پایگاه داده‌ها

Database

جزوه مهندس حمید رضا نیرومند
http://niroomand.ir

توجه:

- استفاده از جزوه بدون حضور در کلاس‌ها پیشنهاد نمی‌شود.
- این یک انتشار غیررسمی و بدون بازبینی نهایی از جزوه و ویژه دانشجویان مهندس نیرومند است.
- این، نسخه ۱,۰,۰ از جزوه است و به مرور غنی‌تر خواهد شد. (لطفاً اشتباهات سهوی که مشاهده می‌کنید را به ایمیل info@niroomand.ir ارسال نمایید).

مقدمه:

Database و سیستم‌های مرتبط با آن به یک جزء حیاتی زندگی بشر تبدیل شده‌اند. بسیاری از فعالیت‌های روزمره ما با Database درگیر است. از جمله: اکثر کارهای بانکی، روال تهیه بلیط هواپیما و ... .

به این نوع از پایگاه داده‌ها که فقط با داده‌های عددی و متنی سروکار دارند در اصطلاح «**پایگاه داه سنتی**» (Traditional) گفته می‌شود، در حالیکه این روزها Multimedia DBs یا GIS و... قابلیت ذخیره سازی عکس‌ها، ویدئو و صدا را نیز دارد.

اما به هر حال، برای آموزش پایه‌ای پایگاه داده باید از پایگاه داده‌های سنتی استفاده کرد.

تعریف عمومی پایگاه داده:

ابتدا تشریح کلمه Database:

Data کلمه‌ای است که هم به صورت جمع و هم به صورت مفرد به کار می‌رود اما در حقیقت یک کلمه جمع است که مفرد آن Datum است، Base هم یعنی پایه، پایگاه.

Data: حقایق شناخته شده یا Fact که می‌تواند ثبت شود. مثل: نام، آدرس، شماره تلفن و ...

تعریف عمومی پایگاه داده:

Database: a Database is a collection of related data

پایگاه داده مجموعه‌ای از داده‌های مرتبط است.

مشخصات عمومی Database:

- یک Database نماد مفاهیمی در دنیای واقعیت است به همین خاطر به Database، مینی‌ورلد (miniworld، خرده جهان واقع) یا Universe Of Discourse (UOD) نیز می‌گویند.
- یک Database یک مجموعه منطقاً مرتبط با معانی غیر مرتبط است.
- مجموعه‌ای از داده‌های تصادفی (random)، Database نامیده نمی‌شود.
- Database برای یک هدف خاص طراحی و ساخته شده است.

تعریف سیستم مدیریت DBMS:

DBMS: DataBase Management System

یک پایگاه داده ممکن است به صورت دستی ایجاد شده باشد یا به صورت کامپیوتری مثل فایل Excel.

یک DBMS مجموعه‌ای از برنامه‌ها است که به کاربر امکان ایجاد و مدیریت پایگاه داده کامپیوتری را می‌دهد.

فواید استفاده از DBMS:

۱- کنترل افزوندگی داده‌ها

۲- کنترل سطح دسترسی‌ها

۳- امکان تهیه نسخه پشتیبان (Backup) و بازگردانی اطلاعات (Recovery)

۴- امکان استفاده همزمان چندین کاربر از پایگاه داده

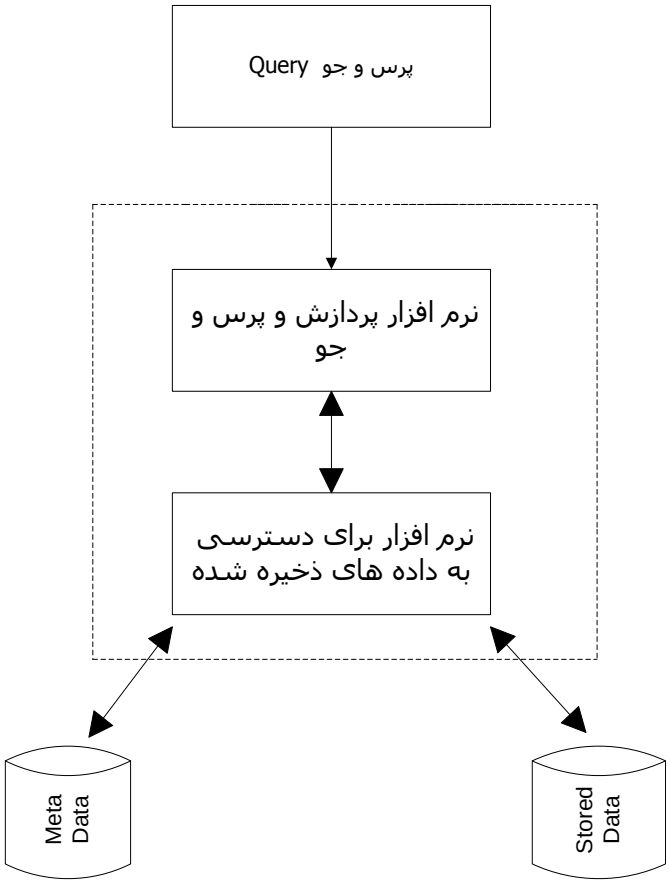
۵- تعیین قیود یکپارچگی

۶- و ...

تعریف سیستم پایگاه داده‌ای (DB System) :

به مجموعه پایگاه داده و سیستم مدیریت پایگاه داده، سیستم پایگاه داده‌ای گفته می‌شود.

شکل کلی محیط سیستم پایگاه داده‌ای:



مثال: پایگاه داده‌ای که شامل ترم‌ها و پروژه‌ها و دانشجوها و نفرات است:

Student	Name	stno	Mager
مشخصات	Ali	۴۳۴۵	cs
دانشجویان	Amin	۲۰۱۰	math

Course	name	Course name	credit	Department
رابطه دوره	CS101	DB	۳	cs
با رشته	M203	stat	۲	Math

Section	Section ID	Course Number	Semaster	Instractor	Year
مشخصات	۸۵	CS101	۸۸۲	Akbari	2010
هر دوره	۸۰	M203	۸۸۲	Ahmadi	2010

Grades	stno	Se ID	grade
نمرات	۳۳۸۰	۸۵	۲۰

تفاوت مشی فایلینگ و با مشی پایگاه داده‌ای:

مشی فایلینگ: None-DBMS Approach

مشی پایگاه داده‌ای: DBMS Approach

معایب مشی فایلینگ:

- تصور کنید اداره امور دانشجویی به اطلاعات دانشجویان نیاز داشته باشد، در مشی فایلینگ نام و نام خانوادگی و شماره دانشجویی و کلیه اطلاعات امور در فایلی ذخیره می‌شود. از طرفی اداره امور مالی نیز که برای محاسبه شهریه دانشجویان، وام و غیره ... به اطلاعاتی مثل نام، آدرس و بسیاری اطلاعات دیگر نیاز دارد که در یک فایل مجزا ذخیره می‌شود. پس بدون حضور سیستم پایگاه داده‌ای بسیاری از اطلاعات به صورت تکراری در فایل‌های مختلف ذخیره می‌شوند که هدر رفتن فضای حافظه را در پی خواهد داشت. به این عیب در اصلاح «**افزونگی داده**» گفته می‌شود.
- برای به روز کردن یک سری اطلاعات باید چندین فایل مختلف ویرایش شود.
- برای هر فایل یک برنامه کاربردی مخصوص به آن نیاز است که در نتیجه سیستم‌های کاربردی پراکنده‌ای خواهیم داشت.

- خطر بروز ناسازگاری داده
- عدم امکان اشتراک داده
- عدم امکان اشتراک داده
- عدم امکان اعمال ضوابط امنیتی روی کل داده‌های سازمان

مزایای مشی پایگاه داده‌ای:

برعکس معایب مشی فایلینگ خواهد بود...

افراد مرتبط با پایگاه داده‌ها

۱- Database Administrator

DBA وظیفه نظارت و مدیریت دو منبع اصلی سیستم یعنی DB و DBMS را دارد.

برخی از وظایف:

- مدیریت دسترسی کاربران سطوح مختلف به DB
- مانیتور کردن و نظارت لحظه به لحظه کارکرد سیستم
- تهیه نرم‌افزارها و سخت افزارهای لازم
- تأمین امنیت سیستم

۲- Database Designers

مشخص کردن داده‌هایی که در پایگاه داده باید ذخیره شوند. تهیه ساختاری برای نمایش و ذخیره سازی داده‌ها در اصطلاح به این ساختار می‌گویند شِما Schema. ایجاد Viewهای مختلف برای گروه‌های مختلف سیستم.

۳- کاربران نهایی یا End Users:

کاربرانی هستند که کار آنها طوری است که نیاز به دسترسی Database دارند برای پرس و جو و Update کردن و غیره

۴- Software Engineers & System Analysis & Application programmers

(مهندسان نرم‌افزار، تحلیل گران سیستم و برنامه نویسان برنامه‌های کاربردی)

نیازهای کاربران نهایی را تعیین می‌کنند، برنامه‌های لازم جهت رفع این نیازمندی‌ها را طراحی و پیاده‌سازی می‌کنند، وظیفه تست، خطایابی و تعمیرات را نیز بر عهده دارند.

نکته: باید با تمام قابلیت‌های DBMS آشنا باشد.

۵- کاربرانی که به طور غیر مستقیم با Database در ارتباط اند:

۵-۱- System Designers & Implementers

وظیفه طراحی و پیاده سازی ماژول‌های DBMS و رابط‌های نرم‌افزاری لازم را بر عهده دارند.

۵-۲- Tool Developers

وظیفه طراحی و پیاده‌سازی پکیج‌های نرم‌افزاری را که استفاده از Database را تسهیل می‌کنند، دارند.

۵-۳- Operators and Maintenance Personel

پرسنل مربوط به نگهداری و انجام‌کارهای عملیاتی سیستم وظیفه اجرا، تعمیرات و مدیریت سخت‌افزارها و نرم‌افزارهای سیستم را بر عهده دارد.

