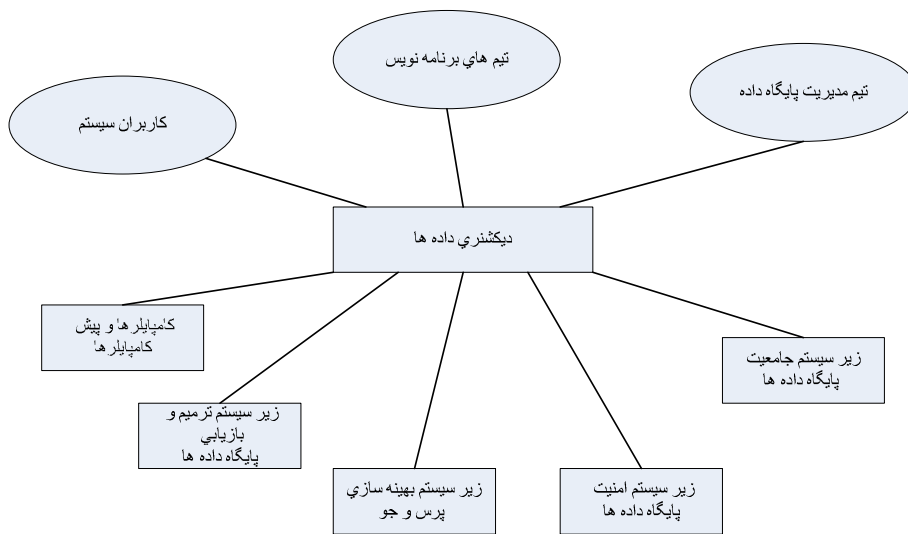
یکی از ویژگی‌های مشی پایگاهی این است که سیستم پایگاهی نه تنها حاوی پایگاه داده‌ها است، بلکه تعریف کامل یا توصیف پایگاه داده‌ها و قواعد ناظر به آنرا نیز در خود دارد. کاتالوگ حاوی داده‌هایی است در مورد داده‌های ذخیره شده در پایگاه داده‌های کاربر. در ادامه نکاتی چند در مورد کاتالوگ سیستم ارائه می‌گردد:

-
1. Integration Control
 2. Repair Control
 3. Database Protection
 4. Backup Generation
 5. Transaction Record

- داده‌های حاوی اطلاعات در مورد داده‌های ذخیره شده در پایگاه‌داده، به متا داده‌ها موسومند.
- در بعضی از کتاب‌ها به کاتالوگ سیستم گاه دیکشنری داده‌ها هم گفته می‌شود، اما در واقع دیکشنری داده‌ها حاوی اطلاعات بیشتری است.
- متا داده‌ها معمولاً از دید کاربر سطح خارجی نهان‌اند، اما مدیر سیستم و یا کاربر مجاز، می‌تواند تا حدی از محتوای کاتالوگ آگاه شود.
- ساختار و محتوای کاتالوگ و دیکشنری داده‌ها در سیستم‌های مختلف یکسان نیست. اما به طور کلی، اطلاعات زیر در آنها ذخیره می‌شود:

دیکشنری داده‌ها معمولاً جزیی از خود سیستم است و اطلاعات زیر در آنها نگهداری می‌شود:

- شماهای خارجی
- شماای ادراکی
- شماای داخلی
- رویه‌های مربوط به تبدیلات بین سه سطح معماری
- شرح ساختار فیزیکی داده‌های ذخیره شده
- مشخصات کاربران و حقوق دستیابی آنها به داده‌های ذخیره شده
- مشخصات برنامه‌های کاربردی تولید شده و ارتباط آنها با درخواست کاربران
- مشخصات پایانه‌های متصل به سیستم و یا کامپیوترهای مشتری و ایستگاههای کاری
- ارتباط بین برنامه‌های کاربردی و داده‌های ذخیره شده
- قواعد مربوط به کنترل صحت و دقت داده‌های ذخیره شده در پایگاه‌داده‌ها
- ضوابط کنترل‌ایمنی داده‌ها
- مشخصات پیکر بندی سخت‌افزاری سیستم و رسانه‌های ذخیره سازی
- اطلاعات متنوع آماری در مورد پایگاه‌داده‌ها، عملیات کاربران، فرکانس اجرای تراکنشها و تعداد دستیابی به اشیاء ذخیره شده
- توابع تعریف شده توسط کاربران



شکل ۴-۳ شمای کلی کاتالوگ سیستم پایگاه داده‌ها

۵- پارامترهای شناخت DBMS

امروز مرتباً شاهد عرضه سیستم‌های با نامهای گوناگون تحت عنوان DBMS به بازار مصرف هستیم. عدم اطلاع کافی خریداران و استفاده کنندگان چنین سیستمی از پارامترهای انتخاب سبب می‌شود تا اولاً انتخاب سیستم مناسب برای یک محیط عملیاتی (سازمان) مخصوصاً محیط بزرگ، گاه بر اساس ملاحظات غیر علمی-فنی صورت پذیرد و گاه اساساً نوعی سردرگمی در انتخاب بروز کند (البته در انتخاب و خرید یک سیستم ملاحظات جز ملاحظات مستقیماً مربوط به خود سیستم و پایگاهی که قرار است به کمک آن ایجاد شود، نیز می‌تواند مطرح باشد). ثانیاً نرم‌افزار خریداری شده منطبق با نیازهای محیط عملیاتی و پاسخگوی احتیاجات اطلاعاتی محیط نباشد. ثالثاً از نرم‌افزار به هر حال انتخاب شده بطور بهینه بهره برداری نشود و گاه پیامدهای نامطلوبی بروز کند.

تصمیم‌گیری در مورد انتخاب سیستم مناسب، مسئله اساسی هر سازمان است. این مسئله مخصوصاً در شرایطی که تحولات تکنولوژیک در نرم‌افزار و سخت‌افزار چنان شتابنده است که هر روز تولیدات نرم‌افزاری و سخت‌افزاری جدیدی به بازار

مصرف عرضه می‌شوند، بالاخص برای محیط‌های عملیاتی بزرگ، بسیار استراتژیک می‌نماید. سازمانهایی که با مشکل برنامه ریزی میان مدت و دراز مدت مواجه‌اند، اگر در تصمیم‌گیری برای انتخاب این تکنولوژی با تخصص وافی و مطالعه همه جانبه و نهایتاً با شناخت کافی اقدام نکنند، نه تنها به اهداف تعریف شده در برنامه ریزی نایل نخواهد شد بلکه چه بسا راه به بیراهه خواهد برد و با مشکلات عدیده‌ای مواجه خواهند شد که رفع آنها به آسانی امکان پذیر نخواهد شد.

با توجه به حل بسیاری از مشکلات تکنیکی در ایجاد سیستم‌های توزیع شده ناهمگن، اینک دیگر مسئله انتخاب فقط یک DBMS و یا فقط یک پیکر بندی نرم‌افزاری-سخت‌افزاری می‌تواند چندان مطرح نباشد. با این‌همه انتخاب اجزاء تشکیل دهنده پیکر بندی یک سیستم ناهمگن به هر حال مطرح است تا یک پیکر بندی با بیشترین کارائی، طراحی و تامین شود.

به منظور کسب شناخت تخصصی باید پارامترهای زیادی را در نظر گرفت که این پارامترها را بصورت کلی در پنج رده به شرح زیر دسته بندی می‌کنیم:

- پارامترهای مربوط به توانایی‌ها و کارایی سیستم
- تسهیلات و جنبه‌های دیگر
- مشخصات کلی سیستم
- پارامترهای مربوط به معماری پایگاه‌داده‌ها
- پارامترهای مربوط به زبان داده‌ای فرعی (واسط کار برنامه ساز).

در ادامه هر یک از دسته‌های اصلی را مورد بحث و بررسی قرار خواهیم داد:

۵-۱ پارامترهای مربوط به توانایی‌ها و کارایی سیستم

- توان سیستم در ایجاد و مدیریت پایگاه‌داده‌ها بر اساس معماری پیشنهادی ANSI یعنی سیستم با معماری سه سطحی.
- حداکثر تعداد کاربران همزمان.
- توانایی سیستم در ترمیم داده‌ها و مکانیسم‌های انجام این کار.
- توانایی سیستم در تامین همزمانی، مکانیسم قفل گذاری، اندازه واحد قفل پذیر و وجود استراتژی کارا در حل مشکل بن بست.

- توانایی سیستم در تامین ایمنی داده‌ها
- توانایی سیستم در حفاظت داده‌ها و تضمین محرمانگی آنها
- توانایی سیستم در تامین رشد پایگاه داده‌ها و درجه سهولت انجام این کار.
- توانایی سیستم در تضمین جامعیت پایگاه
- تعداد زبانهای میزبان مورد پذیرش سیستم.
- مکانیسم هماهنگی عملکرد سیستم با سیستم عامل خاص پایگاه.
- شرایط بروز تنگناها در عملیات ورودی / خروجی و در فعالیت واحد پردازش مرکزی و مکانیسم رفع آن.
- توانایی سیستم در اشتراکی کردن داده‌ها
- امکانات سیستم در مجاز شماری و سطح یاسطوح استفاده از این امکانات (با توجه به معماری چند سطحی پایگاه).
- آستانه پایداری کارایی سیستم (از نظر تعداد تراکنش‌های اجرا شونده مثلا در ثانیه و با توجه به حجم داده‌های ذخیره شده در پایگاه).
- توانایی سیستم در پردازش پرسشهای موازی.
- توانایی سیستم در پذیرش انواع برنامه‌های کاربردی و پرسشها.
- نحوه پردازش زبان داده‌ای فرعی توسط سیستم (کامپایلری، مفسری).
- توانایی سیستم در بهینه سازی پرسشها و نوع بهینه سازی (اعمال الگوریتم‌های بهینه در اجرای پرسشها) و زمان انجام بهینه سازی استراتژی دستیابی (در زمان کامپایل، در اولین بار اجرای پرسش، در هر بار که پرسش اجرا می‌شود).
- مدل‌های تراکنش که می‌پذیرد (بویژه مدل‌های پیشرفته).
- قابلیت گسترش پذیری و ارتقاء سیستم.
- توانایی سیستم در سازماندهی مجدد محیط فیزیکی پایگاه و درجه پویایی و خودکار بودن این کار.
- ایجاد سیستم در خدمت رسانی در محیط شبکه‌ای.
- نوع معماری سیستم پایگاه داده‌ها که سیستم می‌تواند آن را تامین کند.
- قابلیت تعامل سیستم با سیستم‌های متعارف دیگر (با مدل داده‌ای و زبان داده‌ای یکسان یا متفاوت) و مکانیسم ارتباط با آنها.

- قابلیت تعامل سیستم با گونه‌های غیر متعارف سیستم مدیریت پایگاه داده‌ها و مکانیسم و ملزومات همایندی با سیستم‌ها.
- قابلیت تعامل سیستم با سیستم مدیریت پایگاه دانش و مکانیسم همایندی.
- قابلیت تعامل سیستم با سیستم مدیریت پایگاه داده‌های شیء‌گرا و مکانیسم همایندی.
- میزان تغییر کارآیی سیستم از ماشین کلاسیک تا ماشین خاص پایگاه داده‌ها.
- امکانات سیستم برای ایجاد پایگاه داده‌های زمانبند.
- زبانی که سیستم با آن نوشته شده است.
- میزان رعایت اصول نوین مهندسی نرم‌افزار در طراحی و تولید سیستم.
- میزان رعایت اصول شیء‌گرایی در طراحی و پیاده سازی سیستم.
- توانایی سیستم در ایجاد و مدیریت پایگاه‌های بزرگ و خیلی بزرگ (از نظر حجم داده‌ها و تعداد تراکنشها).
- قابلیت سیستم در نهان نگاری داده‌ها
- قابلیت سیستم در پشتیبانی انواع معماری سیستم پایگاه داده‌ها.
- قابلیت سیستم در کنترل اشتباهات برنامه سازی.
- قابلیت سیستم در عملیات در محیط پایگاه داده‌های وب و یا ادغام در "وب"
- قابلیت سیستم در عملیات در محیط پایگاه داده‌های موبایل
- قابلیت جابجایی پذیری سیستم
- متوسط زمان بین دو خرابی پی در پی سیستم
- میزان کار لازم برای پیاده سازی سیستم
- استانداردهای بکار رفته در سیستم
- سیستم عامل مورد نیاز
- قابلیت سیستم در پردازش پیام‌ها.
- تجهیزات لازم برای سیستم از جمله:
- نوع سخت افزار
- حداقل حافظه لازم
- حداقل فضای دیسک

- نوع و تعداد سیستم عامل
- تعدد و تنوع رسانه‌های جانبی

۵-۲ تسهیلات و جنبه‌های دیگر

این تسهیلات و جنبه‌ها می‌توانند همراه خود سیستم و یا به نحوی قابل تامین و استفاده در محیط سیستم باشند (به صورت نرم‌افزار جداگانه به کاربران سیستم عرضه شوند). هر چه سیستم از نظر پذیرش انواع تسهیلات و سازگاری و همایندی با آنها غنی تر باشد، مطلوبتر است و بهره برداری بهتری از سیستم امکان پذیر خواهد بود. برخی از این تسهیلات (ابزارها) عبارتند از:

- امکانات تولید نسخه‌های پشتیبان
- امکانات پرسش به کمک مثال و پرسش به کمک فرم
- امکانات بررسی‌ها و تحلیل‌های آماری
- امکانات گرافیکی
- امکانات نگهداری سیستم
- امکانات بهینه سازی برنامه‌های کاربردی
- امکانات یادگیری طرز کار با سیستم و بهره برداری آن
- امکانات کار در محیط وب
- امکانات مدیریت پویای پرسشها
- امکانات نظارت بر عملیات کاربران در پایگاه
- امکانات مدیریتی برای مدیر پایگاه داده‌ها
- امکانات پشتیبانی از XML
- امکانات پردازش تحلیلی بر خط
- مولد گزارش
- انواع ویرایشگرها
- ابزارهای ایجاد برنامه‌های کاربردی و واسطه‌های کاربردی
- امکانات تعریف گروه‌های کاربردی و تعیین امتیازات هر گروه (در انجام عملیات در پایگاه داده‌ها)

- امکانات دستیابی به داده‌های دوردست
- امکانات تولید خروجی‌های کاربرپسند
- امکانات تنظیم کردن پایگاه
- امکانات تنظیم کردن خود سیستم
- امکانات بارگذاری (گاه موسوم به عمل ورود داده‌ها)، بازبارگذاری، خالی کردن (گاه موسوم به عمل صدور داده‌ها) پایگاه‌داده‌ها
- امکانات تبدیل یک ساختار داده از سطح خارجی معماری ANSI به ساختار داده دیگر از سطح ادراکی همان معماری
- امکانات فعال و غیرفعال کردن پایگاه
- امکانات تهیه آمارهای مربوط به عملیات ذخیره سازی (درج، حذف، بهنگام سازی)
- امکانات سیستم برای مدل‌سازی داده‌ها و طراحی منطقی و فیزیکی پایگاه
- امکانات سیستم برای گشت زنی (گذارگری) در پایگاه‌داده‌ها
- امکانات استفاده از زبانهای سطح بالا (نه به عنوان زبان میزبان) و مکانیسم پیوند به سیستم از طریق آنها
- امکان ایجاد مدولهای اجرایی برای برنامه‌های کاربردی در محیط سیستم به نحوی که خارج از محیط سیستم قابل اجرا باشند
- امکان استفاده از یک سیستم خبره در محیط سیستم
- مولد فرم
- مولد منو
- امکانات شبکه‌ای
- واسط 4GL
- ابزارهای تولید مستندات (در مراحل مختلف تولید یک سیستم کاربردی)

آنچه بر شمرده شد، فهرستی است از تسهیلات و جنبه‌ها، و با توجه به پیشرفت مهندسی نرم‌افزار، امکانات دیگری نیز می‌توانند مطرح باشند. به ویژه که هر روز انواع گوناگونی از این قبیل ابزارها و تسهیلات تولید و به بازار مصرف عرضه می‌شوند. سیستم نه تنها باید بتواند با این امکانات تماس برقرار کند بلکه باید به

سهولت با آنها همایندی داشته باشد.

۵-۳ مشخصات کلی سیستم

- نام نرم‌افزار
- عنوان شرکت سازنده و شرکت فروشنده
- شماره ویراست (نگارش) مورد بررسی - شماره آخرین ویراست
- تاریخ عرضه (اولین نگارش و آخرین نگارش)
- قیمت
- نام کشور سازنده و فروشنده
- شرایط کلی تحویل
- خدمات بعد از تحویل
- کمیت و کیفیت آموزش
- مستندات
- تعداد مشتری‌ها و ماهیت نیازهای اطلاعاتی و پردازشی آنان
- ضمانت (های) شرکت فروشنده
- امکانات شرکت فروشنده در گسترش یا ارتقاء سیستم
- در دسترس بودن فروشنده (هرگاه که لازم باشد)
- وجود تیم فنی پشتیبانی سیستم، در شرکت فروشنده

۵-۴ پارامترهای مربوط به معماری پایگاه داده‌ها

از آنجائیکه معماری پایگاه داده‌ها در سه سطح داخلی، ادراکی و خارجی در نظر گرفته می‌شود، لذا پارامترهای مربوط به معماری را نیز در این سه سطح بررسی می‌کنیم:

۵-۴-۱ سطح داخلی - فیزیکی

- نوع داده
- نام فیلد
- طول داده (حداکثر طول پیش نهاده)

- حداکثر تعداد رکورد
- حداکثر طول هر رکورد
- ثابت یا متغیر بودن طول رکورد
- حداکثر تعداد فیلد در هر نوع
- حداکثر طول هر فیلد
- ثابت یا متغیر بودن طول فیلد
- حداکثر تعداد فایل‌های باز در یک زمان
- حداکثر تعداد تراکنش‌های عمل‌کننده در محیط داخلی-فیزیکی
- حداکثر تعداد شاخص‌ها و ساختار درونی هر شاخص و درجه پویایی آن
- اندازه شاخص
- هزینه ایجاد شاخص
- ساختار فایل‌های موجود
- حداکثر تعداد کلیدها
- حداکثر طول کلید
- نوع کلید
- نحوه انجام عملیات ذخیره سازی در محیط داخلی-فیزیکی و کارایی سیستم در هر عمل
- نحوه جایگیری فایل‌ها روی رسانه خارجی
- نحوه پیاده سازی "هیچمقدار" در این سطح و میزان حافظه مصرفی برای آن
- چگونگی تناظر (نگاشت) بین فایل‌ها و ساخت‌های سطح ادراکی
- چگونگی خوشه واری و موضعی بودن فایل‌ها و رکوردها
- چگونگی فشردگی داده‌ها و نسبت فشردگی سازی
- فشردگی شاخص‌ها و نسبت فشردگی سازی
- اندازه و تعداد بافر مورد نیاز
- اندازه بلاک
- اندازه صفحه (ثابت یا متغیر)
- وجود زبان داده‌ای فرعی برای سطح داخلی و امکانات آن

- میزان دخالت مدیر پایگاه داده‌ها در ایجاد سطح داخلی-فیزیکی و کنترل آن

۵-۴-۲ سطح ادراکی

- وجود سطح ادراکی
- درجه انتزاع سطح ادراکی (که مستقیماً به مدل داده‌ای بستگی دارد)
- وجود زبان داده‌ای فرعی در سطح ادراکی و امکانات آن
- امکانات تعریف قواعد (محدودیت‌های) جامعیت (به عنوان بخشی از مدل داده‌ای) در بیرون از برنامه‌های عملیاتی (با احکام تعریف داده‌ها)
- امکانات تعریف انواع کلید (کلید کاندید، کلید اصلی، کلید خارجی، کلید دیگر و...)
- میزان تامین استقلال داده‌ای فیزیکی
- میزان دینامیسم رشد پایگاه در سطح ادراکی
- امکانات سازماندهی مجدد پایگاه در سطح ادراکی (تغییر در طراحی منطقی و در پی آن تغییر شما)
- چگونگی مکانیسم نگاشت سطح ادراکی به داخلی و جنبه‌های نگاشت (نام داده، نوع داده، طول داده، کد نمایش داده، واحد داده و...)
- پیامدهای سازماندهی مجدد سطح ادراکی در سطح داخلی-فیزیکی
- میزان افزونگی در سطح ادراکی و مکانیسم کنترل آن و میزان انعکاس آن در سطح داخلی-فیزیکی
- وجود مفهوم میدان و امکانات لازم برای آن در سطح ادراکی
- نحوه برخورد با "هیچمقدار" در سطح ادراکی
- امکانات سیستم در مجازشماری به ویژه در این سطح

۵-۴-۳ سطح خارجی

- وجود مفهوم همگانی دید خارجی
- حداکثر تعداد دیدهای هر کاربر و در مجموع، تنوع دیدها از نظر مکانیسم تعریف آنها

- وجود زبان داده‌ای فرعی خارجی و امکانات آن (برای بازیابی، ذخیره سازی و کنترل)
- درجه انتزاع تامین شده در این سطح و ارتفاع این انتزاع
- میزان تامین استقلال داده‌ای منطقی
- چگونگی نگاشت سطح خارجی به ادراکی و محورهای این نگاشت (حدود آزادی عمل کاربران در تعیین جنبه‌های داده)
- قدرت سیستم در انجام عملیات بهنگام سازی دیدها (بهنگام سازی به معنای عام یعنی درج، حذف و تغییر) و تنوع دیدهای پذیرای این عملیات
- وجود کلید اصلی در این سطح (برای بعض دیدها به تشخیص مدیر پایگاه داده‌ها)

۵-۵ پارامترهای مربوط به زبان داده‌ای فرعی

- میزان پابندی احکام زبان به عناصر ساختاری مدل داده‌ای در جهت تامین انتزاع هرچه بیشتر
- وجود دستورات DDL، DML، DCL برای هر سه سطح معماری ANSI
- داشتن هر دو جنبه خودکفا (مستقل) بودن و قابل ادغام بودن
- انواع داده‌ای موجود
- گسترش پذیری نوع داده‌ای و وجود نوع داده‌ای تعریف شده توسط کاربر
- پذیرش BLOB و داده‌های از نوع تصویر، ویدئو و متن
- امکان نوشتن رویه ذخیره شده و رهانا
- جهت جابجایی مکان نما (جلو، عقب و هر دو سو)
- واسط‌های محیط برنامه سازی (مثل ODBC و...)
- درجه کمال از نظر ساختار داده‌ای
- درجه کمال برنامه سازی
- سازگاری با 4GL
- رویه‌ای یا نا رویه‌ای بودن
- مفسری یا کامپایلری بودن

- رعایت اصل وحدت احکام برای عمل واحد در دو سطح انتزاعی و نیز در یک سطح مشخص
- وجود امکان کنترل رکوردهای تکراری برای مدیر پایگاه داده‌ها
- قدرت در برنامه سازی الگوریتم‌های بهینه برای انجام چهار عمل اصلی (بازیابی، حذف، درج و بهنگام سازی)
- سهولت نوشتن برنامه‌های لازم برای انجام چهار عمل اصلی (با توجه به مفاهیم ساختار داده‌ای)
- سهولت یادگیری و نزدیکی به زبان طبیعی
- وجود امکان معرفی میدان و عملیات روی میدان‌ها
- وجود احکام لازم برای اعمال قواعد جامعیتی
- وجود احکام لازم برای ایمنی و حفاظت داده‌ها
- وجود احکام لازم برای اعمال ضوابط مجازشماری
- وجود احکام لازم برای تعیین حدود تراکنش‌ها و کنترل همزمانی آنها
- وجود امکان نوشتن الگوریتم‌های موازی پرسش
- میزان شیئی گرا بودن
- امکانات لازم برای داده‌های زمانبند

۶- محورهای اصلی مقایسه DBMSها

در مقایسه این سیستم‌ها، باید تمام پارامترهای گفته شده در قسمت ۹ این گفتار را در نظر داشت. اما با بررسی این پارامترها می‌توان محورهای اصلی مقایسه را بدست آورد. در زیر محورهای مهمتر را بر می‌شمیریم (فرض بر این است که سیستم‌های مقایسه شونده معماری ANSI را پشتیبانی می‌کنند):

- امکانات تعریف داده‌ها
- انواع داده‌ای مورد پذیرش سیستم
- امکانات عملیات در داده‌ها
- نوع تراکنش‌ها (چند پرسشی یا برنامه، تعداد آنها) و طرز مدیریت آنها
- امکانات پردازش پرسش‌ها، بهینه سازی آنها و زمان بهینه سازی

- امکانات ایجاد دیکشنری داده‌ها
- وضعیت سطح داخلی - فیزیکی پایگاه داده‌ها از نظر ساختار فایلها، شیوه‌های دستیابی (بویژه شاخص بندی) و جنبه‌های دیگر.
- معماری سیستم پایگاهی قابل ایجاد و مدیریت
- محیط سخت افزاری و حداقل امکانات لازم
- محیط سیستم عامل لازم
- تکنیکهای کنترل همزمانی
- امکانات مدیریت محیط پایگاه داده‌ها: کنترل جامعیت، ترمیم، ایمنی و حفاظت.
- تسهیلات طراحی پایگاه داده‌ها (طراحی منطقی و طراحی فیزیکی).
- مکانیسم ادغام زبان داده‌ای فرعی در زبان میزبان (وجود پیش کامپایلر یا فراخوانی توابع).
- تنوع دیدهایی که عملیات ذخیره سازی را می‌پذیرند.
- مکانیسم مجازشماری کاربران (نامتمرکز، متمرکز یا نیمه متمرکز).
- امکانات مورد نیاز مدیر پایگاه داده‌ها.
- امکانات نهان نگاری داده‌ها.
- طرز انجام عمل بهنگام سازی (درجا یا برون از جا).
- الگوریتم اجرای عملگر پیوند و سایر عملگرهای جبر رابطه‌ای.
- امکانات تعامل با سیستم‌های دیگر (نوع میان افزار و...)
- طرز نمایش نتایج عملیات (گرافیک و گزارش).
- پذیرش یا عدم پذیرش زبانهای نسل چهارم و پنجم.
- امکانات تولید برنامه‌های کاربردی.
- امکانات پشتیبانی تصمیم.
- امکانات لازم برای تولید واسطه‌های کاربری.
- امکانات عملیاتی در محیط سیستم "توزیع شده".
- تسهیلات نرم افزاری دیگر.

۷- روش مطالعه سیستم

در مطالعه این نرم‌افزار، به منظور کسب آشنایی مقدماتی با آن (ونه چندان تخصصی)، باید موارد زیر بررسی شود:

- بررسی شرکت سازنده، خانواده نرم‌افزارهای پایگاهی مشابه و تاریخچه سیستم
- حداقل پیکره بندی سخت‌افزاری و نرم‌افزاری لازم
- بررسی وجود اجزاء معماری ANSI برای پایگاه داده‌ها
- امکانات زبان داده‌ای فرعی سیستم
- امکانات سیستم از نظر زبان میزبان
- بررسی مولفه‌های مدل داده‌ای و میزان رابطه‌ای بودن سیستم
- اجزاء تشکیل دهنده سیستم
- روند کلی اجرای برنامه توسط سیستم
- نحوه کار با سیستم، برپاسازی، راه‌اندازی و آماده سازی سیستم، ورود به سیستم و کارهای لازم برای ایجاد پایگاه و انجام عملیات در آن
- تسهیلات جانبی سیستم از جمله واسطه‌های کاربری و....

۸- رویه‌های مستند برای کاربران

برای اینکه کاربران و اعضاء تیم مدیریت پایگاه داده‌ها بتوانند از سیستم استفاده کنند، معمولاً مجموعه‌ای از دستورهای و قواعد، موسوم به رویه‌های مستند توسط عرضه کنندگان سیستم در اختیار خریداران قرار داده می‌شود. در این رویه‌های مستند چگونگی انجام فعالیتهای زیر مشخص شده است:

- برپاسازی سیستم
- طرز ارتباط با سیستم
- طرز استفاده از سیستم
- طرز استفاده از تسهیلات و امکانات آن
- تولید نسخه‌هایی از پایگاه داده‌ها

- طرز تشخیص عیبه‌های سخت‌افزاری و نرم‌افزاری و چگونگی رفع آنها و ترمیم پایگاه‌داده‌ها
- تغییر ساختار پایگاه‌داده‌ها (سازماندهی مجدد)
- تنظیم سیستم
- بهبود بخشیدن کارایی پایگاه‌داده‌ها
- تولید نسخه‌های پشتیبان

۹- هزینه‌ها

استفاده از تکنولوژی پایگاه‌داده‌ها هزینه‌هایی دارد. برخی از اقلام مهم‌تر هزینه عبارتند از:

- هزینه خرید نرم‌افزار اصلی (DBMS)
- هزینه آموزش نرم‌افزار اصلی
- هزینه نگهداری و بهره‌برداری از آن
- هزینه تبدیل سیستم ناپایگاهی به سیستم پایگاهی
- هزینه تهیه ابزارهای نرم‌افزاری دیگر
- هزینه آموزش امکانات نرم‌افزاری
- هزینه تهیه بسته‌های کاربردی
- هزینه آموزش بسته‌های کاربردی
- هزینه تهیه مستندات خود سیستم
- هزینه تهیه مستندات امکانات نرم‌افزاری و بسته‌های کاربردی
- هزینه تنظیم مستندات سیستم پایگاه‌داده‌ها
- هزینه تهیه سخت‌افزار پردازشگر (کامپیوتر از رده‌های مختلف)
- هزینه تهیه سخت‌افزار ذخیره‌سازی
- هزینه تامین شبکه‌های لازم (در صورت لزوم)
- هزینه نگهداری و بهره‌برداری از سیستم کاربردی
- هزینه بهینه‌سازی و گسترش سیستم
- حقوق و مزایای افراد تیم مدیریت و تیم‌های اجرایی

۱۰- مدیر پایگاه داده (DBA)

مدیر پایگاه داده‌ها فردی است متخصص در پایگاه داده‌ها و با مسئولیت علمی-فنی و نیز اداری در محدوده وظایفی که عهده دار است. این مدیر معمولاً همراه با یک تیم تخصصی کار می‌کند که به آن تیم مدیریت پایگاه داده‌ها می‌گویند. هر یک از اعضا این تیم مسئولیت خاصی دارد و در حیطه اختیارات و وظایفش می‌تواند سرپرست یک تیم اجرایی باشد.

۱-۱۰ اصطلاح تیم DBA

در یک محیط کاری برخوردار از سطح مطلوب دانش و تکنولوژی و عمل کننده بر اساس دیسپلین‌ها و استانداردهای علمی و مهندسی و دارای مدیریت پویا، وجود این تیم تخصصی اجتناب ناپذیر است. بعلاوه این تیم باید از مشاورانی در زمینه‌های دیگر مهندسی نرم‌افزار و مهندسی سخت‌افزار استفاده کند و حتی مطلوب این است که بعضی از مشاوران به نحوی عضو خود تیم باشند. برخی از مسئولیت‌های اصلی در این تیم تخصصی عبارتند از:

- مدیر پایگاه داده‌ها
- مدیر داده‌ها
- مدیر امور پژوهشی- توسعه
- مدیر سیستم (های) کاربردی
- مسئول تیمهای برنامه سازی
- مسئول کنترل کارایی DBMS
- مسئول کنترل کارایی خود سیستم پایگاهی
- مسئول نظارت بر عملیات روی پایگاه داده‌ها و انجام فعالیت‌های آماری مربوطه
- مسئول تماس با کاربران زیر محیط‌های سازمان
- مسئول تنظیم مستندات

اصطلاح تیم DBA به دو معنا (در محیط‌های کاری) مطرح است:

- **تیم DBA در معنای محدود:** با این معنا، تیم DBA تیمی است که پایگاه داده‌های سازمان را، پس از ایجاد توسط گروه دیگری از متخصصین، تحویل می‌گیرد و پس از تحویل، وظیفه نگهداری، بهره برداری و گاه بهینه سازی و احتمالاً گسترش "سیستم" را بر عهده دارد.
- **تیم DBA در معنای گسترده:** با این معنا، تیم DBA، خود همه مراحل لازم برای ایجاد پایگاه داده‌های سازمان را انجام می‌دهد و سپس نگهداری، بهره برداری و بهینه سازی و گسترش آن را بر عهده می‌گیرد.

۱۰-۲ مسئولیت‌ها

مدیر داده‌ها فردی است با دانش و تجربه در مدیریت و آشنا با دانش و تکنولوژی پایگاه داده‌ها. برای اطلاع از وظایف مدیر داده‌ها به منابع درس‌های "مهندسی نرم‌افزار"، "تحلیل و طراحی سیستم‌ها" و نیز درس "سیستم اطلاعات مدیریت" مراجعه شود.

۱۰-۳ وظایف

تیم DBA در معنای گسترده وظایفی دارد که اهم آنها را ذکر می‌کنیم. توجه داریم که ترتیب انجام این وظایف لزوماً همان نیست که در زیر آمده است (روشن است که در اجرای هر پروژه، نیاز به یک طرح زمان بندی فعالیتهاست که باید آماده شود). همچنین برخی از این وظایف می‌توانند بطور همزمان انجام شوند.

- مشارکت در تفهیم اهمیت و نقش داده به مدیریت سازمان
- مشارکت در تفهیم اهمیت و مزایای تکنولوژی پایگاه داده‌ها.
- مشارکت در تصمیم گیری در مورد استفاده یا عدم استفاده از این تکنولوژی.
- مشارکت در توجیه علمی-فنی تصمیم استفاده از این تکنولوژی.
- مطالعه دقیق و همه جانبه محیط عملیاتی و برآورد خواسته‌ها و بر آورد نیازهای کاربران (انجام اصولی مهندسی نیازها).

- بررسی روند داده‌ها و روند رویدادها در محیط و رسم نمودار روند داده‌ها و روند رویدادها (یک یا هر دو بسته به شیوه مدلسازی سیستم مورد نظر) و تهیه و تنظیم مستندات لازم.
- مدلسازی معنایی داده‌ها (با مراحل‌ی که دیده شد از جمله رسم نمودار EER).
- تخمین حجم داده‌های ذخیره شدنی در پایگاه داده‌ها.
- تصمیم‌گیری دو مورد تعیین معماری سیستم پایگاه داده‌ها و تعیین مشخصات سیستم (های) کاربردی مورد نیاز.
- مشارکت در انتخاب DBMS (ها) و پیکر بندی سخت‌افزاری و نرم‌افزاری لازم در صورت لزوم (اگر انتخاب نشده باشد).
- تصمیم‌گیری در انتخاب و انتساب اعضاء تیمهای اجرایی.
- تصمیم‌گیری در انتخاب ابزارهای نرم‌افزاری دیگر برای تولید و گسترش سیستم مورد نظر.
- تصمیم‌گیری در مورد زبان (های) برنامه‌سازی مورد نیاز و متناسب با هر کاربرد.
- طراحی سطح ادراکی پایگاه داده‌ها (طراحی منطقی).
- نوشتن شمای ادراکی و برنامه‌های لازم برای ایجاد پایگاه داده‌ها.
- تعیین مجموعه قواعد (محدودیت‌های) جامعیت ناظر به پایگاه داده‌ها.
- نظارت بر تعیین دیدهای خارجی و نوشتن شمهای خارجی.
- تصمیم‌گیری در مورد مشخصات ساختار سطح داخلی پایگاه داده‌ها و تعیین ساختار فایل‌های مناسب، استراتژی‌های دستیابی کارا و نوشتن شمای داخلی.
- انجام طراحی توزیع (در صورت معماری توزیع شده).
- طراحی "برنامه‌های کاربردی"، تراکنش‌های لازم و رویه‌های عملیاتی لازم. (توجه داشته باشیم که هر برنامه‌ای، نیاز به طراحی دارد و تنها پس از طراحی اصولی برنامه، می‌توان برنامه‌سازی کرد)، ایجاد ارتباط دائم با تولید کنندگان "سیستم کاربردی".
- طراحی واسطه‌های کاربردی.
- ایجاد نمونه نخست سیستم پایگاهی و بارگذاری پایگاه با داده‌های تستی.

- نوشتن برنامه‌های لازم برای کنترل پایگاه‌داده‌ها بویژه اعمال محدودیتهای جامعیتی.
- نوشتن برنامه‌های لازم برای بهره برداری از پایگاه‌داده‌ها.
- ایجاد سیستم پایگاهی واقعی (و منطبق با نیازهای کاربران).
- نظارت بر وارد کردن داده‌ها (در حجم محدود).
- انتخاب استراتژی‌های تست مناسب و تست کردن "سیستم" با داده‌های تستی و نیز با داده‌های واقعی در حجم محدود (انجام دو مرحله تست).
- نظارت بر وارد کردن داده‌های واقعی سازمان.
- تست کردن "سیستم" با داده‌های واقعی و در حجم واقعی
- تنظیم دقیق قسمتهای مختلف سیستم و کل سیستم یکپارچه.
- تعیین ضوابط دستیابی کاربران به داده‌ها.
- نظارت در تهیه مستندات لازم در همه مراحل کار.
- وضع استانداردهای لازم در همه مراحل کار و نظارت بر اعمال آنها.
- تصمیم گیری در مورد چگونگی ترمیم پایگاه‌داده‌ها و در صورت لزوم تهیه یا توسعه ابزارهای این کار و انجام ترمیم پایگاه.
- کنترل مداوم کارایی DBMS و تلاش در افزایش کارایی.
- نظارت و کنترل دائم بر عملیاتی که در پایگاه‌داده‌ها انجام می‌شود.
- کنترل جامعیت پایگاه‌داده‌ها.
- تضمین محرمانگی داده‌ها.
- تصمیم گیری در مورد چندی و چگونگی رشد (گسترش) پایگاه‌داده‌ها.
- اتخاذ تدابیر لازم برای ایمنی و حفاظت داده‌ها و اعمال این تدابیر.
- مدیریت کاربران پایانی (ایجاد و تعریف کاربران، گذر واژه‌ها، امتیازها و...).
- تماس دائم با کاربران و شناخت نیازهای جدید آنها.
- تولید نسخه‌های پشتیبان بطور متناوب (با تناوب مناسب).
- تعیین الگوهای استفاده از داده‌ها و بسامد (فرکانس) استفاده از داده‌ها.
- تصمیم گیری در مورد چگونگی سازماندهی مجدد پایگاه‌داده‌ها.

- انجام تبدیل و انتقال داده‌ها از "سیستم‌های موجود" به پایگاه‌داده‌های جدید و انجام تبدیل برنامه‌های کاربردی موجود به گونه‌ای که قابل اجرا در "سیستم کاربردی" جدید باشند.
- تلاش در جهت ارتقاء سطح دانش و فن اعضاء تیم و کاربران (بویژه در زمینه تکنولوژی اطلاعات و سیستم‌های اطلاعاتی).
- تلاش در جهت شناسایی امکانات جدید، گسترش، ارتقاء و کارا تر کردن سیستم با استفاده از این امکانات.
- تهیه و تنظیم انواع آمارها و گزارشات کنترلی و مدیریتی در مورد سیستم پایگاه‌داده‌ها و کاربران.
- تضمین انجام و اتمام "پروژه پایگاهی" در مدت زمان پیش بینی شده و با توجه به محدودیت بودجه.

تمرینات

۱. با توجه به مطالعه گفتار، وظایف اساسی یک DBMS را فهرست کنید.
۲. برای طراحی و تولید یک RDBMS، حداقل چه اجزایی لازم است؟
۳. یک RDBMS را انتخاب و کارهای لازم برای برپاسازی آن را انجام دهید.
۴. در انتخاب و استفاده از یک DBMS، چه هزینه‌های دیگری را باید در نظر داشت (غیر از هزینه‌های برشمرده در گفتار)؟
۵. چه تفاوت (هایی) بین متا داده‌ها و دیکشنری داده‌ها وجود دارد؟
۶. پارامترهای اصلی شناخت یک DBMS را نام ببرید؟
۷. مدیر پایگاه داده را تعریف کنید؟
۸. تیم DBA به چه معنا است؟
۹. کدامیک از وظایف DBA، مستقل از DBMS، قابل انجام است؟
۱۰. کدامیک از موارد بر شمرده در فهرست وظایف تیم DBA، مستقیماً به مرحله پیاده سازی پایگاه داده‌ها مربوط می‌شود؟

فصل ۵

مدل‌ها و ساختارهای داده‌ای پایگاه‌داده‌ها

هدف کلی

در این فصل ابتدا با مفهوم محیط انتزاعی یک پایگاه‌داده و بخش‌های مختلف آن آشنا خواهیم شد. در ادامه سه ساختار داده‌ای اصلی پایگاه‌های داده که شامل ساختار داده‌ای سلسله‌مراتبی، شبکه‌ای و رابطه‌ای می‌باشند را معرفی کرده و عناصر اصلی، عملیات اصلی و نحوه نمایش ارتباط موجودیت‌ها و ویژگی‌های هر یک از ساختارهای داده‌ای شرح داده خواهند شد.

هدف رفتاری

در این فصل عناوین زیر مورد بحث و بررسی قرار می‌گیرند:

- محیط انتزاعی
- بخش ساختاری محیط انتزاعی
- بخش عملیاتی محیط انتزاعی
- بخش جامعیتی محیط انتزاعی
- آشنایی با ساختار داده‌ای سلسله‌مراتبی
- عناصر ساختاری
- طراحی پایگاه سلسله‌مراتبی
- عملیات در پایگاه سلسله‌مراتبی
- برخی ویژگی‌های ساختار (و مدل) داده‌ای سلسله‌مراتبی

- آشنایی با ساختار داده‌ای شبکه‌ای
- تعریف ساختار شبکه
- عناصر ساختاری
- طراحی پایگاه شبکه‌ای
- طرز نمایش ارتباط "یک به چند"
- طرز نمایش ارتباط بازگشتی
- طرز نمایش ارتباط "چند به چند"
- عملیات در پایگاه شبکه‌ای
- برخی ویژگی‌های ساختار (و مدل) داده‌ای شبکه‌ای

۱- محیط انتزاعی^۱

در مفاهیم پایگاه داده‌ها منظور از محیط انتزاعی، محیطی است که مابین سطح داخلی (محیط فایلینگ فیزیکی و فایلینگ منطقی) و سطح ادراکی است. طبیعتاً این محیط باید از سطح فایلینگ مستقل عمل کند. این محیط منطقی خود می‌تواند دارای چند سطح باشد. شکل ۵-۱ نمای کلی از محیط انتزاعی را نشان می‌دهد.

همانطور که در مدل سازی داده‌ها نیاز به امکانی برای نمایش واقعیات داریم، برای طراحی منطقی پایگاه داده‌ها نیز نیاز به یک مدل داده‌ای شامل یک ساختار داده‌ای داریم. مدل داده‌ای امکانی است برای طراحی منطقی پایگاه داده‌ها، تعریف و کنترل آن و انجام عملیات در آن. کاربر با استفاده از مدل داده‌ای می‌تواند هر سه عمل مذکور را در یک محیط انتزاعی انجام دهد. بنابراین می‌توان گفت که مدل داده‌ای تامین کننده محیط انتزاعی پایگاه داده‌ها است و از سه بخش اساسی تشکیل شده است:

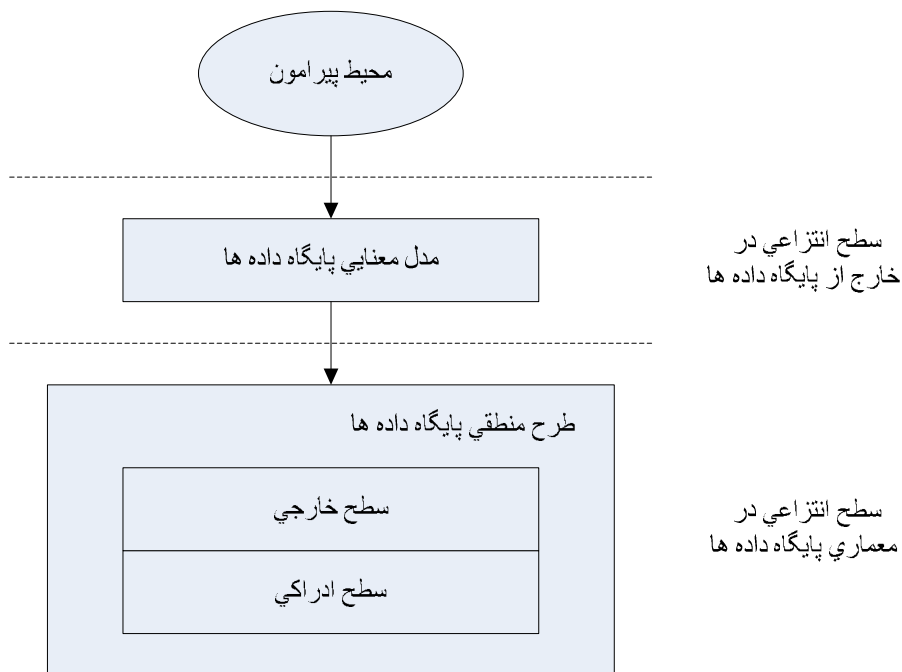
- بخش ساختاری^۲
- بخش عملیاتی^۳
- بخش جامعیتی^۴

1. Abstract Environment
2. Structural
3. Manipulative
4. Integrity

بخش ساختاری، شامل عناصر ساختاری مدل است که همان ساختار داده‌ای اصلی و مفاهیم مربوط به آن است.

بخش عملیاتی، مجموعه امکاناتی است که به وسیله آنها عملیات مورد نظر کاربر در محدوده ساختار داده‌ای و مبتنی بر عنصر ساختاری اساسی آن انجام می‌شود.

بخش جامعیتی، از مجموعه‌ای از قواعد (محدودیت‌های) جامعیتی تشکیل شده است که با استفاده از آنها سیستم مدیریت پایگاه داده‌ها می‌تواند صحت، دقت و سازگاری داده‌ها را در هر لحظه از حیات پایگاه داده، کنترل و تضمین کند.



شکل ۵-۱ سطوح مختلف محیط انتزاعی

با این توصیف هر ساختار داده‌ای نیز فقط بخشی از یک مدل داده‌ای است و حداقل یک عنصر ساختار اساسی دارد که به کمک آن نوع موجودیت یا نوع ارتباط و یا هر دو نمایش داده می‌شوند. در ادامه دلایلی که استفاده از ساختار داده‌ای را بیان می‌کند ارائه شده است:

- ساختار داده‌ای تامین کننده محیط انتزاعی پایگاه داده‌ها و چارچوب طراحی منطقی پایگاه است.
- ساختار داده‌ای مبنا (و چارچوب) طراحی دستورات تعریف داده‌ها، دستورهای عملیات در پایگاه داده‌ها و دستورهای کنترل داده‌ها است.
- ساختار داده‌ای مبنای طراحی DBMSهاست، یعنی با توجه به سه نوع مدل، سه رده DBMS داریم که عبارتند از: RDBMS، HDBMS و NDBMS، به نحوی که هویت "هر سیستم مدیریت پایگاه داده‌ها" همان مدل داده‌ای آن است.
- ساختار داده‌ای ضابطه‌ای است برای مقایسه DBMSها و نیز ارزیابی آنها.
- ساختار داده‌ای مبنایی است برای ایجاد و گسترش تکنیکهای طراحی پایگاه داده‌ها.
- ساختار داده‌ای، مبنای پژوهش در دانش و تکنولوژی پایگاه داده‌ها است.

پایگاه داده‌ها در محیط انتزاعی، مجموعه‌ای است از نمونه‌های متمایز عناصر ساختاری یک ساختار داده‌ای. در مفاهیم پایگاه داده، نمونه‌های کلاسیک ساختار داده‌ای عبارتند از:

- ساختار داده‌ای سلسله مراتبی^۱
- ساختار داده‌ای شبکه^۲
- ساختار داده‌ای رابطه‌ای^۳

با توجه به اینکه ساختارهای داده‌ای سلسله مراتبی و شبکه‌ای از رده خارج شده‌اند، لذا این دو ساختار جهت آشنایی خوانندگان محترم بصورت کلی توضیح داده خواهد شد. ساختار داده‌ای رابطه‌ای در فصل‌های بعدی بصورت کامل مورد بحث و بررسی قرار خواهد گرفت.

1. Hierarchical Data Structure

2. Network Data Structure

3. Relational Data Structure

۲- آشنایی با ساختار داده‌ای سلسله مراتبی

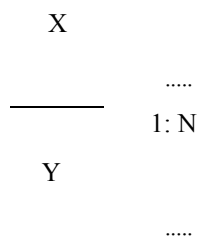
این ساختار قدیمی‌ترین ساختار داده‌ای برای طراحی منطقی پایگاه داده‌ها (در سطح انتزاعی) است. انسان از دیرباز با مفهوم سلسله مراتب آشنا بود و آن را برای رده بندی پدیده‌ها (در معنای عام) در جهان واقع، استفاده می‌کرد. در اواسط دهه ۱۹۶۰ میلادی، وقتی که طراحان سیستم‌های نگهداری داده- داده پردازی دراندیشه یافتن نوعی "رکورد ساختمند" بودند (به جای "رکورد خطی مسطح") همین مفهوم آشنای "سلسله مراتب" را برای نمایش داده‌های مورد نظرشان، مناسب یافتند.

۲-۱ عناصر ساختاری

سلسله مراتب در اساس نوعی درخت است که مفهوم آشنایی است. در ساختار داده‌ای سلسله مراتبی، دو عنصر ساختاری اساسی وجود دارد:

- نوع رکورد
- نوع پیوند پدر-فرزندی

بطوریکه خواهیم دید، نوع رکورد برای نمایش نوع موجودیت به کار می‌رود. بین هر دو نوع رکورد بلافصل از یک مسیر سلسله مراتب، پیوند پدر-فرزندی وجود دارد و با این پیوند ارتباط بین دو نوع موجودیت نمایش داده می‌شود. این نوع پیوند، طبق ماهیت مفهوم سلسله مراتب، ارتباط یک به چند را می‌تواند نمایش دهد. در سوی "یک" ارتباط (۱)، نوع رکورد پدر و در سوی "چند" (N) آن، نوع رکورد فرزند قرار دارد.

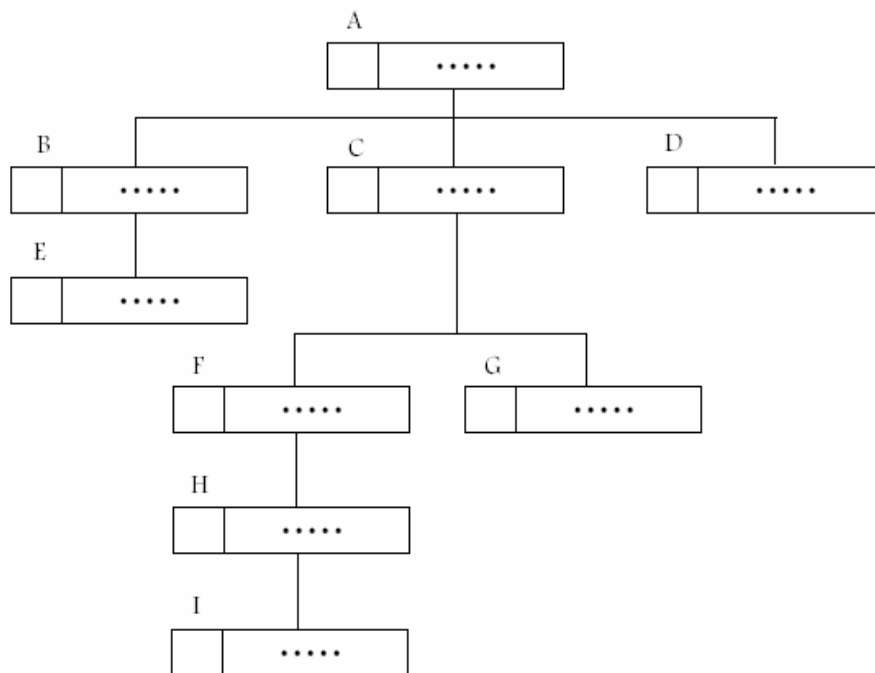


شکل ۲-۵ مفهوم سلسله مراتب

در شکل ۵-۲، رکورد نوع Y فرزند و رکورد نوع X، پدر است. هر نوع رکورد (از هر سطح سلسله مراتب)، می‌تواند از صفر تا n نوع رکورد فرزند (در سطح بلافاصله پائین تر و در مسیرهای مختلف طبعاً) داشته باشد. پس هر نوع رکورد فرزند از یک سطح، خود می‌تواند پدر انواعی از رکوردها در سطح بلافاصله پائین تر و در چند مسیر باشد. بدینسان، سلسله مراتبی از انواع رکوردها در سطوح و مسیرهای مختلف ایجاد می‌شود.

- ریشه سلسله مراتب، رکورد نوع پدر در بالاترین سطح است. بقیه انواع رکوردها، وابستگان ریشه در سطوح مختلف، هستند.
- ریشه در حالت خاص ممکن است اصلاً فرزند نداشته باشد و در این صورت سلسله مراتب را "فقط ریشه" می‌گوییم.

برای مثال یک نوع سلسله مراتب با تعداد ۹ نوع رکورد در شکل ۵-۳ نشان داده شده است.



شکل ۵-۳ نمایش ارتباطی نه نوع رکورد در ساختار سلسله مراتبی

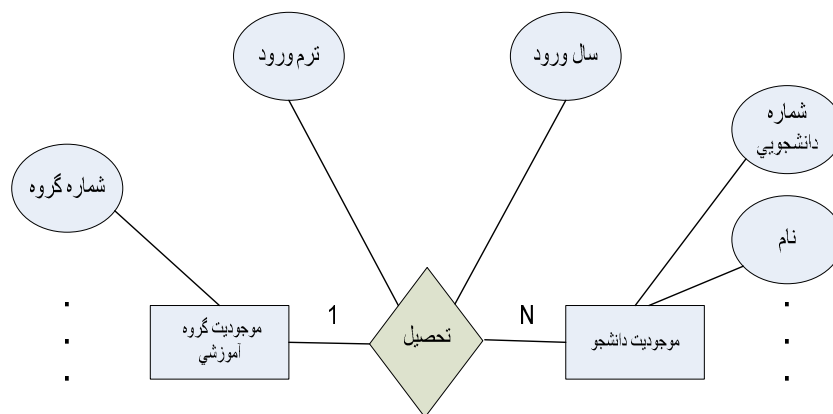
لازم به ذکر است که ترتیب پر شدن زیر مجموعه‌ها بر اساس نیاز اطلاعاتی بوده و هیچ نوع نظم خاصی را ندارد. همچنین لازم بذکر است که هر نوع رکورد فرزند، تنها یک نوع رکورد پدر دارد.

توجه: یک نوع سلسله مراتب، از لحاظ نظری می‌تواند هر تعداد مسیر داشته باشد (عرض) و عمق هر مسیر هم محدودیتی ندارد. اما در سیستم سلسله مراتبی عرض و عمق یک نوع سلسله مراتب معمولاً یک سقف حداکثر (مثلاً ۱۶ سطح) دارد.

۲-۲ طراحی پایگاه سلسله مراتبی

در سطح انتزاعی (از دید کاربر) پایگاه داده سلسله مراتبی، مجموعه‌ای است منطقاً "منظم" از نمونه‌های متمایز یک یا بیش از یک نوع سلسله مراتب. (روشن است که در سطح فیزیکی، نهایتاً مجموعه‌ای است از فایل‌های ذخیره شده بهم مرتبط بر اساس همان پیوندهای منطقی و با ساختار مشخص).

لازم به ذکر است که در هر نمونه سلسله مراتب، یک نمونه متمایز از ریشه داریم (عامل تمایز، کلید ریشه است). هر نمونه از یک رکورد از یک سطح می‌تواند از صفر تا n نمونه از هر نوع رکورد فرزند بلافصل، داشته باشد. نمونه‌های هر نوع رکورد فرزند ذیل یک نمونه پدر مشخص هم از یکدیگر متمایزند. عامل تمایز، همان کلید رکورد فرزند است. به عنوان اولین مثال، نمودار ER شکل ۵-۴ را در نظر می‌گیریم. این نمودار نوع ارتباط بین موجودیت‌ها دانشجو، و گروه آموزشی را نشان می‌دهد:



شکل ۵-۴ نمودار ER مثال ۱

فرض می‌کنیم که یک دانشجو فقط در یک گروه آموزشی تحصیل می‌کند (وابسته به یک گروه آموزشی است). چندی ارتباط "تحصیل کردن" در این مثال، یک به چند است، بنابراین یک نوع سلسله مراتب به صورت شکل ۵-۵ طراحی می‌کنیم:

DEPT

DID	
-----	-------

STUDENT

STID		TERM	YEAR
------	-------	------	------

شکل ۵-۵ سلسله مراتب نوع DEST

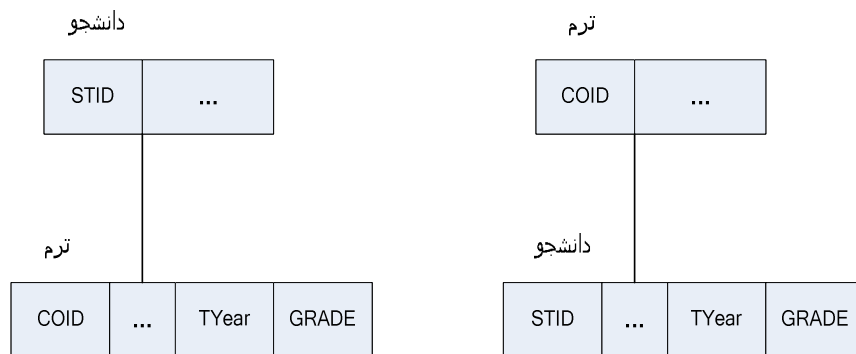
در این طرح ساده، رکورد نوع DEPT ریشه- پدر و رکورد نوع STUD وابسته بلافاصله (فرزند) است. فقط فیلد صفت شناسه دو نوع موجودیت رانشان داده‌ایم. صفت ارتباط، فیلدی از نوع رکورد فرزند می‌شود. پس پایگاه داده سلسله مراتبی مجموعه‌ای است منطقاً منظم از نمونه‌های متمایز یک یا بیش از یک نوع سلسله مراتب.

مثال ۲: در مثال قبل، چندی ارتباط "یک به چند" است. در این مثال، یک ارتباط با چندی "چند به چند" را مطرح می‌کنیم.

نمودار "دانشجو- درس" را در نظر می‌گیریم. می‌دانیم که چندی ارتباط "انتخاب"، چند به چند است. اگر ارتباط چند به چند را ارتباط یک به چند دو سویه بدانیم برای نمایش این ارتباط با عناصر ساختار داده‌ای سلسله مراتبی، منطقاً به دو نوع سلسله مراتب نیاز داریم. دو روش برای طراحی وجود دارد:

روش ۱: طراحی دو نوع سلسله مراتب جداگانه

در این طراحی، یک نمونه رکورد فرزند، می‌تواند تکرار شود: مثلاً در سلسله مراتب نوع STCO، یک نمونه رکورد نوع درس ذیل هر نمونه رکورد دانشجو که آن درس انتخاب کرده باشد، می‌آید. چنین است برای رکورد نوع STUD در سلسله مراتب نوع COST.



شکل ۵-۶ نمایش دو نوع سلسله مراتب جداگانه برای ارتباط چند به چند

این تکرار، در واقع همان پدیده نامطلوب افزونگی است. این افزونگی البته ماهیتاً منطقی یا ادراکی است. اما اگر منجر به بروز افزونگی در محیط فیزیکی ذخیره سازی شود می‌تواند:

- سبب بروز ناسازگاری داده‌ای شود (به یاد داشته باشیم که افزونگی کنترل نشده و ناسازگاری داده‌ای پشت و روی یک سکه‌اند!).
- سبب بروز فزونکاری (بیشکاری) در سیستم در انجام عملیات در پایگاه داده‌ها شود.

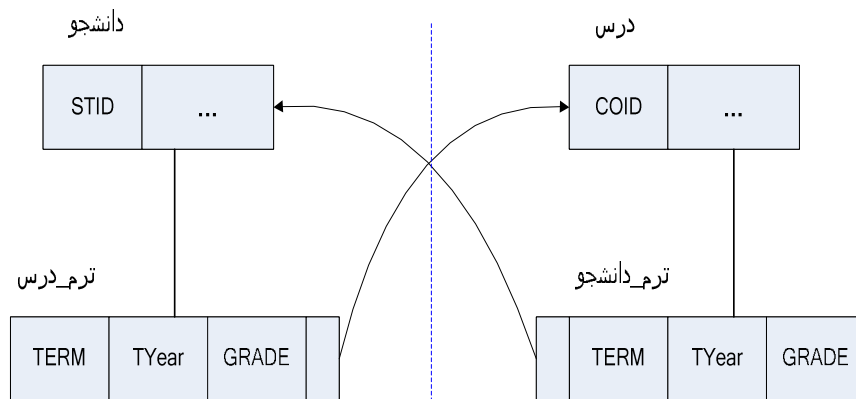
توجه: می‌توان ارتباط با چندی $M:N$ را با یک نوع سلسله مراتب هم طراحی کرد، مثلاً فقط نوع سلسله مراتب $STCO$ را داشت. اما چون منطق طبیعی در سلسله مراتب، با توجه به ماهیت پیوند پدر-فرزندی، ورود از ریشه و طی کردن مسیر مناسب برای رسیدن به نمونه فرزند مورد نظر است، برای پاسخگویی به پرسشی که در آن کلید نمونه فرزند داده شده باشد و نمونه (های) پدر مورد درخواست باشد، دیگر نمی‌توان منطقاً بر اساس رویه مبتنی بر پیوند پدر-فرزندی عمل کرد. بعلاوه با داشتن فقط یک نوع سلسله مراتب برای ارتباط $M:N$ و (عدم وجود نوع سلسله مراتب قرینه)، مشکلاتی ناشی از ماهیت پیوند پدر-فرزندی در عملیات ذخیره سازی در پایگاه داده‌ها بروز می‌کند (از جمله عدم امکان درج نمونه فرزند بدون وجود نمونه پدر و یا لزوم انجام بهنگام سازی منتشر شونده)

روش ۲: طراحی دو نوع سلسله مراتب به هم بسته: تکنیک VPCL^۱

در این روش، از یک نوع رکورد مجازی (حاوی نشان نما: اشاره گر و صفات ارتباط) استفاده می‌شود. به این نوع رکورد اصطلاحاً "فرزند مجازی" گفته می‌شود. هر نمونه از فرزند مجازی به یک نمونه "پدر مجازی" اشاره می‌کند. چنین پیوندی را پیوند پدر-فرزندی مجازی می‌نامیم. در شکل ۷-۵ این روش دیده می‌شود.

در این طرح نوع رکورد COPTR، فرزند مجازی نوع رکورد COURSE و این نوع رکورد، پدر مجازی نوع رکورد COPTR است و نوع رکورد STPTR، فرزند مجازی نوع رکورد STUD و این نوع رکورد، پدر مجازی نوع رکورد STPTR است. اطلاعاتی که هم به پدر و هم به فرزند مربوط می‌شود (صفات ارتباط)، در نوع رکورد اشاره گر مجازی گنجانده می‌شود: در اینجا صفات TR، YRYR و GRADE.

در این طراحی، نمونه‌های دو نوع رکورد COURSE و STUD دیگر تکرار نمی‌شوند. این کاهش افزونگی، البته به قیمت پیچیده‌تر شدن طراحی و پیاده سازی پایگاه داده‌ها تمام می‌شود (به ویژه اگر سلسله مراتب مورد نیاز، بزرگ باشد (در عرض و عمق)). شایان ذکر است که از دیدگاه نظری، این تکنیک ربطی به ماهیت ساختار داده‌ای سلسله مراتبی ندارد، بلکه امکانی است برای نمایش بهم بستگی دو یا بیش از دو نوع سلسله مراتب و از آنجا، امکانی است برای نمایش ارتباط چند به چند.

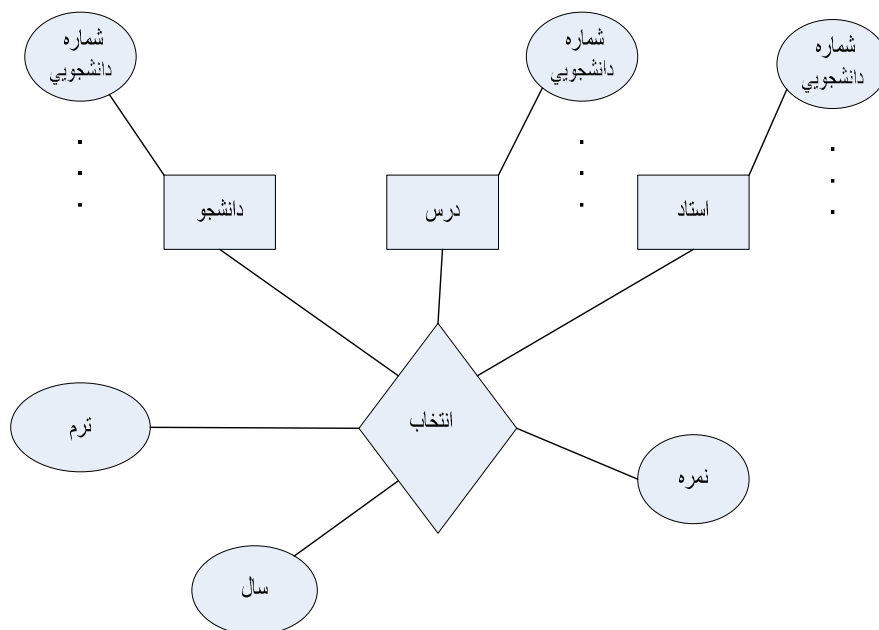


شکل ۷-۵ پیوند پدر-فرزندی مجازی

1. Virtual Parent Child Link

توجه داشته باشیم که PCL و VPCL از نظر مفهومی، یکسان هستند، تفاوت در طرز پیاده سازی آنها است: PCL با استفاده از توالی سلسله مراتبی پیاده سازی می‌شود و VPCL با استفاده از نشانه رو فیزیکی حاوی آدرس و یا نشانه رو منطقی حاوی کلید، از رکورد فرزند مجازی به رکورد پدر مجازی. این تکنیک سبب افزایش کارایی سیستم در پاسخ گویی به بعض پرسشها می‌شود.

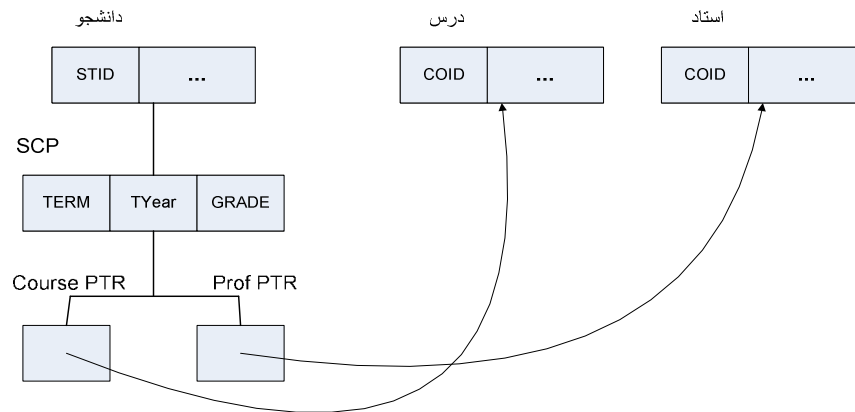
مثال ۳: در این مثال، یک ارتباط سه موجودیتی را در نظر می‌گیریم: ارتباط بین انواع موجودیت‌های دانشجو، درس و استاد که در بحث مدل سازی معنایی داده‌ها دیدیم. نمودار ER مربوطه در شکل ۵-۸ نشان داده شده است:



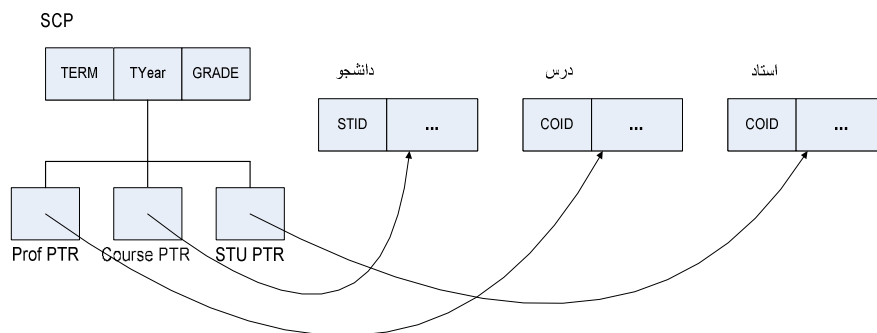
شکل ۵-۸ نمودار ER ارتباط بین سه موجودیت

برای تبدیل این نمودار به طراحی پایگاه سلسله مراتبی، دو روش ممکن به صورت زیر خواهند بود:

روش اول:



روش دوم:



شکل ۵-۹ روش‌های برای نمایش ارتباط سه موجودیتی با ساختار سلسله مراتبی

۲-۳ عملیات در پایگاه سلسله مراتبی

با توجه به عناصر ساختاری اساسی ساختار داده‌ای سلسله مراتبی، برخی از دستورات لازم، بطور کلی چنین‌اند:

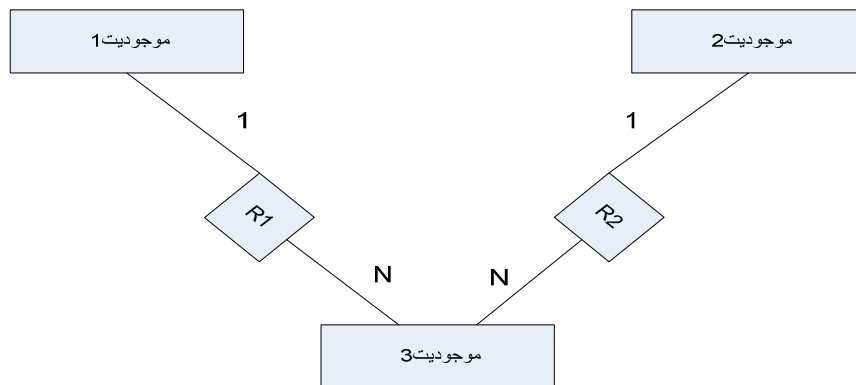
- دستور بازیابی یک نمونه رکورد نوع ریشه.
- دستور بازیابی نمونه ریشه بعدی.
- دستور بازیابی یک نمونه فرزند از یک نمونه پدر (بلافصل یا از نمونه پدران تا ریشه).

- دستور بازیابی نمونه فرزند بعدی از یک نمونه پدر.
- دستور درج یک نمونه فرزند زیر یک نمونه پدر (مشخصات مسیر، در صورت نیاز باید داده شود).
- دستور حذف یک (یا چند) نمونه فرزند از یک نمونه پدر (مشخصات مسیر، در صورت نیاز باید داده شود).
- دستور بهنگام سازی یک (یا چند) نمونه فرزند از یک نمونه پدر (مشخصات مسیر، در صورت نیاز باید داده شود).

۲-۴ برخی ویژگی‌های ساختار (و مدل) داده‌ای سلسله مراتبی

- در ادامه برخی ویژگی‌های ساختار داده‌ای سلسله مراتبی به اختصار بیان شده است:
- مبنای ریاضی ندارد (حداقل در سیستم موجود) و از این رو میزان انتزاع آن به اندازه انتزاع ساختار جدولی نیست (البته گراف یک سویه، بی‌گسست و نابازگشتی می‌تواند مبنای ریاضی مناسبی برای این ساختار داده‌ای باشد).
 - دو عنصر ساختاری اساسی دارد (و نه یکی).
 - برای نمایش ارتباط یک به چند مناسب است (محدودیت ساختار).
 - در عملیات ذخیره سازی مشکلاتی دارد.
 - در نمایش ارتباط "چند به چند" دشواری دارد و نیز در نمایش ارتباط سه گانی یا با درجه بیشتر و همچنین وقتی که دو یا بیش از دو ارتباط با چندی $1:N$ داشته باشیم که در آنها، در طرف N ارتباط، یک نوع موجودیت واحد وجود داشته باشد، به صورت شکل زیر.

لازم به ذکر است که در این جا دو نوع موجودیت $E1$ و $E2$ ممکن است یکسان هم باشند، به بیان دیگر دو یا بیش از دو نوع ارتباط بین دو نوع موجودیت وجود داشته باشد).



شکل ۱۰-۵ نمودار نوع ارتباط ۱ به چند

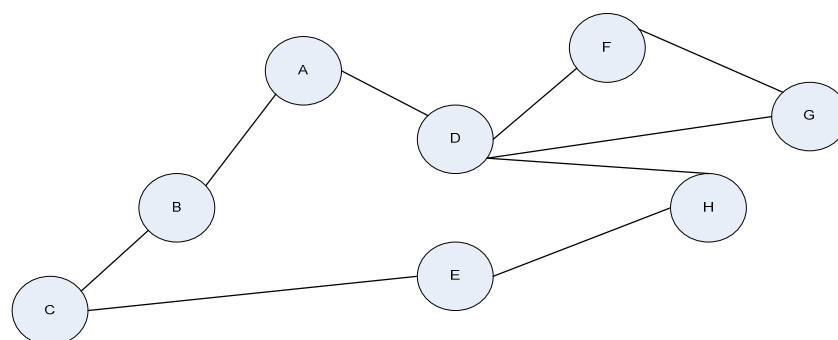
۳- آشنایی با ساختار داده‌ای شبکه‌ای

این ساختار (که گاه به آن ساختار پلکس هم می‌گویند)، نخستین بار در سال ۱۹۶۶ توسط یک گروه کاری به نام DBTG وابسته به ANSI پیشنهاد شده است. در سال ۱۹۷۱ اولین سیستم مدیریت پایگاه‌داده‌های شبکه‌ای (NDBMS) مورد تأیید ANSI، به نام IDMS توسط همان گروه کاری طراحی شد. این سیستم گاه به سیستم کوداسیل و مدل داده‌ای شبکه‌ای به مدل داده‌ای کوداسیل نیز موسوم است. سیستم‌های شبکه‌ای دیگر عبارتند از:

- VAX- DBMS
- IMAGE
- .1100-DMS

۳-۱ تعریف ساختار شبکه

شبکه نوعی گراف جهت دار است که در آن گره‌ها به کمک یال‌هایی بهم بسته‌اند. در شکل ۱۰-۵ یک شبکه با هشت گره دیده می‌شود. این ساختار را می‌توان گسترش یافته ساختار سلسله مراتبی دانست به این معنا که در آن هر نوع گره فرزند می‌تواند بیش از یک نوع گره پدر داشته باشد و بنابراین از این نظر، محدودیت عدم تقارن ساختار سلسله مراتبی را ندارد.



شکل ۵-۱۱ نمایش کلی ساختار شبکه

۲-۳ عناصر ساختاری

در ساختار داده‌ای شبکه‌ای دو عنصر ساختاری اساسی وجود دارد:

- نوع رکورد
- نوع مجموعه

نوع رکورد: برای نمایش نوع موجودیت به کار می‌رود (مثل ساختار سلسله مراتبی). در یک نوع رکورد طبعاً صفاتی وجود دارد که می‌توانند ساده تکرار شونده (بردار در اصطلاح شبکه‌ای) یا مرکب تکرار شونده (گروه تکرار شونده در اصطلاح شبکه‌ای) هم باشند.

نوع مجموعه: در اساس برای نمایش ارتباط 1:N بین دو (چند) نوع موجودیت بکار می‌رود. نوع مجموعه (که به آن مجموعه کوداسیل هم می‌گویند)، از سه جزء تشکیل شده است:

- نام مجموعه
- یک نوع رکورد مالک
- یک نوع رکورد عضو

هر چند این عنصر ساختاری در اساس برای نمایش ارتباط "یک به چند" بین دو یا بیش از دو نوع موجودیت بکار می‌رود، اما می‌توان ارتباط "چند به چند" را هم با آن نمایش داد. در اینجا لازم است به نکات زیر توجه گردد:

- مالک یک نوع مجموعه، می‌تواند مالک در نوع مجموعه دیگر باشد.
- مالک یک نوع مجموعه می‌تواند عضو در نوع مجموعه دیگر باشد.
- عضو یک نوع مجموعه می‌تواند مالک در نوع مجموعه دیگر باشد.
- عضو یک نوع مجموعه می‌تواند عضو در نوع مجموعه دیگر باشد. به این عضو، عضو مشترک یا پیوند دهنده می‌گوئیم و در نمایش ارتباط با چندی N:M بکار می‌آید.

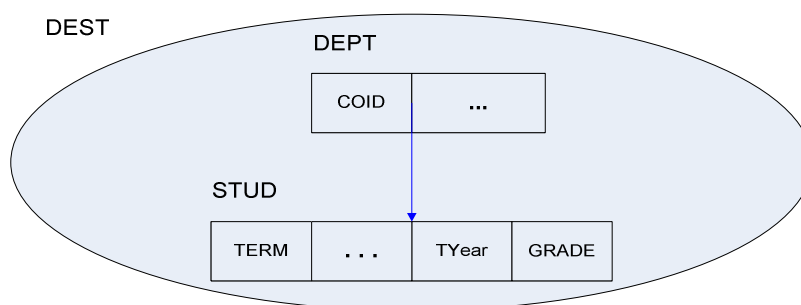
یک نوع مجموعه، می‌تواند نمونه‌هایی داشته باشد. در هر نمونه مجموعه، یک نمونه مشخص و متمایز از رکورد نوع مالک و $n \geq 0$ نمونه متمایز از رکورد نوع عضو وجود دارد. پایگاه‌داده‌های شبکه‌ای در محیط انتزاعی (از دید کاربر) مجموعه‌ای است منطقاً "منظم" از نمونه‌های یک (یا چند) نوع مجموعه.

۳-۳ طراحی پایگاه شبکه‌ای

پایگاه‌داده شبکه‌ای روش خاص خود جهت نمایش ارتباط بین موجودیت‌ها را دارا است. با توجه به اهمیت ارتباط‌های از نوع یک به چند و چند به چند، نحوه نمایش و پیاده سازی این دو نوع ارتباط در زیر توضیح داده شده‌اند.

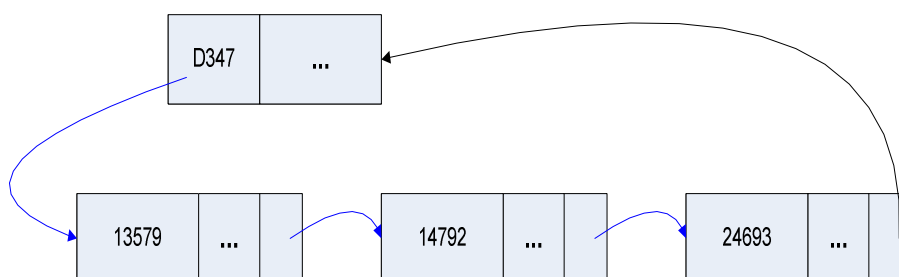
۳-۳-۱ طرز نمایش ارتباط "یک به چند"

در این نوع حالت نوع رکورد سوی "یک" را مالک و نوع رکورد سوی "چند" را عضو در نظر می‌گیریم.



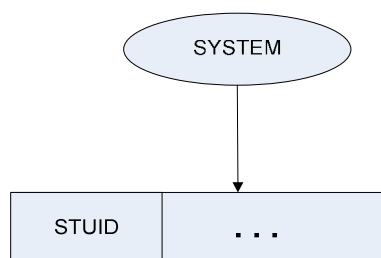
شکل ۵-۱۲ نوع مجموعه کوداسیل DEST

در این نمودار که به آن نمودار باخمن^۱ هم می‌گویند، نوع رکورد DEPT، مالک، و نوع رکورد STUD، عضو است. این نمودار یک نوع مجموعه کوداسیل به نام DEST را نمایش می‌دهد. می‌بینیم که در حالت ارتباط یک به چند، صفت (صفات) ارتباط، فیلد(هایی) از نوع رکورد عضو هستند. در شکل ۱۳-۵ نمونه‌ای از نوع مجموعه DEST دیده می‌شود.



شکل ۱۳-۵ دو نمونه از نوع مجموعه DEST

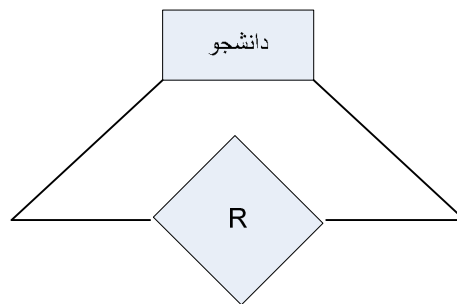
یک نوع مجموعه، می‌تواند چند نوع رکورد عضو داشته باشد که البته یک حالت خاص است و در سیستم‌های شبکه‌ای موجود پذیرفته نیست. رکورد نوع مالک در یک نوع مجموعه می‌تواند مالک در چند نوع مجموعه باشد. گونه خاصی از نوع مجموعه وجود دارد که رکورد نوع مالک ندارد و در واقع "سیستم" خود مالک آن است. شکل ۱۴-۵ نمونه‌ای از رکورد تحت مالکیت سیستم را نشان می‌دهد.



شکل ۱۴-۵ نوع رکورد تحت مالکیت "سیستم"

ارتباط بازگشتی

نوع مجموعه ممکن است بازگشتی باشد. در این گونه نوع مجموعه، یک نوع رکورد هم مالک است و هم عضو (این نوع مجموعه غالباً در سیستمهای شبکه‌ای مجاز نیست). شکل ۱۵-۵ ارتباط موجودیت E با خودش را نشان می‌دهد:



شکل ۱۵-۵ ارتباط موجودیت با خودش

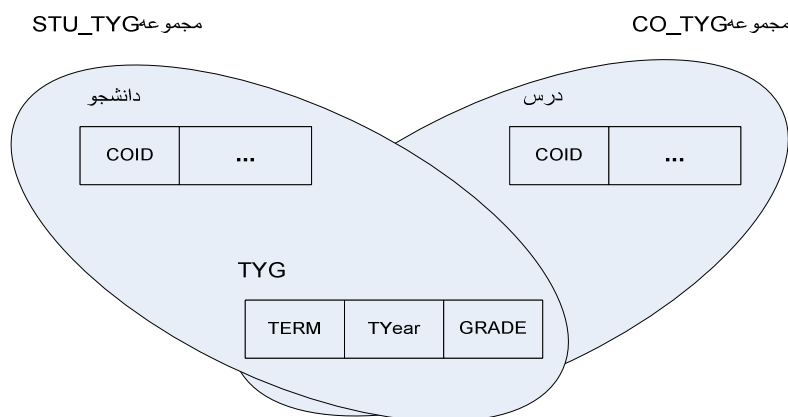
۳-۳-۲ طرز نمایش ارتباط "چند به چند"

برای نمایش ارتباط چند به چند، از یک نوع رکورد پیوند دهنده استفاده می‌کنیم که عضو مشترک بین دو یا بیش از دو نوع مجموعه است (تعداد نوع مجموعه‌ها همان تعداد نوع موجودیتهای شرکت کننده در ارتباط است. به بیان دیگر به تعداد درجه ارتباط، نوع مجموعه با عضو مشترک داریم). این نوع رکورد (که می‌تواند بدون یا دارای فیلد (های) داده‌ای باشد) را گاه رکورد ساختگی می‌نامیم. فیلد(های) این رکورد (در سطح طراحی)، همان صفت (صفات) ارتباط N:M است. برای نمونه در شکل ۱۶-۵ ارتباط "انتخاب" بین دانشجو و درس نشان داده شده است. این ارتباط به کمک دو نوع مجموعه نمایش داده شده است که در آنها رکورد TYGR رکورد پیوند دهنده است.