تاریخچه Database:

۱- Databaseهای اولیه در سال‌های ۱۹۸۰-۱۹۶۰

- وجود داده‌های مشابه بسیار در بانک اطلاعاتی
- عدم انعطاف پذیری مناسب در ایجاد پرس و جوهای لازم
- **بزرگترین مشکل**: عدم حضور DBMS یعنی برای هر بانک اطلاعاتی یک برنامه مخصوص نوشته می‌شد، در نتیجه برای افزودن یک پرس و جوی جدید باید یک برنامه جدید نوشته، تست و خطایابی می‌شد که بسیار وقت‌گیر بود.

۲- ظهور Databaseهای رابطه‌ای در ۱۹۸۰

- طراحی زبان‌های پرس و جوی سطح بالا و قبلیت طراحی Applicationهای منعطف‌تر.

۳- دوره ظهور برنامه‌های شیئ‌گرا و نیاز به طراحی پایگاه داده‌های پیچیده در اواسط و اواخر دهه ۸۰

۴- دوره ظهور پایگاه داده‌های تحت وب در دهه ۹۰

۵- دوره توسعه قابلیت‌های Databaseها برای برنامه‌های کاربردی جدید

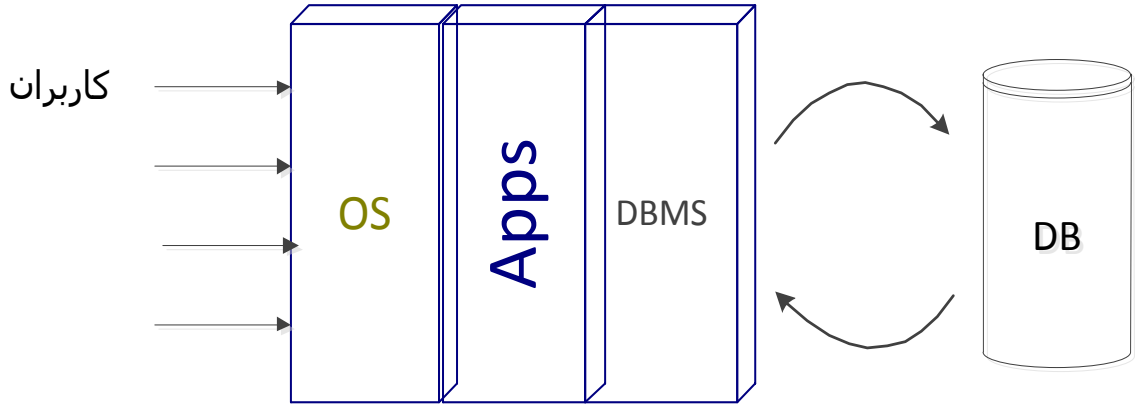
قابلیت‌هایی مثل ذخیره سازی داده‌ها در زمان خاص یا به صورت ماهیانه، گرفتن نسخه پشتیبان حرفه‌ای‌تر و ...

انواع معماری سیستم پایگاه داده‌ها

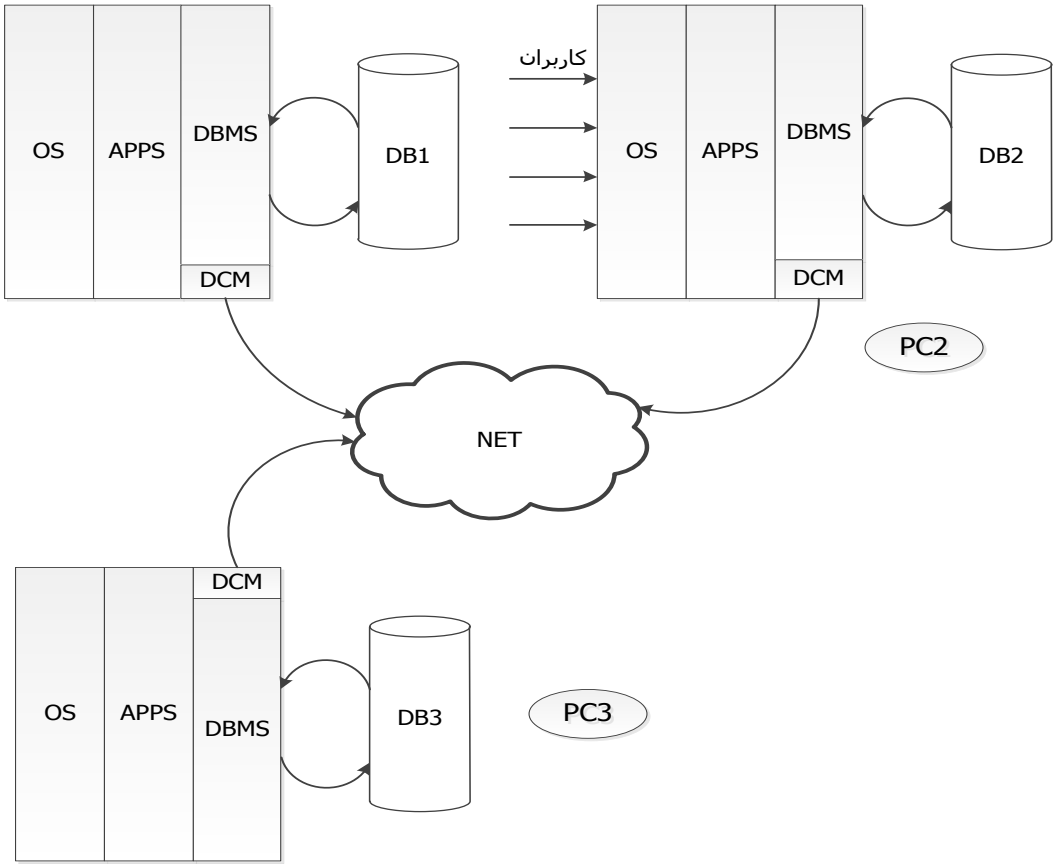
۱- پایگاه داده متمرکز یا Centralized DB یا CDB :

DB و DBMS هر دو بر روی یک کامپیوتر هستند و به شبکه خاصی متصل نیستند.

Centeralized DB (CDB) پایگاه داده های متمرکز



۲- پایگاه داده توزیع شده یا Distributed DB یا DDB :



در این نوع پایگاه داده داریم:

DDB={DB1+DB2+DB3}

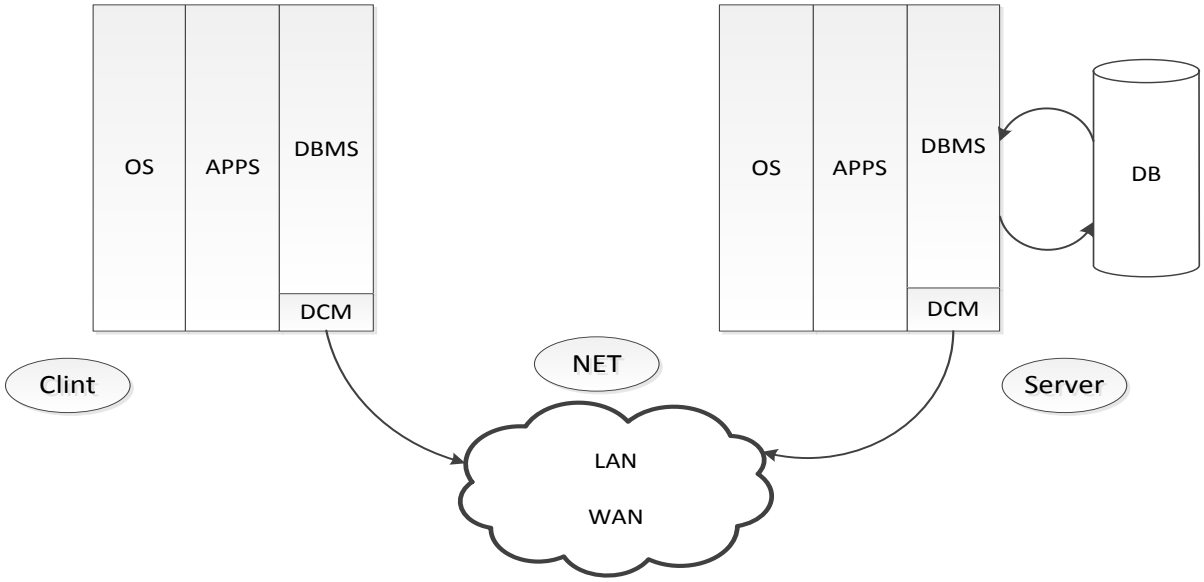
شعب مختلف بانک ملی را به عنوان PC1، PC2 و PC3 در نظر بگیرید. اطلاعات هر شعبه روی سیستم همان شعبه است.

سوال: تکنولوژی DDB با چه تکنولوژی‌هایی درگیر است؟

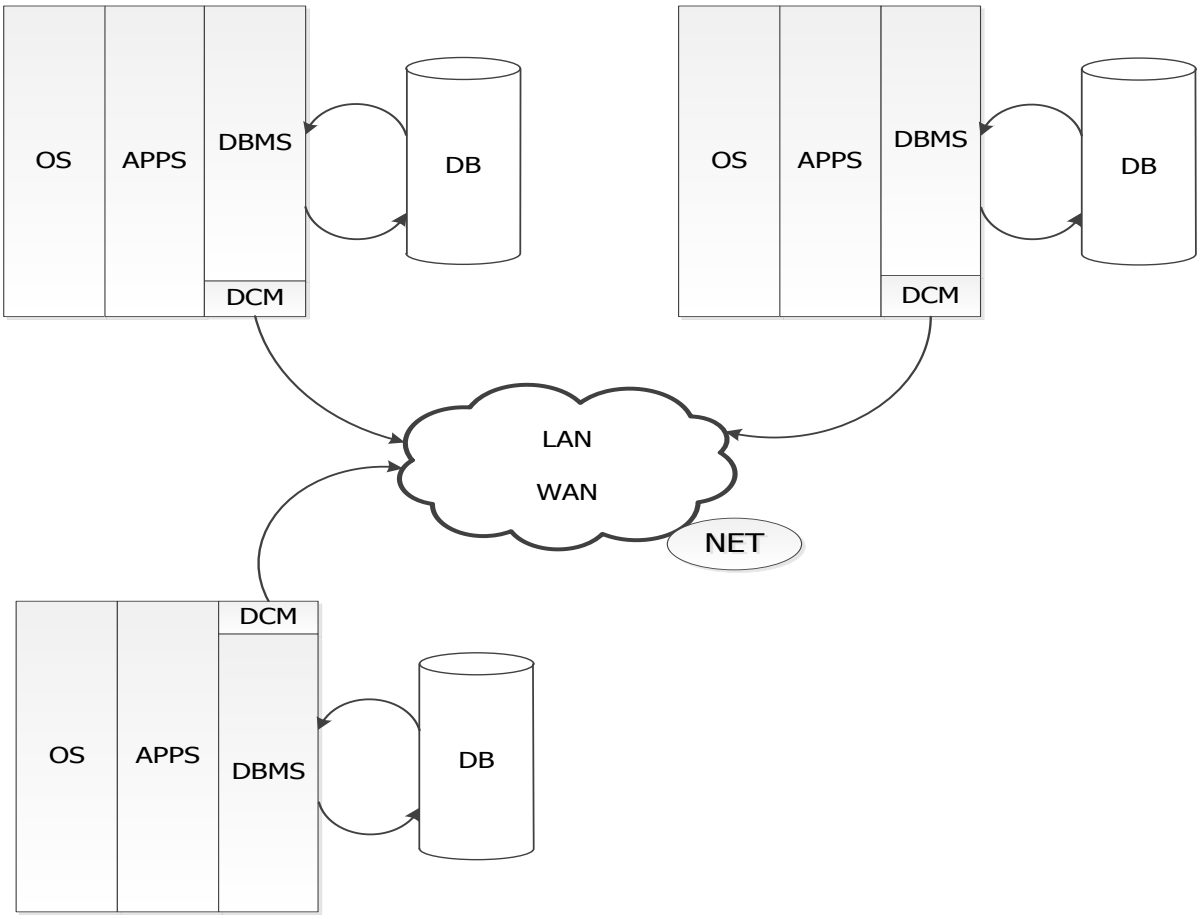
DDB Technology = DB Tech + NET Tech

۳- پایگاه داده کلاینت-سروری یا Client / Server DB یا C/S DB :

مانند سیستم شبکه بانکی شتاب



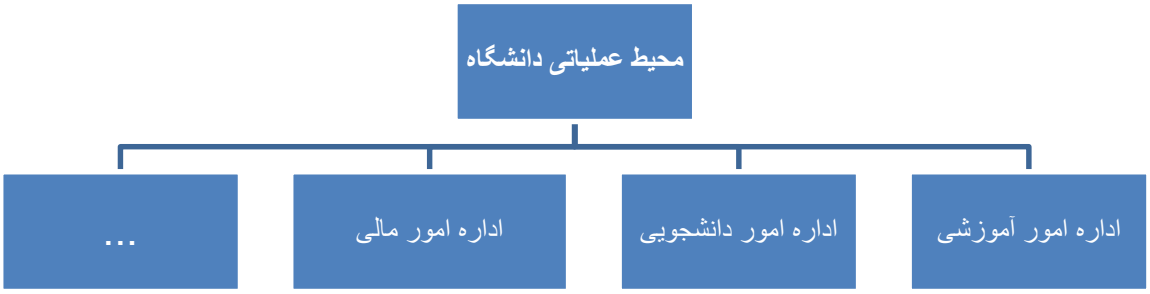
۴- پایگاه داده چند کلاینتی-چند سروری یا Multi Client / Multi Server DB یا MC / MS DB



محیط عملیاتی:

به محیطی که قصد طراحی یک سیستم ذخیره و بازیابی اطلاعات (ذبا) را برای آن داریم (miniworld) خرده جهان واقع یا «محیط عملیاتی» گفته می‌شود.

هر محیط عملیاتی معمولاً از چندین محیط تشکیل شده است:



The Entity-Relationship Model
(مدل E/R)

روند طراحی یک پایگاه داده با تحلیل اطلاعاتی که باید در Database نگه داشته شوند و روابط بین آن‌ها آغاز می‌شود. سپس این داده‌ها باید **مدل سازی معنایی** شوند. برای مدل سازی معنایی داده‌ها روش‌های مختلفی وجود دارد مانند روش Functional یا روش UML اما رایج ترین تکنولوژی روشی است به نام E/R.

E/R: یک مدل گرافیکی است شامل Boxها و فلش‌هایی برای نمایش عناصر داده‌ای و روابط بین آنها.

(در فصل‌های بعد این اشکال گرافیکی با استفاده از مدل رابطه‌ای به جدول تبدیل می‌شوند)

روند پیاده سازی یک Database:

عناصر مدل E/R:

در مدل E/R ساختار Database توسط عناصر زیر به صورت گرافیکی نمایش داده می‌شود.

۱- موجودیت Entity

۲- صفات Attribute

۳- روابط Relationship

۱- موجودیت:

An entity represents a real-world object or concepts such as an employee or project.

تعریف: یک موجودیت، نماد یک شیئ یا مفهوم در دنیای واقعی است. مثل یک کارمند یا یک پروژه.

توجه: در هر محیط عملیاتی مجموعه‌ای از موجودیت‌ها وجود دارد که مدل ساز باید این مجموعه را به درستی تشخیص دهد. برای نمونه در محیط عملیاتی دانشگاه، چند نمونه موجودیت عبارتند از: دانشجو، استاد، کارمند، کلاس، درس، ترم، و ... .

راهنمای عملی تشخیص موجودیت‌های محیط عملیاتی:

- یک موجودیت معمولاً چندین نمونه (instance) در محیط دارد.
- کاربر، مجموعه‌ای از اطلاعات در مورد آن می‌خواهد.
- معمولاً بیش از یک صفت دارد.

نمونه‌های دیگری از موجودیت:

در یک آموزشکده زبان، برخی از موجودیت‌ها عبارتند از:

زبان آموزان، اساتید، کلاس‌ها، ترم‌ها و غیره.

توجه کنید که در این سیستم، خود «آموزشکده» یک موجودیت نیست، چون بند اول در مورد آن برقرار نیست. اما اگر بخواهیم برای کل آموزشکده‌های سراسر کشور برنامه بنویسیم آنگاه آموزشکده خود یک موجودیت خواهد بود.

برای نمونه‌ای دیگر می‌توان از کتابخانه نام برد و موجودیت‌هایی چون، کتاب‌ها، گروه‌های کتاب، قفسه‌ها، اعضا، کتابداران و غیره.

یا در مدرسه: دانش‌آموزان، معلم‌ها، کلاس‌ها، کارمندان، درس‌ها و غیره.

انواع موجودیت:

۱- موجودیت قوی یا مستقل

۲- موجودیت ضعیف یا وابسته

موجودیت قوی: موجودیتی است که وجود آن وابسته به وجود موجودیت دیگری نیست.

برای نمونه در سیستم دانشگاه، یک «دانشجو» یک موجودیت قوی است، حتی اگر هیچ درسی در یک ترم اخذ نکند (مرخصی) باز هم «دانشجو» است. «استاد» نیز موجودیتی قوی است، حتی اگر در یک ترم درسی ارائه نکند باز هم «استاد» است.

موجودیت ضعیف: موجودیتی است که وجودش وابسته به وجود موجودیت دیگری است.

مثل گواهینامه= اگر شخص حذف شود گواهینامه او دیگر مطرح نیست.

مثل والدین= اگر شخص حذف شود دیگر والدین او مطرح نیست.

۲- صفت:

An Attribute represts some property of interest that further describes an Entity

هر موجودیت دارای مشخصاتی است که آن موجودیت را بیشتر توصیف می‌کند. این مشخصات در اصطلاح «صفت» یا Attribute گفته می‌شود.

این موضوع را به صورت زیر نیز نمایش می‌دهند:

$E_i = \{ A_i \}_{i=1,2,...}$ یا $\Rightarrow \sum_{i=1}^{\infty} Ai$

مجموعه‌ای از اطلاعاتی در مورد هر یک از نمونه‌های E_i

مثال: موجودیتی به نام دانشجو دارای صفاتی مانند: نام، نام خانوادگی، نام پدر، آدرس، تلفن و ... است.

بخش‌های تشکیل دهنده یک صفت:

- نام
- معنا
- نوع (عددی، رشته‌ای و...)
- طول مقدار (۲۰ کاراکتری، ۴۰ کاراکتری و...)
- میدان یا دامنه (مقادیر مجاز)

مثل صفت **نمره**:

- Grade
- نمره دانشجو
- Decimal
- ۴
- ۰ تا ۲۰

انواع صفت:

۱- صفت می‌تواند **شناسه** باشد یا نباشد.

صفت شناسه عامل جداسازی یک نمونه موجودیت از یک نمونه دیگر است. (مثل شماره دانشجویی برای موجودیت دانشجو)

صفت شناسه دارای دو ویژگی زیر است:

۱-۱- یکتایی مقدار دارد.

۱-۲- حتی الامکان طول کوتاه‌تر و ترجیحاً از صفات طبیعی است. مثل کد ملی

۲- صفت ممکن است **ساده** باشد یا **مرکب**:

صفت مرکب صفتی است که از تعدادی صفت ساده تشکیل شده است.

مثل صفت **آدرس** شامل: نام شهر، خیابان، کوچه، پلاک

۳- صفت ممکن است **تک مقداری** یا **چند مقداری** (**Multi-Valued**) باشد. مثل صفت شماره تلفن

۴- صفت ممکن است **واقعی** یا **مجازی** باشد.