حال برای درک بهتر موضوع می‌خواهیم پایگاه داده شبکه‌ای را در سطح انتزاعی توصیف کنیم. پایگاه داده شبکه‌ای مجموعه ایست منطقاً منظم از نمونه‌های متمایز یک یا چند مجموعه کوداسیل. در اینجا ذکر این نکته اهمیت دارد که ساختار چند حلقه‌ای

مدل‌ها و ساختارهای داده‌ای پایگاه داده‌ها ۱۰۵

بویژه گونه‌های کاملترش می‌تواند ساختار مناسبی باشد و می‌توان این ساختار را با ساختار شاخص و یا ساختار مستقیم از طریق درهم‌سازی، ترکیب کرد.



شکل ۵-۱۶ نمایش ارتباط "چند به چند" بین دانشجو و درس

۳-۴ عملیات در پایگاه شبکه‌ای

با توجه به عناصر ساختاری ساختار داده‌ای شبکه‌ای، برخی دستورات بطور کلی عبارتند از:

- دستور بازیابی یک نمونه از یک نوع رکورد مالک
- دستور بازیابی نمونه بعدی از یک نوع رکورد مالک
- دستور بازیابی یک نمونه از یک نوع رکورد عضو با داشتن نمونه مالک (ها)
- دستور بازیابی نمونه بعدی از یک نوع رکورد عضو با داشتن نمونه مالک (ها)
- دستور بازیابی یک نمونه مالک از یک نوع مجموعه با داشتن یک نمونه عضو از آن

- دستور درج یک نمونه نوع رکورد مالک یا نوع رکورد عضو
- دستور حذف یک نمونه نوع رکورد مالک یا نوع رکورد عضو
- دستور بهنگام سازی یک نمونه نوع رکورد مالک یا نوع رکورد عضو

۳-۵ برخی ویژگی‌های ساختار (و مدل) داده‌ای شبکه‌ای

در ادامه برخی ویژگی‌ها و محدودیت‌های ساختار داده‌ای شبکه را شرح خواهیم داد:

- مبنای ریاضی ندارد (حداقل در سیستم موجود)، از این رو میزان انتزاع آن در حد ساختار داده‌ای رابطه‌ای (جدولی) نیست (البته گراف می‌تواند مبنای ریاضی مناسبی برای این ساختار داده‌ای باشد).
- دو نوع عنصر ساختاری اساسی دارد.
- ماهیتا خاص نمایش ارتباطات "یک به چند" نیست، یعنی محدودیتی در نمایش ارتباطات با چندی‌های دیگر، ندارد.
- ساخت منطقی رویه بازیابی آن پیچیده تر از ساختارهای دیگر است و ناوش غیر اتوماتیک است.
- خطر بروز ناسازگاری داده‌ها نسبت به ساختار سلسله مراتبی، کمتر است.
- قواعد جامعیت ذاتی دارد.
- بعضی آنومالی‌های مدل سلسله مراتبی در عملیات ذخیره سازی را ندارد.
- فزون کاری احتمالی ناشی از افزونگی که در ساختار سلسله مراتبی می‌تواند وجود داشته باشد، در این ساختار وجود ندارد اما به علت حجم زیاد نشان نماها (اشاره گرها)، ایجاد یا اصلاح آنها می‌تواند سبب بروز فزون کاری در سیستم شود.

تمرینات

۱. دلایل استفاده از ساختار داده‌ای را شرح دهید؟
۲. عناصر اصلی ساختار داده سلسله مراتبی را شرح دهید؟
۳. ویژگی‌ها و محدودیت‌های ساختار داده سلسله مراتبی را شرح دهید؟
۴. عناصر اصلی ساختار داده شبکه‌ای را شرح دهید؟
۵. ویژگی‌ها و محدودیت‌های ساختار داده شبکه‌ای را شرح دهید؟
۶. نحوه پیاده سازی ارتباط بازگشتی را در ساختار داده‌ای شبکه‌ای توضیح دهید؟

فصل ۶

پایگاه داده رابطه‌ای

هدف کلی

در این فصل ابتدا تاریخچه‌ای در مورد بانک اطلاعات رابطه‌ای بیان شده و ساختار داده‌ای رابطه‌ای به تفصیل مورد بحث و بررسی قرار خواهد گرفت. در این راستا مفهوم رابطه شرح داده شده و تناظری بین مفهوم رابطه و جدول بیان خواهد شد. سپس ویژگی‌های رابطه شرح داده شده و در مورد مفاهیمی مانند کلیدها، نوع رابطه و مفهوم دید صحبت خواهد شد. در ادامه مفهوم قواعد جامعیت مطرح شده و انواع قواعد جامعیت توضیح داده خواهند شد.

هدف رفتاری

در این فصل عناوین زیر مورد بحث و بررسی قرار می‌گیرند:

- بانک اطلاعات رابطه‌ای Relational DB
- ساختار داده‌ای رابطه‌ای
- تعریف رابطه
- تناظر بین مفاهیم رابطه‌ای و مفاهیم جدولی
- تعریف صوری جدول
- ویژگی‌های رابطه
- انواع کلید در مدل رابطه‌ای
- انواع رابطه

- مفهوم دید
- قواعد کاربری
- متا قواعد
- قواعد جامعیت در مدل رابطه‌ای

۱- بانک اطلاعات رابطه‌ای^۱

پس از پایگاه‌های داده‌ای سلسله مراتبی و شبکه‌ای، که هر یک دارای ضعف‌هایی بودند، متخصصان در جستجوی مدلی بودند که دارای ساختار داده‌ای با انتزاع قوی باشد. مدل رابطه‌ای در سال ۱۹۷۰ توسط E.F.Codd ابداع گردید. این مدل دارای ساختار داده‌ای با انتزاع قوی بوده و اساساً ساختار داده‌ای در آن بر اساس یک مفهوم ریاضی بنام **رابطه**^۲ استوار است. در اینجا لازم است به این نکته توجه شود که مفهوم رابطه از دیدگاه کاد با مفهوم ریاضی آن تاحدودی متفاوت است. در اینجا برای ارائه ساده‌تر موضوعات، در مباحث این کتاب مفهوم رابطه را با مفهوم ریاضی آن مساوی در نظر می‌گیریم.

۲- ساختار داده‌ای رابطه‌ای

برای طراحی پایگاه‌داده‌ها در سطح انتزاعی پایین تر از سطح مدل سازی، به یک ساختار داده‌ای از یک مدل داده‌ای نیاز است و اساساً همین مدل داده‌ای تامین کننده محیط انتزاعی است.

در پایگاه‌داده رابطه‌ای بالاخص در محیط انتزاعی مورد استفاده کاربر، رابطه نمایشی جدولی دارد و اساساً پایگاه‌داده رابطه‌ای مجموعه‌ای است از تعدادی نوع جدول. مفاهیم ساختار جدولی عبارتند از:

- جدول
- سطر

1. Relational Database (Data Bank)

2. Relation

• ستون

هر جدول از نظر محتوای داده‌ای مجموعه‌ای است از نمونه‌های متمایز از انواع سطرها و هر سطر نیز مجموعه‌ای از مقادیر است که هر کدام از یک مجموعه برگرفته شده‌اند. به هر یک از عناصر سطر یک ستون گویند. لازم به ذکر است که در ساختار جدولی، تنها عنصر ساختاری اساسی همین مفهوم نوع جدول است.

۳- تعریف رابطه

۳-۱ تعریف رابطه از دیدگاه کاد

با فرض وجود n مجموعه $S_1, S_2, \dots, S_n$ ، رابطه R تعریف شده روی این n مجموعه، مجموعه‌ایست از n تایی‌ها (که به آن تاپل^۱ می‌گویند) به نحوی که جزء اول هر n تایی از S_1 ، جزء دوم از S_2 و... مقدار بگیرد. به بیانی دیگر R عبارتست از زیرمجموعه‌ای از ضرب کارترین $S_1 * S_2 * \dots * S_n$. و می‌گوییم رابطه R از درجه n است. هر یک از مجموعه‌های S_1, S_2 و... میدان یا دامنه^۲ نامیده می‌شوند. دامنه، مجموعه مقادیری است که یک صفت از رابطه مشخص می‌توانند داشته باشند.

$$D1 = \{1, 2\}$$

$$D2 = \{T, F, V\}$$

$$D2 * D1 = R1$$

D1	D2
1	T
1	F
1	V
2	T
2	F
2	V

شکل ۶-۱ نمایش رابطه $R1 = D1 * D2$

1. Tuple
2. Domain

برای درک بهتر موضوع مثال بالا که نشان دهنده دو مجموعه و حاصل ضرب بین آن دو است، توجه کنید. شاید این مثال تا حدودی شفافیت لازم را برای خواننده نداشته باشد. اکنون مثال دیگری را با جزئیات بیشتر بیان می‌کنیم. فرض کنید مجموعه‌های زیر را داشته باشیم:

- S1 مجموعه مقادیر شماره دانشجویان
- S2 مجموعه اسامی دانشجویان
- S3 مجموعه مقادیر سطوح تحصیلی در دانشگاه
- S4 مجموعه مقادیر رشته‌های تحصیلی
- S5 مجموعه مقادیر شماره گروه‌های آموزشی

در این صورت STT با پنج صفت خاصه مرتبط با پنج مجموعه بالا، یک رابطه بوده و بصورت زیر نوشته می‌شود:

STT(STID,STNAME,STDEG,STMJR,STDEID)

جدول ۶-۲ نمایش جدولی رابطه STT

STT

STID	STNAME	STDEG	STMGR	STDEID
1	Reza	Bs	computer	V532
2	Hamid	ms	math	V135
...	...	...	...	...

۲-۳ تعریف رابطه

با فرض وجود n میدان $D_1, D_2, \dots, D_n$ (نه لزوماً متمایز)، رابطه R از دو قسمت تشکیل شده است:

۱- **مجموعه عنوان:** مجموعه‌ای نامدار است از n صفت به صورت $A_i: D_i$ که در آن هر A_i نام یک صفت است و هر D_i نام میدان صفت است. لازم به ذکر است که A_i ها از یکدیگر متمایز هستند. به این مجموعه شمای رابطه^۱ نیز می‌گویند که به صورت شماتیک چنین است:

$$\{ \langle A_1: D_1 \rangle, \langle A_2: D_2 \rangle, \dots, \langle A_n: D_n \rangle \}$$

۲- مجموعه بدنه: مجموعه‌ایست از m تاپل t بنحوی که t خود مجموعه‌ایست از n عنصر بصورت $A_i: v_i$ که در آن v_i مقداری است از نوع $(\text{میدان}) D_i$.

$$\{ \langle A_1: v_{i1} \rangle, \langle A_2: v_{i2} \rangle, \dots, \langle A_n: v_{in} \rangle \}$$

$$(i = 1, 2, 3, \dots, m)$$

- پیکر رابطه را گاهی بسط^۱ رابطه یا حالت رابطه^۲ هم می‌گویند.
- مقدار n را درجه^۳ رابطه یا آریتی^۴ می‌گویند.
- مقدار m را کاردینالیتی^۵ رابطه می‌گویند.

۳-۳ تناظر بین مفاهیم رابطه‌ای و مفاهیم جدولی

همانطور که می‌دانید برای پیاده سازی مدل رابطه‌ای در محیط انتزاعی از رابطه استفاده می‌شود. برای پیاده سازی مدل رابطه‌ای در ساختار پایگاه‌داده از مفهوم جدول استفاده می‌شود.

جدول ۳-۶ تناظر بین اجزاء دو مفهوم رابطه و جدول

اجزاء مفهوم رابطه	اجزاء مفهوم جدولی
رابطه	جدول
تاپل	سطر
صفت	ستون
میدان	مجموعه مقادیر ستون
درجه	تعداد ستون‌ها
کاردینالیتی	تعداد سطرها

مفاهیم جدول و رابطه تا حدود زیادی به یکدیگر نزدیک بوده و در مواردی به

-
1. Extension
 2. Relation State
 3. Degree
 4. Arity
 5. Cardinality

اشتباه بجای یکدیگر مورد استفاده قرار می‌گیرند. در جدول ۶-۳ تناظر بین دو مفهوم رابطه و جدول نشان داده شده است.

۳-۴ تعریف صوری جدول

فرض می‌کنیم U مجموعه‌ای از نمادها موسوم به صفات رابطه‌ای باشد و برای هر $A \in U$ مجموعه‌ای از مقادیر به نام $\text{Domain}(A)$ وجود داشته باشد. نماد نمایشگر ستون جدول است و $\text{Domain}(A)$ مقادیری هستند که در ستون A ظاهر می‌شوند. فرض می‌کنیم $H = \{A_1, A_2, \dots, A_n\}$ مجموعه‌ای متناهی از صفات رابطه‌ای باشد. می‌دانیم که ضرب کارتیزین

$$\text{Domain}(A_1) * \dots * \text{Domain}(A_n)$$

از تاپلهایی به صورت $t = (a_1, a_2, \dots, a_n)$ تشکیل شده است که در آن:

$$1 \leq i \leq n, a_i \in \text{Domain}(A_i)$$

تاپل T را می‌توان به گونه دیگری هم تعریف کرد:

$$t = \{A_1, A_2, \dots, A_n\} \rightarrow U \{ \text{Domain}(A_i), 1 \leq i \leq n \}$$

به نحوی که

$$T(A_i) \in \text{Domain}(A_i)$$

ضرب کارتیزین $\text{Domain}(A_1) * \dots * \text{Domain}(A_n)$ را با $\text{tupl}(H)$ نمایش می‌دهیم، به بیان دیگر:

$$\text{Tupl}(H) = \{t \mid t: H \rightarrow \bigcup_{A \in H} \text{Domain}(A), t(A) \in \text{Domain}(A), \text{ for } A \in H\}$$

عناصر $\text{Tupl}(H)$ را تاپلهای روی H می‌نامیم.

رابطه R روی میدانهای $\text{Domain}(A_1), \dots, \text{Domain}(A_n)$ زیر مجموعه‌ای است از $\text{Tupl}(H)$. مجموعه همه این گونه رابطه‌ها را با $\text{rel}(H)$ نمایش می‌دهیم. حال می‌گوییم جدول عبارتست از سه تایی:

$$\tau = (T, H, \rho)$$

که در آن: T نمادی است به عنوان جدول و $H = \{A_1, A_2, \dots, A_n\}$ مجموعه‌ای است از صفات رابطه‌ای که به آن سرآیند τ گوئیم و ρ یک رابطه است، $\rho \in \text{rel}(H)$ و به آن گسترده τ می‌گوئیم.

۴- ویژگی‌های رابطه

رابطه به عنوان تنها عنصر ساختاری اصلی در مدل رابطه‌ای برای نمایش انواع موجودیتها و انواع ارتباطات بکار می‌رود. در واقع در مدل رابطه‌ای هم نوع موجودیت و هم نوع ارتباط با مفهوم رابطه نمایش داده می‌شوند و در نتیجه هم نمونه موجودیت و هم نمونه ارتباط با مفهوم تاپل نشان داده می‌شوند. رابطه دارای چهار ویژگی به شرح ذیل می‌باشد:

- رابطه تاپل تکراری ندارد.
- زیرا بدنه رابطه مجموعه است و مجموعه نمی‌تواند عنصر تکراری داشته باشد.
- تاپل‌ها نظم ندارند.
- زیرا بدنه رابطه مجموعه است و مجموعه در حالت کلی فاقد نظم است.
- صفات رابطه نظم مکانی (از چپ به راست) ندارند.
- زیرا سرآیند رابطه مجموعه است و مجموعه در حالت کلی فاقد نظم است.
- تمام صفات تک مقداری (تجزیه نشدنی) هستند.
- زیرا در نمایش جدولی رابطه، در تقاطع هر سطر و ستون، باید یک مقدار وجود داشته باشد. به بیانی دیگر در هر تاپل، دقیقاً یک مقدار برای هر صفت وجود دارد.

در ادامه برای درک بهتر موضوع مثالی ارائه شده است. در این مثال رابطه‌ای بنام S نشان داده شده است.

مثال ۱: رابطه S را که به صورت جدول ۶-۴ است در نظر بگیرید.

جدول ۶-۴ نمایش جدولی رابطه S

S#	S NAME	STATUS	CITY
S1	S	20	CONDON
S2	J	10	PARIS
S3	B	30	PARIS
S4	C	20	LONDON
S5	A	30	ATHENS

همانگونه که مشاهده می‌کنید در جدول بالا:

- S# نشان دهنده کلید اصلی است. (در مورد مبحث کلید در ادامه توضیح خواهیم داد).
- (S# , SNAME , STATUS , CITY) این ۴ مشخصه نشان دهنده attribute هستند.
- دامنه CITY شامل (ATHENS ,PARIS ,LONDON) است.
- (ATHENS ,LONDON ,PARIS ,PARIS ,LONDON) نیز نشان دهنده تعداد سطرها یا Cardinality می‌باشند.
- (S5 , A , 30 , ATHENS) نشان دهنده تعداد صفات درجه Degree است.

۵- انواع کلید در مدل رابطه‌ای

در مدل رابطه‌ای چند مفهوم اساسی در بحث کلید وجود دارد که در ادامه بررسی می‌شوند:

۵-۱ ابر کلید^۱

هر ترکیبی از صفات جدول را که یکتایی مقدار را داشته باشد، ابر کلید گویند (به صورت دلخواه). به بیانی دیگر هر زیر مجموعه عنوان رابطه که یکتایی مقدار در گسترده (بدنه) رابطه داشته باشد. تعریف دیگر ابر کلید عبارت است از هر ترکیبی از اسامی صفات رابطه که در هیچ دو تاپل مقدار یکسان نداشته باشد.

۵-۲ کلید کاندید^۲

کلید کاندید امکانی است برای ارجاع به "تک تاپل" در رابطه. مجموعه صفات K از

1. Super key
2. Candidate

رابطه R، یک کلید کاندید می‌باشد (مانند S# و SNAME) اگر دارای خواص زیر باشند:

- خاصیت منحصر بفرد بودن (یکتایی مقدار):
هیچ دو tuple مجزائی از رابطه R دارای مقدار یکسانی برای K نباشند. (به طور مثال هیچ دو سطر S# یکی نیست ولی CITY تکرار دارد، status، city هم نمی‌توانند با هم باشند زیرا دو فروشنده، در لندن 20 هستند).
- خاصیت غیر کاهشی (minimal):
هیچ زیر مجموعه مناسبی از K وجود ندارد که دارای خاصیت منحصر بفرد باشد.

به بیانی دیگر هر زیر مجموعه از مجموعه عنوان رابطه که دو خاصیت بالا را داشته باشد، کلید کاندید رابطه است.

در مثال فوق S# و sname کلید انحصاری هستند. خاصیت دوم را ندارند زیرا S# به تنهایی منحصر بفرد است و می‌توان اینها رو شکست مثلاً S# به تنهایی هم می‌تواند کلید کاندید بشود. (هر چیزی را نمی‌توان کلید کاندید به حساب آوریم زیرا کلید کاندید نباید بشکند).

توجه: هر رابطه حداقل دارای یک کلید کاندید می‌باشد (ممکن است تکی باشد یا دو صفت و یا...).

۵-۳ کلید اصلی^۱

یکی از کلیدهای کاندید رابطه که شرایط ذیل را داشته باشد:

- شناسایی کننده نوع موجودیت (تک تاپل) در رابطه باشد. مانند شماره دانشجویی برای هر دانشجو.
- از نظر سائز دارای طول کوتاه تر باشد. بدین معنی که مثلاً بین دو کلید کاندید که یکی از نوع رشته ۲۰ تایی و دیگری یک عدد ۴ بایتی است، کلیدی که دارای نوع عدد ۴ بایتی است، برای کلید اصلی بودن بهتر است.

۵-۴ کلید بدیل^۱

1. Primary Key

هر کلید کاندید بغیر از کلید اصلی را کلید بدیل گویند.

۵-۵ کلید خارجی^۲

دو رابطه R1 و R2 را در نظر بگیرید. هر زیر مجموعه از صفات رابطه R2 که هر مقدار معلومش با یک مقدار از کلید کاندید R1 برابر باشد، کلید خارجی در رابطه R2 است.

نقش کلید خارجی برای نمایش ارتباطات بین انواع موجودیت‌ها (و در نتیجه بین نمونه‌های آنها) بکار می‌رود. برای درک بهتر موضوعات کلید به مثال زیر توجه فرمایید:

مثال ۲: کلیدی که ارتباط جدول مختلف را مشخص و برقرار می‌نماید.

جدول ۵-۶ جدول مثال ۲

C#	CNAME	STATUS	NO....
10	Pen	Q	5
10	Pen	1	10
10	Pen	2	7
10	Ruler	Q	

C# و status می‌توانند کلید کاندید باشند کلید کاندیدی که می‌تواند کلید اصلی بشود ولی به تنهایی نمی‌توانند کلید کاندید باشند.

جدول فروشنده‌ها (سازنده کالا)

جدول ۶-۶ جدول فروشنده‌ها

S#	SNAME	C#	Tell#
100	a	10	
105	b	11	

1. Alternate Key

2. Foreign key

توجه: در جدول فوق s# کلید کاندید می‌باشد.

جدول کالا

جدول ۶-۷ جدول کالا

C#	CNAME	STATUS	NO...
10	Pen	Q	5
10	Pen	1	7
10	Pen	2	10
11	ruler	Q	

نکته: C#، کلید خارجی برای ارتباط بین دو جدول کالا و فروشنده می‌باشد.
نکته: C#، در جدول فروشنده کلید خارجی و در جدول کالا کلید اصلی می‌باشد.

نکاتی در مورد کلید خارجی

فرض کنید که رابطه R2 یک رابطه مبنا باشد آنگاه یک کلید خارجی در رابطه R2 زیر مجموعه‌ای از مجموعه صفات R2 مانند FR می‌باشد به نحوی که:

- رابطه مبنایی به نام R1 با کلید کاندید CK وجود داشته باشند.
- برای تمامی مواقع هر مقدار از FK در مقدار فعلی R2 با مقدار CK در بعضی از Tupleها در مقدار قبلی یکسان باشند.

نکته: اگر صفتی بتواند مقدار تهی (null) بگیرد، نمی‌تواند کلید باشد (مانند tell# (شماره تلفن)).

از ویژگی‌های RDBMS در پایگاه داده رابطه‌ای در مورد جداول این است که:

- داده‌ها توسط کاربر و به صورت جداول دریافت می‌شوند.
- عملکردهایی که کاربر می‌تواند از آنها استفاده نماید که آنها به کاربر این اجازه را می‌دهند تا بتوانند جداول جدیدی را ایجاد نمایند.

مثال: بخشی از جداول یک سیستم اداری به صورت زیر است:

کارمند EMP

جدول ۶-۸ Employee جدول

EMP#	SNAME	DEPT#	SALARY
E1	LOP	D1	40
E2	CHEN	D1	42
E3	FIN	D2	30
E4	SAI	D2	35

بخش‌ها DEPT

جدول ۶-۹ Department جدول

DEPT# #سازمان	DNAME نام سازمان	بودجه
D1	MARKETING	40
D2	DEVELOPMENT	12
D3	RESEARCH	5

SELECT * FROM DEPT WHERE (8 > بودجه)

نکته: در عبارت بالا علامت * نمایانگر همه صفات‌ها است.

در نتیجه خروجی حاصل از عبارت بالا به صورت جدول زیر خواهد بود. همانگونه که مشاهده می‌شود داده‌ها به صورت جدول وارد سیستم شده‌اند و گزارش خواسته شده نیز بر اساس یک سری از عملگرهای استاندارد ارائه شده و نتیجه به صورت جدول (بر اساس نوع عملگر) گزارش می‌گردد.

جدول ۶-۱۰ خروجی حاصل از عبارت ذکر شده

#سازمان	نام سازمان	بودجه
D1	MARKETING	10
D2	DEVELOPMENT	12

نکته: خروجی‌ها از همان نوع ورودی‌ها می‌باشند.

نکته: خروجی هر عملگر می‌تواند به عنوان ورودی عملگر دیگر بکار رود. این نوع

عبارت را عبارت تو در تو^۱ می نامند.

۶- انواع رابطه

انواع رابطه عبارتند از

رابطه نامدار^۲

رابطه‌ایست که با یک نام به سیستم معرفی شده باشد.

رابطه مبنا^۳

نوعی رابطه نامدار است که استقلال وجودی دارد و مشتق از رابطه‌های دیگر نیست و داده‌های ذخیره شده متناظر دارد.

رابطه مشتق^۴

رابطه‌ایست که به کمک یک عبارت رابطه‌ای بر حسب رابطه‌های نامدار دیگر و نهایتاً بر حسب رابطه‌های مبنا تعریف می‌شود.

مفهوم دید

نوعی رابطه نامدار که مشتق از رابطه‌های دیگر است و ماهیتاً رابطه‌ای مجازی^۵ است، یعنی داده‌های ذخیره شده خاص خود را ندارد.

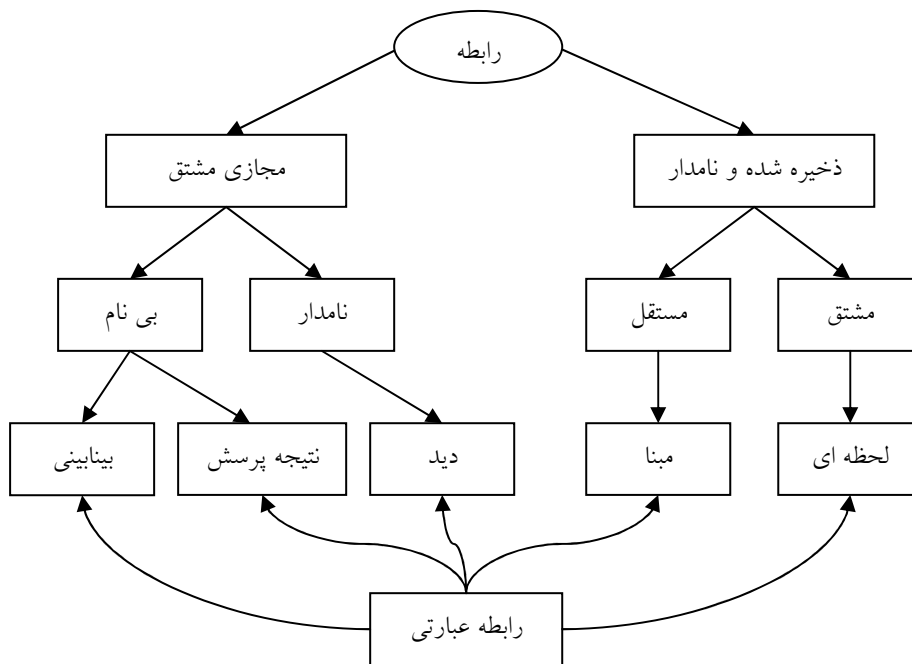
انواع دیگری از رابطه‌ها در کتابهای دیگر نامبرده شده‌اند که در ذیل به آنها اشاره می‌گردد. لازم به ذکر است که روابط مذکور حائلهایی از روابط ذکر شده در بالا هستند:

- رابطه لحظه‌ای^۶
- رابطه عبارتی^۷
- رابطه نتیجه پرسش^۸

1. Nested Expression
2. Named Relation
3. Base Relation
4. Derived Relation
5. Virtual Relation
6. Snapshot
7. Expression Relation
8. Query Result Relation

- رابطه بینابینی^۱
- رابطه ذخیره شده^۲

شکل زیر ارتباط و دسته بندی انواع رابطه را نشان می دهد:



شکل ۶-۱۱ نمایش نحوه ارتباط بین رابطه‌ها

۷- قواعد جامعیت^۳ در مدل رابطه‌ای

در هر محیط عملیاتی همیشه مجموعه‌ای از قواعد معنایی^۴ یا محدودیت‌های جامعیتی^۵ وجود دارند. اگر داده‌های ذخیره شده در پایگاه داده همواره این

1. Intermediate Relation
 2. Stored Relation
 3. Integrity Rules
 4. Semantic Rules
 5. Integrity Constraints

محدودیت‌ها را رعایت کنند، جامعیت پایگاه داده تامین می‌باشد.

جامعیت پایگاه داده‌ها بمعنی صحت، دقت و سازگاری داده‌های ذخیره شده در پایگاه داده‌ها در تمام لحظات است. هر سیستم پایگاه داده باید بتواند جامعیت پایگاه داده را کنترل و تضمین کند. زیرا ممکن است عواملی مانند آنچه که در ذیل نام برده شده‌اند، باعث نقض جامعیت شوند:

- اشتباه در برنامه‌ها
- اشتباه در ورود اطلاعات از سوی کاربران
- مشکلات سخت افزاری و نرم افزاری مرتبط با داده‌ها
- عدم انجام کامل فرایندها بر روی داده‌ها
- وجود افزونگی و تعدد در داده‌های تکراری

برای کنترل و تضمین جامعیت، قواعدی لازم است تا سیستم مدیریت بتواند بر اساس آنها عمل کرده و باعث انطباق محتوای پایگاه داده با واقعیات باشد. این قواعد را قواعد جامعیتی و یا محدودیت‌های جامعیتی گویند.

۷-۱ انواع قواعد جامعیت

قواعد جامعیت در مدل رابطه‌ای به دو رده کلی تقسیم می‌شوند که به شرح هریک خواهیم پرداخت:

- قواعد کاربری^۱
- متا قواعد^۲

۷-۱-۱ قواعد کاربری

قواعد کاربری که گاه به آنها قواعد محیطی یا قواعد وابسته به داده نیز می‌گویند، قواعدی هستند که توسط کاربر مجاز و برای یک پایگاه داده خاص تعریف می‌شوند. این قواعد وابسته به واقعیات محیط هستند و در بعضی کتب به آنها محدودیت‌های

1. User Defined Rules

2. Meta Rules

جامعیتی معنایی^۱ نیز می‌گویند. قواعد کاربری در مدل رابطه‌ای به چهار دسته به شرح زیر تقسیم می‌شوند:

- **محدودیت میدانی^۲**: محدودیتی است ناظر بر میدان هر صفت خاصه و مقادیر مجاز آنرا مشخص می‌کند.
- **محدودیت صفتی^۳**: محدودیتی است ناظر بر یک صفت و نوع آن را مشخص می‌کند.
- **محدودیت رابطه‌ای^۴**: محدودیتی است ناظر بر یک رابطه و مقادیر مجاز یک متغیر رابطه‌ای را مشخص می‌کند.
- **محدودیت پایگاهی^۵**: محدودیتی است ناظر بر دو یا چند متغیر رابطه‌ای به نحوی که آنها را بهم مرتبط می‌کند.

۷-۱-۲ متا قواعد

متا قواعد قواعدی هستند که باید توسط هر سیستم رابطه در هر پایگاه داده رابطه‌ای اعمال گردند. لذا در بعضی کتب بدانها قواعد عام نیز می‌گویند. این قواعد به دو گروه کلی به شرح ذیل تقسیم می‌شوند:

- قاعده جامعیت موجودیتی^۶
- قاعده جامعیت ارجاعی^۷

۷-۱-۲-۱ قاعده جامعیت موجودیتی

این قاعده ناظر بر کلید اصلی است و به شرح ذیل می‌باشد:

- هیچ جزء تشکیل دهنده کلید اصلی رابطه نمی‌تواند "مقدار هیچ" داشته باشد.

دلیل توجیه کننده این قاعده این است که هر مقدار یک کلید اصلی، در واقع شناسه

1. Semantic Integrity Constraint

2. Domain Constraint

3. Attribute Constraint

4. Relation Constraint

5. Database Constraint

6. Entity Integrity Rule

7. Referential Integrity Rule

آن تاپل در رابطه است و عامل تمییز نمونه‌های موجودیت (تاپل‌ها) در رابطه است و بدیهی است که عامل تمییز خود نمی‌تواند مقدار هیچ یا ناشناخته داشته باشد.

۷-۱-۲-۲ قاعده جامعیت ارجاعی

این قاعده ناظر بر کلید خارجی بوده و به شرح ذیل می‌باشد:

- اگر صفت خاصه A_i (ساده یا مرکب) در رابطه R_2 کلید خارجی باشد در این صورت A_i در R_2 می‌تواند مقدار هیچ داشته باشد (به شرطی که جزئی از کلید R_2 نباشد)، در غیر اینصورت باید حتما مقداری باشد که در رابطه مرجع R_1 وجود دارد. به عبارت دیگر مقدار کلید خارجی نمی‌تواند در رابطه مرجع وجود نداشته باشد.

دلیل توجیه این قاعده این است که کلید خارجی عامل ارجاع از یک نمونه موجودیت به نمونه موجودیت دیگر است و بدیهی است که نمی‌توان به نمونه موجودیت ناموجود ارجاع داد.

تمرینات

۱. عناصر اصلی ساختار جدولی را نام ببرید؟
۲. رابطه را تعریف کنید؟

۳. رابطه را از دیدگاه کاد تعریف کنید؟
۴. ویژگی‌های رابطه را نام ببرید؟
۵. انواع کلید را در مدل رابطه‌ای نام برده و توضیح دهید؟
۶. مفهوم کلید خارجی را توضیح دهید؟
۷. انواع رابطه را نام ببرید؟
۸. قاعده جامعیت موجودیتی را توضیح دهید؟
۹. قاعده جامعیت ارجاعی را توضیح دهید؟

فصل ۷

عملیات در پایگاه رابطه‌ای

هدف کلی

در این فصل ابتدا تعریفی از پایگاه داده رابطه‌ای بیان شده و سپس مبحث جبر رابطه‌ای مورد بحث و بررسی قرار خواهد گرفت. در این راستا در ابتدا عملگرهای جبر رابطه‌ای معرفی شده و توانایی این عملگرها در الحاق و جداسازی اطلاعات و نمایش آنها شرح داده خواهد شد. سپس عملگرهای کار بر روی داده‌ها مطرح شده و انواع این عملگرها مورد بررسی قرار خواهند گرفت. در خاتمه نیز مروری مختصر بر روی مبحث بهینه سازی عبارات جستجو در پایگاه داده‌ها خواهیم داشت.

هدف رفتاری

در این فصل عناوین زیر مورد بحث و بررسی قرار می گیرند:

- تعریف پایگاه داده رابطه‌ای
- جبر رابطه‌ای
- عملگرهای جبر رابطه‌ای
- عملگر گزینش
- عملگر پرتو
- عملگر تغییر نام
- عملگر جایگزینی
- عملگرهای مجموعه

- عملگر ضرب دکارتی
- عملگر اجتماع
- عملگر اشتراک
- عملگر تفاضل
- عملگر تقسیم
- عملگر پیوند
- عملگرهای کار بر روی داده‌ها
- عملگرهای درج، بهنگام سازی، حذف
- بهینه سازی پرس و جوها

۱- تعریف

پایگاه داده رابطه‌ای از دید کاربر در محیط انتزاعی مجموعه‌ای از رابطه‌های نرمال است. بدیهی است که جهت انجام عملیات در این محیط انتزاعی نیاز به یک مدل ریاضی است. پس کاربر تعریف شده در پایگاه داده نیاز به یک مدل ریاضی، که می‌تواند مجموعه‌ای از عملگرهای صوری باشد، دارد. این امکانات در واقع بخشی از مدل رابطه‌ای بوده و طبعاً بدون وجود آنها، مدل رابطه‌ای کامل نخواهد بود. برای این منظور امکانات زیر وجود دارد:

- جبر رابطه‌ای^۱
- حساب رابطه‌ای
- محاسبات رابطه‌ای دامنه^۲
- محاسبات رابطه‌ای سطری^۳

لازم به ذکر است که در تئوری رابطه‌ها، هر سه تئوری فوق ارزش یکسانی دارند. در ادامه جبر رابطه‌ای مورد بررسی قرار می‌گیرد.

1. Relational algebra

2. Calculate Relational Domain

3. Calculate Relational Tuple

۲- جبر رابطه‌ای (RA) و عملگرهای آن

جبر رابطه‌ای ارائه شده توسط کاد (codd) شامل هشت عملگر در دو گروه چهار تایی می‌باشد. در بعضی کتب تعدادی عملگر دیگر نیز به این عملگرها اضافه شده‌اند. کلیه این عملگرها برای کار بر روی مفهوم رابطه‌ها تعریف و طراحی شده‌اند. در جبر رابطه‌ای داده‌ها همان رابطه‌ها می‌باشند. مجموعه عملگرهای جبر رابطه‌ای در عبارت زیر مشخص شده است، بدین ترتیب که:

$$RA = \{\{Relations\}, \{ \bowtie, \Pi, \rho, \leftarrow, X, U, -, \cap, \div, \infty, X\theta, \theta\alpha \}\}$$

در بعضی از کتب تعدادی از این عملگرها تحت عنوان عملگر **set** و تعدادی دیگر تحت عنوان عملگر **join** معرفی شده‌اند که این دو گروه نیز توضیح داده خواهند شد. عملگرهای جبر رابطه‌ای در رابطه بالا به ترتیب عبارتند از:

- $\bowtie$ = عملگر گزینش^۱
- Π = عملگر پرتو^۲
- ρ = عملگر تغییر نام^۳
- $\leftarrow$ = عملگر جایگزینی
- Set = عملگر مجموعه، این عملگرها بر روی مجموعه‌ای از رابطه‌ها عمل می‌کنند و شامل عملگرهای
 - ضرب دکارتی^۴ (X)
 - اجتماع^۵ (U)
 - اشتراک^۶ ($\cap$)
 - تفاضل^۷ ($-$)
 - تقسیم ($\div$)

1. select
 2. project
 3. rename
 4. Cartesian product
 5. union
 6. intersect
 7. difference

- Join = عملگر پیوند (الحاقی) شامل عملگرهای
- پیوند طبیعی^۱ (∞)
- نیم پیوند^۲ ($\theta\alpha$)
- پیوند شرطی ($X\theta$)

در ادامه با ذکر مثال هایی به شرح هر یک از این عملگرها خواهیم پرداخت.

عملگرهای اصلی Select (B)

انتخاب سطرهایی از یک رابطه بر اساس شرط مشخص (مشخصات مثلا یک فرد در یک سطر (افرادی که معدلشان مساوی x است)).

عملگر اصلی Project (Π)

انتخاب دیگر ستونهای مشخص از یک رابطه و حذف دیگر ستونها

نام		تلفن	

الحاق (Join)

دارای انواع مختلف بوده و رابطه‌ای را بر می‌گرداند که شامل تمام Tuple های ترکیب شده از دو جدول می‌باشد. برای درک بهتر موضوع در ادامه مثالی از عملگرهای الحاقی ارائه شده است.

1. Natural Join
2. Semi Join

عملیات در پایگاه رابطه‌ای ۱۳۱

R1	R2		R1 $\bowtie$ R2		
a1	b1	→	a1	b1	c1
a2	b2	→	a2	b2	c2
a3	b3		a3	b3	c3

تمرین:

با استفاده از عملگر ($\div$) نتیجه $R1 \div R2 = ? (R3)$ را بدست آورید:

R1	R2	R3													
<table><tr><td>a</td><td>X</td></tr><tr><td>a</td><td>y</td></tr><tr><td>a</td><td>z</td></tr><tr><td>b</td><td>X</td></tr><tr><td>c</td><td>y</td></tr></table>	a	X	a	y	a	z	b	X	c	y	<table><tr><td>x</td></tr><tr><td>y</td></tr></table>	x	y	<table><tr><td>?</td></tr></table>	?
a	X														
a	y														
a	z														
b	X														
c	y														
x															
y															
?															

نکته: یکی از مهمترین خواص، خاصیت بسته بودن (بستار یا closure) می‌باشد. بدین معنا که خروجی هر عملگر رابطه‌ای خود یک رابطه است. بطور مثال می‌توان گفت که اجتماع دو رابطه (دو جدول) یک رابطه (یک جدول) می‌گردد.

عملگر تغییر نام (ρ)

این عملگر یک رابطه را گرفته و نسخه دیگری از آن را که ممکن است به بعضی از صفات اسامی دیگری داده شده باشد را می‌دهد:

S RENAME city AS Scity

تغییر نام صفت city به Scity در رابطه S

اگر بیشتر از یک تغییر انجام گیرد در پرانتز قرار می‌دهیم. بطور مثال:

S RENAME (city AS Scity , S# AS SNUM)

S#	SNAME	STATUS	City
S1	Smith	20	London
S2	Jones	10	Paris
S3	Black	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

با استفاده از دستور RENAME جدول فوق به جدول ذیل تبدیل می‌گردد.

SNUM	SNAME	STATUS	Scity
S1	Smith	20	London
S2	Jones	10	Paris
S3	Black	30	Paris
S4	Clark	20	London
S5	Adoms	30	Athens

مثال:

A

S#	SNAME	STATUS	City
S1	Smith	20	London
S4	Clark	20	London

B

S#	SNAME	STATUS	City
S1	Smith	20	London
S2	Jones	10	Paris

با توجه به دو جدول فوق داریم:

A U B

عملیات در پایگاه رابطه‌ای ۱۳۳

S#	SNAME	STATUS	city
S1	Smith	20	London
S4	Clark	20	London
S2	Jones	10	Paris

$A \cap B$

S#	SNAME	STATUS	City
S1	Smith	20	London

$B - A$

S#	SNAME	STATUS	City
S4	Clark	20	London

$A - B$

S#	SNAME	STATUS	city
S2	Jones	10	Paris

توجه:

$$(A \cup B) \cup C \equiv A \cup (B \cup C) = A \cup B \cup C$$

$$(A \cap B) \cap C \equiv A \cap (B \cap C) = A \cap B \cap C$$

$$(A \times B) \times C \equiv A \times (B \times C) = A \times B \times C$$

دستور (B)

این دستور همان عملگر گزینش (Select) است و در واقع تعدادی از سطرها را انتخاب می‌کنیم. شکل کلی دستور به یکی از دو صورت ذیل می‌باشد:

(نام رابطه) B (در کامپیوتر اجرا نمی‌شود).

شرط یا شرطها

شرط یا شرطها where نام رابطه (در کامپیوتر اجرا نمی‌شود (SQL)).

مثال:

$\sigma_{\text{city} = \text{London}}$ یا $S \text{ where } \text{city} = \text{London}$
 $\text{city} = \text{London}$

S#	SNAME	STATUS	CITY
S1	smith	20	London
S4	clark	20	london

مثال:

$\sigma_{\text{CITY} = \text{"London"} \text{ AND } \text{SNAME} = \text{"smith"}}$

S#	SNAME	STATUS	CITY
S1	smith	20	London

دستور (Π)

این دستور همان عملگر پرتو (Project) می‌باشد که در واقع ستون‌هایی از رابطه را در بر می‌گیرد. شکل کلی دستور به یکی از دو صورت ذیل می‌باشد:

۱- $\Pi_{\text{نام رابطه}}$

نام صفت

۲- $\Pi_{\text{نام صفتها}} [\text{نام رابطه}]$

مثال:

Π_{City} یا $S [\text{city}]$

City
London
Paris
Athens

مثال:

$\Pi_{\text{S\#}, \text{SNAME}}$ یا $S [\text{S\#}, \text{SNAME}]$
 $\text{S\#}$ و SNAME

عملیات در پایگاه رابطه‌ای ۱۳۵

S#	SNAME
S1	Smith
S2	Jones
S3	Black
S4	Clark
S5	Adams

تمرین: نام افرادی که شهرشان لندن (london) است ؟

$$\left(\begin{array}{c} \pi_{\text{city} = \text{"London"}} \\ \text{S#} \end{array} \right)$$

تمرین: S# افرادی که وضعیتشان برابر ۳۰ است.

$$\left(\begin{array}{c} \pi_{\text{STATUS} = 30} \\ \text{S#} \end{array} \right)$$

$$\left(\begin{array}{c} \pi_{\text{S#}} \\ \text{Name} \end{array} \right)$$

(S WHERE CITY = "London")

عملگرهای پیوند (join)

ضرب دکارتی^۱

رابطه‌ای است که ستونهایش مجموع ستونهای دو رابطه و سطرهایش برابر با حاصلضرب (تمامی ترکیبهای ممکن) سطرهای آن دو رابطه (تکراری نیز خواهیم داشت) است.

خواص عملگر پیوند

- اطلاعات کامل

1. Cartesian Product

- حجم یا فضای زیادی اشغال می‌کند (حتی الامکان انجام نمی‌دهیم).
- $a * b = b * a$

کاربردهای ضرب دکارتی (X)

یکی از کاربردهای ضرب دکارتی مواقعی است که بین جداول عامل ارتباط (صفت‌های مشخص) نداشته باشیم (راه حل خوبی نیست زیرا فضای زیادی اشغال می‌کند).

a	b	c	d	*	x	y

بخاطر پرمزینه بودن ضرب، joinهای دیگر ارائه گردیدند (اصل join با ضرب شروع می‌شود).

الحاق طبیعی (∞) Natural join

نتیجه حاصل از این عملیات شامل جدولی است که ستونهایش، ستونهای دو جدولی (بدون تکرار) و سطرهایش، سطرهایی از جدول است که مقادیر همه ستونهای هم نامش مساوی هستند.

A		B			C		
a	b	b	c		a	b	C
1	2	2	4	=	1	2	4
2	4	5	10		1	2	1
		6	8		2	4	3
		4	3				
		2	1				

در مباحث پیوند رابطه‌ها نکات زیر باید مورد توجه قرار گیرد:

$$(A \bowtie B) \bowtie C = A \bowtie (B \bowtie C)$$

$$A \bowtie B = B \bowtie A$$

عملیات در پایگاه رابطه‌ای ۱۳۷

اگر دو جدول A و B دارای صفات مشترکی نباشند ∞ تبدیل به X می‌شود.

$$(X \theta (\theta - \text{join}) \text{teta join})$$

و این عبارت معادل ضرب دکارتی است که شرطی نیز به آن افزوده شده است.

$$R1 * R2 \quad \text{یا} \quad (R1 * R2) \text{ WHERE شرط}$$

شرط

کاربرد: مواقعی که نیاز به join دو رابطه بر اساس شرط خاصی به غیر از برابری آنها باشد.

مثال:

A		B		A * B A.b > B.c			
a	b	c	d	a	b	c	d
x	15	10	K	x	15	10	K
y	20	25	1	y	20	10	K

$$\alpha = \text{semi join}$$

همان $X \theta$ است که فقط شامل ستونهای جدول اول می‌باشد (بدون ستونهای تکراری).

$$R1 \alpha R2$$

شرط

کاربرد: در پایگاه‌های داده توزیع یافته (Distributed DB) بکار می‌رود تا از انتقال اطلاعات زیادی جلوگیری کند.

مثال:

B α A	A	B
--------------	---	---

$$c.B < b.A$$

x	15
y	20

عملگر تقسیم^۱

فرض کنید

$$A (X_1 \text{ و } X_2 \text{ و } \dots \text{ و } X_n \text{ و } Y_1 \text{ و } Y_2 \text{ و } \dots \text{ و } Y_n)$$

و

$$B (Y_1 \text{ و } Y_2 \text{ و } \dots \text{ و } Y_n)$$

در تقسیم دو رابطه، ستونهای باقی مانده و سطرها، سطرهایی است که شرط در همه آنها صادق است.

$$A \text{ DIVIDE By } B = A \div B = (X_1 \text{ و } X_2 \text{ و } \dots \text{ و } X_n)$$

لازم به ذکر است که اگر Y_2 را نداشته باشیم X_2 را به ما نمی‌دهد، اگر Y_n را نداشته باشیم X_n را به ما نمی‌دهد.

ممکن است این نکته به ذهن برسد که اساساً این عملگر چه کاربردهایی می‌تواند داشته باشد. در پاسخ به این پرسش باید گفت که یکی از ساده ترین موارد استفاده از این عملگر پرس و جوهایی است که با استفاده از آنها دنبال مقادیری بگردیم که حاوی یک کلمه خاص باشد. (مثلاً تمام X هایی را بده که Y هایشان برابر $\square$ است).
مثال:

سازنده‌ها ($S\#$ و $SNAME$ و $STATUS$ و $CITY$)

محصولات ($P\#$ و $PNAME$ و $WEIGHT$ و $CITY$)

ترکیبی از محصولات و سازنده ($S\#$ و $P\#$ و Qty)

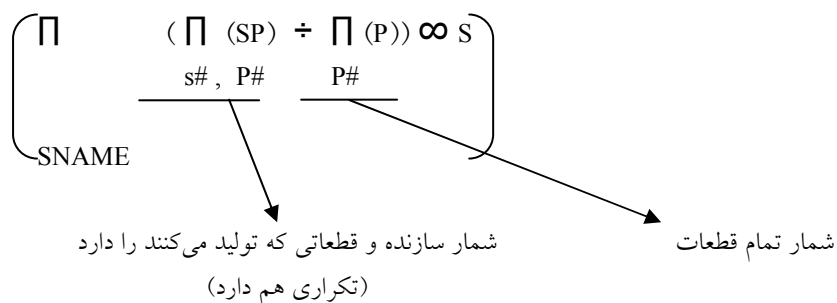


تعداد

سوال:

فروشنده‌گانی که همه (تمام) قطعات را تولید می‌کنند یا نام تولید کنندگان همه

قطعات؟



جواب: S# که همه قطعات را تولید کند.

از آنجائیکه در جستجوس نام قطعه هستیم، پس جواب را با جدول S، join می‌کنیم که در آخر نام آن تولید کنندگان را بدست آوریم.

سوال:

شماره فروشنده‌گانی که حداقل تمام قطعات عرضه شده توسط S2 را عرضه می‌کنند؟

$$\pi_{S\#}(\sigma_{S\#='S2'}(\sigma_{S\#='S2'}(S)))$$

سوال:

جداول زیر را در نظر گرفته به پرسش‌ها پاسخ دهید.

Stud (S# و SNAME و city و avg و clg #)

prof (pname و affice و esp و degree و clg#)

crs (c# و cname و unit و clg#)

sec (sec# و c# و s# و term و pname و score)

clg (clg# و clgname و city و pnam)

الف) شماره دانشجویی، نام، کد دانشکده و معدل دانشجویی که میانگین نمرات آنها بالای ۱۵ می‌باشد.

ب) لیست نام اساتیدی که رئیس دانشکده نیستند.

ج) مشخصات کامل روسای دانشکده‌ها.

- (د) دانشجویانی که همه درسهای استاد x را گرفته‌اند.
 (ه) درسهایی که توسط همه دانشکده‌ها ارائه می‌شود.
 (و) مشخصات دروس ۴ واحدی که در نیمسال اول ۷۸ ارائه شده‌اند.

۳- عملگرهای کار بر روی داده‌ها

هر کاربر در محیط پایگاه داده برای انجام عملیات نیازمند، عملگرهای زیر می‌باشد:

- عملگر درج^۱
- عملگر بهنگام سازی^۲
- عملگر حذف^۳

این عملگرها در واقع امکانات حذف، اضافه و اصلاح اطلاعات را برای کاربر فراهم می‌آورند. در ادامه به شرح هریک خواهیم پرداخت.

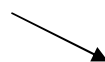
۳-۱ عملگر درج

این عملگر امکان اضافه کردن اطلاعات به جداول اطلاعاتی (رابطه‌ها) را فراهم می‌آورد. این عملگر در حالت کلی امکان افزودن داده‌ها به دو صورت زیر را دارد.

- افزودن مقادیر مشخص بعنوان یک سطر اطلاعاتی به جدول
- افزودن تعدادی سطر اطلاعاتی از جدولی به جدول دیگر

شکل کلی دستور به شرح ذیل می‌باشد:

INSERT source INTO Target



جدول مقصد

همانطور که مشاهده می‌گردد عبارت Source می‌تواند بصورت مجموعه‌ای مشخص از مقادیر که منطبق با جدول مقصد هستند، باشد و یا اینکه خروجی حاصل از یک

1. Insert
 2. Update
 3. Delete

عملیات در پایگاه رابطه‌ای ۱۴۱

جدول و یا خروجی حاصل از اعمال عملگرهای جبر رابطه‌ای بر روی یک یا چند جدول باشد.

مثال:

```
INSERT (S WHERE city = 'london') INTO TEMP
```

آن سطرهایی که city = 'London' است را در رابطه S، را به جدول temp اضافه می‌کند.

۲-۳ عملگر بهنگام سازی

این عملگر امکان اصلاح اطلاعات ذخیره شده در یک جدول را برای کاربر فراهم می‌آورد. شکل کلی دستور به شرح ذیل می‌باشد:

```
UPDATE target Comma list-assignment  
یک انتساب
```

مثال:

```
UPDATE P city = 'Paris' WHERE color = 'Red'
```

عبارت بالا بدین معنی است که از رابطه P اندسته از سطرهایی را که مقدار صفت خاصه رنگ در آنها قرمز است، مقدار صفت خاصه شهر را برابر با پاریس کنیم. این عملگر را بصورت نمادین (نمایش) می‌توان به صورت زیر نشان داد.

(P)

```
WHERE color = 'Red'  
City = 'Paris'
```

لازم بذکر است که عبارت بالا در بعضی از کتب بصورت کامل تر بیان می‌شود که در بحث فعلی مدنظر نیست.

۳-۳ عملگر حذف

این عملگر جهت حذف سطرهایی از اطلاعات درون جداول بکار می‌رود. شکل کلی دستور به شرح ذیل می‌باشد:

```
DELETE target R ← R - E
```

مثال:

DELETE S WHERE STATUS < 20

معنی دستور مذکور بدین صورت است: حذف کن از رابطه S آنهایی را که STATUS آن کمتر از ۲۰ است (پس یک سطر حذف می‌گردد).
توجه: رابطه R و E باید دارای شرط Same arity باشد (تعداد و نوع ستون‌هایشان یکسان است).

۴- کامل بودن جبر رابطه‌ای

همانطور که می‌دانید می‌توان عبارات جبر رابطه‌ای را با استفاده از ترکیب عملگرها نوشت و حاصل هر عبارت معتبر جبر رابطه‌ای باز هم یک رابطه است. لذا می‌توان چنین ارزیابی کرد که هر رابطه معتبر از مجموعه رابطه‌ها را می‌توان به کمک یک عبارت جبر رابطه‌ای نوشت. با این وصف می‌توان گفت که جبر رابطه‌ای از نظر رابطه‌ای کامل است یا به بیانی دیگر جبر رابطه‌ای کمال رابطه‌ای را دارد. بدین علت می‌توان جبر رابطه‌ای را معیار تشخیص کمال رابطه‌ای برای زبان‌های رابطه‌ای دانست. یعنی می‌توان گفت زبانی دارای کمال رابطه‌ای است که حداقل هر رابطه‌ای که با عبارت جبر رابطه‌ای قابل تعریف باشد، توسط آن زبان هم تعریف شدنی باشد.

۵- حساب رابطه‌ای

نوع دیگری از امکانات انجام عملیات در رابطه‌ها، حساب رابطه‌ای است که منطقاً معادل جبر رابطه‌ای می‌باشد. یعنی برای هر عبارت جبر رابطه‌ای، یک عبارت معادل در حساب رابطه‌ای وجود دارد و بالعکس. در اینجا لازم است به این نکته توجه شود که جبر رابطه‌ای، دستوری بوده و شبیه به زبان برنامه نویسی است و حساب رابطه‌ای بصورت توصیفی بوده و به زبان طبیعی نزدیک‌تر است. حساب رابطه‌ای خود دارای دو شاخه می‌باشد:

- حساب تاپلی^۱
- حساب میدانی^۲

در ادامه به شرح هریک از این دو نوع حساب خواهیم پرداخت:

۱-۵ حساب رابطه‌ای تاپلی

در حساب رابطه‌ای تاپلی، یک مفهوم مهم به نام متغیر تاپلی وجود دارد. متغیر تاپلی متغیری است که تنها مقادیر مجازش، تاپل‌های رابطه هستند. این متغیر در شکل کلی به صورت زیر تعریف می‌شود

<range var definition>:: RANGEVAR <range var name> RANGEOVER
<Relational expression commalist>

برای درک بهتر موضوع به عبارات زیر توجه نمایید:

1-RANGEVAR STUD RANGE OVER STT;
2-RANGEVAR STUD RANGEOVER
(STUD WHERE STUD.STMJR = 'comp.eng')

۱-۱-۵ شکل کلی عبارت حساب تاپلی

اگر t یک متغیر تاپلی روی رابطه $R(A_1, A_2, \dots, A_N)$ باشد، در اینصورت شکل کلی عبارت حساب تاپلی بصورت زیر خواهد بود:

(target-items(s)) [WHERE f]

که در آن (target-items(s)) به مفهوم فهرستی از صفات متغیر تاپلی T است:

$T.A_1, T.A_2, \dots, T.A_N$

برای درک بهتر موضوع به مثال‌های زیر توجه نمایید:

ST.STID
ST.STID, ST.STDEID AS W
ST.STID WHERE ST.STDEID = '123'

1. Tuple Oriented
2. Domain Oriented

۵-۱-۲ سور وجودی و سور همگانی

در حساب رابطه تاپلی دو سور وجود دارد:

- سور وجودی^۱
- سور همگانی^۲

سور وجودی

سور وجودی را با $\exists$ نمایش داده و عبارت آنرا بصورت زیر می‌نویسیم

$$\exists T(f)$$

عبارت مذکور بدین معنا است که حداقل یک مقدار برای متغیر T وجود دارد به نحوی که f به مفهوم "درست" ارزیابی شود.

سور همگانی

سور همگانی را با $\forall$ نمایش داده و عبارت آنرا بصورت زیر می‌نویسیم

$$\forall T(f)$$

عبارت مذکور بدین معنا است که به ازاء تمام مقادیر متغیر T ، F به مفهوم "درست" ارزیابی می‌شود.

این دو سور به روش‌های زیر قابل تبدیل به یکدیگر می‌باشند:

$$\begin{aligned} \text{FORALL } T(f) &\equiv \text{NOT EXISTS } T(\text{NOT } f) \\ \text{EXISTS } T(f) &\equiv \text{NOT } (\text{FORALL } T(\text{NOT } f)) \\ \text{FORALL } T((f) \text{ AND } (g)) &\equiv \text{NOT EXISTS } T(\text{NOT } (f) \text{ OR NOT } (g)) \\ \text{FORALL } T((f) \text{ OR } (g)) &\equiv \text{NOT EXISTS } T(\text{NOT } (f) \text{ AND NOT } (g)) \\ \text{EXISTS } T((f) \text{ OR } (g)) &\equiv \text{NOT FORALL } T(\text{NOT } (f) \text{ AND NOT } (g)) \\ \text{EXISTS } T((f) \text{ AND } (g)) &\equiv \text{NOT FORALL } T(\text{NOT } (f) \text{ OR NOT } (g)) \\ \text{FORALL } T(f) &\Rightarrow \text{EXISTS } T(f) \\ \text{NOT EXISTS } T(f) &\Rightarrow \text{NOT FORALL } T(f) \end{aligned}$$

۵-۱-۳ عبارت مطمئن

وقتی از سور وجودی، سور همگانی و سور نفی در نوشتن یک عبارت حساب

1. Existential

2. Universal

رابطه‌ای استفاده می‌کنیم، باید اطمینان حاصل کنیم که عبارت حسابی نوشته شده دارای معنا باشد. یک عبارت حسابی را عبارت مطمئن^۱ گوییم هرگاه نتیجه ارزیابی آن، تعداد محدودی از تاپلها باشد. در غیر اینصورت آن عبارت را عبارت نامطمئن^۲ می‌گوییم.

۵-۲ حساب رابطه میدانی

در این نوع حساب، بجای متغیر تاپلی، متغیر میدانی^۳ داریم. متغیر میدانی متغیری است که از یک میدان مقدار می‌گیرد. تفاوت اصلی حساب میدانی با حساب تاپلی در این است که در حساب میدانی، یک شرط اضافی به نام شرط عضویت^۴ وجود دارد. شرط عضویت بصورت زیر نوشته می‌شود:

$$R(A_1:v_1, A_2:v_2, \dots, A_n:v_n)$$

که در آن R نام رابطه، A_i نام صفت و v_i یک مقدار از میدان و یا یک لیترا است. این شرط به مفهوم "درست" ارزیابی می‌شود اگر و تنها اگر تاپلی در R وجود داشته باشد که مقادیر داده شده برای صفات را داشته باشد (تاپل عضوی از مجموعه بدنه رابطه باشد). برای مثال به عبارت زیر توجه نمایید:

STT (STID: '345', stdeg: 'xwe')

در عبارت فوق این شرط به مفهوم "درست" ارزیابی می‌شود اگر و فقط اگر تاپلی در STT با شماره دانشجویی '345' و رشته کارشناسی 'xwe' وجود داشته باشد.

۶- بهینه سازی پرس و جوها

یکی از مباحثی که در پایگاه‌های داده مطرح است بحث بهینه سازی عبارت پرس و

1. Safe expression
2. Unsafe expression
3. Domain variable
4. Membership condition
5. Optimization Query

جو می‌باشد. از آنجائیکه مبحث بهینه سازی پرس و جو مربوط به دوره کارشناسی نیست، لذا در اینجا صرفاً توضیحات مختصری در اینباره ارائه خواهیم کرد. تمامی تلاش طراحان پایگاه داده‌ها در کوتاه کردن مدت زمان ارسال پاسخ به متقاضی است. در عین حال خروجی یکسان در همه حال مد نظر می‌باشد. بعنوان مثال دو عبارت زیر را در نظر بگیرید:

$$1- \prod_{A, B} B \quad (R) \quad A = B$$

$$2- \prod_{A = B} B \quad (R) \quad A, B$$

۱- ابتدا از رابطه R آن‌هایی که $A = B$ است را جدا می‌کند (سطری)، بعد از کل آنها ستون‌ها را بیرون می‌کشد.

۲- ابتدا از رابطه R دو ستون A و B را جدا می‌کند، بعد آن‌هایی را که $A = B$ است را جدا می‌کند (سطرهایی که $A = B$).

در اینجا ممکن است سئوالاتی برای کاربر مطرح گردد. مانند:

- آیا همه پاسخ‌ها معادل اند؟ بلی
- آیا همه پاسخ‌ها هم سرعت‌اند؟ خیر

بنابراین جهت افزایش سرعت سیستم پایگاه داده سعی بر آن است بهترین روش بکار گرفته شود و یا به عبارت بهتر سعی می‌کنیم تا آنجا که می‌توانیم هر چه سریعتر جداول را کوچک کرده و روی آنها کار کنیم.

یکی از سئوالاتی که مطرح می‌گردد این است که سرعت Quarry های متفاوت چقدر تفاوت دارد؟ در پاسخ باید گفت که: بسیار زیاد، زیرا داده‌ها و جداول در حافظه جانبی قرار دارند. در این راستا ممکن است بگوییم که کاربر باید آنقدر تبه‌ر داشته باشد تا بتواند عبارت پرسو جوی خود را بهینه نماید. ولی آیا واقعاً چنین کاری ممکن است؟ در پاسخ باید گفت: خیر. زیرا:

- کاربر متخصص نیست.

- از سایر جداول و چگونگی آنها مطلع نمی‌باشد.
- اکثریت کاربران از ساختار و مکانیزم درونی پایگاه داده بی اطلاع هستند.

پس بهترین راه حل این است که بگوییم DBMS باید Quarry را بهینه سازی نماید. طبیعی است که باید مکانیزم مشخصی برای اینکار وجود داشته باشد. هرچند که بحث بهینه سازی پرسو جو در این کتاب نمی‌گنجد، ولی برای ارائه یک شمای ساده از این عمل الگوریتم ذیل را (که بسیار ساده و مختصر در نظر گرفته شده است) ارائه می‌نماییم.

۱-۶ تبدیل Q به Q.o

برای بهینه سازی پرس و جوها لازم است تا رویه مشخصی در نظر گرفته شود. در ادامه قواعد مربوط به بهینه سازی پرس و جوها مطرح شده است:

قاعده اول: B را هر چه سریعتر (زودتر) انجام دهیم.

قاعده دوم: شرطهای ترکیبی را به شرطهای سوالی تبدیل نمایید (شرطهای دارای OR، AND و... را تا حد امکان بشکنید).

$$B \quad (R) \quad B \quad B \left(\begin{array}{c} (R) \\ p2 \end{array} \right) \\ p1 \cap p2 \rightarrow p1$$

قاعده سوم: Π را زودتر انجام دهیم (پس از B).

۲-۶ سایر قواعد و عملگرهای بهینه سازی

قاعده اول:

$$a \bowtie b = b \bowtie a$$

قاعده دوم:

$$(a \bowtie b) \bowtie c = a \bowtie (b \bowtie c)$$

توجه: لازم بذکر است که عبارات این دو عبارات قاعده از لحاظ زمان با هم برابر نیستند.

مثال: مشخصات کامل دروس و گروه‌های درسی آنها را بدست آورید.

در اینجا این سؤال مطرح می‌شود که کدام یک از دو عبارت زیر سریعتر پاسخ می‌دهند؟

$$\text{crs} \infty \text{sec}$$

یا

$$\text{sec} \infty \text{crs}$$

این دو دستور از نظر سرعت با یکدیگر بسیار متفاوتند. زیرا سایز دو جدول کاملاً متفاوت می‌باشد جدول crs (فقط مشخصات درس‌ها می‌باشد) معمولاً کوچک است و جدول sec بسیار بزرگتر از آن، زیرا هم گروه‌های مختلف دروس را شامل می‌شود و هم دانشجویان آنها را. فرض کنید جدول crs در حافظه cash جا بگیرد، در این صورت الگوریتم‌های دو راه حل بالا را بررسی می‌کنیم:

$$\text{crs} \infty \text{sec}$$

الگوریتم (۱): برای هر سطر جدول crs

برای هر سطر جدول sec

مقایسه کن

انتخاب کن

در الگوریتم یک باید سطرهای بسیار زیاد جدول sec را به دفعات وارد حافظه اصلی کنیم و مقایسه و انتخاب را انجام دهیم به عبارت دیگر تعداد دستیابی به دیسک (منظور حافظه جانبی) به اندازه حاصلضرب سایز دو جدول است.

$$\text{sec} \infty \text{crs}$$

الگوریتم (۲): برای هر سطر جدول sec

برای هر سطر جدول crs

مقایسه کن

انتخاب کن

در الگوریتم دو، هر سطر sec را فقط یکبار به حافظه اصلی می‌آوریم زیرا جدول crs به طور کامل در حافظه cash قرار دارد و همه مقایسه‌ها یکبار انجام می‌گیرد و تعداد دستیابی به دیسک به اندازه سایز جدول sec است. بطور مثال اگر سایز جدول crs را

عملیات در پایگاه رابطه‌ای ۱۴۹

۱۰۰ و جدول sec را ۱۰۰۱ فرض کنیم الگوریتم اول ۱۰۰۰۰۰۰ بار بیشتر به دیسک دستیابی پیدا می‌کند.

تمرینات

۱. چه امکاناتی برای انجام عملیات در مدل رابطه‌ای وجود دارند؟

۲. عملگرهای جبر رابطه‌ای را نام ببرید؟
۳. عملگرهای Select و Project چه تفاوتی با یکدیگر دارند؟
۴. عملگر پیوند را توضیح داده و خواص آنرا نام ببرید؟
۵. عملگرهای کار بر روی داده‌ها را نام ببرید؟

فصل ۸

آشنایی با زبان رابطه‌ای SQL

هدف کلی

در این فصل ابتدا تاریخچه‌ای از زبان SQL و دلایل گستردگی آن ارائه خواهد شد. سپس دستورات تعریف داده‌ها مانند تعریف شما و جدول و... مطرح شده و در ادامه دستورات مربوط به پرس‌وجوی داده‌ها بیان خواهد شد. پس از توضیح این گروه از دستورات، دستورات کار با داده‌ها شرح داده خواهند شد. پس از آن جستجوهای پیشرفته‌تر مطرح و در این راستا عملگرهای پیوند و سایر عملگرهای مرتبط با جستجوهای پیشرفته بیان خواهند شد. در پایان نیز درباره دستورات کنترل مجوزهای دسترسی صحبت خواهد شد.

هدف رفتاری

در این فصل عناوین زیر مورد بحث و بررسی قرار می‌گیرند:

- مقدمه‌ای بر SQL
- دلایل گستردگی SQL
- دستورات تعریف داده‌ها DDL
- تعریف پایگاه داده
- تعریف شما
- تعریف جدول
- دستورات پرس و جو (مشاهده) داده‌ها

- دستورات کار بر روی داده‌ها
- جستجوهای پیشرفته با عملگر پیوند
- عملگر گروه بندی
- عملگر مرتب سازی
- توابع تجمعی
- استفاده از پرس و جوهای تودرتو
- استفاده از زیرپرسش‌ها (پرسش‌های فرعی)
- دستورات کنترل مجوز دسترسی

۱ - مقدمه‌ای بر SQL

یک زبان رابطه‌ای قبل از هر چیز باید بتواند در محیط انتزاعی رابطه عمل نماید و طبیعتاً باید بر اساس ریاضیات رابطه‌ای مانند جبر رابطه‌ای استوار باشد. در این فصل زبان SQL^۱ بعنوان یک زبان رابطه‌ای استاندارد^۲ شده و شاید بعنوان بهترین زبان رابطه‌ای مورد بحث و بررسی قرار می‌گیرد.

اولین نسخه این زبان بعنوان قسمتی از سیستم R^۳ در اواخر دهه ۱۹۸۰ توسط شرکت IBM طراحی شد. از آنجائیکه پس از مدتی شرکت‌ها و اشخاص مختلف هر کدام اقدام به ایجاد یک نسخه از زبان SQL با اسامی مختلف کردند، لذا در سال ۱۹۸۴ موسسه استانداردهای ملی آمریکا (ANSI) و سازمان استانداردهای بین‌المللی (ISO) نسخه استاندارد زبان SQL را ارائه کردند و در سال ۱۹۸۷ اولین نسخه استاندارد این زبان بنام ANSI/ISO SQL-89 را که اصطلاحاً به آن SQL1 می‌گفتند، ارائه کردند. پس از گذشت چند سال و با توجه به توسعه توانایی‌ها و انتظارات از این زبان، نسخه SQL-92 به نام SQL2 توسط این موسسه‌ها ارائه گردید. با ورود مفاهیم شیء گرایی به محیط پایگاه‌های داده رابطه‌ای، زبان SQL-99 که به SQL3 معروف است، توسط موسسه‌های مذکور ارائه گردید. امروزه نسخه جدیدی از زبان استاندارد SQL بنام SQL-2003 نیز

1. Standard Query Language

2. Standard Relational Language

3. R System

آشنایی با زبان رابطه‌ای SQL ۱۵۳

ارائه شده است که هنوز در پایگاه‌های داده رابطه‌ای مورد استفاده قرار نگرفته است. مفاهیم اصلی زبان SQL در کلیه زبان‌های مذکور یکسان هستند.

۲ - دلایل گستردگی SQL

شرکت‌های فعال در زمینه پایگاه‌های داده رابطه‌ای با توجه به نیاز خود اقدام به طراحی زبانهای رابطه‌ای کرده‌اند که اصولاً همگی از استانداردهای ANSI/ISO پشتیبانی می‌کنند. این زبان‌ها مفاهیم زبان SQL را بصورت کامل پشتیبانی کرده و علاوه بر آن دستورات دیگری نیز برای ساختار خود دارند. امروزه اکثریت پایگاه‌های داده رابطه‌ای زبان SQL2 را بعنوان استاندارد زبان پایگاه‌داده خود انتخاب کرده‌اند. با توجه به اینکه پیاده سازی زبان SQL3 بعنوان یک زبان رابطه‌ای شیء گرا دارای پیچیدگی‌های خاص خود می‌باشد، لذا امروزه از بین پایگاه‌های داده رابطه‌ای تنها پایگاه‌داده اراکل^۱ محصول شرکت اراکل، قادر به پشتیبانی از زبان SQL3 می‌باشد. بصورت کلی می‌توان اهم دلایل گستردگی استفاده از زبان SQL را به شرح زیر عنوان کرد:

- مورد قبول همه است (همه متخصصین از آن استفاده می‌کنند و آنرا قبول می‌کنند).
- شبه رویه‌ای است (زبان پرس و جوئی است).
- به صورت تعبیه شده (توکار^۲) در دیگر محیط‌های برنامه سازی است. مانند محیط Access، Delphi، Oracle و....

برخی امکانات مهمتر زبان SQL به شرح ذیل می‌باشند که بعضی از آنها در زبان SQL1 وجود نداشته و به مرور به زبان‌های SQL2 و SQL3 افزوده شده‌اند:

- دستورات تعریف داده‌ها
- دستورات پرس و جو (مشاهده) داده‌ها
- دستورات کار بر روی داده‌ها

1. ORACLE

2. Embedded

- دستورات کنترل مجوز دسترسی
- دستورات نوشتن رویه‌ها و توابع
- دستورات کنترل جامعیت
- دستورات کنترل تراکنش‌ها

با توجه به اینکه چهار دسته اول دستورات ذکر شده تقریباً در همه پایگاه‌های داده موجود می‌باشد لذا در ادامه چهار سری دستورات اول را به تفصیل مورد بحث و بررسی قرار خواهیم داد. بعضی از پایگاه‌های داده و یا زبان‌های برنامه نویسی (که زبان SQL را درون خود دارند)، مجموعه دستورات نوشتن رویه‌ها و توابع، کنترل جامعیت و کنترل تراکنش‌ها را یا ندارند و یا در اختیار کاربر قرار نمی‌دهند. لذا این نوع دستورات در این کتاب توضیح داده نمی‌شوند.

۳ - دستورات تعریف داده‌ها DDL^۱

این دستورات در حالت کلی شامل تعریف پایگاه‌داده، تعریف شما^۲، تعریف کاربر^۳ و تعریف جداول^۴ اطلاعاتی و دیده‌ها می‌باشند.

۳-۱ تعریف پایگاه‌داده

شکل کلی دستور به شرح ذیل می‌باشد:

CREATE DATABASE database -name

با استفاده از این دستور پایگاه‌داده مورد نظر را ایجاد می‌نمائیم که خود پایگاه‌داده می‌تواند شامل جداول متعددی باشد. لازم به ذکر است که تعریف پایگاه‌داده اساساً دارای پارامترهای بسیاری است که موضوع این کتاب نیست.

1. Data Definition Language

2. Schema

3. User Definition

4. Table Definition

۳-۲ تعریف شما

این تعریف در حالت کلی شامل تعریف کاربر، تعریف جداول و تعریف دیدها و بسیاری مفاهیم دیگر می‌باشد و ساختار آن بسته به پایگاه داده انتخاب شده متفاوت می‌باشد. شکل کلی دستور (که در آن صرفاً تعاریف Table و View آمده است) به شرح ذیل می‌باشد.

```
Schema ::= CREATE SCHEMA
          AUTHORIZATION User
          { Schema-element-list }
Schema-element ::= base-table definition
                  | View definition
                  | grant-operation
```

جهت حذف یک شما از عبارت زیر استفاده می‌شود:

```
DROP SCHEMA Name [ Restrict | Cascade ]
```

۳-۳ جدول

۳-۳-۱ تعریف جدول

با این دستور امکان تعریف جداول اطلاعاتی برای کاربر فراهم می‌شود. شکل کلی دستور به صورت زیر می‌باشد:

```
Base-table definition ::= CREATE TABLE base-table
                        (base-table-element commalist)
Base-table-element ::= column-definition
                     | Unique constraint-definition
Column-definition ::= column data-type [ NOT NULL [ UNIQUE ] ]
Unique Constraint-definition ::= UNIQUE (Column-Cammalist)
```

برای مثال فرض کنید می‌خواهیم یک جدول شامل شماره و نام قاره‌های مختلف جهان درست کنیم. عبارت ایجاد کننده جدول مذکور به شرح زیر می‌باشد:

```
CREATE TABLE Continent
(CntId int ,
CntName char(30))
```

حال چنانچه بخواهیم عبارت مذکور را کامل تر کنیم لازم است محدودیت‌هایی را به آن اضافه نماییم.

```
CREATE TABLE Continent
(CntId int Unique,
CntName char (30) NOT NULL)
```

لازم به ذکر است که در 2-SQL کلمات کلیدی دیگری به عبارت ایجاد جدول افزوده شده‌اند بگونه‌ایکه امکان افزودن مقدار اولیه به یک صفت خاصه و یا امکان افزودن محدودیت‌ها به جدول در هنگام ایجاد جدول فراهم شده است. برای مثال با استفاده از کلمه کلیدی Default در جلوی یک صفت خاصه می‌توان مقدار پیش فرض را برای آن ستون تعیین نمود. در عبارت زیر مقدار پیش فرض ستون سال برابر با ۱۳۸۴ در نظر گرفته شده است:

```
CREATE TABLE Info
(InfoId int unique,
InfoYear int default 1384,
InfoName char (50) NOT NULL)
```

تمرین:

سیستم پایگاه‌داده دانشگاه را به صورت زیر در نظر بگیرید، طراحی به زبان SQL آنرا کامل نمائید:

```
Stud (S#, Sname, S -add, S -tel)
Prof (P#, Pname, P -add, P -off)
Crs (C#, Cname, unit)
Enroll(S#, C#, Sec#, term, score)
Sec (C#, Sec#, term, time, place, P#)
```

۳-۲-۳ اصلاح ساختار جدول

یکی از ملزومات اساسی در مورد جداول، امکان اصلاح ساختار جداول و حذف و اضافه کردن ستونهای مربوط به آن می‌باشد. بدیهی است که در مواقعی لازم می‌شود تا ستونهای جدیدی (صفات خاصه) به ساختار جدولی که از قبل طراحی کرده‌ایم اضافه گردد یا ستون‌هایی از جدول مذکور حذف گردد. همچنین این امکان نیز

آشنایی با زبان رابطه‌ای SQL ۱۵۷

وجود دارد که در مواردی تشخیص دهیم که نوع داده مربوط به یک ستون باید تغییر یابد. برای این منظور از دستور ALTER TABLE استفاده می‌شود که ساختار کلی این دستور در ذیل آمده است:

```
ALTER TABLE tablename  
{ ADD [COLUMN] column definition }  
| { ALTER [COLUMN] column name  
{ SET DEFAULT default option } | { DROP DEFAULT } }  
| { DROP [COLUMN] column name }  
| { ADD table constraint definition }  
| { DROP CONSTRAINT constraint name };
```

همانطور که در بالا مشاهده می‌شود لیست تغییرات قابل اعمال بر روی یک جدول عبارتند از:

- افزودن یک ستون به جدول
- حذف یک ستون از جدول
- افزودن یک محدودیت به جدول
- حذف یک محدودیت از جدول
- افزودن مقدار اولیه به یک ستون
- حذف مقدار اولیه یک ستون

ممکن است در اینجا این سؤال به ذهن برسد که چگونه می‌توان نوع داده‌ای یک ستون را تغییر داد. در اینجا لازم است به این نکته اشاره شود که تغییر نوع داده‌ای یک ستون اساساً در بسیاری از پایگاه‌های داده در نظر گرفته نشده است و این مهم بواسطه نوع قرار گرفتن مقادیر مربوط به ستون‌ها در فضای فیزیکی حافظه می‌باشد بگونه‌ای که حتی در بعضی پایگاه‌های داده امکان تغییر ترتیب ستون‌ها در یک جدول وجود ندارد. امکان تغییر نوع داده مربوط به یک ستون فقط در بعضی از پایگاه‌های داده و بصورت منطقی طراحی شده است. حال در ادامه به منظور آشنایی بهتر با این نوع دستور چند مثال مختلف را نمایش می‌دهیم. فرض کنید می‌خواهیم ستونی به نام fname و از نوع رشته‌ای را به جدولی به نام SalesPeople اضافه نماییم. عبارت SQL مربوطه به شرح ذیل خواهد بود:

```
ALTER TABLE SalesPeople ADD fname char(10);
```

همانطور که مشاهده می‌نمایید کلمه کلیدی COLUMN بعد از کلمه ADD نوشته نشده است. در بعضی از پایگاه‌های داده استفاده از کلمه کلیدی COLUMN در این شرایط اجباری نمی‌باشد، حال آنکه در بعضی دیگر اجباری است. حال به مثال دیگری توجه نمایید که در آن مقدار پیش فرض برای یک ستون پایگاه‌داده تعریف می‌شود.

```
ALTER TABLE Salespeople
ALTER COLUMN city
ADD DEFAULT 'London';
```

۳-۳-۳ تغییر نوع داده‌های یک ستون جدول

همانطور که در بالا اشاره شد تغییر نوع داده یک ستون از جدول در هر پایگاه‌داده‌ای امکان پذیر نیست. با توجه به اینکه این دستور در بعضی پایگاه‌های داده فراهم شده است لذا شکل دستور مذکور در ذیل آمده است.

```
ALTER TABLE tablename
MODIFY (< col -name > < new type >)
```

مثال: این دستور تعداد کاراکترهای صفت درس را به 40 تغییر می‌دهد.

```
ALTER TABLE crs
MODIFY (Cname char(40));
```

توجه: تغییر نوع داده در اکثر SQLها غیر مجاز است و در بعضی از پایگاه‌های داده قبل از تایید عمل مذکور اقدام به بررسی محتوای داده‌های ستون مورد نظر می‌گردد و در صورت قابل قبول بودن عمل تبدیل نوع این درخواست تائید می‌گردد. بعنوان مثال تغییر نوع داده‌ای از کاراکتر به عدد فقط در صورتیکه تمامی مقادیر یک ستون از نوع عدد باشند امکان پذیر است و چنانچه حتی یک مقدار شامل حروف باشد، انجام این کار غیر ممکن است.

۳-۳-۴ حذف یک جدول

در مواردی ممکن است لازم باشد یک جدول از پایگاه‌داده حذف گردد. برای انجام

اینکار دستور زیر موجود می‌باشد. لازم به ذکر است که در هنگام حذف یک جدول باید بررسی گردد که چنانچه جداول دیگری به این جدول مرتبط باشند، چه عملی باید بروی جداول مرتبط انجام گیرد و آیا اساساً امکان حذف جدول وجود دارد یا خیر. شکل کلی دستور مذکور به شرح زیر می‌باشد:

DROP TABLE tablename [RESTRICT | CASCADE];

۴- دستورات پرس و جو (مشاهده) داده‌ها

یکی از مهمترین خواسته‌های کاربران هر پایگاه داده، امکان بازیابی داده‌های ثبت شده در جداول می‌باشد. به این منظور دستوراتی ارائه شده‌اند که امکان بازیابی سطرها یا ستون‌هایی از یک جدول و یا چند جدول اطلاعاتی را فراهم می‌آورند که به آنها اصطلاحاً زبان پرس و جوی داده‌ها^۱ می‌گویند. مهمترین دستور از این نوع دستورات دستور SELECT می‌باشد. این دستور یکی از پیچیده ترین دستورات در زبان SQL می‌باشد و خود به تنهایی شامل دستورات متنوعی می‌باشد. شکل کلی دستور به شرح ذیل می‌باشد:

```
SELECT [DISTINCT]
{ value expression
[ AS column name ] } , ... }
| { qualifier.* }
| *
FROM { { table name [ AS ] [ correlation name ] } , ...
[ WHERE predicate ]
[ GROUP BY [ table name | correlation name ].column name
[ COLLATE collation name ] ]
[ HAVING predicate ]
[ { UNION | INTERSECT | EXCEPT } [ ALL ]
Select statement | { TABLE table name }
| table value constructor ]
[ ORDER BY { { out put column | positive integer } [ ASC | DESC ] , ... } ] ;
```

همانطور که در عبارت بالا مشاهده می‌گردد این دستور بسیار پیچیده و کلی می‌باشد و اساساً دارای زیر دستوراتی می‌باشد که در بحث این کتاب نمی‌گنجد. برای

1. Data Query Language (DQL)

راحتی بیشتر خوانندگان، سعی می‌کنیم در ابتدا مثالهای ساده تر در مورد این دستور را مورد بررسی قرار دهیم و در ادامه همین فصل در خصوص حالات پیچیده تر این دستور و طریقه بازیابی داده‌ها از چندین جدول اطلاعاتی را شرح خواهیم داد. حالت ساده شده دستور مذکور بصورت زیر قابل نمایش است که به آن اصطلاحاً SFW نیز می‌گویند که حروف اول هر قسمت می‌باشد.

```
SELECT column name , ...
FROM table name
WHERE criteria
```

که در آن column name نام ستون‌ها، table name نام جدول و criteria شرط مورد نظر برای انتخاب و بازیابی رکوردها می‌باشد.