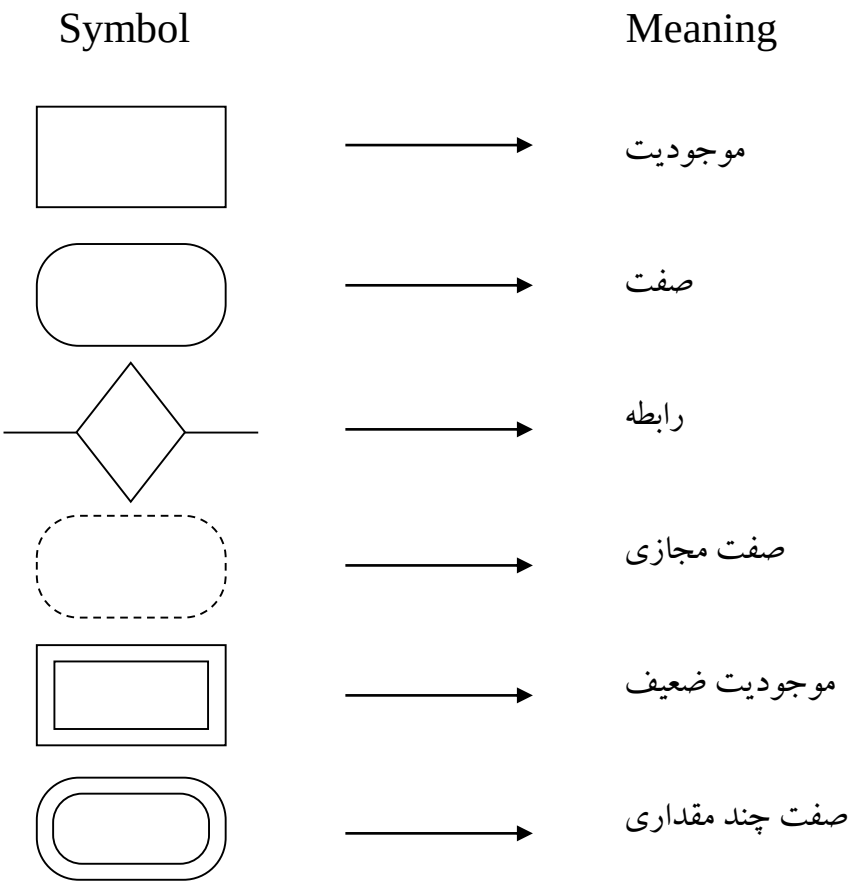
صفت واقعی مقدار ذخیره شده در Database است. مثل نام، نام خانوادگی، نمره و ...

صفت مجازی مقدار ذخیره شده ندارد، بلکه از پردازش از روی مقادیر ذخیره شده به دست می‌آید.

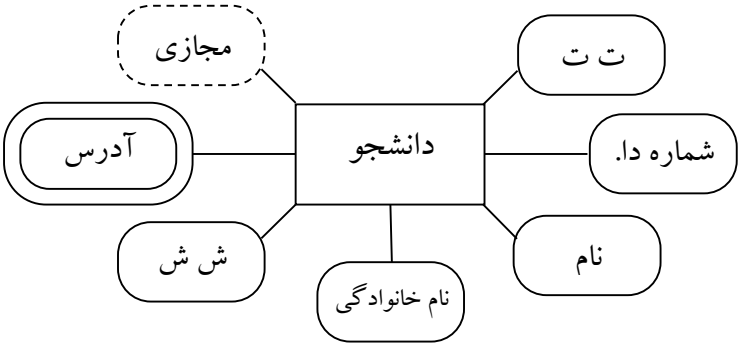
مثل معدل یا سن

توجه: به صفت مجازی، صفت محاسباتی یا صفت مشتق‌شده یا Derived Attribute نیز گفته می‌شود.

E/R Diagram: یک دیاگرام E/R گرافی است که نماد موجودیت‌ها صفات و روابط است.



مثال: نمودار نمایش دانشجو در محیط عملیاتی دانشگاه را رسم کنید.



۳- رابطه‌ها:

Relationships are connections among two or more Entity sets.

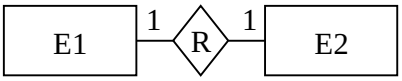
رابطه‌ها ارتباط بین ۲ یا چند موجودیت هستند.

به طور مثال درس و دانشجو ۲ موجودیت هستند. رابطه (اخذ کردن) ۲ موجودیت را به هم متصل می‌کند:

«دانشجو» اخذ می‌کند «درس» را

انواع رابطه بین ۲ موجودیت:

۱- رابطه One-One یا یک به یک:

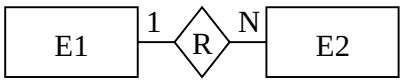


اگر یک نمونه از موجودیت E1 با یک نمونه از موجودیت E2 رابطه داشته باشد و بالعکس آنگاه رابطه آنها یک به یک است.

مثال: رابطه مدیریت بین استاد و دانشکده.

یک نمونه از دانشکده فقط توسط یک استاد مدیریت می‌شود و بالعکس

۲- رابطه Many-One یا یک به چند یا ۱ به N:



اگر یک نمونه از موجودیت E1 با N نمونه از موجودیت E2 در ارتباط باشد در حالیکه یک نمونه از موجودیت E2 فقط با یک نمونه از موجودیت E1 در ارتباط است، در اینصورت، رابطه یک به چند برقرار خواهد بود.

مثال: رابطه بین «دانشگاه» و«دانشجو» در سیستم دانشگاه‌های کل کشور .

هر دانشجو فقط در یک دانشگاه می‌تواند تحصیل کند اما هر دانشگاه می‌تواند با چندین دانشجو رابطه داشته باشد.

۳- رابطه Many-Many یا چند به چند یا N به M :

اگر یک نمونه از موجودیت E1 با M نمونه از موجودیت E2 و یک نمونه از موجودیت E2 با N نمونه از موجودیت E1 ارتباط داشته باشد بین E1 و E2 رابطه N به M برقرار است.

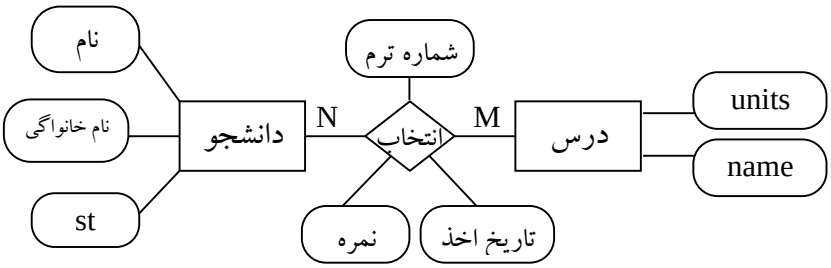
مثال: رابطه بین «دانشجو» و «درس».

یک «دانشجو» چندین «درس» اخذ می‌کند و یک «درس» نیز توسط چندین «دانشجو» اخذ می‌شود.

نکته: در محیط‌های عملیاتی بیش از همه این رابطه به چشم می‌خورد.

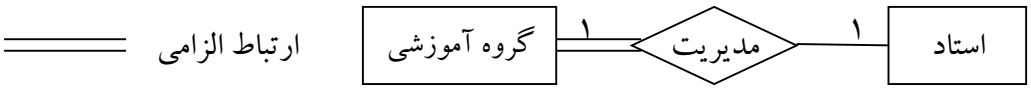
نکته: ارتباط نیز می‌تواند صفت یا صفاتی داشته باشد از این منظر، ارتباط نیز نوعی موجودیت است.

مثال:



نکته: مشارکت یک نوع موجودیت در یک رابطه ممکن است الزامی یا (کامل) باشد یا نباشد. مشارکت الزامی زمانی است که تمام نمونه‌های موجودیت در آن ارتباط شرکت داشته باشند.

مثال:



هر گروه آموزشی حتماً باید توسط یک استاد مدیریت شود اما لزوماً یک استاد مدیریت یک گروه آموزشی را بر عهده ندارد.

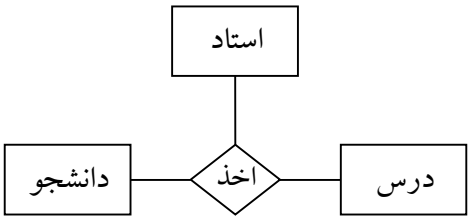
نکاتی در مورد ارتباط:

ارتباط ممکن است بین یک موجودیت و خودش باشد. در این حالت ارتباط Self-Relation نامیده می‌شود.

نکته: ارتباط بزرگتر از سه موجودیتی در محیط عملیاتی توصیه نمی‌شود.

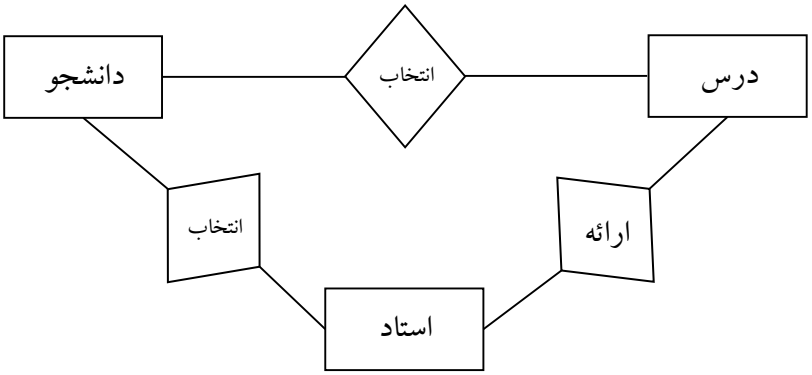
نکته: با داشتن یک ارتباط سه موجودیتی می‌توان سه ارتباط دو موجودیتی به دست آورد اما بر عکس آن همیشه درست نیست. اگر چنین کنیم در اصطلاح دچار دام پیوندی (Connection trap) شده‌ایم.

مثال:



مدل اول: یک ارتباط سه موجودیتی

اگر کاربر به طور قطع از سه ارتباط دو موجودیتی یک ارتباط سه موجودیتی به دست بیاورد دچار دام پیوندی شده است.



مدل دوم: سه ارتباط دو موجودیتی

اگر بخواهیم مدل دوم را به مدل اول تبدیل کنیم (دام پیوندی) به مشکل زیر بر می‌خوریم. فرض کنید داریم:

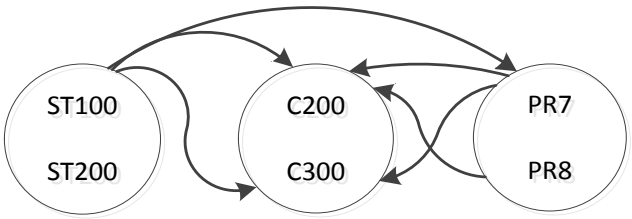
R1= دانشجوی ST100، درس C200 را انتخاب کرده است.

R2= دانشجوی ST100، دانشجوی استاد PR7 است.

R3= استاد PR7 درس C200 را تدریس می کند.

اگر مدل دوم را به مدل اول تبدیل کنیم، نتیجه زیر گرفته می‌شود که ممکن است درست نباشد:

R4= دانشجوی ST100 درس C200 را با استاد PR7 انتخاب کرده است.

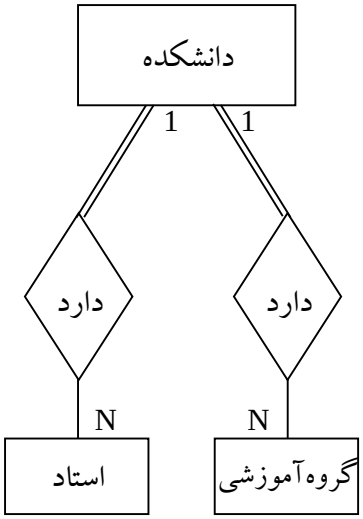


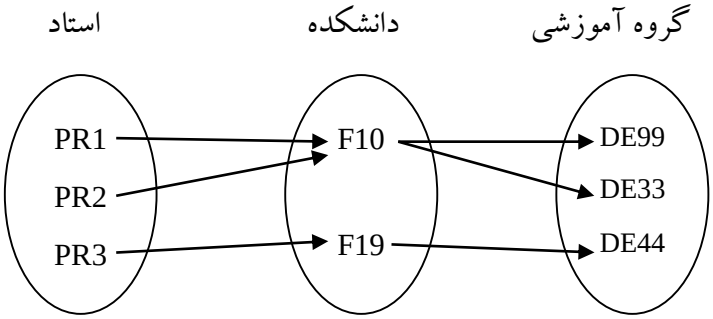
در شکل بالا درست است که با PR7 در ارتباط است و درس C200 را نیز اخذ کرده اما ممکن است این درس را با PR8 اخذ کرده باشد و درس C300 را با PR7 گرفته باشد.

مشکلات روش ER:

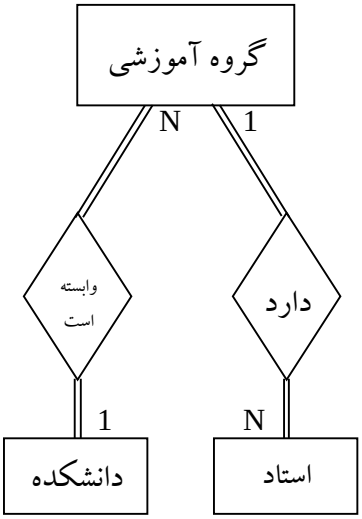
دام چند شاخه یا چتری: این دام وقتی تولید می‌شود که یک نوع موجودیت با دیگر موجودیت ارتباط ۱ به N با مشارکت الزامی داشته باشد.

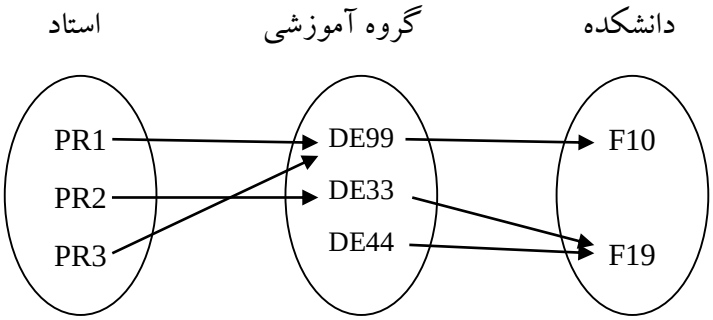
مثال:



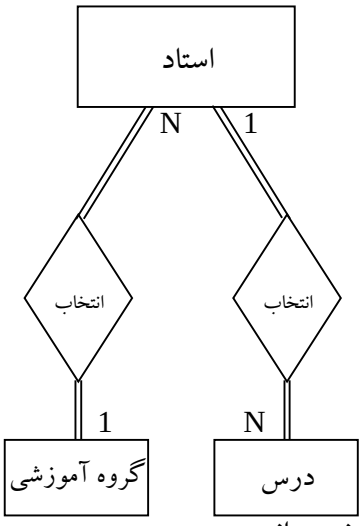


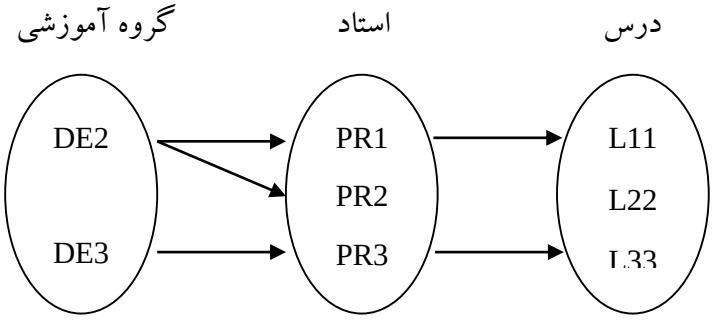
توضیح: در این مثال نمی‌توان گفت استاد PR1 عضو گروه آموزشی DE99 است یا گروه آموزشی DE33.
اگر این مدل سازی به صورت زیر تغییر کند مشکل برطرف می‌شود:





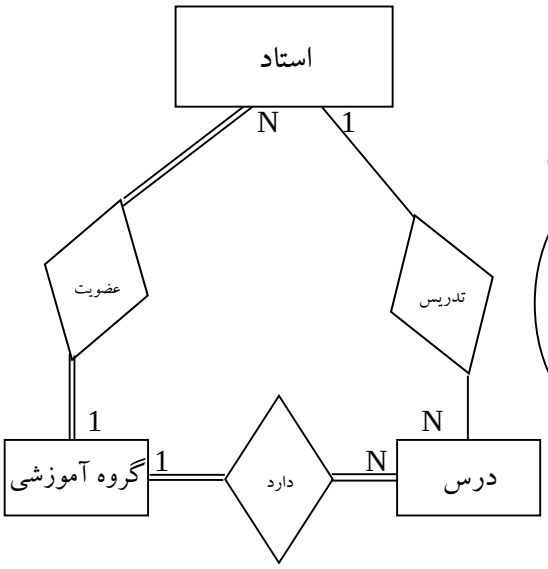
دام شکاف یا گسل: زمانی ایجاد می‌شود که یک موجودیت با موجودیت دیگر، ارتباط N به ۱ از نوع الزامی داشته باشد ولی با یک موجودیت دیگر ارتباط ۱ به N غیر الزامی داشته باشد.

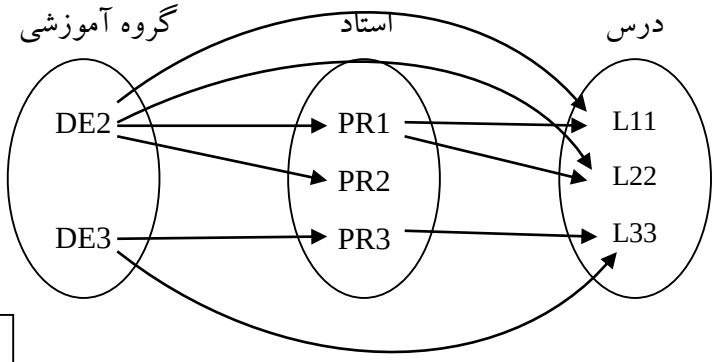




سؤال: در این مثال درس L22 از درس‌های کدام گروه آموزشی است؟ جواب: نمی‌دانیم.

اگر مدل بالا به صورت زیر تغییر کند مشکل حل می‌شود:





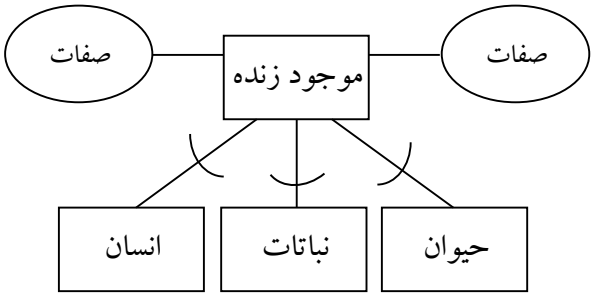
EER (Extended ER or Enhanced ER):

روش ER کمبودهایی دارد که در مباحث شیئ‌گرایی برخی از روابط بین اشیاء در دنیای واقعی را نمی‌توان به این روش مدل‌سازی کرد به همین خاطر گسترش داده شده و روش EER ایجاد شده است. این مدل شبیه کلاس‌ها و وراثت در برنامه نویسی شیئ‌گرا است.

- رابطه IS-A (گونه‌ای است از): برای نمایش ارتباط یک نوع موجودیت با گونه‌هایش از این رابطه استفاده می‌شود.

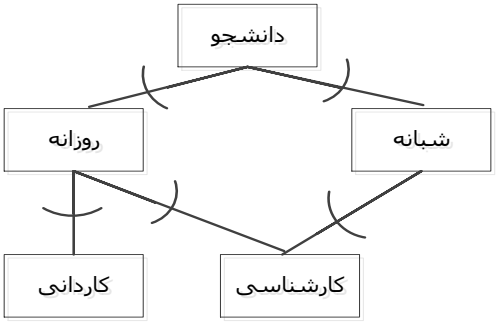
معمولاً به صورت کمان یا مثلث درج می‌شود.