در حالت بسیار ساده می‌توان فرض کرد که می‌خواهیم رکوردهای یک جدول (مثلاً جدول اطلاعات پرسنلی EMP) را بازیابی نماییم. فرض کنیم جدول EMP دارای ساختار زیر باشد و بغیر از صفت خاصه EMPNO که کلید اصلی می‌باشد سایر مقادیر می‌توانند مقدار NULL را بپذیرند:

Name	نام صفت خاصه
EMPNO	شماره پرسنلی
ENAME	نام پرسنل
JOB	شغل
MGR	شماره مدیر
HIREDATE	تاریخ استخدام

در عبارت ذیل می‌خواهیم کلیه رکوردهای جدول را بازیابی نماییم. عبارت مورد نظر به شرح ذیل می‌باشد:

```
Select *
From EMP;
```

همانطور که در عبارت بالا مشاهده می‌شود علامت * به معنای انتخابی تمامی ستون‌های مربوط به سطرها می‌باشد. این دستور می‌تواند به شرح ذیل نیز باشد:

آشنایی با زبان رابطه‌ای SQL ۱۶۱

```
Select EMP.*  
From EMP ;
```

همانطور که مشاهده می‌گردد در این دستور قبل از استفاده از علامت *، نام جدول نیز نوشته شده است که در این مورد بعداً توضیح خواهیم داد. حال در ادامه فرض کنیم می‌خواهیم فقط ستون‌های شماره پرسنلی و نام پرسنل را برای تمامی پرسنل بازیابی نماییم. عبارت مذکور به شرح ذیل می‌باشد:

```
Select empno , ename  
From EMP ;
```

می‌توانستیم قبل از اسامی ستونها، نام جدول را نیز بکار ببریم که از این کار اجتناب کردیم. حال فرض کنیم می‌خواهیم بدانیم که اگر حقوق هر یک از پرسنل را ۱۰٪ افزایش دهیم چه میزانی حقوق برای هر یک از پرسنل خواهیم داشت. عبارت مذکور به شرح زیر خواهد بود:

```
Select empno,ename,sall * 0.1  
From emp;
```

چنانچه بخواهیم عبارت را واضح تر بیان کنیم لازم است که برای عبارت $sall * 0.1$ یک نام مستعار مانند NEWSALL در نظر بگیریم که عبارت مذکور به شرح زیر خواهد بود.

```
Select empno,ename,sall * 0.1 as newsall  
From emp;
```

حال برای مشخص شدن عبارت WHERE (بیان کننده شرط) مثال دیگری را مطرح می‌کنیم. فرض کنیم می‌خواهیم اطلاعات کسانی را مشاهده کنیم که میزان حقوق آنها از ۱۰۰۰۰ تومان بیشتر باشد. به عبارت زیر توجه نمایید.

```
SELECT *  
FROM emp  
WHERE sall > 10000 ;
```

در این عبارت جلوی کلمه WHERE عبارت شرط مطرح شده است. چنانچه صرفاً بعضی از صفات خاصه مطرح باشند می‌توان بجای استفاده از علامت *، اسامی ستونها (صفات خاصه) را نوشت.

یکی از نکات بسایر مهم این است که لازم است ترتیب سه کلمه SELECT و FROM و WHERE رعایت شود. چنانچه ترتیب مذکور رعایت نگردد سیستم با پیغام خطا مواجه می‌گردد. همچنین می‌توان در عبارات، کلمه WHERE را در صورت نبودن شرط بکار نبرد، ولی حتماً باید کلمات کلیدی SELECT و FROM در عبارت باشند.

تذکر: در بعضی شرایط امکان دارد که که کاربر بخواهد مقداری را در سیستم محاسبه نماید که اصلاً ارتباطی با هیچ جدولی ندارد. از آنجاییکه لازم است حتماً بعد از کلمه FROM نام جدول ذکر گردد، لذا در بعضی از پایگاه‌های داده یک جدول فرضی تعریف شده است. مثلاً در پایگاه‌داده ORACLE نام جدول فرضی DUAL می‌باشد. به عبارت ذیل توجه نمایید:

```
SELECT 50 * 147
FROM DUAL ;
```

تا این مرحله از دستور SELECT بصورت بسیار ساده مورد بررسی قرار گرفت. قسمت‌های پیشرفته تر این دستور در ادامه همین فصل مورد بررسی قرار خواهد گرفت. در ادامه دستورات کار بر روی داده‌ها را مورد بررسی قرار خواهیم داد.

۵ - دستورات کار بر روی داده‌ها

یکی از بدیهی ترین نیازها برای کاربران، وجود امکان کار بر روی داده‌ها می‌باشد که متناسب با این نیاز دستوراتی تحت عنوان زبان کار با داده‌ها^۱ طراحی شده است. این عملیات می‌تواند شامل موارد زیر باشد:

- INSERT - افزودن رکورد (هایی) به یک جدول
- UPDATE - اصلاح اطلاعات موجود در یک جدول
- DELETE - حذف رکورد (هایی) از جدول

هر یک از این عملیات دارای دستورات خاص خود می‌باشند که در ادامه به شرح

1. Data Manipulation Language (DML)

هر یک خواهیم پرداخت.

۵-۱ دستور INSERT

این دستور به منظور وارد کردن (افزودن) رکورد یا رکوردهایی به یک جدول تعریف شده است. شکل کلی دستور به شرح ذیل می‌باشد:

```
INSERT INTO table name  
[ (column name , ...) ]  
{ VALUES (value , ...) }  
| sub query ;
```

همانطور که در دستور بالا مشاهده می‌گردد، در انتهای دستور از عبارت sub query استفاده شده است که این دستور به منظور انتخاب چندین رکورد از یک جئول و افزودن یکباره آنها به جدول مذکور می‌باشد. در ابتدا برای آشنایی خوانندگان با این دستور، افزودن یک رکورد به جدول را مورد بررسی قرار می‌دهیم. در دستور بالا در ابتدا نام جدول را که می‌خواهیم مقادیری را به آن بیافزاییم وارد می‌کنیم و سپس اسامی ستونهای از رکورد را که می‌خواهیم مقادیری را در آنها وارد نماییم را نوشته و سپس مقادیر مرتبط با هر ستون را وارد می‌نماییم. برای مثال فرض کنیم می‌خواهیم جدولی بنام SalesPeople که دارای چهار ستون می‌باشد، یک رکورد با مقادیر مشخصی ذیل وارد نماییم. عبارت مذکور به این شرح می‌باشد:

```
INSERT INTO Salespeople  
VALUES (1001,'Peel','London', 12);
```

حال فرض کنیم مثلاً نمی‌دانیم که شخص مذکور در شهر لندن اقامت دارد و می‌خواهیم رکورد مذکور را وارد نماییم بدون آنکه کقداری برای صفت خاصه شهر در نظر بگیریم. عبارت مذکور به شرح زیر می‌باشد:

```
Insert Into salespeople  
Values (1001,'Peel', NULL, 12);
```

توجه: برای آندسته از ستونهای که بعنوان کلید اصلی هستند و یا اینکه حتماً باید مقداری (غیر از NULL) را داشته باشند، لازم است مقداری در نظر گرفته شود، در غیر اینصورت امکان افزودن رکورد جدید وجود ندارد.

لازم به ذکر است که ترتیب وارد کردن مقادیر به ترتیب تعریف ستونها فرض شده است. چنانچه بخواهیم ترتیب ورود مقادیر برای ستونها را تغییر دهیم و یا اینکه صرفاً بخواهیم بجای قرار دادن مقدار NULL برای یک ستون، اصلاً نام آن ستون را بکار ببریم، لازم است نام ستونها به ترتیب مورد نظر در جلوی نام جدول وارد شده و سپس به همان ترتیب، مقادیر مرتبط وارد شوند. برای توضیح بیشتر فرض کنیم جدولی بنام Customers داریم که فقط می‌خواهیم در ستون‌های city و cname و cnum آن جدول مقادیری را وارد نماییم. به عبارت ذیل توجه نمایید:

```
INSERT INTO Customers (city, cname, cnum)
VALUES ('London','Hoffman', 2001);
```

تا این مرحله صرفاً یک رکورد به جدول وارد شده است. چنانچه بخواهیم تعدادی رکورد را از جدول دیگری انتخاب و به جدول مذکور وارد نماییم، بجای استفاده از کلمه Values، لازم است تا یک عبارت Select را وارد نماییم. برای روشن شدن این مطلب به عبارت زیر توجه نمایید:

```
INSERT INTO LondonStaff
SELECT *
FROM salespeople
Where city = 'London';
```

حال فرض کنید می‌خواهیم صرفاً دو ستون بنام‌های FDate و Total از جدولی بنام DayTotals را با مقادیر انتخاب شده از جدول دیگری پر کنیم. مثال زیر بیانگر این مطلب می‌باشد:

```
Insert Into daytotals (fdate,total)
Select odate,income
From orders ;
```

نکته بسیار مهم در نوشتن این عبارت این است که تعداد ستونهای انتخاب شده در عبارت Select باید برابر با تعداد ستونهای جدول اصلی باشد.

۲-۵ دستور UPDATE

آشنایی با زبان رابطه‌ای SQL ۱۶۵

یکی از مهمترین نیازهای کاربران، توانایی اعمال تغییرات در داده‌هایی است که قبلاً در سیستم وارد کرده‌اند. این نیاز ناشی از تغییر داده‌ها و یا وادر کردن مقدار برای ستونهایی است که قبلاً هیچ مقداری برای آن در نظر گرفته نشده بود. دستور UPDATE به منظور اعمال تغییرات در مقادیر رکوردهای ثبت شده در جداول طراحی شده و شکل کلی آن به شرح ذیل می‌باشد:

```
UPDATE table name
SET { column name = { value expression
    | NULL
    | DEFAULT } } , ...
[ { WHERE predicate } ] ;
```

در ادامه برای تشریح بهتر دستور مذکور مثالهایی آورده شده است که حالات مختلف استفاده از دستور را نشان می‌دهد. در ساده ترین حالت فرض کنید می‌خواهیم مقدار صفت خاصه rating از جدول Customers را برابر با ۲۰۰ کنیم.

```
UPDATE customers
SET rating = 200;
```

همانطور که در بالا مشاهده می‌کنید هیچ نوع شرطی برای انتخاب رکوردها در نظر گرفته نشده است. لذا با اجرا دستور بالا مقدار صفت خاصه rating برای تمامی رکوردها برابر با ۲۰۰ می‌شود. در مثال زیر یک شرط را جهت بروز رسانی جدول اضافه می‌کنیم:

```
UPDATE customers
SET rating = 200
WHERE snum = 1001;
```

تا این مرحله صرفاً یک صفت خاصه از جدول را بروز رسانی کرده‌ایم. در ادامه طریقه بروز رسانی چند صفت خاصه بصورت همزمان را مشاهده خواهیم کرد.

```
UPDATE Salespeople
SET sname = 'Gibson', city = 'Boston', comm. = 0.1
WHERE snum = 1004;
```

همانطور که مشاهده می‌کنید هر یک از صفات خاصه با یک کاما از دیگری جدا شده است. تا این قسمت از کار معمولاً مقدار یک صفت خاصه را تغییر داده‌ایم. ولی

این امکان نیز وجود دارد که مقدار جدید، عبارتی شامل صفات خاصه نیز باشد. برای مثال ممکن است بخواهیم مقدار یک صفت خاصه را دو برابر کنیم.

```
Update salespeople
SET comm = comm * 2;
```

همانطور که مشاهده می‌شود مقدار جدید در واقع از پردازش بر روی مقدار قبلی همان صفت خاصه بدست آمده است. این امکان وجود دارد که از مقادیر سایر صفات خاصه نیز استفاده شود. حال فرض کنید می‌خواهیم مقدار یک صفت خاصه را تبدیل به NULL (مقدار هیچ) نماییم. عبارت زیر طریقه انجام این کار را نشان می‌دهد.

```
UPDATE customers
SET rating = NULL
WHERE city = 'London';
```

عمل تبدیل یک مقدار به NULL در واقع حذف مقدار یک ستون از رکورد می‌باشد که در اینجا به آن بروز رسانی می‌گوییم. مفهوم حذف یک مقدار از یک ستون با مفهوم حذف یک رکورد (سطر) متفاوت می‌باشد. در ادامه دستور DELETE را جهت حذف رکوردها توضیح خواهیم داد.

۳-۵ دستور DELETE

ممکن است در مواقعی کاربران بخواهند رکورد (هایی) را از جدول اطلاعاتی حذف نمایند. در واقع کل اطلاعات یک رکورد (و نه فقط بعضی ستون‌ها) حذف خواهد شد. برای این منظور دستور DELETE طراحی شده است. شمای کلی این دستور در زیر آمده است:

```
DELETE FROM table name
{ WHERE predicate } ;
```

برای درک بهتر دستور، مثالهایی در ذیل آمده است. در ابتدا لازم بذکر است چنانچه در هنگام استفاده از دستور DELETE، شرطی آورده نشود بدین معنا است که کلیه اطلاعات موجود در آن جدول حذف شود. برای مثال دستور ذیل بیانگر حذف کلیه رکوردهای موجود در جدول Customers می‌باشد.

```
DELETE FROM customers;
```

حال در مثالی دیگر یک عبارت شرطی جهت تعیین رکوردهای مورد نظر جهت حذف نشان داده شده است.

```
DELETE FROM salespeople  
WHERE snum = 1003;
```

لازم به ذکر است که عبارت شرط می‌تواند یک یا تعدادی از رکوردها را در برگیرد. با این وصف دستور DELETE توانایی حذف یک یا چند رکورد یا تمامی رکوردهای یک جدول را دارد.

۶- جستجوهای پیشرفته با عملگر پیوند^۱

تا بحال کلیه مثال‌های ارائه شده در این فصل بر اساس یک جدول طراحی شده بودند. یکی از انتظارات بسیار مهم کاربران یک پایگاه داده، امکان بازیابی اطلاعات از جداول مرتبط با هم می‌باشد. به این منظور عملگر پیوند بین جداول در زبان SQL در نظر گرفته شده است. این عملگر در واقع جزئی از دستور SELECT می‌باشد. در واقع عملگر پیوند باعث ایجاد ارتباط بین جداول بر اساس کلید اصلی و کلید خارجی بین آنها می‌باشد. البته در مواردی نیز ممکن است که ارتباط بین جداول بر اساس یک عبارت خاص یا یک ترکیب شرطی باشد. در حالت کلی عملگر پیوند به دو صورت می‌تواند باشد:

- عملگر پیوند داخلی^۲
- عملگر پیوند خارجی^۳

این عملگرها هریک دارای انواع مختلفی می‌باشد که در ادامه به شرح هر یک خواهیم پرداخت:

1. Join
2. Inner Join
3. Outer Join

۶-۱ عملگر پیوند داخلی

این عملگر در واقع متداول ترین نوع پیوند بین دو جدول می‌باشد که بر اساس آن تمامی رکوردهایی از دو جدول که در شرط تعریف شده در پیوند صدق می‌کنند، مورد بررسی قرار می‌گیرند. به بیانی دیگر اگر جدول (رابطه) سمت چپ در پیوند را R1 و جدول سمت راست در پیوند را R2 بنامیم، صرفاً تمامی رکوردهایی از دو جدول که شرط تعریف شده در پیوند در مورد آنها صادق باشد، مورد بازیابی قرار خواهند گرفت.

R1 JOIN R2 ON criteria
OR
R1 INNER JOIN R2 ON criteria

استفاده از عبارت Inner Join به این علت است که در بعضی از زبان‌ها این نوع پیوند را پیوند داخلی می‌نامند. برای درک بهتر دو جدول اطلاعات پرسنل (EMP) و ساختار سازمانی (DEPT) را به صورت آنچه که در زیر آمده است در نظر می‌گیریم:

جدول ۸-۱ جدول اطلاعات پرسنل EMP

EMP			
EMPNO	ENAME	SAL	DEPTNO
7369	SMITH	800	20
7499	ALLEN	1600	30
7521	WARD	1250	30
7566	JONES	2975	20
7654	MARTIN	1250	30
7698	BLAKE	2850	30
7782	CLARK	2450	10
7788	SCOTT	3000	20
7839	KING	5000	10
7844	TURNER	1500	30
7876	ADAMS	1100	20
7900	JAMES	950	30

آشنایی با زبان رابطه‌ای SQL ۱۶۹

7902	FORD	3000	20
7934	MILLER	1300	10

جدول ۸-۲: جدول اطلاعات ساختار سازمانی DEPT

DEPT		
DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

حال فرض کنیم می‌خواهیم دو جدول مذکور را بر اساس صفت خاصه Deptno به یکدیگر پیوند داده و ببینیم هر یک از پرسنل در کدام واحد مشغول بکار هستند. عبارت مذکور به شرح ذیل می‌باشد:

```
Select Emp.ename, Emp.deptno, Dept.dname
From Emp Join Dept
On emp.deptno = dept.deptno
```

همانطور که ملاحظه می‌گردد دو جدول Emp و Dept بر اساس صفت خاصه Deptno که در جدول Dept بعنوان کلید اصلی و در جدول Emp بعنوان کلید خارجی می‌باشد، به یکدیگر پیوند داده شده‌اند. با این وصف خروجی حاصل از عبارت مذکور به صورت زیر خواهد بود:

جدول ۸-۳: خروجی حاصل از عبارت SQL

ENAME	DEPTNO	DNAME
SMITH	20	RESEARCH

ALLEN	30	SALES
WARD	30	SALES
JONES	20	RESEARCH
MARTIN	30	SALES
BLAKE	30	SALES
CLARK	10	ACCOUNTING
SCOTT	20	RESEARCH
KING	10	ACCOUNTING
TURNER	30	SALES
ADAMS	20	RESEARCH
JAMES	30	SALES
FORD	20	RESEARCH
MILLER	10	ACCOUNTING

توجه داشته باشید که می‌توان عبارت بالا را بدون استفاده از عملگر پیوند نیز نوشت. نحوه نوشتن عبارت مذکور بدون عملگر پیوند به شرح ذیل می‌باشد:

```
Select Emp.ename, Emp.deptno, Dept.dname
From Emp , Dept
Where emp.deptno = dept.deptno
```

همانطور که مشاهده می‌کنید می‌توان اسامی جداول مورد نظر را در جلوی کلمه FROM بکار برد. با این وصف می‌توان بدون نیاز به استفاده از عملگر پیوند اقدام به برقراری ارتباط بین دو یا چند جدول نمود. ولی بهتر است از عملگر پیوند استفاده گردد. در هر حال هنگامیکه می‌خواهیم از بیش از یک جدول استفاده نماییم، لازم است نام جداول را قبل از نام ستون‌ها (صفات خاصه) مورد نظر بکار ببریم. این وضعیت بیشتر زمانی اهمیت پیدا می‌کند که صفات خاصه همانام در جداول مورد پیوند داشته باشیم.

حال فرض کنید می‌خواهیم بجای اسامی جداول از نام‌های مستعار استفاده نماییم. مثال زیر بیانگر نحوه پیاده سازی این وضعیت می‌باشد.

```
Select E.ename, E.deptno, D.dname
From Emp E Join Dept D
```

On E.deptno = D.deptno

این وضعیت بیشتر زمانی نمود پیدا می‌کند که بخواهیم چند جدول با اسامی مختلف و یا نسبتاً طولانی را به یکدیگر پیوند بزنیم و یا اینکه در مواردی بخواهیم یک جدول را با خودش پیوند بزنیم.

عملگر پیوند طبیعی^۱

حال می‌خواهیم عبارت مذکور را بر اساس عملگر Natural join بنویسیم. لازم به ذکر است که نوع پیوند Natural Join در بعضی از پایگاه‌های داده طراحی شده است و ممکن است در بعضی از پایگاه‌های داده این نوع پیوند پشتیبانی نشود. نوع Natural Join بر این اساس عمل می‌کند که کلید ارتباطی را بصورت اتوماتیک از دیکشنری پایگاه‌داده می‌خواند. به بیانی دیگر در این نوع از پیوند نیازی به تعریف شرط پیوند نیست. مثال زیر بیانگر این وضعیت می‌باشد:

```
Select Emp.ename, Emp.deptno, Dept.dname  
From Emp Natural Join Dept
```

۶-۳ عملگرهای پیوند خارجی^۲

در بسیاری از موارد ممکن است بعضی مقادیر ستونهایی که می‌خواهیم به یکدیگر پیوند بزنیم، دارای مقدار NULL باشند. در چنین شرایطی چنانچه از عملگر پیوند بصورت عادی استفاده نماییم، تعدادی از رکوردها را نمی‌توانیم بازیابی نماییم. حال چنانچه بخواهیم بگوییم که در پیوند مذکور تمامی رکوردهای اطلاعاتی مورد نظر از یک جدول نمایش داده شود و از جدول دوم‌اندسته از رکوردهایی که در شرط بین دو جدول صدق می‌کنند نمایش داده شوند، لازم است تا از پیوند خارجی استفاده گردد. پیوند خارجی به صورت‌های زیر قابل پیاده سازی می‌باشد:

- پیوند خارجی چپ^۳
- پیوند خارجی راست^۱

1. Natural Join

2. Outer Join

3. Left Outer Join

• پیوند خارجی کامل^۲

در ادامه هر یک از پیوندهای مذکور مورد بررسی قرار خواهند گرفت.

۶-۳-۱ پیوند خارجی چپ

هر گاه بخواهیم در خروجی حاصل از ارتباط بین دو جدول، تمامی رکوردهای مورد نظر از جدول سمت چپ (جدول اول) را انتخاب کرده و از جدول سمت راست صرفاً آن دسته از رکوردهایی که شرط ارتباط بین دو جدول را دارند نمایش بدهیم، از پیوند خارجی چپ استفاده می‌کنیم. بدیهی است در چنین شرایطی برای بعضی از رکوردهای جدول سمت چپ، هیچ مقداری از جدول سمت راست نمی‌توان یافت. با این اوصاف برای آن دسته از ستونهای انتخاب شده از جدول سمت راست که مقداری ندارند، مقدار NULL در نظر گرفته خواهد شد.

برای درک بهتر موضوع همان دو جدول EMP و DEPT که کمی قبل تر معرفی شدند را مورد بررسی قرار می‌دهیم. بدین منظور با استفاده از عبارت زیر مقدار Deptno یکی از رکوردهای جدول EMP را برابر با NULL قرار می‌دهیم.

```
Update emp
set deptno = NULL
where ename = 'KING';
```

حال به عبارت زیر که در آن جدول DEPT بعنوان جدول سمت راست پیوند و جدول EMP بعنوان جدول سمت چپ پیوند می‌باشند توجه نمایید:

```
Select e.ename , e.deptno , d.dname
LEFT JOIN DEPT d From EMP e
ON e.deptno = d.deptno
```

و یا

```
Select e.ename , e.deptno , d.dname
EMP e LEFTOUTER JOIN DEPT d From
ON e.deptno = d.deptno
```

1. Right Outer Join

2. Full Outer Join

آشنایی با زبان رابطه‌ای SQL ۱۷۳

دو عبارت مذکور از لحاظ معنایی با یکدیگر برابر هستند و خروجی حاصل از عبارت بالا به شرح جدول ۸-۴ خواهد بود. همانطور که مشاهده می‌کنید با توجه به اینکه مقدار deptno از جدول EMP برابر با NULL شده است، لذا هیچ رکورد متناظری با جدول DEPT وجود نداشته و مقدار جایگزین برای ستونهای مربوطه برابر با NULL می‌باشد.

ممکن است این سؤال مطرح گردد که چرا عبارت مذکور به دو شکل مختلف نوشته شده است. در پاسخ به این سؤال باید گفت که در زبان SQL ارائه شده از سوی ANSI از کلمه Left Outer Join استفاده شده است. ولی در بعضی از پایگاه‌های داده این عبارت به صورت Left Join نوشته شده است. حتی در بعضی از پایگاه‌های داده از علامت " + " بجای این عبارت استفاده شده است.

جدول ۸-۴ خروجی حاصل از عبارت SQL پیوند خارجی چپ

ENAME	DEPTNO	DNAME
SMITH	20	RESEARCH
ALLEN	30	SALES
WARD	30	SALES
JONES	20	RESEARCH
MARTIN	30	SALES
BLAKE	30	SALES
CLARK	10	ACCOUNTING
SCOTT	20	RESEARCH
KING		
TURNER	30	SALES
ADAMS	20	RESEARCH
JAMES	30	SALES
FORD	20	RESEARCH
MILLER	10	ACCOUNTING

هر گاه بخواهیم در خروجی حاصل از ارتباط بین دو جدول، تمامی رکوردهای مورد نظر از جدول سمت راست (جدول دوم) را انتخاب کرده و از جدول سمت چپ صرفاً آن دسته از رکوردهایی که شرط ارتباط بین دو جدول را دارند نمایش بدهیم، از پیوند خارجی راست استفاده می‌کنیم. بدیهی است در چنین شرایطی برای بعضی از رکوردهای جدول سمت راست پیوند، هیچ مقداری از جدول سمت چپ نمی‌توان یافت. با این اوصاف برای آن دسته از ستونهای انتخاب شده از جدول سمت چپ پیوند که مقداری ندارند، مقدار NULL در نظر گرفته خواهد شد.

برای درک بهتر موضوع همان دو جدول EMP و DEPT که کمی قبل تر معرفی شدند را مورد بررسی قرار می‌دهیم. همانطور که می‌دانید در جدول DEPT یک رکورد با deptno = 40 وجود دارد. ولی هیچ رکوردی در جدول EMP وجود ندارد که دمقدار صفت خاصه deptno در آن برابر با ۴۰ باشد. حال به عبارت زیر که در آن جدول DEPT بعنوان جدول سمت چپ پیوند و جدول EMP بعنوان جدول سمت راست پیوند می‌باشند توجه نمایید:

```
Select e.ename , e.deptno , d.dname
RIGHT JOIN DEPT d From EMP e
ON e.deptno = d.deptno
```

و یا

```
Select e.ename , e.deptno , d.dname
EMP e RIGHT OUTER JOIN DEPT d From
ON e.deptno = d.deptno
```

خروجی حاصل از عبارت بالا به شرح ذیل خواهد بود:

جدول ۸-۵ خروجی حاصل از عبارت SQL پیوند خارجی راست

ENAME	DEPTNO	DNAME
CLARK	10	ACCOUNTING
MILLER	10	ACCOUNTING
SMITH	20	RESEARCH
SCOTT	20	RESEARCH
ADAMS	20	RESEARCH
FORD	20	RESEARCH

آشنایی با زبان رابطه‌ای SQL ۱۷۵

JONES	20	RESEARCH
ALLEN	30	SALES
JAMES	30	SALES
TURNER	30	SALES
BLAKE	30	SALES
MARTIN	30	SALES
WARD	30	SALES
		OPERATIONS

همانطور که مشاهده می‌کنید با توجه به اینکه مقدار deptno از جدول EMP برابر با NULL شده است، لذا هیچ رکورد متناظری با جدول DEPT وجود نداشته و مقدار جایگزین برای ستونهای مربوطه برابر با NULL می‌باشد.

۶-۳-۳ پیوند خارجی کامل

قبل از توضیح این نوع از پیوند لازم است به این نکته مهم توجه شود که این نوع پیوند فقط در بعضی از پایگاه‌های داده طراحی شده است و در بسیاری از پایگاه‌های داده اصلاً چنین پیوند وجود ندارد.

هر گاه بخواهیم در خروجی حاصل از ارتباط بین دو جدول، تمامی رکوردهای مورد نظر از هر دو جدول چه آنهاییکه در شرط پیوند هستند و چه آنهاییکه در شرط پیوند صادق نیستند، در خروجی حاصل بیایند، از این نوع پیوند استفاده می‌گردد. بدیهی است که هر یک از مقادیر ستونهایی که در شرط پیوند صادق نباشند، با مقدار NULL جایگزین خواهند شد. برای درک بهتر موضوع همان دو جدول EMP و DEPT که کمی قبل تر معرفی شدند را مورد بررسی قرار می‌دهیم. بدین منظور با استفاده از عبارت زیر مقدار Deptno یکی از رکوردهای جدول EMP را برابر با NULL قرار می‌دهیم.

```
Update emp  
set deptno = NULL  
where ename = 'KING';
```

حال به عبارت زیر که در آن جدول DEPT بعنوان جدول سمت راست پیوند و جدول EMP بعنوان جدول سمت چپ پیوند می‌باشند توجه نمایید:

```
Select e.ename , e.deptno , d.dname
EMP e FULL OUTER JOIN DEPT d From
ON e.deptno = d.deptno
```

خروجی حاصل از عبارت بالا به شرح ذیل خواهد بود. همانطور که مشاهده می‌کنید با توجه به اینکه مقدار deptno از جدول EMP برابر با NULL شده است، لذا دو رکورد در خروجی حاصل دارای مقادیر NULL می‌باشند.

جدول ۸-۶ خروجی حاصل از عبارت SQL پیوند خارجی کامل

ENAME	DEPTNO	DNAME
SMITH	20	RESEARCH
ALLEN	30	SALES
WARD	30	SALES
JONES	20	RESEARCH
MARTIN	30	SALES
BLAKE	30	SALES
CLARK	10	ACCOUNTING
SCOTT	20	RESEARCH
TURNER	30	SALES
ADAMS	20	RESEARCH
JAMES	30	SALES
FORD	20	RESEARCH
MILLER	10	ACCOUNTING
		OPERATIONS
KING		

۷ - سایر عملگرها در جستجوهای پیشرفته

۷-۱ عملگر گروه بندی^۱

آشنایی با زبان رابطه‌ای SQL ۱۷۷

فرض کنید می‌خواهیم رکوردهای (سطرها) یک جدول را بر اساس مقادیر یک یا چند ستون آن جدول گروه بندی نماییم بگونه‌ایکه رکوردهای موجود در هر گروه دارای یک مقدار مشترک از ستونهای مورد نظر در گروه بندی باشند. برای انجام این امر از عملگر GroupBy استفاده می‌شود. برای مثال فرض کنید می‌خواهیم رکوردهای جدول EMP را بر اساس مقادیر صفت خاصه (ستون) deptno گروه بندی نماییم. عبارت مورد نظر به شرح ذیل خواهد بود:

```
Select *  
From EMP  
Group By deptno
```

۷-۲ عملگر مرتب سازی

همانطور که مشاهده می‌گردد عملگر Group By اقدام به دسته بندی رکوردها بر اساس مقادیر یک (یا چند) ستون می‌نماید. ولی این عملگر منطقاً الزام به مرتب سازی مقادیر بصورت صعودی یا نزولی نمی‌کند و صرفاً گروه بندی را انجام می‌دهد. حال اگر لازم باشد بگونه‌ای اقدام به مرتب سازی خروجی بنماییم لازم است از عملگر ORDERBY استفاده گردد. این عملگر توانایی مرتب سازی رکوردها به صورت صعودی و نزولی دارد. برای انتخاب نوع مرتب سازی یکی از دو عملگر زیر مورد استفاده قرار می‌گیرند:

ASC : رکوردها را بصورت صعودی مرتب سازی می‌کند

DESC : رکوردها را بصورت نزولی مرتب سازی می‌کند

به عبارت زیر که رکوردهای جدول EMP را بر اساس صفت خاصه ename و بصورت صعودی مرتب می‌کند توجه نمایید:

```
Select *  
From EMP  
ORDER BY ename ASC
```

حال فرض کنید می‌خواهیم رکوردهای جدول مذکور را بر اساس دو صفت خاصه مرتب سازی نماییم. عبارت زیر بیانگر نحوه انجام این موضوع می‌باشد:

```
Select *
```

From EMP
ORDER BY deptno DESC , ename ASC

همانطور که مشاهده می‌نمایید در عبارت فوق ابتدا رکوردها بر اساس مقدار صفت خاصه deptno بصورت نزولی مرتب شده و سپس بر اساس صفت خاصه ename بصورت صعودی مرتب می‌گردند.

۷-۳ توابع تجمعی

یکی از نیازهای معمول کاربران انجام محاسباتی بر روی بعضی ستونها می‌باشد. بعنوان مثال فرض کنید می‌خواهید میانگین نمره دانش آموزان در یک کلاس برای یک درس را بدست آورید. بدیهی است که این عمل بصورت ستونی انجام می‌گیرد و با سایر دستوراتی که تا بحال گفته‌ایم متفاوت می‌باشد. توابعی که بر روی مقادیر ستونها عمل می‌کنند را توابع تجمعی می‌نامیم. لیستی از معروفترین این توابع به شرح ذیل می‌باشند:

COUNT	: تعداد مقادیر در یک ستون را ارائه می‌دهد.
SUM	: حاصل جمع مقادیر یک ستون را ارائه می‌دهد.
AVG	: میانگین مقادیر یک ستون را ارائه می‌دهد.
MAX	: بالاترین مقدار در بین مقادیر یک ستون را ارائه می‌دهد.
MIN	: پایین ترین مقدار در بین مقادیر یک ستون را ارائه می‌دهد.

نکته‌ای که باید بدان توجه کرد این است که خروجی حاصل از توابع مذکور بصورت یک مقدار می‌باشند. لازم به ذکر است که هر پایگاه‌داده‌ای علاوه بر توابع مذکور، تعدادی تابع دیگر نیز طراحی کرده است. برای مثال در بعضی از پایگاه‌های داده توابعی جهت محاسبه واریانس و یا انحراف معیار طراحی شده‌اند. برای درک بهتر موضوع، جدول EMP را در نظر بگیرید. فرض کنید می‌خواهیم تعداد پرسنلی که مقدار صفت خاصه deptno در آنها برابر با ۱۰ باشد را، بدست آوریم. عبارت مذکور به شرح ذیل خواهد بود:

```
SELECT Count (*) as Tedad
FROM EMP
WHERE deptno = 10 ;
```

آشنایی با زبان رابطه‌ای SQL ۱۷۹

خروجی حاصل از عبارت مذکور به شرح ذیل خواهد بود:

TEDAD
3

حال فرض کنید می‌خواهیم بالاترین و پایین‌ترین میزان حقوق پرسنلی که مقدار صفت خاصه deptno در آنها برابر با ۱۰ باشد را، بدست آوریم. عبارت مذکور به شرح ذیل خواهد بود:

```
SELECT Max(sal) as maxval , Min(sal) as minval  
From EMP  
Where deptno =10 ;
```

خروجی حاصل از عبارت مذکور به شرح ذیل خواهد بود:

MAXVAL	MINVAL
5000	1300

حال فرض کنید می‌خواهیم حداقل میانگین حقوق را برای پرسنل تعریف شده در جدول EMP بر حسب صفت خاصه deptno بدست آوریم. در این شرایط لازم است علاوه بر استفاده از توابع تجمعی از Group By نیز استفاده نمود. به عبارت زیر که بیانگر این درخواست می‌باشد توجه کنید:

```
Select deptno , Min(sal) as MinVal  
From emp  
Group by deptno
```

خروجی حاصل از عبارت مذکور به شرح ذیل خواهد بود:

DEPTNO	MINVAL
10	1300
20	800
30	950

۷-۴ عملگر HAVING

همانطور که می‌دانید تابحال از عملگر WHERE برای تعیین شرایط استفاده می‌کردیم. ولی عملگر Where فقط قادر بود بر روی ستونها عمل نماید، حال آنکه گاهی لازم است عملگر تعیین شرایط بر روی سطرها عمل نماید. این وضعیت بیشتر زمانی رخ می‌دهد که بخواهیم از Group By استفاده نماییم و یا اینکه بخواهیم از توابع تجمعی که اساساً بر روی سطرها عمل می‌کنند، استفاده نماییم. در چنین شرایطی بجای استفاده از کلمه Where از کلمه Having استفاده می‌کنیم. برای درک بهتر موضوع به مثال زیر توجه نمایید:

```
Select deptno
From emp
Group By deptno
Having COUNT (*) > 4
```

خروجی حاصل از عبارت بالا به شرح ذیل خواهد بود:

DEPTNO
20
30

همانطور که مشاهده می‌نمایید از آنجائیکه تعداد رکوردهایی که در آنها مقدار deptno برابر ۱۰ باشد، کمتر از ۴ تا هستند، لذا رکوردی که بیانگر مقدار ۱۰ در ستون deptno باشد در خروجی مشاهده نمی‌گردد. یکی از نکات مهم در زبان SQL این است که عملگرهای Having و Where می‌توانند بصورت ترکیبی در یک عبارت SQL مورد استفاده قرار گیرند.

۷-۵ عملگر BETWEEN

یکی از ویژگی‌های مورد نیاز کاربران، امکان انتخاب رکوردها بر اساس مقادیری است که در یک بازه قرار داشته باشند. برای مثال فرض کنید لیست پرسنلی را بخواهیم که حقوق آنها بین ۱۰۰۰ و ۲۰۰۰ باشد. عبارت زیر نحوه استفاده از دستور Between را نشان می‌دهد:

```
Select ename, Sal
From EMP
Where sal Between 1000 and 2000 ;
```

خروجی حاصل از عبارت مذکور به شرح زیر خواهد بود:

ENAME	SAL
ALLEN	1600
WARD	1250
MARTIN	1250
TURNER	1500
ADAMS	1100
MILLER	1300

۷-۶ عملگر LIKE

در بعضی موارد کاربران نیاز دارند تا عبارتی را در بین مقادیر یک صفت خاصه جستجو نمایند. این وضعیت بیشتر زمانی رخ می‌دهد که کاربران در حال کار با مقادیر رشته‌ای باشند. در این شرایط از عملگر Like استفاده می‌نماییم. برای مثال فرض کنید می‌خواهیم در جدول EMP به دنبال افرادی بگردیم که در صفت خاصه ename آنها کلمه 'AM' باشد. عبارت زیر نحوه نوشتن چنین درخواستی را نشان می‌دهد:

```
Select ename ,deptno, sal
From emp
'; %AM% Where ename LIKE '
```

همانطور که مشاهده می‌کنید علامت " % " نشان می‌دهد که کلمه " AM " می‌تواند در بین مقادیر موجود باشد و لازم نیست حتما در ابتدا یا انتهای آن مقادیر باشد. خروجی حاصل از عبارت بالا به شرح زیر خواهد بود.

ENAME	DEPTNO	SAL
ADAMS	20	1100
JAMES	30	950

۸ - استفاده از پرس و جوهای تودرتو^۱

در بسیاری از موارد مقادیر مورد نظر در عبارت where خود بصورت مشخص نبوده و بر اساس یک عبارت پرس و جوس دیگر بدست می آیند. در چنین شرایطی لازم است تا بجای استفاده از عبارات جداگانه، از عبارات تودرتو استفاده کرد تا خروجی هر یک از عبارات درونی بعنوان مقادیر مورد استفاده در عبارت بیرونی تر در نظر گرفته شود. در ادامه انواعی از عبارات تودرتو را نشان خواهیم داد.

۸-۱ استفاده از زیرپرسش‌ها^۲ (پرسش‌های فرعی)

یکی از نیازهای کاربران امکان طرح پرسشی درون پرسش دیگر می‌باشد. هر گاه بخواهیم پرسشی را درون پرسشی دیگر مطرح نماییم، به آن زیر پرسش می‌گویند. بدیهی است که هر پرسشی چه بصورت پرسش بیرونی باشد و یا بصورت پرسش درونی، لازم است که با عبارت Select آغاز گردد. زیر پرسش‌ها معمولاً بعنوان قسمتی از عبارت Where بکار می‌روند.

عباراتی که بصورت زیرپرسش باشند معمولاً می‌توانند بصورت تودرتو^۳ نوشته شوند. در چنین شرایطی برای اجرای هر عبارت بیرونی لازم است ابتدا عبارت جستجوی درونی محاسبه شده و سپس عبارت بیرونی مورد بررسی قرار گیرد. به بیانی دیگر مسیر حل در عبارات تودرتو از درونی ترین عبارت به سمت بیرونی ترین عبارت خواهد بود. عبارت زیر بعنوان نمونه‌ای از این نوع پرسش‌ها می‌باشد:

```
Select ename,deptno,sal
From EMP
Where Sal > (Select AVG (sal) as Xval
From EMP)
```

خروجی حاصل از عبارت بالا به شرح ذیل می‌باشد:

1. nested Queries
2. Sub query
3. Nested

آشنایی با زبان رابطه‌ای SQL ۱۸۳

ENAME	DEPTNO	SAL
JONES	20	2975
BLAKE	30	2850
CLARK	10	2450
SCOTT	20	3000
KING	10	5000
FORD	20	3000

۸-۲ عملگر IN

معمولاً در زیر پرسش‌ها شرایط بگونه‌ایست که کاربر می‌خواهد مقداری را از بین یک مجموعه مقادیر بدست آورد. برای این منظور از عملگر IN استفاده می‌گردد. این عملگر تقریباً شبیه به عملگر Between می‌باشد با این تفاوت که مقداری را درون یک مجموعه مقادیر مورد جستجو قرار می‌دهد. مثال زیر نمونه‌ایست از عباراتی که بدین صورت نوشته می‌شوند:

```
Select ename,sal
  From emp
 Where empno IN
      (Select empno
        From emp
        Where sal >2000)
```

خروجی حاصل از عبارت بالا به صورت جدول زیر می‌باشد:

ENAME	SAL
JONES	2975
BLAKE	2850
CLARK	2450
SCOTT	3000
KING	5000
FORD	3000

ممکن است این سؤال به ذهن برسد که عبارت مذکور را می‌توان بصورتی ساده‌تر نیز نوشت که در آن نیازی به استفاده از عملگر IN نباشد. در اینجا لازم است گفته شود که عبارت مذکور فقط جهت نشان دادن نحوه استفاده از عملگر IN می‌باشد. یکی از نکات کلیدی در استفاده از عملگر مذکور این است که باید زیرپرسش جلوی عملگر IN حتما دارای یک ستون ممنوع با ستون مورد پرسش باشد و نمی‌توان در زیرپرسش از هر ستونی بعنوان خروجی استفاده کرد.

۸-۳ عملگر EXISTS

یکی دیگر از عبارات مورد استفاده در پرس و جویهای تودرتو، استفاده از کلمه کلیدی EXISTS می‌باشد. این دستور نیز تا حدودی شبیه به کلمه کلیدی IN می‌باشد. معمولاً از عملگر EXISTS به ندرت استفاده می‌شود. در هنگام استفاده از این عملگر می‌توان در قسمت Where عبارت درونی از مقادیر عبارت بیرونی استفاده کرد. عملگر EXISTS تا حدودی با سایر عملگرهای مربوط به عبارات پرس و جوی تودرتو متفاوت می‌باشد. زیرا ترکیبی از مقادیر مربوط به عبارات بیرونی و درونی درون شرط عبارات درونی قرار می‌گیرد. برای مثال به عبارت زیر توجه کنید. همانطور که مشاهده می‌کنید مقدار صفت خاصه STID از رابطه بیرونی در قسمت WHERE عبارت درونی مورد مقایسه قرار گرفته است.

```
SELECT STNAME
FROM STT
WHERE EXISTS (SELECT *
FROM STCOT
WHERE STCOT.STID = STT.STID
AND COID = '1234')
```

۹- دستورات کنترل مجوز دسترسی

دستورات کنترل کننده مجوز دسترسی در پایگاه‌های داده به دو صورت کلی زیر تقسیم می‌شود:

- دسترسی به داده‌ها
- دسترسی به امکانات مدیریتی

با توجه به اینکه بحث دسترسی به امکانات مدیریتی در این قسمت از کتاب مطرح نیست، لذا صرفاً کنترل دسترسی به داده‌ها مورد بررسی قرار می‌گیرد. دستورات کنترل‌کننده دسترسی به داده‌ها به شرح زیر می‌باشد:

- دستور اعطا^۱ اختیارات
- دستور لغو^۲ اختیارات

۹-۱ دستور اعطا اختیارات

برای اعطا امتیاز از دستور GRANT استفاده می‌شود. شکل کلی این دستور به شرح زیر می‌باشد:

```
GRANT Privileges | ALL Privileges  
ON Object  
TO Users [WITH GRANT OPTION];
```

در قسمت Privileges از یکی یا ترکیبی از کلمات زیر استفاده می‌گردد:

```
INSERT  
DELETE  
UPDATE  
SELECT
```

در قسمت Object یکی از اشیاء موجود در Schema یک کاربر انتخاب شده و در قسمت Users نام یکی از کاربران پایگاه داده انتخاب می‌گردد.

عبارت WITH GRANT OPTION در مواردی مورد استفاده قرار می‌گیرد که بخواهیم جدا از اعطا اختیارات مذکور به یک کاربر، به او اجازه دادن همان اختیارات به کاربر دیگری را نیز بدهیم. برای درک بهتر موضوع به عبارات زیر توجه کنید:

```
Grant delete,update,insert  
ON EMP  
TO ALI;
```

```
Grant select  
On dept  
TO PERSON[With Grant Option];
```

1. Grant
2. Revoke

۹-۲ دستور لغو اختیارات

برای لغو امتیاز از دستور REVOKE استفاده می‌شود. شکل کلی این دستور به شرح زیر می‌باشد:

```
REVOKE Privileges | ALL Privileges
ON Object
FROM Users
```

در قسمت Privileges از یکی یا ترکیبی از کلمات زیر استفاده می‌گردد:

```
INSERT
DELETE
UPDATE
SELECT
```

در قسمت Object یکی از اشیاء موجود در Schema یک کاربر انتخاب شده و در قسمت Users نام یکی از کاربران پایگاه داده انتخاب می‌گردد. برای درک بهتر موضوع به عبارات زیر توجه کنید:

```
Revoke select
On DEPT
FROM ALI;
```

۱۰- امکانات و ویژگی‌های SQL2 و SQL3

با توجه به توضیحات ارائه شده در خصوص زبان SQL، در ادامه درخصوص امکانات و ویژگی‌های دو زبان استاندارد SQL2 و SQL3 به اختصار مطالبی بیان شده است.

۱۰-۱ امکانات SQL2

این نسخه از SQL در سال ۱۹۹۲ میلادی به عنوان SQL استاندارد معرفی گردید. این نسخه در واقع نسخه تصحیح شده و کاملتر نسخه SQL1 بود. در این دستور امکاناتی به نسخه SQL1 افزوده شده و یا بعضی از دستورات SQL1 کامل تر شدند. در ادامه تغییرات اعمال شده به اختصار آمده است:

- افزودن دستورات مربوط به ایجاد، اصلاح و حذف Domain
- تکمیل دستورات مربوط به ایجاد و حذف جدول
- افزودن دستور اصلاح جدول
- افزودن دستور تعریف جدول موقت
- امکان نوشتن Select از یک Select دیگر
- امکان نوشتن Select بعد از WHERE
- تکمیل دستور JOIN
- اصلاح و تکمیل عملگرهای خاص مانند فرااجتماع، اسکالر و...
- افزودن امکانات جامعیتی و ایمنی و...

۱۰-۲ امکانات SQL3

این نسخه از SQL برای محیط‌های ORDBMS^۱ تولید شده است. در واقع پژوهشگران به این نتیجه رسیدند که تکنولوژی‌های RDBMS و OODBMS هیچیک به تنهایی کامل نبوده و صرفاً ترکیب این دو تکنولوژی پاسخگوی نیازهای آتی خواهد بود. لذا تصمیم گرفته شد نسخه ای از SQL ارائه گردد تا پاسخگوی نیازهای آتی باشد. در این نسخه امکانات زیر افزوده شده‌اند.

- ایجاد امکان تعریف انواع داده‌های جدید. مدل داده ای قوی تر برای نمایش داده‌هایی که امکان نمایش آنها با ساختار جدولی وجود ندارد.
- رویه ای شدن زبان. به نحوی که SQL را از نظر برنامه سازی و محاسباتی کامل و انرا به یک زبان برنامه سازی تبدیل می‌کند.
- ایجاد امکان راه‌انداز. راه‌انداز^۲ قاعده (محدودیت) و یا قواعدی است که قبل یا بعد از بروز یک رویداد در پایگاه‌داده‌ها و یا بجای یک رویداد باید اعمال گردد. این قاعده یا محدودیت در سطح برنامه سازی، بصورت رویه ای از پیش تعریف شده است که بصورت شرطی یا غیر شرطی، قبل یا بعد از انجام یک عمل در پایگاه‌داده‌ها بصورت اتوماتیک اجرا می‌شود

1. Object Relational Database Management System

2. Trigger

تمرینات

تمرین ۱: رابطه‌های زیر را در نظر بگیرید

Student (SID, Name, Major, Gradelevel, Age)
 Class (Name, Time, Room)
 Enrollment (Student Number, Class Name, Polition Number)
 Junior (Snum, Name, Major)
 Honor-Student (Number, Name, Interset)
 Faculty (FID, Name, Department)

STUDENT

SID	Name	Major	Gradelevel	Age
100	J	Hestory	G	21
150	P	Acount	S	19
200	B	Math	G	50
250	G	Hestory	S	50
300	B	Acount	S	41
350	RU	Math	J	20
400	R	Acount	F	18
450	J	Hestory	S	24

Enrollment

Student Num	Class Name	Polition Num
100	BD445	1
150	BA200	1
200	BD445	2
250	CS250	1
300	CS150	1

آشنایی با زبان رابطه‌ای SQL ۱۸۹

350	BA200	2
400	BF410	1
450	CS250	2

Class

Name	Time	Room
BA200	F9	110
BD445	F3	213
BF410	F8	213
CS150	F3	314
CS250	F12	210

خروجی حاصل از عبارات زیر را بدت آورید؟

```
SELECT Name, Major, Age
FROM Student
WHERE Major = 'Account'
ORDER by Name
```

```
SELECT Name, Major, Age
FROM Student
WHERE Gradelevel IN ['F', 'S']
```

```
SELECT Count (*)
FROM Student
```

```
SELECT Count (DISTINCT Major)
FROM Student
```

```
SELECT Major, Count(*)
FROM Student
Order By Major
```

```
SELECT Major, Count(*)
FROM Student
GROUP By Major
HAVING Count (*) > 2
```

```
SELECT Major, AVG(Age)
FROM Student
WHERE Gradelevel = 'S'
GROUP By Major
HAVING COUNT (*) > 1
```

```

SELECT Name
FROM Student
WHERE SID IN
      (SELECT Student Number
       FROM Enrollment
       WHERE Class Name = '445')

```

تمرین ۲:

بانک اطلاعاتی عرضه کنندگان قطعات و پروژه‌ها به شرح زیر تنظیم شده است ، بر اساس آنها به سوالات مطرح شده پاسخ دهید ؟

```

S { S# , SNAME , STATUS , CITY }
PRIMARY KEY { S# }
P { P# , PNAME , COLOR , WEIGHT , CITY }
PRIMARY KEY { P# }
J { J# , JNAME , CITY }
PRIMARY KEY { J# }
SPJ { S# , P# , J# , QTY }
PRIMARY KEY { S# , P# , J# }
FOREIGN KEY { S# } REFERENCES S
FOREIGN KEY { P# } REFERENCES P
FOREIGN KEY { J# } REFERENCES J

```

شماره قطعاتی را مشخص کنید که بیش از یک عرضه کننده آنها را عرضه می‌کنند؟
اسامی عرضه کنندگانی که قطعه P2 را عرضه نمی‌کنند ؟

فصل ۹

نرمال سازی

هدف کلی

در این فصل ابتدا تعریفی از رابطه نرمال را ارائه کرده و معایب و مزایای رابطه‌های نرمال و غیرنرمال را بیان خواهیم کرد. سپس با مفهوم تئوری وابستگی و در ادامه مفاهیم تئوری وابستگی تابعی، وابستگی تابعی و حالت‌های وابستگی آشنا خواهیم شد. در ادامه مفهوم نرمال سازی و انواع صورت‌های نرمال رابطه‌ها را به تفصیل مورد بحث و بررسی قرار خواهیم داد.

هدف رفتاری

در این فصل عناوین زیر مورد بحث و بررسی قرار می‌گیرند:

- تعریف رابطه نرمال
- مزایا و معایب رابطه‌های نرمال
- مزایا و معایب رابطه‌های غیر نرمال
- مفاهیم تئوری وابستگی و انواع وابستگی‌ها
- حالت‌های وابستگی
- نمودار وابستگی‌های تابعی
- صورت‌های نرمال
- فرایند نرمال سازی

۱- تعریف رابطه نرمال

یکی از مهم ترین مباحث در پایگاه‌های داده رابطه‌ای، بحث نرمال سازی رابطه‌ها است. حال این سؤال به ذهن می‌رسد که اصولاً رابطه نرمال چیست. در ادامه تعاریفی از رابطه نرمال را ارائه می‌کنیم.

- **تعریف اول:** رابطه نرمال رابطه‌ای است که مقادیر تمام صفاتش تجزیه نشدنی باشند.
- **تعریف دوم:** رابطه نرمال رابطه‌ای است که مقادیر هیچیک از میدانهایش، یک مقدار رابطه‌ای با کاردینالیتی بزرگتر از یک نباشد.

در توصیف تعاریف بالا می‌توان گفت یک مقدار ساده تجزیه نشدنی را می‌توان یک مقدار رابطه‌ای از درجه یک و کاردینالیتی یک دانست. به بیانی دیگر در بدنه رابطه نرمال، صفت ساده یا مرکب چند مقداری وجود ندارد و اصطلاحاً می‌گوییم رابطه مسطح است. با توجه به تعاریف بالا **رابطه غیر نرمال** را می‌توان بصورت زیر توصیف کرد:

رابطه‌ای که در آن مقادیر حداقل یک صفت، خود مقادیر رابطه‌ای باشند را رابطه غیر نرمال گوییم. به بیانی دیگر رابطه‌ای که حداقل یک صفت ساده یا مرکب چند مقداری داشته باشد، رابطه غیر نرمال است.

۱-۱ دلایل لزوم نرمال بودن رابطه

همانطور که می‌دانید نرمال بودن رابطه دلیل ریاضی ندارد و می‌توان گفت صرفاً جنبه سادگی آن مطرح است. سادگی عنصر اساسی ساختار رابطه‌ای خود سادگی‌های زیر را در بر دارد:

- سادگی در نمایش ظاهری رابطه
- سادگی دستورات زبان
- سادگی اجرای عملیات در پایگاه داده

۲-۱ معایب رابطه نرمال

رابطه نرمال معایبی دارد که به شرح ذیل آمده است:

- بروز پدیده افزونگی که می تواند فیزیکی هم باشد.
- طولانی تر شدن کلید رابطه
- عدم امکان نمایش داده های پیچیده
- دشواری در نمایش طبیعی مفهوم سلسله مراتب
- دشواری در نمایش مفهوم وراثت

۱-۳ مزایا و معایب رابطه غیر نرمال

رابطه غیر نرمال نیز دارای مزایایی به شرح ذیل می باشد

- کاهش میزان افزونگی
- کوتاه شدن کلید
- امکان نمایش داده های پیچیده
- دشواری کمتر در نمایش مفهوم سلسله مراتب
- دشواری کمتر در نمایش مفهوم وراثت
- افزایش سرعت عملیات (در بعضی موارد) در بازیابی داده ها

بدیهی است که در کنار این مزایا، رابطه غیر نرمال خود نیز دارای معایبی می باشد

که این معایب در ذیل آمده است:

- پیچیدگی در نمایش داده ها
- پیچیدگی در نگارش دستورات
- پیچیدگی در اجرای دستورات

۲- مفاهیم تئوری وابستگی^۱

در بحث نرمال سازی رابطه ها، به مفاهیم تئوری وابستگی نیاز داریم. در این قسمت

با برخی مفاهیم وابستگی آشنا می شویم.

1. Dependency

۱-۲ انواع وابستگی‌ها

وابستگی دارای حالات مختلفی است که در ذیل آمده است:

- وابستگی تابعی^۱
- وابستگی پیوندی^۲
- وابستگی چند مقداری^۳

۱-۱-۲ وابستگی تابعی

مهمترین نوع وابستگی وابستگی تابعی است. در حالت کلی وابستگی تابعی را می‌توان اینگونه توصیف کرد:

اگر A و B دو صفت در شمای R باشند، آنگاه وابستگی تابعی $A \rightarrow B$ برقرار است به شرطی که برای تمامی رابطه‌ها در R به ازای هر مقدار A فقط یک مقدار B وجود داشته باشد.

وابستگی تابعی را می‌توان بدین صورت نیز توصیف کرد:

فرض می‌کنیم که R یک متغیر رابطه‌ای و A و B دو زیر مجموعه دلخواه از مجموعه عنوان R یعنی H باشند. می‌گوییم B با A وابستگی تابعی دارد و چنین نمایش می‌دهیم: $A \rightarrow B$ اگر و فقط اگر در هر مقدار ممکن از متغیر رابطه‌ای R ، به هر مقدار A فقط یک مقدار B متناظر باشد. به بیان دیگر اگر t_1 و t_2 دو تاپل متمایز از R باشند، در اینصورت

$$\text{If } t_1.A = t_2.A \text{ then } t_1.B = t_2.B$$

۲-۱-۲ وابستگی تابعی کامل

$A \rightarrow B$ را وابستگی تابعی کامل^۴ گویند اگر B به هیچ زیرمجموعه از A وابسته نباشد. وابستگی تابعی از ویژگی‌های معنایی صفات است و به بیان دیگر هر

1. functional Dependency

2. Join Dependency

3. Multi Value Dependency

4. Full Functional Dependency

وابستگی تابعی بین دو صفت، نمایشگر یک قاعده سمتیک در جهان واقعی است. مثلا وقتی می‌گوییم:

STID→STMJR

معنایش این است که یک دانشجو فقط در یک رشته تحصیل می‌کند.

۲-۲ حالت‌های وابستگی

وابستگی دارای حالات مختلفی می‌باشد که در ذیل نام برده شده‌اند:

- وابستگی به کلید
- وابستگی به بخشی از کلید
- وابستگی غیرکلید به غیرکلید
- وابستگی معکوس
- کلید یا بخشی از آن به صفت یا صفت‌های دیگر وابسته باشد

۳-۲ تعریف $F +$

اگر F یک مجموعه از وابستگی‌های تابعی باشد، آنگاه مجموعه تمام وابستگی‌های تابعی قابل استخراج از آنرا مجموعه پوششی F می‌نامند و با $F +$ نشان می‌دهند.

روش یافتن $F +$:

برای یافتن $F +$ عملیات زیر را به ترتیب انجام می‌دهیم:

- بازتاب

IF $B \subseteq A$ THEN $A \rightarrow B$

- افزایشی

IF $A \rightarrow B, C$ THEN $AC \rightarrow BC$

((توجه: C صفت می‌باشد))

- انتقال

IF $A \rightarrow B, B \rightarrow C$ THEN $A \rightarrow C$

چون اعمال این قواعد مشکل می‌باشد، آنها را به صورت ساده تر بیان می‌کنیم.

• اجتماع

IF $A \rightarrow B$ AND $A \rightarrow C$ THEN $A \rightarrow BC$

• تجزیه

IF $A \rightarrow BC$ THEN $A \rightarrow B$ AND $A \rightarrow C$

• ترکیب

IF $A \rightarrow B$, $C \rightarrow D$ THEN $AC \rightarrow BD$

با اعمال این قواعد سعی می‌کنیم وابستگی‌های تکراری را حذف نموده و به مجموعه وابستگی کمینه یا بهینه برسیم.

مثال:

$F^+ = \{ A \rightarrow B, A \rightarrow C, B \rightarrow C, AB \rightarrow C, AC \rightarrow D \}$

$\min(F^+) = \{ A \rightarrow B, B \rightarrow C, A \rightarrow D \}$

مثال:

$R = (U, V, W, X, Y, Z)$

$F = \{ U \rightarrow XY, X \rightarrow Y, XY \rightarrow ZV \}$

$F^+ = \{ U \rightarrow XY, X \rightarrow Y, XY \rightarrow ZV, U \rightarrow ZV \}$

$F^+ = \{ U \rightarrow X, U \rightarrow Y, X \rightarrow Y, XY \rightarrow ZV, U \rightarrow ZV \}$

$F^+ = \{ U \rightarrow X, U \rightarrow Y, X \rightarrow Y, X \rightarrow ZV, U \rightarrow ZV \}$

$F^+ = \{ U \rightarrow X, U \rightarrow Y, X \rightarrow Y, U \rightarrow Z, U \rightarrow V, X \rightarrow Z, X \rightarrow V \}$

تمرین:

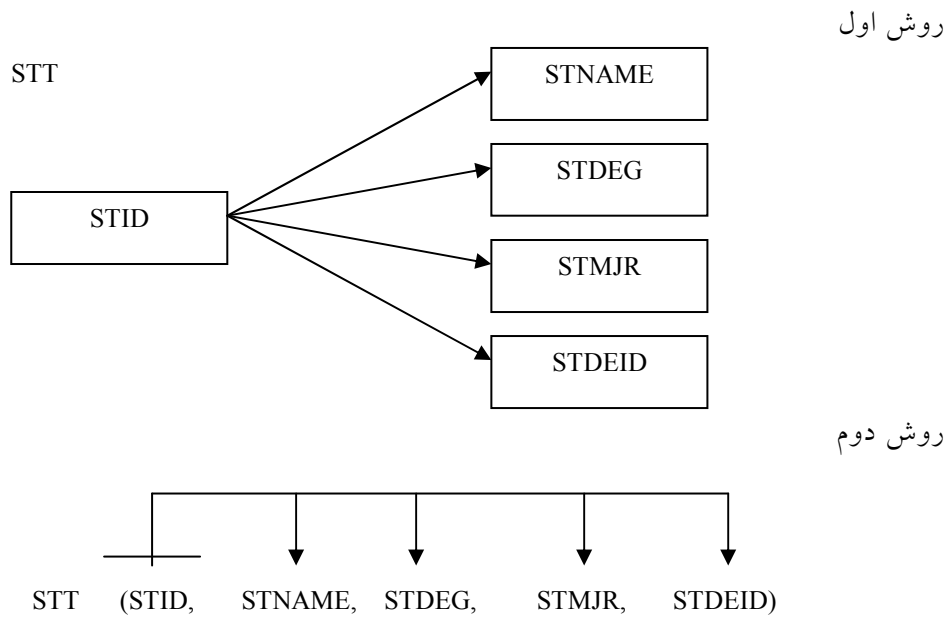
$R = (U, V, W, X, Y, Z, O, P, Q)$

$F = \{ U \rightarrow VXQ, UVP \rightarrow O, OQ \rightarrow YZ, UP \rightarrow XY \}$

۲-۴ نمودار وابستگی‌های تابعی^۱

جهت درک بهتر از وابستگی‌ها، نمودار وابستگی‌های تابعی را رسم می‌کنیم. نحوه نمایش نمودار وابستگی تابعی برای رابطه STT به یکی از دو شکل زیر قابل ترسیم است.

1. Functional Dependency Diagram



۳- صورت های نرمال

کاد واضع مدل رابطه ای ، در آغاز سه سطح نرمال را تعریف کرد. این سه سطح عبارت هستند از:

- صورت نخست نرمال^۱
- صورت دوم نرمال^۲
- صورت سوم نرمال^۳

پژوهشگران و متخصصین بعداً سه نوع دیگر از صور نرمال را به شرح زیر تعریف کردند:

- صورت نرمال بویس - کاد^۴
- صورت چهارم نرمال^۵

-
1. First Normal Form
 2. Second Normal Form
 3. Third Normal Form
 4. Boyce-Codd Normal Form
 5. Forth Normal Form

- صورت پنجم نرمال^۱

امروزه دو نوع دیگر از صورت‌های نرمال تعریف شده‌اند که البته در شرایط بسیار خاص رخ می‌دهند. این دو صورت نرمال عبارتند از:

- صورت نرمال میدان-کلیدی
- صورت نرمال تحدید-اجتماع

در اینجا لازم است به نکات زیر توجه شود:

- چنانچه رابطه‌ای در هر یک از حالات نرمال، بصورت غیرنرمال باشد، می‌گوییم رابطه در آن سطح دارای آنومالی است.
- در واقع هشت صورت نرمال مذکور هر یک از سطح قبلی خود نرمال تر هستند (بغیر از BCNF که شرایط خاصی دارد).
- همچنین لازم به ذکر است که صورت‌های BCNF و 4NF و 5NF دارای شرایط خاصی بوده و تقریباً چنانچه رابطه در یکی از این سه سطوح نرمال باشد، در دو مورد دیگر نیز نرمال خواهد بود (این موضوع همیشگی نیست).
- حالات بالاتر از 3NF شرایط خاصی هستند که در خیلی از مواقع اصلاً رخ نمی‌دهند. لذا چنانچه رابطه‌ای را تا سطح 3NF نرمال کنیم، بطور معمول می‌گوییم نرمال سازی را انجام داده‌ایم. با توجه به تعاریف بالا می‌توان سطوح نرمال سازی را بصورت ذیل نشان داد:

DKNF C 5NF C 4NF C BCNF C 3NF C 2NF C 1NF

۴- فرایند نرمال سازی^۲

با توجه به داده‌های عملیاتی و ارتباط بین موجودیتها، لازم است تا قبل از پیاده سازی جداول طراحی شده، آنها را نرمال سازی نماییم. مراحل نرمال سازی رابطه‌ها به شرح ذیل می‌باشند:

1. Fifth Normal Form

2. Normalization

نرمال سازی ۲۰۱

- مراحل نرمال سازی
- مشخص کردن جداول
- شناسایی داده‌ها و ارتباطشان
- رسم نمودار وابستگی
- طراحی جداول اولیه
- در نظر گرفتن وابستگی‌ها
- اعمال فرمهای نرمال

در ادامه رابطه‌های نرمال را مورد بحث و بررسی قرار خواهیم داد:

۴-۱ رابطه 1NF

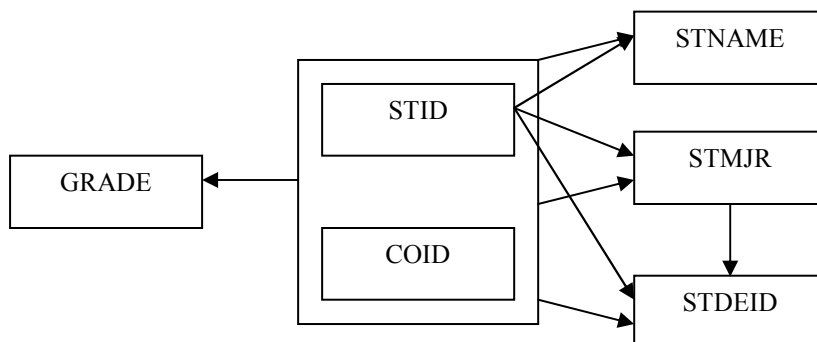
رابطه‌ای 1NF است که:

- همه کلیدهای آن تعریف شده باشد
- همه صفت‌های آن به کلید اصلی وابسته باشند
- صفت‌های ترکیبی نداشته باشیم

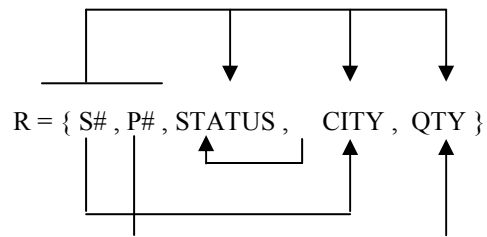
به بیانی دیگر رابطه‌ای 1NF است اگر و فقط اگر هر صفت آن در هر تاپل، تک مقداری باشد (صفت چند مقداری نداشته باشد).

مثال ۱: رابطه STCOR را که در آن STID و COID تواما کلید اصلی هستند را در نظر بگیرید:

STCOR (STID, COID, STNAME, GRADE, STMJR, STDEID)



مثال ۲:



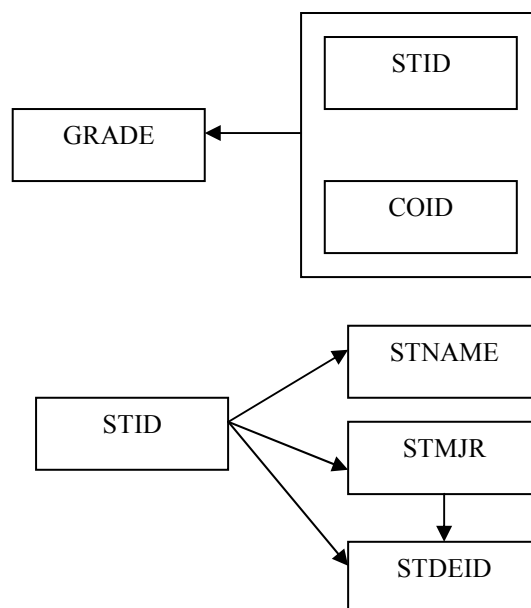
۲-۴ رابطه 2NF

رابطه‌ای 2NF است که

- 1NF باشد
- صفت‌های آن به زیر مجموعه‌های کلید اصلی وابسته نباشد

به بیانی دیگر رابطه‌ای 2NF است که اولاً 1NF باشد و ثانياً تمام صفات غیر کلید با کلید اصلی وابستگی تابعی کامل داشته باشند.

مثال ۱:



نرمال سازی ۲۰۳

مثال ۲:

$R = (Std\#, cr\#, st.name, g/u, place, mintopas, grade, time)$



1NF می باشد برای 2NF بودن:

$R1 = (Std\#, cr\#, grade)$

$R2 = (Std\#, st.name, g/u)$

$R3 = (cr\#, place, mintopas, time)$

بعد از اینکه جداول شکسته شد ، باید 1NF و 2NF بودن هرکدام از جداول بدست آمده را بررسی کنید (هر سه جدول بدست آمده فوق هم 1NF و هم 2NF می باشد).

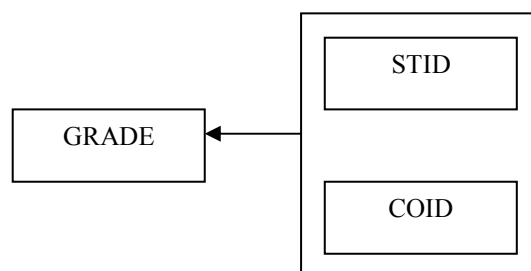
۳-۴ رابطه 3NF

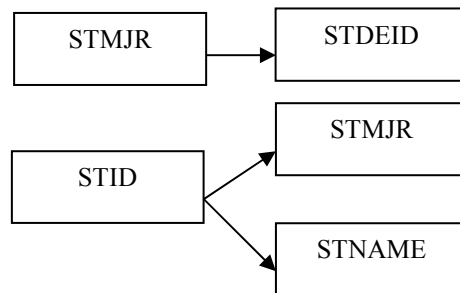
جدولی 3NF است که

- در 2NF باشد
- وابستگی انتقالی نداشته باشد (وابستگی غیر کلید به غیر کلید)

به بیانی دیگر رابطه ای 3NF است اگر و فقط اگر 2NF باشد و هر صفت غیرکلید با کلید اصلی، وابستگی تابعی بی واسطه داشته باشد.

مثال ۱:





۴-۴ رابطه BCNF

رابطه BCNF حالت خاصی از رابطه نرمال است و دارای تعاریف متعددی می‌باشد که در ذیل آمده است:

تعریف اول: جدولی BCNF است که

- در 3NF باشد
 - وابستگی معکوس نداشته باشد (وابستگی کلید به غیر کلید)
- تعریف دوم: رابطه‌ای BCNF است اگر و فقط اگر در آن هر دترمینان، کلید کاندید باشد.

تعریف سوم: رابطه $R(H)$ ، با مجموعه وابستگی‌های تابعی F ، در BCNF است اگر برای هر وابستگی تابعی F^+ به صورت $A \rightarrow B$ که در آن $A \subset R(H)$ و $B \not\subset R(H)$ است، حداقل یکی از دو حالت زیر برقرار باشد:

- $A \rightarrow B$ یک وابستگی غیر مهم باشد یعنی $B \subset A$
 - A سوپر کلید رابطه R باشد
- تعریف چهارم: رابطه‌ای BCNF است اگر و فقط اگر سمت چپ هر وابستگی تابعی مهم و کاهش ناپذیر، کلید کاندید رابطه باشد.

۴-۵ رابطه 4NF

رابطه‌ای 4NF است اگر

- BCNF باشد
- در آن وابستگی چند مقداری مهم وجود نداشته باشد.

مشکل آنومالی در سطح 4NF بدین صورت است که گاهی دو رابطه‌ای که از یکدیگر مجزا هستند را به اشتباه و صرفاً بواسطه اینکه فرض کرده‌ایم می‌توانیم تحت یک رابطه قرار دهیم، با یکدیگر ترکیب می‌کنیم و تحت عنوان یک رابطه در پایگاه داده قرار می‌دهیم. پس از ورود اطلاعات در رابطه جدید، این رابطه را به دو رابطه قبلی تجزیه می‌کنیم و طبیعتاً مقادیر رابطه کلی در د رابطه تقسیم می‌شوند. حال چنانچه بخواهیم دو رابطه را با هم ترکیب کنیم، با حذف بعضی از تاپل‌ها مواجه می‌شویم. البته این شرایط بسیار خاص بوده و امکان رخداد آن نیز بسیار کم است. برای مثال رابطه زیر را که اشتباهاً نرمال شده است را در نظر بگیرید:

PSR(PRID,STID,RNUM)

PRID	STID	RNUM
Pr22	S2	R1
Pr22	S1	R1
Pr22	S3	R1
Pr22	S2	R2
Pr22	S1	R2
Pr22	S3	R2
Pr33	S2	R1
Pr33	S6	R1

درج تاپلی مانند (Pr22,S8) عملاً به مفهوم درج دو تاپل در جدول PSR بصورت زیر خواهد بود:

(Pr22,S8,R1)
(Pr22,S8,R2)

حال چنانچه رابطه PSR را عملاً از ابتدا بصورت دو رابطه PS و PR مانند زیر در نظر بگیریم دیگر دچار چنین مشکلاتی نخواهیم شد.

PS(PRID,STID)
PR(PRID,RNUM)

در فرایندهای نرمال سازی تا سطح 4NF، الگوریتم عمومی در هر مرحله عبارت بود از تجزیه رابطه به دو رابطه نرمالتر به گونه‌ایکه با پیوند دو رابطه، همان رابطه اصلی بدست می‌آید. اما ممکن است در وضع خاصی رابطه را به دو رابطه تجزیه کنیم ولی با پیوند دو رابطه، رابطه اولیه بدست نیاید. به مثال زیر توجه کنید:

PRCODE (PRID, COID, DEID)

PRID	COID	DEID
Pr11	C1	D4
Pr11	C3	D1
Pr22	C1	D1
Pr11	C1	D1

حال این رابطه را به دو رابطه زیر تجزیه می‌کنیم. پس رابطه‌های زیر از تجزیه رابطه PRCODE بدست می‌آیند:

PRCO (PRID, COID)
CODE (COID, DEID)

PRCO

PRID	COID
Pr11	C1
Pr11	C3
Pr22	C1

CODE

COID	DEID
C1	D4
C3	D1
C1	D1

حال چنانچه دو رابطه را با یکدیگر ترکیب کنیم در خروجی حاصل یک رکورد اضافی مشاهده می‌گردد.

Proc JOIN Code = XPRCODE

→ رکورد اضافه

PRID	COID	DEID
Pr11	C1	D4
Pr11	C1	D1
Pr11	C3	D1
Pr22	C1	D4
Pr22	C1	D1

به این شرایط آنومالی در سطح 5NF می‌گویند. رفع این مشکل صرفاً تا قبل از ورود داده‌ها ممکن خواهد بود.
با این وصف، تعریف رابطه 5NF به شرح زیر خواهد بود:

نرمال سازی ۲۰۷

رابطه‌ای 5NF است اگر تمام وابستگی‌های پیوندی آن ناشی از کلیدهای کاندید آن باشد. با این وصف اگر بتوانیم یک وابستگی پیوندی در رابطه R پیدا کنیم که در پرتوهایش کلید کاندید R وجود نداشته باشد، رابطه 5NF نیست.

توجه: آنومالی در سطوح 4NF و 5NF صرفاً در شرایط خاص و برای محیط‌های خاص رخ می‌دهد و در حالت عادی این نوع آنومالی‌ها در طراحی ساختار اطلاعاتی سیستم‌ها رخ نمی‌دهد. لذا بطور معمول اگر بتوانیم ساختار اطلاعاتی سیستم مورد نظر را حداکثر تا سطح 3NF و یا BCNF نرمال کنیم، اصطلاحاً می‌گوییم نرمال سازی کامل انجام شده است.

تمرینات

۱. رابطه نرمال را تعریف کنید.
۲. مزایا و معایب رابطه نرمال را نام ببرید
۳. رابطه غیر نرمال دارای چه مزایا و معایبی خواهد بود
۴. انواع وابستگی را نام ببرید
۵. وابستگی تابعی را توضیح دهید
۶. حالت‌های وابستگی را نام ببرید
۷. صورت‌های نرمال را نام ببرید
۸. صورت نرمال اول را توضیح دهید
۹. صورت نرمال سوم به چه صورت است
۱۰. صورت نرمال BCNF را شرح دهید
۱۱. صورت نرمال 5NF چگونه است

فصل ۱۰

معماری‌های مختلف سیستم پایگاه‌داده

هدف کلی

در این فصل به موضوع معماری پایگاه‌داده‌ها می‌پردازیم. ابتدا معماری‌های متمرکز پایگاه‌داده‌ها را شرح داده و سپس وارد بحث معماری‌های غیر متمرکز پایگاه‌داده‌ها می‌شویم. در بحث معماری پایگاه‌های داده نامتمرکز، معماری‌هایی مانند معماری مشتری-خدمت‌گزار، معماری توزیع شده، معماری با پردازش موازی، معماری چند پایگاهی و سیستم پایگاهی همراه را با ذکر توضیحات در مورد هر یک از انواع معماری‌های مذکور مورد بحث و بررسی قرار خواهیم داد.

هدف رفتاری

در این فصل عناوین زیر مورد بحث و بررسی قرار می‌گیرند:

- معماری متمرکز
- معماری نامتمرکز
- معماری سیستم پایگاهی مشتری - خدمت‌گزار
- معماری سیستم پایگاهی توزیع شده
- معماری با پردازش موازی
- معماری با حافظه مشترک
- معماری با دیسک‌های مشترک
- معماری بی اجزا مشترک

- معماری سلسله مراتبی
- معماری چند پایگاهی
- سیستم پایگاهی همراه

۱- مقدمه

یکی از مباحث مهم و کلیدی در معماری پایگاه داده، اجزاء تشکیل دهنده و پیکر بندی یا طرز ترکیب اجزاء سیستم و چگونگی تعامل اجزاء با یکدیگر است. در معماری پایگاه داده‌ها حداقل یک پایگاه داده‌ها، یک سیستم مدیریت پایگاه داده‌ها، یک سیستم عامل، یک کامپیوتر با دستگاه‌های جانبی و تعدادی برنامه کاربردی و کاربر وجود دارند.

معماری پایگاه داده‌ها بستگی به دو عنصر اصلی سیستم یعنی سخت افزار و نرم افزار مدیریت پایگاه داده‌ها دارد. البته عوامل دیگری هم در طراحی این معماری دخالت دارند که اهم این عوامل عبارتند از:

- موقعیت جغرافیایی کاربران
- نیازهای کاربران
- ماهیت پردازش‌ها و تراکنش‌ها
- تعداد تراکنش‌ها
- حجم داده‌های ذخیره شدنی
- موقعیت مکانی داده‌ها و ارتباطات بین آنها
- ماهیت کاربردهای مورد نظر

۲- انواع معماری

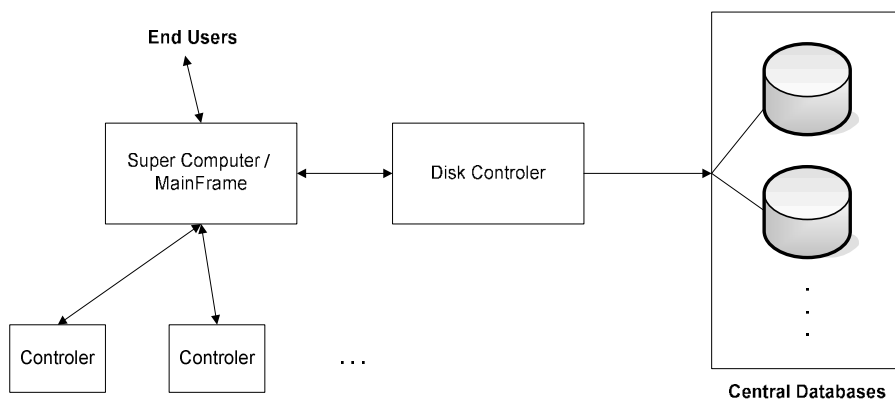
در اساس دو نوع معماری برای سیستم پایگاهی وجود دارد که این دو معماری عبارتند از:

- معماری متمرکز
- معماری نامتمرکز

در ادامه هر یک از انواع معماری مورد بحث و بررسی قرار خواهند گرفت.

۱-۲ معماری متمرکز

در این معماری یک پایگاه داده‌ها روی یک سیستم کامپیوتری و بدون ارتباط با سیستم کامپیوتری دیگر، ایجاد می‌شود. سخت‌افزار این سیستم می‌تواند کامپیوتر شخصی متوسط و یا بزرگ باشد و طبعا قدرت، توانش و کارایی سیستم نیز متفاوت است.



شکل ۱-۱۰ نمایشی از معماری متمرکز

سیستم با معماری متمرکز که روی یک کامپیوتر شخصی ایجاد می‌شود، تک کاربری، برای کاربردهای کوچک و با امکانات محدود است و سیستم مدیریت نیز توانش چندانی ندارد. اما سیستم معماری متمرکز روی کامپیوترهای متوسط و به ویژه بزرگ متصل به تعداد زیادی پایانه، می‌تواند سیستم کارایی باشد. شکل ۱-۱۰ طرحی از معماری متمرکز ارائه شده است.

۲-۲ معماری نامتمرکز

معماری نامتمرکز، خود دارای انواع مختلفی است از جمله:

- معماری مشتری-خدمتگذار
- معماری توزیع شده

- معماری چند پایگاهی
- معماری با پردازش موازی
- معماری موبایل

در ادامه هر یک از انواع معماری‌های غیر متمرکز را مورد بحث و بررسی قرار خواهیم داد.

۲-۲-۱ معماری سیستم پایگاهی مشتری - خدمتگذار

۲-۲-۱-۱ تعریف

معماری مشتری-خدمتگذار تعریف واحد و مورد پذیرش همگان ندارد. در معنای عام می‌توان این معماری را چنین تعریف کرد:

هر معماری که در آن قسمتی از پردازش را یک برنامه، سیستم یا ماشین انجام دهد و انجام قسمت دیگر از پردازش را از برنامه، سیستم یا ماشین دیگر بخواهد، معماری مشتری-خدمتگذار نامیده می‌شود.

- در واقع وظایفی که باید "سیستم" انجام دهد به دو دسته تقسیم می‌شوند:
- دسته‌ای که انجام آن بر عهده خدمتگذار است
 - دسته‌ای که توسط مشتری انجام می‌شود.

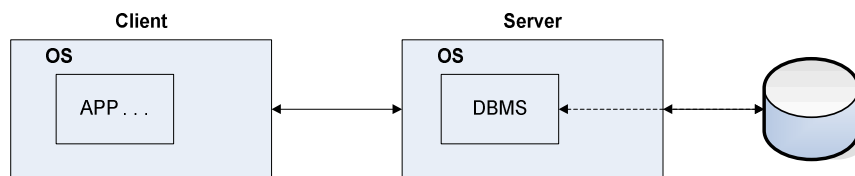
بدین ترتیب یک معماری دو سطحی داریم که برخورد با پیچیدگی سیستم‌های (DBMS) جدید و نیز مشکل توزیع را تسهیل می‌کند. ماشین در این تعریف در معنای عام آن آمده است (فیزیکی یا منطقی). با این تعریف، در این معماری یک ماشین (یا سیستم یا برنامه) خدمتی را به ماشین (یا سیستم یا برنامه) دیگر ارائه می‌کند، از این رو به این ماشین (یا سیستم یا برنامه)، خدمتگذار می‌گویند. بنابراین، منظور از معماری مشتری-خدمتگذار آن نوع از معماری است که در آن مسئولیت‌ها به طور منطقی تقسیم شده است.