می‌توان به جای کمان از △ استفاده کرد.

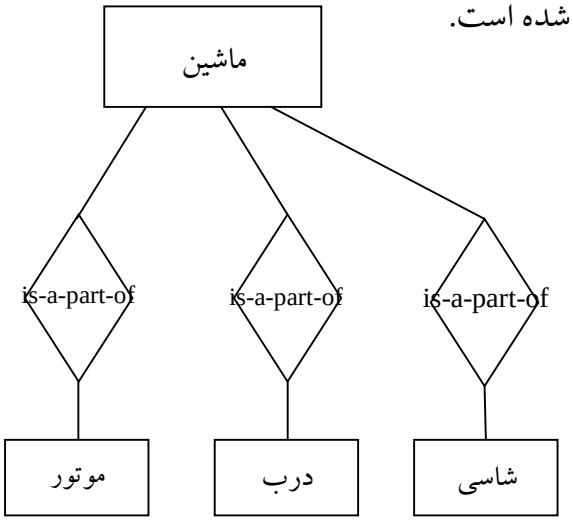


صفات موجودات زنده توسط حیوانات و انسان به ارث می‌برند.

نکته: یک زیر-موجودیت (sub-entity) می‌تواند بیش از یک موجودیت والد (super-entity) داشته باشد.



رابطه IS-A-PART-OF: ارتباط بین یک شیئ کل است و اشیاء تشکیل دهنده آن. یا به عبارت دیگر می‌گوییم یک موجودیت از تعدادی موجودیت جزء تشکیل شده است.



نمودار ER نشان دهنده پایگاه داده‌هایی است که قرار است طراحی و پیاده سازی شوند بنابراین نمایش Database است در بالاترین سطح انتظار.

تخصیص: مشخص کردن گونه‌های خاص یک شیئ را گویند. به طور مثال در مورد موجود زنده سه گونه خاص عبارتند از: انسان حیوان نبات

تعمیم: عکس عمل تخصیص است یعنی با داشتن زیر نوع‌های خاص صفات آن‌ها را در یک مجموعه صفات برای یک ابر موجودیت در نظر می گیریم.

مشهورترین روش‌های مدل‌سازی داده:

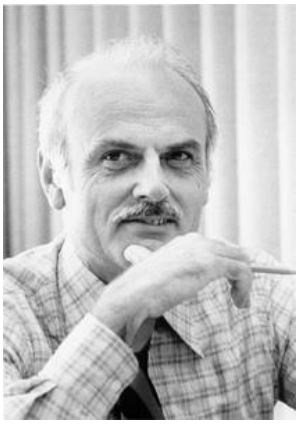
1. RDM (Relational Data Model) مدل داده‌ای رابطه‌ای
2. HDM (Hierarchical Data Model) مدل داده‌ای سلسله مراتبی
3. NDM (Network Data Model) مدل داده‌ای شبکه‌ای
4. ODM (Object Data Model) مدل داده‌ای شیئ‌گرا
5. ORDM (Object Relational Data Model) مدل داده‌ای شیئ‌گرا رابطه‌ای

مدل‌های HDM و NDM از ابتدایی‌ترین مدل‌ها بوده و به دلیل برخی مشکلات از جمله عدم وضوح کاربری دیگر کاربرد خاصی ندارد.

مدل‌های ODM و ORDM مدل‌های تازه وارد به حساب می‌آیند.

آشنایی با مدل RDM

- اولین بار توسط آقای CODD در سال ۱۹۷۰ و سپس توسط آقای C.J.Date مطرح شد.



Edgar 'Ted' Codd

C.J. Date

- اولین DBS که رابطه‌ای بود در اوایل سال ۱۹۸۰ توسط شرکت‌های oracle و IBM مهم‌ترین RDBMS (سیستم‌های مدیریت پایگاه داده رابطه‌ای) عبارتند از:

Oracle و RDB محصول شرکت Oracle و SQL Server و Access محصول شرکت Microsoft

مفاهیم RDBMS

رابطه: رابطه یک مفهوم ریاضی است اما از دیدگاه کاربری، نمایش جدولی دارد. به همین خاطر به RDM ممکن است TDM (Tabular Data Model) نیز گفته شود.

عناصر مدل رابطه‌ای یا جدولی:

جدول (Table): در طراحی Database با TDM هر نوع موجودیت و رابطه با جدول نمایش داده می‌شود.

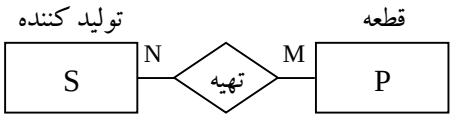
سطر (Row): هر نمونه موجودیت و هر نمونه ارتباط با مفهوم سطر نمایش داده می‌شود.

ستون (Column): هر صفت به صورت ستون‌های یک جدول نمایش داده می‌شود.

در طراحی در روش TDM نمودارهای ER به مجموعه‌ای از جدول‌ها تبدیل می‌شود.

هر نوع موجودیت مستقل یک جدول را تشکیل می‌دهد و روابط M به N نیز یک جدول به خود اختصاص می‌دهند.

مثال: مسأله تولید کننده و قطعه.



تعریف رابطه در مدل RBM: با فرض وجود m مجموعه از مقادیر D_1 و D_2 و ...و D_m هر یک با نوع مشخص (و نه لزوماً متمایز)، رابطه R تعریف شده روی میدان m از دو قسمت تشکیل شده است.

۱- Heading (عنوان)

۲- Body (بدنه)

S

S#	SName	Staus	City
S1	SN1	90	C1
S2	SN2	20	C2
S3	SN3	10	C3

SP

S#	P#	City
S1	P1	100
S2	P2	50
S3	P3	10

مجموعه Heading مجموعه‌ای است نام‌دار از اسامی صفات رابطه.

مثال: برای مثال تولید کننده و قطعه داریم

S (S#, SName, Status, City)
P (P#,PName,Color,Weight)
SP (S#,P#,City)

مجموعه Body بدنه‌ای است از تاپل‌ها:

$\longrightarrow \{<A_1:V_{11}> , <A_2:V_{12}> ,..., <A_n:V_n>\}$

مثال: برای سطر اول جدول S داریم:

{<S#:'S1'> , <SName:'SN1'> , <Status:'10'> , <City:C1>}

نکته: مجموعه بدنه، «متغیر در زمان» است یعنی در هر لحظه ممکن است صفر یا چندین تاپل (tuple) یا سطر در آن موجود باشد. (مثل بدو تعریف رابطه که این مجموعه تهی است و هیچ رابطه‌ای ندارد.) اما مجموعه عنوان، «ثابت در زمان» است. یعنی اگر صفتی از آن کم یا زیاد کنیم از نظر ریاضی با رابطه اول متفاوت است.

درجه رابطه: به تعداد صفات یک رابطه درجه رابطه گفته می‌شود. (در اصطلاح گفته می‌شود: کاردینالیتی مجموعه عنوان)

اصطلاح فارسی	اصطلاح انگلیسی	درجه
یگانی	Unary	۱
دوگانی	Binary	۲
سه‌گانی	Ternary	۳
nگانی	n-ary	n

نکته: به درجه (Degree) در اصطلاح arity نیز گفته می‌شود.

مثال: مدل S در مثال قبل یک مدل چهارگانی است.

کاردینالیتی رابطه: تعداد تاپل‌های مجموعه بدنه را کاردینالیتی رابطه گویند.

تناظر بین مفاهیم رابطه‌ای و جدولی

مفاهیم رابطه	مفاهیم جدول
رابطه	جدول
تاپل	سطر
صفت	ستون
درجه رابطه	تعداد ستون
کاردینالیتی رابطه	تعداد سطرها
میدان	محدوده مقادیر مجاز ستون
کلید	شناسه

نکته ۱: جدول صرفاً یک امکان است برای نمایش رابطه.

نکته ۲: در مدل رابطه‌ای هم موجودیت و هم ارتباط (رابطه) نمایش داده می‌شوند.

خصوصیات رابطه:

۱- رابطه تاپل تکراری ندارد. (مثلاً دو سطر در جدول دانشجویان با مشخصات مشابه نداریم.)

۲- صفات رابطه نظم مکانی ندارند.

$R(A_1, A_2) = R(A_2, A_1)$

S (S# , SName ,...) = S (SName , S# ,...)

۳- تاپل‌ها نظم ندارند.

۴- طبق تعیین آقای کاد صفات رابطه باید اتومیک (Atomic) باشد. (ساده باشد، یعنی مرکب نباشد.)

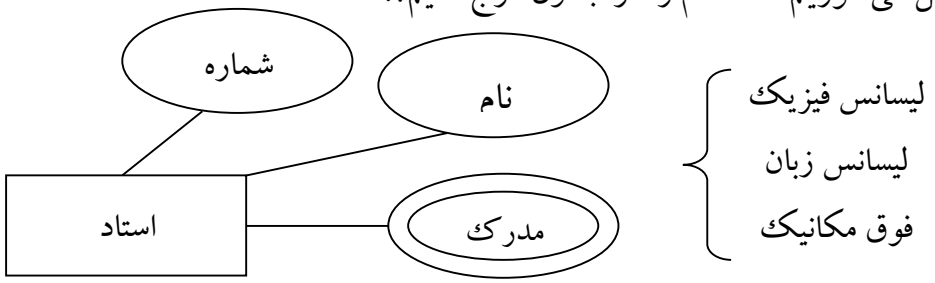
۵- تمام صفات، تک مقداری هستند. یعنی در تقاطع سطر و ستون باید یک مقدار وجود داشته باشند.

تعریف رابطه Normal: رابطه‌ای است که صفات ساده یا مرکب چند مقداری نداشته باشد. به عبارت دیگر مقادیر صفات در تاپل‌های مختلف تکرار شونده نباشند. (مثل شماره تلفن)

Prof (مدرک، نام، شماره)

Prof (P20 , Ali, { لیسانس فیزیک / لیسانس زبان / فوق مکانیک })

در مورد مدرک به مشکل می‌خوریم که کدام را در جدول درج کنیم؟!



دلیل لزوم Normal بودن رابطه: دلیل یک چیز است و آن سادگی!

- سادگی در نمایش ظاهری رابطه
- سادگی دستورات
- سادگی در اجرای عملیات در پایگاه داده.

رابطه غیر Normal: رابطه‌ای که صفات ساده یا مرکب چند مقداری داشته باشد. به این رابطه، رابطه تودرتو نیز گفته می‌شود.

Key یا کلید در مدل رابطه‌ای:

انواع کلید:

۱- کلید سوپر یا ابر کلید — S.K = Super Key

۲- کلید کاندید — C.K = Candidate Key

۳- کلید اصلی — P.K = Primary Key

۴- کلید بدیل — A.K = Alternative Key

۵-کلید خارجی — F.K = Foreign Key

$$R(H) = R(A_1, A_2, \ldots A_i, A_H \ldots A_n)$$

با توجه به این رابطه کلیدهای مختلف را تعریف می‌کنیم.

سوپر کلید: هر زیر مجموعه از H که یکتایی مقدار داشته باشد.

کلید کاندید: هر زیر مجموعه از H که دو خاصیت زیر را داشته باشد:

۱- یکتایی مقدار داشته باشد.

۲- کاهش ناپذیر باشد. یعنی اگر یکی از عناصر این مجموعه را حذف کنیم دیگر شاهد یکتایی مقدار نباشیم.

مثال: در سیستم کتابخانه چنین جدولی خواهیم داشت.

Book

B#	BName	Author

Member

M#	MName	MAddress

B-M

B#	M#	Date

در جدول بوک B# کلید کاندید است.

نکته: برای یافتن یک سطر خاص از طریق کلید کاندید می‌توان آن را یافت.

در جدول B-M ترکیب B# , M# , Date کلید کاندید است.

سؤال: آیا می‌توان فقط ترکیب B# و M# را کلید کاندید گرفت؟ خیر. چون ممکن است یک عضو خاص یک کتاب خاص را چندین بار اخذ کرده باشد.

سؤال: آیا هر رابطه حداقل یک کلید کاندید دارد؟ بله. زیرا در بدترین حالت با ترکیب تمام صفات به H می‌رسیم و طبق تعریف رابطه (که هیچ تاپل تکراری نداریم) H یکتایی مقدار دارد.

- رابطه‌ای که در آن خود H کلید کاندید باشد رابطه All-Key یا «تمام کلید» گفته می‌شود.
- رابطه ممکن است بیش از یک کلید کاندید داشته باشد.

کلید اصلی: طراح پایگاه داده باید یکی از کلیدهای کاندید را به عنوان کلید اصلی رابطه انتخاب و معرفی کند. از کلید اصلی در تمام بخش‌های سیستم برای یافتن سطر مورد نظر استفاده می‌شود.

سؤال: آیا هر رابطه حتماً یک کلید اصلی دارد؟ بله، چون حداقل یک کلید کاندید داریم پس حتماً یک کلید اصلی داریم.

کلید بدیل: هر کلید کاندید، غیر از کلید اصلی، کلید بدیل نامیده می‌شود.

مثل ترکیب Date , #M در جدول B-K، اگر هر عضو در هر روز قادر به بردن فقط یک کتاب باشد.

کلید خارجی: صفت A_i (ساده یا مرکب) از R_2 در R_2 کلید خارجی نامیده می‌شود اگر A_i در R_1 کلید کاندید باشد.

به طور مثال: #B در B-M کلید خارجی است. همچنین #M در B-M کلید خارجی است.

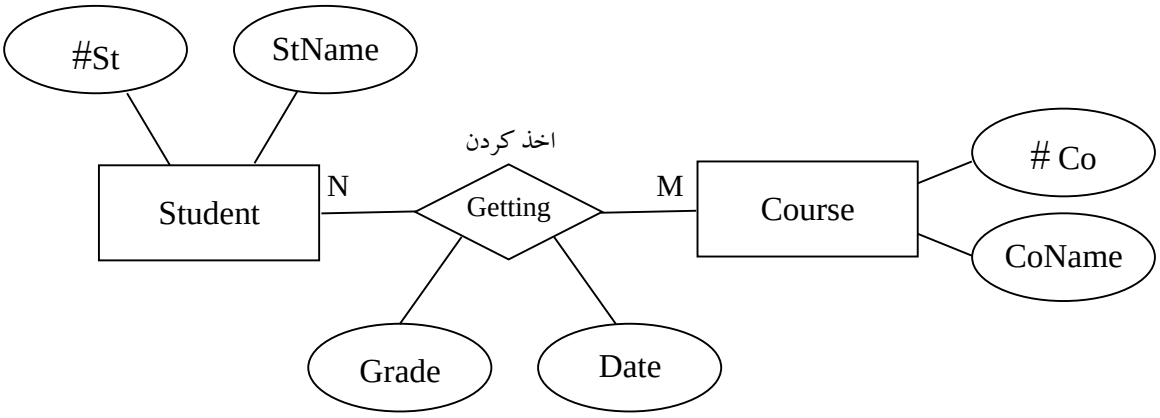
نکته: کلید خارجی امکان اصلی برای نمایش ارتباط بین موجودیت‌ها است.

مثال: صفت #B و #M در رابطه B-M نشان دهنده ارتباط چند به چند «اخذ» بین موجودیت کتاب و عضو است.

تبدیل مدل ER به مدل رابطه‌ای

بررسی حالات مختلف روابط

حالت اول: دو موجودیت مستقل و ارتباط چند به چند



مثال:

Student (St# , StName ,...)
C.K

Course (Co# , CoName ,...)
C.K

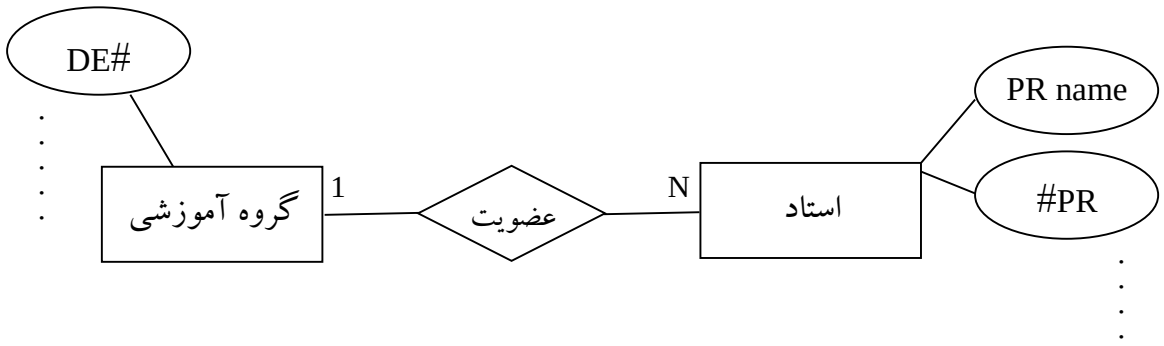
S-C (St# , Co# , Date , Grade)
F.K F.K
C.K

* در این حالت تعداد رابطه برابر است با n+1 که n تعداد موجودیت‌هاست.

* در نمایش ارتباط بین دو موجودیت، کلید کاندید رابطه، حداقل از ترکیب کلیدهای خارجی به دست می‌آید. یعنی ممکن است صفت یا صفاتی از ارتباط هم در کلید کاندید به کار رود.

حالت دوم: دو موجودیت مستقل و یک ارتباط یک به N:

در این حالت موجودیت طرف ۱ را با یک رابطه و موجودیت طرف N و ارتباط را با هم در یک رابطه دیگر نمایش می‌دهیم.



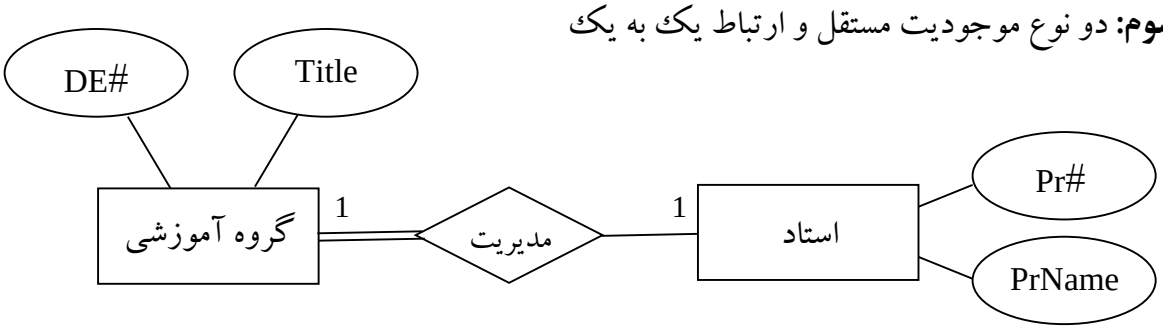
در رابطه دوم کلید کاندید رابطه اول به عنوان کلید خارجی حضور دارد و ارتباط مورد نظر را برقرار می‌کند.

نکته: کلید کاندید طرف یک را در طرف N نگه می‌داریم.

DEPT (DE# , Title , ...)
C.K

PROF (Pr# , PrName , ... , DE#)
C.K F.K

حالت سوم: دو نوع موجودیت مستقل و ارتباط یک به یک



برای موجودیتی که سمت یک ارتباط است یک رابطه تعیین می‌کنیم و برای موجودیتی که در سمت مشارکت الزامی رابطه است یک رابطه دیگر تعریف خواهیم کرد که کلید کاندید رابطه اول در این رابطه کلید خارجی است.