توجه: با توجه به این تقسیم کار، ممکن است این موضوع به ذهن برسد که حتما حداقل دو کامپیوتر در این معماری وجود دارد. حال آنکه لزوما چنین نیست، دو

برنامه، دو زیر سیستم از یک سیستم کامپیوتری و... می‌توانند با هم تقسیم کار داشته باشند.... اما در محیط‌های جدید، معمولاً با تعدادی کامپیوتر شخصی، ایستگاه کار، چاپگر، خدمتگذار فایل و تجهیزات دیگر، ایجاد می‌شود.

۲-۱-۲-۲ معماری پایگاهی مشتری-خدمتگذار

در این معماری معمولاً دو گروه کامپیوتر داریم: گروه مشتری و گروه خدمتگذار. تمام داده‌ها در کامپیوترهای خدمتگذار ذخیره می‌شوند و تمام برنامه‌های کاربردی در کامپیوترهای مشتری اجرا می‌شوند. در شکل ۲-۱۰ مثالی از طرح ساده شده این معماری با یک مشتری و یک خدمتگذار دیده می‌شود (گاه موسوم به معماری دو ردیفی. البته معماری سه ردیفی (و از نظر تئوریک، n ردیفی، $n \geq 2$) هم وجود دارد).

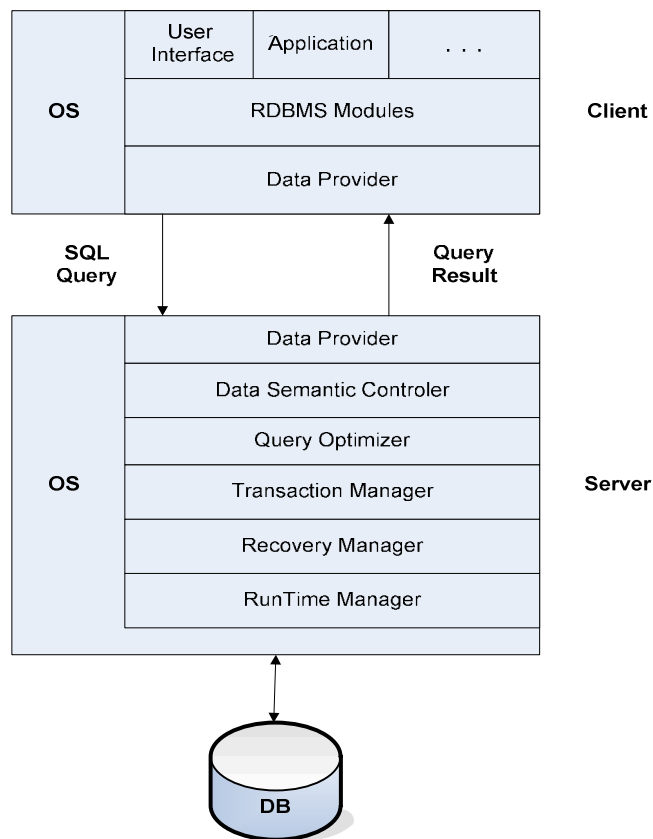


شکل ۲-۱۰ معماری مشتری-خدمتگذار

کاربر در ماشین مشتری می‌تواند از طریق واسط زمانی (مثلاً زبان داده‌ای فرعی) یا واسط‌های دیگر مثل واسط گرافیکی، واسط فرمی و... عمل کند. در شکل ۳-۱۰، عناصر محیط نرم‌افزاری این معماری با فرض وجود یک RDBMS دیده می‌شود. در این شکل، واحدهایی از RDBMS در ماشین مشتری و بقیه واحدهای آن در ماشین خدمتگذار قرار دارند.

برای تسهیل تماس بین مشتری و خدمتگذار، ابزارهایی مثل ODBC و JDBC تولید شده‌اند. این ابزارها در اساس واسط برنامه کاربردی هستند که به مشتری‌ها امکان می‌دهند تا پرسش‌ها به صورت احکام SQL تولید شده و به ماشین خدمتگذار فرستاده شوند. با استفاده از این واسط استاندارد، هر ماشین مشتری می‌تواند با هر ماشین خدمتگذار متصل باشد و نیاز نیست که محیط سیستمی دو ماشین، همگن باشد. در

بعضی از کاربردها، ممکن است واسطه‌های خاص در ماشین مشتری وجود داشته باشد و حتی خود ماشین مشتری هم می‌تواند ماشین خاصی باشد.



شکل ۱۰-۳ یک طرح ممکن از اجزاء نرم‌افزاری معماری مشتری-خدمتگذار

در بعضی از سیستم‌ها، تماس مشتری با خدمتگذار از طریق فراخوان دور انجام می‌شود. تمام فراخوان‌های دور از سوی ماشین مشتری، در یک (یا چند) ترکنش در ماشین خدمتگذار جای داده می‌شوند تا در این ماشین اجرا شوند، به گونه‌ای که اگر تراکنش به هر دلیلی، طرد شود، خدمتگذار می‌تواند تاثیرات اجرای هر یک از فراخوان‌های دور را از بین برده، پایگاه‌داده‌ها را به وضع صحیح قبل از اجرای تراکنش برگرداند.

۲-۱-۲-۲ طرح‌های معماری

۲-۱-۲-۲-۱ از نظر تعداد مشتری و خدمتگذار

از نظر تعداد مشتری و خدمت گذار طرح‌های زیر وجود دارند:

- یک مشتری - یک خدمتگذار
- چند مشتری - یک خدمتگذار
- یک مشتری - چند خدمتگذار
- چند مشتری - چند خدمتگذار

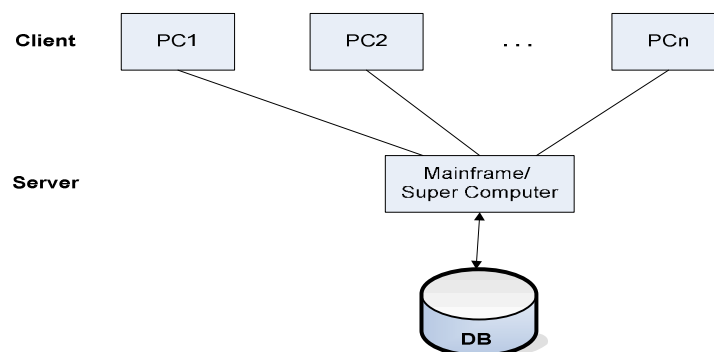
۲-۱-۲-۲-۲ از نظر پیکربندی سخت‌افزاری

از نظر پیکربندی سخت‌افزاری، دو نوع معماری به شرح زیر وجود دارد:

- معماری حول کامپیوتر بزرگ
- معماری حول شبکه

معماری حول کامپیوتر بزرگ

در این طرح، ماشین خدمتگذار یک کامپیوتر بزرگ است و پایگاه‌داده‌ها روی همین کامپیوتر ایجاد و مدیریت می‌شود و تعدادی کامپیوتر شخصی، از خدمات پایگاهی این کامپیوتر بزرگ استفاده می‌کنند. مثالی از طرح این معماری در شکل ۱۰-۴ نشان داده شده است.



شکل ۱۰-۴ معماری مشتری - خدمتگذار حول کامپیوتر بزرگ

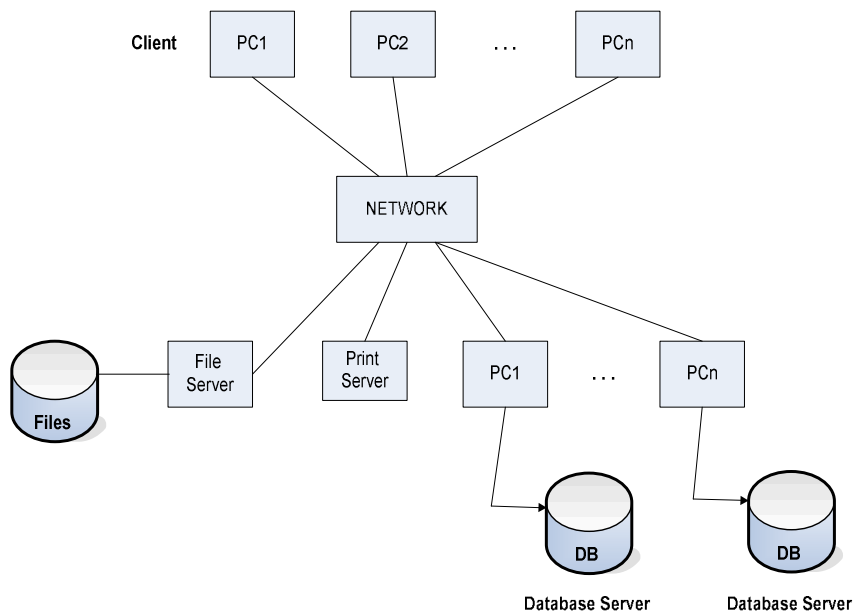
معماری حول شبکه

در این طرح، تعدادی کامپیوتر شخصی به عنوان خدمتگذار و تعدادی دیگر به عنوان مشتری، از طریق شبکه بهم مرتبط‌اند. یک (یا بیش از یک) کامپیوتر شخصی، خدمتگذار پایگاه داده‌هاست و خدمتگذاران دیگری هم می‌توانند وجود داشته باشند. مثالی از طرح این معماری در شکل ۵-۱۰ دیده می‌شود.

۲-۱-۴ مزایای معماری مشتری - خدمتگذار

در مقایسه با معماری متمرکز، این نوع معماری مزایای زیر را دارد:

- تقسیم پردازش
- کاهش ترافیک شبکه (در معماری حول شبکه)
- استقلال ایستگاههای کار
- اشتراک داده‌ها



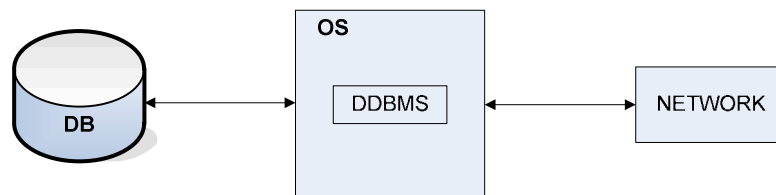
شکل ۵-۱۰ معماری مشتری - خدمتگذار حول شبکه

۲-۲-۲ معماری سیستم پایگاهی توزیع شده

۲-۲-۲-۱ تعریف

مجموعه‌ای از داده‌های ذخیره شده، که منطقاً به یک سیستم تعلق دارند، ولی در کامپیوترهای مختلف که در یک یا بیش از یک شبکه توزیع شده‌اند، قرار گرفته‌اند. می‌توان گفت که در این معماری تعدادی پایگاه‌داده‌های ذخیره شده روی کامپیوترهای مختلف داریم که از نظر کاربران، پایگاه واحدی هستند. به بیان دیگر، این معماری مجموعه‌ای است از چند پایگاه‌داده منطقاً یکپارچه (مجموع)، ولی به طور فیزیکی توزیع شده روی یک شبکه کامپیوتری. توجه داشته باشید که در این معماری، در سطح طراحی منطقی پایگاه، در آغاز یک پایگاه‌داده‌های یکپارچه داریم که طراح بر اساس یک استراتژی توزیع و یک طرح تخصیص مشخص، داده‌هایش را در چند کامپیوتر توزیع می‌کند. کامپیوترها با یکدیگر چنان همکاری دارند که هر کاربر می‌تواند به داده‌های مورد نیازش در هر کامپیوتر دستیابی داشته باشد به گونه‌ای که انگار داده‌ها در کامپیوتر خودش ذخیره شده باشند.

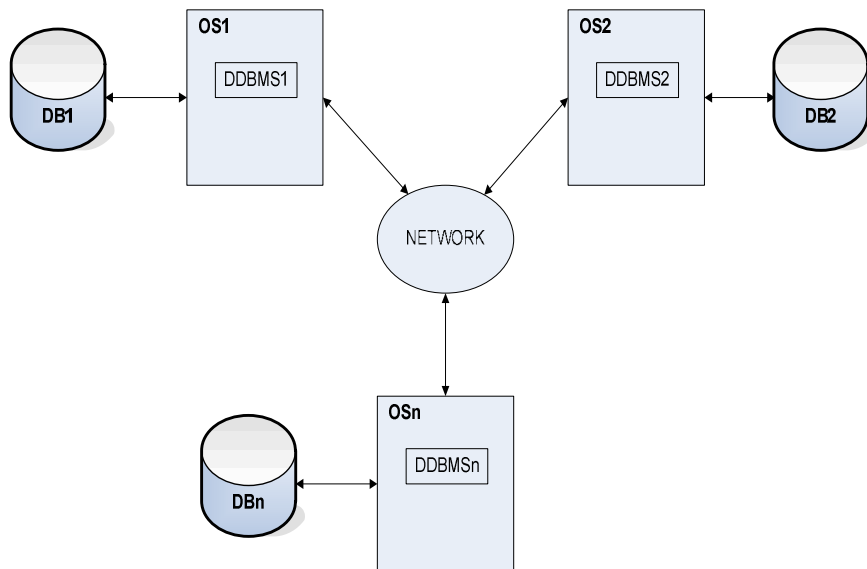
در این معماری هر کامپیوتر خود یک سیستم پایگاه‌داده‌هاست یعنی: پایگاه‌داده‌ها، سیستم مدیریت پایگاه‌داده‌ها و مدیر انتقال داده‌ها دارد. اصطلاحاً می‌گوییم تعدادی DBMS محلی داریم و برای ایجاد هماهنگی بین این سیستم‌های محلی، عنصر نرم‌افزاری خاصی که نوعی گسترش DBMS است، لازم است. در واقع در هر کامپیوتر، یک DDBMS داریم، یعنی یک DBMS با توانش ایجاد و مدیریت پایگاه‌داده‌های توزیع شده. بنابراین نمای کلی هر کامپیوتر به صورتی است که در شکل ۶-۱۰ دیده می‌شود.



شکل ۶-۱۰ نمای یک کامپیوتر در معماری توزیع شده

کامپیوترها از طریق امکانات شبکه‌ای (محلی، گسترده و یا متحرک) به هم مرتبط‌اند و هر کامپیوتر اجزا و عناصر سخت‌افزاری و نرم‌افزاری خود را دارد. این اجزا و عناصر می‌توانند همگن یا ناهمگن باشند. مثالی از طرح کلی این معماری در شکل ۷-۱۰ دیده می‌شود. با توجه به تعریف این نوع معماری و طرح کلی آن، ویژگیهای این سیستم را می‌توان چنین برشمرد:

- مجموعه‌ای است از داده‌های منطقاً مرتبط و اشتراکی
- بعضی بخشها ممکن است بطور تکراری (در چند نسخه) در کامپیوترها ذخیره شده باشند.
- کامپیوترها از طریق یک شبکه بهم مرتبط‌اند.
- داده‌های ذخیره شده در هر کامپیوتر تحت کنترل یک DBMS است.
- DBMS در هر کامپیوتر، می‌تواند برنامه‌های کاربردی محلی، را بطور اتوماتیک اجرا کند.
- هر DBMS حداقل در اجرای یک برنامه کاربردی سرتاسری مشارکت دارد.

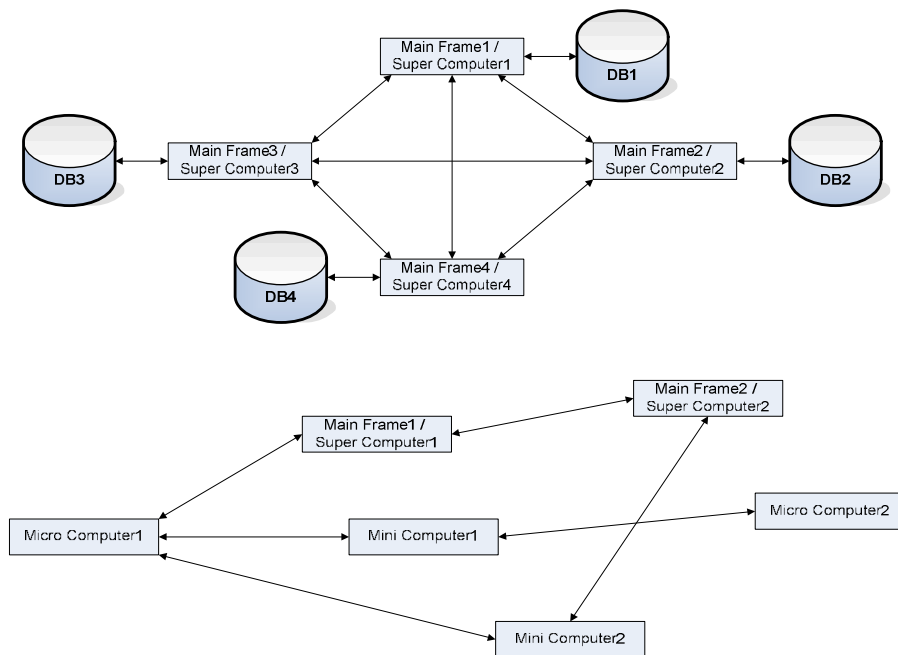


شکل ۷-۱۰ معماری توزیع شده با سه کامپیوتر

۲-۲-۲-۲ پیکربندی سخت‌افزاری

گفتیم که اجزاء تشکیل دهنده این معماری می‌توانند همگن یا ناهمگن باشند، به ویژه از نظر سخت‌افزاری، پیکربندی‌های گوناگونی پنداشتنی است. در مثال شکل ۸-۱۰ دو پیکربندی دیده می‌شود. در این معماری اجزاء زیر می‌توانند همگن یا ناهمگن باشند:

- سخت‌افزار پردازشگر
- سیستم عامل
- سیستم مدیریت پایگاه داده (بوژه از نظر مدل داده‌ای و زبان پایگاهی)
- پروتوکل‌های شبکه
- سخت‌افزار ارتباط
- سخت‌افزار ذخیره سازی
- تسهیلات و ابزارهای (نرم‌افزاری) جانبی



شکل ۸-۱۰ دو پیکربندی سخت‌افزاری در معماری توزیع شده

۲-۲-۳ قواعد

اصل مهم در معماری سیستم پایگاهی توزیع شده این است که "سیستم" باید چنان عمل کند که کاربران دقیقاً مثل محیط پایگاه داده‌های متمرکز معمولی از آن استفاده کنند (و این ویژگی، یکی از تفاوت‌های این معماری با معماری مشتری-خدمتگذار است). برای رعایت این اصل مهم، در هر سیستم پایگاهی توزیع شده قواعدی باید رعایت شوند که در واقع اهداف این سیستم هم هستند. برخی از این عوامل عبارتند از:

- خود مختاری محلی (داخلی)
- تداوم عملیات
- ناوابستگی کامپیوترها به کامپیوتر اصلی
- ناوابستگی برنامه‌ها به مکان ذخیره سازی داده‌ها
- ناوابستگی برنامه‌ها به طرز جابجایی داده‌ها در کامپیوترها.
- پردازش پرسشها به گونه‌ای توزیع شده
- ناوابستگی برنامه‌ها به سخت افزار
- ناوابستگی برنامه‌ها به سیستم عامل
- ناوابستگی برنامه‌ها به سیستم مدیریت پایگاه داده‌ها
- ناوابستگی برنامه‌ها به شبکه

۲-۲-۴ مزایا

برخی از مزایای این معماری عبارتست از:

- سازگاری و هماهنگی با ماهیت سازمان‌های نوین
- کارایی بیشتر در پردازش داده‌ها به ویژه در پایگاه داده‌های بزرگ
- دستیابی بهتر به داده‌ها
- اشتراک داده‌ها
- افزایش پردازش موازی
- کاهش هزینه ارتباطات
- تسهیل گسترش سیستم

- استفاده از پایگاه‌داده‌های از قبل موجود

۲-۲-۵ معایب

این معماری معایبی هم دارد از جمله:

- پیچیدگی طراحی سیستم
- پیچیدگی پیاده سازی
- کاهش کارایی در برخی موارد
- هزینه بیشتر
- مصرف حافظه بیشتر

۲-۳ معماری با پردازش موازی

این معماری با ساخت و گسترش ماشین‌های موازی، برای ایجاد پایگاه‌داده‌های خیلی بزرگ و نیز در سیستم‌هایی که چند هزار تراکنش در ثانیه را پردازش می‌کند، مورد توجه قرار گرفت. گونه گسترش یافته معماری توزیع شده است و برای تامین کارایی بیشتر، دستیابی پذیری بالا و گسترش پذیری سریع طراحی می‌شود. در اینگونه سیستم‌ها معمولاً تعداد زیادی تراکنش در ثانیه (حدود چند هزار) و بطور موازی اجرا می‌شوند. مطالعه ماشین‌های موازی از حدود این درس خارج است، تنها اشاره می‌شود که ماشین موازی بر اساس یکی از دو طرح کلی زیر طراحی و تولید می‌شود:

- چند پردازنده قوی (دو یا چهار در حال حاضر و گاه بیشتر)
- تعدادی پردازنده کوچک (گاه چند سطر یا حتی بیشتر)

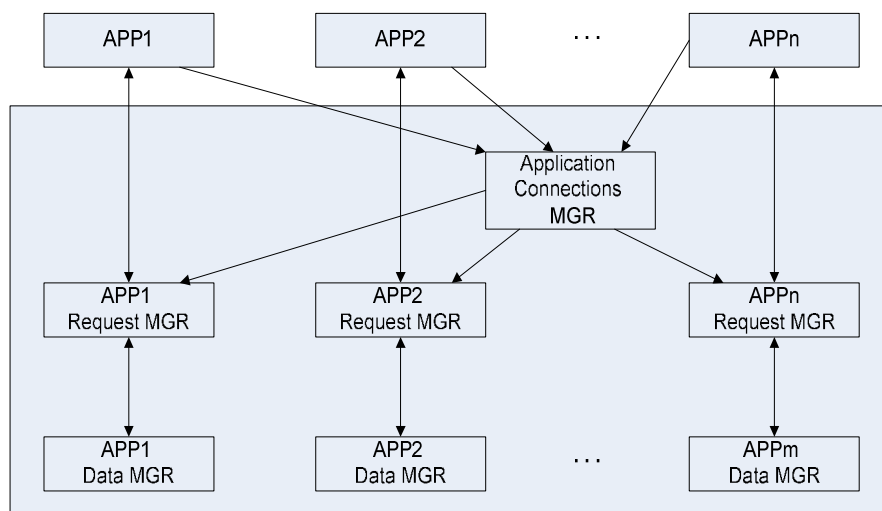
در ماشین با پردازش موازی دو هدف اساسی وجود دارد:

- افزایش سرعت (کارایی)، یعنی انجام یک کار در زمان کمتر
- افزایش مقیاس کار (گسترش کار) یعنی انجام کارهای بزرگ تر در زمان کمتر.

۲-۳-۲-۲ طرح کلی معماری

این معماری به صورت کلی در شکل ۹-۱۰ دیده می‌شود. می‌بینیم که در چنین سیستمی، در اساس سه واحد اصلی وجود دارد:

- مدیر تماس‌های اجرایی کاربران
- مدیر درخواست‌ها
- مدیر داده‌ها



شکل ۹-۱۰ طرح کلی معماری موازی

۲-۳-۲-۲ طرح‌ها

برای ایجاد پایگاه‌داده‌ها با معماری پردازش موازی، به طور کلی، چهار مدل یا طرح وجود دارد:

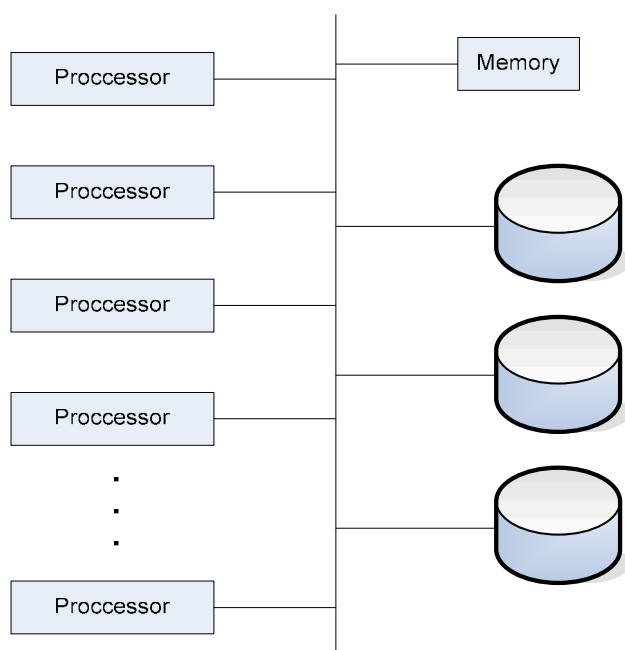
- معماری با حافظه مشترک
- معماری با دیسک مشترک
- معماری بی اجزاء مشترک
- معماری سلسله مراتبی

معماری‌های مختلف سیستم پایگاه داده ۲۲۳

برای ایجاد پایگاه‌داده‌های موازی، علاوه بر استفاده از یکی از این معماری‌ها، روش‌هایی نیز برای جایدهی داده‌ها روی دیسک‌ها و یا در حافظه‌های نهان و نیز بهینه‌سازی پرسش‌های کاربران به کار گرفته می‌شوند، بنابراین صرف وجود معماری موازی برای ایجاد پایگاه‌داده‌های موازی کافی نیست.

۲-۲-۳-۱ معماری با حافظه مشترک

در این طرح، پردازنده‌ها به حافظه مشترک دسترسی دارند. مزیت این طرح این است که ارتباط بین پردازنده‌ها بطور کارا انجام می‌شود. زیرا پیام‌های بین پردازنده‌ها با نوشتن در حافظه مبادله می‌شود که زمان آن کمتر از میکروثانیه است. داده‌های ذخیره شده در حافظه در اختیار همه پردازنده‌ها قرار دارند.



شکل ۱۰-۱۰ معماری با حافظه مشترک

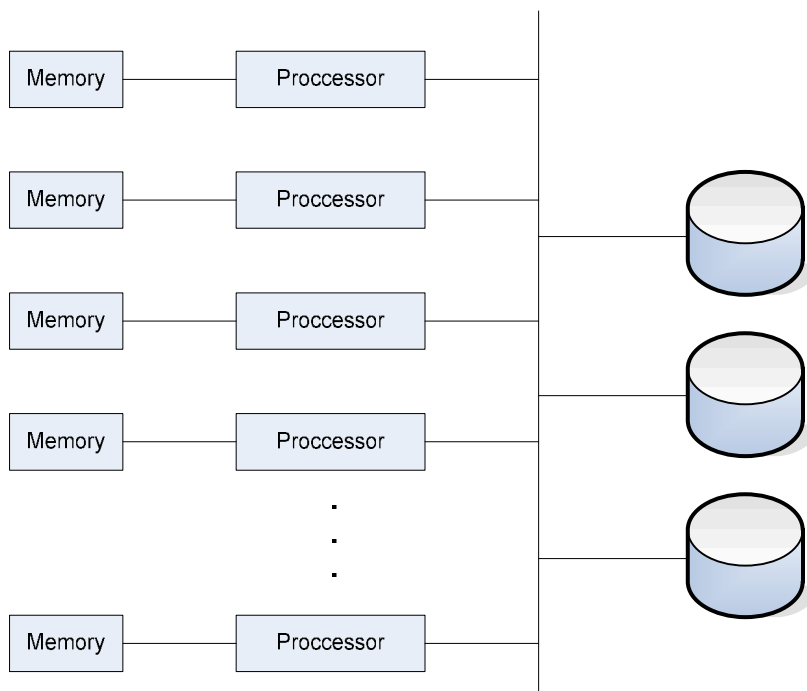
عیب این معماری در این است که نمی‌توان بیش از ۳۲ یا ۶۴ پردازنده داشت، زیرا احتمال بروز تنگنا در بازسای حافظه‌ای یا شبکه ارتباطی افزایش می‌یابد. البته اگر

در هر پردازنده حافظه نهان با اندازه بزرگ وجود داشته باشد، دفعات مراجعه به حافظه اصلی کاهش می‌یابد، هر چند نمی‌توان همه داده‌ها را در این بافرها جای داد. طرح کلی این معماری در شکل ۱۰-۱۰ نشان داده شده است (در این طرح و طرح‌های بعدی معماری موازی، P به جای پردازنده و M به جای حافظه است).

۲-۲-۳-۲-۲ معماری با دیسک‌های مشترک

در این طرح، تمام پردازنده‌ها به تمام دیسک‌ها از طریق شبکه ارتباطی دسترسی دارند. هر پردازنده، حافظه اختصاصی خود را دارد. مزایای این نوع معماری به شرح ذیل می‌باشد:

- عدم بروز تنگنا در باس‌های حافظه
- تسهیل تحمل خرابی، زیرا در صورت خراب شدن یک پردازنده یا حافظه، پردازنده دیگر می‌تواند کار را ادامه دهد. می‌توان از سیستم RAID هم استفاده کرد.

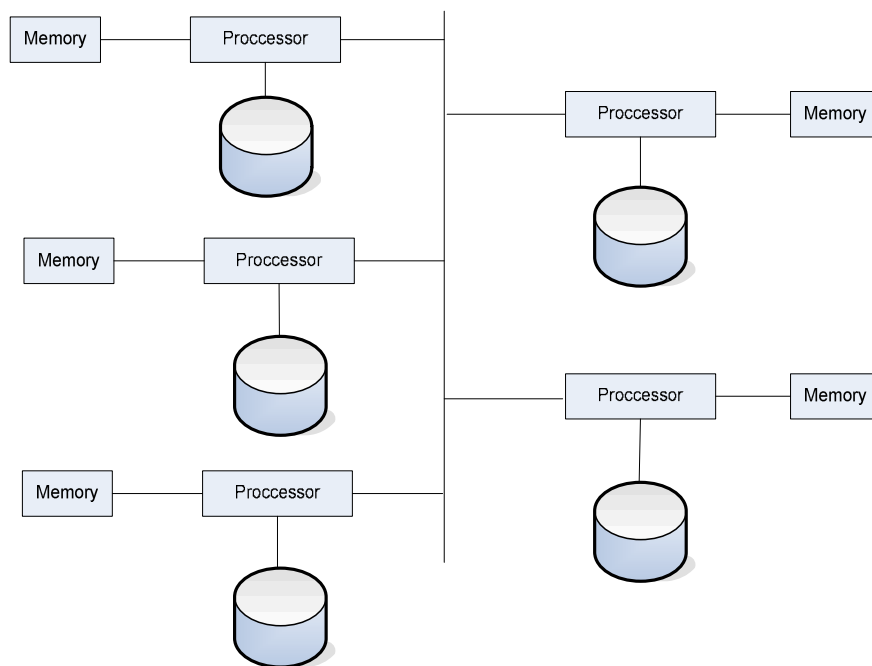


شکل ۱۰-۱۱ معماری با دیسک‌های مشترک

این معماری دارای عیوبی نیز است که مهمترین عیب آن دشواری در گسترش سیستم است، زیرا با افزایش دیسک‌ها و پردازنده‌ها، در ارتباط بین اجزا، تنگنا ایجاد می‌شود و سرعت ارتقا بین آنها کاهش می‌یابد. طرح کلی این سیستم در شکل ۱۰-۱۱ نشان داده شده است.

۲-۲-۳-۲ معماری بی اجزاء مشترک

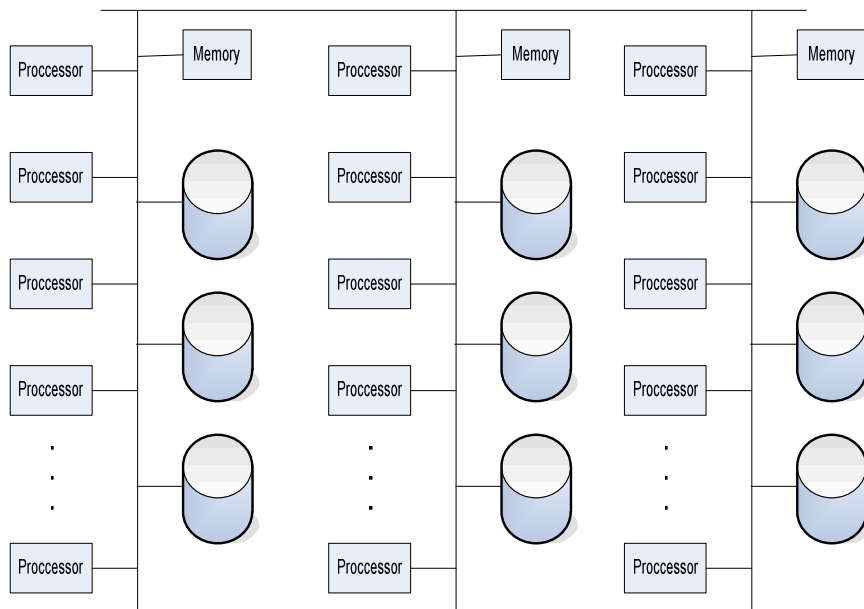
در این طرح، هر ماشین، پردازنده، حافظه و دیسک (های) خود را دارد. یک شبکه ارتباطی با سرعت بالا این ماشین‌ها را به هم مرتبط می‌کند. هر ماشین نوعی خدمت‌گزار پایگاهی است. در شکل ۱۰-۱۲، طرح کلی این معماری دیده می‌شود. مزیت این نوع معماری در تسهیل گسترش است. ولی عیب این نوع معماری در این است که هزینه ارتباط و دستیابی‌های غیر محلی زیاد است.



شکل ۱۰-۱۲ معماری بی اجزاء مشترک

۴-۲-۳-۲ معماری سلسله مراتبی

در این طرح، ویژگی‌های سه طرح پیش با هم وجود دارد. در بالاترین سطح سیستم، تعدادی گره با شبکه ارتباطی به هم مرتبط‌اند و اجزا مشترک ندارند. پس در این سطح، معماری بی اشتراک داریم. هر گره خود می‌تواند تعداد کمی پردازنده با حافظه مشترک داشته باشد یا یک سیستم با دیسک‌های مشترک باشد. بنابراین یک سلسله مراتب از معماری‌های پیش گفته ایجاد می‌شود. طرح کلی این معماری در شکل ۱۰-۱۳ دیده می‌شود.



شکل ۱۰-۱۳ معماری سلسله مراتبی

۴-۲-۲ معماری چند پایگاهی

این معماری نوعی معماری نامتمرکز است که در آن، کامپیوترها خود مختاری کامل دارند. در سالهای اخیر گرایش زیادی به این معماری ایجاد شده است. در این معماری، معمولاً چند سیستم با معماری توزیع شده و یا چند سیستم با معماری متمرکز از قبل وجود دارد و هر سیستم، روی عملیات محلی خود کنترل

کامل دارد. اما برای آنکه کاربران بتوانند نیازهای اطلاعاتی خود را تامین کنند بی‌آنکه مستقیماً با اجزاء معماری مرتبط باشند و در واقع بتوانند از یک سیستم چند پایگاهی بهره‌برداری کنند، به یک لایه نرم‌افزاری خاصی نیاز است. این لایه نرم‌افزاری امکان می‌دهد تا در هر سیستم، مدیر پایگاه‌داده‌ها روی داده‌های پایگاه خود کنترل کامل داشته باشد و نیازی به کنترل متمرکز نباشد و در عین حال خدمات لازم به کاربران داده شود.

این لایه نرم‌افزاری در سطح بالاتر از DBMS ها و احیاناً سیستم‌های فایلینگ موجود در پیکر بندی سیستم چند پایگاهی، قرار دارد. این لایه چنان عمل می‌کند که کاربران سیستم‌های مختلف، مجموعه پایگاه‌ها را به صورت یک پایگاه واحد می‌بینند. واضح است که در سیستم چند پایگاهی، یک شمای ادراکی جامع (سراسری) و در هر یک از DBMS ها، یک شمای ادراکی محلی وجود دارد (و نیز شمای دیگر، طبق آنچه که در معماری ANSI دیدیم. در چنین سیستمی طبعاً به یک کاتالوگ جامع نیاز است. این معماری علاوه بر مشکلات تکنیکی موجود در یک سیستم توزیع شده، مشکلات خاص خود را نیز دارد که در اینجا از ذکر آنها خودداری می‌شود.

۲-۲-۵ سیستم پایگاهی همراه

۲-۲-۵-۱ تعریف

با رشد سریع تکنولوژی ارتباطات، اینک دیگر انسان می‌تواند از هر جا، به هر داده ذخیره شده در هر جای دیگر دستیابی داشته باشد، هر چند هنوز محدودیت‌هایی در این "داده فرستی- داده گیری" وجود دارد. اما تا آنجا که به حیطه دانش و تکنولوژی پایگاه‌داده‌ها مربوط می‌شود، نوع نوینی از سیستم پایگاه‌داده‌ها پدید آمده و در حال گسترش یافتن است: سیستم پایگاه‌داده‌های همراه (جایجا شونده). ما در قسمتهای قبل انواع معماری‌های سیستم پایگاهی را دیدیم، از جمله معماری مشتری-خدمتگذار. اما این معماری و حتی گونه گسترده تر آن، یعنی معماری پایگاه توزیع شده، هنوز مشکلاتی از جمله "عملی نبودن در همه جا"، ایمنی سیستم و داده‌ها، هزینه و... دارند. سیستم پایگاه‌های همراه راه حلی است برای برخی از این دشواری‌ها، به ویژه در کاربردهای شخصی.

۲-۵-۲ معماری سیستم پایگاهی همراه

در این معماری، یک یا بیش از یک کامپیوتر متوسط یا بزرگ، نقش خدمتگذار پایگاهی را ایفا می‌کند. این کامپیوتر را خدمتگذار میزبان می‌نامند. هر کاربر، کامپیوتر کوچک همراه خود را دارد (با خود حمل می‌کند: نوعی دفترکار همراه). در این کامپیوتر داده‌های عملیاتی و برنامه‌های کاربردی مورد نیازش ذخیره شده‌اند. پایگاه داده‌های ذخیره شده در کامپیوتر خدمتگذار میزبان، حاوی داده‌های پویا است. بدین ترتیب کاربر می‌تواند از هر جا، با زیر سیستم خدمتگذار پایگاهی مرتبط شده و پردازش‌های مورد نظرش را انجام دهد. در این سیستم لزومی ندارد که پایگاه‌های همراه و پایگاه میزبان در یک شبکه باشند. ارتباط بین "همراه" و میزبان در بازه‌های زمانی نامنظم و معمولاً کوتاه برپا می‌گردد. بین کامپیوترهای همراه، مگر در شرایط خاص، چندان ارتباط برقرار نمی‌گردد.

دو جنبه اساسی این سیستم که باید در طراحی و پیاده سازی به آن توجه شود عبارتند از: مدیریت کارای پایگاههای همراه و ایجاد ارتباط پویا و سریع بین پایگاههای همراه و پایگاه میزبان.

تمرینات

۱. دو نوع اصلی معماری پایگاه داده‌ها را نام ببرید؟
۲. معماری متمرکز را شرح دهید؟
۳. انواع معماری نامتمرکز را نام ببرید؟
۴. سیستم پایگاهی مشتری-خدمت گذار را توضیح دهید؟
۵. معماری سیستم پایگاهی توزیع شده را توضیح دهید؟
۶. مزایای معماری سیستم پایگاهی توزیع شده را نام ببرید؟
۷. چهار مدل معماری با پردازش موازی را نام ببرید؟
۸. معماری پایگاه داده همراه را توضیح دهید؟

ضمیمه ۱

مجموعه سئوالات

تست های سری ۱

۱- کدامیک از موارد زیر بیانگر اختلاف بین پایگاه داده ها و پایگاه بصیرت (Knowledge Base) می باشد ؟ (کارشناسی ارشد- آزاد۷۲)

۱. هر دو پایگاه از یک نوع بوده و فرقی با هم ندارند.
۲. پایگاه بصیرت دینامیک ولی پایگاه داده های استا (static) می باشد.
۳. پایگاه داده ها اغلب با زبان SQL و پایگاه بصیرت اغلب با زبان QUEL کد می شود.
۴. هیچکدام

۲- اطلاعات (Information) به چه معنی می باشد ؟ (مسابقات آموزشکده های فنی)

۱. داده هایی که روی آنها عمل مقایسه ای صورت گیرد.
 ۲. داده هایی که روی آنها عمل محاسبات صورت گیرد.
 ۳. داده هایی که به نمایش در آمده باشند.
 ۴. داده هایی که روی آنها عمل پردازش صورت گیرد.
- ۳- داده + ویژگی + موجودیت = (مسابقات آموزشکده های فنی -۸۰)
۱. اطلاع
 ۲. پردازش
 ۳. پرونده

۴. دانش

۴- کدام گزینه در رابطه با نسل دوم ذخیره و بازیابی اطلاعات نادرست است ؟

۱. این نسل به نسل شیوه‌های دستیابی معروف است.
۲. تکرار ذخیره سازی در این نسل بسیار کاهش یافته.
۳. تغییر در رسانه‌های ذخیره سازی بر روی برنامه‌های کاربردی تاثیر چندانی ندارد.

۴. در این نسل از رسانه‌های با دستیابی مستقیم مثل دیسک استفاده شده است.

۵- کدام تعریف در رابطه با ((داده)) درست است ؟

۱. نمایش واقعیات، پدیده‌ها، مفاهیم یا معلومات به صورتی صوری و مناسب برای برقراری ارتباط، تفسیر یا پردازش توسط انسان یا امکانات خودکار
۲. از نظر ساختاری و از دیدگاه بانک اطلاعاتی، داده عبارت است از مقادیر صفات خاصه انواع موجودیتها

۳. هر دو گزینه ۱ و ۲

۴. اطلاعات کاراکتری که وارد سیستم دستی شده و به آن معنایی منتسب می‌شود.

۶- کدام عبارت درست است ؟

۱. داده همان اطلاعات است.
۲. داده اطلاعات پردازش شده است.
۳. اطلاعات، داده پردازش شده است.
۴. اطلاعات مقادیر خام است.

۷- Attribute عبارت است از.....

۱. همان موجودیت است.
۲. صفات خاصه توضیحی برای هر موجودیت است.
۳. همان رکورد است.

۴. سطرهای یک جدول است.

۸- File System چیست ؟

۱. همان DBMS است.

۲. به ساختار کلی نامگذاری، ذخیره سازی و سازماندهی فایلها در یک سیستم عامل گفته می شود.

۳. انباره ای برای یک مجموعه از فایلهای داده ای است.

۴. هر سه گزینه.

۹- کدام گزینه از ویژگی های نسل اول ذخیره و بازیابی اطلاعات نیست ؟

۱. نسخه های متعددی از یک فایل نگهداری می شود.

۲. ساختار فایل ها ترتیبی است.

۳. پردازش در محیط های بلادرنگ و Online می تواند انجام شود.

۴. طراحی ساختار فیزیکی فایلها هم، بر عهده کاربر است.

۱۰- کدام گزینه در رابطه با بانک های اطلاعاتی و بانکهای معرفت صحیح است ؟

۱. بانک معرفت حاوی تعداد زیادی واقعیت ساده است که به طور صریح بیان شده اند، همراه با تعداد کمی قواعد عام که به طور ضمنی بیان می شوند.

۲. بانک معرفت حاوی مجموعه ای از واقعیت های ساده و قواعد عام که به طور صریح بیان شده اند.

۳. بانک اطلاعاتی مجموعه ای است از تعداد زیادی واقعیت ساده که به طور صریح بیان شده اند. همراه با تعداد کمی قواعد عام که به طور ضمنی بیان می شوند.

۴. ۲ و ۳

۱۱- جزء داده ها (Data Element) چیست ؟ (مسابقات آموزشدهای فنی -۷۶)

۱. قسمتی از اطلاعات است.

۲. محلی موقت برای یک مقدار اطلاعات است.

۳. مقداری است که در فیلد ذخیره می شود.

۴. هر مقداری می تواند باشد.

پاسخ تست های سری ۱

۱- (۲) پایگاه بصیرت به تدریج بر بصیرتهایش (Knowledge Base) می افزاید.

۲- (۴) داده ورودی سیستم است که پس از پردازش تبدیل به اطلاعات شده و از سیستم خارج می شود.

۳- (۱)

۴- (۲) در نسل دوم تکرار ذخیره سازی هنوز در حد نسبتا بالایی وجود دارد.

۵- (۳) گزینه ۴ کامل نیست. داده هر نمایشی اعم از کاراکتری یا کمیتهای آنالوگ و یا غیره است. یعنی فقط به اطلاعات کاراکتری داده گفته نمی شود و داده می تواند وارد هر سیستمی شود و نه فقط سیستم دستی.

۶- (۳)

۷- (۲)

۸- (۲)

۹- (۳) گزینه ۳ از ویژگی های نسل دوم است.

۱۰- (۴)

۱۱- (۲)

تست های سری ۲

۱- شکل زیر در نمودار ER چه مفهومی را می‌رساند ؟

۱. هر استاد فقط و فقط یک درس (نه کمتر و نه بیشتر) و هر درس توسط یک استاد ارائه می‌شود.

۲. هر استاد یک درس و هر درس توسط یک استاد ارائه می‌شود. ممکن است استادی درسی ارائه نکند.

۳. هر استاد یک درس و هر درس توسط یک استاد ارائه می‌شود.

۴. یک استاد یک درس و یک درس توسط یک استاد ارائه می‌شود. ولی ممکن است استاد آن درس تغییر کند.

۲- در نمودار EER کدام گزینه درست است ؟

۱. صفت مشتق را با زیر خط، صفت کلیدی را با ۲ خط و صفت چند مقداری را با خط چین نشان می‌دهیم.

۲. صفت مشتق را با خط چین، صفت مرکب را با زیر خط و صفت وابسته را با ۲ خط نمایش می‌دهیم.

۳. صفت مشتق را با زیر خط، صفت چند مقداری را با ۲ خط و صفت کلیدی را با خط چین نشان می‌دهیم.

۴. صفت مشتق را با خط چین، صفت کلیدی را با زیر خط و صفت چند مقداری را با ۲ خط نشان می‌دهیم.

۳- به ارتباط بین موجودیتها.... گویند.

۱. Entity

۲. Attribute

۳. Tuple

۴. ER

۴- در مدل رابطه‌ای کدام نوع صفت را نداریم ؟

۱. صفت کلید

۲. صفت چند مقداری

۳. صفت مشتق

۴. هر سه را داریم

۵- کدام گزینه در رابطه با نمودار EER زیر صادق است ؟

۱. به محض حذف دانشجو از بانک اطلاعاتی، وابستگان او نیز حذف خواهند شد.

۲. وابستگان یک موجودیت قوی می باشند.

۳. موجودیت دانشجو، کلید اصلی موجودیت وابستگان را به ارث می برد.

۴. ۱ و ۲

۶- کدام گزینه نادرست است ؟

۱. در نمودار ER هر ارتباطی بین موجودیتها دارای سمانتیک خاصی است.

۲. صفت مشتق صفتی است که به کمک صفتهای دیگر می توان آن را محاسبه کرد.

۳. معنای نمودار ER زیر آن است که ارتباط استاد اختیاری است.

۴. هیچکدام

۷- شکل زیر نمایانگر چه نوع ارتباطی است ؟

۱. existence dependency

۲. is-a

۳. cardinality

۴. connectivity

۸- ارتباط بین موجودیتها کدام است ؟

۱. Attribute

۲. record

۳. ER

۴. entity

۹- منظور از Connection trap چیست ؟

۱. استنتاج نادرست از ارتباط بین موجودیتها

۲. ارتباطات دو طرفه نادرست

۳. ارتباط نادرست میان سه entity

۴. هر سه گزینه

۱۰- کدام موجودیت از نوع weak می باشد ؟

۱. استاد

۲. درس

۳. خانواده استاد

۴. دانشجو

۱۱- کدام مورد جزء انواع صفات خاصه نمی باشد ؟

۱. ذخیره شده یا مشتق

۲. تک مقداری یا چند مقداری

۳. ساده یا مرکب

۴. معرفه یا نکره

۱۲- کدام گزینه ویژگی موجودیت نمی باشد ؟

۱. یک نوع موجودیت معمولاً بیش از یک صفت دارد.

۲. موجودیت تک صفتی وجود ندارد.

۳. موجودیت تک نمونه ای وجود دارد.

۴. یک نوع موجودیت معمولاً نمونه های متمایز از یکدیگرند.

۱۳- کدام گزینه در رابطه با کلید (صفت شناسه) درست می باشد ؟

۱. می تواند یکتا نباشد.

۲. می تواند هیچ مقدار (NULL) باشد.

۳. می تواند مرکب باشد.

۴. می تواند مشتق باشد.

۱۴- در مسابقه بسکتبال ارتباط موجودیتهای گل و بازیکن چگونه است ؟

۱. ارتباط بازیکن با گل به صورت 1:n می باشد.

۲. ارتباط بازیکن با گل به صورت n:n می باشد.

۳. ارتباط بازیکن با گل به صورت 1:1 می باشد.

۴. نامشخص است.

پاسخ تست های سری ۲

۱- (۱) اگر علامت نقطه داخل مستطیل موجودیت باشد یعنی شرکت در ارتباط

اجباری است. یعنی استاد ها نمی توانند درس ندهند.

۲- (۴)

۳- (۴)

۴- (۲) مثلاً صفت مدرک برای استاد چند مقداری است چرا که ممکن است یک

استاد چند مدرک داشته باشد.

۵- (۱) ممکن است وجود یک پدیده وابسته به وجود پدیده دیگری باشد، یعنی

در صورت حذف عضوی از آن پدیده، عضوهای وابسته هم لازم باشد به طور

خودکار حذف شوند. این نوع وابستگی را وابستگی وجودی و پدیده وابسته را

موجودیت ضعیف weak entity می نامند. پدیده وابسته باید کلید پدیده اصلی

را که به آن وابسته است به ارث ببرد. پدیده وابسته را با دو مستطیل تودرتو

نمایش می دهیم.

۶- (۳) مشارکت یک نوع موجودیت در یک نوع ارتباط را الزامی (یا کامل)

می گوئیم اگر تمام نمونه های آن نوع موجودیت در آن نوع ارتباط شرکت کنند.

در غیراینصورت مشارکت غیر الزامی (اختیاری یا ناکامل) است.

اجباری_الزامی_کامل = Total participation

اختیاری _ غیر الزامی_ناکامل = Partial participation

در بعضی کتابها مشارکت الزامی را با دو خط موازی که موجودیت را به ارتباط

متصل می کند نمایش می دهند. پس مشارکت استاد در تدریس الزامی است

یعنی هر استاد حتما حداقل باید یک درس را تدریس کند.

۷- (۲) ارتباطاتی که از نماد مثلث (مفهوم ارث بری) استفاده می‌کنند. ارتباطات ((هست)) یا ((is-a)) نامیده می‌شوند.

۸- (۳)

۹- (۱)

۱۰- (۳) با حذف استاد اطلاعات خانواده او نیز حذف می‌گردد.

۱۱- (۴)

۱۲- (۲) موجودیت تک صفتی می‌تواند وجود داشته باشد مثلاً جدولی که فقط یک ستون دارد.

۱۳- (۳) فیلد کلید نمی‌تواند مقدار نداشته باشد یعنی نمی‌تواند NULL باشد.

۱۴- (۱) یک بازیکن می‌تواند چند گل بزند ولی هر گل توسط یک بازیکن زده می‌شود.

تست های سری ۳

۱- مزایای یک پایگاه داده ها (Data Base) نسبت به فایل های متعارف چیست ؟
(کارشناسی ارشد - آزاد ۷۲)

۱. ۴ و ۲

۲. کنترل حساب شده مقدار افزونگی در پایگاه داده ها

۳. اطمینان از صحت داده ها (Data Validation) کمتر مورد نیاز خواهد بود.

۴. دستیابی مشترک به داده ها

۲- کدامیک از موارد زیر جزء وظایف DBA نمی باشد ؟ (کارشناسی ارشد - آزاد ۷۲)

۱. نوشتن Data Dictionary برای پایگاه داده ها

۲. نظارت بر عملکرد پایگاه داده ها (Performance Monitoring)

۳. تهیه رویه ها و استراتژی تهیه Backup و نحوه احیای پایگاه داده ها

۴. تهیه schema برای پایگاه داده ها

۳- در یک سیستم مدیریت پایگاه داده ها (DBMS) کدامیک از امکانات زیر جزء عناصر اصلی تشکیل دهنده DBMS محسوب نمی شوند: (کارشناسی ارشد - دولتی ۷۲)

۱. امکان پردازش زبان طبیعی برای کار با پایگاه

۲. امکان کار با داده ها به کمک یک DSL (Data Sublanguage)

۳. امکان تامین جمعیت و بی نقصی (integrity) پایگاه

۴. امکان تامین ایمنی پایگاه

۴- دو مرحله از مراحل طراحی یک بانک اطلاعاتی عبارتند از طراحی ادراکی (Conceptual Design) و طراحی منطقی (Logical Design) این دو چه تفاوت اساسی با هم دارند ؟ (کارشناسی ارشد - دولتی ۷۵)

۱. طراحی ادراکی به مدل خاصی مربوط می شود و پس از انتخاب مدل صورت می گیرد ولی طراحی منطقی به مدل خاصی بستگی ندارد.

۲. طراحی ادراکی مکمل طراحی منطقی است و پس از آن انجام می گیرد.

۳. طراحی منطقی به صورت کلی به سیستم می‌نگرد و با روشهایی مانند ER انجام می‌گیرد.

۴. طراحی منطقی به مدل خاصی مربوط می‌شود و پس از انتخاب مدل صورت می‌گیرد ولی طراحی ادراکی به مدل خاصی بستگی ندارد.

۵- فرق زبانهای پرس و جوی (Query) رویه‌ای (Procedural) و نارویه‌ای (Non-procedural) این است که:

۱. در زبانهای نارویه‌ای کاربر (user) چگونگی به عینیت (materialize) در آوردن دیدش را تصریح می‌کند و به طور ضمنی آنچه را که می‌خواهد، بیان می‌کند ولی در زبان های رویه‌ای بر عکس است.

۲. در زبان های نارویه‌ای کاربر فقط آنچه را که می‌خواهد تصریح می‌کند، در حالی که در زبان های رویه‌ای به علاوه باید چگونگی به عینیت در آوردن دیدش را نیز بیان نماید.

۳. اساساً با هم فرقی ندارند.

۴. در زبانهای رویه‌ای کاربر فقط آنچه را که می‌خواهد، تصریح می‌کند در حالیکه در زبانهای نارویه‌ای به علاوه باید چگونگی به عینیت در آوردن دیدش را نیز بیان نماید.

۶- کدامیک از موارد زیر از عناصر اصلی بانک اطلاعاتی است ؟ (مسابقات آموزشکده‌های فنی-۷۶)

۱. اشتراک منابع

۲. امنیت داده‌ها

۳. پروژه‌ها

۴. سخت‌افزار

۷- DBMS چیست ؟ (مسابقات آموزشکده‌های فنی-۷۶)

۱. رابطی است بین بانک اطلاعاتی و کاربر و نیز زبانی است برای تعریف بانک اطلاعاتی
۲. نرم افزار مدیریت و توصیف بانک اطلاعاتی و حفاظت و بازیابی داده هاست.
۳. همان بانک اطلاعاتی است.
۴. یک برنامه کاربردی است که در زبان اطلاعاتی داده ها را تعریف می کند.
- ۸- عناصر تشکیل دهنده محیط بانک اطلاعاتی کدام است ؟ (مسابقات آموزشکده های فنی-۷۷)
۱. موجودیت- صفت خاصه- داده- فایل
۲. سخت افزار- نرم افزار- فایل- داده
۳. نرم افزار- کاربر- داده- فایل
۴. سخت افزار- نرم افزار- کاربر- داده
- ۹- کدام یک از ویژگیهای یک بانک اطلاعاتی نیست ؟ (مسابقات آموزشکده های فنی)
- ۱- کنترل متمرکز
- ۲- کاهش میزان افزونگی
- ۳- پدیده ناسازگاری داده ها
- ۴- استفاده اشتراکی و همزمان داده ها
- ۱۰- DML چیست ؟ (مسابقات آموزشکده های فنی-۷۷)
- ۱- یک نوع بانک اطلاعاتی خاص است
- ۲- وظیفه تعریف فیلد ها را در بانک اطلاعاتی دارد
- ۳- دستورهایی است برای تعریف داده ها
- ۴- عهده دار اعمال هر گونه تغییر در بانک اطلاعاتی است
- ۱۱- کدامیک از موارد زیر در یک سیستم بانک اطلاعاتی جزو الزامات کارآیی است ؟ (مسابقات آموزشکده های فنی-۷۷)
- ۱- گزارش از کارکنان یک بخش سازمان

- ۲- ثبت نام بخش های یک سازمان
- ۳- محدودیت تعداد کارمندان یک بخش
- ۴- ثبت و گزارش مقدار ساعات اضافه کاری هر کارمند
- ۱۲- کدام تعریف برای DBMS کاملتر است ؟ (مسابقات آموزشکده های فنی ۷۷ و ۸۰)
- ۱- نرم افزار سیستم مدیریت بانک اطلاعاتی جهت کار و انجام عملیات روی داده ها
- ۲- نرم افزاری برای ارتباط بین کاربران و فایلها در بانک اطلاعاتی
- ۳- سیستم عامل مخصوص بانکهای اطلاعاتی
- ۴- نرم افزاری برای مدیریت داده ها در بانک اطلاعاتی
- ۱۳- کدامیک تعریف دیکشنری داده ها (catalog) در بانک اطلاعاتی می باشد ؟ (مسابقات آموزشکده های فنی -۷۷ و ۸۰)
- ۱- یک بانک شامل مشخصات کاربران بانک اطلاعاتی
- ۲- یک بانک که مشخصات سایر بانکهای اطلاعاتی در آن وجود دارد
- ۳- یک بانک شامل مشخصات داده های بانک اطلاعاتی
- ۴- یک بانک شامل مشخصات صفات خاصه در یک بانک اطلاعاتی
- ۱۴- DDL چیست ؟ (مسابقات آموزشکده های فنی ۷۷)
- ۱- زبانی است برای نوشتن بانک اطلاعاتی
- ۲- زبانی است برای تعریف داده ها
- ۳- زبانی است برای کار در بانک اطلاعاتی
- ۴- زبانی است برای دستکاری داده ها
- ۱۵- کدامیک از موارد زیر از اجزای معماری سیستم بانک اطلاعاتی ارائه شده توسط ANSI نیست ؟ (مسابقات آموزشکده های فنی -۷۸)
- ۱- سیستم مدیریت بانک اطلاعات (DBMS)
- ۲- زبان فرعی داده ای (DSL)

۳- پیش کامپایلر (Precompiler)

۴- زبان میزبان (HL)

۱۶- در پایگاه داده ها، با کاهش افزونگی داده ها نتیجه حاصل کدام است ؟ (مسابقات آموزش شده های فنی -۷۸)

۱- استقلال داده ها

۲- امنیت داده ها (security)

۳- کاهش ناسازگاری داده ها (Inconsistency)

۴- اشتراک داده ها

۱۷- کدامیک از موارد زیر جزء وظایف DBMS محسوب نمی شود ؟ (مسابقات آموزش شده های فنی -۷۸)

۱- تبدیل احکام / شمای ادراکی به سطح داخلی و بالعکس

۲- تعیین و محاسبه طول بلاک، آدرس فیزیکی رکوردها و اندازه بافر

۳- شناسایی کاربر و تشخیص اجازه دسترسی آن به پایگاه داده (DB)

۴- بررسی درخواست کاربر از نظر صحت گرامری (Syntax) آن

۱۸- یک سیستم بانک اطلاعاتی شامل چه مواردی می باشد ؟ (مسابقات آموزش شده های فنی -۷۸)

۱- داده های ذخیره شده- کاربران- DBMS و سخت افزار

۲- داده های ذخیره شده- DBMS و DBA

۳- داده های ذخیره شده- کاربران- AM

۴- داده های ذخیره شده- کاربران- AM و سخت افزار

۱۹- integrity به عنوان یکی از امتیازات database به چه معناست ؟ (مسابقات آموزش شده های فنی -۷۹)

۱- کاهش افزونگی داده ها و پایگاه داده ها

۲- صحت داده ها و پردازش ها و پیروی از قواعد سیستم

- ۳- محافظت داده‌ها در برابر خطرات
- ۴- پرهیز از ناهمخوانی داده‌ها در پایگاه داده‌ها
- ۲۰- طراح بانک، از داده‌های ذخیره شده در بانک چگونه دیدی در بالاترین سطح انتزاعی دارد؟ (مسابقات آموزشکده‌های فنی -۸۰)
- ۱- ادراکی
- ۲- خارجی
- ۳- داخلی
- ۴- فیزیکی
- ۲۱- کدامیک از موارد زیر جزء ویژگی های بانک است؟ (مسابقات آموزشکده‌های فنی -۸۰)
- ۱- امکان استفاده از بازیابی استنتاجی
- ۲- امکان پردازش دستورات کاربر به زبان طبیعی
- ۳- استفاده همزمان و اشتراکی از داده‌ها
- ۴- پدیده ناسازگاری داده‌ها
- ۲۲- کدام گزینه از اجزاء معماری سیستم بانک اطلاعاتی در ANSI نمی‌باشد؟
- ۱- Mapping
- ۲- MSD
- ۳- HL
- ۴- DSL
- ۲۳- منظور از CASE در پایگاه داده‌ها چیست؟
- ۱- ابزار کمکی در طراحی و پیاده سازی بانک اطلاعاتی
- ۲- حالات مختلف پیاده سازی بانک اطلاعاتی
- ۳- یکی از ساختارهای رابطه‌ای، شبکه‌ای یا سلسله مراتبی
- ۴- حالت خاصی که ممکن است بانک به حالت آنومالی برسد
- ۲۴- چهار کنترلی که باید روی تمامی تراکشن های بانک اطلاعاتی اعمال گردد تا صحت و جامعیت آن تضمین شود، کدامند؟

- ۱- data dictionary -isolation-security atomicity-
- ۲- isolation -anomaly-integrity durability-
- ۳- atomicity -isolation-consistency durability-
- ۴- consistency -data dictionary-security anomaly-

۲۵- در کاتالوگ سیستم کدام گزینه وجود دارد ؟

- ۱- نام موجودیتها و ارتباطات بین آنها
- ۲- شمای ادراکی
- ۳- شماهای خارجی کاربران-ساختار فیزیکی بانک
- ۴- هر سه

۲۶- منظور از DML چیست ؟

- ۱- احکام کار با (پردازش) داده ها
- ۲- احکام کنترلی
- ۳- احکام تعریف داده ها
- ۴- هیچکدام

۲۷- تصویر ادراکی یعنی:

- ۱- دید DBA از کل بانک
- ۲- دید کاربر از کل بانک
- ۳- دید DBA از بخشی از بانک
- ۴- دید کاربر از بخشی از بانک

۲۸- کدام گزینه از معایب بانک اطلاعاتی نیست ؟ (آزمایشی - کاوشگران - ۸۰)

- ۱- در صورت عدم کنترل مناسب ممکن است جامعیت داده ها به خطر افتد.
- ۲- سربارهای کارایی ممکن است زیاد باشد.
- ۳- برنامه نویسی پیچیده می باشد.
- ۴- در صورت عدم کنترل مناسب ممکن است امنیت به خطر بیفتد.

۲۹- کدام گزینه در رابطه با تراکنش ها نادرست است ؟

- ۱- تراکنش ها اتمی هستند یعنی اینکه یا به طور کامل اجراء می شوند یا اصلاً اجراء نمی شوند حتی اگر سیستم در وسط کار خراب شود.
- ۲- تراکنش یک واحد منطقی از کار است و معمولاً شامل چندین عمل بانک اطلاعاتی است.
- ۳- عمل COMMIT پیام موفقیت انجام تراکنش را ارسال می کند.
- ۴- عمل ROLLBACK باعث می شود عملیات از اول دوباره تکرار شود.
- ۳۰- کدام گزینه از وظایف DDL است ؟
 - ۱- ساختار داده ها
 - ۲- تعریف فیلد ها
 - ۳- محل فایلها
 - ۴- همه گزینه ها
- ۳۱- مدیران داده (DA) وظیفه شان چیست ؟
 - ۱- کنترل database
 - ۲- کنترل DBMS و داده ها
 - ۳- اجرای database
 - ۴- استفاده از database
- ۳۲- کدام گزینه برای توضیح دیتابیس کاملتر است ؟
 - ۱- بیان کننده فایل های داده ای است
 - ۲- بیان کننده جداول و صفات خاصه است
 - ۳- یک نوع زبان پردازشی جهت مدیریت داده ای است
 - ۴- بیان کننده موجودیتها، صفات خاصه و ارتباط بین موجودیتها است
- ۳۳- کدام گزینه تعریف کاملتری از DBMS را ارائه می کند ؟
 - ۱- همان دیتابیس است.
 - ۲- رابط بین دیتابیس و کاربر است.

۳- جهت تعریف و کار با دیتابیس استفاده می شود.

۴- ۳ و ۲

۳۴- منظور از DBMS Environment کدام است ؟

۱- Procedure-People -Data -Software -Hardware

۲- User -Software -Hardware

۳- Database -Software - Hardware

۴- Hardware-Software -People -Data

۳۵- DML چیست ؟

۱- زبانی جهت پرس و جوی داده ها

۲- زبانی جهت گزارش گیری

۳- ۱ و ۲

۴- زبانی جهت تعریف داده ها

۳۶- DDL چیست ؟

۱- برای انجام اعمال محاسباتی استفاده می شود.

۲- برای درج داده ها و به روز رسانی آنها استفاده می گردد.

۳- صفات مشخصه و خصوصیات دیتابیس را تعریف می کند.

۴- همان DML می باشد.

۳۷- کدام گزینه نادرست است ؟

۱- یکی از مزایای DBMS ایجاد محدودیتهای امنیتی است.

۲- Database مجموعه ای از چند رکورد است.

۳- External view دید طراح بانک است.

۴- End user افرادی هستند که از دیتابیس استفاده می کنند.

۳۸- در مدل ANSI / SPARC کدام دید مربوط به داده ها و نگرشی بر محدودیتهای

سیستم است ؟

۱- External Level

۲- Conceptual level

Internal -۳

1,3 -۴

۳۹- کدام گزینه توسط سطح ادراکی ارائه می گردد ؟

۱- مشخص ساختن داده‌ها

۲- تعریف دیتابیس

۳- تعریف سطرهای جداول

۴- مشخص ساختن رسانه ذخیره سازی

۴۰- DBA کیست ؟

۱- برنامه نویس دیتابیس

۲- پیاده سازی تصمیمات اداره کننده داده‌ها (DA)

۳- اجراء کننده دیتابیس

۴- طراح دیتابیس

۴۱- کدام گزینه نادرست است ؟

۱- در استقلال فیزیکی داده‌ها اگر تغییری در ذخیره سازی داده‌ها انجام گیرد، برنامه‌های کاربردی هیچ تغییری نمی‌کنند.

۲- در استقلال منطقی داده‌ها تغییر تصویر ادراکی بانک از دید کاربران و برنامه‌های آنها مخفی می‌ماند.

۳- CASE یک ابزار کمکی است که در طراحی و پیاده سازی بانک مورد استفاده قرار می‌گیرد.

۴- ناسازگاری و نایمن بودن داده‌ها، جامعیت بانک را خدشه دار می‌کند ولی گسست پیوندهای سمانتیک برای جامعیت مشکلی ایجاد نمی‌کند.

۴۲- در پایگاه داده‌ها با کاهش افزونگی، نتیجه حاصل کدام است ؟

۱- استقلال داده‌ها

2- security

۳- کاهش Inconsistency

۴- اشتراک داده‌ها

۴۳- کدام گزینه نادرست است ؟

۱- یک سیستم بانک اطلاعاتی شامل موارد روبرو است: داده‌های ذخیره شده، کاربران، DBMS، سخت‌افزار

۲- DBMS زبانی است برای تعریف بانک اطلاعاتی

۳- DBMS نرم‌افزار سیستم مدیریت بانک اطلاعاتی جهت کار و انجام عملیات روی داده‌ها می‌باشد.

۴- DBMS میهمان سیستم عامل است.

۴۴- در کدامیک از انواع ارتباط بین DBMS با سایر عناصر نرم‌افزاری محیط، امکان اشتراکی

شدن داده‌ها برای برنامه‌ها وجود ندارد ؟

۱- یک DBMS برای هر برنامه کاربردی وجود داشته باشد.

۲- یک DBMS تحت کنترل سیستم عامل وجود داشته باشد.

۳- برنامه‌ها تحت کنترل DBMS اجراء شوند.

۴- هیچکدام.

۴۵- کدامیک از موارد زیر از وظایف DBMS نمی‌باشد ؟

۱- برقراری امنیت

۲- تامین امکان تعریف و ایجاد بانک

۳- بازیابی و بهنگام سازی بانک

۴- تامین امکان کنترل جامعیت بانک

۴۶- DSL چه نوع زبانی است ؟

۱- Procedural

۲- Declarative

۳- OOP

۴- Non-structured

۴۷- الگوی ANSI / SPARC چیست ؟

۱- رابطه بین موجودیتها را بیان می‌کند.

- ۲- اجتماعی از دیدگاههای مجزای کاربران است.
- ۳- اجتماعی از دیدگاههای انبارشی و اجتماع دیدگاه کاربران و دیدگاههای مجزای کاربران است.
- ۴- اجتماعی از دیدگاههای انبارشی و مجزای کاربران است.
- ۴۸- مجتمع بودن عبارت است از:
 - ۱- جمع پذیری نرم افزارهای بانک اطلاعاتی
 - ۲- وحدت چندین فایل اطلاعاتی مجزا با امکان تکرار تکنیکی اطلاعات
 - ۳- وحدت چندین فایل اطلاعاتی مجزا بودن هرگونه تکراری
 - ۴- وحدت چندین رکورد حذف شده
- ۴۹- schema یک بانک اطلاعاتی عبارت است از:
 - ۱- تشریح DBMS
 - ۲- تشریح دیتابیس برای DBMS
 - ۳- تشریح داده‌ها
 - ۴- معرفی کاربران
- ۵۰- منظور از View در مدل ANSI چیست ؟
 - ۱- داده‌های دیده شده در بانکهای اطلاعاتی
 - ۲- نمایی از اطلاعات مورد نیاز طراح
 - ۳- مشاهده داده‌ها به همان شکل که در هر سطح مشاهده می‌شود.
 - ۴- نمایی از داده‌های مورد نیاز در DBMS
- ۵۱- دید داخلی عبارت است از:
 - ۱- شمای مربوط به استفاده از سیستم مدیریت بانک اطلاعاتی
 - ۲- شمای نحوه ذخیره داده‌ها در بانک اطلاعاتی
 - ۳- شمای مربوط به کاربران
 - ۴- شمای مربوط به فایلها، سکتورها و شیارها

۵۲- کدام گزینه در رابطه با فرهنگ داده‌ها درست است ؟

۱- DBMS امکانات فرهنگ داده‌ها را فراهم می‌سازد و یک بانک اطلاعاتی برای سیستم است.

۲- DBMS امکانات فرهنگ داده‌ها را فراهم می‌سازد و یک بانک اطلاعاتی برای کاربران است.

۳- فرهنگ داده‌ها توضیحاتی در مورد جداول به کاربر می‌دهد.

۴- فرهنگ داده‌ها یکی از امکاناتی است که DBA در اختیار دیتابیس قرار می‌دهد.

۵۳- کدام گزینه نادرست است ؟

۱- در اکثر نرم‌افزارهای پایگاه‌داده‌ها برای ذخیره سازی اطلاعات یک entity از جدول استفاده می‌شود.

۲- END USER نگهدارندگان محیط DBMS می‌باشند.

۳- DBMS امکانات فرهنگ داده‌ها را فراهم ساخته و یک بانک اطلاعاتی برای سیستم می‌باشد.

۴- از مزایای پایگاه‌داده نسبت به فایل‌های معمولی می‌توان کنترل حساب شده مقدار افزونگی و اشتراک داده‌ها را نام برد.

۵۴- کدام گزینه درست است ؟

۱- مجتمع بودن یعنی وحدت چندین فایل اطلاعاتی مجزا بودن هر گونه تکراری

۲- کوچکترین واحد داده‌های مفهومی را ((داده‌های تجزیه پذیر)) گویند.

۳- دیتابیس مجموعه‌ای از رکوردهای تکراری است.

۴- محدودیت‌های جامعیتی، شمای هر رابطه و اطلاعات امنیتی توسط DDL تعیین می‌گردد.

۵۵- کدام گزینه درست است ؟

۱- سیستم مدیریت پایگاه اطلاعاتی از محل فیزیکی رکوردها بر روی رسانه ذخیره سازی خبر دارد.

۲- Pre-compiler در بانکهای اطلاعاتی دستورهای بانکی را به دستورهای زبان برنامه نویسی تبدیل می کند.

۳- کلیه اطلاعات سیستمی از دید خارجی، ادراکی و داخلی درون DBMS قرار دارد.

۴- کاتالوگ سیستم فقط با دستورات DML قابل تغییر است.

۵۶- کدام گزینه نادرست است ؟

۱. External view سطح مشاهدات مجزای کاربران را بیان می کند.

۲. host language یک زبان بانک اطلاعاتی است.

۳. نگاشت خارجی / مفهومی کلید استقلال منطقی داده ها است.

۴. در استقلال فیزیکی طراح در تغییرات سطح داخلی / ادراکی آزاد است.

۵۷- استقلال منطقی شامل کدام مورد نمی باشد ؟

۱. تعریف یک رابطه جدید در شما

۲. تبدیل یک رابطه به دو رابطه کوچکتر

۳. جایگزینی یک رابطه از شما

۴. ۲ و ۱

۵۸- خاصیت isolation در تراکنش ها به چند مفهوم می باشد ؟

۱. هر تراکنش اگر به تنهایی اجراء شود بانک را از حالتی صحیح به حالت صحیح دیگری منتقل می سازد.

۲. اثر تراکنش های هم روند روی یکدیگر چنان است که انگار هر کدام در انزوا انجام می شود.

۳. کنترل isolation توسط واحد recovery management صورت می گیرد.

۴. تراکنش ها ممکن است اثر مخرب بر روی هم داشته باشند.

۵۹- دو نوع پایان برای تراکنش ها عبارتند از:

۱. Abort , Commit

۲. Run , ready

۳. Abort , Retry

۴. Commit , Success

۶۰- همخوانی یا سازگاری (Consistency) برای تراکنش ها به چه معناست ؟

۱. هر تراکنش اگر به تنهایی اجراء شود بانک اطلاعاتی را از حالتی صحیح به حالت صحیح دیگر منتقل می کند.

۲. در تراکنش باید تمامی قوانین جامعیت بانک اطلاعاتی را رعایت کند.

۳. ۲۱

۴. هیچکدام

۶۱- خاصیت اتمی بودن برای تراکنش ها به چه مفهوم می باشد ؟

۱. تمام دستورات یک تراکنش یا باید اجراء شود و یا هیچکدام از آنها نباید اجراء شود.

۲. تراکنش می تواند وسط کار لغو گردد و نیازی به ترمیم داده ها در این حالت نیست.

۳. تراکنش باید کامل اجراء شود و هیچگاه نمی تواند وسط کار لغو گردد.

۴. تراکنش در کل بانک باید صورت گرفته و نیازی به لغو کردن آن نیست.

۶۲- کدام اعمال توسط واحد Recovery management صورت می گیرد ؟

۱. یکپارچگی

۲. پیاپی

۳. همخوانی

۴. ۲۱

۶۳- خاصیت پیاپی (durability) در تراکنشها به چه معناست ؟

۱. تراکنشهایی که به مرحله Commit رسیده اند به صورت اتفاقی حذف نمی شوند.

۲. تراکنشهایی که به مرحله Commit رسیده اند اثرشان ماندنی می باشد.

۳. تراکنشهایی که به مرحله Commit رسیده اند حتی در صورت زلزله یا آتش سوزی ناپیستی از بین بروند.

۴. هر سه گزینه.

۶۴- مشخصات اصلی یک تراکنش نسبت به یک برنامه معمولی (در محیط غیر بانکی) کدام است ؟

۱. تراکنش نمی تواند به تعویق بیافتد.
۲. تراکنش به DBMS وابسته نیست.
۳. هدف از کنترل تراکنش ها بالا بردن سرعت است.
۴. تراکنش توسط DBMS کنترل می شود.

۶۵- در محیط پایگاه داده کدام نرم افزار وجود ندارد ؟

۱. نرم افزار DBMS
 ۲. رویه های ذخیره شده
 ۳. نرم افزار سیستم عامل
 ۴. نرم افزار برنامه های کاربردی
- ۶۶- درون کاتالوگ سیستم چه اطلاعاتی ذخیره نمی گردد ؟

۱. لغت نامه داده ها
 ۲. مقادیر داده های درون جداول
 ۳. تعداد نسخه های هر فایل و ترتیب زمانی آنها
 ۴. حق دستیابی افراد به داده های مختلف
- ۶۷- در رابطه با دید داخلی کدام گزینه درست می باشد ؟
۱. این دید مبتنی بر یک یا بیش از یک ساختار فایل است.
 ۲. این دید در سطح فیزیکی فایلینگ مطرح می گردد.
 ۳. این دید به طراحی منطقی معروف است.

۴. هر سه گزینه

۶۸- کدام گزینه درست است ؟

۱. زبان DSL می تواند مستقل از HL نباشد.

۲. زبان DSL می تواند مستقل از HL باشد.
۳. زبان HL یکی از زبانهای برنامه نویسی متعارف است.
۴. هر سه گزینه
- ۶۹- گزینه صحیح کدام است ؟
۱. یک کاربر می تواند چند دید داشته باشد.
۲. چند کاربر می توانند یک دید مشترک داشته باشند.
۳. مجموعه دیدهای کاربران را سطح خارجی می نامند.
۴. هر سه گزینه
- ۷۰- چه نوع سخت افزاری در پایگاه داده ها وجود دارد ؟
۱. ذخیره سازی
۲. ارتباطی
۳. پردازشی
۴. هر سه گزینه

پاسخ تست های سری ۳

- ۱- (۱) در بانک اطلاعاتی اطمینان از صحت داده ها بسیار مهم است. کاهش افزونگی و دستیابی مشترک از مزایای بانک اطلاعاتی است.
- ۲- (۱) Data Dictionary توسط خود DBMS ساخته و به روز در می آید. (به صورت خودکار)
- ۳- (۱)
- ۴- (۱) در طراحی منطقی مدل مورد استفاده (رابطه ای- شبکه ای- سلسله مراتبی) مهم نیست و شمای کلی مثلا با نمودار ER ترسیم می شود. اما طراحی ادراکی بستگی به مدل مورد استفاده دارد.

۵- (۲) در زبانهای غیر رویه‌ای کاربر فقط می‌گوید چه می‌خواهد ولی شیوه و الگوریتم انجام آن کار را بیان نمی‌کند. در زبانهای رویه‌ای کاربر باید الگوریتم انجام کارش را نیز بیان کند.

۶- (۴) اشتراک منابع و امنیت داده‌ها از مزایای بانک اطلاعاتی هستند. عناصر اصلی بانک اطلاعاتی عبارتند از: نرم‌افزار- سخت‌افزار- داده‌ها و کاربر

۷- (۲)

۸- (۴)

۹- (۳) برعکس در پایگاه داده‌ها ناسازگاری داده‌ها نباید وجود داشته باشد.

۱۰- (۴) DML بخشی از زبان DSL است و به دستورات کار با داده‌ها

(Data Manipulation Language = DML) گفته می‌شود.

۱۱- (۳) برای آنکه کارایی را ثابت نگه داریم باید تعداد کارمندان را محدود کنیم.

۱۲- (۱)

۱۳- (۳) به کاتالوگ فراداده (Meta data) یا داده‌هایی در مورد داده‌ها (Data about data) نیز گفته می‌شود.

۱۴- (۲) DDL بخشی از زبان DSL است و به دستورات تعریف داده‌ها (Data Definition Language = DDL) گفته می‌شود.

۱۵- (۳) اجزاء معماری ANSI عبارت است از: دید داخلی- دید خارجی- دید ادراکی- تبدیلات بین سطوح- زبان فرعی داده‌ای (DSL)- زبان میزبان (HL) و سیستم مدیریت بانک اطلاعاتی (DBMS)

۱۶- (۳) با کاهش افزونگی داده‌ها، داده مشابه کمتری وجود دارد و لذا احتمال ناسازگاری این داده‌ها کاهش می‌یابد. مثلاً اگر آدرس فردی تغییر کند تمامی جداولی که آدرس آن شخص را دارند تغییر کند. بدیهی است هر چقدر تعداد این جداول بیشتر باشد تغییر سخت‌تر انجام می‌گیرد و احتمال ایجاد ناسازگاری بیشتر می‌شود.

۱۷- (۲)

۱۸- (۱) عناصر اصلی سیستم بانک اطلاعاتی عبارت است از: سخت افزار- نرم افزار-

داده ها و کاربران

۱۹- (۲)

۲۰- (۱)

۲۱- (۳)

۲۲- (۲)

۲۳- (۱) CASE به معنای Computer Aided Software Engineering می باشد.

۲۴- (۳)

۲۵- (۴)

۲۶- (۱)

۲۷- (۱)

۲۸- (۳) در بانک اطلاعاتی برنامه نویسی ساده تر است.

۲۹- (۴) عمل ROLLBACK عدم موفقیت اجرای تراکنش را گزارش می کند و به

مدیر تراکنش می گوید که اشکالی پیش آمده است. بانک اطلاعاتی ممکن است در

حالت ناسازگار باشد و تمام بهنگام سازی ها که توسط آن واحد کاری انجام شده است

باید لغو یا رد شود.

۳۰- (۴)

۳۱- (۲)

۳۲- (۴)

۳۳- (۴)

۳۴- (۱)

۳۵- (۳)

۳۶- (۳)

۳۷- (۳) دید طراح بانک Conceptual view می‌باشد. دید خارجی سطح مشاهدات مجزای استفاده کنندگان را بیان می‌کند.

۳۸- (۲)

۳۹- (۱)

۴۰- (۲)

۴۱- (۴) عدم وجود ارتباطات بین موجودیتهای (گسست پیوندهای سمانتیک) نیز باعث خدشه دار شدن جامعیت بانک می‌شود.

۴۲- (۳) (ناسازگاری داده‌ها = Inconsistency) با کاهش افزونگی داده‌ها، داده‌ی مشابه کمتری وجود دارد و لذا احتمال ناسازگاری این داده‌ها کاهش می‌یابد.

۴۳- (۲)

۴۴- (۱)

۴۵- (۱) برقراری امنیت بر عهده DBA می‌باشد.

۴۶- (۲) در یک زبان بیانی (declarative یا Non-procedural) کاربر می‌گوید چه می‌خواهد ولی رویه انجام کار را بیان نمی‌کند. ولی مثلاً در C و پاسکال که رویه‌ای (procedural) هستند کاربر باید رویه و الگوریتم کار را نیز بیان کند.

۴۷- (۳)

۴۸- (۲)

۴۹- (۲)

۵۰- (۳)

۵۱- (۲)

۵۲- (۱)

۵۳- (۲) END USER افرادی هستند که از دیتابیس استفاده می‌کنند.

۵۴- (۴) گزینه ۱: مجتمع بودن یعنی وحدت چندین فایل اطلاعاتی مجزا با هر گروه تکراری

- گزینه ۲: کوچکترین واحد داده‌های مفهومی را مقادیر غیر قابل تجزیه گویند.
- گزینه ۳: دیتابیس مجموعه‌ای از چند رکورد است.
- ۵۵- (۳) گزینه ۱: DBMS از نوع فیلدها و ساختار فایلها خبر دارد ولی از محل فیزیکی رکوردها بر روی هارد دیسک خبر ندارد.
- گزینه ۲: Pre-compiler دستورهای زبان برنامه نویسی را به دستورات بانکی تبدیل می‌کند.
- گزینه ۴: کاتالوگ سیستم با برخی احکام DML و برخی احکام DDL قابل تغییر است.
- ۵۶- (۲) host language یک زبان برنامه نویسی است که امکان استفاده و ارتباط با بانک اطلاعاتی را فراهم می‌سازد.
- ۵۷- (۳)
- ۵۸- (۲) گزینه ۱ خاصیت consistency است. کنترل isolation توسط واحد concurrency control انجام می‌گیرد. بر طبق خاصیت انزوا همروندی تراکنش ها باید کنترل شود تا اثر مخرب بر روی هم نداشته باشند.
- ۵۹- (۱)
- ۶۰- (۳)
- ۶۱- (۱)
- ۶۲- (۴)
- ۶۳- (۴)
- ۶۴- (۴)
- ۶۵- (۳)
- ۶۶- (۲)
- ۶۷- (۱) این دید در سطح منطقی فایلینگ مطرح شده و به طراحی فیزیکی معروف است.
- ۶۸- (۴)

۶۹- (۴)

۷۰- (۴)

تست های سری ۴

- ۱- در کدام مدل پایگاه داده ها، مجموعه ای مرتب از درختها داریم که در آن فرزندی بدون پدر وجود ندارد ؟ (مسابقات آموزشکده های فنی-۸۰)

۱. رابطه‌ای
 ۲. سلسله مراتبی
 ۳. شبکه‌ای
 ۴. لیست معکوس
- ۲- کدامیک از موارد زیر، جزو ویژگیهای ساختار داده‌ای مدل شبکه‌ای است ؟
(مسابقات آموزشکده‌های فنی-۷۹)
۱. اصل وحدت عملگر، در یک عمل واحد رعایت می‌شود.
 ۲. از دید کاربر دارای وضوح است.
 ۳. برای محیط‌های دارای ارتباطات یک به چند دوسویه، مدل مناسبی نمی‌باشد.
 ۴. در عملیات ذخیره سازی آنومالی ندارد.
- ۳- کدامیک از موارد زیر جزء مدل‌های پایگاه اطلاعاتی نیستند ؟ (مسابقات آموزشکده‌های فنی و حرفه‌ای- ۷۸)
۱. مدل رابطه‌ای
 ۲. مدل سلسله مراتبی
 ۳. مدل اطلاعاتی
 ۴. مدل شبکه‌ای
- ۴- شکل زیر از کدام مدل برگرفته شده است ؟ (مسابقات آموزشکده‌های فنی و حرفه‌ای- ۷۷)
- ۱- رابطه‌ای
 - ۲- سلسله مراتبی
 - ۳- شبکه‌ای
 - ۴- بانکی
- ۵- کدام گزینه نادرست است ؟ (مسابقات آموزشکده‌های فنی-۷۷)

۱- در مدل رابطه‌ای بانک داده‌ها از دید کاربر از تعدادی جدول تشکیل شده است.

۲- در مدل سلسله مراتبی مسیر منطقی همیشه از ریشه و از بالا به پایین است.

۳- در مدل شبکه‌ای هر گره فرزند می‌تواند دارای بیش از یک پدر باشد.

۴- در مدل شبکه‌ای از مفاهیم ریاضی مجموعه‌ها برای نمایش داده‌ها و ارتباط آنها استفاده می‌شود.

۶- کدامیک از موسسات زیر روی استانداردسازی زبانهای پایگاه‌داده‌ها کار نکرده‌اند ؟
(کارشناسی ارشد- آزاد ۷۲)

۱- CODASYL DBTG

۲- ISO

۳- ANSI/SPARC

۴- 2,3

۷- کدامیک از موارد زیر بیانگر فرق بین یک مدل شبکه‌ای داده (Network) و یک مدل Entity

Relationship(ER) می‌باشد ؟ (کارشناسی ارشد- آزاد ۷۲)

۱- در مدل شبکه‌ای هر Entity فقط یک مولد (parent) دارد در مدل ER هر Entity می‌تواند بیشتر از یک مولد داشته باشد.

۲- در مدل ER روابط M:M پشتیبانی می‌شود در مدل شبکه‌ای خیر.

۳- در مدل ER یک Entity می‌تواند از نوع ضعیف (weak) باشد در رابطه شبکه کلید Entity از نوع قوی (strong) می‌باشد.

۴- ۲ و ۳

۸- می‌خواهیم برای یک دانشکده یک بانک اطلاعاتی آموزشی طراحی کنیم. از بین موجودیتهای اساسی، موجودیت درس را در نظر می‌گیریم و ارتباط زیر را:
در مدل شبکه‌ای برای نشان دادن ارتباط فوق در سطح ادراکی:
۱- حداقل دو مجموعه به معنایی که در CODASYL مطرح است، لازم است.

- ۲- در نظر گرفتن یک مجموعه به معنایی که در CODASYL مطرح است، کافی است.
- ۳- بیش از دو مجموعه به معنایی که در CODASYL مطرح است لازم نیست.
- ۴- در نظر گرفتن یک مجموعه به معنایی که در CODASYL مطرح است کافی است ولی بهتر است طراحی را با دو مجموعه انجام داد.
- ۹- کدام گزینه در رابطه با مدل سلسله مراتبی درست نمی باشد ؟
- ۱- هر گره پدر می تواند چندین فرزند داشته باشد ولی هر فرزند فقط یک پدر دارد.
- ۲- این مدل در عملیات حذف و بهنگام سازی آنومالی دارد.
- ۳- اگر پرس و جوئی را قرینه کنیم همواره رویه پاسخگویی نیز به همان صورت قرینه می شود.
- ۴- این مدل در عملیات درج دارای آنومالی است.
- ۱۰- کدام گزینه در رابطه با مدل شبکه ای درست نمی باشد ؟
- ۱- در این ساختار یک نمونه رکورد عضو می تواند عضو دو مالک متمایز باشد.
- ۲- عمل بازیابی ساده تر از مدل سلسله مراتبی است.
- ۳- این مدل در عملیات درج و حذف آنومالی ندارد.
- ۴- این مدل در عملیات بهنگام سازی آنومالی ندارد.
- ۱۱- کدام گزینه بیانگر تصویر ادراکی عام است ؟
- ۱- NIAM , EER
- ۲- شبکه ای- سلسله مراتبی- جدولی
- ۳- رابطه- جدول
- ۴- معماری ANSI
- ۱۲- بانک اطلاعاتی رابطه ای عبارت است از:
- ۱- بانکی است که در آن هر گونه تکرار همراه با جداول دیده می شود.
- ۲- بانکی است که در مقادیرش قابل تجزیه اند.

۳- بانک اطلاعاتی است که به عنوان مجموعه‌ای از روابط و جداول در نظر گرفته می‌شود.

۴- هیچکدام.

۱۳- کدام گزینه درست است ؟

۱- سیستم رابطه‌ای دارای ساختار فیزیکی است.

۲- سیستم رابطه‌ای دارای ساختار منطقی است.

۳- سیستم رابطه‌ای دارای ساختار منطقی و فیزیکی است.

۴- هیچکدام.

۱۴- کدام گزینه جزو مدل‌های داده‌ای است ؟

۱- Entity Relation Ship

۲- Record- based data Model

۳- Networking Model

۴- Semantic Model

۱۵- کدام عبارت نادرست است ؟

۱- برای مدلینگ ارتباطات یک به چند یکسویه بین انواع موجودیتها ساختار سلسله مراتبی مناسبتر است.

۲- در ساختار سلسله مراتبی، آنومالی در عملیات ذخیره سازی وجود دارد.

۳- زمان پاسخگویی به پرس و جوها در مدل رابطه‌ای بسیار کم است.

۴- ساختار سلسله مراتبی دارای یک رویه پاسخگویی واحد برای پرس و جوهایی قرینه نمی‌باشد.

۱۶- کدام گزینه نادرست است ؟

۱- اصل وحدت عملگر، در یک عمل واحد (مثلا درج) در مدل شبکه‌ای رعایت می‌شود.

۲- در مدل سلسله مراتبی مسیر منطقی همیشه از ریشه و از بالا به پایین است.

۳- در مدل رابطه‌ای بانک داده‌ها از دید کاربر از تعدادی جدول تشکیل شده است.

- ۴- در مدل شبکه‌ای هر فرزند می‌تواند دارای بیش از یک پدر باشد.
- ۱۷- کدام گزینه در رابطه با خصوصیات مدل سلسله مراتبی نادرست است ؟
 - ۱- خاص محیط‌هایی است که در آنها ارتباط‌های یک به چند یک سویه وجود دارد.
 - ۲- قدیمی‌ترین ساختار داده‌یی در سطح انتزاعی برای طراحی بانک است.
 - ۳- در عملیات ذخیره سازی آنومالی دارد.
 - ۴- برای پاسخگویی به پرس و جوهای قرینه رویه‌های پاسخگویی قرینه دارد.
- ۱۸- کدام گزینه در رابطه با خصوصیات مدل شبکه‌ای نادرست است ؟
 - ۱- عملگر بازیابی خاصیت تقارن دارد.
 - ۲- در عملیات ذخیره سازی آنومالی دارد.
 - ۳- از مفهوم ریاضی مجموعه ی کوداسیلی استفاده میکند.
 - ۴- برای محیط‌های دارای ارتباط یک به چند دوسویه مدل مناسبی است.
- ۱۹- دو مرحله از مراحل طراحی یک بانک اطلاعاتی عبارتند از طراحی ادراکی Conceptual Design و طراحی منطقی Logical Design این دو چه تفاوت اساسی با هم دارند ؟ (کارشناسی ارشد- دولتی ۷۵)
- ۱- طراحی ادراکی به مدل خاصی مربوط می‌شود و پس از انتخاب مدل صورت می‌گیرد ولی طراحی منطقی به مدل خاصی بستگی ندارد.
- ۲- طراحی ادراکی مکمل طراحی منطقی است و پس از آن انجام می‌گیرد.
- ۳- طراحی منطقی به صورت کلی به سیستم می‌نگرد و با روشهایی مانند ER انجام می‌گیرد.
- ۴- طراحی منطقی به مدل خاصی مربوط می‌شود و پس از انتخاب مدل صورت می‌گیرد ولی طراحی ادراکی به مدل خاصی بستگی ندارد.
- ۲۰- جهت پیاده سازی یک بانک اطلاعاتی کدام فعالیت در اولویت قرار دارد ؟ (کارشناسی ناپیوسته- آزاد ۸۰)

- ۱- طراحی مدل داده‌ها به صورت منطقی
 - ۲- طراحی مدل داده‌ها به صورت فیزیکی
 - ۳- طراحی مدل داده‌ها به صورت فیزیکی و منطقی
 - ۴- هیچکدام
- ۲۱- چرا مدل سلسله مراتبی آنومالی دارد ؟
- ۱- چون در هنگام اصلاح عمل بهنگام سازی باید منتشر شوند باشد.
 - ۲- چون هنگام درج باید رکورد پدر را داشته باشیم تا بتوانیم فرزند را درج کنیم.
 - ۳- چون در عمل حذف، ممکن است اطلاعات ناخواسته دیگری نیز حذف گردد.
 - ۴- هر سه گزینه

پاسخ تست های سری ۴

- ۱- (۲)
- ۲- (۴) مدل شبکه‌ای در عملیات درج، حذف و بهنگام سازی آنومالیهای مدل سلسله مراتبی را ندارد. این مدل از دید کاربر واضح نیست. مدل شبکه‌ای برای نمایش ارتباطات دوسویه مناسب می‌باشد.
- ۳- (۳)
- ۴- (۲) مدل سلسله مراتبی مشابه شکل یک درخت می‌باشد.
- ۵- (۴) مفاهیم مجموعه در مدل رابطه‌ای استفاده می‌گردد.
- ۶- (۲)
- ۷- (۴) ممکن است وجود یک پدیده وابسته به وجود پدیده‌ای دیگر باشد. مثلاً به محض حذف دانشجو از بانک دانشگاه (بر اثر فارغ التحصیلی یا اخراج شدن) وابستگان او نیز (مثل همسر و فرزند) از سیستم کمک هزینه باید حذف شوند. در این حال پدیده وابسته را موجودیت ضعیف (weak entity) می‌نامند.
- ۸- (۲)

- ۹- (۳) منظور از آنومالی (anomaly) یعنی وجود دشواری در انجام یک عمل خاص و یا عدم امکان انجام عمل و یا بروز عوارض نامطلوب در پی انجام عمل.
- ۱۰- (۲) عمل بازیابی در مدل شبکه‌ای پیچیده تر از مدل سلسله مراتبی می‌باشد.
- ۱۱- (۱)
- ۱۲- (۳)
- ۱۳- (۲)
- ۱۴- (۲)
- ۱۵- (۳)
- ۱۶- (۱) اصل وحدت عملگر در مدل شبکه‌ای رعایت نمی‌شود. مثلاً برای درج از دو عملگر connect و store می‌توان استفاده کرد. store برای درج نمونه رکورد و connect برای برقراری پیوند بین نمونه رکورد می‌باشد.
- ۱۷- (۴) مدل سلسله مراتبی برای پاسخگویی به پرس و جوهای قرینه، رویه‌های پاسخگویی قرینه ندارد.
- ۱۸- (۲) مدل شبکه‌ای در عملیات ذخیره سازی آنومالی ندارد.
- ۱۹- (۱) در طراحی منطقی مدل مورد استفاده (رابطه‌ای- شبکه‌ای- سلسله مراتبی) مهم نیست و شمای کلی مثلاً با نمودار ER ترسیم می‌شود. اما طراحی ادراکی بستگی به مدل مورد استفاده دارد.
- ۲۰- (۱)
- ۲۱- (۴)

تست های سری ۵

- ۱- کلید اصلی چیست ؟ (مسابقات آموزشکده‌های فنی - ۷۶)

۱. برای مرتب سازی اطلاعات بانک اطلاعاتی استفاده می شود.
 ۲. ترکیبی از اجزای داده ای است که مهم و کلیدی می باشد.
 ۳. جزء داده ای است که محل دقیق رکورد را مشخص می کند.
 ۴. کاربران بانک اطلاعاتی بوسیله آن با بانک ارتباط برقرار می کنند.
- ۲- کلید اصلی چیست ؟ (مسابقات آموزشکده های فنی - ۷۷)
۱. جزء داده ای است که محل رکورد مورد نظر ما را مشخص می کند.
 ۲. ترکیبی از چند جزء داده است.
 ۳. با آن می توان اطلاعات را مرتب سازی و sort نمود.
 ۴. رابطه ای است بین کاربر و بانک اطلاعاتی.
- ۳- با توجه به مفاهیم کلید در سیستم های رابطه ای کدام گزینه مصداق ندارد ؟ (مسابقات آموزشکده های فنی - ۷۸)
۱. کلید کاندید یکی از کلیدهای اصلی است که طراح آن را به عنوان کلید کاندید بر می گزیند.
 ۲. کلید اصلی یکی از کلید های کاندید است که طراح آن را به عنوان کلید اصلی بر می گزیند.
 ۳. کلید کاندید هر زیر مجموعه ای از عناصر رابطه است که یکتا بوده و کهننگی یا ایجاز (minimality) داشته باشد.
 ۴. کلید خارجی کلیدی است که در رابطه دیگر کلید اصلی باشد.
- ۴- درجه رابطه در بانک مدل رابطه ای چیست ؟ (مسابقات آموزشکده های فنی - ۷۸)
۱. به مقادیر یک صفت خاصه گفته می شود.
 ۲. به میدان مقادیر یک صفت خاصه گفته می شود.
 ۳. تعداد تاپلهای رابطه است.
 ۴. تعداد صفات خاصه موجودیت است.

۵- کدام گزینه در بانک رابطه‌ای صحیح نیست ؟ (مسابقات آموزشکده‌های فنی - ۷۹ و مشابه مسابقات ۸۰)

۱. تاپلها نظم دارند.
 ۲. تاپل تکراری در رابطه وجود ندارد.
 ۳. صفات خاصه نظم ندارند.
 ۴. همه صفات خاصه تجزیه ناپذیرند.
- ۶- کدام گزینه به مفهوم ((مجموعه مقادیر مجاز یک صفت)) می‌باشد ؟ (مسابقات آموزشکده‌های فنی - ۷۹)

۱. Integrity
۲. Persistent
۳. Tuple
۴. Domain

۷- در رابطه با جامعیت در مدل رابطه‌ای کدام گزینه نادرست است ؟

۱. جامعیت دامنه‌ای یعنی تمام صفات در تمامی رابطه‌ها از نوع دامنه خود باشند.
۲. جامعیت درون رابطه‌ای یعنی هر رابطه‌ای ارتباطش با رابطه دیگر صحیح باشد.
۳. جامعیت دامنه‌ای یعنی کلیدهای دارای مقدار تهی (NULL) یا تکراری نباشند.

۴. جامعیت ارجاع یعنی کلید خارجی درست تعریف شده باشد.

۸- کدام تعریف در رابطه با کلیدها نادرست است ؟

۱. Super key یعنی هر ترکیبی از صفت ها که خاصیت کلید داشته باشند ولی این ترکیب باید کمینه (minimal) باشد.
۲. Primary key یکی از کلید های کاندید است که توسط مدیر بانک انتخاب می‌شود.
۳. Alternative key کلیدهای کاندید به غیر از کلید اصلی می‌باشند.
۴. Foreign key صفتی است در یک رابطه که در رابطه دیگر کلید اصلی یا فرعی است.

۹- این جمله تعریف چیست ؟ ((زیر مجموعه‌ای از ضرب دکارتی چند دامنه))

۱. Domain

۲. Relation

۳. Tuple

۴. Join

۱۰- کدام گزینه نادرست است ؟

۱. قاعده جامعیت ارجاعی در رابطه با کلید خارجی است.

۲. قاعده جامعیت موجودیتی در رابطه با کلید اصلی است.

۳. Super key کلید اصلی است که یکتایی و minimality دارد.

۴. کلید کاندید غیر از کلید اصلی را Alternative می‌گویند.

۱۱- رابطه عبارت است از:

۱. پل ارتباطی برای فیلدها

۲. مفهوم ریاضی یک جدول

۳. جمع شدن چندین جدول در یک قالب بزرگتر

۴. هیچکدام

۱۲- کدامیک از روشهای زیر بهترین راه برای حصول اطمینان از آن است که کلیدهای

اصلی و خارجی از هماهنگی برخوردارند ؟ (کارشناسی ناپیوسته- آزاد ۸۰)

۱. از طریق اعمال پیش فرضهای (default values)

۲. از طریق Referential Integrity

۳. از طریق Entity

۴. هیچکدام

۱۳- کدام گزینه درباره رابطه نادرست است ؟

۱. کاردینالیتی رابطه تعداد صفات خاصه رابطه در یک لحظه از حیات آن است.

۲. در رابطه تابلها نظم ندارند.

۳. همه مقادیر صفات خاصه در رابطه تجزیه ناپذیر (atomic) می‌باشند.

۴. رابطه از دو مجموعه Heading و body تشکیل یافته است.

۱۴- کلید خارجی چیست ؟

۱. امکان پیوند دادن رابطه‌های بانک اطلاعاتی را می‌دهد.
۲. امکان رجوع از رابطه‌ای به رابطه دیگر را می‌دهد.
۳. امکان خارج شدن از رابطه‌ای را می‌دهد.
۴. ۱ و ۲

۱۵- کدام گزینه نادرست است ؟

۱. سیستم رابطه‌ای دارای ساختار منطقی است.
۲. جدول مشتق (drived table) رابطه‌ای است که از جدول دیگر اخذ گردیده است.
۳. در رابطه همه مقادیر صفات خاصه قابل تجزیه می‌باشند.
۴. بانک اطلاعاتی رابطه‌ای به عنوان مجموعه‌ای از روابط و جدولها در نظر گرفته می‌شود.

۱۶- چه رابطه‌ای بین کلید های اصلی و خارجی در یک پایگاه اطلاعاتی باید وجود داشته باشد ؟ (کارشناسی ناپیوسته- آزاد ۸۰)

۱. در زمانی که رابطه‌ای بین دو جدول به وجود می‌آید، جدول پدر دراین ارتباط به کلید خارجی بستگی دارد و جدول فرزند به کلید اصلی بستگی خواهد داشت.
 ۲. در زمانی که رابطه‌ای بین دو جدول به وجود می‌آید، جدول پدر دراین ارتباط به کلید اصلی بستگی دارد و جدول فرزند به کلید خارجی بستگی خواهد داشت.
 ۳. ستون داده‌های کلید اصلی در جدول شامل ستون کلیدهای خارجی می‌باشد.
 ۴. هیچ رابطه‌ای بین کلید اصلی و خارجی وجود ندارد.
- ۱۷- کدامیک از اقلام زیر بهترین تعریف به عنوان یک شخص، مکان چیز یا مفهوم از داده‌های جمع آوری شده‌اند ؟ (کارشناسی ناپیوسته- آزاد ۸۰)

۱. یک رابطه

۲. یک کلید اصلی

۳. یک جدول

۴. یک فیلد

۱۸- جهت پیاده سازی یک بانک اطلاعاتی کدام فعالیت در اولویت قرار دارد ؟

(کارشناسی ناپیوسته- آزاد ۸۰)

۱. طراحی مدل داده‌ها منطقی

۲. طراحی مدل داده‌ها فیزیکی

۳. طراحی مدل داده‌ها به صورت فیزیکی و منطقی

۴. هیچکدام

۱۹- کدامیک از جملات زیر مزیت و برتری یک ستون غیر قابل تجزیه در یک طراحی

نیست ؟ (کارشناسی ناپیوسته- آزاد ۸۰)

۱. بهتر می‌توان رابطه کلیدهای اصلی و خارجی را در آنها نگهداری نمود.

۲. بهتر می‌توان از یک ستون Query یا سوال نمود.

۳. بهتر می‌توان صحت اطلاعات (integrity) را بدست آورد.

۴. بهتر می‌توان ستون‌ها را به روز آورد.

۲۰- در اغلب نرم‌افزارهای پایگاه داده‌ها چه چیزی برای ذخیره سازی اطلاعات یک

Entity یا موجودیت به کار می‌رود ؟ (کارشناسی ناپیوسته- آزاد ۸۰)

۱. کلید اصلی

۲. فیلد

۳. جدول

۴. بانک اطلاعاتی

۲۱- زمانی که در حال مطالعه مجموعه‌ای از نیازهای بانک اطلاعاتی در حال طراحی هستیم بهترین روش برای مشخص نمودن موجودیت (Entity) ها کدام است ؟
(کارشناسی ناپیوسته - آزاد ۸۰)

۱. به دنبال افعال جملات باشیم و از آنها برای تعریف Entity استفاده کنیم.
 ۲. به دنبال اسامی و نامهای مختلف باشیم و از آنها برای تعریف Entity ها استفاده کنیم.
 ۳. به دنبال صفت ها در جملات باشیم و از آنها برای تعریف Entity ها استفاده کنیم.
 ۴. همه اقلام فوق درست است.
- ۲۲- کدام گزینه نادرست است ؟
۱. یک DBMS کاملاً رابطه‌ای (Fully relational) است اگر ساختار آن جدولی (tabolar) بوده و محدودیتهای جامعیتی را رعایت کند.
 ۲. در یک رابطه یک صفت غیر کلید، هیچ ارتباطی با کلید اصلی ندارد.
 ۳. منحصر به فرد بودن و غیر قابل کاهشی از ویژگیهای کلید کاندید است.
 ۴. اختصاص یک کلید نشانگر یک عامل محدود کننده در جهان واقعی پایگاه داده است.
- ۲۳- کدام گزینه نادرست است ؟
۱. رابطه دارای یک عنوان و یک بدنه است.
 ۲. رابطه دارای مجموعه‌ای از کلیدهای کاندید است.
 ۳. در مورد متغیرهای رابطه‌ای پایه (Base Relational) دقیقاً یکی از کلیدهای کاندید باید به عنوان کلید اصلی انتخاب شود.
 ۴. رابطه پایه‌ای بر حسب روابط پایه‌ای دیگر محاسبه می‌شود.
- ۲۴- هنگام تعریف یک رابطه پایه (Base relation) کدام گزینه درست است ؟
۱. حداقل یک کلید کاندید وجود دارد.

۲. می‌تواند کلید اصلی وجود نداشته باشد.
 ۳. کلید کاندید وجود ندارد.
 ۴. باید حتماً کلید خارجی وجود داشته باشد.
- ۲۵- دو رابطه به شرطی سازگارند که....
۱. دارای مجموعه‌ای از اسامی و صفات یکسان باشند.
 ۲. صفات خاصه متناظرشان بر روی دامنه یکسانی تعریف شده باشند.
 ۳. تعداد سطر و ستونهای یکسانی داشته باشند.
 ۴. ۱ و ۲
- ۲۶- کلید خارجی....
۱. در یک رابطه فیلد معمولی و در رابطه دیگر کلید اصلی است.
 ۲. فیلدی مشترک بین دو یا چند رابطه که در پیوند رابطه استفاده می‌گردد.
 ۳. ۱ و ۲
 ۴. همان کلید فرعی است.
- ۲۷- کلید جدول زیر کدام است ؟
۱. (A,C)
 ۲. (B)
 ۳. (A)
 ۴. (B,C)
- ۲۸- کدام گزینه نادرست است ؟
۱. رابطه زیر مجموعه‌ای از ضرب کارتین چند دامنه است.
 ۲. دامنه‌های رابطه لزوماً متمایز نمی‌باشند.
 ۳. در طول حیات جدول، Extention ثابت و Interntion متغیر است.
 ۴. رابطه از Heading و Body تشکیل یافته است.

۲۹- ((تمام اطلاعات موجود در بانک اطلاعاتی فقط به یک روش نمایش داده می شوند، یعنی به صورت مقادیری در موقعیتهای ستونی از سطرهای جدول)). این جمله کدام قاعده است ؟

۱. قاعده جامعیت ارجاعی

۲. قاعده اطلاعات

۳. قاعده درون رابطه ای

۴. قاعده برون رابطه ای

پاسخ تست های سری ۵

۱- (۳)

۲- (۱)

۳- (۱) توجه کنید کلید اصلی یکی از کلیدهای کاندید است نه اینکه کلید کاندید یکی از کلیدهای اصلی باشد.

۴- (۴)

۵- (۱) ترتیب تاپلها (رکوردها) در جدول مهم نیست.

۶- (۴)

۷- (۲) جامعیت درون رابطه ای یعنی هر رابطه ای به تنهایی صحیح می باشد. مثلاً عضو تکراری نداشته باشد و کلیدهایش درست باشند. جامعیت دامنه ای هر دو گزینه ۱ و ۳ را در بر می گیرد.

۸- (۱) Super key هر ترکیبی از صفتهاست که خاصیت کلید داشته باشد. این تنها نوع کلید است که کمینه نیست یعنی زیر مجموعه ای از آن هم ممکن است کلید باشد.

۹- (۲)

۱۰- (۳) ابر کلید یا Super key به هر کلیدی گفته می‌شود که تنها ویژگی یکتایی را داشته باشد. کلید کاندید حالت خاصی از ابر کلید و همچنین کلید اصلی حالت خاصی از کلید کاندید است.

۱۱- (۲)

۱۲- (۲) (قاعده جامعیت ارجاعی = Referential Integrity rule)

۱۳- (۱) تعداد تاپلهای رابطه در یک لحظه از حیات آن کاردینالیتی رابطه نام دارد.

۱۴- (۴)

۱۵- (۳) در رابطه صفات خاصه غیر قابل تجزیه هستند.

۱۶- (۲)

۱۷- (۴) صورت تست در واقع تعریف موجودیت است و هر موجودیت را می‌توان با یک جدول (که همان رابطه است) نشان داده ولی چون رابطه همان جدول است پس احتمالاً منظور طراح گزینه ۴ بوده است.

۱۸- (۱)

۱۹- (۱) بدیهی است هنگامی که ستون مرکبی داشته باشیم. مثل ستون آدرس که از ستون های شهر - خیابان - کوچه و پلاک تشکیل شده باشد به روز در آوردن ستونهای آن مشکلتر از وقتی است که ستونها ساده و غیر مرکب باشند.

۲۰- (۳) موجودیت فرد یا شیئی یا چیزی است که در مورد آن می‌خواهیم اطلاعاتی داشته باشیم، اطلاعات هر موجودیت را می‌توان در یک جدول خاصی ذخیره کرد.

۲۱- (۲) مثلاً در جمله ((دانشجوی مشروطی دارای معدل کمتر از ۱۲ می‌باشد))

دانشجو موجودیت است و برای آن باید یک جدول داشته باشیم که معدل یکی از فیلدهای آن باشد. مشروط بودن دانشجو نیز یک فیلد آن جدول است.

۲۲- (۲) یک صفت غیر کلیدی با کلید اصلی ارتباط دارد.

۲۳- (۴) رابطه پایه‌ای بر حسب روابط پایه‌ای دیگر محاسبه نمی‌شود.

۲۴- (۱)

۲۵- (۴)

۲۶- (۳)

۲۷- (۱) کلید باید یکتا باشد. زیر ستون B دو تا K وجود دارد. زیر ستون A دو تا t وجود دارد. زیر جفت ستون (B,C) دو تا (K,S) وجود دارد.

۲۸- (۳) به بسط یک جدول در هر لحظه Extention و به نام و مجموعه صفات رابطه چکیده رابطه یا Intention گفته می شود یعنی اسم جدول به همراه heading برابر intention است. در طول حیات یک جدول Extention متغیر ولی Intention ثابت است.

۲۹- (۲)

تست های سری ۶

۱- پیوند طبیعی دو جدول زیر چند سطر و چند ستون دارد ؟ (مسابقات آموزشکده های فنی - ۷۹)

s

S#	sname	city
S1	فن آوران	تهران
S2	ایران قطعه	تبریز
S3	پولادین	تبریز

sp

S#	P#	تعداد Qty
S1	P1	300
S1	P2	200
S1	P3	400
S2	P1	300
S2	P2	400
S3	P2	200

۱. ۱۸ سطر و ۶ ستون

۲. ۱۸ سطر و ۹ ستون

۳. ۹ سطر و ۶ ستون

۴. ۶ سطر و ۵ ستون

۲- عبارت A JOIN B را در نظر بگیرید. اگر تمام صفات خاصه A و B یکسان باشند عبارت فوق معادل کدام عبارت زیر است ؟ (مسابقات آموزشکده های فنی - ۷۹)

۱. A UNION B

۲. A MINUS B

۳. A INTERSECT B

۴. A TIMES B

۳- دو جدول زیر را در نظر بگیرید سپس بگوئید در نتیجه الحاق آنها توسط دستور داده شده، کلید اصلی چه می شود ؟

DEPT			EMP			
DEPT #	DNAME	BUDGET	EMP#	ENAME	DEPT#	SALARY
D1	Marketing	10M	E1	Lopez	D1	40K
D2	Development	12M	E2	Cheng	D1	42K
D3	Research	5M	E3	Finzi	D2	30K
			E4	Saito	D2	35K

Join:

DEPT and EMP over DEPT#

DEPT# ۱.

EMP# ۲.

BUDGET ۳.

EMP# , DEPT# ۴.

۴- در جدول EMP در تست قبلی، کلید خارجی کدام است ؟

EMP# ۱.

SALARY ۲.

DEPT# ۳.

۴. هیچکدام. چون D1 و D2 تکرار شده‌اند، DEPT# نمی‌تواند کلید خارجی

باشد.

۵- تقاضای زیر در مورد بانک اطلاعاتی عرضه کنندگان و قطعات چه کاری انجام

می‌دهد؟

Result: = ((S join SP) where P# ' p2') { S# , city};

۱. شهر عرضه کننده‌ای را می‌دهد که قطعه p2 را ساخته است.

۲. شماره عرضه کننده و شهر عرضه کننده‌ای را که قطعه p2 را تولید می‌کند،

بازیابی می‌کند.

۳. شماره عرضه کننده‌ای را می‌دهد که قطعه p2 در همان شهر آن تولید شده

است.

۴. هیچکدام

تذکر: سه جدول تهیه کننده S (S#, Sname, City) و قطعه P (P#, Pname, Color, City) و محموله SP (S#, P#, Qty) را در نظر گرفته و به ۴ سوال زیر پاسخ دهید.

۶- کدام گزینه شماره تهیه کنندگان و قطعات تولیدی آنها را چاپ می کند به شرط آنکه قطعه تولیدی آنها بیشتر از ۲۰۰ عدد باشد ؟

۱. $\Pi s\#, p\# (\sigma Qty > 200 (SP))$

۲. $\Pi s\#, p\# (\sigma Qty > 200 (S \bowtie P))$

۳. $\sigma Qty > 200 (\Pi s\#, p\# (SP))$

۴. 1,3

۷- در رابطه با دو دستور زیر کدام گزینه درست است ؟

(الف) $\sigma p\# = p1 (S \bowtie SP)$ و (ب) $\Pi s\#, p\# (\sigma Qty > 200 (SP))$

۱. این دو دستور دقیقاً معادل یکدیگرند و هیچ ارجحیتی نسبت به هم ندارند.

۲. این دو دستور معادل نیستند.

۳. این دو دستور معادلند ولی ب کاراتر و بهینه تر از الف است.

۴. این دو دستور معادلند ولی الف کاراتر و بهینه تر از ب است.

۸- کدام گزینه اسامی تهیه کنندگان قطعه p2 را می دهد ؟

۱. $\Pi sname (\sigma p\# = p2 (S \bowtie P))$

۲. $\Pi sname (\sigma p\# = p2 (S \bowtie SP))$

۳. $\sigma p\# = p2 (\Pi sname (P \bowtie SP))$

۴. $\sigma p\# = p2 (\Pi sname (P \bowtie S))$

۹- دستور زیر چه می کند ؟

$\Pi sname (\Pi s\# ((\Pi p\# (\sigma color = 'red' (P)) \bowtie SP) \bowtie S)$

۱. شماره اسامی تهیه کنندگانی را می دهد که قطعات قرمز رنگ را تولید کرده اند.

۲. شماره و اسامی تهیه کنندگانی را می دهد که شماره قطعه آنها موجود بوده و

رنگ آنها قرمز می باشد.

۳. اسامی تهیه کنندگانی را می دهد که یک قطعه قرمز رنگ تهیه می کنند.

۴. اسامی تهیه کنندگانی را می یابد که حداقل یک قطعه قرمز رنگ تهیه می کنند.
۱۰- کدام عملگر ها را می توان بوسیله عملگر های دیگر در جبر رابطه ای شبیه سازی کرد؟

۱. تقسیم - اجتماع - پیوند
 ۲. اشتراک - ضرب دکارتی - جایگزینی
 ۳. اشتراک - تقسیم - پیوند طبیعی
 ۴. ضرب دکارتی - تقسیم - اجتماع
- ۱۱- کدام فرمول در جبر رابطه ای نادرست است ؟ (σ نماد عملگر گزینش یا select می باشد)

۱. $A \times B = B \times A$
 ۲. $\sigma_p(A \cap B) = \sigma_p(A) \cap \sigma_p(B)$
 ۳. $A \cap B = B - (B - A)$
 ۴. $\sigma_p(B - A) = \sigma_p(b) - \sigma_p(a)$
- ۱۲- اگر A و B دو رابطه باشند، کلید اصلی A MINUS B چه می شود ؟

۱. کلید اصلی A
 ۲. کلید اصلی B
 ۳. کلید اصلی $A \cap B$
 ۴. کلید اصلی $A \cup B$
- ۱۳- اگر Divide by را به صورت سه عملوندی در نظر بگیریم، کدام فرمول در جبر رابطه ای صحیح است ؟

۱. $(A \text{ Divide by } B) \text{ TIMES } B \geq A$
۲. $(A \text{ Divide by } B) \text{ TIMES } B \neq A$
۳. $(A \text{ Divide by } B) \text{ TIMES } B \leq A$
۴. $(A \text{ Divide by } B) \text{ TIMES } B = A$

۱۴- رابطه دارای مجموعه‌ای از صفات (عنوان) و مجموعه‌ای از چند تایی‌ها (پیکر) است. رابطه DEE مجموعه‌ای خالی از چند تایی‌هاست. رابطه DUM نیز مجموعه‌ای خالی از صفات می‌باشد. حال کدام گزینه درست است؟

۱. عضو خنثی در حساب معمولی عدد یک است ($n * 1 = 1 * n = n$). در جبر

رابطه‌ای معادل عدد ۱ رابطه DEE است و داریم $R \text{ TIMES } DEE = DEE$

$TIMES R = R$

۲. عضو صفر در حساب دارای خاصیت $n * 0 = 0 * n = 0$ است در جبر رابطه‌ای

معادل عدد صفر رابطه DUM می‌باشد.

۳. هر دو گزینه

۴. هیچکدام

۱۵- کدام عملگر در جبر رابطه‌ای پایه‌ای نمی‌باشد و به کمک سایر عملگرها می‌توان آن را شبیه سازی کرد؟

۱. پیوند طبیعی (Join یا ∞)

۲. ضرب دکارتی ($\times$ یا TIMES)

۳. تفاضل (MINUS یا -)

۴. ۱ و ۳

۱۶- در جدول زیر

DEPT

D#	D name	Budget
D1	M	10
D2	D	20
D3	R	5

نتیجه دستور زیر چیست؟

Project:

DEPT over D# , Budget

۱. ۲ صفت خاصه و هیچ تاپل

۲. هیچ تاپل و ۲ صفت خاصه

۳. ۲ صفت خاصه و ۳ تاپل

۴. ۲ تاپل و ۳ صفت خاصه

۱۷- کدام عملگر در جبر رابطه‌ای وجود دارد، ولی ممکن است در بعضی از بانکهای

اطلاعاتی پیاده سازی نشده باشد ؟

۱. Join

۲. Subtract

۳. Add

۴. هر سه گزینه

۱۸- فرض کنید جدول دانشکده به صورت زیر شامل ((شماره، نام دانشکده و شهر))

باشد:

Clg (clg# , clgname , city)

و جدول درس شامل ((شماره درس، نام درس، تعداد واحد و شماره دانشکده ارائه

دهند)) باشد:

Crs (c# , c name , unit , clg#)

حال به کمک دستورات جبر رابطه‌ای، درسهایی که توسط همه دانشکده‌ها ارائه

می‌شوند را بدهید.

۱. Clg Join Crs giving temp ; select temp where cname ALL

۲. Project clg [clg#] Giving temp ; Crs Divideby temp

۳. Clg join crs giving temp ; select cname

۴. Project clg [clg#] Giving temp ; clg join crs join temp divideby
cname

۱۹- حاصل عبارت زیر کدام است ؟

R1(A , B , C)			DIV	R2 (C)
a1	b1	c1		C1
a1	b1	c3		C3
a2	b2	c2		
a3	b1	c1		
a3	b1	c3		

۱.

R3(A , B)	
a1	b1
a3	b1

۲.

R3(A , B)	
a2	b2

۳.

R3(A , B , C)		
a1	b1	c1
a3	b1	c1
a1	b1	c3
a3	b1	c3

۴. هیچکدام

۲۰- حاصل عبارت زیر کدام گزینه است ؟

R1(A , B , C)			DIV	R2 (B , C)	
a1	b1	c1		b1	c1
a1	b2	c2		b2	c2
a1	b2	c3		b3	c3
a2	b1	c1			
a2	b2	c2			

۱.

R3(A , B , C)		
a2	b1	c1
a2	b2	c2

۲.

R3 (A)
a2

۳.

R3(A)
a1

۴.

R3(A, B)	
b1	c1
b2	c2

21- شماره قطعاتی را بدهید که توسط همه تهیه کنندگان تهیه شده‌اند.

$$\begin{aligned} \text{Temp} &\leftarrow \Pi s\# \\ sp &\div \text{Temp} \end{aligned}$$

$$\begin{aligned} \text{Temp} &\leftarrow \Pi s\#(S) \\ \Pi p\# (sp \div \text{Temp}) \end{aligned}$$

$$\begin{aligned} & \text{.۳} \\ \Pi p\# (sp \infty S) \end{aligned}$$

۴. هر سه گزینه

۲۲- اگر DIVIDEBY همان عملگر معرفی شده توسط Codd (یعنی دو عملوندی)

باشد کدام رابطه صحیح است ؟

$$(A \text{ TIMES } B) \text{ DIVIDEBY } B = A \quad \text{.۱}$$

$$(A \text{ DIVIDEBY } B) \text{ TIMES } B \leq A \quad \text{.۲}$$

$$(A \text{ TIMES } B) \text{ DIVIDEBY } B \neq A \quad \text{.۳}$$

$$\text{.۴} \quad ۲ \text{ و } ۱$$

۲۳- با توجه به بسط جداول داده شده در این فصل، خروجی دستور $S \infty SP \infty P$

چه می‌شود؟

$$\left\{ \begin{array}{llllll} \text{.۱} & \text{آهن} & \text{قرمز} & ۳۰۰ & P1 & \text{تهران} & \text{فن آوران} & S1 \\ & \text{مس} & \text{سبز} & ۴۰۰ & P2 & \text{تبریز} & \text{ایران قطعه} & S2 \end{array} \right.$$

$$\left\{ \begin{array}{llllll} \text{.۲} & \text{آهن} & \text{قرمز} & ۳۰۰ & P1 & \text{تهران} & \text{فن آوران} & S1 \\ & \text{آهن} & \text{قرمز} & ۳۰۰ & P1 & \text{تبریز} & \text{ایران قطعه} & S2 \\ & \text{آهن} & \text{قرمز} & ۳۰۰ & P1 & \text{تهران} & \text{فن آوران} & S1 \end{array} \right.$$

۳. مس سبز ۲۰۰ P2 تبریز پولادین S3

۴. هیچکدام

۲۴- اگر رابطه A از درجه n باشد، چند تصویر مختلف از A وجود دارد؟

۱. $(n-1)!$

۲. $n!$

۳. 2^n

۴. $2n$

۲۵- خاصیت بسته بودن (closure) به چه معناست؟

۱. خروجی هر عملگر رابطه‌ای، یک رابطه است.

۲. الحاق دو رابطه رابطه‌ای جامع‌تر از هر دو می‌باشد.

۳. ضرب دکارتی دو رابطه مجموعه مادر هر دو می‌باشد.

۴. اجتماع دو رابطه در هر حال امکان پذیر است.

۲۶- کدام عملگرهای جبر رابطه‌ای به عملوندهای سازگار نیاز دارند؟

۱. U

۲. -

۳. ∞

۴. ۲ و ۱

۲۷- جبر رابطه‌ای....

۱. یک زبان اجرایی برای بانک اطلاعاتی رابطه‌ای است.

۲. همان SQL است.

۳. زبانی از نسل چهارم (4GL) می‌باشد.

۴. به DBMS می‌گویند چگونه یک رابطه جدید از یک یا چند رابطه ساخته شود.

۲۸- زبانی دارای اکمال رابطه‌ای است که....

۱. حداقل همتوان با جبر رابطه‌ای باشد.

۲. حداکثر همتوان با جبر رابطه‌ای باشد.

۳. هر رابطه‌ای که با عبارات جبر رابطه‌ای قابل تعریف است، توسط آن زبان نیز تعریف شدنی باشد.

۴. ۳ و ۱

۲۹- کدام فرمول نادرست است ؟

۱. $A \cap B = A - (A - B)$

۲. $A \cap B \neq B \cap A$

۳. $A \cap B = \sigma (A \cap b) \times A$

۴. $A \cap B = B \cap A$

۳۰- $X \cup Y$ دو جدول زیر چند سطر و چند ستون دارد ؟

Y

شهر تولد	معدل	شماره	نام
تهران	۱۷	۱۲۳	علی
تبریز	۱۵	۵۷۴	جواد

X

شهر تولد	معدل	شماره	نام
تهران	۱۷	۱۲۳	علی
تهران	۱۷	۹۳۴	مجید

۱. ۴ ستون و ۳ سطر

۲. ۳ ستون و ۴ سطر

۳. ۴ ستون و ۴ سطر

۴. ۳ ستون و ۳ سطر

۳۱- کدامیک از دستورات زیر کارا تر است ؟

الف $(SP \cap S) \text{ where } p\# = 'p4'$ [sname]

ب $(SP \text{ where } p\# = 'p4') \cap S$ [sname]

۱. الف

۲. ب

۳. کارایی هر دو یکسان است.

۴. اتفاقی است و نمی توان گفت.

۳۲- اگر A رابطه‌ای با صفات X , Y , B رابطه‌ای دیگر با صفت خاصه X باشد، کدام گزینه معادل عبارت $A \div B$ می‌باشد ؟

۱. $A [Y] - (A - [X])$

۲. $A [Y] - ((A [Y] \times B) - A) [Y]$

۳. $A [X] - ((A [X] \times B) - A) [X]$

۴. $A [X] - (A - B [X])$

۳۳- با در نظر گرفتن بانک اطلاعاتی تهیه کنندگان و قطعات، کدام دستور مشخصات تهیه کنندگانی را می‌دهد که تمام قطعات را عرضه کرده‌اند ؟

۱. $SP (S\#, P\#) \text{ divided by } P (P\#) \text{ Join } S$

۲. $(SP \text{ Join } S) \text{ Minus } P$

۳. $SP (S\#, P\#) \text{ divided by } P (P\#) \text{ Union } S$

۴. $(SP \text{ Join } P) \text{ Minus } S$

۳۴- با در نظر گرفتن بانک اطلاعاتی تهیه کنندگان و قطعات، کدام دستور مشخصات تهیه کنندگانی را می‌دهد که قطعه P3 را تهیه نمی‌کنند ؟

۱. $[\Pi S\# (S \in SP) \cup [\sigma_{P\# < 'P3'} (SP)]]$

۲. $[\Pi S\# (S) - \sigma_{P\# < 'P3'} (SP)] \in S$

۳. $[\Pi S\# (S) - (\Pi S\# (\sigma_{P\# = 'P3'} (SP)))] \in S$

۴. هیچکدام

۳۵- دو رابطه X و Y زیر را در نظر گرفته و بگوئید X times Y چند سطر خواهد داشت ؟

x
a
b
c
d
m

y
e
a
f
g

۱. ۲۰ سطر

۲. ۱۵ سطر

۳. ۹ سطر

۴. ۲۵ سطر

۳۶- پیوند طبیعی دو جدول زیر چند سطر و ستون خواهد داشت ؟

M	N
a	m
b	n
c	k

P	M	Q
P1	a	x
P2	b	y
P3	c	z
P4	d	u

۱. ۴ , ۴

۲. ۴ , ۵

۳. ۳ , ۳

۴. ۳ , ۴

۳۷- کدامیک از عملگرهای زیر در بانک اطلاعاتی بر روی جداول کار می کنند

؟(کارشناسی ناپیوسته- آزاد ۸۰)

۱. Join

۲. Add

۳. Subtract

۴. هیچکدام

۳۸- جدول سمت چپی با چه دستوری تبدیل به جدول سمت راستی می شود ؟

P#	Weight
P1	15
P2	13
P3	18

P#	Weight	GRM
P1	15	15000
P2	13	13000
P3	18	18000

۱. extend P Add (weight * 1000) AS GRM

۲. Add P (weight * 1000) AS GRM

۳. P Union weight * 1000 AS GRM

۴. هیچکدام

۳۹- در شکل زیر DEND مقسوم، MED میانجی و DOR مقسوم علیه است. حاصل

دستور DEND DIVIDEBY DOR PER MED کدام گزینه است ؟

DEND: _____ MED: _____ DOR: _____

S#
S1
S2
S3
S4
S5

S#	P#
S1	P1
S1	P2
S1	P3
S1	P4
S1	P5
S1	P6
S2	P1
S2	P2
S3	P2
S4	P2
S4	P4
S4	P5

P#
P2
P4

۱.

S#	P#
S1	P1
S2	P3

۲.

P#
P1
P3

۳.

S#
S1
S2

۴.

S#
S1
S4

۴۰- در تست قبلی اگر DOR $\frac{P\#}{P1}$ جدول باشد خروجی چه می شود ؟

۱.

S#
S1
S4

۲.

S#
S1
S2

۳.

P#
P1
P3

۴.

S#
S1
S2
S3
S4

۴۱- در جبر رابطه‌ای $A - (A - B)$ معادل کدام است ؟

۱. $A - (B - A)$

۲. $A \cap B$

۳. $B - (B - A)$

۴. 3,2

پاسخ تست های سری ۶

۱- (۴) پیوند طبیعی این دو جدول به صورت زیر می شود:

S#	P#	تعداد	Sname	city
S1	P1	300	فن آوران	تهران
S1	P2	200	فن آوران	تهران
S1	P3	400	فن آوران	تهران
S2	P1	300	ایران قطعه	تهران
S2	P2	400	ایران قطعه	تهران
S3	P2	200	پولادین	تهران

۲- (۳) منظور از INTERSECT همان اشتراک $\cap$ و منظور از TIMES همان ضرب دکارتی $\times$ می باشد.

۳- (۲) نتیجه الحاق دو جدول داده شده به صورت زیر است:

DEPT#	DNAME	BUDGET	EMP#	ENAME	SALARY
D1	Marketing	10M	E1	Lopez	40K
D1	Marketing	10M	E2	Cheng	42K
D2	Development	12M	E3	Finzi	30K
D2	Development	12M	E4	Saito	35K

با توجه به جدول فوق روشن است که ستون EMP# کلید اصلی است و چون داده های DEPT# تکرار شده نمی توان کلید باشد.

۴- (۳) در تعریف کلید خارجی یکتائی جزو شروط آن نیست. فقط باید آن ستون در جدول دیگری کلید اصلی باشد.

۵- (۲) نماد به کار رفته مربوط به زبان Tutorial D است که در کتاب دیت استفاده شده است. ستونهای خروجی در علامت آکولاد نوشته می شوند.

۶- (۱) توجه کنید که گزینه ۳ نادرست است زیرا هنگامی که عملگر Π ستونهای S# و p# را استخراج می کند دیگر ستون Qty وجود ندارد که عملگر $Qty > 200$ قابل اجرا باشد.

۷- (۳) در دستور الف ابتدا دو جدول s و sp پیوند خورده سپس سطرهایی از آن انتخاب می شود. در دستور ب ابتدا سطرهایی از جدول sp انتخاب شده سپس آن سطرها با جدول S پیوند می خورد بدیهی است که پیوند دو جدول کوچکتر زمان و فضای کمتری می خواهد.

۸- (۲) برای اسامی تهیه کنندگان قطعه p2 از جداول s و sp باید استفاده کنیم. لذا این دو جدول باید به همدیگر پیوند بخورند.

۹- (۴) از داخلی ترین پرائتز دستورات را اجراء کنید.

شماره قطعات قرمز رنگ را می دهد: $A \leftarrow \Pi_{p\#} (\sigma_{color='red'}(p))$

شماره تهیه کنندگان را می دهد که حداقل یک قطعه قرمز رنگ تولید کرده اند:

$$A \leftarrow \Pi_{S\#} (A \bowtie SP)$$

نام تهیه کنندگانی را می دهد که حداقل یک قطعه قرمز رنگ را تولید کرده اند:

$$\Pi_{Sname} (B \bowtie S)$$

۱۰- (۳)

۱۱- (۴)

$$\sigma_p(a-b) = \sigma_p(a) - \sigma_p(b)$$

ضرب دکارتی در ریاضیات توابع، خاصیت جابه جایی ندارد ولی در مورد جداول از آنجا که جابه جایی ستونها (فیلدها) مجاز است لذا ضرب دکارتی دو جدول خاصیت جابه جایی دارد.

۱۲- (۱) A-B یعنی سطرهایی که در A هست ولی در B نیست. پس A-B زیر مجموعه ای از A می شود و بدیهی است که کلید اصلی جدول A، کلید اصلی زیر مجموعه آن نیز خواهد بود.

۱۳- (۳)

۱۴- (۱) DUM مثل صفر نیست.

A TIMES DUM = DUM TIMES R = an empty relation with the same heading as R

پس گزینه ۲ اشتباه است.

۱۵- (۱)

۱۶- (۳) خروجی به شکل زیر است:

D#	Budget
D1	10
D2	20
D3	5

۱۷- (۲)

۱۸- (۲) عموماً وقتی در سوال، کلمات ((همه)) یا ((تمام)) دیده می‌شود باید از عملگر تقسیم استفاده کرد.

دستور Giving temp [clg#] project clg ابتدا شماره همه دانشکده‌ها را می‌دهد. سپس Crs divideby temp جدول Crs را بر شماره دانشکده‌ها تقسیم کرده و بدین ترتیب درسهایی که توسط همه دانشکده‌ها ارائه می‌شوند را می‌دهد.

۱۹- (۱)

۲۰- (۳)

۲۱- (۲)

۲۲- (۴) در حساب معمولی ضرب و تقسیم عکس یکدیگرند ولی در جبر رابطه‌ای TIMES و DIVIDEBY عکس یکدیگر نیستند چرا که اولاً تقسیم جدید سه عملوندی است، ثانیاً تقسیم A بر B و سپس تشکیل ضرب دکارتی از نتیجه حاصل و رابطه B منجر به رابطه‌ای می‌شود که ممکن است همانند A باشد، اما احتمالاً زیر مجموعه‌ای از A است (فرمول گزینه ۲). بنابراین عملگر DIVIDEBY در گزینه ۲ خیلی شبیه تقسیم صحیح در حساب معمولی است (یعنی از باقی مانده صرف‌نظر می‌کند)

۲۳- (۱) توجه کنید که هنگام پیوند $SP \infty S$ با P باید دو ستون city و p# هر دو یکسان باشند.

۲۴- (۳) این تعداد 2^n مشابه تعداد زیر مجموعه‌های یک مجموعه با n عضو است و شامل تصویر همانی (که نتیجه آن همانند رابطه اصلی A است) و تصویر تهی نیز می‌باشد.

۲۵- (۱)

۲۶- (۴) در اجتماع، اشتراک و تفاضل رابطه‌ها باید سازگار باشند، یعنی از نظر تعداد صفات خاصه و دامنه بایستی با یکدیگر سازگار باشند. اجتماع، اشتراک و تفاضل دو رابطه A و B سازگار است با عنوان A و B و بدنه‌ای شامل مجموعه تاپلهای A و B.

۲۷- (۴)

۲۸- (۴)

۲۹- (۳)

۳۰- (۱)

$X \cap Y$ یک سطر و ۴ ستون خواهد داشت.

$X - Y$ نیز یک سطر و ۴ ستون خواهد داشت.

XUY

شهر تولد	معدل	شماره	نام
تهران	۱۷	۱۲۳	علی
تهران	۱۷	۹۳۴	مجید
تبریز	۱۵	۵۷۴	جواد

تهران، ۱۷، ۱۲۳، علی $X \cap Y \rightarrow$

تهران، ۱۷، ۹۳۴، مجید $Y - X \rightarrow$

۳۱- (۲) ب کاراتر است چرا که گزینش را زودتر از پیوند انجام داده است.

۳۲- (۲)

۳۳- (۱) با دستور $P \#$ divided by $P \#$ ($S \#$ و $p \#$) شماره تهیه کنندگانی بدست می آید که تمام قطعات را عرضه می کنند. سپس هنگامی که نتیجه این دستور با S الحاق شود مشخصات این تهیه کنندگان بدست می آید.

۳۴- (۳)

۳۵- (۱) جدول X دارای ۵ سطر و جدول Y دارای ۴ سطر است پس X times Y دارای $4 \times 5 = 20$ سطر است.

۳۶- (۴) نتیجه پیوند طبیعی به شکل زیر است:

M	N	P	Q
a	m	P1	x
b	n	P2	y
c	k	P3	z

۳۷- (۱) در جبر رابطه‌ای عملگر Join یا پیوند را داریم. همچنین عملگرهای اشتراک و اجتماع و تفاضل (Minus یا Subtract) را نیز داریم. ولی عملگر Add یا جمع را نداریم. در اکثر نرم‌افزارهای بانک اطلاعاتی عملگر تفاضل پیاده سازی نشده است ولی عملگر پیوند در تمامی آنها وجود دارد.

۳۸- (۱) دستور extend برای اضافه کردن ستون به یک جدول استفاده می‌شود.

۳۹- (۴) یعنی کدام S# ها هم p2 و هم p4 را تهیه کرده‌اند.

۴۰- (۲)

۴۱- (۴)

تست های سری ۷

۱- اگر p و q یک و ف باشند معادل عبارت زیر کدام است ؟

IF p THEN q END IF

۱. P AND (NOT q)

۲. IF NOT p then NOT q

۳. (NOT p) OR q

۴. معادل ندارد.

۲- اگر ST و CO و STCO متغیرهای تاپلی رابطه‌های STT و COT و STCOT باشند

آنگاه عبارت زیر در حساب رابطه‌ای تاپلی چه می‌کند ؟

(ST.STID,ST.STDEG)WHERE EXISTS ST(ST.STID=STCO.STID AND STCO.COID='COM111');

۱. شماره دانشجویی و دوره تحصیلی آنهایی که درس COM111 را انتخاب کرده‌اند، می‌دهد.
۲. شماره دانشجویی و شماره درس افرادی را می‌دهد که درس COM111 را انتخاب کرده‌اند.
۳. سطح تحصیلی دانشجویانی را می‌دهد که درس COM111 را انتخاب کرده‌اند.
۴. این دستور غلط است.
- ۳- کدام گزینه در حساب رابطه‌ای تاپلی نادرست است ؟
 ۱. عبارت ST.STID مجموعه تمام شماره دانشجویان در رابطه STT را می‌دهد.
 ۲. عبارت ST عنوان جدول ST را می‌دهد.
 ۳. عبارت (STCO.STID,STCO.GRADE) شماره دانشجویان و نمره آنها از رابطه STCOT را می‌دهد.
 ۴. عبارت (CO.COID AS CONUM)WHERE COTYPE='P'; شماره درسهای عملی (عملی = 'P') از رابطه COT را می‌دهد.
- ۴- کدام دستور در حساب رابطه‌ای تاپلی عنوان درسهایی را می‌دهد که دانشجو با شماره ۱۲۳۴ انتخاب کرده است ؟
 ۱. COTITLE WHERE EXISTS STCO (STCO.COID=CO.COID AND STCO.STID='1234');
 ۲. COTITLE WHERE STCO.STID='1234';
 ۳. COTITLE WHERE STCO.COID=CO.COID AND STCO.STID='1234';
 ۴. هیچکدام
- ۵- کدام عبارت در حساب رابطه‌ای تاپلی شماره و نام درسهای دو واحدی را می‌دهد ؟
 ۱. CO.COID,CO.COTITLE WHERE EXISTS CO.CREDIT='2';
 ۲. CO.COID,CO.COTITLE WHERE CO.CREDIT='2';
 ۳. CO.CREDIT='2' FORALL WHERE CO.COID,CO.COTITLE;
 ۴. CO.COID,CO.COTITLE FORALLWHERE CO.CREDIT='2';

۶- عبارت زیر در حساب رابطه‌ای تاپلی نام دانشجویانی را می‌دهد که حداقل یک درس عملی را انتخاب کرده باشند. کدام گزینه فرم معادل پیشوندی (Prenex) آن است؟

ST.STNAME WHERE EXISTS STCO (ST.STID=STCO.STID AND EXISTS CO (CO.COID = STCO.COID AND CO.COTYPE='P'));

۱. ST.STNAME EXISTS WHERE (ST.STID=STID AND CO.COID =STCO.COID AND CO.COTYPE='P');

۲. ST.STNAME WHERE NOT FORALL STCO(ST.STID=STCO.STID AND CO(CO.COID=STCO.COID AND CO.COTYPE='P'));

۳. ST.STNAME WHERE EXISTS STCO (EXISTS CO (ST.STID=STCO.STID AND CO.COID=STCO.COID AND CO.COTYPE='P'));

۴. هیچکدام

۷- خروجی عبارت زیر در حساب رابطه‌ای تاپلی چیست ؟

ST.STNAME WHERE NOT EXISTS STCO (STCO.STID=ST.STID AND STCO.COID='COM333')

۱. اسامی دانشجویانی که درس شماره COM333 را انتخاب نکرده‌اند و شماره آنها در جدول STCOT نیست.

۲. اسامی و شماره دانشجویانی که درس شماره COM333 را انتخاب کرده‌اند.

۳. اسامی دانشجویانی را می‌دهد که درس شماره COM333 را انتخاب نکرده‌اند.

۴. تمامی اسامی دانشجویانی که درس شماره COM333 را انتخاب کرده‌اند.

۸- عبارت زیر در حساب رابطه‌ای تاپلی چه می‌کند ؟ (bs یعنی دوره کارشناسی)

CO.COTITLE WHERE FORALL ST (EXISTS STCO(STCO.STID=ST.STID AND STCO.COID=CO.COID AND ST.STDEG='bs'));

۱. این عبارت نادرست است و ایراد دارد.

۲. عنوان درسهایی را می‌دهد که تمام دانشجویان دوره کارشناسی که شماره‌های یکسان دارند آنها را انتخاب کرده باشند.

۳. شماره درس و شماره دانشجویانی را می‌دهد که تمام درسهای کارشناسی را گذرانده باشند.
۴. عنوان درسهایی را می‌دهد که تمام دانشجویان دوره کارشناسی آنها را انتخاب کرده باشند.

۹- اگر متغیر میدانی CO بر روی فیلد COID تعریف شده باشد، در حساب رابطه‌ای میدانی دستور زیر چه می‌کند ؟
 CO WHERE COT (COID: CO, CODEID: 'D111');

۱. شماره درسهای گروه آموزشی 'D111' را می‌دهد.
۲. شماره درسها و شماره گروه آموزشی درسهایی را می‌دهد که شماره آنها D111 باشد.
۳. گروه آموزشی را می‌دهد که شماره درس آن D111 باشد.
۴. تمام گروه‌هایی را می‌دهد که درس D111 را ارائه کرده‌اند.
- ۱۰- اگر متغیر میدانی CO بر روی فیلد COID و متغیر میدانی ST بر روی فیلد STID و متغیر CRED بر روی فیلد CREDIT تعریف شده باشند کدام گزینه در حساب رابطه‌ای میدانی نادرست است ؟

۱. عبارت CO مجموعه شماره همه درسها را نشان می‌دهد.
۲. عبارت ST WHERE STT (STID: ST) مجموعه شماره دانشجویان موجود در رابطه STT را نشان می‌دهد.
۳. دستور زیر شماره درسهای یک واحدی از گروه آموزشی D222 را می‌دهد:
 CO WHERE EXISTS CRED (CRED='1' AND COT (COID: CO, CREDIT: CRED, CODEID='D222'))
۴. هیچکدام

- ۱۱- دستور زیر در حساب رابطه‌ای میدانی چه می‌کند ؟
 STNAM WHERE EXISTS ST (STT(STID: ST, STNAME: STNAM) AND NOT STCOT (STID: ST, COID: 'COM333'))
۱. نام دانشجویانی را می‌دهد که درس COM333 را انتخاب نکرده‌اند.

۲. نام دانشجویانی را می‌دهد که از درس COM333 افتاده‌اند.
 ۳. مشخصات دانشجویانی را می‌دهد که حداقل درس COM333 را انتخاب کرده‌اند.
 ۴. نام دانشجویانی را می‌دهد که تمام درسهای هم گروه با COM333 را انتخاب نکرده‌اند.
- ۱۲- کدام گزینه نادرست است ؟
۱. قدرت جبر رابطه‌ای با حساب رابطه‌ای یکسان است.
 ۲. حساب رابطه‌ای با حساب رابطه‌ای تاپلی از قدرت محاسباتی مساوی برخوردارند.
 ۳. حساب رابطه‌ای مبتنی بر شاخه‌ای از ریاضیات به نام Predicate calculus می‌باشد.
 ۴. جبر رابطه‌ای توصیفی است در حالی که حساب رابطه‌ای دستوری است.

پاسخ تستهای سری ۷

- ۱- (۳) در ریاضیات می‌دانیم که $p \rightarrow q = \text{NOT } p \text{ OR } q$
- ۲- (۱)
- ۳- (۲) عبارت ST تمام رابطه STT را می‌دهد. (تمام تاپلهای آن را)
- ۴- (۱)
- ۵- (۲) توجه کنید که وقتی اطلاعات فقط در یک جدول است از سورهای EXISTS یا FORALL استفاده نمی‌شود.
- ۶- (۳)

۷- (۳)

۸- (۴)

۹- (۱)

۱۰- (۴)

۱۱- (۱)

۱۲- (۴) حساب رابطه‌ای توصیفی یا نارویه‌ای است ولی جبر رابطه‌ای دستوری یا رویه‌ای است.

تست های سری ۸

۱- SQL چیست ؟

۱. یک زبان قوی است برای ایجاد و دسترسی به بانک اطلاعاتی
 ۲. یک برنامه کاربردی است که از DBMS فرمان می‌گیرد.
 ۳. قسمتی از بانک اطلاعاتی است و مخصوص دستکاری داده‌ها می‌باشد.
 ۴. DDL و DML روی هم SQL را می‌سازند.
- ۲- کدام دستور SQL نمی‌تواند باعث به وجود آمدن ناسازگاری در داده‌ها شود ؟

- ۱. INSERT
- ۲. DELETE
- ۳. UPDATE
- ۴. SELECT

۳- در بانک اطلاعاتی ((تولید کننده- قطعه- پروژه)) دستور SQL زیر چه خروجی تولید می کند ؟

```
Select S# , P# , J#  
From S , P , J  
Where S.city=P.city AND P.city=J.city
```

۱. شماره تمام تهیه کنندگان- شماره قطعه و شماره پروژه هایی را می دهد که تهیه کننده قطعه و پروژه دو به دو از یک شهر نیستند.

۲. شماره تمام تهیه کنندگان- شماره قطعه و شماره پروژه هایی را می دهد که از یک شهر هستند.

۳. شماره تمام تهیه کنندگان- شماره قطعه و شماره پروژه هایی را می دهد که از یک شهر نیستند.

۴. شماره تمام تهیه کنندگان- شماره قطعه و شماره پروژه هایی را می دهد که تهیه کننده قطعه و پروژه دوه دو از یک شهر هستند.

۴- اگر جدول P دارای فیلدهای (P# , Pname , Color , Weight , city) ، جدول S دارای فیلدهای (S# , Sname , Status , City) و جدول SP دارای فیلدهای (S# , P# , City) باشند با زبان SQL به سوال زیر پاسخ دهید:

((تمام جفت شهرهایی (City) را بیابید که تهیه کننده (S) ساکن شهر اول، قطعه ای (P) انبار شده و شهر دوم را تهیه کرده باشد.))

۱.

```
Sallect city From S , P  
Where S.city = P.city
```

۲.

```
Sallect S.city , P.city From S , P  
Where S.city = P.city
```

۳.

```
Sallect distinct S. city,P.city From S,P, SP
```

Where S. S# = SP.S# AND SP.P# = P.P#

۴. هیچکدام

۵- منظور از SQL چیست ؟

۱. زبان پرس و جوی ساخت یافته است.

۲. زبان فرعی داده‌ای است.

۳. زبان داده‌ای مستقل است.

۴. همان زبان میزبان است.

۶- دستور مقابل نام چه افرادی را چاپ می‌کند ؟

Select name

From T

where name like "%رضا%"

(فرض کنید T جدولی است که یکی از ستون‌های آن نام (name) می‌باشد.)

۱. افرادی را چاپ می‌کند که ((رضا)) در نام آنها آمده است.

۲. افرادی را که اسم آنها ((رضا)) باشند چاپ می‌کند.

۳. افرادی را که اسم فامیل آنها ((رضا)) است فقط چاپ می‌کند.

۴. افرادی را که اسم یا اسم فامیل آنها ((رضا)) است چاپ می‌کند.

۷- با توجه به دو جدول S و T کدام دستور باعث نمایش خروجی زیر می‌شود ؟

S:Name	T: Name	Name	خروجی :
Able	Able	Able	
Bravo	Baker	Charlie	
Charlie	Charlie	Exitor	
Decon	Exitor	Goober	
Exitor	Falconer		
Goober	Fuber		
	Goobar		

۱. Select * From S union select * From T ;

۲. Select * From S Minus select * From T ;

۳. Select * From S,T where S.Name = T.name ;

۴. Select * From S Intersect select * From T ;

۸- با توجه به دو جدول S و T کدام دستور خروجی زیر را می دهد ؟

S:Name	T: Name	Name : خروجی
Able	Able	Bravo
Bravo	Baker	Decon
Charlie	Charlie	
Decon	Dean	
Exitor	Exitor	
Goober	Falconer	
	Fuber	
	Goobar	

۱. Select * From T Intersect select * From S ;

۲. Select * From T Minus select * From S ;

۳. Select * From S Minus select * From T ;

۴. Select * From T Union select * From S ;

۹- در بانک اطلاعاتی عرضه کنندگان و قطعات نتیجه نهایی دستور شبه SQL زیر

چيست؟

CREATE VIEW V AS

((S JOIN SP) where P# = 'P2') {S# , city} ;

Result: = (V where city = 'london') {s#} ;

۱. شماره عرضه کننده را برای عرضه کنندگان لندن که قطعه P2 را تولید

می کنند، می دهد.

۲. شماره عرضه کننده را برای عرضه کنندگان ساکن شهر لندن، می دهد.

۳. شماره عرضه کننده را برای عرضه کنندگان تولید کننده قطعه P2، می دهد.

۴. شهر و شماره عرضه کنندگانی را می دهد که قطعه انبار شده در شهر لندن را

تولید کرده اند.

۱۰- دستور روبرو در SQL چه می کند ؟

Select *

From SP

۱. تمام جدول SP را چاپ می کند.
 ۲. این دستور غلط است چون قسمت where ندارد.
 ۳. نام تمام فیلدهای جدول SP را چاپ می کند.
 ۴. کل اطلاعات ستون های اول و آخر جدول را چاپ می کند.
- ۱۱- دستور زیر چه می کند ؟

Select S.*, P. * from S, P

۱. تمام سطرها و ستونهای دو جدول S و P را زیر هم چاپ می کند.
 ۲. دو جدول S و P را به طور کامل کنار هم چاپ می کند.
 ۳. ضرب کارتیزین دو جدول S و P را می دهد.
 ۴. پیوند طبیعی دو جدول S و P را می دهد.
- ۱۲- دستور زیر چه می کند ؟

Select S# from S

Where city = (select city from S where S# = S1)

۱. شماره تهیه کنندگانی را می دهد که تهیه کننده S1 قطعات خود را در آن انبار کرده است.
 ۲. شماره تهیه کنندگانی را می دهد که در همان شهری ساکن باشند که S1 ساکن است.
 ۳. این طرز نوشتن اصلاً در SQL قابل قبول نیست.
 ۴. شهرهایی را می دهد که تهیه کننده S1 در آن جا شعبه دارد.
- ۱۳- با کدام دستور SQL زیر می توان طول فیلد رشته ای را از ۳۰ به ۴۰ افزایش داد ؟
۱. Alter table ... Modify...
 ۲. Alter table ... Add ...
 ۳. Alter table ... Replace...
 ۴. اینکار با این دستور امکان پذیر نمی باشد.
- ۱۴- اگر teacher نام جدول مشخصات اساتید و Sec جدول گروه درس (شامل فیلدهای شماره درس، شماره دانشجو، نام استاد و نمره) و نیز tname نام استاد باشد، آنگاه دستور SQL زیر چه می کند ؟

Select * from teacher where tname in
(select tname from sec where score = 10 And tname
not in (" بهتاش" و "امیری"));

۱. مشخصات اساتید امیری و بهتاش که نمره ۱۰ نداده‌اند را می‌دهد.
 ۲. مشخصات اساتیدی را می‌دهد که دانشجویان آنها از تمام دروس ۱۰ گرفته‌اند
(به جز بهتاش و امیری)
 ۳. مشخصات اساتیدی به جز امیری و بهتاش که نمره ۱۰ داده‌اند را نمایش
می‌دهد.
 ۴. مشخصات اساتیدی را می‌دهد که به دانشجویانی درس داده‌اند که نمره ۱۰
گرفته‌اند ولی جزو شاگردان بهتاش و امیری نمی‌باشند.
- ۱۵- کدام گزینه درست است ؟
۱. در SQL برای انجام یک عمل راه حل های متعددی وجود دارد و این یکی از
مهمترین مزایای SQL است.
 ۲. هرگاه پرس و جوئی را در SQL هم بتوان با پیوند جداول انجام داد و هم با
Select متداخل برای بالابردن کارایی بهتر است از Select متداخل استفاده
کنیم.
 ۳. در SQL تابع Count (*) سطرهای NULL را نمی‌شمارد.
 ۴. در دستور select به جای between می‌توان از AND و به جای IN می‌توان از
OR استفاده کرد.
- ۱۶- کدام گزینه نادرست است ؟
۱. SQL یک زبان مبدل است یعنی یک یه چند رابطه را به عنوان ورودی قبول
می‌کند و نتیجه عملیات روی آنها همواره یک رابطه است.
 ۲. جبر رابطه‌ای خاصیت ((بسته بودن)) دارد. یعنی عملوند ها رابطه است و
نتیجه نیز رابطه می‌باشد.

۳. به خاطر بسته بودن SQL و جبر رابطه‌ای می‌توان محاسبات تودرتو در هر دو داشت.

۴. SQL یک زبان procedural می‌باشد.

۱۷- کدامیک از زبان‌های نسل زیر، در جهت تقلید زبان طبیعی است ؟

۱. 3GL

۲. 4GL

۳. Access

۴. SGL

۱۸- برای ایجاد جدول پایه از چه دستوری در SQL استفاده می‌کنیم ؟

۱. create table

۲. create dbms

۳. struct table

۴. struct dbms

۱۹- دستور زیر در بانک اطلاعاتی تهیه کنندگان و قطعات چه می‌کند ؟

Select S#, status from S

Where city = "Tabriz"

Order by status desc

۱. شماره تهیه کنندگانی را می‌دهد که در شهر تبریز دارای وضعیت نزولی می‌باشند.

۲. شماره و وضعیت تهیه کنندگانی را می‌دهد که در شهر تبریز هستند، مرتب شده بر اساس وضعیت به صورت نزولی

۳. شماره و وضعیت تهیه کنندگانی را می‌دهد که در شهر تبریز هستند، مرتب شده بر اساس وضعیت به صورت صعودی

۴. وضعیت و شهر شرکتهائی را می‌دهد که در تبریز وضعیت نزولی دارند.

۲۰- دستور زیر در بانک اطلاعاتی تهیه کنندگان و قطعات چه می‌کند ؟

Update S Set status=3 * status

Where city = 'Tehran'

۱. تهیه کنندگانی که وضعیت سه برابر تولید کنندگان تهران دارند را حذف می‌کند.

۲. وضعیت تولیدکنندگان ساکن شهر تهران را سه برابر می‌کند.

۳. وضعیت تولیدکنندگان ساکن شهر تهران را برابر عدد ۳ می‌کند.
۴. شهرهایی که وضعیت سه برابر تهران دارند را به روز در می‌آورد.
- ۲۱- اگر S یک دید بر روی جدول پایه‌ای A باشد، برای آنکه هر گونه اصلاحی روی دید S مجاز باشد کدام محدودیت باید رعایت شود؟
۱. دید S باید تمام فیلدهای NULL جدول A را داشته باشد.
 ۲. دید S باید شامل کلید اصلی جدول A باشد.
 ۳. دید S باید شامل کلید فرعی جدول A باشد.
 ۴. هیچ شرط خاصی برای این موضوع لازم نمی‌باشد.
- ۲۲- اگر عبارتی را بخواهیم، که با ABC شروع می‌شوند از کدام گزینه استفاده می‌کنیم؟
۱. starting ABC
 ۲. like ABC%
 ۳. like ABC\$
 ۴. like ABC*
- ۲۳- دو جدول S (x, y, z, w) و S'(x, y, z', w') را در نظر گرفته و بگوئید پس از اجرای دستور زیر، دید V چه ستونهایی خواهد داشت؟
- Create View V AS
(Select x,y from S) union (select x,y from S')
۱. V (x, y, z, w, z', w')
 ۲. V (z', w)
 ۳. V (z, w, z', w')
 ۴. V (x, y)
- ۲۴- در بانک اطلاعاتی تهیه کنندگان و قطعات، کدام دستور شماره قطعه تمام قطعاتی را که توسط بیش از یک تولید کننده تهیه شده‌اند را می‌دهد؟
۱. select P# from SP group by S# having count (*)>1
 ۲. select P# from P,SP where count (*)>1
 ۳. select P# from SP group by P# having count (*)>1
 ۴. select P# from S,P,SP where count (*)>1
- ۲۵- جدول y دارای ستون های price و code می‌باشد. کدام دستور شماره کالائی که بالاترین قیمت را دارد، چاپ می‌کند؟
۱. select code from y where price = (select max(price)from y)
 ۲. select code from y where price = max(price)
 ۳. 2,1

۴. select code from y first , y second where price = max(price)
 ۲۶- در رابطه با دید تعریف شده زیر کدام گزینه درست می باشد ؟ (جدول پایه ای R دارای ستونهای x , y , z می باشد)

Create view K(x , y ,w)
 AS select x , y , sum (z)
 From R

۱. در دید K می توان عملیات ذخیره سازی را انجام داد ولی استقلال داده ای نقض می شود.

۲. چون select قسمت where ندارد عملیات ذخیره سازی عولرض نامطلوب خواهد داشت.

۳. در دید k نمی توان عملیات ذخیره سازی را انجام داد.

۴. در دید k می توان عملیات ذخیره سازی را انجام داد.

۲۷- دستور زیر در بانک اطلاعاتی تهیه کنندگان و قطعات چه می کند ؟

Select distinct S.city ,P.city
 From (S Join SP using S#) Join P using P#

۱. تمام زوج شهرهایی را می دهد که تهیه کننده شهر اول، قطعه ای را تهیه می کند که در شهر دوم انبار شده است.

۲. تمام زوج شهرهایی را می دهد که تهیه کننده و محل انبار یکی است.

۳. تمام زوج شهرهایی را می دهد که تهیه کننده شهر دوم، قطعه ای را تهیه می کند که در شهر اول انبار شده است.

۴. تمام زوج شهرهایی را می دهد که تهیه کننده و محل انبار یکی نباشند.

۲۸- دستور زیر چه می کند ؟

Select sname from S
 Where S# not in (select S# from SP)

۱. نام تهیه کنندگانی را می دهد که تمام قطعات را تهیه می کنند.

۲. نام تهیه کنندگانی را می دهد که هیچ قطعه ای را تهیه نمی کنند.

۳. این دستور غلط است.

۴. نام تهیه کنندگانی را می‌دهد که حداقل یک قطعه را تهیه کرده‌اند.

۲۹- کدام گزینه درست است ؟

۱. دستور `select S# from SP Group by S# having Count (S#)>3`

شماره تهیه کنندگانی را می‌دهد که بیشتر از ۳ قطعه تهیه می‌کنند.

۲. دستور `select S# from SP Group by S# having AVG (Qty)>200`

شماره تهیه کنندگانی را می‌دهد که میانگین قطعه تهیه شده آنها بیشتر از ۲۰۰ باشد.

۳. هیچکدام

۴. هر دو

۳۰- جدول دانشجو (stud) شامل دو ستون نام (Sname) و معدل (m) می‌باشد. کدام

دستور نام دانشجویانی را می‌دهد که معدل آنها از میانگین همه معدل‌ها بیشتر است ؟

۱. `Select Sname from stud where m> AVG (m)`

۲. `Select Sname from stud where m> (select AVG (m) from stud)`

۳. هیچکدام

۴. هر دو

۳۱- کدام دستور سطرهایی از جدول S را می‌دهد که ستون city آن ناشناخته

نمی‌باشد؟

۱. `select * from S where city is not NULL`

۲. `select * from S where not (city is NULL)`

۳. 2,1

۴. `select * from S where city = not NULL`

۳۲- Group by در SQL پیاده سازی کدام عملگر در جبر رابطه‌ای است ؟

۱. Restrict

۲. Summarize

۳. extend

۴. union

۳۳- عملگر IN در SQL معادل کدام عملگر است ؟

۱. =ANY

۲. <ANY

۳. ALL

۴. = ALL

۳۴- کدام نوع جدول همواره قابل دسترسی است ولی ممکن است بهنگام سازی در آن امکان پذیر نباشد ؟

۱. جدول اصلی

۲. جدول مجازی

۳. جدول موقتی

۴. هر سه گزینه

۳۵- یک سیستم بانک اطلاعاتی را کاملاً رابطه‌ای (Fully relational) گوییم اگر:

۱. مبتنی بر مفاهیم و ساختار رابطه‌ای بوده و در DSL آن، همان زبان جبر رابطه‌ای باشد.

۲. مبتنی بر مفاهیم و ساختار رابطه‌ای بوده و در DSL آن، همان زبان محاسبات رابطه‌ای باشد.

۳. مبتنی بر مفاهیم و ساختار رابطه‌ای بوده و در DSL آن حداقل هم توان زبان محاسبات رابطه‌ای باشد.

۴. مبتنی بر مفاهیم و ساختار رابطه‌ای بوده و در DSL آن حداقل هم توان زبان جبر رابطه‌ای باشد.

پاسخ تست های سری ۸

۱- (۱) SQL زبان پرس و جوی ساختیافته و قوی برای ایجاد، جستجو و دستکاری داده‌های یک بانک می‌باشد.

۲- (۴) دستور select فقط برای جستجو در بانک است و تغییری را در آن ایجاد نمی‌کند. تنها دستوراتی که باعث تغییر در بانک می‌شوند می‌توانند احتمالاً باعث ناسازگاری داده‌ها شوند. دستور INSERT برای درج، DELETE برای حذف و UPDATE برای تغییر است.

۳- (۲) شرط جلوی Where می‌گویید شهر هر سه (تولیدکننده-قطعه و پروژه) باید یکی باشد.

۴- (۳)

۵- (۱) البته ظاهراً گزینه ۲ نیز درست می‌باشد.

۶- (۱) به جای علامت % هر کاراکتر یا مجموعه‌ای از کاراکترها می‌تواند قرار گیرد مثل

* موجود در DOS

۷- (۴) اگر توجه کنید خروجی اشتراک بین دو ستون Name در دو جدول S و T می‌باشد برای یافتن اشتراک دو جدول از Intersect استفاده می‌کنیم.

۸- (۳) اگر توجه کنید خروجی تفاضل S-T می‌باشد یعنی چه نام‌هایی در جدول S هست که در جدول T نیست.

۹- (۱) اولین قدم در پردازش این تقاضا این است که عبارتی که V را تعریف می‌کند به جای V قرار دهیم:

((((S JOIN SP) where P# = 'P2') {S#, City })
Where city = 'London') {S#}

این عبارت به صورت زیر خلاصه می‌شود:

((S Where city = 'London') Join
(SP Where P# = 'P2')) {S#}

۱۰- (۱)

۱۱- (۳)

۱۲- (۲) ابتدا select داخلی اجرا شده و شهر S1 را می‌دهد و سپس select بیرونی اجرا می‌شود و تهیه کنندگانی را می‌دهد که همشهری S1 می‌باشند.

۱۳- (۱) مثلاً ALTER table student Modify (Lname char (40))

۱۴- (۳)

۱۵- (۴)

گزینه ۱: اینکه در SQL یک عمل را می توان به صورت های مختلف انجام داد یکی از معایب SQL به حساب می آید چرا که برنامه نویس را باین مساله درگیر می کند که کدام راه حل کارتر است.

گزینه ۲: پیوند جداول کارتر از select متداخل است.

گزینه ۳: Count (*) برای شمارش سطرهای جداول استفاده می شود دراین تابع نمی توان از Distinct استفاده کرد و سطرهای NULL را هم می شمارد.

۱۶- (۴) SQL یک زبان محاوره ای و غیر رویه ای (non-procedural) است که در آن برنامه نویس تنها می گوید چه می خواهد ولی روش بدست آوردن جواب را معین نمی سازد. در رابطه با گزینه (۱) ممکن است بگوئید خروجی می تواند یک عدد تنها باشد. ولی همین عدد تنها به عنوان خروجی یک دستور SQL، در واقع رابطه ای است که یک سطر و یک ستون دارد.

۱۷- (۲)

۱۸- (۱)

۱۹- (۲)

۲۰- (۲)

۲۱- (۲)

۲۲- (۲)

۲۳- (۴)

۲۴- (۳)

۲۵- (۱)

۲۶- (۳) اگر در دید K تغییراتی دهیم که ستون W آن (که جمع ستون Z جدول پایه ای R می باشد) تغییر کند، آنگاه DBMS نمی تواند تغییرات W را به Z اعمال کند.

۲۷- (۱)

۲۸- (۲)

۲۹- (۴)

۳۰- (۲) باید توجه داشت که گزینه ۱ نادرست است و جواب غلطی می‌دهد. چرا که تا هر جای جدول که پیش می‌رود، مقدار تدریجی $AVG(m)$ عددی غلط را نشان می‌دهد در صورتی که اگر مثلاً دانشجوی سطر دهم معدلش از این عدد بیشتر باشد، در خروجی ظاهر می‌شود. در گزینه ۲ ابتدا $select$ داخلی اجراء شده و نتیجه آن که یک عدد است استخراج می‌گردد.

۳۱- (۳) توجه کنید که دو عبارت زیر معادل یکدیگرند:

$(< \text{نام ستون} > \text{ is NULL }) \iff \text{NOT } (< \text{نام ستون} > \text{ is not NULL })$

۳۲- (۲)

۳۳- (۱)

۳۴- (۲)

۳۵- (۱)

تست های سری ۹

۱- کدام گزینه در رابطه با ((وابستگی تابعی)) درست است ؟

۱. صفت خاصه Y از رابطه R با صفت خاصه X از رابطه R وابستگی تابعی

کامل دارد، اگر Y با X وابستگی تابعی داشته باشد ولی با هیچ یک از زیر

مجموعه‌های X وابستگی تابعی نداشته باشد.

۲. صفت خاصه Y از رابطه R با صفت خاصه X از رابطه R وابستگی تابعی دارد

اگر و فقط اگر در طول حیات رابطه به هر مقدار X حداقل یک مقدار Y

متناظر باشد.

۳. هر دو

۴. هیچکدام

۲- اگر رابطه $R = \{A, B, C, D, E, F, G\}$ دارای FD های زیر باشد:

$$F = \{ AF \rightarrow BE, FC \rightarrow DE, F \rightarrow CD, D \rightarrow E, C \rightarrow A \}$$

کلید کاندید این رابطه کدام است ؟

۱. (A, F, C, D)

۲. (F, A)

۳. (F, A, G)

۴. (F, G)

۳- کدام مجموعه FD ها کاهش ناپذیرند ؟

$$F = \{ A \rightarrow BC, A \rightarrow D, A \rightarrow E \} \quad ۱.$$

$$F = \{ AB \rightarrow C, A \rightarrow B, A \rightarrow D, A \rightarrow E \} \quad ۲.$$

$$F = \{ A \rightarrow B, A \rightarrow C, A \rightarrow D, A \rightarrow E \} \quad ۳.$$

$$F = \{ A \rightarrow A, A \rightarrow B, A \rightarrow C, A \rightarrow D, A \rightarrow E \} \quad ۴.$$

۴- در رابطه $R = \{ A, B, C, D \}$ وابستگی های زیر وجود دارد:

$$F = \{ A \rightarrow BE, B \rightarrow C, A \rightarrow B, AB \rightarrow C, AC \rightarrow D \}$$

مجموعه کاهش ناپذیر (کمینه) از وابستگی ها کدام است ؟

$$\{ A \rightarrow B, B \rightarrow C, A \rightarrow D \} \quad ۱.$$

$$\{ A \rightarrow B, A \rightarrow C, A \rightarrow D \} \quad ۲.$$

$$\{ B \rightarrow C, B \rightarrow D, A \rightarrow C \} \quad ۳.$$

$$\{ A \rightarrow B, B \rightarrow D, D \rightarrow C \} \quad ۴.$$

۵- رابطه $R = \{ A, B, C, D, E, F, G \}$ دارای وابستگی های تابعی زیر است:

$$F = \{ A \rightarrow B, BC \rightarrow DE, AEF \rightarrow G \}$$

کدام وابستگی از آن قابل استنتاج است ؟

$$ACF \rightarrow DG \quad ۱.$$

$$AC \rightarrow DE \quad ۲.$$

$$EF \rightarrow G \quad ۳.$$

$$۲ \text{ و } ۴ \quad ۴.$$

۶- کدام گزینه در رابطه با دو مجموعه از FD های زیر برای رابطه $R = \{A, B, C, D, E\}$

درست است ؟

$$F = \{ A \rightarrow B, AB \rightarrow C, D \rightarrow AC, D \rightarrow E \}$$

$$G = \{ A \rightarrow BC, D \rightarrow AE \}$$

۱. F و G هم ارزند.

۲. G و F هم ارز نیستند.

۳. G مجموعه کمینه F می باشد.

۴. F مجموعه بهینه G می باشد.

۷- کدام گزینه درست است ؟

۱. برای هر مجموعه از FD ها، حداکثر یک مجموعه هم ارز وجود دارد که کاهش ناپذیر است.

۲. برای هر مجموعه از FD ها، حداقل یک مجموعه هم ارز وجود دارد که کاهش ناپذیر است.

۳. برای هر مجموعه از FD ها، دقیقاً و فقط یک مجموعه هم ارز وجود دارد که کاهش ناپذیر است.

۴. هیچکدام

۸- اگر برای رابطه $R = \{A, B, C, D, E, F, G, H, I, J\}$ وابستگی های زیر را داشته باشیم کدام گزینه درست خواهد بود ؟

$F = \{ABC \rightarrow E, AB \rightarrow G, B \rightarrow F, C \rightarrow J, CJ \rightarrow I, G \rightarrow H\}$

۱. F یک مجموعه کاهش پذیر است و $\{A, B, C, D\}$ کلید کاندید است.

۲. F یک مجموعه کاهش ناپذیر است و $\{A, B, C, D\}$ کلید کاندید است.

۳. F یک مجموعه کاهش ناپذیر است و $\{A, B, C, D, G\}$ کلید کاندید است.

۴. F یک مجموعه کاهش پذیر است و $\{A, B, C\}$ کلید کاندید است.

۹- رابطه TIMETABLE را با صفات زیر در نظر بگیرید:

دوره زمانی در روز (۱ تا ۸) $P =$ و شماره روز هفته $D =$

شماره درس $L =$ نام معلم $T =$ شماره کلاس $C =$

فرض کنید هر درس در دوره خاصی (D,P) ارائه می شود و شماره دروس منحصر به

فرد می باشند. کلید کاندید این رابطه کدام است ؟

۱. L

۲. DPC

۳. DPT

۴. هر سه گزینه

۱۰- مجموعه کاهش ناپذیر مجموعه وابستگی زیر کدام است ؟

$$F = \{A \rightarrow D, A \rightarrow C, CD \rightarrow B, AD \rightarrow C\}$$

۱. $\{CD \rightarrow B, AD \rightarrow C\}$

۲. $\{A \rightarrow D, A \rightarrow C, AD \rightarrow C\}$

۳. $\{A \rightarrow D, CD \rightarrow B, AD \rightarrow C\}$

۴. $\{A \rightarrow D, A \rightarrow C, CD \rightarrow B\}$

۱۱- از مجموعه FD زیر کدام رابطه را نمی توان استنتاج کرد ؟

$$F = \{A \rightarrow D, CD \rightarrow B, AD \rightarrow C\}$$

۱. $C \rightarrow B$

۲. $A \rightarrow B$

۳. $A \rightarrow C$

۴. $A \rightarrow CD$

۱۲- کدام گزینه درست است ؟

۱. به مجموعه تمام وابستگی های استنتاج شده، مجموعه پوششی می گوئیم.

۲. اگر مجموعه پوششی دو مجموعه FD با هم برابر باشند آنگاه دو مجموعه

وابستگی با هم معادلند.

۳. قوانین ۳ گانه آرمسترانگ برای بدست آوردن مجموعه پوششی کافی است.

۴. هر سه گزینه

۱۳- نمودار وابستگی تابعی:

۱. نموداری است که جداول را تجزیه می کند.

۲. نموداری است که شروع کار آن با کلیدهای کاندید است.

۳. نموداری است که شروع کار آن با کلیدهای خارجی است.

۴. نموداری است که وابستگی به کلید خارجی را نمایش می دهد.

۱۴- وابستگی تابعی بدیهی کدام است ؟

۱. اگر سمت چپ آن، زیر مجموعه ای از سمت راست باشد.

۲. اگر سمت چپ آن قابل تجزیه شدن نباشد.

۳. اگر سمت راست آن فقط یک صفت خاصه باشد.

۴. اگر سمت راست آن، زیر مجموعه ای از سمت چپ باشد.

۱۵- در رابطه $R = \{A, B, C, D, E, F, G\}$ با FD های زیر کلید کاندید کدام است ؟
 $F = \{D \rightarrow A, E \rightarrow B, A \rightarrow B, AB \rightarrow EFC\}$

۱. (A, B, D)

۲. (D)

۳. (D, G)

۴. (B, D, E)

۱۶- کدام گزینه نادرست است ؟

۱. در رابطه تمام کلید، بین اجزای کلید می تواند وابستگی تابعی وجود داشته باشد.

۲. اگر K سوپر کلید رابطه R باشد آنگاه مجموعه عنوان R با K وابستگی تابعی دارد.

۳. از $AB \rightarrow C$ در حالت کلی نمی توان نتیجه گرفت که $A \rightarrow C$ و $B \rightarrow C$

۴. FD ها محدودیت جامعیت را نشان می دهند.

۱۷- رابطه $REL(X, Y, Z)$ را در نظر می گیریم. کدام یک از گزاره های زیر نادرست است ؟

۱. اگر $X \rightarrow Y$ و $Y \rightarrow X$ داریم: $X \rightarrow Z$

۲. رابطه حاصل JOIN دو رابطه $REL1(X, Y)$ و $REL2(X, Z)$ است اگر و

فقط اگر داشته باشیم: $X \rightarrow Y$

۳. اگر $Y \rightarrow X$ و $Z \rightarrow X$ داریم: $(Y, Z) \rightarrow X$

۴. اگر $X \rightarrow Y$ و $X \rightarrow Z$ داریم: $X \rightarrow (Y, Z)$

پاسخ تست های سری ۹

۱- (۱) صفت خاصه Y از رابطه R با صفت خاصه X از رابطه R وابستگی تابعی دارد و می نویسیم $R.X \rightarrow R.Y$ اگر و فقط اگر در طول حیات رابطه به هر مقدار از X در رابطه دقیقاً یک مقدار Y از رابطه R متناظر باشد.

۲- (۴) ابتدا FOPT را بدست می آوریم:

$$AF \rightarrow BE \Rightarrow AF \rightarrow B, AF \rightarrow E$$

$$\begin{aligned}
FC \rightarrow DE &\Leftrightarrow FC \rightarrow D, FC \rightarrow E \\
F \rightarrow CD &\Leftrightarrow F \rightarrow C, F \rightarrow D \\
F \rightarrow C, FC \rightarrow DE &\Leftrightarrow F \rightarrow DE \Leftrightarrow F \rightarrow D, F \rightarrow E \\
F \rightarrow C, C \rightarrow A &\Leftrightarrow F \rightarrow A \\
F \rightarrow A, FA \rightarrow B &\Leftrightarrow F \rightarrow B \\
F \rightarrow A, FA \rightarrow E &\Leftrightarrow F \rightarrow E
\end{aligned}$$

$$FOPT = \{F \rightarrow A, F \rightarrow B, F \rightarrow C, F \rightarrow D, F \rightarrow E, D \rightarrow E, C \rightarrow A\}$$

پس F همه صفتهای دیگر به جز G را می دهد پس (F,G) کلید کاندید است. این کلید منحصر به فرد است زیرا هیچ صفتی F و G را نمی دهد. یعنی در هر کلید کاندید دیگر وجود این دو صفت لازم است.

۳- (۳) در گزینه ۱ در سمت راست یک صفت تنها نیست.

در گزینه ۲: $A \rightarrow B, AB \rightarrow C \Leftrightarrow A \rightarrow C$ یعنی سمت چپ قابل کاهش است.

در گزینه ۴: $A \rightarrow A$ قابل حذف شدن است و عضو زائدی است.

۴- (۱)

$$\begin{aligned}
A \rightarrow BC &\Leftrightarrow A \rightarrow B, A \rightarrow C \\
A \rightarrow B, AB \rightarrow C &\Leftrightarrow A \rightarrow C \\
A \rightarrow C, AC \rightarrow D, A \rightarrow D &\Leftrightarrow F = \{A \rightarrow B, A \rightarrow C, B \rightarrow C, A \rightarrow B, A \rightarrow C, A \rightarrow D\} \\
F &= \{A \rightarrow B, A \rightarrow C, B \rightarrow C, A \rightarrow D\}
\end{aligned}$$

از آنجا که:

$$\begin{aligned}
A \rightarrow B, B \rightarrow C &\Leftrightarrow A \rightarrow C \\
F_{min} &= \{A \rightarrow B, B \rightarrow C, A \rightarrow D\}
\end{aligned}$$

پس

۵- (۴)

۶- (۱)

$$F \text{ مجموعه } A \rightarrow B, AB \rightarrow C \Leftrightarrow A \rightarrow C$$

$$\begin{aligned}
D \rightarrow AC &\Leftrightarrow D \rightarrow A, D \rightarrow C \\
F &= \{A \rightarrow B, A \rightarrow C, D \rightarrow A, D \rightarrow C, D \rightarrow E\} \\
\Rightarrow F_{min} &= \{A \rightarrow B, A \rightarrow C, D \rightarrow A, D \rightarrow E\} \\
G \text{ مجموعه } \rightarrow G_{min} &= \{A \rightarrow B, A \rightarrow C, D \rightarrow A, D \rightarrow E\}
\end{aligned}$$

۷- (۲)

۸- (۱)

مجموعه کاهش پذیر است چرا که $C \rightarrow J, CJ \rightarrow I \Rightarrow C \rightarrow I$

$\{A, B, C, D, G, J\}$ یک فوق کلید است (مجموعه‌ای از تمام صفاتی که در سمت چپ FD ها وجود دارد). به دلیل وجود $C \rightarrow J$ می‌توانیم J را از مجموعه فوق حذف کنیم. به دلیل وجود $AB \rightarrow G$ می‌توانیم G را از مجموعه حذف کنیم. چون A و B و C و D هیچکدام در سمت راست FD ها ظاهر نشده‌اند پس $\{A, B, C, D\}$ کلید کاندید است.

۹- (۴)

$L \rightarrow T, L \rightarrow C, L \rightarrow D, L \rightarrow P$

اگر مثلاً بگوئیم استاد اکبری سه شنبه ۸ تا ۱۰ صبح، آنگاه شماره درس و شماره کلاس او مشخص می‌گردد پس $DPT \rightarrow CL$. همچنین اگر مثلاً بگوئیم سه شنبه ۸ تا ۱۰ صبح کلاس شماره ۲۱۵، آنگاه نام استاد و شماره درس تعیین می‌گردد، پس $DPC \rightarrow TL$

۱۰- (۴)

$A \rightarrow D, AD \rightarrow C \Rightarrow A \rightarrow C$
 $F_{min} = \{ A \rightarrow D, A \rightarrow C, CD \rightarrow B \}$

۱۱- (۱)

$A \rightarrow D, AD \rightarrow C \Rightarrow A \rightarrow C$
 $A \rightarrow D, A \rightarrow C \Rightarrow A \rightarrow CD$
 $A \rightarrow CD, CD \rightarrow B \Rightarrow A \rightarrow B$

۱۲- (۴)

۱۳- (۲)

۱۴- (۴) مثلاً $AB \rightarrow A$ گزینه‌های ۲ و ۳ ویژگی‌های وابستگی تابعی کمینه را بیان می‌کنند.

۱۵- (۳)

$D \rightarrow A, A \rightarrow B \Rightarrow D \rightarrow B,$
 $D \rightarrow A, D \rightarrow B \Rightarrow D \rightarrow AB$
 $D \rightarrow AB, AB \rightarrow EFC \Rightarrow D \rightarrow E, D \rightarrow F, D \rightarrow C$

۱۶- (۱) در رابطه تمام کلید، بین اجزای کلید وابستگی تابعی وجود ندارد.

۱۷- (۲)

تست های سری ۱۰

۱- فرض کنید لیستی از درس های دانشجویان در اختیار داریم که از ستونهای زیر تشکیل شده است:

تعداد واحد	نام درس	کد درس	نام دانشجو	کد دانشجو
هر دانشجو ممکن است چندین درس داشته باشد. اگر بخواهیم این اطلاعات را در یک بانک اطلاعاتی به گونه ای ذخیره کنیم که حداقل تکرار اطلاعات را داشته باشیم حداقل به چند جدول (table) نیاز داریم ؟				

۲. ۳

۳. ۲

۴. ۴

۲- کدامیک از گزاره های زیر در رابطه با ((عمده ترین اهداف نرمال ترسازی رابطه ها)) نادرست است ؟

۱. کاهش بعضی انواع افزونگی (Redundancy)
 ۲. کاهش سربار (Overhead) سیستم در پاسخگوئی در عمل بازیابی
 ۳. تسهیل در اعمال بعضی محدودیتهای جامعیتی (integrity constraints)
 ۴. اجتناب از بعضی از انواع آنومالی در عملیات روی پایگاه
- ۳- کدام گزینه از اهداف کلی فرایند نرمال سازی (normalization) نیست ؟
۱. تسهیل اعمال بعضی از قواعد جامعیتی (integrity rules)
 ۲. تسهیل پیاده سازی دید کاربر (user view)
 ۳. حذف بعضی از انواع قواعد وابستگی تابعی (functional dependency)
 ۴. کاهش بعضی از انواع افزونگی (redundancy)
- ۴- کدام گزینه نادرست است ؟

۱. رابطه 1NF را همواره می توان به تعدادی رابطه 2NF تبدیل کرد.
۲. در تبدیل رابطه 1NF به 2NF باید چنان عمل کرد که وابستگی های تابعی غیر کامل موجود در رابطه 1NF از میان بروند.
۳. همیشه پس از تجزیه یک رابطه به دو رابطه و پیوند مجدد روابط حاصله ، لزوما به رابطه نخستین می رسیم.
۴. با انجام عمل پیوند طبیعی بر روی روابط 2NF می توان روابط 1NF را بدست آورد.

۵- کدام گزینه در رابطه با 2NF درست است ؟

۱. فقط در درج ممکن است آنومالی داشته باشیم.

۲. فقط در حذف ممکن است آنومالی داشته باشیم.
۳. در حذف و بهنگام سازی ممکن است آنومالی داشته باشیم.
۴. در درج ، حذف و بهنگام سازی ممکن است آنومالی داشته باشیم.
- ۶- کدام تعریف در رابطه با صورتهای نرمال نادرست است ؟
 ۱. رابطه‌ای نرمال است که تمام مقادیر صفات خاصه اش اتومیک باشند.
 ۲. رابطه‌ای 3NF است اگر و فقط اگر صفات خاصه غیر کلید متقابلاً به یکدیگر ناوابسته باشند و همچنین با کلید اصلی رابطه ، وابستگی تابعی کامل داشته باشند.
 ۳. رابطه‌ای 2NF است اگر و فقط اگر 1NF باشد و همچنین اگر صفات خاصه غیر کلید متقابلاً به یکدیگر ناوابسته باشند.
 ۴. رابطه‌ای 3NF است اگر و فقط اگر 2NF بوده و هر صفت خاصه غیر کلید به طور مستقیم (بی واسطه) با کلید اصلی وابستگی داشته باشد.
- ۷- کدام گزینه قضیه هیث (Heath) را بیان می کند ؟ (رابطه R را با سه صفت خاصه A و B و C در نظر بگیرید)
 ۱. اگر $R.A \rightarrow R.B$ همیشه می توان این رابطه را به دو رابطه $R_2(A,C)$ و $R_1(A,B)$ تجزیه کرد به نحوی که گمشدگی اطلاعات پیش نیاید.
 ۲. اگر $R.A \rightarrow R.B$ و $R.A \rightarrow R.C$ می توان این رابطه را به دو رابطه $R_2(A,C)$ و $R_1(A,B)$ تجزیه کرد.
 ۳. اگر $R.A \rightarrow R.B$ همیشه می توان این رابطه را به دو رابطه $R_2(A,C)$ و $R_1(A,B)$ تجزیه کرد ولی ممکن است گمشدگی اطلاعات پیش آید.
 ۴. $R.A \rightarrow R.B$ و $R.A \rightarrow R.C$ ممکن است نتوان این رابطه را به دو رابطه $R_2(A,C)$ و $R_1(A,B)$ تجزیه کرد.
- ۸- در کدام مورد زیر 3NF مطلوبترین صورت نیست ؟
 ۱. وقتی رابطه دارای چند کلید کاندید باشد.

۲. وقتی که کلیدهای کاندید رابطه مرکب باشند.
 ۳. وقتی که کلیدهای کاندید با یکدیگر اشتراک صفت خاصه داشته باشند.
 ۴. هر سه مورد.
- 9- در کدام حالت جدول 3NF ممکن است مشکل داشته باشد ؟
۱. جدول دارای حداقل دو کلید کاندید باشد.
 ۲. کلیدهای کاندید ترکیبی باشند.
 ۳. کلیدهای کاندید ترکیبی ، صفتهای مشترکی داشته باشند.
 ۴. وقتی که هر سه شرط فوق برقرار باشد.
- 10- کدام گزینه اهداف کلی نرمال سازی را بیان نمی کند ؟
۱. رعایت قواعد جامعیتی
 ۲. کاهش افزونگی
 ۳. کاهش سربار سیستم و ساده کردن دید کاربر
 ۴. کاهش بعضی از آنومالی ها
- 1۱- کدام گزینه درست است ؟
۱. رابطه R در سطح 3NF است اگر و فقط اگر 2NF باشد و هر صفت غیر کلید با کلید اصلی وابستگی تابعی کامل داشته باشد.
 ۲. رابطه R در سطح 3NF است اگر و فقط اگر 2NF باشد و صفات غیر کلید آن متقابلاً به یکدیگر ناوابسته باشند.
 ۳. رابطه R در سطح 3NF است اگر و فقط اگر 2NF باشد و وابستگی انتقالی (transitive) نداشته باشد.
 ۴. ۲ و ۳
- 12- صفت عمده Prime Attribute کدام است ؟
۱. صفتی که جزء تشکیل دهنده کلید باشد.
 ۲. همان صفت غیر کلید است.

۳. همان کلید است.

۴. هیچکدام

13- کدام گزینه نادرست است ؟

۱. ممکن است در مرحله 2NF کردن یک رابطه ، رابطه 1NF به بیش از دو رابطه تجزیه شود.

۲. یک صفت غیر کلید ممکن است بتواند با یک صفت غیر کلید دیگر وابستگی تابعی ناتام (غیر کانل) داشته باشد.

۳. در رابطه دو گانی (باینری) BCNF است.

۴. هیچکدام

14- تعریف BCNF کدام است ؟

۱. رابطه ای BCNF است اگر در آن هر دترمینان ، کلید کاندید باشد.

۲. رابطه R در BCNF است هرگاه یک وابستگی تابعی به صورت $A \rightarrow B$ (این وابستگی نامهم نباشد) در مجموعه FD های R وجود داشته باشد ، A سوپر کلید R باشد.

۳. رابطه ای BCNF است اگر و فقط اگر سمت چپ هر FD مهم و کاهش ناپذیر ، کلید کاندید رابطه باشد.

۴. هر سه گزینه

۱۵- رابطه $R(SID, CID, PID)$ یعنی (شماره استاد ، شماره درس ، شماره دانشجو) R را در نظر بگیرید. قواعد سمانتیک زیر در این رابطه وجود دارد: الف) یک دانشجو ، یک درس را با یک استاد انتخاب می کند. ب) یک استاد یک درس را تدریس می کند. ج) یک درس توسط چند استاد تدریس می شود. این رابطه در چه سطح نرمال بودن قرار دارد ؟

۱. 2NF

۲. 3NF

۳. BCNF

۴. 4NF

۱۶- رابطه‌ای 2NF است که:

۱. 1NF بوده و صفت‌های آن به زیر مجموعه‌های کلید اصلی وابستگی نداشته باشد.

۲. 1NF بوده و وابستگی انتقالی (تعدی) نداشته باشد.

۳. ۲و۱

۴. هیچکدام

۱۷- در رابطه $R=\{A,B,C,D\}$ ، مجموعه FD های زیر را داریم:

$A \rightarrow D, D \rightarrow B, A \rightarrow C$

سطح نرمال بودن این رابطه و تجزیه مطلوب آن کدام است؟

۱. 3NF، $R1(\underline{D},B)$ ، $R2(\underline{A},D,C)$

۲. BCNF، $R1(\underline{A},D)$ ، $R2(\underline{A},B,C)$

۳. 3NF، $R1(\underline{A},C,D)$ ، $R2(\underline{A},D)$

۴. 2NF، $R1(\underline{D},B)$ ، $R2(\underline{A},C,D)$

۱۸- با حذف کلید وابسته، چه صورتی از فرم نرمال ایجاد می شود؟

۱. 1NF

۲. 2NF

۳. 3NF

۴. 4NF

۱۹- با حذف ابر کلید، چه صورتی از فرم نرمال ایجاد می شود؟

۱. 1NF

۲. 2NF

۳. 3NF

۴. 4NF

۲۰- جامع ترین نرمال سازی بر مبنای وابستگی تابعی توسط کدام صورت نرمال ارائه

می شود؟

۱. 1NF

۲. 2NF

۳. 3NF

۴. BCNF

۲۱- با حذف وابستگی‌های چند مقداری (MVD) چه صورتی از فرم نرمال ایجاد می

شود؟

۱. 2NF
۲. 3NF
۳. 4NF
۴. BCNF

۲۲- در تبدیل..... به..... آن را چنان به پرتوهایی تجزیه می کنیم که تمام وابستگی های پیوندی که از کلیدهای کاندید ناشی نمی شود حذف گردد.

۱. 3NF-2NF
۲. 4NF-3NF
۳. BCNF-3NF
۴. 5NF-4NF

۲۳- برای اینکه یک پایگاه داده ای رابطه ای بتواند به درستی در نرم افزار متداول طراحی شود ، آن پایگاه داده ها باید حداقل بر اساس چه فرم نرمال (Normal Form) قرار گرفته باشد ؟

۱. اول
۲. دوم
۳. سوم
۴. چهارم

۲۴- اگر جدولی یک فیلد داشته باشد که بستگی به قسمتی از کلید اصلی داشته ، این نکته نقض کدامیک از قوانین فرم نرمال است ؟

۱. فرم اول نرمال
۲. فرم دوم نرمال
۳. فرم سوم نرمال
۴. فرم چهارم نرمال

۲۵- اگر ۲ فیلد غیر کلیدی بر یکدیگر بستگی داشته باشند ، این نقض کدامیک از قوانین فرم های نرمال است ؟

۱. اولین فرم نرمال
۲. دومین فرم نرمال
۳. سومین فرم نرمال

۴. چهارمین فرم نرمال

۲۶- به طور کلی چگونه تاثیری از عملیات نرمالیزه کردن بر روی کارکرد بانک اطلاعاتی به وجود می آید؟

۱. انعطاف کمتری در ذخیره سازی داده ها دارد.

۲. انعطاف بیشتری در ذخیره سازی داده ها دارد.

۳. انعطافی در ذخیره سازی داده ها ندارد.

۴. هیچکدام

۲۷- وقتی در یک جدول تعدادی ستون برای ذخیره سازی موردهای مختلف از یک داده وجود دارد مانند ... , phonenum1 , phonenum2 این نکته کدام یک از موارد فرمهای نرمال را نقض می کند؟

۱. فرم اول نرمال

۲. فرم دوم نرمال

۳. فرم سوم نرمال

۴. فرم چهارم نرمال

۲۸- چه دلیلی دارد که اطلاعات یک پایگاه اطلاعاتی را بین چندین فایل توزیع نمائیم؟

۱. برای جلوگیری از خرابی داده ها

۲. برای اینکه در بانکهای بزرگ سیستم عامل دچار مشکل نشود.

۳. برای مدیریت بهتر داده ها

۴. برای بهبود سرعت دستیابی به داده ها

۲۹- جدول در.... است که الف) همه کلیدهای آن تعریف شده باشند ب) تمام صفت‌های آن به کلید اصلی وابستگی تابعی داشته باشند ج) صفت‌های آن از دامنه تودرتو (nested domain) نباشند.

۱. 1NF

۲. 2NF

۳. 3NF

۴. 4NF

۳۰- در کدام فرم نرمال وابستگی انتقالی (transitive) وجود ندارد ؟

۱. 2NF
۲. 3NF, 2NF
۳. 4NF, 3NF
۴. 2NF

۳۱- جدول دانشجو (Stud) در یک موسسه آموزشی کوچک که دانشجوی همانم ندارد را در نظر بگیرید:

Stud (S# , Sname , address , avg)

S# شماره دانشجو ، Sname نام دانشجو ، Address آدرس دانشجو و avg معدل است.

این جدول حداکثر در چه سطح نرمال قرار دارد ؟

۱. 2NF
۲. 3NF
۳. 1NF
۴. BCNF

۳۲- کدام گزینه در رابطه با نرمال سازی درست است ؟

۱. اگر طراح تشخیص دهد که تجزیه یک جدول هر چند افزودنی هم داشته باشد ، باعث پائین آمدن سرعت اکثر پرس و جوها می شود ، مجاز است از نرمالتر سازی آن صرف نظر کند.
۲. روابط 3NF ی که می توانند تبدیل به BCNF شوند را حتما باید BCNF کرد
۳. در تجزیه رابطه 3NF به روابط BCNF هیچگاه اطلاعات و یا وابستگی هایی گم نمی شود.
۴. هر سه گزینه

۳۳- جدول زیر حداکثر در چه سطح نرمالی قرار دارد ؟

نام کتاب	نام درس	شهر دانشکده	نام دانشکده	نام استاد
هربرت شیلد	برنامه سازی	تهران	کامپیوتر	امیری
هربرت شیلد	برنامه سازی	اراک	برق	امیری
لیپ شوتز	ساختمان داده	تهران	کامپیوتر	امیری
لیپ شوتز	ساختمان داده	اراک	برق	امیری
دیت	پایگاه داده	تهران	کامپیوتر	امیری

دیت	پایگاه داده	اراک	برق	امیری
۱. 2NF				
۲. 3NF				
۳. BCNF				
۴. 4NF				

پاسخ تست های سری ۱۰

۱- (۲) این سه جدول عبارتند از:

جدول ۱: جدول دانشجو: شامل فیلدهای کد دانشجو و نام دانشجو

جدول ۲: جدول درس: شامل فیلدهای کد درس ، نام درس و تعداد واحد

جدول ۳: جدول دانشجو- درس: شامل فیلدهای کد دانشجو و کدهای درس

۲- (۲) از آنجا که نرمال تر سازی باعث تجزیه جداول می شود گاهی اوقات برای بازیابی نیاز به پیوند مجدد این جداول داریم که باعث افزایش سربار سیستم می گردد.

۳- (۲) تسهیل پیاده سازی دید کاربر از اهداف کلی فرآیند نرمال سازی زیرا در برخی مواقع این کار باعث مشکل تر شدن پیاده سازی دید کاربر نیز می شود.

۴- (۳) همیشه پس از تجزیه یک رابطه به دو رابطه و پیوند مجدد رابطه های حاصله لزوما به رابطه نخستین نمی رسیم.

۵- (۴) در رابطه 2NF ممکن است آنومالیهایی در درج ، حذف و بهنگام سازی داشته باشیم.

۶- (۳) گزینه های ۱ و ۲ درست می باشند. رابطه ای 2NF است اگر و فقط اگر 1NF باشد و هر صفت خاصه غیر کلید با کلید اصلی وابستگی تابعی کامل داشته باشد. توجه کنید تعریف گزینه ۴ برای 3NF معادل تعریف گزینه ۲ می باشد.

۷- (۱)

۸- (۴)

۹- (۴)

۱۰- (۳) نرمال سازی از آنجا که باعث تجزیه جداول می شود دید کاربر را به علت تعدد جداول و ارتباط بین آنها ممکن است مشکل سازد همچنین از آنجا که برای بازیابی بعضی پرس و جوها نیاز به پیوند جدولها خواهد بود ممکن است بار سیستم را افزایش دهد.

۱۱- (۴) در گزینه ۱ وقتی که رابطه در 2NF باشد ، خود به خود هر صفت خاصه غیر کلید با کلید اصلی وابستگی تابعی کامل خواهد داشت.

۱۲- (۱)

۱۳- (۴)

۱۴- (۴) کتابهای مختلف عبارات متفاوتی را برای تعریف حالات نرمال به کار می برند که همگی آنها در معنای نهایی معادل یکدیگر می باشند.

۱۵- (۲)

۱۶- (۳)

۱۷- (۴)

۱۸- (۳)

۱۹- (۲)

۲۰- (۴)

۲۱- (۳)

۲۲- (۴)

۲۳- (۳)

۲۴- (۲)

۲۵- (۳)

۲۶- (۱) نرمال سازی با اعمال محدودیتهایی جلوی یکسری افزونگی و نیز آنومالیا را می گیرد.

۲۷- (۴)

۲۸- (۳) این در واقع همان بحث نرمال سازی است ، ما داده ها را به جای ذخیره در یک جدول در چند جدول ذخیره می سازیم تا آنومالیاها و مشکلات را کاهش دهیم. این عمل بر عکس سرعت عملیات را کاهش می دهد. پس گزینه ۴ غلط است.

۲۹- (۱)

۳۰- (۳) جدولی در 3NF است که الف (در 2NF باشد ب) وابستگی انتقالی نداشته باشد. وابستگی انتقالی یعنی وابستگی دو صفت غیر کلیدی به هم زیرا هر یک از آنها به کلید اصلی وابسته اند. رابطه ای که 4NF باشد حتما 3NF است و ویژگیهای آن را دارد.

۳۱- (۴) S# و Sname کلید کاندید می باشند و Sname → Stud , S# → Stud پس

چون هر دترمینان فقط کلید کاندید می باشد در سطح BCNF قرار دارد.

۳۲- (۱) در بعضی موارد نباید روابط 3NF را به BCNF تبدیل کرد پس گزینه ۲ نادرست است. همچنین مواردی ایت که تجزیه روابط 3NF به روابط BCNF وابستگی ها را از بین می برد پس گزینه ۳ نیز نادرست است.

۳۳- (۳) جدول مذکور در سطح BCNF است زیرا هیچگونه وابستگی به جز وابستگی به کلید اصلی ، که شامل تمام ستونها است ، وجود ندارد. ولی جدول فوق افزونگی دارد ، چرا که دانشکده های استاد امیری و نیز دروسی که تدریس کرده است تکرار شده است. برای رفع این مشکل و تبدیل آن به 4NF می توان آن را به دو جدول زیر تجزیه کرد:

نام کتاب	نام درس	نام استاد	شهر دانشکده	نام دانشکده	نام استاد
هربرت شیلد	برنامه سازی	امیری	تهران	کامپیوتر	امیری
لیپ شوتز	ساختمان داده	امیری	اراک	برق	امیری
دیت	پایگاه داده	امیری			

ضمیمہ ۲

SQL 99 _ BNF

BNF Grammar for ISO/IEC 9075:1999

Database Language SQL (SQL-99)

```
--p
Note that this version of this file includes the corrections from ISO 9075:1999/Cor.1:2000.
--/p
--p
The plain text version of this grammar is
--## <a href='sql-99.bnf'> sql-99.bnf </a>.
--/p
--hr
--h2 Key SQL Statements and Fragments
--/h2
--bl
--li ALTER DOMAIN <alter domain statement>
--li ALTER TABLE <alter table statement>
--li CLOSE cursor <close statement>
--li Column definition <column definition>
--li COMMIT WORK <commit statement>
--li CONNECT <connect statement>
--li CREATE ASSERTION <assertion definition>
--li CREATE CHARACTER SET <character set definition>
--li CREATE COLLATION <collation definition>
--li CREATE DOMAIN <domain definition>
--li CREATE FUNCTION <schema function>
--li CREATE PROCEDURE <schema procedure>
--li CREATE SCHEMA <schema definition>
--li CREATE TABLE <table definition>
--li CREATE TRANSLATION <translation definition>
--li CREATE TRIGGER <trigger definition>
--li CREATE VIEW <view definition>
--li Data type <data type>
--li DEALLOCATE PREPARE <deallocate prepared statement>
--li DECLARE cursor <declare cursor> <dynamic declare cursor>
--li DECLARE LOCAL TEMPORARY TABLE <temporary table declaration>
--li DELETE <delete statement: positioned> <delete statement: searched> <dynamic delete statement:
positioned>
--li DESCRIBE <describe statement>
--li DESCRIPTOR statements <system descriptor statement>
--li DISCONNECT <disconnect statement>
```

```
--li EXECUTE <execute statement>
--li EXECUTE IMMEDIATE <execute immediate statement>
--li FETCH cursor <fetch statement>
--li FROM clause <from clause>
--li GET DIAGNOSTICS <get diagnostics statement>
--li GRANT <grant statement>
--li GROUP BY clause <group by clause>
--li HAVING clause <having clause>
--li INSERT <insert statement>
--li Literals <literal>
--li OPEN cursor <open statement>
--li ORDER BY clause <order by clause>
--li PREPARE <prepare statement>
--li REVOKE <revoke statement>
--li ROLLBACK WORK <rollback statement>
--li SAVEPOINT <savepoint statement>
--li Search condition <search condition> <regular expression>
--li SELECT <query specification>
--li SET CATALOG <set catalog statement>
--li SET CONNECTION <set connection statement>
--li SET CONSTRAINTS <set constraints mode statement>
--li SET NAMES <set names statement>
--li SET SCHEMA <set schema statement>
--li SET SESSION AUTHORIZATION <set session user identifier statement>
--li SET TIME ZONE <set local time zone statement>
--li SET TRANSACTION <set transaction statement>
--li SQL Client MODULE <SQL-client module definition>
--li UPDATE <update statement: positioned> <update statement: searched> <dynamic update
statement: positioned>
--li Value expression <value expression>
--li WHERE clause <where clause>
--/bl
--hr
--h2 Identifying the version of SQL in use
--/h2
--p
This material (starting with <SQL object identifier> ) is defined in section 6.3 "Object Identifier for
Database Language SQL" of ISO/IEC 9075-1:1999 (SQL Framework).
It is used to express the capabilities of an implementation. The package names are identifiers such as
'PKG001', equivalent to 'Enhanced datetime facilities', as defined in the informative Annex B to SQL
Framework. Each such package identifies a number of features that are provided when the SQL object
identifier claims to provide the package.
--/p
<SQL object identifier>::= <SQL provenance> <SQL variant>
<SQL provenance>::= <arc1> <arc2> <arc3>
<arc1>::=iso | 1 | iso <leftparen> 1 <rightparen>
<arc2>::=standard | 0 | standard <leftparen> 0 <rightparen>
<arc3>::=9075
<SQL variant>::= <SQL edition> <SQL conformance>
<SQL edition>::= <1987> | <1989> | <1992> | <1999>
<1987>::=0 | edition1987 <leftparen> 0 <rightparen>
<1989>::= <1989 base> <1989 package>
<1989 base>::=1 | edition1989 <leftparen> 1 <rightparen>
<1989 package>::= <integrity no> | <integrity yes>
<integrity no>::=0 | IntegrityNo <leftparen> 0 <rightparen>
<integrity yes>::=1 | IntegrityYes <leftparen> 1 <rightparen>
<1992>::=2 | edition1992 <leftparen> 2 <rightparen>
<1999>::=3 | edition1999 <leftparen> 3 <rightparen>
<SQL conformance>::= <level> <parts> <packages>
```

३३४ SQL 99 _ BNF

```
<level>::= <low> | <intermediate> | <high>
<low>::=0 | Low <leftparen> 0 <rightparen>
<intermediate>::=1 | Intermediate <leftparen> 1 <rightparen>
<high>::=2 | High <leftparen> 2 <rightparen>
<parts>::= <Part 3> <Part 4> <Part 5> <Part 6> <Part 7> <Part 8> <Part 9> <Part 10>
```

```
--p
--small
--i
```

The parenthesized (i) and (n) are italic in the SQL standard. It is not clear exactly what this should look like, despite all the Information. However, it is also not important; this is not really a part of the SQL language per se. Note that the package numbers are PKG001 to PKG009, for example. We still have to devise a mechanism to persuade bnf2yacc.pl to ignore this information.

```
--/i
--/small
--/p
<packages>::= <Package PKG(i)>...
<Part (n)>::= <Part (n) no> | <Part (n) yes>
<Part (n) no>::=0 | Part-(n)No <leftparen> 0 <rightparen>
<Part (n) yes>::=!! (as specified in ISO/IEC 9075-(n))
<Package PKG(i)>::= <Package PKG(i)Yes> | <Package PKG(i)No>
<Part 3 yes>::= <Part 3 conformance>
<Part 3 conformance>::=3 | sqlcli1999 <leftparen> 3 <rightparen>
<Part 4 yes>::= <Part 4 conformance> <Part 4 module>
<Part 4 conformance>::=4 | sqlpsm1999 <leftparen> 4 <rightparen>
<Part 4 module>::= <Part 4 module yes> | <Part 4 module no>
<Part 4 module yes>::=1 | moduleyes <leftparen> 1 <rightparen>
<Part 4 module no>::=0 | moduleno <leftparen> 0 <rightparen>
<Part 5 yes>::= <Part 5 conformance> <Part 5 direct> <Part 5 embedded>
```

```
--p
--small
--i
```

The original used sqlbindings199x, but the x should clearly be a 9.

```
--/i
--/small
--/p
<Part 5 conformance>::=5 | sqlbindings1999 <leftparen> 5 <rightparen>
<Part 5 direct>::= <Part 5 direct yes> | <Part 5 direct no>
<Part 5 direct yes>::=1 | directyes <leftparen> 1 <rightparen>
<Part 5 direct no>::=0 | directno <leftparen> 0 <rightparen>

<Part 5 embedded>::= <Part 5 embedded no> | <Part 5 embedded languages>...
<Part 5 embedded no>::=0 | embeddedno <leftparen> 0 <rightparen>
<Part 5 embedded languages>::=
    | <Part 5 embedded Ada>
    | <Part 5 embedded C>
    | <Part 5 embedded COBOL>
    | <Part 5 embedded Fortran>
    | <Part 5 embedded MUMPS>
    | <Part 5 embedded Pascal>
    | <Part 5 embedded PL/I>
<Part 5 embedded Ada>::=1 | embeddedAda <leftparen> 1 <rightparen>
<Part 5 embedded C>::=2 | embeddedC <leftparen> 2 <rightparen>
<Part 5 embedded COBOL>::=3 | embeddedCOBOL <leftparen> 3 <rightparen>
<Part 5 embedded Fortran>::=4 | embeddedFortran <leftparen> 4 <rightparen>
<Part 5 embedded MUMPS>::=5 | embeddedMUMPS <leftparen> 5 <rightparen>
<Part 5 embedded Pascal>::=6 | embeddedPascal <leftparen> 6 <rightparen>
<Part 5 embedded PL/I>::=7 | embeddedPLI <leftparen> 7 <rightparen>
```

```
--hr
```

--h2 Basic Definitions of Characters Used, Tokens, Symbols, Etc.

```
--/h2
--p
Most of this section would normally be handled within the lexical analyzer rather than in the grammar
proper. Further, the original document does not quote the various single characters, which makes it hard
to process automatically.
--/p
<SQL terminal character>::= <SQL language character>
<SQL language character>::= <simple Latin letter> | <digit> | <SQL special character>
<simple Latin letter>::=
    <simple Latin upper case letter>
    |
    <simple Latin lower case letter>
<simple Latin upper case letter>::=
A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z
<simple Latin lower case letter>::=
a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z
<digit>::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<SQL special character>::=
    <space>
    |
    <double quote>
    |
    <percent>
    |
    <ampersand>
    |
    <quote>
    |
    <leftparen>
    |
    <rightparen>
    |
    <asterisk>
    |
    <plus sign>
    |
    <comma>
    |
    <minus sign>
    |
    <period>
    |
    <solidus>
    |
    <colon>
    |
    <semicolon>
    |
    <less than operator>
    |
    <equals operator>
    |
    <greater than operator>
    |
    <question mark>
    |
    <left bracket>
    |
    <right bracket>
    |
    <circumflex>
    |
    <underscore>
    |
    <vertical bar>
    |
    <left brace>
    |
    <right brace>
<space>::=!! (See the Syntax Rules)
<double quote>::="
<percent>::=%
<ampersand>::=&
<quote>::='
<leftparen>::=(
<rightparen>::=)
<asterisk>::=*
<plus sign>::=+
<comma>::=,
<minus sign>::=-
<period>::=.
<solidus>::=/
<colon>::=:
<semicolon>::=;
<less than operator>::=<
```

३३९ SQL 99 _ BNF

```
<equals operator>::==
<greater than operator>::=>
<question mark>::=?
<left bracket>::=[
<right bracket>::=]
<circumflex>::=^
<underscore>::=_
<vertical bar>::=|
<left brace>::={
<right brace>::=}
--hr
--h2 Literal Numbers, Strings, Dates and Times
--/h2
--p
--small
--i <SQL-client module definition> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--p
<SQL-client module definition>::=
    <module name clause>
    <language clause>
    <module authorization clause>
    [ <module path specification> ]
    [ <module transform group specification> ]
    [ <temporary table declaration>... ]
    <module contents>...
<module name clause>::=
    MODULE [ <SQL-client module name> ] [ <module character set specification>
]
<SQL-client module name>::= <identifier>
<identifier>::= <actual identifier>
<actual identifier>::= <regular identifier> | <delimited identifier>
<regular identifier>::= <identifier body>
--p
--small
--i <identifier body> modified per ISO 9075:1999/Cor.1:2000(E).
--/i
--/p
--p
--i <identifier body> also rationalized by removing curly brackets around <identifier part> because they
are unnecessary and inconsistent with other places where '...' modifies a single non-terminal.
--/i
--/small
--/small
--p
<identifier body>::= <identifier start> [ <identifier part>... ]
<identifier start>::= <initial alphabetic character> | <ideographic character>
<initial alphabetic character>::=!! (See the Syntax Rules)
<ideographic character>::=!! (See the Syntax Rules)
<identifier part>::=
    | <alphabetic character>
    | <ideographic character>
    | <decimal digit character>
    | <identifier combining character>
    | <underscore>
    | <alternate underscore>
    | <extender character>
    | <identifier ignorable character>
```

```

| <connector character>
<alphabetic character>::=!! (See the Syntax Rules)
<decimal digit character>::=!! (See the Syntax Rules)
<identifier combining character>::=!! (See the Syntax Rules)
<alternate underscore>::=!! (See the Syntax Rules)
<extender character>::=!! (See the Syntax Rules)
<identifier ignorable character>::=!! (See the Syntax Rules)
<connector character>::=!! (See the Syntax Rules)
<delimited identifier>::= <double quote> <delimited identifier body> <double quote>
<delimited identifier body>::= <delimited identifier part>...
<delimited identifier part>::= <nondoublequote character> | <doublequote symbol>
<nondoublequote character>::=!! (See the Syntax Rules)
--p
--small
--i
Note that the two successive double quote characters must have no other character (such as a space)
between them. The lexical analyzer would normally deal with this sort of issue.
--/i
--/small
--/p
<doublequote symbol>::= <double quote> <double quote>
<module character set specification>::= NAMES ARE <character set specification>
<character set specification>::=
| <standard character set name>
| <implementation-defined character set name>
| <user-defined character set name>
<standard character set name>::= <character set name>
<character set name>::=[ <schema name> <period> ] <SQL language identifier>
<schema name>::=[ <catalog name> <period> ] <unqualified schema name>
<catalog name>::= <identifier>
<unqualified schema name>::= <identifier>
<SQL language identifier>::=
| <SQL language identifier start> [ { <underscore> | <SQL language identifier part> } ... ]
| <SQL language identifier start>::= <simple Latin letter>
| <SQL language identifier part>::= <simple Latin letter> | <digit>
| <implementation-defined character set name>::= <character set name>
| <user-defined character set name>::= <character set name>
| <language clause>::= LANGUAGE <language name>
| <language name>::=ADA | C | COBOL | FORTRAN | MUMPS | PASCAL | PLI | SQL
| <module authorization clause>::=
| SCHEMA <schema name>
| AUTHORIZATION <module authorization identifier>
| SCHEMA <schema name> AUTHORIZATION <module authorization
identifier>
| <module authorization identifier>::= <authorization identifier>
| <authorization identifier>::= <role name> | <user identifier>
| <role name>::= <identifier>
| <user identifier>::= <identifier>
| <module path specification>::= <path specification>
| <path specification>::=PATH <schema name list>
| <schema name list>::= <schema name> [ { <comma> <schema name> } ... ]
| <module transform group specification>::= <transform group specification>
| <transform group specification>::=
| TRANSFORM GROUP { <single group specification> | <multiple group specification> }
| <single group specification>::= <group name>
| <group name>::= <identifier>
| <multiple group specification>::= <group specification> [ { <comma> <group specification> } ... ]
| <group specification>::= <group name> FOR TYPE <user-defined type>
| <user-defined type>::= <user-defined type name>

```

३६१ SQL 99 _BNF

```
<user-defined type name>::= <schema qualified type name>
<schema qualified type name>::=[ <schema name> <period> ] <qualified identifier>
<qualified identifier>::= <identifier>
<temporary table declaration>::=
    DECLARE LOCAL TEMPORARY TABLE <table name>
        <table element list>
        [ ON COMMIT <table commit action> ROWS ]
<table name>::= <local or schema qualified name>
<local or schema qualified name>::=[ <local or schema qualifier> <period> ] <qualified identifier>
<local or schema qualifier>::= <schema name> | MODULE
<table element list>::=
    <leftparen> <table element> [ { <comma> <table element> }... ] <rightparen>
<table element>::=
    | <column definition>
    | <table constraint definition>
    | <like clause>
    | <self-referencing column specification>
    | <column options>
<column definition>::=
    <column name>
    { <data type> | <domain name> }
    [ <reference scope check> ]
    [ <default clause> ]
    [ <column constraint definition>... ]
    [ <collate clause> ]
<column name>::= <identifier>
--hr
--h2 Data Types
--/h2
<data type>::=
    | <predefined type>
    | <row type>
    | <user-defined type>
    | <reference type>
    | <collection type>
<predefined type>::=
    <character string type> [ CHARACTER SET <character set specification> ]
    | <national character string type>
    | <binary large object string type>
    | <bit string type>
    | <numeric type>
    | <boolean type>
    | <datetime type>
    | <interval type>
<character string type>::=
    CHARACTER [ <leftparen> <length> <rightparen> ]
    | CHAR [ <leftparen> <length> <rightparen> ]
    | CHARACTER VARYING <leftparen> <length> <rightparen>
    | CHAR VARYING <leftparen> <length> <rightparen>
    | VARCHAR <leftparen> <length> <rightparen>
    | CHARACTER LARGE OBJECT [ <leftparen> <large object length>
<rightparen> ]
    | CHAR LARGE OBJECT [ <leftparen> <large object length> <rightparen> ]
    | CLOB [ <leftparen> <large object length> <rightparen> ]
<length>::= <unsigned integer>
<unsigned integer>::= <digit>...
<large object length>::= <unsigned integer> [ <multiplier> ] | <large object length token>
<multiplier>::=K | M | G
<large object length token>::= <digit>... <multiplier>
```

```

<national character string type>::=
    NATIONAL CHARACTER [ <leftparen> <length> <rightparen> ]
    |
    NATIONAL CHAR [ <leftparen> <length> <rightparen> ]
    |
    NCHAR [ <leftparen> <length> <rightparen> ]
    |
    NATIONAL CHARACTER VARYING <leftparen> <length> <rightparen>
    |
    NATIONAL CHAR VARYING <leftparen> <length> <rightparen>
    |
    NCHAR VARYING <leftparen> <length> <rightparen>
    |
    NATIONAL CHARACTER LARGE OBJECT [ <leftparen> <large object
length> <rightparen> ]
    |
    NCHAR LARGE OBJECT [ <leftparen> <large object length> <rightparen> ]
    |
    NCLOB [ <leftparen> <large object length> <rightparen> ]
<binary large object string type>::=
    BINARY LARGE OBJECT [ <leftparen> <large object length> <rightparen> ]
    |
    BLOB [ <leftparen> <large object length> <rightparen> ]
<bit string type>::=
    BIT [ <leftparen> <length> <rightparen> ]
    |
    BIT VARYING <leftparen> <length> <rightparen>
<numeric type>::= <exact numeric type> | <approximate numeric type>
<exact numeric type>::=
    NUMERIC [ <leftparen> <precision> [ <comma> <scale> ] <rightparen> ]
    |
    DECIMAL [ <leftparen> <precision> [ <comma> <scale> ] <rightparen> ]
    |
    DEC [ <leftparen> <precision> [ <comma> <scale> ] <rightparen> ]
    |
    INTEGER
    |
    INT
    |
    SMALLINT
<precision>::= <unsigned integer>
<scale>::= <unsigned integer>
<approximate numeric type>::=
    FLOAT [ <leftparen> <precision> <rightparen> ]
    |
    REAL
    |
    DOUBLE PRECISION
<boolean type>::=BOOLEAN
<datetime type>::=
    DATE
    |
    TIME [ <leftparen><time precision> <rightparen> ] [ <with or without time zone> ]
    |
    TIMESTAMP [ <leftparen> <timestamp precision> <rightparen> ] [ <with or
without time zone> ]
<time precision>::= <time fractional seconds precision>
<time fractional seconds precision>::= <unsigned integer>
<with or without time zone>::=WITH TIME ZONE | WITHOUT TIME ZONE
<timestamp precision>::= <time fractional seconds precision>
<interval type>::=INTERVAL <interval qualifier>
<interval qualifier>::= <start field> TO <end field> | <single datetime field>
<start field>::=
    <non-second primary datetime field> [ <leftparen> <interval leading field
precision> <rightparen> ]
    |
    <non-second primary datetime field>::=YEAR | MONTH | DAY | HOUR | MINUTE
    |
    <interval leading field precision>::= <unsigned integer>
    |
    <end field>::=
    <non-second primary datetime field>
    |
    SECOND [ <leftparen> <interval fractional seconds precision> <rightparen> ]
    |
    <interval fractional seconds precision>::= <unsigned integer>
    |
    <single datetime field>::=
    <non-second primary datetime field> [ <leftparen> <interval leading field
precision> <rightparen> ]
    |
    SECOND [ <leftparen> <interval leading field precision> [ <comma> <interval
fractional seconds precision> ] <rightparen> ]
<row type>::=ROW <row type body>
<row type body>::= <leftparen> <field definition> [ { <comma> <field definition> }... ] <rightparen>

```

३६३ SQL 99 _BNF

```
<field definition>::= <field name> <data type> [ <reference scope check> ] [ <collate clause> ]
<field name>::= <identifier>
<reference scope check>::=
    REFERENCES ARE [ NOT ] CHECKED [ ON DELETE <reference scope check
action> ]
<reference scope check action>::= <referential action>
<referential action>::=
    CASCADE
    |
    SET NULL
    |
    SET DEFAULT
    |
    RESTRICT
    |
    NO ACTION
<collate clause>::=COLLATE <collation name>
<collation name>::= <schema qualified name>
<schema qualified name>::=[ <schema name> <period> ] <qualified identifier>
<reference type>::=REF <leftparen> <referenced type> <rightparen> [ <scope clause> ]
<referenced type>::= <user-defined type>
<scope clause>::=SCOPE <table name>
<collection type>::= <data type> <array specification>
<array specification>::=
    <collection type constructor> <left bracket or trigraph> <unsigned integer>
<right bracket or trigraph>
<collection type constructor>::=ARRAY
<left bracket or trigraph>::= <left bracket> | <left bracket trigraph>
--p
--small
--i
The trigraphs are strictly sequences of characters, not sequences of tokens. There may not be any spaces
between the characters. Normally, the lexical analyzer would return the trigraphs as a simple symbol.
--/i
--/small
--/p
<left bracket trigraph>::= <question mark> <question mark> <leftparen>
<right bracket or trigraph>::= <right bracket> | <right bracket trigraph>
<right bracket trigraph>::= <question mark> <question mark> <rightparen>
<domain name>::= <schema qualified name>
<default clause>::=DEFAULT <default option>
<default option>::=
    <literal>
    |
    <datetime value function>
    |
    USER
    |
    CURRENT_USER
    |
    CURRENT_ROLE
    |
    SESSION_USER
    |
    SYSTEM_USER
    |
    CURRENT_PATH
    |
    <implicitly typed value specification>
--hr
--h2 Literals
--/h2
<literal>::= <signed numeric literal> | <general literal>
<signed numeric literal>::=[ <sign> ] <unsigned numeric literal>
<sign>::= <plus sign> | <minus sign>
<unsigned numeric literal>::= <exact numeric literal> | <approximate numeric literal>
<exact numeric literal>::=
    <unsigned integer> [ <period> [ <unsigned integer> ] ]
    |
    <period> <unsigned integer>
<approximate numeric literal>::= <mantissa> E <exponent>
<mantissa>::= <exact numeric literal>
```

```

<exponent>::= <signed integer>
<signed integer>::=[ <sign> ] <unsigned integer>
<general literal>::=
    | <character string literal>
    | <national character string literal>
    | <bit string literal>
    | <hex string literal>
    | <binary string literal>
    | <datetime literal>
    | <interval literal>
    | <boolean literal>
<character string literal>::=
    [ <introducer> <character set specification> ]
    <quote> [ <character representation>... ] <quote>
    [ { <separator> <quote> [ <character representation>... ] <quote> }... ]
<introducer>::= <underscore>
<character representation>::= <nonquote character> | <quote symbol>
<nonquote character>::=!! (See the Syntax Rules.)
--p
--small
--i
The <quote symbol> rule consists of two immediately adjacent <quote> marks with no spaces. As usual,
this would be best handled in the lexical analyzer, not in the grammar.
--/i
--/small
--/p
<quote symbol>::= <quote> <quote>
<separator>::={ <comment> | <white space> }...
<comment>::= <simple comment> | <bracketed comment>
<simple comment>::= <simple comment introducer> [ <comment character>... ] <newline>
<simple comment introducer>::= <minus sign> <minus sign> [ <minus sign>... ]
<comment character>::= <nonquote character> | <quote>
<newline>::=!! (See the Syntax Rules)
--p
--small
--i
The <bracketed comment> rule included '!! (See the Syntax Rules)'. This probably says something about
the <slash> <asterisk> and <asterisk> <slash> needing to be adjacent characters rather than adjacent
tokens.
--/i
--/small
--/p
<bracketed comment>::=
    <bracketed comment introducer> <bracketed comment contents> <bracketed
comment terminator>
<bracketed comment introducer>::= <slash> <asterisk>
<bracketed comment contents>::=[ { <comment character> | <separator> }... ]
<bracketed comment terminator>::= <asterisk> <slash>
<white space>::=!! (See the Syntax Rules)
<national character string literal>::=
    N <quote> [ <character representation>... ] <quote>
    [ { <separator> <quote> [ <character representation>... ] <quote> }... ]
<bit string literal>::=
    B <quote> [ <bit>... ] <quote>
    [ { <separator> <quote> [ <bit>... ] <quote> }... ]
<bit>::=0 | 1
<hex string literal>::=
    X <quote> [ <hexit>... ] <quote>
    [ { <separator> <quote> [ <hexit>... ] <quote> }... ]

```

३१० SQL 99 _BNF

```
<hexit>::=<digit>|A|B|C|D|E|F|a|b|c|d|e|f
<binary string literal>::=
    X<quote>[ { <hexit> <hexit> }... ]<quote>
    [ { <separator> <quote>[ { <hexit> <hexit> }... ]<quote> }... ]
<datetime literal>::=<date literal>|<time literal>|<timestamp literal>
<date literal>::=DATE<date string>
<date string>::=<quote><unquoted date string><quote>
<unquoted date string>::=<date value>
<date value>::=<years value><minus sign><months value><minus sign><days value>
<years value>::=<datetime value>
<datetime value>::=<unsigned integer>
<months value>::=<datetime value>
<days value>::=<datetime value>
<time literal>::=TIME<time string>
<time string>::=<quote><unquoted time string><quote>
<unquoted time string>::=<time value>[<time zone interval>]
<time value>::=<hours value><colon><minutes value><colon><seconds value>
<hours value>::=<datetime value>
<minutes value>::=<datetime value>
<seconds value>::=<seconds integer value>[<period>[<seconds fraction>]]
<seconds integer value>::=<unsigned integer>
<seconds fraction>::=<unsigned integer>
<time zone interval>::=<sign><hours value><colon><minutes value>
<timestamp literal>::=TIMESTAMP<timestamp string>
<timestamp string>::=<quote><unquoted timestamp string><quote>
<unquoted timestamp string>::=<unquoted date string><space><unquoted time string>
<interval literal>::=INTERVAL[<sign>]<interval string><interval qualifier>
<interval string>::=<quote><unquoted interval string><quote>
<unquoted interval string>::=[<sign>]{<year-month literal>|<day-time literal>}
<year-month literal>::=<years value>|[<years value><minus sign>]<months value>
<day-time literal>::=<day-time interval>|<time interval>
<day-time interval>::=
    <days value>[<space><hours value>[<colon><minutes value>[<colon>
<seconds value>]]]
<time interval>::=
    <hours value>[<colon><minutes value>[<colon><seconds value>]]
    |
    <minutes value>[<colon><seconds value>]
    |
    <seconds value>
<boolean literal>::=TRUE|FALSE|UNKNOWN
<datetime value function>::=
    <current date value function>
    |
    <current time value function>
    |
    <current timestamp value function>
    |
    <current local time value function>
    |
    <current local timestamp value function>
<current date value function>::=CURRENT_DATE
<current time value function>::=
    CURRENT_TIME[<leftparen><time precision><rightparen>]
<current timestamp value function>::=
    CURRENT_TIMESTAMP[<leftparen><timestamp precision><rightparen>]
<current local time value function>::=
    LOCALTIME[<leftparen><time precision><rightparen>]
<current local timestamp value function>::=
    LOCALTIMESTAMP[<leftparen><timestamp precision><rightparen>]
<implicitly typed value specification>::=<null specification>|<empty specification>
<null specification>::=NULL
<empty specification>::=ARRAY<left bracket or trigraph><right bracket or trigraph>
--hr
--h2 Constraints
```

```

--/h2
<column constraint definition>::=
    [ <constraint name definition> ] <column constraint> [ <constraint characteristics> ]
<constraint name definition>::=CONSTRAINT <constraint name>
<constraint name>::= <schema qualified name>
<column constraint>::=
    NOT NULL
    |
    <unique specification>
    |
    <references specification>
    |
    <check constraint definition>
<unique specification>::=UNIQUE | PRIMARY KEY
<references specification>::=
    REFERENCES <referenced table and columns>
    [ MATCH <match type> ] [ <referential triggered action> ]
<referenced table and columns>::=
    <table name> [ <leftparen> <reference column list> <rightparen> ]
<reference column list>::= <column name list>
<column name list>::= <column name> [ { <comma> <column name> } ... ]
<match type>::=FULL | PARTIAL | SIMPLE
<referential triggered action>::=
    <update rule> [ <delete rule> ]
    |
    <delete rule> [ <update rule> ]
<update rule>::=ON UPDATE <referential action>
<delete rule>::=ON DELETE <referential action>
<check constraint definition>::=CHECK <leftparen> <search condition> <rightparen>
--hr
--h2 Search Condition
--/h2
<search condition>::= <boolean value expression>
<boolean value expression>::=
    <boolean term>
    |
    <boolean value expression> OR <boolean term>
<boolean term>::=
    <boolean factor>
    |
    <boolean term> AND <boolean factor>
<boolean factor>::=[ NOT ] <boolean test>
<boolean test>::= <boolean primary> [ IS [ NOT ] <truth value> ]
<boolean primary>::=
    <predicate>
    |
    <parenthesized boolean value expression>
    |
    <nonparenthesized value expression primary>
<predicate>::=
    <comparison predicate>
    |
    <between predicate>
    |
    <in predicate>
    |
    <like predicate>
    |
    <null predicate>
    |
    <quantified comparison predicate>
    |
    <exists predicate>
    |
    <unique predicate>
    |
    <match predicate>
    |
    <overlaps predicate>
    |
    <similar predicate>
    |
    <distinct predicate>
    |
    <type predicate>
<comparison predicate>::= <row value expression> <comp op> <row value expression>
<row value expression>::= <row value special case> | <row value constructor>
<row value special case>::= <value specification> | <value expression>
<value specification>::= <literal> | <general value specification>

```

३६४ SQL 99 _BNF

```

<general value specification>::=
    | <host parameter specification>
    | <SQL parameter reference>
    | <SQL variable reference>
    | <dynamic parameter specification>
    | <embedded variable specification>
    | CURRENT_DEFAULT_TRANSFORM_GROUP
    | CURRENT_PATH
    | CURRENT_ROLE
    | CURRENT_TRANSFORM_GROUP_FOR_TYPE <user-defined type>
    | CURRENT_USER
    | SESSION_USER
    | SYSTEM_USER
    | USER
    | VALUE

<host parameter specification>::= <host parameter name> [ <indicator parameter> ]
<host parameter name>::= <colon> <identifier>
<indicator parameter>::=[ INDICATOR ] <host parameter name>
<SQL parameter reference>::= <basic identifier chain>
<basic identifier chain>::= <identifier chain>
<identifier chain>::= <identifier> [ { <period> <identifier> } ... ]
<value expression>::=
    | <numeric value expression>
    | <string value expression>
    | <datetime value expression>
    | <interval value expression>
    | <boolean value expression>
    | <user-defined type value expression>
    | <row value expression>
    | <reference value expression>
    | <collection value expression>

<numeric value expression>::=
    | <term>
    | <numeric value expression> <plus sign> <term>
    | <numeric value expression> <minus sign> <term>

<term>::=
    | <factor>
    | <term> <asterisk> <factor>
    | <term> <solidus> <factor>

<factor>::=[ <sign> ] <numeric primary>
<numeric primary>::=
    | <value expression primary>
    | <numeric value function>

<value expression primary>::=
    | <parenthesized value expression>
    | <nonparenthesized value expression primary>

<parenthesized value expression>::= <leftparen> <value expression> <rightparen>
<nonparenthesized value expression primary>::=
    | <unsigned value specification>
    | <column reference>
    | <set function specification>
    | <scalar subquery>
    | <case expression>
    | <cast specification>
    | <subtype treatment>
    | <attribute or method reference>
    | <reference resolution>
    | <collection value constructor>
    | <routine invocation>

```

```

| <field reference>
| <element reference>
| <method invocation>
| <static method invocation>
| <new specification>
<unsigned value specification>::= <unsigned literal> | <general value specification>
<unsigned literal>::= <unsigned numeric literal> | <general literal>
<column reference>::=
| <basic identifier chain>
| MODULE <period> <qualified identifier> <period> <column name>
<set function specification>::=
| COUNT <leftparen> <asterisk> <rightparen>
| <general set function>
| <grouping operation>
<general set function>::=
| <set function type> <leftparen> [ <set quantifier> ] <value expression>
<rightparen>
<set function type>::= <computational operation>
<computational operation>::= AVG | MAX | MIN | SUM | EVERY | ANY | SOME | COUNT
<set quantifier>::= DISTINCT | ALL
<grouping operation>::= GROUPING <leftparen> <column reference> <rightparen>
--hr
--h2 Queries
--/h2
<scalar subquery>::= <subquery>
<subquery>::= <leftparen> <query expression> <rightparen>
<query expression>::= [ <with clause> ] <query expression body>
<with clause>::= WITH [ RECURSIVE ] <with list>
<with list>::= <with list element> [ { <comma> <with list element> } ... ]
<with list element>::=
| <query name>
| [ <leftparen> <with column list> <rightparen> ]
| AS <leftparen> <query expression> <rightparen>
| [ <search or cycle clause> ]
<query name>::= <identifier>
<with column list>::= <column name list>
<search or cycle clause>::=
| <search clause>
| <cycle clause>
| <search clause> <cycle clause>
<search clause>::=
| SEARCH <recursive search order> SET <sequence column>
<recursive search order>::=
| DEPTH FIRST BY <sort specification list>
| BREADTH FIRST BY <sort specification list>
<sort specification list>::= <sort specification> [ { <comma> <sort specification> } ... ]
--p
--small
--i <sort specification> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
<sort specification>::= <sort key> [ <ordering specification> ]
<sort key>::= <value expression>
<ordering specification>::= ASC | DESC
<sequence column>::= <column name>
<cycle clause>::=
| CYCLE <cycle column list>
| SET <cycle mark column> TO <cycle mark value>

```

३६९ SQL 99 _BNF

```

        DEFAULT <non-cycle mark value>
        USING <path column>
<cycle column list>::= <cycle column> [ { <comma> <cycle column> }... ]
<cycle column>::= <column name>
<cycle mark column>::= <column name>
<cycle mark value>::= <value expression>
<non-cycle mark value>::= <value expression>
<path column>::= <column name>
<query expression body>::= <non-join query expression> | <joined table>
<non-join query expression>::=
    <non-join query term>
    |
    <query expression body> UNION [ ALL | DISTINCT ] [ <corresponding spec> ]
<query term>
    |
    <query expression body> EXCEPT [ ALL | DISTINCT ] [ <corresponding spec> ]
] <query term>
<non-join query term>::=
    <non-join query primary>
    |
    <query term> INTERSECT [ ALL | DISTINCT ] [ <corresponding spec> ]
<query primary>
<non-join query primary>::=
    <simple table>
    |
    <leftparen> <non-join query expression> <rightparen>
<simple table>::= <query specification> | <table value constructor> | <explicit table>
<query specification>::= SELECT [ <set quantifier> ] <select list> <table expression>
<select list>::= <asterisk> | <select sublist> [ { <comma> <select sublist> }... ]
<select sublist>::= <derived column> | <qualified asterisk>
<derived column>::= <value expression> [ <as clause> ]
<as clause>::= [ AS ] <column name>
<qualified asterisk>::=
    <asterisked identifier chain> <period> <asterisk>
    |
    <all fields reference>
<asterisked identifier chain>::= <asterisked identifier> [ { <period> <asterisked identifier> }... ]
<asterisked identifier>::= <identifier>
<all fields reference>::= <value expression primary> <period> <asterisk>
<table expression>::= <from clause> [ <where clause> ] [ <group by clause> ] [ <having clause> ]
<from clause>::= FROM <table reference list>
<table reference list>::= <table reference> [ { <comma> <table reference> }... ]
<table reference>::= <table primary> | <joined table>
<table primary>::=
    <table or query name> [ [ AS ] <correlation name> [ <leftparen> <derived
column list> <rightparen> ] ]
    |
    <derived table> [ AS ] <correlation name> [ <leftparen> <derived column list>
<rightparen> ]
    |
    <lateral derived table> [ AS ] <correlation name> [ <leftparen> <derived column
list> <rightparen> ]
    |
    <collection derived table> [ AS ] <correlation name> [ <leftparen> <derived
column list> <rightparen> ]
    |
    <only spec> [ [ AS ] <correlation name> [ <leftparen> <derived column list>
<rightparen> ] ]
    |
    <leftparen> <joined table> <rightparen>
<table or query name>::= <table name> | <query name>
<correlation name>::= <identifier>
<derived column list>::= <column name list>
<derived table>::= <table subquery>
<table subquery>::= <subquery>
<lateral derived table>::= LATERAL <leftparen> <query expression> <rightparen>
<collection derived table>::=
    UNNEST <leftparen> <collection value expression> <rightparen> [ WITH
ORDINALITY ]

```

```

<collection value expression>::= <value expression primary>
<only spec>::= ONLY <leftparen> <table or query name> <rightparen>
<joined table>::= <cross join> | <qualified join> | <natural join> | <union join>
<cross join>::= <table reference> CROSS JOIN <table primary>
<qualified join>::= <table reference> [ <join type> ] JOIN <table reference> <join specification>
<join type>::= INNER | <outer join type> [ OUTER ]
<outer join type>::= LEFT | RIGHT | FULL
<join specification>::= <join condition> | <named columns join>
<join condition>::= ON <search condition>
<named columns join>::= USING <leftparen> <join column list> <rightparen>
<join column list>::= <column name list>
<natural join>::= <table reference> NATURAL [ <join type> ] JOIN <table primary>
<union join>::= <table reference> UNION JOIN <table primary>
<where clause>::= WHERE <search condition>
--p
--small
--i Rules from <group by clause> to <grouping set> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
<group by clause>::= GROUP BY <grouping element list>
<grouping element list>::= <grouping element> [ { <comma> <grouping element> } ... ]
<grouping element>::=
    | <ordinary grouping set>
    | <rollup list>
    | <cube list>
    | <grouping sets specification>
    | <grand total>
<grouping column reference>::= <column reference> [ <collate clause> ]
<rollup list>::= ROLLUP <leftparen> <grouping column reference list> <rightparen>
<grouping column reference list>::=
    <grouping column reference> [ { <comma> <grouping column reference> } ... ]
<cube list>::= CUBE <leftparen> <grouping column reference list> <rightparen>
<grouping sets specification>::= GROUPING SETS <leftparen> <grouping set list> <rightparen>
<grouping set list>::= <grouping set> [ { <comma> <grouping set> } ... ]
<grouping set>::=
    | <ordinary grouping set>
    | <rollup list>
    | <cube list>
    | <grouping sets specification>
    | <grand total>
<ordinary grouping set>::=
    | <grouping column reference>
    | <leftparen> <grouping column reference list> <rightparen>
<grand total>::= <leftparen> <rightparen>
<concatenated grouping>::= <grouping set> <comma> <grouping set list>
<having clause>::= HAVING <search condition>
<table value constructor>::= VALUES <row value expression list>
<row value expression list>::= <row value expression> [ { <comma> <row value expression> } ... ]
<explicit table>::= TABLE <table name>
--hr
--h2 Query expression components
--/h2
<query term>::= <non-join query term> | <joined table>
<corresponding spec>::=
    CORRESPONDING [ BY <leftparen> <corresponding column list>
<rightparen> ]
<corresponding column list>::= <column name list>
<query primary>::= <non-join query primary> | <joined table>

```

३०१ SQL 99 _BNF

```

<case expression>::= <case abbreviation> | <case specification>
<case abbreviation>::=
    NULLIF <leftparen> <value expression> <comma> <value expression>
    <rightparen>
    |
    COALESCE <leftparen> <value expression> { <comma> <value expression>
    } ... <rightparen>
<case specification>::= <simple case> | <searched case>
<simple case>::= CASE <case operand> <simple when clause> ... [ <else clause> ] END
<case operand>::= <value expression>
<simple when clause>::= WHEN <when operand> THEN <result>
<when operand>::= <value expression>
<result>::= <result expression> | NULL
<result expression>::= <value expression>
<else clause>::= ELSE <result>
<searched case>::= CASE <searched when clause> ... [ <else clause> ] END
<searched when clause>::= WHEN <search condition> THEN <result>
<cast specification>::= CAST <leftparen> <cast operand> AS <cast target> <rightparen>
<cast operand>::= <value expression> | <implicitly typed value specification>
<cast target>::= <domain name> | <data type>
--p
--small
--i <subtype treatment> to <target subtype> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
<subtype treatment>::=
    TREAT <leftparen> <subtype operand> AS <target subtype> <rightparen>
<subtype operand>::= <value expression>
<target subtype>::= <user-defined type>
<attribute or method reference>::=
    <value expression primary> <dereference operator> <qualified identifier>
    [ <SQL argument list> ]
<dereference operator>::= <right arrow>
<right arrow>::= ->
<SQL argument list>::=
    <leftparen> [<SQL argument> [ { <comma> <SQL argument> } ... ] ] <rightparen>
<SQL argument>::= <value expression> | <generalized expression> | <target specification>
<generalized expression>::= <value expression> AS <user-defined type>
<target specification>::=
    <host parameter specification>
    |
    <SQL parameter reference>
    |
    <column reference>
    |
    <SQL variable reference>
    |
    <dynamic parameter specification>
    |
    <embedded variable specification>
<reference resolution>::= Deref <leftparen> <reference value expression> <rightparen>
<reference value expression>::= <value expression primary>
<collection value constructor>::= <array value expression>
<array value expression>::= <array value constructor> | <array concatenation> | <value expression
primary>
<array value constructor>::= <array value list constructor>
<array value list constructor>::=
    ARRAY <left bracket or trigraph> <array element list> <right bracket or trigraph>
<array element list>::= <array element> [ { <comma> <array element> } ... ]
<array element>::= <value expression>
<array concatenation>::=
    <array value expression 1> <concatenation operator> <array value expression 2>
<array value expression 1>::= <array value expression>
<concatenation operator>::= ||

```

```

<array value expression 2>::= <array value expression>
<routine invocation>::= <routine name> <SQL argument list>
<routine name>::=[ <schema name> <period> ] <qualified identifier>
<field reference>::= <value expression primary> <period> <field name>
<element reference>::=
    <array value expression> <left bracket or trigraph> <numeric value expression>
<right bracket or trigraph>
<method invocation>::= <direct invocation> | <generalized invocation>
<direct invocation>::=
    <value expression primary> <period> <method name> [ <SQL argument list> ]
<method name>::= <identifier>
<generalized invocation>::=
    <leftparen> <value expression primary>
    AS <data type> <rightparen> <period> <method name>
    [ <SQL argument list> ]

--p
--small
--i It is not remotely clear why this was needed in this grammar.
--/i
--i <constructor method selection> added per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
<constructor method selection>::= <routine invocation>
<static method invocation>::=
    <user-defined type> <double colon> <method name> [ <SQL argument list> ]

--p
--small
--i
Note that <double colon> must be a pair of characters with no intervening space, not a pair of colon
symbols separated by arbitrary white space. Normally, the lexical analyzer would return <double colon>
as a symbol.
--/i
--/small
--/p
<double colon>::= <colon> <colon>
<new specification>::=NEW <routine invocation>
<numeric value function>::=
    | <position expression>
    | <extract expression>
    | <length expression>
    | <cardinality expression>
    | <absolute value expression>
    | <modulus expression>
<position expression>::= <string position expression> | <blob position expression>
<string position expression>::=
    POSITION <leftparen> <string value expression> IN <string value expression>
<rightparen>
<string value expression>::= <character value expression> | <bit value expression> | <blob value
expression>
<character value expression>::= <concatenation> | <character factor>
<concatenation>::= <character value expression> <concatenation operator> <character factor>
<character factor>::= <character primary> [ <collate clause> ]
<character primary>::= <value expression primary> | <string value function>
<string value function>::= <character value function> | <blob value function> | <bit value function>
<character value function>::=
    <character substring function>
    | <regular expression substring function>
    | <fold>

```

२०२ SQL 99 _BNF

```

|          <form-of-use conversion>
|          <character translation>
|          <trim function>
|          <character overlay function>
|          <specific type method>
<character substring function>::=
    SUBSTRING <leftparen> <character value expression> FROM <start position>
    [ FOR <string length> ] <rightparen>
<start position>::= <numeric value expression>
<string length>::= <numeric value expression>
--p
--small
--i <regular expression substring function> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
<regular expression substring function>::=
    SUBSTRING <leftparen> <character value expression> SIMILAR
    <character value expression> ESCAPE <escape character> <rightparen>
<escape character>::= <character value expression>
<fold>::= { UPPER | LOWER } <leftparen> <character value expression> <rightparen>
<form-of-use conversion>::=
    CONVERT <leftparen> <character value expression>
    USING <form-of-use conversion name> <rightparen>
<form-of-use conversion name>::= <schema qualified name>
<character translation>::=
    TRANSLATE <leftparen> <character value expression>
    USING <translation name> <rightparen>
<translation name>::= <schema qualified name>
<trim function>::= TRIM <leftparen> <trim operands> <rightparen>
<trim operands>::= [ [ <trim specification> ] [ <trim character> ] FROM ] <trim source>
<trim specification>::= LEADING | TRAILING | BOTH
<trim character>::= <character value expression>
<trim source>::= <character value expression>
<character overlay function>::=
    OVERLAY <leftparen> <character value expression> PLACING <character
value expression>
    FROM <start position> [ FOR <string length> ] <rightparen>
<specific type method>::= <user-defined type value expression> <period> SPECIFICTYPE
<user-defined type value expression>::= <value expression primary>
<blob value function>::= <blob substring function> | <blob trim function> | <blob overlay function>
<blob substring function>::=
    SUBSTRING <leftparen> <blob value expression> FROM <start position>
    [ FOR <string length> ] <rightparen>
<blob value expression>::= <blob concatenation> | <blob factor>
<blob concatenation>::= <blob value expression> <concatenation operator> <blob factor>
<blob factor>::= <blob primary>
<blob primary>::= <value expression primary> | <string value function>
<blob trim function>::= TRIM <leftparen> <blob trim operands> <rightparen>
<blob trim operands>::= [ [ <trim specification> ] [ <trim octet> ] FROM ] <blob trim source>
<trim octet>::= <blob value expression>
<blob trim source>::= <blob value expression>
<blob overlay function>::=
    OVERLAY <leftparen> <blob value expression> PLACING <blob value
expression>
    FROM <start position> [ FOR <string length> ] <rightparen>
<bit value function>::= <bit substring function>
<bit substring function>::=
    SUBSTRING <leftparen> <bit value expression> FROM <start position>

```

```

[ FOR <string length> ] <rightparen>
<bit value expression>::= <bit concatenation> | <bit factor>
<bit concatenation>::= <bit value expression> <concatenation operator> <bit factor>
<bit factor>::= <bit primary>
<bit primary>::= <value expression primary> | <string value function>
<blob position expression>::=
    POSITION <leftparen> <blob value expression> IN <blob value expression>
<rightparen>
<extract expression>::=
    EXTRACT <leftparen> <extract field> FROM <extract source> <rightparen>
<extract field>::= <primary datetime field> | <time zone field>
<primary datetime field>::= <non-second primary datetime field> | SECOND
<time zone field>::= TIMEZONE_HOUR | TIMEZONE_MINUTE
<extract source>::= <datetime value expression> | <interval value expression>
<datetime value expression>::=
    <datetime term>
    |
    <interval value expression> <plus sign> <datetime term>
    |
    <datetime value expression> <plus sign> <interval term>
    |
    <datetime value expression> <minus sign> <interval term>
<interval term>::=
    <interval factor>
    |
    <interval term 2> <asterisk> <factor>
    |
    <interval term 2> <solidus> <factor>
    |
    <term> <asterisk> <interval factor>
<interval factor>::=[ <sign> ] <interval primary>
<interval primary>::= <value expression primary> | <interval value function>
<interval value function>::= <interval absolute value function>
<interval absolute value function>::=ABS <leftparen> <interval value expression> <rightparen>
<interval value expression>::=
    <interval term>
    |
    <interval value expression 1> <plus sign> <interval term 1>
    |
    <interval value expression 1> <minus sign> <interval term 1>
    |
    <leftparen> <datetime value expression> <minus sign>
    <datetime term> <rightparen> <interval qualifier>
<interval value expression 1>::= <interval value expression>
<interval term 1>::= <interval term>
<datetime term>::= <datetime factor>
<datetime factor>::= <datetime primary> [ <time zone> ]
<datetime primary>::= <value expression primary> | <datetime value function>
<time zone>::=AT <time zone specifier>
<time zone specifier>::=LOCAL | TIME_ZONE <interval primary>
<interval term 2>::= <interval term>
<length expression>::= <char length expression> | <octet length expression> | <bit length expression>
<char length expression>::=
    { CHAR_LENGTH | CHARACTER_LENGTH } <leftparen> <string value
expression> <rightparen>
<octet length expression>::=
    OCTET_LENGTH <leftparen> <string value expression> <rightparen>
<bit length expression>::=
    BIT_LENGTH <leftparen> <string value expression> <rightparen>
<cardinality expression>::=
    CARDINALITY <leftparen> <collection value expression> <rightparen>
<absolute value expression>::=
    ABS <leftparen> <numeric value expression> <rightparen>
<modulus expression>::=
    MOD <leftparen> <numeric value expression dividend> <comma> <numeric
value expression divisor> <rightparen>
<numeric value expression dividend>::= <numeric value expression>
<numeric value expression divisor>::= <numeric value expression>

```

३०० SQL 99 _BNF

```

<row value constructor>::=
    <row value constructor element>
    | [ ROW ] <leftparen> <row value constructor element list> <rightparen>
    | <row subquery>
<row value constructor element>::= <value expression>
<row value constructor element list>::=
    <row value constructor element>[ {<comma><row value constructor
element>}... ]
<row subquery>::= <subquery>
<comp op>::=
    <equals operator>
    | <not equals operator>
    | <less than operator>
    | <greater than operator>
    | <less than or equals operator>
    | <greater than or equals operator>
--p
--small
--i
The <not equals> , <less than or equals operator> and <greater than or equals operator> should be
handled by the lexical analyzer as token symbols, not by the grammar. As usual, spaces are not allowed
between the two characters.
--/i
--/small
--/p
<not equals operator>::= <less than operator> <greater than operator>
<less than or equals operator>::= <less than operator> <equals operator>
<greater than or equals operator>::= <greater than operator> <equals operator>
<between predicate>::=
    <row value expression> [ NOT ] BETWEEN [ ASYMMETRIC | SYMMETRIC
]
    <row value expression> AND <row value expression>
<in predicate>::= <row value expression> [ NOT ] IN <in predicate value>
<in predicate value>::= <table subquery> | <leftparen> <in value list> <rightparen>
--p
--small
--i Previously, the expression in curly braces was not in square brackets.
--/i
--i Consequently, every <in value list> had to have at least two items in it.
--/i
--/small
--/p
<in value list>::= <row value expression> [ { <comma> <row value expression> }... ]
<like predicate>::= <character like predicate> | <octet like predicate>
<character like predicate>::=
    <character match value> [ NOT ] LIKE <character pattern> [ ESCAPE <escape
character> ]
<character match value>::= <character value expression>
<character pattern>::= <character value expression>
<octet like predicate>::=
    <octet match value> [ NOT ] LIKE <octet pattern> [ ESCAPE <escape octet> ]
<octet match value>::= <blob value expression>
<octet pattern>::= <blob value expression>
<escape octet>::= <blob value expression>
<null predicate>::= <row value expression> IS [ NOT ] NULL
<quantified comparison predicate>::= <row value expression> <comp op> <quantifier> <table
subquery>
<quantifier>::= <all> | <some>
<all>::= ALL

```

```

<some>::=SOME | ANY
<exists predicate>::=EXISTS <table subquery>
<unique predicate>::=UNIQUE <table subquery>
<match predicate>::=
    <row value expression> MATCH [ UNIQUE ] [ SIMPLE | PARTIAL | FULL ]
    <table subquery>
<overlaps predicate>::= <row value expression 1> OVERLAPS <row value expression 2>
<row value expression 1>::= <row value expression>
<row value expression 2>::= <row value expression>
<similar predicate>::=
    <character match value> [ NOT ] SIMILAR TO <similar pattern> [ ESCAPE
<escape character> ]
<similar pattern>::= <character value expression>
--hr
--h2 Regular Expressions for SIMILAR TO
--/h2
--p
These regular expressions are not referenced anywhere else in the document, but define the structure that
the <character value expression> used in <similar pattern> must have. Structurally, these regular
expressions are similar to 'egrep' expressions, except they use underscore in place of dot, and percent is
equivalent to dot star in 'egrep'. The other omission is the use of caret (aka circumflex) to mark the start
of the matched text and dollar to mark the end of the matched text.
--/p
<regular expression>::=
    <regular term>
    |
    <regular expression> <vertical bar> <regular term>
<regular term>::=
    <regular factor>
    |
    <regular term> <regular factor>
<regular factor>::=
    <regular primary>
    |
    <regular primary> <asterisk>
    |
    <regular primary> <plus sign>
<regular primary>::=
    <character specifier>
    |
    <percent>
    |
    <regular character set>
    |
    <leftparen> <regular expression> <rightparen>
<character specifier>::= <non-escaped character> | <escaped character>
<non-escaped character>::=!! (See the Syntax Rules)
<escaped character>::=!! (See the Syntax Rules)
<regular character set>::=
    <underscore>
    |
    <left bracket> <character enumeration>... <right bracket>
    |
    <left bracket> <circumflex> <character enumeration>... <right bracket>
    |
    <left bracket> <colon> <regular character set identifier> <colon> <right
bracket>
<character enumeration>::=
    <character specifier>
    |
    <character specifier> <minus sign> <character specifier>
<regular character set identifier>::= <identifier>
--hr
<distinct predicate>::=
    <row value expression 3> IS DISTINCT FROM <row value expression 4>
<row value expression 3>::= <row value expression>
<row value expression 4>::= <row value expression>
<type predicate>::=
    <user-defined type value expression> IS [ NOT ] OF <leftparen> <type list>
<rightparen>

```

३०४ SQL 99 _BNF

```
<type list>::=
    <user-defined type specification> [ { <comma> <user-defined type specification> } ...
]
<user-defined type specification>::=
    <inclusive user-defined type specification>
    |
    <exclusive user-defined type specification>
<inclusive user-defined type specification>::= <user-defined type>
<exclusive user-defined type specification>::= ONLY <user-defined type>
<parenthesized boolean value expression>::= <leftparen> <boolean value expression> <rightparen>
<truth value>::= TRUE | FALSE | UNKNOWN
--hr
--h2 More about constraints
--/h2
<constraint characteristics>::=
    <constraint check time> [ [ NOT ] DEFERRABLE ]
    |
    [ NOT ] DEFERRABLE [ <constraint check time> ]
<constraint check time>::= INITIALLY DEFERRED | INITIALLY IMMEDIATE
<table constraint definition>::=
    [ <constraint name definition> ] <table constraint> [ <constraint characteristics> ]
<table constraint>::= <unique constraint definition> | <referential constraint definition> | <check
constraint definition>
--p
--small
--i
The standard documents UNIQUE ( VALUE ) but there is no explanation of why that is different from
the UNIQUE <leftparen> VALUE <rightparen> used here.
--/i
--/small
--/p
<unique constraint definition>::=
    <unique specification> <leftparen> <unique column list> <rightparen>
    |
    UNIQUE <leftparen> VALUE <rightparen>
<unique column list>::= <column name list>
<referential constraint definition>::=
    FOREIGN KEY <leftparen> <referencing columns> <rightparen> <references
specification>
<referencing columns>::= <reference column list>
<like clause>::= LIKE <table name>
<self-referencing column specification>::=
    REF IS <self-referencing column name> <reference generation>
<self-referencing column name>::= <column name>
<reference generation>::= SYSTEM GENERATED | USER GENERATED | DERIVED
<column options>::= <column name> WITH OPTIONS <column option list>
<column option list>::=
    [ <scope clause> ] [ <default clause> ] [ <column constraint definition>... ] [
<collate clause> ]
<table commit action>::= PRESERVE | DELETE
--hr
--h2 Module contents
--/h2
<module contents>::=
    <declare cursor>
    |
    <externally-invoked procedure>
    |
    <dynamic declare cursor>
<declare cursor>::=
    DECLARE <cursor name> [ <cursor sensitivity> ] [ <cursor scrollability> ]
CURSOR
    [ <cursor holdability> ] [ <cursor returnability> ] FOR <cursor specification>
<cursor name>::= <local qualified name>
```

```

<local qualified name>::=[ <local qualifier> <period> ] <qualified identifier>
<local qualifier>::=MODULE
<cursor sensitivity>::=SENSITIVE | INSENSITIVE | ASENSITIVE
<cursor scrollability>::=SCROLL | NO SCROLL
<cursor holdability>::=WITH HOLD | WITHOUT HOLD
<cursor returnability>::=WITH RETURN | WITHOUT RETURN
<cursor specification>::=<query expression> [ <order by clause> ] [ <updatability clause> ]
<order by clause>::=ORDER BY <sort specification list>
<updatability clause>::=FOR { READ ONLY | UPDATE [ OF <column name list> ] }
--hr
--h2 SQL Procedures
--/h2
<externally-invoked procedure>::=
    PROCEDURE <procedure name>
        <host parameter declaration setup> <semicolon>
        <SQL procedure statement> <semicolon>
<procedure name>::=<identifier>
--p
--small
--i <host parameter declaration setup> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--p
<host parameter declaration setup>::=<host parameter declaration list>
<host parameter declaration list>::=
    <leftparen> <host parameter declaration>
    [ { <comma> <host parameter declaration> }... ] <rightparen>
<host parameter declaration>::=
    <host parameter name> <host parameter data type>
    |
    <status parameter>
<host parameter data type>::=<data type> [ <locator indication> ]
<locator indication>::=AS LOCATOR
<status parameter>::=SQLSTATE
<SQL procedure statement>::=<SQL executable statement>
<SQL executable statement>::=
    <SQL schema statement>
    |
    <SQL data statement>
    |
    <SQL control statement>
    |
    <SQL transaction statement>
    |
    <SQL connection statement>
    |
    <SQL session statement>
    |
    <SQL diagnostics statement>
    |
    <SQL dynamic statement>
--hr
--h2 SQL Schema Definition Statements
--/h2
<SQL schema statement>::=
    <SQL schema definition statement>
    |
    <SQL schema manipulation statement>
--p
--small
--i <SQL schema definition statement> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--p
<SQL schema definition statement>::=
    <schema definition>
    |
    <table definition>
    |
    <view definition>

```

```

|      <SQL-invoked routine>
|      <grant statement>
|      <role definition>
|      <domain definition>
|      <character set definition>
|      <collation definition>
|      <translation definition>
|      <assertion definition>
|      <trigger definition>
|      <user-defined type definition>
|      <user-defined cast definition>
|      <user-defined ordering definition>
|      <transform definition>
|      <SQL-server module definition>
<schema definition>::=
    CREATE SCHEMA <schema name clause> [ <schema character set or path> ]
    [ <schema element>... ]
<schema name clause>::=
    <schema name>
|    AUTHORIZATION <schema authorization identifier>
|    <schema name> AUTHORIZATION <schema authorization identifier>
<schema authorization identifier>::= <authorization identifier>
<schema character set or path>::=
    <schema character set specification>
|    <schema path specification>
|    <schema character set specification> <schema path specification>
|    <schema path specification> <schema character set specification>
<schema character set specification>::=
    DEFAULT CHARACTER SET <character set specification>
<schema path specification>::= <path specification>
--p
--small
--i <schema element> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
<schema element>::=
|    <table definition>
|    <view definition>
|    <domain definition>
|    <character set definition>
|    <collation definition>
|    <translation definition>
|    <assertion definition>
|    <trigger definition>
|    <user-defined type definition>
|    <schema routine>
|    <grant statement>
|    <role definition>
|    <user-defined cast definition>
|    <user-defined ordering definition>
|    <transform definition>
<table definition>::=
    CREATE [ <table scope> ] TABLE <table name> <table contents source>
    [ ON COMMIT <table commit action> ROWS ]
<table scope>::= <global or local> TEMPORARY
<global or local>::=GLOBAL | LOCAL
<table contents source>::=
    <table element list>

```

```

| OF <user-defined type> [ <subtable clause> ] [ <table element list> ]
<subtable clause>::= UNDER <supertable clause>
<supertable clause>::= <supertable name>
<supertable name>::= <table name>
<view definition>::=
    CREATE [ RECURSIVE ] VIEW <table name> <view specification>
    AS <query expression> [ WITH [ <levels clause> ] CHECK OPTION ]
<view specification>::= <regular view specification> | <referenceable view specification>
<regular view specification>::= [ <leftparen> <view column list> <rightparen> ]
<view column list>::= <column name list>
<referenceable view specification>::= OF <user-defined type> [ <subview clause> ] [ <view element list> ]
<subview clause>::= UNDER <table name>
<view element list>::=
    <leftparen> [ <self-referencing column specification> <comma> ]
    <view element> [ { <comma> <view element> } ... ] <rightparen>
<view element>::= <view column option>
<view column option>::= <column name> WITH OPTIONS <scope clause>
<levels clause>::= CASCADED | LOCAL
<domain definition>::=
    CREATE DOMAIN <domain name> [ AS ] <data type>
    [ <default clause> ] [ <domain constraint> ... ] [ <collate clause> ]
<domain constraint>::=
    [ <constraint name definition> ] <check constraint definition> [ <constraint
characteristics> ]
<character set definition>::=
    CREATE CHARACTER SET <character set name>
    [ AS ] <character set source> [ <collate clause> ]
<character set source>::= GET <character set specification>
<collation definition>::=
    CREATE COLLATION <collation name> FOR <character set specification>
    FROM <existing collation name> [ <pad characteristic> ]
<existing collation name>::= <collation name>
<pad characteristic>::= NO PAD | PAD SPACE
<translation definition>::=
    CREATE TRANSLATION <translation name> FOR <source character set
specification>
    TO <target character set specification> FROM <translation source>
<source character set specification>::= <character set specification>
<target character set specification>::= <character set specification>
<translation source>::= <existing translation name> | <translation routine>
<existing translation name>::= <translation name>
<translation routine>::= <specific routine designator>
--p
--small
--i <specific routine designator> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
<specific routine designator>::=
    SPECIFIC <routine type> <specific name>
    |
    <routine type> <member name> [ FOR <user-defined type name> ]
--p
--small
--i <specific routine designator> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
<routine type>::=
    ROUTINE | FUNCTION | PROCEDURE

```

❧ SQL 99 _BNF

```

| [ INSTANCE | STATIC | CONSTRUCTOR ] METHOD
<specific name>::= <schema qualified name>
<member name>::= <schema qualified routine name> [ <data type list> ]
<schema qualified routine name>::= <schema qualified name>
<data type list>::=
    <leftparen> [ <data type> [ { <comma> <data type> }... ] ] <rightparen>
<assertion definition>::=
    CREATE ASSERTION <constraint name>
    CHECK <leftparen> <search condition> <rightparen> [ <constraint
characteristics> ]
<trigger definition>::=
    CREATE TRIGGER <trigger name> <trigger action time> <trigger event>
    ON <table name> [ REFERENCING <old or new values alias list> ] <triggered
action>
<trigger name>::= <schema qualified name>
<trigger action time>::= BEFORE | AFTER
<trigger event>::= INSERT | DELETE | UPDATE [ OF <trigger column list> ]
<trigger column list>::= <column name list>
<old or new values alias list>::= <old or new values alias>...
<old or new values alias>::=
    OLD [ ROW ] [ AS ] <old values correlation name>
    | NEW [ ROW ] [ AS ] <new values correlation name>
    | OLD TABLE [ AS ] <old values table alias>
    | NEW TABLE [ AS ] <new values table alias>
<old values correlation name>::= <correlation name>
<new values correlation name>::= <correlation name>
<old values table alias>::= <identifier>
<new values table alias>::= <identifier>
<triggered action>::=
    [ FOR EACH { ROW | STATEMENT } ]
    [ WHEN <leftparen> <search condition> <rightparen> ] <triggered SQL
statement>
<triggered SQL statement>::=
    <SQL procedure statement>
    | BEGIN ATOMIC { <SQL procedure statement> <semicolon> }... END
<user-defined type definition>::= CREATE TYPE <user-defined type body>
--p
--small
--i <user-defined type body> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
<user-defined type body>::=
    <user-defined type name> [ <subtype clause> ] [ AS <representation> ]
    [ <instantiateable clause> ] <finality> [ <reference type specification> ]
    [ <ref cast option> ] [ <cast option> ] [ <method specification list> ]
<subtype clause>::= UNDER <supertype name>
<supertype name>::= <user-defined type>
<representation>::= <predefined type> | <member list>
<member list>::= <leftparen> <member> [ { <comma> <member> }... ] <rightparen>
<member>::= <attribute definition>
<attribute definition>::=
    <attribute name> <data type> [ <reference scope check> ] [ <attribute default> ]
    [ <collate clause> ]
<attribute name>::= <identifier>
<attribute default>::= <default clause>
<instantiateable clause>::= INSTANTIABLE | NOT INSTANTIABLE
<finality>::= FINAL | NOT FINAL

```

```

<reference type specification>::= <user-defined representation> | <derived representation> | <system-
generated representation>
<user-defined representation>::=REF USING <predefined type>
<ref cast option>::=[ <cast to ref> ] [ <cast to type> ]
<cast to ref>::=
    CAST <leftparen> SOURCE AS REF <rightparen> WITH <cast to ref identifier>
<cast to ref identifier>::= <identifier>
<cast to type>::=
    CAST <leftparen> REF AS SOURCE <rightparen> WITH <cast to type
identifier>
<cast to type identifier>::= <identifier>
<derived representation>::=REF FROM <list of attributes>
<list of attributes>::=
    <leftparen> <attribute name> [ { <comma> <attribute name> }... ] <rightparen>
<system-generated representation>::=REF IS SYSTEM GENERATED
<cast option>::=[ <cast to distinct> ] [ <cast to source> ]
<cast to distinct>::=
    CAST <leftparen> SOURCE AS DISTINCT <rightparen> WITH <cast to distinct
identifier>
<cast to distinct identifier>::= <identifier>
<cast to source>::=
    CAST <leftparen> DISTINCT AS SOURCE <rightparen> WITH <cast to source
identifier>
<cast to source identifier>::= <identifier>
<method specification list>::= <method specification> [ { <comma> <method specification> }... ]
<method specification>::= <original method specification> | <overriding method specification>
<original method specification>::=
    <partial method specification> [ SELF AS RESULT ] [ SELF AS LOCATOR ]
    [ <method characteristics> ]
--p
--small
--i <partial method specification> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
<partial method specification>::=
    [ INSTANCE | STATIC | CONSTRUCTOR ] METHOD <method name>
    <SQL parameter declaration list> <returns clause> [ SPECIFIC <specific
method name> ]
<SQL parameter declaration list>::=
    <leftparen> [ <SQL parameter declaration>
    [ { <comma> <SQL parameter declaration> }... ] ] <rightparen>
<SQL parameter declaration>::=
    [ <parameter mode> ] [ <SQL parameter name> ] <parameter type> [ RESULT ]
<parameter mode>::=IN | OUT | INOUT
<SQL parameter name>::= <identifier>
<parameter type>::= <data type> [ <locator indication> ]
<returns clause>::=RETURNS <returns data type> [ <result cast> ]
<returns data type>::= <data type> [ <locator indication> ]
<result cast>::=CAST FROM <result cast from type>
<result cast from type>::= <data type> [ <locator indication> ]
--p
--small
--i <specific method name> added per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
<specific method name>::=[ <schema name> <period> ] <qualified identifier>
<method characteristics>::= <method characteristic>...

```

۳۶۳ SQL 99 _ BNF

```

--p
--small
--i <method characteristic> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--p
<method characteristic>::=
    | <language clause>
    | <parameter style clause>
    | <deterministic characteristic>
    | <SQL-data access indication>
    | <null-call clause>
<parameter style clause>::=PARAMETER STYLE <parameter style>
<parameter style>::=SQL | GENERAL
<deterministic characteristic>::=DETERMINISTIC | NOT DETERMINISTIC
<SQL-data access indication>::=
    | NO SQL
    | CONTAINS SQL
    | READS SQL DATA
    | MODIFIES SQL DATA
<null-call clause>::=
    | RETURNS NULL ON NULL INPUT
    | CALLED ON NULL INPUT
<overriding method specification>::=OVERRIDING <partial method specification>
<schema routine>::= <schema procedure> | <schema function>
<schema procedure>::=CREATE <SQL-invoked procedure>
<SQL-invoked procedure>::=
    PROCEDURE <schema qualified routine name>
    <SQL parameter declaration list> <routine characteristics> <routine body>
<routine characteristics>::=[ <routine characteristic>... ]
--p
--small
--i <routine characteristic> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--p
<routine characteristic>::=
    | <language clause>
    | <parameter style clause>
    | SPECIFIC <specific name>
    | <deterministic characteristic>
    | <SQL-data access indication>
    | <null-call clause>
    | <dynamic result sets characteristic>
<dynamic result sets characteristic>::=
    DYNAMIC RESULT SETS <maximum dynamic result sets>
<maximum dynamic result sets>::= <unsigned integer>
<routine body>::= <SQL routine body> | <external body reference>
<SQL routine body>::= <SQL procedure statement>
--p
--small
--i <external body reference> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--p
<external body reference>::=
    EXTERNAL [ NAME <external routine name> ] [ <parameter style clause> ]
    [ <transform group specification> ] [ <external security clause> ]
<external routine name>::= <identifier> | <character string literal>

```

```

<external security clause>::=
    | EXTERNAL SECURITY DEFINER
    | EXTERNAL SECURITY INVOKER
    | EXTERNAL SECURITY IMPLEMENTATION DEFINED
<schema function>::=CREATE <SQL-invoked function>
<SQL-invoked function>::=
    { <function specification> | <method specification designator> } <routine body>
<function specification>::=
    FUNCTION <schema qualified routine name> <SQL parameter declaration list>
    <returns clause> <routine characteristics> [ <dispatch clause> ]
<dispatch clause>::=STATIC DISPATCH
--p
--small
--i <method specification designator> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
<method specification designator>::=
    [ INSTANCE | STATIC | CONSTRUCTOR ] METHOD <method name>
    <SQL parameter declaration list> [ <returns clause> ] FOR <user-defined type
name>
<grant statement>::= <grant privilege statement> | <grant role statement>
<grant privilege statement>::=
    GRANT <privileges> TO <grantee> [ { <comma> <grantee> }... ]
    [ WITH HIERARCHY OPTION ] [ WITH GRANT OPTION ] [ GRANTED BY
<grantor> ]
<privileges>::= <object privileges> ON <object name>
<object privileges>::=ALL PRIVILEGES | <action> [ { <comma> <action> }... ]
<action>::=
    | SELECT
    | SELECT <leftparen> <privilege column list> <rightparen>
    | SELECT <leftparen> <privilege method list> <rightparen>
    | DELETE
    | INSERT [ <leftparen> <privilege column list> <rightparen> ]
    | UPDATE [ <leftparen> <privilege column list> <rightparen> ]
    | REFERENCES [ <leftparen> <privilege column list> <rightparen> ]
    | USAGE
    | TRIGGER
    | UNDER
    | EXECUTE
<privilege column list>::= <column name list>
<privilege method list>::=
    <specific routine designator> [ { <comma> <specific routine designator> }... ]
<object name>::=
    | [ TABLE ] <table name>
    | DOMAIN <domain name>
    | COLLATION <collation name>
    | CHARACTER SET <character set name>
    | MODULE <module name>
    | TRANSLATION <translation name>
    | TYPE <user-defined type name>
    | <specific routine designator>
<grantee>::=PUBLIC | <authorization identifier>
<grantor>::=CURRENT_USER | CURRENT_ROLE
<grant role statement>::=
    GRANT <role granted> [ { <comma> <role granted> }... ] TO <grantee> [ {
<comma> <grantee> }... ]
    [ WITH ADMIN OPTION ] [ GRANTED BY <grantor> ]
<role granted>::= <role name>

```

۳۶۵ SQL 99 _BNF

```
<role definition>::=CREATE ROLE <role name> [ WITH ADMIN <grantor> ]
<SQL-invoked routine>::= <schema routine> | <module routine>
<user-defined cast definition>::=
    CREATE CAST <leftparen> <source data type> AS <target data type>
<rightparen>
    WITH <cast function> [ AS ASSIGNMENT ]
<source data type>::= <data type>
<cast function>::= <specific routine designator>
--p
--small
--i <user-defined ordering specification> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
<user-defined ordering definition>::=
    CREATE ORDERING FOR <user-defined type name> <ordering form>
<ordering form>::= <equals ordering form> | <full ordering form>
<equals ordering form>::=EQUALS ONLY BY <ordering category>
<ordering category>::= <relative category> | <map category> | <state category>
<relative category>::=RELATIVE WITH <relative function specification>
<relative function specification>::= <specific routine designator>
<map category>::=MAP WITH <map function specification>
<map function specification>::= <specific routine designator>
<state category>::=STATE [ <specific name> ]
<full ordering form>::=ORDER FULL BY <ordering category>
--p
--small
--i <transform definition> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
<transform definition>::=
    CREATE { TRANSFORM | TRANSFORMS } FOR <user-defined type name>
    <transform group>...
<transform group>::= <group name> <leftparen> <transform element list> <rightparen>
<transform element list>::= <transform element> [ <comma> <transform element> ]
<transform element>::= <to sql> | <from sql>
<to sql>::=TO SQL WITH <to sql function>
<to sql function>::= <specific routine designator>
<from sql>::=FROM SQL WITH <from sql function>
<from sql function>::= <specific routine designator>
--hr
--h2 SQL Schema Manipulation Statements
--/h2
<SQL schema manipulation statement>::=
    | <drop schema statement>
    | <alter table statement>
    | <drop table statement>
    | <drop view statement>
    | <alter routine statement>
    | <drop routine statement>
    | <drop user-defined cast statement>
    | <revoke statement>
    | <drop role statement>
    | <alter domain statement>
    | <drop domain statement>
    | <drop character set statement>
    | <drop collation statement>
    | <drop translation statement>
```

```

|      <drop assertion statement>
|      <drop trigger statement>
|      <alter type statement>
|      <drop data type statement>
|      <drop user-defined ordering statement>
|      <drop transform statement>
|      <drop module statement>
<drop schema statement>::=DROP SCHEMA <schema name> <drop behavior>
<drop behavior>::=CASCADE | RESTRICT
<alter table statement>::=ALTER TABLE <table name> <alter table action>
<alter table action>::=
|      <add column definition>
|      <alter column definition>
|      <drop column definition>
|      <add table constraint definition>
|      <drop table constraint definition>
<add column definition>::=ADD [ COLUMN ] <column definition>
<alter column definition>::=ALTER [ COLUMN ] <column name> <alter column action>
<alter column action>::=
|      <set column default clause>
|      <drop column default clause>
|      <add column scope clause>
|      <drop column scope clause>
<set column default clause>::=SET <default clause>
<drop column default clause>::=DROP DEFAULT
<add column scope clause>::=ADD <scope clause>
<drop column scope clause>::=DROP SCOPE <drop behavior>
<drop column definition>::=DROP [ COLUMN ] <column name> <drop behavior>
<add table constraint definition>::=ADD <table constraint definition>
<drop table constraint definition>::=DROP CONSTRAINT <constraint name> <drop behavior>
<drop table statement>::=DROP TABLE <table name> <drop behavior>
<drop view statement>::=DROP VIEW <table name> <drop behavior>
<alter routine statement>::=
    ALTER <specific routine designator> <alter routine characteristics> <alter
routine behaviour>
<alter routine characteristics>::= <alter routine characteristic>...
<alter routine characteristic>::=
|      <language clause>
|      <parameter style clause>
|      <SQL-data access indication>
|      <null-call clause>
|      <dynamic result sets characteristic>
|      NAME <external routine name>
<alter routine behaviour>::=RESTRICT
<drop routine statement>::=DROP <specific routine designator> <drop behavior>
<drop user-defined cast statement>::=
    DROP CAST <leftparen> <source data type> AS <target data type>
<rightparen>
    <drop behavior>
<revoke statement>::= <revoke privilege statement> | <revoke role statement>
<revoke privilege statement>::=
    REVOKE [ <revoke option extension> ] <privileges> FROM <grantee>
    [ { <comma> <grantee> }... ] [ GRANTED BY <grantor> ] <drop behavior>
<revoke option extension>::=GRANT OPTION FOR | HIERARCHY OPTION FOR
<revoke role statement>::=
    REVOKE [ADMIN OPTION FOR] <role revoked> [{ <comma> <role
revoked>}...]
    FROM <grantee> [ { <comma> <grantee> }... ] [ GRANTED BY <grantor> ]
    <drop behavior>

```

٣٦٧ SQL 99 _BNF

```

<role revoked>::= <role name>
<drop role statement>::=DROP ROLE <role name>
<alter domain statement>::=ALTER DOMAIN <domain name> <alter domain action>
<alter domain action>::=
    | <set domain default clause>
    | <drop domain default clause>
    | <add domain constraint definition>
    | <drop domain constraint definition>
<set domain default clause>::=SET <default clause>
<drop domain default clause>::=DROP DEFAULT
<add domain constraint definition>::=ADD <domain constraint>
<drop domain constraint definition>::=DROP CONSTRAINT <constraint name>
<drop domain statement>::=DROP DOMAIN <domain name> <drop behavior>
<drop character set statement>::=DROP CHARACTER SET <character set name>
<drop collation statement>::=DROP COLLATION <collation name> <drop behavior>
<drop translation statement>::=DROP TRANSLATION <translation name>
<drop assertion statement>::=DROP ASSERTION <constraint name>
<drop trigger statement>::=DROP TRIGGER <trigger name>
<alter type statement>::=ALTER TYPE <user-defined type name> <alter type action>
<alter type action>::=
    | <add attribute definition>
    | <drop attribute definition>
    | <add original method specification>
    | <add overriding method specification>
    | <drop method specification>
<add attribute definition>::=ADD ATTRIBUTE <attribute definition>
<drop attribute definition>::=DROP ATTRIBUTE <attribute name> RESTRICT
<add original method specification>::=ADD <original method specification>
<add overriding method specification>::=ADD <overriding method specification>
--p
--small
--i <drop method specification> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
<drop method specification>::=DROP <specific method specification designator> RESTRICT
--p
--small
--i <specific method specification designator> added per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
<specific method specification designator>::=
    SPECIFIC METHOD <specific method name>
    | [ INSTANCE | STATIC | CONSTRUCTOR ] METHOD <method name> [ <data
type list> ]
<drop data type statement>::=DROP TYPE <user-defined type name> <drop behavior>
--p
--small
--i <drop user-defined ordering statement> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
<drop user-defined ordering statement>::=
    DROP ORDERING FOR <user-defined type name> <drop behavior>
--p
--small
--i <drop transform statement> modified per ISO 9075:1999/Cor.1:2000(E)
--/i

```

```
--/small
--p
<drop transform statement>::=
    DROP { TRANSFORM | TRANSFORMS } <transforms to be dropped>
    FOR <user-defined type name> <drop behavior>
<transforms to be dropped>::=ALL | <transform group element>
<transform group element>::= <group name>
--hr
--h2 SQL Data Manipulation Statements
--/h2
<SQL data statement>::=
    <open statement>
    | <fetch statement>
    | <close statement>
    | <select statement: single row>
    | <free locator statement>
    | <hold locator statement>
    | <SQL data change statement>
<open statement>::=OPEN <cursor name>
<fetch statement>::=
    FETCH [ [ <fetch orientation> ] FROM ] <cursor name> INTO <fetch target list>
<fetch orientation>::=
    NEXT | PRIOR | FIRST | LAST
    | { ABSOLUTE | RELATIVE } <simple value specification>
<simple value specification>::=
    <literal>
    | <host parameter name>
    | <SQL parameter reference>
    | <SQL variable reference>
    | <embedded variable name>
<fetch target list>::= <target specification> [ { <comma> <target specification> }... ]
<close statement>::=CLOSE <cursor name>
<select statement: single row>::=
    SELECT [ <set quantifier> ] <select list> INTO <select target list> <table
expression>
<select target list>::= <target specification> [ { <comma> <target specification> }... ]
<free locator statement>::=
    FREE LOCATOR <locator reference> [ { <comma> <locator reference> }... ]
<locator reference>::= <host parameter name> | <embedded variable name>
<hold locator statement>::=
    HOLD LOCATOR <locator reference> [ { <comma> <locator reference> }... ]
<SQL data change statement>::=
    <delete statement: positioned>
    | <delete statement: searched>
    | <insert statement>
    | <update statement: positioned>
    | <update statement: searched>
<delete statement: positioned>::=
    DELETE FROM <target table> WHERE CURRENT OF <cursor name>
--p
--small
--i <target table> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--p
<target table>::=
    <table name>
    | [ ONLY ] <leftparen> <table name> <rightparen>
<delete statement: searched>::=
```

❧ SQL 99 _BNF

```

DELETE FROM <target table> [ WHERE <search condition> ]
<insert statement>::=
    INSERT INTO <insertion target> <insert columns and source>
<insertion target>::= <table name>
<insert columns and source>::= <from subquery> | <from constructor> | <from default>
<from subquery>::=
    [ <leftparen> <insert column list> <rightparen> ] [ <override clause> ] <query
expression>
<insert column list>::= <column name list>
<from constructor>::=
    [ <leftparen> <insert column list> <rightparen> ] [ <override clause> ]
    <contextually typed table value constructor>
<override clause>::= OVERRIDING USER VALUE | OVERRIDING SYSTEM VALUE
<contextually typed table value constructor>::=
    VALUES <contextually typed row value expression list>
<contextually typed row value expression list>::=
    <contextually typed row value expression>
    [ { <comma> <contextually typed row value expression> } ... ]
<contextually typed row value expression>::=
    <row value special case>
    | <contextually typed row value constructor>
<contextually typed row value constructor>::=
    <contextually typed row value constructor element>
    | [ ROW ] <leftparen> <contextually typed row value constructor element list>
<rightparen>
<contextually typed row value constructor element>::=
    <value expression> | <contextually typed value specification>
<contextually typed value specification>::=
    <implicitly typed value specification> | <default specification>
<default specification>::= DEFAULT
<contextually typed row value constructor element list>::=
    <contextually typed row value constructor element>
    [ { <comma> <contextually typed row value constructor element> } ... ]
<from default>::= DEFAULT VALUES
<update statement: positioned>::=
    UPDATE <target table> SET <set clause list> WHERE CURRENT OF <cursor name>
<set clause list>::= <set clause> [ { <comma> <set clause> } ... ]
<set clause>::=
    <update target> <equals operator> <update source>
    | <mutated set clause> <equals operator> <update source>
--p
--small
--i <update target> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
<update target>::=
    <object column>
    | <object column> <left bracket or trigraph> <simple value specification> <right
bracket or trigraph>
<object column>::= <column name>
<update source>::= <value expression> | <contextually typed value specification>
<mutated set clause>::= <mutated target> <period> <method name>
<mutated target>::= <object column> | <mutated set clause>
<update statement: searched>::=
    UPDATE <target table> SET <set clause list> [ WHERE <search condition> ]
--hr
--h2 SQL Control Statements
--/h2

```

```

<SQL control statement>::=
    | <call statement>
    | <return statement>
    | <assignment statement>
    | <compound statement>
    | <case statement>
    | <if statement>
    | <iterate statement>
    | <leave statement>
    | <loop statement>
    | <while statement>
    | <repeat statement>
    | <for statement>
<call statement>::=CALL <routine invocation>
<return statement>::=RETURN <return value>
<return value>::= <value expression> | NULL
--hr
--h2 Transaction Management
--/h2
<SQL transaction statement>::=
    | <start transaction statement>
    | <set transaction statement>
    | <set constraints mode statement>
    | <savepoint statement>
    | <release savepoint statement>
    | <commit statement>
    | <rollback statement>
<start transaction statement>::=
    START TRANSACTION <transaction mode> [ { <comma> <transaction mode> } ... ]
<transaction mode>::= <isolation level> | <transaction access mode> | <diagnostics size>
<isolation level>::=ISOLATION LEVEL <level of isolation>
<level of isolation>::=
    | READ UNCOMMITTED
    | READ COMMITTED
    | REPEATABLE READ
    | SERIALIZABLE
<transaction access mode>::=READ ONLY | READ WRITE
<diagnostics size>::=DIAGNOSTICS SIZE <number of conditions>
<number of conditions>::= <simple value specification>
<set transaction statement>::=SET [ LOCAL ] <transaction characteristics>
<transaction characteristics>::=
    TRANSACTION <transaction mode> [ { <comma> <transaction mode> } ... ]
<set constraints mode statement>::=
    SET CONSTRAINTS <constraint name list> { DEFERRED | IMMEDIATE }
<constraint name list>::=ALL | <constraint name> [ { <comma> <constraint name> } ... ]
<savepoint statement>::=SAVEPOINT <savepoint specifier>
--p
--small
--i <savepoint specifier> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
<savepoint specifier>::= <savepoint name>
<savepoint name>::= <identifier>
<simple target specification>::=
    | <host parameter specification>
    | <SQL parameter reference>
    | <column reference>
    | <SQL variable reference>

```

371 SQL 99 _BNF

```

| <embedded variable name>
<release savepoint statement>::=RELEASE SAVEPOINT <savepoint specifier>
<commit statement>::=COMMIT [ WORK ] [ AND [ NO ] CHAIN ]
<rollback statement>::=ROLLBACK [ WORK ] [ AND [ NO ] CHAIN ] [ <savepoint clause> ]
<savepoint clause>::=TO SAVEPOINT <savepoint specifier>
--hr
--h2 Connection Management
--/h2
<SQL connection statement>::= <connect statement> | <set connection statement> | <disconnect
statement>
<connect statement>::=CONNECT TO <connection target>
<connection target>::=
    <SQL-server name> [ AS <connection name> ] [ USER <connection user name>
]
|
    DEFAULT
<SQL-server name>::= <simple value specification>
<connection name>::= <simple value specification>
<connection user name>::= <simple value specification>
<set connection statement>::=SET CONNECTION <connection object>
<connection object>::=DEFAULT | <connection name>
<disconnect statement>::=DISCONNECT <disconnect object>
<disconnect object>::= <connection object> | ALL | CURRENT
--hr
--h2 Session Attributes
--/h2
<SQL session statement>::=
    <set session user identifier statement>
    |
    <set role statement>
    |
    <set local time zone statement>
    |
    <set session characteristics statement>
    |
    <set catalog statement>
    |
    <set schema statement>
    |
    <set names statement>
    |
    <set path statement>
    |
    <set transform group statement>
<set session user identifier statement>::=
    SET SESSION AUTHORIZATION <value specification>
<set role statement>::=SET ROLE <role specification>
<role specification>::= <value specification> | NONE
<set local time zone statement>::=SET TIME ZONE <set time zone value>
<set time zone value>::= <interval value expression> | LOCAL
<set session characteristics statement>::=
    SET SESSION CHARACTERISTICS AS <session characteristic list>
<session characteristic list>::= <session characteristic> [ { <comma> <session characteristic> } ... ]
<session characteristic>::= <transaction characteristics>
<SQL diagnostics statement>::= <get diagnostics statement> | <signal statement> | <resignal statement>
<get diagnostics statement>::=GET DIAGNOSTICS <SQL diagnostics information>
<SQL diagnostics information>::= <statement information> | <condition information>
<statement information>::=
    <statement information item> [ { <comma> <statement information item> } ... ]
<statement information item>::=
    <simple target specification><equals operator><statement information item name>
<statement information item name>::=
    NUMBER
    |
    MORE
    |
    COMMAND_FUNCTION
    |
    COMMAND_FUNCTION_CODE
    |
    DYNAMIC_FUNCTION
    |
    DYNAMIC_FUNCTION_CODE

```

```

|      ROW_COUNT
|      TRANSACTIONS_COMMITTED
|      TRANSACTIONS_ROLLED_BACK
|      TRANSACTION_ACTIVE
<condition information>::=
    EXCEPTION <condition number>
    <condition information item> [ { <comma> <condition information item> }... ]
<condition number>::= <simple value specification>
<condition information item>::=
    <simple target specification> <equals operator> <condition information item name>
<condition information item name>::=
    CATALOG_NAME
|    CLASS_ORIGIN
|    COLUMN_NAME
|    CONDITION_IDENTIFIER
|    CONDITION_NUMBER
|    CONNECTION_NAME
|    CONSTRAINT_CATALOG
|    CONSTRAINT_NAME
|    CONSTRAINT_SCHEMA
|    CURSOR_NAME
|    MESSAGE_LENGTH
|    MESSAGE_OCTET_LENGTH
|    MESSAGE_TEXT
|    PARAMETER_MODE
|    PARAMETER_NAME
|    PARAMETER_ORDINAL_POSITION
|    RETURNED_SQLSTATE
|    ROUTINE_CATALOG
|    ROUTINE_NAME
|    ROUTINE_SCHEMA
|    SCHEMA_NAME
|    SERVER_NAME
|    SPECIFIC_NAME
|    SUBCLASS_ORIGIN
|    TABLE_NAME
|    TRIGGER_CATALOG
|    TRIGGER_NAME
|    TRIGGER_SCHEMA
<dereference operation>::=
    <reference value expression> <dereference operator> <attribute name>
<method reference>::=
    <value expression primary> <dereference operator> <method name> <SQL
argument list>
<method selection>::= <routine invocation>
--p
--small
--i <new invocation> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
<new invocation>::= <method invocation> | <routine invocation>
<static method selection>::= <routine invocation>
<token>::= <nondelimiter token> | <delimiter token>
<nondelimiter token>::=
    <regular identifier>
|    <key word>
|    <unsigned numeric literal>
|    <national character string literal>

```

SQL 99 _BNF

```

| <bit string literal>
| <hex string literal>
| <large object length token>
| <multiplier>
<key word> ::= <reserved word> | <non-reserved word>
--p
--small
--i <reserved word> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
<reserved word> ::=
ABSOLUTE | ACTION | ADD | AFTER | ALL | ALLOCATE | ALTER | AND |
ANY | ARE
| ARRAY | AS | ASC | ASSERTION | AT | AUTHORIZATION
| BEFORE | BEGIN | BETWEEN | BINARY | BIT | BLOB | BOOLEAN | BOTH
| BREADTH | BY
| CALL | CASCADE | CASCADED | CASE | CAST | CATALOG | CHAR |
CHARACTER
| CHECK | CLOB | CLOSE | COLLATE | COLLATION | COLUMN | COMMIT
| CONDITION | CONNECT | CONNECTION | CONSTRAINT | CONSTRAINTS
| CONSTRUCTOR | CONTINUE | CORRESPONDING | CREATE | CROSS |
CUBE
| CURRENT | CURRENT_DATE |
CURRENT_DEFAULT_TRANSFORM_GROUP
| CURRENT_TRANSFORM_GROUP_FOR_TYPE | CURRENT_PATH |
CURRENT_ROLE
| CURRENT_TIME | CURRENT_TIMESTAMP | CURRENT_USER | CURSOR |
CYCLE
| DATA | DATE | DAY | DEALLOCATE | DEC | DECIMAL | DECLARE |
DEFAULT
| DEFERRABLE | DEFERRED | DELETE | DEPTH | Deref | DESC
| DESCRIBE | DESCRIPTOR | DETERMINISTIC
| DIAGNOSTICS | DISCONNECT | DISTINCT | DO | DOMAIN | DOUBLE
| DROP | DYNAMIC
| EACH | ELSE | ELSEIF | END | END-EXEC | EQUALS | ESCAPE | EXCEPT
| EXCEPTION | EXEC | EXECUTE | EXISTS | EXIT | EXTERNAL
| FALSE | FETCH | FIRST | FLOAT | FOR | FOREIGN | FOUND | FROM | FREE
| FULL | FUNCTION
| GENERAL | GET | GLOBAL | GO | GOTO | GRANT | GROUP | GROUPING
| HANDLE | HAVING | HOLD | HOUR
| IDENTITY | IF | IMMEDIATE | IN | INDICATOR
| INITIALLY | INNER | INOUT | INPUT | INSERT | INT | INTEGER
| INTERSECT | INTERVAL | INTO | IS | ISOLATION
| JOIN
| KEY
| LANGUAGE | LARGE | LAST | LATERAL | LEADING | LEAVE | LEFT
| LEVEL | LIKE | LOCAL | LOCALTIME | LOCALTIMESTAMP | LOCATOR |
LOOP
| MAP | MATCH | METHOD | MINUTE | MODIFIES | MODULE | MONTH
| NAMES | NATIONAL | NATURAL | NCHAR | NCLOB | NESTING | NEW |
NEXT
| NO | NONE | NOT | NULL | NUMERIC
| OBJECT | OF | OLD | ON | ONLY | OPEN | OPTION
| OR | ORDER | ORDINALITY | OUT | OUTER | OUTPUT | OVERLAPS
| PAD | PARAMETER | PARTIAL | PATH | PRECISION
| PREPARE | PRESERVE | PRIMARY | PRIOR | PRIVILEGES | PROCEDURE |
PUBLIC

```

REFERENCING	READ READS REAL RECURSIVE REDO REF REFERENCES
RETURN	RELATIVE RELEASE REPEAT RESIGNAL RESTRICT RESULT
	RETURNS REVOKE RIGHT ROLE ROLLBACK ROLLUP ROUTINE
	ROW ROWS
SELECT	SAVEPOINT SCHEMA SCROLL SEARCH SECOND SECTION
	SESSION SESSION_USER SET SETS SIGNAL SIMILAR SIZE
SQLEXCEPTION	SMALLINT SOME SPACE SPECIFIC SPECIFICTYPE SQL
	SQLSTATE SQLWARNING START STATE STATIC SYSTEM_USER
	TABLE TEMPORARY THEN TIME TIMESTAMP
TRANSACTION	TIMEZONE_HOUR TIMEZONE_MINUTE TO TRAILING
	TRANSLATION TREAT TRIGGER TRUE
UPDATE	UNDER UNDO UNION UNIQUE UNKNOWN UNNEST UNTIL
	USAGE USER USING
	VALUE VALUES VARCHAR VARYING VIEW
WRITE	WHEN WHENEVER WHERE WHILE WITH WITHOUT WORK
	YEAR
	ZONE
--p	
--small	
--i <non-reserved word>	modified per ISO 9075:1999/Cor.1:2000(E)
--/i	
--/small	
--/p	
<non-reserved word>::=	
ATOMIC	ABS ADA ADMIN ASENSITIVE ASSIGNMENT ASYMMETRIC
	ATTRIBUTE AVG
	BIT_LENGTH
CHAR_LENGTH	C CALLED CARDINALITY CATALOG_NAME CHAIN
CHARACTER_SET_CATALOG	CHARACTERISTICS CHARACTER_LENGTH
CHARACTER_SET_NAME	CHARACTER_SET_NAME CHARACTER_SET_SCHEMA CHECKED
CLASS_ORIGIN	COALESCE COBOL COLLATION_CATALOG COLLATION_NAME
COLLATION_SCHEMA	COLUMN_NAME COMMAND_FUNCTION
COMMAND_FUNCTION_CODE	COMMITTED
CONNECTION_NAME	CONDITION_IDENTIFIER CONDITION_NUMBER
CONSTRAINT_SCHEMA	CONSTRAINT_CATALOG CONSTRAINT_NAME
CONTAINS	CONVERT COUNT CURSOR_NAME
DEFINED	DATETIME_INTERVAL_CODE DATETIME_INTERVAL_PRECISION
	DEFINER DEGREE DERIVED DISPATCH
	EVERY EXTRACT
	FINAL FORTRAN
	G GENERATED GRANTED
	HIERARCHY
INVOKER	IMPLEMENTATION INSENSITIVE INSTANCE INSTANTIABLE

३१० SQL 99 _BNF

		K KEY_MEMBER KEY_TYPE
		LENGTH LOWER
MESSAGE		M MAX MIN MESSAGE_LENGTH MESSAGE_OCTET_LENGTH
TEXT		MOD MORE MUMPS
		NAME NULLABLE NUMBER NULLIF
		OCTET_LENGTH ORDERING OPTIONS OVERLAY OVERRIDING
		PASCAL PARAMETER_MODE PARAMETER_NAME
		PARAMETER_ORDINAL_POSITION PARAMETER_SPECIFIC_CATALOG
		PARAMETER_SPECIFIC_NAME PARAMETER_SPECIFIC_SCHEMA PLI
POSITION		REPEATABLE RETURNED_CARDINALITY RETURNED_LENGTH
		RETURNED_OCTET_LENGTH RETURNED_SQLSTATE
ROUTINE_CATALOG		ROUTINE_NAME ROUTINE_SCHEMA ROW_COUNT
		SCALE SCHEMA_NAME SCOPE SECURITY SELF SENSITIVE
SERIALIZABLE		SERVER_NAME SIMPLE SOURCE SPECIFIC_NAME STATEMENT
STRUCTURE		STYLE SUBCLASS_ORIGIN SUBSTRING SUM SYMMETRIC
SYSTEM		TABLE_NAME TOP_LEVEL_COUNT TRANSACTIONS_COMMITTED
		TRANSACTIONS_ROLLED_BACK TRANSACTION_ACTIVE
TRANSFORM		TRANSFORMS TRANSLATE TRIGGER_CATALOG
TRIGGER_SCHEMA		TRIGGER_NAME TRIM TYPE
		UNCOMMITTED UNNAMED UPPER
<delimiter token>::=		<character string literal>
		<date string>
		<time string>
		<timestamp string>
		<interval string>
		<delimited identifier>
		<SQL special character>
		<not equals operator>
		<greater than or equals operator>
		<less than or equals operator>
		<concatenation operator>
		<right arrow>
		<left bracket trigraph>
		<right bracket trigraph>
		<double colon>
		<double period>
<CLI routine>::=		<CLI routine name> <CLI parameter list> [<CLI returns clause>]
<CLI routine name>::=		<CLI name prefix> <CLI generic name>
<CLI name prefix>::=		<CLI by-reference prefix> <CLI by-value prefix>
<CLI by-reference prefix>::=		SQLR
<CLI by-value prefix>::=		SQL
<CLI generic name>::=		AllocConnect AllocEnv AllocHandle AllocStmt
		BindCol BindParameter
		Cancel CloseCursor ColAttribute ColumnPrivileges Columns Connect
CopyDesc		DataSources DescribeCol Disconnect
		EndTran Error ExecDirect Execute
		Fetch FetchScroll ForeignKeys FreeConnect FreeEnv FreeHandle FreeStmt

GetDiagField | GetConnectAttr | GetCursorName | GetData | GetDescField | GetDescRec |
 | GetDiagRec | GetEnvAttr | GetFeatureInfo | GetFunctions | GetInfo | GetLength
 GetTypeInfo | GetParamData | GetPosition | GetSessionInfo | GetStmtAttr | GetSubString |
 | MoreResults
 | NextResult | NumResultCols
 | ParamData | Prepare | PrimaryKeys | PutData
 | RowCount
 SetStmtAttr | SetConnectAttr | SetCursorName | SetDescField | SetDescRec | SetEnvAttr |
 | SpecialColumns | StartTran
 | TablePrivileges | Tables
 | <implementation-defined CLI generic name>
 <implementation-defined CLI generic name>::=!! (See the Syntax Rules)
 <CLI parameter list>::=
 <leftparen> <CLI parameter declaration> [{ <comma> <CLI parameter
 declaration> } ...] <rightparen>
 <CLI parameter declaration>::=
 <CLI parameter name> <CLI parameter mode> <CLI parameter data type>
 <CLI parameter name>::=!! (See the individual CLI routine definitions)
 <CLI parameter mode>::=IN | OUT | DEFIN | DEFOUT | DEF
 <CLI parameter data type>::=
 | INTEGER
 | SMALLINT
 | ANY
 | CHARACTER <leftparen> <length> <rightparen>
 <CLI returns clause>::=RETURNS SMALLINT
 <assignment statement>::=
 SET <assignment target> <equals operator> <assignment source>
 <assignment target>::= <target specification> | <modified field reference> | <mutator reference>
 <SQL variable reference>::= <basic identifier chain>
 <modified field reference>::= <modified field target> <period> <field name>
 <modified field target>::=
 <target specification> | <leftparen> <target specification> <rightparen> | <modified field
 reference>
 <mutator reference>::= <mutated target specification> <period> <method name>
 <mutated target specification>::=
 <target specification> | <leftparen> <target specification> <rightparen> |
 <mutator reference>
 <assignment source>::= <value expression> | <contextually typed source>
 <contextually typed source>::=
 <implicitly typed value specification> | <contextually typed row value
 expression>
 <compound statement>::=
 [<beginning label> <colon>] BEGIN [[NOT] ATOMIC]
 [<local declaration list>] [<local cursor declaration list>] [<local handler
 declaration list>]
 [<SQL statement list>] END [<ending label>]
 <beginning label>::= <statement label>
 <statement label>::= <identifier>
 <local declaration list>::= <terminated local declaration> ...
 <terminated local declaration>::= <local declaration> <semicolon>
 <local declaration>::= <SQL variable declaration> | <condition declaration>
 <SQL variable declaration>::=
 DECLARE <SQL variable name list> <data type> [<default clause>]
 <SQL variable name list>::=
 <SQL variable name> [{ <comma> <SQL variable name> } ...]
 <SQL variable name>::= <identifier>

SQL 99 _BNF

```
<condition declaration>::=
    DECLARE <condition name> CONDITION [ FOR <sqlstate value> ]
<condition name>::= <identifier>
<sqlstate value>::=SQLSTATE [ VALUE ] <character string literal>
<local cursor declaration list>::= <terminated local cursor declaration>...
<terminated local cursor declaration>::= <declare cursor> <semicolon>
<local handler declaration list>::= <terminated local handler declaration>...
<terminated local handler declaration>::= <handler declaration> <semicolon>
<handler declaration>::=
    DECLARE <handler type> HANDLER FOR <condition value list> <handler action>
<handler type>::=CONTINUE | EXIT | UNDO
<condition value list>::= <condition value> [ { <comma> <condition value> }... ]
<condition value>::=
    <sqlstate value> | <condition name> | SQLEXCEPTION | SQLWARNING | NOT FOUND
<handler action>::= <SQL procedure statement>
<SQL statement list>::= <terminated SQL statement>...
<terminated SQL statement>::= <SQL procedure statement> <semicolon>
<ending label>::= <statement label>
<case statement>::= <simple case statement> | <searched case statement>
<simple case statement>::=
    CASE <simple case operand 1> <simple case statement when clause>... [ <case
statement else clause> ] END CASE
<simple case operand 1>::= <value expression>
<simple case statement when clause>::=
    WHEN <simple case operand 2> THEN <SQL statement list>
<simple case operand 2>::= <value expression>
<case statement else clause>::=ELSE <SQL statement list>
<searched case statement>::=
    CASE <searched case statement when clause>... [ <case statement else clause> ]
END CASE
<searched case statement when clause>::=
    WHEN <search condition> THEN <SQL statement list>
<if statement>::=
    IF <search condition> <if statement then clause>
    [ <if statement elseif clause>... ] [ <if statement else clause> ]
    END IF
<if statement then clause>::=THEN <SQL statement list>
<if statement elseif clause>::=ELSEIF <search condition> THEN <SQL statement list>
<if statement else clause>::=ELSE <SQL statement list>
<iterate statement>::=ITERATE <statement label>
<leave statement>::=LEAVE <statement label>
<loop statement>::=
    [ <beginning label> <colon> ] LOOP <SQL statement list> END LOOP [ <ending label> ]
<while statement>::=
    [ <beginning label> <colon> ] WHILE <search condition> DO <SQL statement
list> END WHILE [ <ending label> ]
<repeat statement>::=
    [ <beginning label> <colon> ] REPEAT <SQL statement list> UNTIL <search
condition> END REPEAT [ <ending label> ]
<for statement>::=
    [ <beginning label> <colon> ] FOR <for loop variable name> AS
    <cursor name> [ <cursor sensitivity> ] CURSOR FOR ] <cursor specification>
    DO <SQL statement list> END FOR [ <ending label> ]
<for loop variable name>::= <identifier>
<signal statement>::=SIGNAL <signal value> [ <set signal information> ]
<signal value>::= <condition name> | <sqlstate value>
<set signal information>::=SET <signal information item list>
<signal information item list>::=
    <signal information item> [ { <comma> <signal information item> }... ]
```

```

<signal information item>::=
    <condition information item name> <equals operator> <simple value
specification>
<resignal statement>::=RESIGNAL [ <signal value> ] [ <set signal information> ]
<SQL-server module definition>::=
    CREATE MODULE <SQL-server module name> [ <SQL-server module
character set specification> ]
    [<SQL-server module schema clause>][<SQL-server module path specification>]
    [ <temporary table declaration> ]
    <SQL-server module contents>...
    END MODULE
<SQL-server module name>::= <schema qualified name>
<SQL-server module character set specification>::=NAMES ARE <character set specification>
<SQL-server module schema clause>::=SCHEMA <default schema name>
<default schema name>::= <schema name>
<SQL-server module path specification>::= <path specification>
<SQL-server module contents>::= <SQL-invoked routine> <semicolon>
<module routine>::= <module procedure> | <module function>
<module procedure>::=[ DECLARE ] <SQL-invoked procedure>
<module function>::=[ DECLARE ] <SQL-invoked function>
<drop module statement>::=DROP MODULE <SQL-server module name> <drop behavior>
<triggered SQL statement>::= <SQL procedure statement>
--hr
--h2 Dynamic SQL
--/h2
--p
Much, if not all, of the following material comes from ISO/IEC 9075-5:1999, SQL/Bindings.
--/p
<SQL dynamic statement>::=
    | <system descriptor statement>
    | <prepare statement>
    | <deallocate prepared statement>
    | <describe statement>
    | <execute statement>
    | <execute immediate statement>
    | <SQL dynamic data statement>
<system descriptor statement>::=
    | <allocate descriptor statement>
    | <deallocate descriptor statement>
    | <set descriptor statement>
    | <get descriptor statement>
<allocate descriptor statement>::=
    ALLOCATE [ SQL ] DESCRIPTOR <descriptor name> [ WITH MAX <occurrences> ]
<descriptor name>::=[ <scope option> ] <simple value specification>
<scope option>::=GLOBAL | LOCAL
<embedded variable name>::= <colon> <host identifier>
<host identifier>::=
    | <Ada host identifier>
    | <C host identifier>
    | <COBOL host identifier>
    | <Fortran host identifier>
    | <MUMPS host identifier>
    | <Pascal host identifier>
    | <PL/I host identifier>
<Ada host identifier>::=!! (See the Syntax Rules.)
<C host identifier>::=!! (See the Syntax Rules.)
<COBOL host identifier>::=!! (See the Syntax Rules.)
<Fortran host identifier>::=!! (See the Syntax Rules.)
<MUMPS host identifier>::=!! (See the Syntax Rules.)

```

३७९ SQL 99 _BNF

```

<Pascal host identifier>::=!! (See the Syntax Rules.)
<PL/I host identifier>::=!! (See the Syntax Rules.)
<occurrences>::= <simple value specification>
<deallocate descriptor statement>::=DEALLOCATE [ SQL ] DESCRIPTOR <descriptor name>
<set descriptor statement>::=SET [ SQL ] DESCRIPTOR <descriptor name> <set descriptor
information>
<set descriptor information>::=
    <set header information> [ { <comma> <set header information> }... ]
    |
    VALUE <item number><set item information>[ {<comma><set item information>...}]
<set header information>::=
    <header item name> <equals operator> <simple value specification 1>
<header item name>::=
    COUNT | KEY_TYPE | DYNAMIC_FUNCTION |
    DYNAMIC_FUNCTION_CODE | TOP_LEVEL_COUNT
<simple value specification 1>::= <simple value specification>
<item number>::= <simple value specification>
<set item information>::=
    <descriptor item name> <equals operator> <simple value specification 2>
<descriptor item name>::=
    CARDINALITY
    | CHARACTER_SET_CATALOG
    | CHARACTER_SET_NAME
    | CHARACTER_SET_SCHEMA
    | COLLATION_CATALOG
    | COLLATION_NAME
    | COLLATION_SCHEMA
    | DATA
    | DATETIME_INTERVAL_CODE
    | DATETIME_INTERVAL_PRECISION
    | DEGREE
    | INDICATOR
    | KEY_MEMBER
    | LENGTH
    | LEVEL
    | NAME
    | NULLABLE
    | OCTET_LENGTH
    | PARAMETER_MODE
    | PARAMETER_ORDINAL_POSITION
    | PARAMETER_SPECIFIC_CATALOG
    | PARAMETER_SPECIFIC_NAME
    | PARAMETER_SPECIFIC_SCHEMA
    | PRECISION
    | RETURNED_CARDINALITY
    | RETURNED_LENGTH
    | RETURNED_OCTET_LENGTH
    | SCALE
    | SCOPE_CATALOG
    | SCOPE_NAME
    | SCOPE_SCHEMA
    | TYPE
    | UNNAMED
    | USER_DEFINED_TYPE_CATALOG
    | USER_DEFINED_TYPE_NAME
    | USER_DEFINED_TYPE_SCHEMA
<simple value specification 2>::= <simple value specification>
<item number>::= <simple value specification>
<get descriptor statement>::=
    GET [ SQL ] DESCRIPTOR <descriptor name> <get descriptor information>

```

```

<get descriptor information>::=
    <get header information> [ { <comma> <get header information> }... ]
    |
    VALUE <item number> <get item information> [ { <comma> <get item
information> }... ]
<get header information>::=
    <simple target specification 1> <equals operator> <header item name>
<simple target specification 1>::= <simple target specification>
<get item information>::=
    <simple target specification 2> <equals operator> <descriptor item name>
<simple target specification 2>::= <simple target specification>
<prepare statement>::=
    PREPARE <SQL statement name> FROM <SQL statement variable>
<SQL statement name>::=
    <statement name>
    |
    <extended statement name>
<statement name>::= <identifier>
<extended statement name>::=[ <scope option> ] <simple value specification>
<SQL statement variable>::= <simple value specification>
<deallocate prepared statement>::=DEALLOCATE PREPARE <SQL statement name>
<describe statement>::= <describe input statement> | <describe output statement>
<describe input statement>::=
    DESCRIBE INPUT <SQL statement name> <using descriptor> [ <nesting option> ]
<using descriptor>::=USING [ SQL ] DESCRIPTOR <descriptor name>
<nesting option>::=WITH NESTING | WITHOUT NESTING
<describe output statement>::=
    DESCRIBE [ OUTPUT ] <described object> <using descriptor> [ <nesting option> ]
<described object>::=
    <SQL statement name> | CURSOR <extended cursor name> STRUCTURE
<extended cursor name>::=[ <scope option> ] <simple value specification>
<execute statement>::=
    EXECUTE <SQL statement name> [ <result using clause> ] [ <parameter using clause> ]
<result using clause>::= <output using clause>
<output using clause>::= <into arguments> | <into descriptor>
<into arguments>::=INTO <into argument> [ { <comma> <into argument> }... ]
<into argument>::= <target specification>
<dynamic parameter specification>::= <question mark>
<embedded variable specification>::= <embedded variable name> [ <indicator variable> ]
<indicator variable>::=[ INDICATOR ] <embedded variable name>
<into descriptor>::=INTO [ SQL ] DESCRIPTOR <descriptor name>
<parameter using clause>::= <input using clause>
<input using clause>::= <using arguments> | <using input descriptor>
<using arguments>::=USING <using argument> [ { <comma> <using argument> }... ]
<using argument>::= <general value specification>
<using input descriptor>::= <using descriptor>
<execute immediate statement>::=EXECUTE IMMEDIATE <SQL statement variable>
<SQL dynamic data statement>::=
    <allocate cursor statement>
    |
    <dynamic open statement>
    |
    <dynamic fetch statement>
    |
    <dynamic close statement>
    |
    <dynamic delete statement: positioned>
    |
    <dynamic update statement: positioned>
<allocate cursor statement>::=ALLOCATE <extended cursor name> <cursor intent>
<cursor intent>::= <statement cursor> | <result set cursor>
<statement cursor>::=
    [ <cursor sensitivity> ] [ SCROLL ] CURSOR [ WITH HOLD ] [ WITH RETURN
]
    FOR <extended statement name>
<result set cursor>::=

```

३८१ SQL 99 _BNF

```

FOR PROCEDURE <specific routine designator>
<dynamic open statement>::=
    OPEN <dynamic cursor name> [ <input using clause> ]
<dynamic cursor name>::= <cursor name> | <extended cursor name>
<dynamic fetch statement>::=
    FETCH [ [ <fetch orientation> ] FROM ] <dynamic cursor name> <output using
clause>
<dynamic close statement>::=CLOSE <dynamic cursor name>
<dynamic delete statement: positioned>::=
    DELETE FROM <target table> WHERE CURRENT OF <dynamic cursor name>
<dynamic update statement: positioned>::=
    UPDATE <target table> SET <set clause list> WHERE CURRENT OF <dynamic
cursor name>
--p
--small
--i
Note that <double period> must be a pair of period characters with no intervening space, not a pair of
period symbols separated by arbitrary white space. Normally, the lexical analyzer would return <double
period> as a symbol.
--/i
--/small
--/p
<double period>::= <period> <period>
<direct SQL statement>::= <directly executable statement> <semicolon>
<directly executable statement>::=
    | <direct SQL data statement>
    | <SQL schema statement>
    | <SQL transaction statement>
    | <SQL connection statement>
    | <SQL session statement>
    | <direct implementation-defined statement>
<direct SQL data statement>::=
    | <delete statement: searched>
    | <direct select statement: multiple rows>
    | <insert statement>
    | <update statement: searched>
    | <temporary table declaration>
<direct select statement: multiple rows>::= <query expression> [ <order by clause> ]
<set catalog statement>::=SET <catalog name characteristic>
<catalog name characteristic>::=CATALOG <value specification>
<set schema statement>::=SET <schema name characteristic>
<schema name characteristic>::=SCHEMA <value specification>
<set names statement>::=SET <character set name characteristic>
<character set name characteristic>::=NAMES <value specification>
<set path statement>::=SET <SQL-path characteristic>
<SQL-path characteristic>::=PATH <value specification>
<set transform group statement>::=SET <transform group characteristic>
<transform group characteristic>::=
    | DEFAULT TRANSFORM GROUP <value specification>
    | TRANSFORM GROUP FOR TYPE <user-defined type> <value specification>
<direct implementation-defined statement>::=!! (See the Syntax Rules)
<embedded SQL declare section>::=
    <embedded SQL begin declare>
    [ <embedded character set declaration> ]
    [ <host variable definition>... ]
    <embedded SQL end declare>
    | <embedded SQL MUMPS declare>
<embedded SQL begin declare>::=
    <SQL prefix> BEGIN DECLARE SECTION [ <SQL terminator> ]

```

```

<SQL prefix> ::= EXEC SQL | <ampersand> SQL <leftparen>
<SQL terminator> ::= END-EXEC | <semicolon> | <rightparen>
<embedded character set declaration> ::=
    SQL NAMES ARE <character set specification>
<host variable definition> ::=
    | <Ada variable definition>
    | <C variable definition>
    | <COBOL variable definition>
    | <Fortran variable definition>
    | <MUMPS variable definition>
    | <Pascal variable definition>
    | <PL/I variable definition>
<Ada variable definition> ::=
    <Ada host identifier> [ { <comma> <Ada host identifier> } ... ] <colon>
    <Ada type specification> [ <Ada initial value> ]
<Ada type specification> ::=
    | <Ada qualified type specification>
    | <Ada unqualified type specification>
    | <Ada derived type specification>
<Ada qualified type specification> ::=
    Interfaces.SQL <period> CHAR [ CHARACTER SET [ IS ] <character set
specification> ]
    <leftparen> 1 <double period> <length> <rightparen>
    Interfaces.SQL <period> BIT <leftparen> 1 <double period> <length>
<rightparen>
    Interfaces.SQL <period> SMALLINT
    Interfaces.SQL <period> INT
    Interfaces.SQL <period> REAL
    Interfaces.SQL <period> DOUBLE_PRECISION
    Interfaces.SQL <period> BOOLEAN
    Interfaces.SQL <period> SQLSTATE_TYPE
    Interfaces.SQL <period> INDICATOR_TYPE
<Ada unqualified type specification> ::=
    CHAR <leftparen> 1 <double period> <length> <rightparen>
    BIT <leftparen> 1 <double period> <length> <rightparen>
    SMALLINT
    INT
    REAL
    DOUBLE_PRECISION
    BOOLEAN
    SQLSTATE_TYPE
    INDICATOR_TYPE
<Ada derived type specification> ::=
    | <Ada CLOB variable>
    | <Ada BLOB variable>
    | <Ada user-defined type variable>
    | <Ada CLOB locator variable>
    | <Ada BLOB locator variable>
    | <Ada user-defined type locator variable>
    | <Ada array locator variable>
    | <Ada REF variable>
<Ada CLOB variable> ::=
    SQL TYPE IS CLOB <leftparen> <large object length> <rightparen>
    [ CHARACTER SET [ IS ] <character set specification> ]
<Ada BLOB variable> ::=
    SQL TYPE IS BLOB <leftparen> <large object length> <rightparen>
<Ada user-defined type variable> ::=
    SQL TYPE IS <user-defined type> AS <predefined type>
<Ada CLOB locator variable> ::=

```

SQL 99 _BNF

```

SQL TYPE IS CLOB AS LOCATOR
<Ada BLOB locator variable>::=
    SQL TYPE IS BLOB AS LOCATOR
<Ada user-defined type locator variable>::=
    SQL TYPE IS <user-defined type name> AS LOCATOR
<Ada array locator variable>::=
    SQL TYPE IS <collection type> AS LOCATOR
<Ada REF variable>::=
    SQL TYPE IS <reference type>
<Ada initial value>::=
    <Ada assignment operator> <character representation>...
<Ada assignment operator>::= <colon> <equals operator>
<C variable definition>::=
    [<C storage class>][<C class modifier>]<C variable specification> <semicolon>
<C storage class>::=auto | extern | static
<C class modifier>::=const | volatile
<C variable specification>::= <C numeric variable> | <C character variable> | <C derived variable>
<C numeric variable>::=
    { long | short | float | double } <C host identifier> [ <C initial value> ]
    [ { <comma> <C host identifier> [ <C initial value> ] }... ]
<C initial value>::=
    <equals operator> <character representation>...
<C character variable>::=
    <C character type> [ CHARACTER SET [ IS ] <character set specification> ]
    <C host identifier> <C array specification> [ <C initial value> ]
    [ { <comma> <C host identifier> <C array specification> [ <C initial value> ]
}... ]
<C character type>::=char | unsigned char | unsigned short
<C array specification>::= <left bracket> <length> <right bracket>
<C derived variable>::=
    <C VARCHAR variable>
    <C NCHAR variable>
    <C NCHAR VARYING variable>
    <C CLOB variable>
    <C NCLOB variable>
    <C BLOB variable>
    <C bit variable>
    <C user-defined type variable>
    <C CLOB locator variable>
    <C BLOB locator variable>
    <C array locator variable>
    <C user-defined type locator variable>
    <C REF variable>
<C VARCHAR variable>::=
    VARCHAR [ CHARACTER SET [ IS ] <character set specification> ]
    <C host identifier> <C array specification> [ <C initial value> ]
    [ { <comma> <C host identifier> <C array specification> [ <C initial value> ]
}... ]
<C NCHAR variable>::=
    NCHAR [ CHARACTER SET [ IS ] <character set specification> ]
    <C host identifier> <C array specification> [ <C initial value> ]
    [ { <comma> <C host identifier> <C array specification> [ <C initial value> ]
}... ]
<C NCHAR VARYING variable>::=
    NCHAR VARYING [ CHARACTER SET [ IS ] <character set specification> ]
    <C host identifier> <C array specification> [ <C initial value> ]
    [ { <comma> <C host identifier> <C array specification> [ <C initial value> ]
}... ]
<C CLOB variable>::=

```

```

SQL TYPE IS CLOB <leftparen> <large object length> <rightparen>
[ CHARACTER SET [ IS ] <character set specification> ]
<C host identifier> [ <C initial value> ] [ { <comma> <C host identifier> [ <C
initial value> ] }... ]
<C NCLOB variable>::=
SQL TYPE IS NCLOB <leftparen> <large object length> <rightparen>
[ CHARACTER SET [ IS ] <character set specification> ]
<C host identifier> [ <C initial value> ] [ { <comma> <C host identifier> [ <C
initial value> ] }... ]
<C BLOB variable>::=
SQL TYPE IS BLOB <leftparen> <large object length> <rightparen>
<C host identifier> [ <C initial value> ] [ { <comma> <C host identifier> [ <C
initial value> ] }... ]
<C bit variable>::=
BIT <C host identifier> <C array specification> [ <C initial value> ]
[ { <comma> <C host identifier> <C array specification> [ <C initial value> ]
}... ]
<C user-defined type variable>::=
SQL TYPE IS <user-defined type> AS <predefined type>
<C host identifier> [ <C initial value> ] [ { <comma> <C host identifier> [ <C
initial value> ] }... ]
<C CLOB locator variable>::=
SQL TYPE IS CLOB AS LOCATOR
<C host identifier> [ <C initial value> ] [ { <comma> <C host identifier> [ <C
initial value> ] }... ]
<C BLOB locator variable>::=
SQL TYPE IS BLOB AS LOCATOR
<C host identifier> [ <C initial value> ] [ { <comma> <C host identifier> [ <C
initial value> ] }... ]
<C array locator variable>::=
SQL TYPE IS <collection type> AS LOCATOR
<C host identifier> [ <C initial value> ] [ { <comma> <C host identifier> [ <C
initial value> ] }... ]
<C user-defined type locator variable>::=
SQL TYPE IS <user-defined type> AS LOCATOR
<C host identifier> [ <C initial value> ] [ { <comma> <C host identifier> [ <C
initial value> ] }... ]
<C REF variable>::=
SQL TYPE IS <reference type>

<COBOL variable definition>::=
{01|77} <COBOL host identifier> <COBOL type specification>
[ <character representation>... ] <period>
<COBOL type specification>::=
<COBOL character type>
|
<COBOL national character type>
|
<COBOL bit type>
|
<COBOL numeric type>
|
<COBOL integer type>
|
<COBOL derived type specification>
<COBOL character type>::=
[ CHARACTER SET [ IS ] <character set specification> ]
{ PIC | PICTURE } [ IS ] { X [ <leftparen> <length> <rightparen> ] }...
<COBOL national character type>::=
[ CHARACTER SET [ IS ] <character set specification> ]
{ PIC | PICTURE } [ IS ] { N [ <leftparen> <length> <rightparen> ] }...
<COBOL bit type>::=
{ PIC | PICTURE } [ IS ] { X [ <leftparen> <length> <rightparen> ] }...
USAGE [ IS ] BIT

```

३८० SQL 99 _BNF

```

<COBOL numeric type>::=
    { PIC | PICTURE } [ IS ] S <COBOL nines specification>
    [ USAGE [ IS ] ] DISPLAY SIGN LEADING SEPARATE
<COBOL nines specification>::= <COBOL nines> [ V [ <COBOL nines> ] ] V <COBOL nines>
<COBOL nines>::={ 9 [ <leftparen> <length> <rightparen> ] }...
<COBOL integer type>::= <COBOL binary integer>
<COBOL binary integer>::=
    { PIC | PICTURE } [ IS ] S <COBOL nines> [ USAGE [ IS ] ] BINARY
<COBOL derived type specification>::=
    | <COBOL CLOB variable>
    | <COBOL NCLOB variable>
    | <COBOL BLOB variable>
    | <COBOL user-defined type variable>
    | <COBOL CLOB locator variable>
    | <COBOL BLOB locator variable>
    | <COBOL array locator variable>
    | <COBOL user-defined type locator variable>
    | <COBOL REF variable>
<COBOL CLOB variable>::=
    [ USAGE [ IS ] ] SQL TYPE IS CLOB <leftparen> <large object length>
<rightparen>
    [ CHARACTER SET [ IS ] <character set specification> ]
<COBOL NCLOB variable>::=
    [ USAGE [ IS ] ] SQL TYPE IS NCLOB <leftparen> <large object length>
<rightparen>
    [ CHARACTER SET [ IS ] <character set specification> ]
<COBOL BLOB variable>::=
    [ USAGE [ IS ] ] SQL TYPE IS BLOB <leftparen> <large object length>
<rightparen>
<COBOL user-defined type variable>::=
    [ USAGE [ IS ] ] SQL TYPE IS <user-defined type> AS <predefined type>
<COBOL CLOB locator variable>::=
    [ USAGE [ IS ] ] SQL TYPE IS CLOB AS LOCATOR
<COBOL BLOB locator variable>::=
    [ USAGE [ IS ] ] SQL TYPE IS BLOB AS LOCATOR
<COBOL array locator variable>::=
    [ USAGE [ IS ] ] SQL TYPE IS <collection type> AS LOCATOR
<COBOL user-defined type locator variable>::=
    [ USAGE [ IS ] ] SQL TYPE IS <user-defined type name> AS LOCATOR
<COBOL REF variable>::=
    [ USAGE [ IS ] ] SQL TYPE IS <reference type>
<Fortran variable definition>::=
    <Fortran type specification> <Fortran host identifier> [ { <comma> <Fortran
host identifier> } ... ]
--p
--small
--i
The standard documents 'CHARACTER KIND = n' but there is no explanation of the italic 'n' that is
used. Presumably, it is an integer literal, hence <unsigned integer>.
--/i
--/small
--/p
<Fortran type specification>::=
    CHARACTER [ <asterisk> <length> ] [ CHARACTER SET [ IS ] <character set
specification> ]
    | CHARACTER KIND <equals operator> <unsigned integer> [ <asterisk>
<length> ] [ CHARACTER SET [ IS ] <character set specification> ]
    | BIT [ <asterisk> <length> ]
    | INTEGER

```

```

|      REAL
|      DOUBLE PRECISION
|      LOGICAL
|      <Fortran derived type specification>
<Fortran derived type specification>::=
|      <Fortran CLOB variable>
|      <Fortran BLOB variable>
|      <Fortran user-defined type variable>
|      <Fortran CLOB locator variable>
|      <Fortran BLOB locator variable>
|      <Fortran user-defined type locator variable>
|      <Fortran array locator variable>
|      <Fortran REF variable>
<Fortran CLOB variable>::=
|      SQL TYPE IS CLOB <leftparen> <large object length> <rightparen>
|      [ CHARACTER SET [ IS ] <character set specification> ]
<Fortran BLOB variable>::=
|      SQL TYPE IS BLOB <leftparen> <large object length> <rightparen>
<Fortran user-defined type variable>::=
|      SQL TYPE IS <user-defined type> AS <predefined type>
<Fortran CLOB locator variable>::=
|      SQL TYPE IS CLOB AS LOCATOR
<Fortran BLOB locator variable>::=
|      SQL TYPE IS BLOB AS LOCATOR
<Fortran user-defined type locator variable>::=
|      SQL TYPE IS <user-defined type name> AS LOCATOR
<Fortran array locator variable>::=
|      SQL TYPE IS <collection type> AS LOCATOR
<Fortran REF variable>::=
|      SQL TYPE IS <reference type>
<MUMPS variable definition>::=
|      <MUMPS numeric variable> <semicolon>
|      <MUMPS character variable> <semicolon>
|      <MUMPS derived type specification> <semicolon>
<MUMPS numeric variable>::=
|      <MUMPS type specification> <MUMPS host identifier>
|      [ { <comma> <MUMPS host identifier> } ... ]
<MUMPS type specification>::=
|      INT
|      DEC [ <leftparen> <precision> [ <comma> <scale> ] <rightparen> ]
|      REAL
<MUMPS character variable>::=
|      VARCHAR <MUMPS host identifier> <MUMPS length specification>
|      [ { <comma> <MUMPS host identifier> <MUMPS length specification> } ... ]
<MUMPS length specification>::=
|      <leftparen> <length> <rightparen>
<MUMPS derived type specification>::=
|      <MUMPS CLOB variable>
|      <MUMPS BLOB variable>
|      <MUMPS user-defined type variable>
|      <MUMPS CLOB locator variable>
|      <MUMPS BLOB locator variable>
|      <MUMPS user-defined type locator variable>
|      <MUMPS array locator variable>
|      <MUMPS REF variable>
<MUMPS CLOB variable>::=
|      SQL TYPE IS CLOB <leftparen> <large object length> <rightparen>
|      [ CHARACTER SET [ IS ] <character set specification> ]
<MUMPS BLOB variable>::=

```

		SQL TYPE IS BLOB <leftparen> <large object length> <rightparen>
<MUMPS user-defined type variable>::=		SQL TYPE IS <user-defined type> AS <predefined type>
<MUMPS CLOB locator variable>::=		SQL TYPE IS CLOB AS LOCATOR
<MUMPS BLOB locator variable>::=		SQL TYPE IS BLOB AS LOCATOR
<MUMPS user-defined type locator variable>::=		SQL TYPE IS <user-defined type name> AS LOCATOR
<MUMPS array locator variable>::=		SQL TYPE IS <collection type> AS LOCATOR
<MUMPS REF variable>::=		SQL TYPE IS <reference type>
<Pascal variable definition>::=		<Pascal host identifier> [{ <comma> <Pascal host identifier> } ...] <colon>
		<Pascal type specification> <semicolon>
<Pascal type specification>::=		PACKED ARRAY <left bracket> 1 <double period> <length> <right bracket>
		OF CHAR [CHARACTER SET [IS] <character set specification>]
OF BIT		PACKED ARRAY <left bracket> 1 <double period> <length> <right bracket>
		INTEGER
		REAL
		CHAR [CHARACTER SET [IS] <character set specification>]
		BIT
		BOOLEAN
		<Pascal derived type specification>
<Pascal derived type specification>::=		
		<Pascal CLOB variable>
		<Pascal BLOB variable>
		<Pascal user-defined type variable>
		<Pascal CLOB locator variable>
		<Pascal BLOB locator variable>
		<Pascal user-defined type locator variable>
		<Pascal array locator variable>
		<Pascal REF variable>
<Pascal CLOB variable>::=		SQL TYPE IS CLOB <leftparen> <large object length> <rightparen>
		[CHARACTER SET [IS] <character set specification>]
<Pascal BLOB variable>::=		SQL TYPE IS BLOB <leftparen> <large object length> <rightparen>
<Pascal user-defined type variable>::=		SQL TYPE IS <user-defined type> AS <predefined type>
<Pascal CLOB locator variable>::=		SQL TYPE IS CLOB AS LOCATOR
<Pascal BLOB locator variable>::=		SQL TYPE IS BLOB AS LOCATOR
<Pascal user-defined type locator variable>::=		SQL TYPE IS <user-defined type name> AS LOCATOR
<Pascal array locator variable>::=		SQL TYPE IS <collection type> AS LOCATOR
<Pascal REF variable>::=		SQL TYPE IS <reference type>
<PL/I variable definition>::=		{ DCL DECLARE }
		{ <PL/I host identifier> <leftparen> <PL/I host identifier> [{ <comma> <PL/I
host identifier> } ...] <rightparen> }		
		<PL/I type specification> [<character representation> ...] <semicolon>
<PL/I type specification>::=		{ CHAR CHARACTER } [VARYING] <leftparen> <length> <rightparen>
		[CHARACTER SET [IS] <character set specification>]
		BIT [VARYING] <leftparen> <length> <rightparen>
		<PL/I type fixed decimal> <leftparen> <precision> [<comma> <scale>]
<rightparen>		
		<PL/I type fixed binary> [<leftparen> <precision> <rightparen>]

```

|          <PL/I type float binary> <leftparen> <precision> <rightparen>
|          <PL/I derived type specification>
<PL/I type fixed decimal>::=
|          { DEC | DECIMAL } FIXED
|          FIXED { DEC | DECIMAL }
<PL/I type fixed binary>::=
|          { BIN | BINARY } FIXED
|          FIXED { BIN | BINARY }
<PL/I type float binary>::=
|          { BIN | BINARY } FLOAT
|          FLOAT { BIN | BINARY }
<PL/I derived type specification>::=
|          <PL/I CLOB variable>
|          <PL/I BLOB variable>
|          <PL/I user-defined type variable>
|          <PL/I CLOB locator variable>
|          <PL/I BLOB locator variable>
|          <PL/I user-defined type locator variable>
|          <PL/I array locator variable>
|          <PL/I REF variable>
<PL/I CLOB variable>::=
|          SQL TYPE IS CLOB <leftparen> <large object length> <rightparen>
|          [ CHARACTER SET [ IS ] <character set specification> ]
<PL/I BLOB variable>::=
|          SQL TYPE IS BLOB <leftparen> <large object length> <rightparen>
<PL/I user-defined type variable>::=
|          SQL TYPE IS <user-defined type> AS <predefined type>
<PL/I CLOB locator variable>::=SQL TYPE IS CLOB AS LOCATOR
<PL/I BLOB locator variable>::=SQL TYPE IS BLOB AS LOCATOR
<PL/I user-defined type locator variable>::=
|          SQL TYPE IS <user-defined type name> AS LOCATOR
<PL/I array locator variable>::=SQL TYPE IS <collection type> AS LOCATOR
<PL/I REF variable>::=SQL TYPE IS <reference type>
<embedded SQL end declare>::=
|          <SQL prefix> END DECLARE SECTION [ <SQL terminator> ]
<embedded SQL MUMPS declare>::=
|          <SQL prefix> BEGIN DECLARE SECTION
|          [ <embedded character set declaration> ] [ <host variable definition>... ]
|          END DECLARE SECTION <SQL terminator>
<embedded SQL host program>::=
|          <embedded SQL Ada program>
|          <embedded SQL C program>
|          <embedded SQL COBOL program>
|          <embedded SQL Fortran program>
|          <embedded SQL MUMPS program>
|          <embedded SQL Pascal program>
|          <embedded SQL PL/I program>
<embedded SQL Ada program>::=!! (See the Syntax Rules.)
<embedded SQL C program>::=!! (See the Syntax Rules.)
<embedded SQL COBOL program>::=!! (See the Syntax Rules.)
<embedded SQL Fortran program>::=!! (See the Syntax Rules.)
<embedded SQL MUMPS program>::=!! (See the Syntax Rules.)
<embedded SQL Pascal program>::=!! (See the Syntax Rules.)
<embedded SQL PL/I program>::=!! (See the Syntax Rules.)
<embedded SQL statement>::= <SQL prefix> <statement or declaration> [ <SQL terminator> ]
<statement or declaration>::=
|          <declare cursor>
|          <dynamic declare cursor>
|          <temporary table declaration>

```

३८९ SQL 99 _BNF

```

|         <embedded authorization declaration>
|         <embedded path specification>
|         <embedded transform group specification>
|         <embedded exception declaration>
|         <handler declaration>
|         <SQL-invoked routine>
|         <SQL procedure statement>
<dynamic declare cursor>::=
    DECLARE <cursor name> [ <cursor sensitivity> ] [ <cursor scrollability> ]
CURSOR
    [ <cursor holdability> ] [ <cursor returnability> ] FOR <statement name>
<embedded authorization declaration>::=DECLARE <embedded authorization clause>
<embedded authorization clause>::=
    SCHEMA <schema name>
    | AUTHORIZATION <embedded authorization identifier>
    | [ FOR STATIC { ONLY | AND DYNAMIC } ]
    | SCHEMA <schema name> AUTHORIZATION <embedded authorization
identifier>
    [ FOR STATIC { ONLY | AND DYNAMIC } ]
<embedded authorization identifier>::= <module authorization identifier>
<embedded path specification>::= <path specification>
<embedded transform group specification>::= <transform group specification>
<embedded exception declaration>::=WHENEVER <condition> <condition action>
<condition>::= <SQL condition>
--p
--small
--i
The standard documents 'SQLSTATE ( <SQLSTATE class value> [ , <SQLSTATE subclass value> ] )',
but it is not clear why the <leftparen> , <comma> and <rightparen> are not designated more accurately.
--/i
--/small
--/p
<SQL condition>::=
    <major category>
    | SQLSTATE <leftparen> <SQLSTATE class value> [ <comma> <SQLSTATE
subclass value> ] <rightparen>
    | CONSTRAINT <constraint name>
<major category>::=SQLEXCEPTION | SQLWARNING | NOT FOUND
<SQLSTATE class value>::= <SQLSTATE char> <SQLSTATE char> !! (See the Syntax Rules.)
<SQLSTATE char>::= <simple Latin upper case letter> | <digit>
<SQLSTATE subclass value>::= <SQLSTATE char> <SQLSTATE char> <SQLSTATE char> !! (See
the Syntax Rules.)
<condition action>::=CONTINUE | <go to>
<go to>::= { GOTO | GO TO } <goto target>
<goto target>::= <host label identifier> | <unsigned integer> | <host PL/I label variable>
<host label identifier>::=!! (See the Syntax Rules.)
<host PL/I label variable>::=!! (See the Syntax Rules.)
<interval primary>::=
    <value expression primary> [ <interval qualifier> ]
    | <interval value function>
<module authorization clause>::=
    SCHEMA <schema name>
    | AUTHORIZATION <module authorization identifier>
    | [ FOR STATIC { ONLY | AND DYNAMIC } ]
    | SCHEMA <schema name> AUTHORIZATION <module authorization
identifier>
    [ FOR STATIC { ONLY | AND DYNAMIC } ]
<preparable statement>::=
    <preparable SQL data statement>

```

```

|      <preparable SQL schema statement>
|      <preparable SQL transaction statement>
|      <preparable SQL control statement>
|      <preparable SQL session statement>
|      <preparable implementation-defined statement>
<preparable SQL data statement>::=
|      <delete statement: searched>
|      <dynamic single row select statement>
|      <insert statement>
|      <dynamic select statement>
|      <update statement: searched>
|      <preparable dynamic delete statement: positioned>
|      <preparable dynamic update statement: positioned>
<dynamic single row select statement>::= <query specification>
<dynamic select statement>::= <cursor specification>
<preparable dynamic delete statement: positioned>::=
    DELETE [ FROM <target table> ] WHERE CURRENT OF [ <scope option> ]
<cursor name>
<preparable dynamic update statement: positioned>::=
    UPDATE [ <target table> ] SET <set clause list> WHERE CURRENT OF [
<scope option> ] <cursor name>
<preparable SQL schema statement>::= <SQL schema statement>
<preparable SQL transaction statement>::= <SQL transaction statement>
<preparable SQL control statement>::= <SQL control statement>
<preparable SQL session statement>::= <SQL session statement>
<preparable implementation-defined statement>::=!! (See the Syntax Rules.)
--hr
--h2 END OF SQL-99 GRAMMAR
--/h2
--hr
--h2 Notes on Automatically Converting the SQL-99 Grammar to a YACC Grammar
--/h2
--p
Automatic translation of this grammar is non-trivial for a number of reasons. One is that the grammar
has a number of actions '!!
(See the Syntax Rules.)' which cannot be translated automatically. Another is that the grammar contains
rules that are usually better handled by the lexical analyzer than the grammar proper. Then there are
incomplete rules such as those which reference parts 6
to 10 (they are not defined; indeed, part 7, which was going to be SQL/Temporal, is in complete
abeyance), and the packages (almost completely undefined in the grammar). It is not clear whether these
can be ignored, or annotated out of the way.
--/p
--p
Another complication is automatically generating rules to deal with optional components and repetitive
components in the grammar. Square brackets do not contain alternative non-terminals; all those
expressions are contained within curly brackets within the square brackets. However, some square
brackets do contain alternative terminals. Curly brackets contain and group mandatory elements.
However, they are usually used in conjunction with the 'one or more times' repeater ellipsis '...' mark.
--/p
--hr

```

واژه نامه

انگلیسی به فارسی

A			
Abort	طرد	ALL-Key	تمام کلید
Abstract machine	ماشین انتزاعی	Alternate Key(A K)	کلید دیگر
Abstract Data Type (ADT)	نوع داده ای انتزاعی	Analogue	قیاسی
Abstract Data Type (ADT)	نوع داده مجرد	Anomaly	آنومالی
Abstractive	انتزاعی	Antisemijoin	ضد نیم پیوند
Abstractive construct	ساخت انتزاعی	Application perspective	دیدگاه کاربردی
Access rules	قواعد دستیابی	Application Programming Interface (API)	واسط برنامه سازی کاربرد
Access Method (AM)	شیوه دستیابی	Application developer	کاربرد ساز (بهره بردار)
Access right	حقوق دستیابی	Application Program (AP)	برنامه کاربردی (کاربردار)
Access strategy	دستیابی	Application	تبدیل برنامه
Accessability	دستیابی بهتر	Program conversion	های کاربردی
Accuracy	دقت	Application system	سیستم کاربردی
Action	عملی	Arity	آریتی
Active rule	قاعده فعال	Assertion	اظهار
Active Data Objects (ADO)	شیئی های داده ای فعال	Asset	سرمایه
Active Database Management System (ADBMS)	سیستم فعال مدیریت پایگاه داده	Association	بستگی
Active dictionary	دیکشنری فعال	Asymmetry	عدم تقارن
Active DBMS (ADBMS)	سیستم فعال	Atomic	تجزیه نشدنی
Activity diagram	نمودار فعالیت	Atomicity	تجزیه ناپذیری
Actor	کنشگر	Attribute	صفت
Ad hoc (Unplanned)	موردی تابع جمعی یا گروهی	Attribute features	جنبه های صفت
Aggregate function	گروهی	Attribute inheritance	وراثت صفت
Aggregation	تجمع	Attribute preserving	حافظه صفات محدودیت
		Attribute constraint	صفتی
		Attributes set	مجموعه صفات
		Augmentation	افزایش

Authorization	مجاز شماری	طراحی به کمک کامپیوتر-تولید به کمک کامپیوتر	
Authorization	شناسه مجاز	کد کامپیوتر	
Identifier (AuthID)	شماری	صفحت محاسبه شده	
Automatic Design Tool (ADT)	ابزار (امکان) طراحی خودکار	واسط در سطح فراخوان (CLI)	
Automatic navigation	ناوش خودکار	میدان کاندید	
Axiom	ارز آغاز (اصل موضوعه	کلید کاندید	
B			
Bachman diagram	نمودار با چمن افزارگان پشت	پوش کانونیک	
Back office server	صفحه	کار دینالیتی نسبت	
Back-end	پسا ماشین	کار دینالیتی تسلسلی	
Backend	خدمتگذار	مدیر کاتالوگ	
Dbserver	پایگاهی پسا	کاتالوگ	
Backup	پشتیبان	دسته بندی	
Bags	توده	دسته	
Base relation	رابطه مبنا	متمرکز	
Base attribute	مبنا	کنترل متمرکز	
Base Table (BT)	جدول مبنا (پایه) نوع داده ای	پایگاه متمرکز تکنیک تایید (تصدیق)	
Basic data type	مبنایی	Certification	دام شکاف محدودیت ناظر به ستون
Batch	یکجا	Chasm trap	نقطه واریسی
Batch mode	اسلوب یکجا	Check constraints	چمبرلین
Behavioural	رفتاری	Checkpointing	چن
Benchmarking	محک زنی	CHEMBERLIN	نمودار چن
Bi- structure	دو ساختاره	CHEN	رده
Bi-directional association	بستگی دو سویه	CHEN notation	نمودار رده
Body	پیکر (بنه)	Class	رده بندی
Bound	بسته ساختار داده ای	Class diagram	فراکرد (کلاز)
Box data structure	جعبه ای	Classification	طراحی بهتر و واضح
BOYCE	بویس	Clause	یک مشتری-چند خدمتگذار
Boyce-Codd Normal Form	صورت نرمال بایس-کاد	Clean design	کامپیوتر مشتری
Bridge	پل	Client / Multi-Server (C / MS)	مشتری / خدمتگذار
Browsing	گشت زنی	Client machine	پایگاه داده مشتری-
Business rule	قواعد تجاری	Client / Server (C/S)	خدمتگذار
Business rules	قواعد فعالیتها	Client / Server Database (C/S DB)	جهان بسته
Buttleneck	تنگنا	Closed World	تفسیر جهان
C			
C.J.DATE	دیت	Closed World	
Cache memory	حافظه نهان		

interpretation	بسته	Consistency	سازگاری
Closure	بستار	Constraint	محدودیت
Clustering	خوشه واری	Constraint specification language	زبان توصیف محدودیت
CODASYL data model	مدل داده ای کوداسیل	Constructor	ساختگر
CODD	کاد	CONTAINS	شامل است
Collection	گردایه	Containmentment	شمول
Command driven Command	هدایت شده با فرمان	Continuous operation	تداوم عملیات
Language Interface (CLI)	واسط زبان فرمان	Conventional	سنتی
Commit	تثبیت	Conventional system	سیستم متعارف
Communative	انتقال دهنده گی هم رسانش (ارتباط)	Correctness	صحت
Communication	قیاسوند	Cost-benefit analysis	تحلیل برآورد هزینه / مزایا
Comparand	سیستم رابطهای کامل	Cover	پوش
Complete relational system	پیچیده (مختلط)	CREATE view	ایجاد دید
Complex	نوع داده ای پیچیده	Cross join	پیوند ضرب
Complex Data Type (CDT)	پرسش پیچیده	Current AuthID	شناسه جاری
Complex query	سازند	Cursor	مکان نما
Component	مرکب	D	
Composite	ترکیب	Dangling	معلق
Composition	کمال محاسباتی	Data Sub Language (DSL)	زبان داده ای فرعی
Computational completeness	شمای ادراکی	Data compression	فشرده سازی داده ها
Conceptual schema	طراحی ادراکی	Data Structure(DS)	ساختار داده ای دسترسی (بهتر)
Conceptual design	سطح ادراکی (مفهومی)	Data accessibility	به داده ها
Conceptual level	افزونگی ادراکی	Data Administrator (DA)	مدیر داده
Conceptual redundancy	دید ادراکی	Data bank	بانک داده ها
Conceptual view	نگاشت ادراکی / داخلی	Data communication manager	مدیر انتقال داده
Conceptual / Internal mapping	همزمان	Data Control (DC)	کنترل داده
Concurrent	طرح اجرای همروند	Data Control Language (DCL)	دستورات کنترل داده ها
Concurrent schedule		Data conversion and data transfer	تبدیل و انتقال داده ها
Conference On Data System		Data Definition	تعریف داده
Langeage (CODASYL)	سیستم کوداسیل	Data Definition Language (DDL)	دستورات تعریف داده ها
Configuration	پیکربندی	Data encryption	نهان نگاری داده
Connection traps	دامهای پیوندی	Data entry	وارد کردن داده
Connector	پیوند دهنده	Data export	صدور داده ها

Data extraction	استخراج داده ها	support	پایگاه
Data Flow Diagram (DFD)	نمودار روند داده	Database Administrator (DBA)	مدیر پایگاه داده ها
Data import	ورودی داده ها	Database approach	مشی پایگاهی محدودیت
Data inconsistency	ناسازگاری داده	Database constraint	پایگاهی
Data independence	استقلال داده ای	Database environment management	مدیریت محیط پایگاه داده ها
Data Management System (DMS)	سیستم مدیریت داده ها	Database Machine (DBM)	ماشین پایگاه داده ها
Data Manager	مدیر داده ها دستور عملیات	Database Management System (DBMS)	سیستم مدیریت پایگاه داده ها ایمنی و حفاظت
Data Manipulation Language (DML)	در داده ها دستورات عملیات روی داده ها	Database Protection	پایگاه داده ها کیفیت پایگاه داده ها
Data marts	بازار داده ها	Database quality	خدمتگزار
Data materialization	ساختن داده	Database server	پایگاه داده ها
Data migration	کوچاندن داده	Datalogical	داده شناسانه
Data mining and knowledge discovery system	سیستم کاوی و کشف شناخت	Datalogical	دیتالوگ
Data Model	مدل داده ای نامگذاری داده ها	Data-object	شیء-داده فروشنندگان سیستمها
Data naming	جایدهی داده	DBMS vendors	ادغام در وب
Data placement	اشتراک داده ها	DBMS-web integration	مدیر انتقال داده ها
Data sharing	منبع داده ای	DC manager	بن بست
Data source	نوع داده ای	Deadlock	معماری نامتمرکز
Data type	گسترش پذیری	Decentralized architecture	تصمیم گیری
Data type extensibility	نوع داده ای الگوی استفاده	Decision making	پشتیبانی تصمیم
Data usage pattern	از داده ها سیستم انبارش داده ها	Decision support	سیستم پشتیبان تصمیم
Data Warehousing		Decision Support System (DSS)	اعلانی
Data Communication	داده رسانی	Declarative	تجزیه
Data keeping- data processing	داده داری- داده پردازي	Decomposition	استنتاج
Database	پایگاه داده	Deduction	پایگاه داده استنتاجی
Database growth	رشد پایگاه داده	Deductive database	طول پیش نهاده
Database Language (DBL)	زبان پایگاهی	Default length	مقدار پیش نهاده
Database space	فضای پایگاه داده وضعیت	Default value	واریسی با تأخیر
Database state	پایگاه داده تنظیم پایگاه داده ها	Deferred checking	بهنگام سازی با تأخیر
Database tuning	امکان مدیریتی برای مدیر	Deffered update	صفت معرف
Database administration		Defining attribute	درجه
		Degree	

Denormalization	کاهش درجه (سطح) نرمالیتی	Dynamic data	داده پویا
Dependency theory	تئوری وابستگی	Dynamic SQL	پویا SQL
Derived	مشتق	E	
Derived relation	رابطه مشتق	Elementary type	نوع ابتدایی
Descriptive	توصیفی	E-mail server	خدمتگزار پست الکترونیکی
Descriptive attribute	صفت توصیفی	Embedded	ادغام شده
Design by analysis	طراحی با تحلیل	Embedded Multi Valued Dependency (EMVD)	وابستگی چند مقدار ی ادغام شده
Design tool	ابزار طراحی	End user	کاربر پایانی
Destructor	ویرانگر	Engine	موتور
Database perspective	دیدگاه پایگاهی	Enhanced (Extended) ER	روشن EER
Determinant	دترمینان	Entity Identifier (EID)	شناسه موجودیت
Developement	ایجاد	Entity integrity rule	قاعده جامعیت موجودیتی
Device configuration	پیکربندی رسانه دو عملوندي	Entity relationship	ارتباط (ER)
Diadic	دو عملوندي	Entity type	نوع موجودیت
Direct materialization	ساختن مستقیم	Entity types set	مجموعه انواع موجودیتها
Discriminator	ممیزه	Equational Dependency (ED)	وابستگی برابری
Disjoint	مجزا	Equi-Join	پیوند به شرط تساوی
Diskless machine	ماشین بدون دیسک	ER diagram	نمودار ER
Distributed	توزیع شده	Essential	اساسی
Distributed Database (DDB)	پایگاه داده توزیع شده (پراکنده)	Event journaling	رویداد نگاری
Divide	تقسیم	Event-condition-action rules	قواعد رویداد-شرط-اقدام
Document server	مدارک و اسناد	Event Flow Diagram (EFD)	نمودار روند رویداد
Documentation	مستند سازی	Evolutionary Prototyping	نمونه سازی گسترشی
Documented procedure	رویه مستند	Execution plan	طرح اجرا
Domain	میدان	Executive	سیستم اطلاعات اجرایی
Domain - Key	صورت نرمال	Information System	سیستم اطلاعات اجرایی
Normal Form	میدان-کلیدی	Existance	وجود
Domain oriented	حساب میدانی	dependency	وابستگی وجود
Domain constraint	میدانی	Existential	سور وجودی
Domain variable	متغیر میدانی	Exploitation	بهره برداری
Downsizing	کاهش اندازه	Expression relation	رابطه عبارتی
Driver manager	مدیر درایور	Extended cartesian product	ضرب کاتزین گسترش یافته
DROP view	حذف دید	Extended data type	نوع داده ای گسترش یافته
Dummy record	رکورد ساختگی		
Dumping	نسخه برداری		
Durability	مانایی (دوام)		

Extension	گسترده
External level	سطح خارجی
External/Conceptual mapping (E / C)	نگاشت خارجی / ادراکی خارجی
External / External	خارجی
External schema	شمای خارجی
External view	دید خارجی
F	
Facts	واقعیات
FAGIN	فاگین
Fail (Failure)	عیب (نقص)
Fan trap	دام یک-چندی
Fault tolerance	تحمل خرابی نمودار وابستگیهای تابعی
FD 's diagram	تجزیه حافظ
FD' s preserving decomposition	وابستگیها
Federated	فدرال صورت پنجم
Fifth Normal Form	نرمال
File server	خدمتگذار فایل
Filing System (FS)	سیستم فایلینگ صورت نخست
First Normal Form	نرمال
Five- minute rule	قاعده پنج دقیقه
Flat	مسطح
Flat Linear record	رکورد خطی
Foreign Key (FK)	کلید خارجی
Form generator	مولد فرم هدایت شده با فرم
Form driven	فرم
Formal	صوری
Four(4)GL	زبان نسل چهارم
Fourth Normal Form	صورت چهارم
Form	نرمال
Free	آزاد
Front-end	پیشا
Full outer join	فرا پیوند کامل
Fully Functional Dependency (F FD)	وابستگی تابعی تام (کامل)
Fully relational system	سیستم رابطه‌ای تام
Functional Dependency (FD)	وابستگی تابعی

Functional analysis	تحلیل عملکردی
Functionality	عملیاتی
Fuzzy data mining	سیستم ژولایی
and knowledge discovery system	داده کاوی و کشف دانش
G	
Gantt chart	چارت گانت
Gateway	گذار
General purpose	همه منظوره
General unification	یکسان عمومی
Generalization	تعمیم
Global application	کاربردی
Global conceptual schema	سرناسری شمای ادراکی جامع
Global System Catalog (G S C)	کاتالوگ جامع
GRANT	اعطاء
Graphic driven	کاربر گرافیکی پیوند به شرط
Greater then Join	بزرگتر پیوند به شرط
Greater then or Equal	مساوی یا بزرگتر
Grow and shrink	رشد و کاهش
H	
Harmonic mean	متوسط هارمونیک
Heading	سر آید
HEATH theorem	قضیه هیث
Help	کاربریار
Heterogenous	ناهمگن پایگاه‌داده های کنترل سیستم های ناهمگن موازی
Heterogenous parrallel systems control database	داده نهان
Hidden data	ساختار داده ای
Hierarchycal Data Structure (HDS)	سلسله مراتبی
Hierarchycal Database (HDB)	پایگاه‌داده سلسله مراتبی
Homogenous	همگن
Horizontal decomposition	تجزیه افقی
Horizontal fragmentation	پارسازی (تقطیع) افقی
Host Language	زبان میزبان

Host database	پایگاه داده میزبان خدمتگزار
Host server	میزبان
Hybrid	پارسازي
fragmentation	ترکيبي
Hyper graph	ساختار داده اي
Database Structure	هایپرگرافي

I

Identifying	شناسايي
Identifying entity	موجودیت شناسا
Identifying relationship	ارتباط شناسا
Identity projection	پرتو هماني
Immediate	واریسي
checking	بلافاصله
Immediate update	بهنگام سازي بلافاصله
Implementation	پیاده سازي
Implementation diagram	نمودار پیاده سازي
Import / Export interface	واسط ورود / صدور
Inapplicable	غیر قابل اعمال
Inclusion	صورت نرمال
Dependency	وابستگی شمول
Normal Form	وابستگی شمول
Inclusion	وابستگی شمول
Dependency (ID)	وابستگی شمول
Independent	مستقل (خودکفا)
(Stand alone)	باز سازي شاخص
Index rebuilding	ساختن غیر مستقیم
Indirect materialization	استقراء
Induction	قواعد استنتاج
Inference rules	غیر صوري
Informal	فضاي اطلاعاتي
Information space	ذخیره و بازیابی اطلاعات
Information Storage and Retrieval (ISR)	سیستم اطلاعاتي
Information System	تکنولوژی اطلاعات
Information Technology (IT)	ارتباط دهندگی
Informative	پرسش دروني
Inner query	بهنگام سازي
Inplace updating	در جا
Instance	نمونه

Instances set	مجموعه نمونه
Integrated	مجموع-یکپارچه سیستم جامعه
Integrated system	(یکپارچه)
Integrity	جامعیت
Integrity rule	قاعده- محدودیت
(constraint)	جامعیت محدودیت
Integrity constraint	جامعیتی
Intelligent system	سیستم هوشمند
Intension	رابطه
Inter related	بهم مرتبط اندرکنش
Interaction	(تعامل)
Interactive	اندرکنشی
Interactive mode	اسلوب تعاملی
Intermediate relation	رابطه بینابینی
Internal level	سطح داخلی
Internal view	دید داخلی
Internal schema	شمای داخلی
Interoperability	تعامل
Interpretation	تفسیر
Intrinsic	ذاتی
Intuitive	شهودی
Inverted list data structure	ساختار داده اي لیستهای وارون
Irreducible	کاهش ناپذیر
Irreducibility	کاهش ناپذیری
IS-A	هست يك...
IS-A PART-OF	جزیی است از...
Islands of information	جزایر اطلاعاتي
Isolation	جدایی (انفراد)

J

Java-script	جاوا اسکریپت
Job	کار
Join	پیوند
Join attribute	صفت پیوند
Join Dependency	وابستگی پیوندی
Join selectivity factor	ضریب گزینش عملکرد پیوند

K

Key-based inclusion	وابستگی شمول مبتهی بر کلید
---------------------	-------------------------------

dependency		Logical sequence	توالی منطقی
Key preserving	دارای کلید	Logical unit of work	واحد منطقی کار
Key-Complete	صورت نرمال	Lookup table	جدول جستجو
Normal Form	کاملاً کلید	Loop trap	دام حلقه‌ای
Knowledge Base Management	سیستم مدیریت	Loosely coupled	پیوند سست
System (KBMS)	پایگاه شناخت	Lost decomposition	تجزیه با حذف
Knowledge-based system	سیستم شناختی		
	L	M	
	پایگاه داده	Main computer	کامپیوتر بزرگ
Large Database	بزرگ		معماری حول
Layer	لایه	Main frame centric	کامپیوتر بزرگ
Left augmentation	افزایش چپ	Main Memory	پایگاه داده‌های
Left outer join	فرا پیوند چپ	Database (MMDB)	حافظه اصلی
Less than Join	کوچکتر	Manipulative	عملیاتی
	پیوند به شرط	Many to many	چند به چند
Less than or Equal	مساوی یا	Map driven	هدایت شده با نقشه
Join	کوچکتر	Mapping	نگاشت
Library function	تابع کتابخانه‌ای	Materialized view	دید به عینیت در آمده
Link	پیوند	Meaning	معنا
Load	بارگذاری	Mediator	میانجی
Local	محلی	Member	عضو
Local application	کاربردی محلی	Membership condition	شرط عضویت
Local conceptual schema	شمای ادراسی محلی		هدایت شده با
Local database	پایگاه داده محلی	Menu driven	منو
	خود مختاری	Menu generator	مولد منو
Location anatomy	محلی	Message server	خدمت‌گزار پیام
	اندازه واحد قفل	Meta byte	متا بایت
Lock granularity	پذیر	Meta-constraint	متا محدودیت
	واحد قفل شدنی	Meta data	متا داده
Lock granular	(قفل پذیر)	Meta rules	متا قواعد
Locking	قفل گذاری	Micro real world	خرد جهان واقع
	تکنیک قفل	Middleware	میان افزار
Locking technique	گذاری	Mini computer	کامپیوتر متوسط
	قفل گذاری-تولید	Minimal	کپینه
Locking / Logging	فایلهای ثبت	Minimal relational system	سیستم رابطه‌ای حداقل
Logic		Missing data	داده نهست
programming	زبان برنامه	Missing information	اطلاع نهست
Language	سازی منطق	Mixed design method	روش ترکیبی
logical Data	استقلال داده‌ای	Mobile office	دفتر کار همراه
Independence	منطقی	Mobile database	خدمات
Logical construct	ساخت منطقی		
Logical design	طراحی منطقی		
Logical design refinement	بهبود طراحی منطقی		

server	پایگاه داده های موبایل	سازي
Mobile database system	سیستم پایگاه داده ای با معماری سیستم	N
Mobile database system	پایگاه های همراه	Named data items
Mobile DB	پایگاه داده موبایل	Named relation
Mobility and personal database	پایگاه داده های شخصی و همراه	Natural Join (NJ)
Module	واحد شناسه برای	Natural language driven
Module AuthID	تعریف ماجول	Natural language like
Monadic	تک عملوندي	Nature
Monitoring	نظارت	Navigation
Mono- server	تک خدمتگزار	Nested Relational Query Language (NRQL)
Multi database management system	سیستم مدیریت چند پایگاهی	Nested statement
Multi Database System (MDBS)	معماری چند پایگاهی	Nested query
Multi ring	چند حلقه ای	Nested relation
Multi- Client / Server (MC / S)	چند مشتری- یک خدمتگزار	Network
Multi- Client / Multi- Server(MC / MS)	چند مشتری- چند خدمتگزار	Network Data Structure (NDS)
Multi- media	چند رسانه ای	Network Database Language (NDL)
Multi database	چند پایگاهی	Network centric
Multi-Dimensional OLAR (MDOLAP)	سیستم پردازش تحلیلی برخط	Network topology
Multimedia-data mining	سیستم داده کاوی چند رسانه ای مجموعه چند	News server
Multimember set	عضوي	No action
Multiple inheritance	وراثت (توارث) چند گانه	Non database approach
Multiple-user access	دستیابی چند کاربردي	Non orthogonal design
Multiplicity	چندي	Non procedural
Multi query	چند پرسشی	No additive
Multi-server	چند خدمتگزار	Non-loss (Lossless)
Multi set	چند مجموعه	Nontrivial
Multi-user	چند کاربري	Normal relation
Multi valued	چند مقداري	Not Normal relation (NNR)
Multi valued Dependency (MVD)	وابستگی چند مقداري	Not-Equi-Join
Multi version concurrency	همروندی چند نسخه ای	Nullifying
Multi versioning	تکنیک چند نسخه	Null value
		O
		Object
		شیئي

Object Oriented Database Language	زبان پایگاه‌های شیئی گرا	environment	سفارش
Object Query Language (OQL)	زبان پرسش شیئی گرا	Ordering	نظم-آراستار
Object Oriented Language(OOL)	زبان شیئی گرا	Oriented graph	گراف جهت دار
Object behavior	رفتار شیئی مدل داده ای	Orthogonal design	طراحی متعامد
Object Data Model	شیئی گرا	Outer query	پرسش بیرونی
Object modeling	مدلسازی شیئی	Outer union	فرا اجتماع
Object Modeling Technique (OMT)	تکنیک مدلسازی شیئی	Outplace updating	بهنگام سازی برون از جا
Object Oriented Database Management System (OODBMS)	سیستم مدیریت پایگاه‌داده های شیئی گرا	Overhead	فزونکاری (بیشکاری)
Object Type	نوع شیئی	Overlapping	همپوشا
Object-Relational Database Management System (ORDBMS)	سیستم مدیریت پایگاه‌داده های شیئی - رابطه‌ای	Owner	مالک
Object-Relational Data Model (ORDM)	مدل رابطه‌ای شیئی گرا	P	
Occurrence	نمود	Page	صفحه
On-line	برخط (پیوسته)	Panel interface	واسط پانل
On Line Analytic Processing (OLAP)	پردازشهای تحلیلی بر خط	Parallel processing	پردازش موازی هدایت شده با
One to many	یک به چند	Parameter driven	پارامتر
One to one	یک به یک (یک بوده)	Parent-Child Link type (PCL)	نوع پیوند پدر - فرزندی
One fact - one relation	واقعیت (یک رابطه پرسش تک سطحی	Partial	جزیی
One level query	قاعده یک دقیقه	Partial Key	کلید جزئی مشارکت غیرالزامی (ناکامل)
One- minute rule	امکان پردازش تحلیلی بر خط	Partial participation	شرکت کننده دیکشنری غیر فعال
Online Analytical Processing (OLAP)	کنترلر بر خط اسلوب پیوسته (بر خط)	Participant	بیشترین کارایی
Online controller	عملوند	Passive dictionary	زمان اوج
Online mode	محیط عملیاتی	Peak performance	همپراز
Operand	اسلوب عملیاتی	peak time	کارایی
Operational environment	گزیدار (گزینه)	Peer-to-peer	همیشگی
Operational mode	محیط ورود	Performance	پایا
Option		Permanent (planned)	جدول ماندگار
Order-entry		Persistent	ماجولهای ذخیره شده ماندگار
		Persistent (Permanent) table	افزونی
		Persistent Stored Modules (PSM)	فیزیکی
		Physical redundancy	استقلال داده ای
		Physical Data Independence (PDI)	فیزیکی
		Physical design	طراحی فیزیکی
		Piece	تکه
		Plex	پلکس
		Pointer	نشان نما -

Portability	جابجایی پذیری	optimization	پرسشها
Positional ordering	نظم مکانی	Queries set	مجموعه پرسش
Post relational	پسارابطه‌ای	Query Language (QL)	زبان پرسش
Precompiled	پیش کامپایلر	Query By Example (QBE)	پرسش به کمک مثال
Predicate	مسند	Query modification	تغییر پرسش
Predicate calculus	مسندات صورت	Query optimizer	بهینه ساز پرسش
Prenex form	پیشوندی	Query processor	پردازشگر پرسش
Pre-relational	پیش رابطه‌ای	Query server	سرور پرسش
Prescriptive	دستوری	Query By Form (QBF)	پرسش به کمک فرم
Presentation	نمایش	Query optimization	بهینه سازی پرسش
Price per TPS	TPS		
Price / performance ratio	نسبت هزینه / کارایی	R	
Primary Key (PK)	کلید اصلی	Range variable	متغیر طیفی
Primary domain	میدان اصلی	Range query	پرسش طیفی
Prime attribute	صفت عمده	Real	واقعی
Printer server	خدمتگزار چاپ	Real world	جهان واقع
Privacy	محرمانگی	Real time application	کاربرد بلادرنگ
Privilege	امتياز		پایگاده بی درنگ
Problem context	متن مسئله	Real-time DBMS	اشیاء جهان واقع
Procedural	رویه ای	Real-world object	سیستم مدیریت پایگاده های بی درنگ
Procedure	رویه	Real-time database management system	نوع رکورد
Production rules	قواعد تولید	Record type	ترمیم
Program-data independence	استقلال برنامه از داده	Recovery	ارتباط بازگشتی
Programming Language (PL)	زبان برنامه سازی	Recursive relationship	افزونی
Programming completeness	کمال برنامه سازی	Redundancy	آرایه دیسکهای ارزان افزونه
Project	پرتو بهنگام سازی	Redundant Array of Inexpensive Disks (RAID)	صورت نرمال بی افزونی
Propagating update	منتشر شونده	Redundancy Free Normal Form	مهندسی دوباره
Proposition	گزاره	Reengineering	ارجاع
Protection	حفاظت نمونه نخست-	Reference	نمودار ارجاع
Prototype	نمونه سازی	Reference diagram	حلقه ارجاع
Pseudo transitivity	شبه تعدی	Reference cycle	گراف ارجاع
		Reference graph	مسیرگراف
		Reference path	مرجع
		Referenced	
Q			
Qualified association	بستگی مقید		
Qualifier	قید		
Quantifier	چندی نما		
Queries	بهنگام سازی		

Referencing	رجوع کننده		شونده
Referential integrity rule	قاعده جامعیتی ارجاعی	Request manager	مدیر درخواستها
Reflexive association	بستگی انعکاسی	Request modification	تغییر درخواست
Reflexivity	انعکاس	Requirements	مهندسی نیازها
Relationally completeness	کمال رابطه‌ای	Engineering (RE) Requirements	نمونه سازی
Relation assignment	انتساب رابطه	Prototyping	نیازی
Relation constraint	محدودیت رابطه‌ای	Research-Development (RD)	پژوهش و توسعه
Relation predicate	سند ارتباط	Response time	زمان پاسخدهی
Relation schema	شماهی رابطه	Responding procedure	رویه پاسخگو
Relation state	حالت رابطه	Restrict	تحدید‌کنش (مشروط)
Relation type	مولد نوع رابطه	Restricted Restriction - Union	صور ت نرمال
generator (Relval)	مقدار رابطه‌ای	Normal Form	تحدید- اجتماع
Relation variable (Relvar)	متغیر رابطه‌ای	Restriction predicate	مسند گزینش
Relational expression	عبارت جبر رابطه‌ای	Result relation	رابطه نتیجه
Relational synthesis	رابطه‌ای	REVOKE	لغو
Relational algebra	جبر رابطه‌ای	Right outer join	فرا پیوند راست مناسب
Relational assignment	انتساب رابطه‌ای	Rightsizing	سازای اندام
Relational calculus	حساب رابطه‌ای	Rissanen	ریسانن
Relational Data Structure (RDS)	ساختار داده‌ای ارتباطی	Role	نقش
Relational Database Management System (RDBMS)	سیستم مدیریت پایگاه داده‌های رابطه‌ای	Root only	فقط ریشه
Relationship type	نوع ارتباط	R-system	سیستم R
Reload	باز بارگذاری	Rule based optimizer	بهینه ساز مبتنی بر قاعده
Remote call	فراخوان از دور	Rules	قواعد
Remote data access	دستیابی از دور به داده ها	Run time manager	مدیر زمان اجرا
Remote data access	دستیابی به داده های دوردست	Run time supervisor	ناظر زمان اجرا
Remote Data Access System (RDAS)	سیستم با امکان دستیابی به داده ها از دور		
Renaming	دگرنامی		
Replica	نسخه		
Replication	نسخه سازی		
Report generator	مولد گزارش		
Repreting group	گروه تکرار		

Search argument	نشاندن جستجو	database system	جغرافیایی
Second Normal Form	صورت دوم نرمال	Special purpose	تک منظوره
Security	ایمنی	Specialization	تخصیص
Selective setup	برپایی گزینشی	Speedup	افزایش سرعت (کارایی)
Self dependency	خود وابستگی	Spread sheet	گستر برگ
Self-relationship	ارتباط با خود	Spurious	اطلاع جعلی
Self referencing	به خود رجوع کننده	SQL-Session	شناسه کار با SQL
Semantic	معنایی	AuthID	محدودیت
Semantic unit of data	واحد معنایی داده	State constraint	وضعیتی
Semantic concept	مفهوم معنایی محدودیت	Storage Definition Language (SDL)	زبان تعریف ذخیره سازی
Semantic constraint	معنایی	Stored procedure	رویه ذخیره شده
Semantic Data Modeling (SDM)	مدلسازی معنایی داده	Stored data	داده ذخیره شده
Semantic integrity constraint	محدودیت جامعیت معنایی بهینه ساز	Stored data	ذخیره شده رابطه ذخیره شده
Semantic optimizer	معنایی	Stored relation	قوی
Semantic rule	قواعد سمانتیک	Strong	ساختاری
Semi-join	نیم پیوند	Structural	کمال ساختاری
Semi structured	نیم ساختمند طرح اجرایی متوالی	Structural completeness	محدودیت ساختاری
Serial schedule	توالی پذیر	Structural constraint	میدان ساختمند ساختمند
Serializable	مدیر تماسهای اجرایی کاربران	Structural domain	رکورد ساختمند
Session manager	نوع مجموعه	Structured	زبان SQL
Set type	پرسش مجموعه	Structured record	زیر نوع موجودیت
Set-oriented query	برپایی	Structured query	پرسش فرعی
Setup	اشتراکی	Structured Query Language (SQL)	زیر جدول
Shared	زیر نوع مشترک	Sub entity type	زیر رده
Shared entity	عارضه جانبی	Sub query	زیر نوع
Side effect	ساده	Sub table	گروه بندی
Single	تک کاربری	Subtype	کامپیوتر خیلی
Single user (Personal)	تک مقداری	Subtype	ابرنکند
Single-valued	مانه (سایت) اندازه تنظیم	Summarize	زیر نوع
Site	شونده	Super computer	زیر جدول
Smart sizing	لحظه ای	Super Key (S. K)	زیر رده
Snapshot	تسهیلات	Superentity type	کلید جایگزین
Software facilities (tools)	نرم افزارهای	Supertable	نهان
Spatial and geographic	فضایی و	Supertype	تعریف شده در سیستم
		Surrogate	سیستم ساز - پیاده ساز
		System defined	
		System developer	

System specification	مشخصات سیستم	integrity	
System variable	متغیر سیستمی	Transaction Processing Performance Council (TPC)	سازمان TPC خدمت‌گزار تراکنشها مدیریت همروندی تراکنش محدودیت گذاری تراگذاری تعدي قواعد نامرئي بودن
T		Transaction server	تراکنشها
Tabular Data Structure (TDS)	ساختار داده اي جدولي	Transactions concurrency management	مدیریت همروندی تراکنش محدودیت گذاری تراگذاری تعدي قواعد نامرئي بودن
Tabular system	سیستم جدولی پایگامداده	Transition constraint	محدودیت گذاری تراگذاری تعدي قواعد نامرئي بودن
Tabular Database	جدولی	Transitive	تراگذاری تعدي قواعد نامرئي بودن
Task	وظیفه	Transitivity	تعدي قواعد نامرئي بودن
Temporal	زمانمند	Transparency rules	قواعد نامرئي بودن
Temporal Dependency	وابستگی زمانی نمودار موجودیت-ارتباط زمانمند	Trigger	رها نا (راه انداز)
Temporal ERD	ارتباط زمانمند	Trivial	نا مهم (بدیهی)
Temporal data	داده زمانمند پایگامداده	True proposition	گزاره درست
Temporal database	زمانمند	Tuple oriented	حساب تاپلی
Temporary table	جدول موقت	Tuple variable	متغیر تاپلی تکنیک قفل گذاری دو مرحله ای دو ردیفی سازگار نوع محدودیت نوع
Temporary view	دید موقت	Two Phase Locking (2PL)	دو مرحله ای
Test data	داده تستی	Two-tier	دو ردیفی
Test strategy	استراتژی تست	Type compatible	سازگار نوع
Third Normal Form	صورت سوم نرمال	Type constraint	محدودیت نوع
Three level architecture	معماری سه سطحی	U	
Three-schema architecture	معماری سه شمایی	Undefined	تعریف نشده
Three-tier	سه ردیفی	Unfederated	نا فدرال
Three-Valued Logic (3VL)	منطق سه ارزشی	Unidirectional association	بستگی یک سویه
Throughput	توان عملیاتی	Union type	نوع اجتماع
Tightly coupled	پیوند قوی	Uniqueness	یکتایی
Time stamping	تکنیک زمانمهر	Universal	سور همگانی (عمومی)
Tool developer	ابزار ساز	Universal concept	مفهوم همگانی
Top-down design method	روش بالا به پایین مشارکت الزامی	Universal Database Management System (UDBMS)	سیستم مدیریت پایگامداده های همگانی
Total participation	(کامل)	Universal Modeling Language (UML)	زبان عمومی مدل سازی
Tow-Valued Logic (2VL)	منطق دو ارزشی تسهیل برای ردگیری	Universal relation	رابطه عمومی
Trace facility	ردگیری	Universe of Discourse	جهان مورد نظر
Transactions Log Files (T. L. F)	فایلهای ثبت تراکنشها	Universe of relations	مجموعه رابطه های ممکن
Transaction	تراکنش	Unknown	ناشناخته
Transaction	جامعیت تراکنش		

Unload	خالی کردن
Unsafe expression	عبارت نامطمئن
Unstructured	نا ساختمند
Upgradeability	ارتقاء
Upsizing	افزایش اندازه نمودار مورد
Use case diagram	کاربرد
User Friendly Interface (U F I)	واسط کاربر پسند
User groups	واسط کاربری
User defined rule	قواعد کاربری
User usage monitoring	امکان نظارت بر عملیات کاربران تعریف شده
User defined	توسط کاربر
User Interface (UI)	واسط کاربر یکپارچه سازی
Users' Ers integration	نمودار های ER کاربران

V

	اعمال قواعد
Value -based	ایمنی مبتنی بر
security constraint	مقادیر
Vector	بردار
Version	نگارش
Vertical decomposition	تجزیه عمودی
Vertical fragmentation	پار سازی عمودی
Very Large Database (VLDB)	پایگاه داده خیلی بزرگ
Very large information integrated system	سیستم اطلاعاتی بسیار بزرگ
Very large distributed information system	سیستم اطلاعات توزیع شده بسیار قابلیت بهنگام
View updatability	سازش دید

View Definition Language (VDL)	زبان تعریف دید
View definition condition (s)	شرط تعریف داده
View derivation	اشتقاق دید
View on view	دید روی دید
Virtual Child	فرزند مجازی
Virtual record type	نوع رکورد مجازی
Virtual relation	رابطه مجازی
Virtual table	جدول مجازی
Virtual attribute	صفت مجازی موجودیت
Virtual entity	مجازی
Virtual Parent	
Child Link (VPCL)	تکنیک VPCL
Virtual Parent (V P)	پدر مجازی
Visual-basic -script	وی-بی-اسکرپت
Voice driven	کاربر صوتی

W

Weak	ضعیف
Web DB	پایگاه داده وب
Well designed relation	رابطه خوش طرح
Well Formed Formula (WFF)	فرمول خوش ساخت
Work station	ایستگاه کار
Work station	ایستگاه کاری
Working table	جدول کاری
Workload	بار کاری
World Wide Web (W W W)	شبکه جهانی اطلاع رسانی
World Wide Web Database system (WWW-DB)	سیستم پایگاه داده در شبکه جهانی اطلاع رسانی

واژه نامه

فارسی به انگلیسی

Essential	اساسی	Redundant	آ
Data extraction	استخراج داده ها	Array of Inexpensive Disks (RAID)	آرایه دیسکهای ارزان افزونه
Test strategy	استراتژی تست	Arity	آریتی
Access strategy	استراتژی دستیابی	Free	آزاد
Induction	استقراء	Anomaly	آنومالی
Program-data independence	استقلال برنامه از داده	Super Key (SK)	ابرکلید
Data independence	استقلال داده ای	Automatic Design Tool	ابزار طراحی خودکار
Physical Data Independence	استقلال داده ای فیزیکی	Tool developer	ابزار ساز
logical Data Independence	استقلال داده ای منطقی	Design tool	ابزار طراحی
Deduction	استنتاج	Nested statement	احکام مرکب (تو در تو)
Online mode	اسلوب پیوسته (بر خط)	DBMS-web integration	ادغام در وب
Interactive mode	اسلوب تعاملی	Embedded	ادغام شده
Operational mode	اسلوب عملیاتی	Self-relationship	ارتباط با خود
Batch mode	اسلوب یکجا	Recursive relationship	ارتباط بازگشتی
Data sharing	اشتراک داده ها	Informative Identifying relationship	ارتباط دهنده گی
Shared	اشتراکی	Upgradability	ارتقاء
View derivation	اشتقاق دید	Reference	ارجاع
Real-world object	اشیاء جهان واقع	Axiom	اصل موضوعه
Spurious	اطلاع جعلی (اضافی)		
Missing	اطلاع نهست		

Development	ایجاد	information	
CREATE view	ایجاد دید	Assertion	اظهار
Work station	ایستگاه کار	GRANT	اعطاء
Work station	ایستگاه کاری	Declarative	اعلانی
Security	ایمنی	Value-based security	اعمال قواعد ایمنی مبتنی
Database Protection	حفاظت پایگاه‌داده‌ها	constraint	بر مقادیر
	ب	Back office server	افزارگان پشت صحنه
Workload	بارکاری	Augmentation	افزایش
Load	بارگذاری	Upsizing	افزایش اندازه
Reload	باز بارگذاری	Left augmentation	افزایش چپ
Data marts	بازار داده‌ها	Speedup	افزایش سرعت (کارایی)
Index rebuilding	بازسازی شاخص		افزایش مقیاس کار -
Data bank	بانک داده‌ها	Scale up	گسترش پذیری
Setup	برپایی	Redundancy	افزونگی
Selective setup	برپایی گزینشی	Physical redundancy	افزونگی فیزیکی
On-line	برخط (پیوسته)	Conceptual redundancy	افزونگی ادراکی
Vector	بردار	Data usage pattern	الگوی استفاده از داده‌ها
Application Program (AP)	برنامه کاربردی_ کاربردار	Privilege	امتياز
Closure	بستار	Online Analytical Processing	امکان پردازش تحلیلی بر خط
Qualified association	بستگی مقید	Database administration support	امکان مدیریتی برای مدیر پایگاه
Association	بستگی	User usage monitoring	امکان نظارت بر عملیات کاربران
Reflexive association	بستگی انعکاسی	Abstractive Relation assignment	انتزاعی
Bi-directional association	بستگی دو سویه	Relational assignment	انتساب رابطه
Unidirectional association	بستگی یک سویه	Commutative	انتساب رابطه‌ای
Bound	بسته	Smart sizing	انتقال دهندگی
Deadlock	بن بست	Lock granularity	اندازه تنظیم شونده
BOYCE	بویس	Interaction	اندازه واحد قفل پذیر
Self referencing Logical design refinement	به خود رجوع کننده	Interactive	اندركنش (تعامل)
Exploitation	بهره برداری	Reflexivity	اندركنشی
Inter related	بهم مرتبط		انعکاس
In place updating	بهنگام سازی در جا		
Differed update	بهنگام سازی با تاخیر		

واژه نامه فارسی به انگلیسی ۴۰۹

(C/S DB)	خدمتگذار	Outplace updating	بهنگام سازی برون از جا
Mobile DB	پایگاه داده موبایل	Immediate update	بهنگام سازی بلافاصله
Host database	پایگاه داده میزبان	Queries optimization	بهنگام سازی پرسشها
Web DB	پایگاه داده وب	Propagating update	بهنگام سازی منتشر شونده
Main Memory Database	پایگاه داده های حافظه اصلی	Query optimizer	بهینه ساز پرسش
Mobility and personal database	پایگاه داده های شخصی و همراه	Rule based optimizer	بهینه ساز مبتنی بر قاعده
Heterogeneous parallel systems control database	پایگاه داده های کنترل سیستم های ناهمگن موازی	Semantic optimizer	بهینه ساز معنایی
Centralized database	پایگاه متمرکز	Query optimization	بهینه سازی پرسش
Virtual Parent	پدر مجازی	No additive Non-loss (Lossless)	بی حشو
Project Identity projection	پرتو	Peak performance	بی گمشدگی
Parallel processing	پرتو همانی		بیشترین کارایی
Query processor	پردازشگر پرسش	Horizontal fragmentation	پ
On Line Analytic Processing	پردازشهای تحلیلی بر خط	Hybrid fragmentation	پارسازی (تقطیع) افقی
Query By Form	پرسش به کمک فرم	Vertical fragmentation	پارسازی ترکیبی
Query By Example (QBE)	پرسش به کمک مثال	Persistent Database	پارسازی عمودی
Outer query	پرسش بیرونی	Large Database	پایا
Complex query	پرسش پیچیده	Local database	پایگاه داده
One level query	پرسش تک سطحی	Deductive database	پایگاه داده بزرگ
Nested query	پرسش تو در تو	Real-time DBMS	پایگاه داده محلی
Inner query	پرسش درونی	Distributed Database (DDB)	پایگاه داده استنتاجی
Range query	پرسش طیفی	Tabular Database	پایگاه داده بی درنگ
Subquery	پرسش فرعی	Very Large Database	پایگاه داده توزیع شده (پراکنده)
Set-oriented query	پرسش مجموعه ای	Temporal database	پایگاه داده جدولی
Research-Development (RD)	پژوهش-توسعه	Hierarchical Database (HDB)	پایگاه داده خیلی بزرگ
Back-end	پسا	Client / Server Database	پایگاه داده زمانمند
Post relational	پسا رابطه‌ای		پایگاه داده سلسله مراتبی
Backup	پشتیبان		پایگاه داده مشتری -

Application Program conversion	تبدیل برنامه های کاربردی	Decision support	پشتیبانی تصمیم
Data conversion and data transfer	تبدیل و انتقال داده ها	Bridge	پل
Commit	تثبیت	Cover	پوش
Decomposition	تجزیه	Canonical cover	پوش کانونیک
Horizontal decomposition	تجزیه افقی	Dynamic SQL	پویا SQL
Lost decomposition	تجزیه با حذف	Implementation	پیاده سازی
FD' s preserving decomposition	تجزیه حافظ وابستگیها	Complex	پیچیده (مختلط)
Vertical decomposition	تجزیه عمودی	Pre-relational	پیش رابطه‌ای
Atomicity	تجزیه ناپذیری	Precompiled	پیش کامپایلر
Atomic	تجزیه نشدنی	Front-end	پیشا
Aggregation	تجمع	Body	پیکر (بدنه)
Restrict	تحدید-گزینش	Configuration	پیکربندی
Cost-benefit analysis	تحلیل برآورد هزینه / مزایا	Device configuration	پیکربندی رسانه
Functional analysis	تحلیل عملکردی	Join	پیوند
Fault tolerance	تحمل خرابی	Link	پیوند
Specialization	تخصیص	Greater then Join	پیوند به شرط بزرگتر
Continuous operation	تداوم عملیات	Equi-Join	پیوند به شرط تساوی
Transaction	تراکنش		پیوند به شرط عدم تساوی
Transitive	تراگذاری	Not-Equi-Join	تساوی
Composition	ترکیب	Less than Join	پیوند به شرط کوچکتر
Recovery	ترمیم		پیوند به شرط مساوی یا بزرگتر
Cascade	تسلسلی	Greater then or Equal	پیوند به شرط مساوی یا کوچکتر
Trace facility	تسهیل برای ردگیری	Less than or Equal Join	پیوند دهنده
Software facilities (tools)	نرم افزار (ابزارها)	Connector	پیوند سست
Decision making	تصمیم گیری	Loosly coupled	پیوند ضربه
Interoperability	تعامل	Cross join	پیوند طبیعی
Transitivity	تعدی	Natural Join (NJ)	پیوند قوی
Data Definition	تعریف داده	Tightly coupled	ت
User defined	تعریف شده توسط	Dependency theory	تئوری وابستگی
		Aggregate function	تابع جمعی یا گروهی
		Libraey function	تابع کتابخانه ای

Logical sequence	توالی منطقی	کاربر	
Throughput	توان عملیاتی	تعریف شده در سیستم	System defined
Network topology	توپولوژی شبکه	تعریف شما	Schema definition
Bags	توده	تعریف نشده	Undefined
Distributed	توزیع شده	تعمیم	Generalization
Descriptive	توصیفی	تعویقی (مشروط)	Restricted Query
ج		تغییر پرسش	modification
Portability	جابجایی پذیری	تغییر درخواست	Request modification
Integrity	جامعیت	تغییر شما	Schema evolution
Transaction integrity	جامعیت تراکنش	تفسیر	Interpretation
Java-script	جاوا اسکریپت	تفسیر جهان بسته	Closed World interpretation
Data placement	جایدهی داده	تقسیم	Divide
Relational algebra	جبر رابطه‌ای	تک خدمتگذار	Mono- server
Isolation	جدایی (انفراد)	تک عملوندی	Monadic
Lookup table	جدول جستجو	تک کاربری	Single user (Personal)
Working table	جدول کاری	تک مقداری	Single-valued
Persistent (Permanent) table	جدول ماندگار	تک منظوره	Special purpose
Base Table	جدول مبنا(پایه)	تکنولوژی اطلاعات	Information Technology (IT)
Virtual table	جدول مجازی	تکنیک VPC	Virtual Parent Child Link
Temporary table	جدول موقت	تکنیک تایید (تصدیق)	Certification
Islands of information	جزایر اطلاعاتی	تکنیک چند نسخه سازی	Multiversioning
Partial	جزیی	تکنیک زمانمهر	Timestamping
IS-A PART-OF	جزیی است از...	تکنیک قفل گذاری	Locking technique
Attribute features	جنبه های صفت	تکنیک قفل گذاری دو مرحله ای	Two Phase Locking (2PL)
Closed World	جهان بسته		Object Modeling Technique
World Wide Web (W W W)	جهان تازه (شبکه جهانی اطلاع رسانی)	تکنیک مدلسازی شیئ	Piece
Universe of Discourse	جهان مورد نظر	تکه	ALL-Key
Real world	جهان واقع	تمام کلید	Database tuning
چ		تنظیم پایگاه داده ها	Buttleneck
Gantt chart	چارت گانت	تنگنا	Serializable
CHEMBERLIN	چمبرلین	توالی پذیر	

موبایل	CHEN	چن
Host server	Many to many	چند به چند
News server	Multi database	چند پایگاهی
Database server	Multi query	چند پرسشی
Query server	Multi ring	چند حلقه ای
Message server	Multi-server	چند خدمتگذار
Transaction server	Multi- media	چند رسانه ای
Printer server	Multi user	چند کاربری
File server	Multi set	چند مجموعه
Document server	Multi- Client /	
Micro real world	Multi-Server(MC / MS)	چند مشتری - چند خدمتگذار
Location anatomy	Multi- Client / Server(MC /S)	چند مشتری - یک خدمتگذار
Self dependency	Multi valued	چند مقداری
Clustering	Multiplicity	چندی
	Quantifier	چندی نما
Dynamic data		ح
Test data	Attribute preserving	حافظه صفات
Data keeping-data processing	Cache memory	حافظه نهان
Stored data	Relation state	حالت رابطه
Data	DROP view	حذف دید
Communication	Tuple oriented	حساب تاپلی
Temporal data	Relational calculus	حساب رابطه‌ای
Data logical	Domain oriented	حساب میدانی
Hidden data	Protection	حفاظت
Missing data	Access right	حقوق دستیابی
Key preserving	Reference cycle	حلقه ارجاع
Loop trap	Spatial and geographic database system	حیطه های فضایی و جغرافیایی
Chasm trap		خ
Fan trap	External /	
Connection traps	External	خارجی / خارجی
Determinant	Unload	خالی کردن
Degree	Mobile database server	خدمات پایگاه‌داده های
Data accessibility		
Data		

واژه نامه فارسی به انگلیسی ۴۱۳

Active dictionary	دیکشنری فعال	Manipulation Language	
	ذ	Data Definition Language	دستورات تعریف داده ها
Intrinsic	ذاتی	Data Manipulation Language	دستورات عملیات روی داده ها
Stored data	ذخیره شده	Data Control Language	دستورات کنترل داده ها
Information Storage and Retrieval (ISR)	ذخیره و بازیابی اطلاعات	Prescriptive	دستوری
	ر	Category	دسته
Intension	رابطه	Categorization	دسته بندی
Intermediate relation	رابطه بینابینی	Remote data access	دستیابی از دور به داده ها
Nested relation	رابطه تودر تو	Remote data access	دستیابی به داده های دوردست
Well designed relation	رابطه خوش طرح	Accessibility	دستیابی بهتر
Stored relation	رابطه ذخیره شده	Multiple-user access	دستیابی چند کاربردی
Expression relation	رابطه عبارتی	Mobile office	دفتر کار همراه
Universal relation	رابطه عمومی	Accuracy	دقت
Not Normal relation (NNR: N2R)	رابطه غیر نرمال	Renaming	دگرنامی
Base relation	رابطه مبنا	2- Decomposable	دو رابطه
Derived relation	رابطه مشتق	Two-tier	دو ردیفی
Named relation	رابطه نامدار	Bi- structure	دو ساختاره
Result relation	رابطه نتیجه	Dyadic	دو عملوندی
Normal relation	رابطه نرمال	C.J.DATE	دیت
Referencing	رجوع کننده	Data logical	دیتالوگ
Class	رده	Conceptual view	دید ادراکی
Classification	رده بندی	Materialized view	دید به عینیت در آمده (ساخته شده)
Database growth	رشد پایگاه داده	External view	دید خارجی
Grow and shrink	رشد و کاهش	Internal view	دید داخلی
Object behavior	رفتار شیء	View on view	دید روی دید
Behavioral	رفتاری	Temporary view	دید موقت
Flat Linear record	رکورد خطی مسطح	Database perspective	دیدگاه پایگاهی
Dummy record	رکورد ساختگی	Application perspective	دیدگاه کاربردی
Structured record	رکورد ساختمند	Passive dictionary	دیکشنری غیر فعال

Constraint specification language	زبان توصیف محدودیت	Enhanced (Extended) ER Top-down design method	روش EER روش بالا به پایین
Data Sub Language Network Database Language Object Oriented	زبان داده ای فرعی زبان سیستم شبکه ای	Mixed design method	روش ترکیبی
Language(OOL)	زبان شیء گرا	Relational synthesis Event journaling	روش سنتز رابطه‌ای رویداد نگاری
Universal Modeling Language (UML)	زبان عمومی مدل سازی	Procedure	رویه
Host Language	زبان میزبان	Procedural Responding procedure	رویه ای رویه پاسخگو
4GL	زبان نسل چهارم	Stored procedure Documented procedure	رویه ذخیره شده رویه مستند
Super table	زیر جدول	Trigger	رها نا (راه انداز)
Super type	زیر رده	Virtual relation	ز
Super entity type	زیر نوع	ZANIOLO	زابطه مجازی زانیولو
peak time	زمان اوج	Programming Language (PL) Structured Query Language (SQL)	زبان برنامه سازی زبان SQL
Response time	زمان پاسخدهی	Logic programming Language Database Language Object Oriented Database Language	زبان برنامه سازی منطق زبان پایگاهی زبان پایگاهی شیء گرا
Temporal	زمانمند	Query Language Nested relational Query language Object Query	زبان پرسش زبان پرسش با رابطه غیر نرمال
Sub table	زیر جدول	Language(OQL)	زبان پرسش شیء گرا
Subtype	زیر رده	View Definition Language	زبان تعریف دید
Subtype	زیر نوع	Storage Definition Language	زبان تعریف ذخیره سازی
Shared entity	زیر نوع مشترک		
Sub entity type	زیر نوع موجودیت		
س			
Abstractive construct	ساخت انتزاعی		
Logical construct	ساخت منطقی		
Data Structure Tabular Data Structure (TDS)	ساختار داده ای جدولی		
Relational Data Structure	ساختار داده ای ارتباطی		
Box data structure	ساختار داده ای جعبه ای		
Hierarchical Data Structure Network Data Structure (NDS)	ساختار داده ای سلسله مراتبی ساختار داده ای شبکه ای		

Executive Information System (EIS)	سیستم اطلاعات اجرایی	Inverted list data structure	ساختار داده ای لیستهای وارون
Very large distributed information system	سیستم اطلاعات توزیع شده بسیار	Hyper graph Database Structure	ساختار داده ای هایپرگرافی
Information System (IS)	سیستم اطلاعاتی	Structural Constructor	ساختاری ساختگر
Very large information integrated system	سیستم اطلاعاتی بسیار بزرگ	Structured Data materialization	ساختمند ساختن داده
Data Warehousing	سیستم انبارش داده ها	Indirect materialization	ساختن غیر مستقیم
Remote Data Access System	سیستم با امکان دستیابی به داده ها از دور	Direct materialization	ساختن مستقیم
Mobile database system	سیستم پایگاه داده ای با معماری	Single Type compatible	ساده سازگار نوع سازگاری
World Wide Web Database system (WWW-DB)	سیستم پایگاه داده در شبکه جهانی اطلاع رسانی	Consistency Transaction Processing Performance Council (TPC)	سازمان TPC
Mobile database system	سیستم پایگاههای همراه	Component	سازند
Multi-Dimensional OLAR	سیستم پردازش تحلیلی برخط	Heading	سر آیند
Decision Support System Integrated system	سیستم پشتیبان تصمیم سیستم جامعه (یکپارچه)	Asset Conceptual level	سرمایه سطح ادراکی (مفهومی)
Tabular system	سیستم جدولی	External level	سطح خارجی
Multimedia-data mining	سیستم داده کاوی چند رسانه ای	Internal level	سطح داخلی
Fully relational system	سیستم رابطه ای تام	Conventional Relation predicate	سنتی سند ارتباط
Minimal relational system	سیستم رابطه ای حداقل	Existential	سور وجودی
Complete relational system	سیستم رابطه ای کامل	Universal	سور همگانی (عمومی)
Fuzzy data mining and knowledge discovery system	سیستم ژولای داده کاوی و کشف دانش	Three-tier Data mining and knowledge discovery system	سه ردیفی سیستم داده کاوی و کشف شناخت
		Conference On Data System Language (CODASYL)	سیستم کوداسیل
		R-system	سیستم R

CONTAINS	شامل است	System developer	سیستم ساز (پیاده ساز)
Network	شبکه	Knowledge-based system	سیستم شناختی
Pseudo transitivity	شبه تعدی	Filing System	سیستم فایلینگ
Natural language like	شبه زبان طبیعی	Active DBMS	سیستم فعال
View definition condition (s)	شرط تعریف داده	Active Database Management System	سیستم فعال مدیریت پایگاه داده
Membership condition	شرط عضویت	Application system	سیستم کاربردی
Participant	شرکت کننده	Conventional system	سیستم متعارف
Schema	شما	Database Management System (DBMS)	سیستم مدیریت پایگاه داده ها
Conceptual schema	شمای ادراکی	Real-time database management system	سیستم مدیریت پایگاه داده های بی درنگ
Global conceptual schema	شمای ادراکی جامع	Relational Database Management System	سیستم مدیریت پایگاه داده های رابطه ای
Local conceptual schema	شمای ادراکی محلی	Object - Relational Database Management System	سیستم مدیریت پایگاه داده های شیئی - رابطه ای
External schema	شمای خارجی	Object Oriented Database Management System	سیستم مدیریت پایگاه داده های شیئی گرا
Internal schema	شمای داخلی	Universal Database Management System	سیستم مدیریت پایگاه داده های همگانی (عمومی)
Relation schema	شمای رابطه	Knowledge Base Management System (KBMS)	سیستم مدیریت پایگاه شناخت
Containment	شمول	Multi database management system	سیستم مدیریت چند پایگاهی
Identifying	شناسای شناسه برای تعریف	Data Management System (DMS)	سیستم مدیریت داده ها
Module AuthID	ماجول	Intelligent system	سیستم هوشمند
Schema AuthID	شناسه تعریف شما		ش
Current AuthID	شناسه جاری		
SQL-Session AuthID	شناسه کار با SQL		
Authorization Identifier	شناسه مجاز شماری		
Entity Identifier (EID)	شناسه موجودیت		
Intuitive	شهودی		
Object	شیئی		
Data-object	شیئی-داده		
Active Data Objects (ADO)	شیئی های داده ای فعال		
Access Method	شیوه دستیابی		
ص			

product	یافته	Correctness	صحت
Join selectivity factor	گزینه‌ب ضریب	Data export	صدور داده ها
Weak	عملکرد پیوند ضعیف	Attribute	صفت
ط		Descriptive attribute	صفت توصیفی
Conceptual design	طراحی ادراکی	Prime attribute	صفت عمده
Design by analysis	طراحی با تحلیل	Virtual attribute	صفت مجازی
	طراحی/تولید به کمک	Calculated attribute	صفت محاسبه شده
CAD/CAM	کامپیوتر	Join attribute	صفت پیوند
Clean design	طراحی بهتر و واضح	Defining attribute	صفت معرف
Physical design	طراحی فیزیکی	Page	صفحه
Orthogonal design	طراحی متعامد	Prenex form	صورت پیشوندی
Logical design	طراحی منطقی	Boyce-Codd Normal Form	صورت نرمال بایس-کاد
Non orthogonal design	طراحی نا متعامد	Redundancy	صورت نرمال بی افزونگی
Execution plan	طرح اجرا	Free Normal Form	صورت نرمال تحدید-اجتماع
Serial schedule	طرح اجرای متوالی	Restriction - Union Normal Form (RUNF)	صورت نرمال کاملاً کلید
Concurrent schedule	طرح اجرای همروند	Key-Complete Normal Form	صورت نرمال میدان-کلیدی
Abort	طرده	Domain - Key Normal Form	صورت نرمال وابستگی شمول
Default length	طول پیش نهاده	Inclusion Dependency Normal Form	صورت پنجم نرمال
ع		Fifth Normal Form	صورت چهارم نرمال
Side effect	عارضه جانبی	Fourth Normal Form	صورت دوم نرمال
Relational expression	عبارت جبر رابطه‌ای	Second Normal Form	صورت سوم نرمال
Safe expression	عبارت مطمئن	Third Normal Form	صورت نخست نرمال
Unsafe expression	عبارت نامطمئن	First Normal Form	صورتی
No action	عدم اقدام	Formal	ضد نیم پیوند
Asymmetry	عدم تقارن	ض	ضرب کاترین گسترش
Member	عضو	Anti semi join	
Operand	عملوند	Extended Cartesian	
Action	عملی		
Functionally	عملیاتی		
Manipulative	عملیاتی		
Fail (Failure)	عیب (نقص)		
غ			

theorem		Informal	غیر صوری
Locking	قفل گذاری	Inapplicable	غیر قابل اعمال
	قفل گذاری و تولید	ف	
Locking/logging	فایل‌های ثبت	FAGIN	فاگین
Rules	قواعد	Transactions	
Inference rules	قواعد استنتاج	Log Files	فایل‌های ثبت تراکنشها
Business rule	قواعد تجاری	Federated	فدرال
Production rules	قواعد تولید	Outer union	فرا اجتماع
Access rules	قواعد دسترسی	Left outer join	فرا پیوند چپ
	قواعد رویداد-شرط-	Right outer join	فرا پیوند راست
Event- condition -action rules	اقدام	Full outer join	فرا پیوند کامل
Semantic rule	قواعد سمانتیک	Remote call	فراخوان از دور
Business rules	قواعد فعالیتها	Clause	فراکرد (کلاز)
User defined rule	قواعد کاربری	Virtual Child	فرزند مجازی
Transparency rules	قواعد نامرئی بودن	Well Formed Formula (WFF)	فرمول خوش ساخت
Strong	قوی	DBMS vendors	فروشندهگان سیستمها
Analogue	قیاسی	Overhead	فزونکاری (بیشکاری)
Qualifier	قید	Data compression	فشرده سازی داده ها
ک		Information space	فضای اطلاعاتی
Catalogue	کاتالوگ	Database space	فضای پایگاه‌داده
Global System Catalog (GSC)	کاتالوگ جامع	Named data items	فقره داده نامدار
CODD	کاد	Root only	فقط ریشه
Job	کار	ق	
Performance	کارایی	View	قابلیت بهنگام سازش دید
Natural language driven	کاربر با زبان طبیعی	updatability	قاعده جامعیت
End user	کاربر پایانی	Entity integrity rule	موجودیتی
Voice driven	کاربر صوتی	Referential integrity rule	قاعده جامعیتی ارجاعی
Graphic driven	کاربر گرافیکی		قاعده (محدودیت)
Real time application	کاربرد بلادرنگ	Integrity rule (constraint)	جامعیت
Application developer	کاربرد ساز (بهره بردار)	Five- minute rule	قاعده پنج دقیقه
Global application	کاربردی سرتاسری	Active rule	قاعده فعال
Local application	کاربردی محلی	One- minute rule	قاعده یک دقیقه
		HEATH	قضیه هیث

Summarize	گروه بندی	Help	کاربر یار
Repeating group	گروه تکرار شونده	Cardinality	کاردینالیتی
Proposition	گزاره	Client machine	کامپیوتر مشتری
True proposition	گزاره درست	Main computer	کامپیوتر بزرگ
Option	گزیندار (گزینه)	Super computer	کامپیوتر خیلی بزرگ
Spread sheet	گستر برگ	Mini computer	کامپیوتر متوسط
Extension	گسترده	Downsizing	کاهش اندازه
Scalability	گسترش پذیری	De normalization	کاهش درجه (سطح) نرمالیتی
Data type extensibility	گسترش پذیری نوع داده ای	Irreducible	کاهش ناپذیر
Browsing	گشت زنی (گذار گری)	Irreducibility	کاهش ناپذیری
ل		Primary Key	کلید اصلی
Layer	لایه	Surrogate	کلید جایگزین نهان
Snapshot	لحظه ای	Partial Key	کلید جزئی
REVOKE	لغو	Foreign Key	کلید خارجی
م		Alternate Key	کلید دیگر
Persistent Stored Modules (PSM)	ماجولهای ذخیره شده ماندگار	Candidate Key	کلید کاندید
Abstract machine	ماشین انتزاعی	Programming completeness	کمال برنامه سازی
Diskless machine	ماشین بدون دیسک	Relationally completeness	کمال رابطه ای
Database Machine	ماشین پایگاه داده ها	Structural completeness	کمال ساختاری
	ماشین خدمتگذار	Computational completeness	کمال محاسباتی
Backend DB server	پایگاهی پسا	Data Control	کنترل داده
Owner	مالک	Centralized control	کنترل متمرکز
Durability	مانایی (دوام)	Online controller	کنترلر بر خط
Site	مانه (سایت)	Actor	کنشگر
Nature	ماهیت	Data migration	کوچاندن داده
Base attribute	مینا	Minimal	کهنه
Meta byte	متا بایت	Database quality	کیفیت پایگاه داده ها
Meta data	متا داده	گ	
Meta rules	متا قواعد	Gateway	گذار
Meta-constraint	متا محدودیت	Reference graph	گراف ارجاع
Tuple variable	متغیر تاپلی	Oriented graph	گراف جهت دار
Relation variable(Relvar)	متغیر رابطه ای	Collection	گردابه

Benchmarking	محک زنی	System variable	متغیر سیستمی
Local	محلی	Range variable	متغیر طیفی
Operational environment	محیط عملیاتی	Domain variable	متغیر میدانی
Order-entry environment	محیط ورود سفارش	Centralized	متمرکز
Data Model (DM)	مدل داده ای	Problem context	متن مسئله
Object Data Model (ODM)	مدل داده ای شیء گرا	Harmonic mean	متوسط هارمونیک
CODASYL data model	مدل داده ای کوداسیل	Authorization	مجاز شماری
Object-Relational Data Model	مدل رابطه‌ای شیء گرا	Integrated	مجتمع (یکپارچه)
Object modeling	مدلسازی شیئی	Disjoint	مجزا
Semantic Data Modeling	مدلسازی معنایی داده	Entity types set	مجموعه انواع موجودیتها
Data communication manager	مدیر انتقال داده ها	Queries set	مجموعه پرسش
DC manager	مدیر انتقال داده ها	Multimember set	مجموعه چند عضوی
Database Administrator	مدیر پایگاه‌داده ها	Universe of relations	مجموعه رابطه های ممکن
Session manager	مدیر تماسهای اجرایی کاربران	Attributes set	مجموعه صفات
Data Administrator	مدیر داده ها	Instances set	مجموعه نمونه
Data Manager	مدیر داده ها	Constraint	محدودیت
Driver manager	مدیر درایور	Semantic integrity constraint	محدودیت جامعیت معنایی
Request manager	مدیر درخواستها	Integrity constraint	محدودیت جامعیتی
Run time manager	مدیر زمان اجرا	Relation constraint	محدودیت رابطه‌ای
Catalog manager	مدیر کاتالوگ	Structural constraint	محدودیت ساختاری
Database environment management	مدیریت محیط پایگاه‌داده ها	Attribute constraint	محدودیت صفتی
Transactions concurrency management	مدیریت همروندی تراکنش	Transition constraint	محدودیت گذاری
Referenced	مرجع	Semantic constraint	محدودیت معنایی
Composite Independent (Stand alone)	مستقل (خودکفا)	Domain constraint	محدودیت میدانی
Documentation	مستند سازی	Check constraints	محدودیت ناظر به ستون
		Type constraint	محدودیت نوع
		State constraint	محدودیت وضعیتی
		3D- constraint	محدودیت D ³
		Database constraint	محدودیت پایگاهی
		Privacy	محرمانگی

Tow-Valued Logic (2VL)	منطق دو ارزشی	Flat	سطح
Three-Valued Logic (3VL)	منطق سه ارزشی	Predicate Restriction predicate	مسند
Engine	موتور	Predicate calculus	مسند گزینش
Entity relationship	موجودیت-ارتباط (ER)	Reference path	مسندات
Identifying entity	موجودیت شناسا	Total participation	مسیرگراف
Virtual entity	موجودیت مجازی	Partial participation	مشارکت الزامی (کامل)
Ad hoc (Unplanned)	موردی	Client/Server	مشارکت غیرالزامی
Form generator	مولد فرم	Derived System specification	(ناکامل)
Report generator	مولد گزارش	Database approach	مشتری / خدمتگذار
Menu generator	مولد منو	Non database approach	مشتق
Relation type generator	مولد نوع رابطه	Dangling Multi Database System (MDBS)	مشخصات سیستم
Nontrivial	مهم (مطرح)	Network centric	مشی پایگاهی
Reengineering Requirements Engineering	مهندسی دوباره	Main frame centric	مشی ناپایگاهی
Middleware	میان افزار	Three level architecture	معلق
Mediator	میانجی	Three-schema architecture	معماری چند پایگاهی
Domain	میدان	Decentralized architecture	معماری حول شبکه
Primary domain	میدان اصلی	Meaning	معماری حول کامپیوتر
Structural domain	میدان ساختمان	Semantic	بزرگ
Candidate domain	میدان کاندید	Semantic concept	معماری سه سطحی
Non procedural	نا رویه ای (ناروشمند)	Universal concept	معماری سه شمایی
Unstructured	نا ساختمان	Default value	معماری نامتمرکز
Unfederated	نا فدرال	Relation value (Relval)	معنا
Trivial	نا مهم (بدیهی)	Cursor	معنایی
Data inconsistency	ناسازگاری داده	Discriminator	مفهوم معنایی
Unknown	ناشناخته	Rightsizing	مفهوم همگانی
Run time supervisor	ناظر زمان اجرا	Data source	مقدار پیش نهاده
Data naming	نامگذاری داده ها		مقدار رابطه ای
Navigation	ناوش (غواصی)		مکان نما
Automatic navigation	ناوش خودکار		ممیزه

زمانمند		Heterogenous	ناهمگن
Use case diagram	نمودار مورد کاربرد	Cardinality ratio	نسبت کاردینالیتی
FD 's diagram	نمودار وابستگیهای تابعی	Price / performance ratio	نسبت هزینه / کارایی
Instance	نمونه	Replica	نسخه
Evolutionary Prototyping	نمونه سازی گسترشی	Dumping	نسخه برداری
Requirements Prototyping	نمونه سازی نیازی	Replication	نسخه سازی
Prototype	نمونه نخست-نمونه سازی	Pointer	نشان نما (اشاره گر)
Elementary type	نوع ابتدایی	Search argument	نشانوند جستجو
Union type	نوع اجتماع	Monitoring	نظارت
Relationship type	نوع ارتباط	Ordering	نظم-آراستار
Parent-Child Link type (PCL)	نوع پیوند پدر-فرزندی	Positional ordering	نظم مکانی
Data type	نوع داده ای	Role	نقش
Abstract Data Type (ADT)	نوع داده ای انتزاعی	Checkpointing	نقطه واریسی
Complex Data Type (CDT)	نوع داده ای پیچیده	Version	نگارش
Extended data type	نوع داده ای گسترش یافته	Mapping	نگاشت
Basic data type	نوع داده ای مبنایی	Conceptual / Internal mapping	نگاشت ادراکی / داخلی
Abstract Data Type (ADT)	نوع داده مجرد	External / Conceptual mapping	نگاشت خارجی / ادراکی
Record type	نوع رکورد	Presentation	نمایش
Virtual record type	نوع رکورد مجازی	Occurrence	نمود
Object Type	نوع شیء	ER diagram	نمودار ER
Set type	نوع مجموعه	Reference diagram	نمودار ارجاع
Entity type	نوع موجودیت	Bachman diagram	نمودار با چمن
Data encryption	نهان نگاری داده	Implementation diagram	نمودار پیاده سازی
Semi-join	نیم پیوند	CHEN notation	نمودار چن
Semi structured و	نیم ساختمانمند	Class diagram	نمودار رده
Equational Dependency	وابستگی برابری	Data Flow Diagram (DFD)	نمودار روند داده
Join	وابستگی پیوندی	Event Flow Diagram (EFD)	نمودار روند رویداد
Dependency Functional	وابستگی تابعی	Activity diagram	نمودار فعالیت
Dependency		Temporal ERD	نمودار موجودیت-ارتباط

واژه نامه فارسی به انگلیسی ۴۲۳

inheritance	گانه	Fully Functional Dependency	وابستگی تابعی تام (کامل)
Attribute inheritance	وراثت صفت	Multivalued Dependency	وابستگی چند مقداری
Data import	ورودی داده ها	Embedded Multi Valued Dependency	وابستگی چند مقداری ادغام شده
Database state	وضعیت پایگاه داده	Temporal Dependency	وابستگی زمانی
Task	وظیفه	Inclusion Dependency	وابستگی شمول
Visual-basic-script	وی-بی-اسکرپت	Key-based inclusion dependency	وابستگی شمول مبتنی بر کلید
Destructor	ویرانگر	Existance dependency	وابستگی وجود واحد
Parameter driven	هدایت شده با پارامتر	Module Schema generator	واحد تولید شما
Form driven	هدایت شده با فرم	Lock granular Semantic unit of data	واحد قفل شدنی
Command driven	هدایت شده با فرمان	Logical unit of work	واحد منطقی کار
Menu driven	هدایت شده با منو	Data entry	وارد کردن داده
Map driven	هدایت شده با نقشه	Deferred checking	واریسی با تاخیر
Price per TPS	هزینه به ازاء TPS	Immediate checking	واریسی بلافاصله
IS-A	هست یک...	Application Programming Interface (API)	واسط برنامه سازی کاربرد
Overlapping Communication	همپوشا	Panel interface	واسط پانل
Multiversion concurrency	همروندی چند نسخه ای	Call Level Interface	واسط در سطح فراخوان
Concurrent	همزمان	Command Language Interface	واسط زبان فرمان
Peer-to-peer	همپراز	User Interface	واسط کاربر
Homogenous General purpose	همگن همه منظوره	User Friendly Interface	واسط کاربر پسند
Permanent (planned)	همیشگی	User groups	واسط کاربری
Nullvalue	هیچمقدار	Import / Export interface	واسط ورود / صدور
Nullifing	هیچمقدار گذاری	Real	واقعی
One fact-one relation	یک واقعیت-یک رابطه	Facts	واقعیات
One to many	یک به چند	Multiple	وراثت (توارث) چند
One to one	یک به یک		
Client / Multi-Server (C/MS)	یک مشتری-چندخدمتگزار		
Users' Ers integration	یکپارچه سازی نمودارهای ER کاربران		

Uniqueness	یکتایی
Batch	یکجا
General unification	یکسان عمومی

منابع و ماخذ

1	The Relational Model for Database Management , 2 nd Edition CODD.E.F	Addison-Wesley 1990
2	Mastering SQL , 1 st Edition Gruber, Martin	Sybex 2000
3	A Guide to the SQL Standard , 4 th Edition DATE, C.J	Addison-Wesley 1997
4	ORACLE PL/SQL , 1 st Edition JOSEPH , C.Trezzo	Osborne 1999
5	Guide to Client/Server Databases , 2 nd Edition SALEMI , joe	Ziff-David Press 1996
6	Inside SQL server 2000, 1 st Edition Delaney , Kalen	Microsoft Press 2001
7	Database System Concepts , 4 th Edition SILBERSCHATZ , Henry	MC Graw-Hill 2002
8	مفاهیم بنیادی پایگاه داده ها , ویراست جدید روحانی رانکوهی ، سید محمد تقی	جلوه ۱۳۸۳
9	پایگاه داده ها , چاپ ششم مقیمی ، حمیدرضا	گسترش علوم پایه ۱۳۸۳
10	ذخیره و بازیابی اطلاعات: مقدمه ای بر سیستم و ساختار فایلها , چاپ نهم روحانی رانکوهی ، سید محمد تقی	جلوه ۱۳۷۹