PROF (Pr# , PrName , ...)
C.K

Deft (DE# , Title , ... , Pr#)
C.K F.K

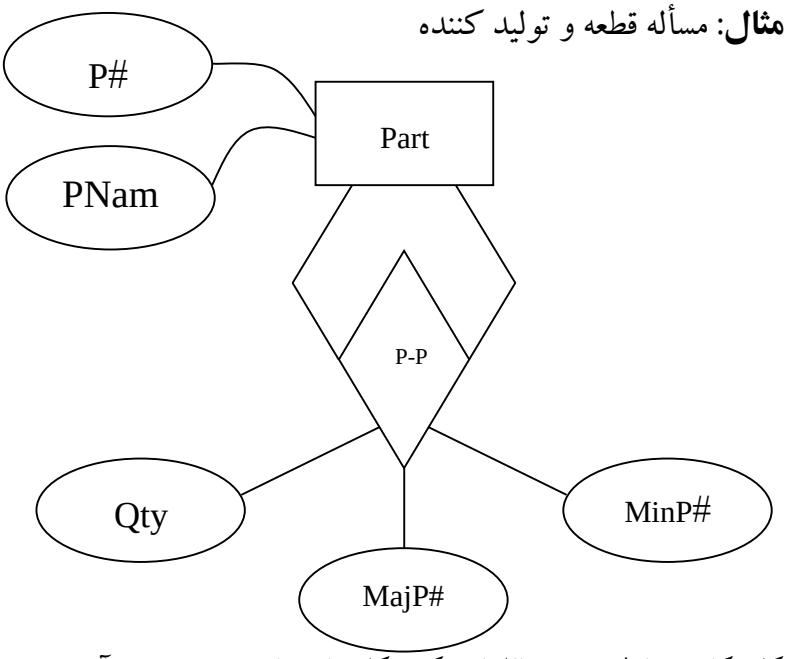
نکته: اگر رابطه ۱ به N خود صفاتی داشت صفات را به رابطه‌ای که شامل کلید خارجی است اضافه می‌کنیم.

حالت چهارم: یک نوع موجودیت داریم که با خودش رابطه N به M دارد.

مثال: مسأله قطعه و تولید کننده

MinP# : شماره قطعه فرعی

MajP#: شماره قطعه اصلی



Part (P# , P name , ...)
C.K

P-P = (MajP# , MinP# , Qty)
F.K F.K
C.K

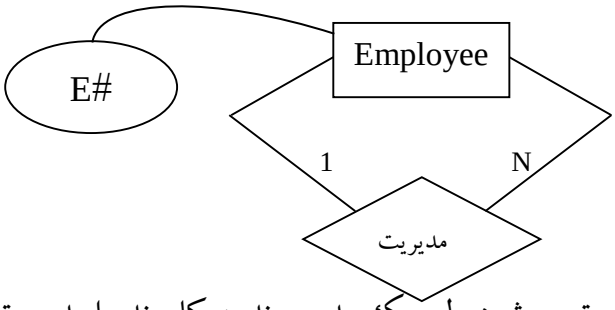
کلید کاندید رابطه دوم، حداقل از ترکیب کلیدهای خارجی به دست می‌آید.

حالت پنجم: یک نوع موجودیت که با خودش رابطه یک به N دارد.

این حالت با یک رابطه طراحی می‌شود.

EMPL (E#, EName, E1name …… , EManager#)

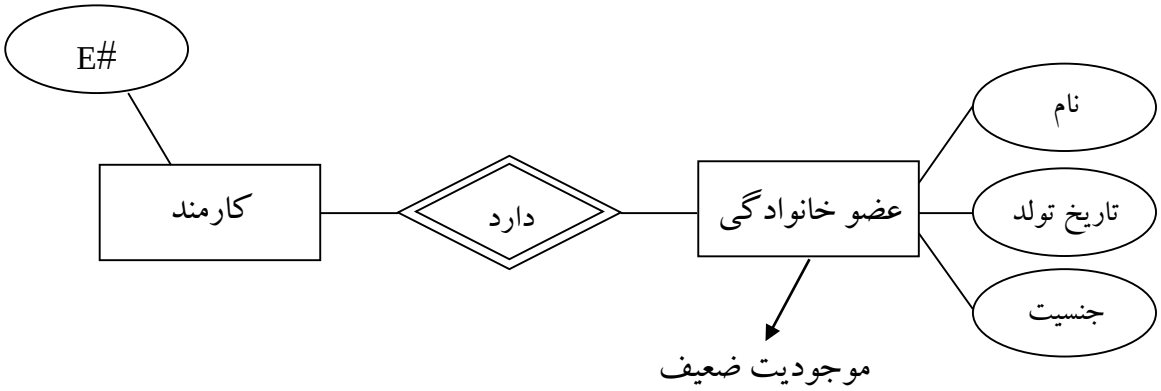
C.K — F.K (شماره کارمندی مدیر)



یک کارمند توسط یک مدیر مدیریت می‌شود ولی یک مدیر چندین کارمند را مدیریت می کند.

حالت ششم: ارتباط موجودیت قوی با موجودیت ضعیف.

معمولاً ارتباط موجودیت قوی با موجودیت ضعیف ۱ به N است.



در این حالت برای موجودیت قوی یک رابطه:

EMPL (E#, EName, ELName, Address, ……)

C.K

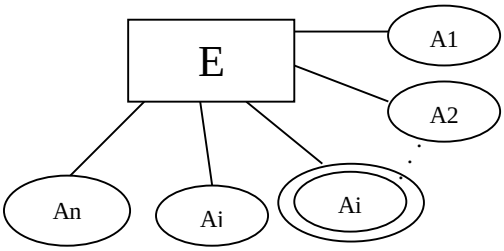
و برای موجودیت ضعیف و ارتباط آن که می‌تواند صفت هم داشته باشد یک رابطه در نظر می گیریم:

MEM (MName, Birthday, Gender, …… E#)

C.K — F.K C.K

حالت هفتم: وجود صفت چند مقداری [اعم از ساده یا مرکب]

در این حالت ۲ رابطه یکی برای موجودیت یکی برای صرفت چند مقداری در نظر می گیریم.

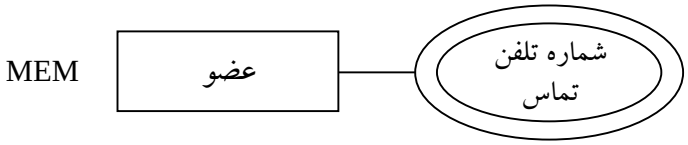


مثال ۱: **صفت چند مقداری ساده:**

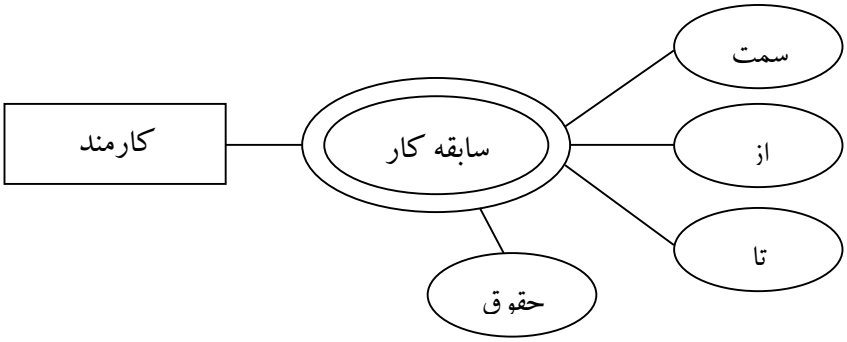
یک جدول برای عضو و یک جدول برای Tel در نظر می گیریم.

MEM (M#, Mname, ...)
C.K

Tel (TelNumber, Code, ..., M#)
F.K
C.K



مثال ۲: **صفت مرکب:**



EMP (E#, Ename, ...)
C.K

EMP jobtistory (E#, JobTittle, BDate, EDate, ...)
F.K
C.K

نکته: اگر تعداد صفات چند مقداری یک موجودیت n باشد n+1 رابطه باید طراحی کرد.

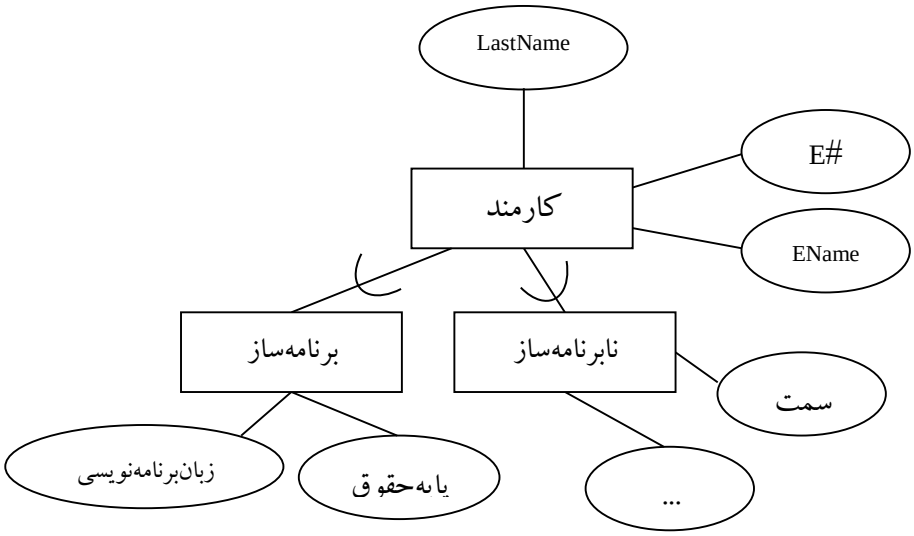
حالت هشتم: رابطه IS-A

اگر یک نوع موجودیت، n زیر نوع داشته باشد n+1 رابطه طراحی می‌شود.

EMP (E#, name, Last name, ...)
C.K

NOPRM (title, ... , E#)
F.K

PRM (PrLanquequ, Fee, ... , E#)
F.K



حالت نهم: IS-A-PART-OF

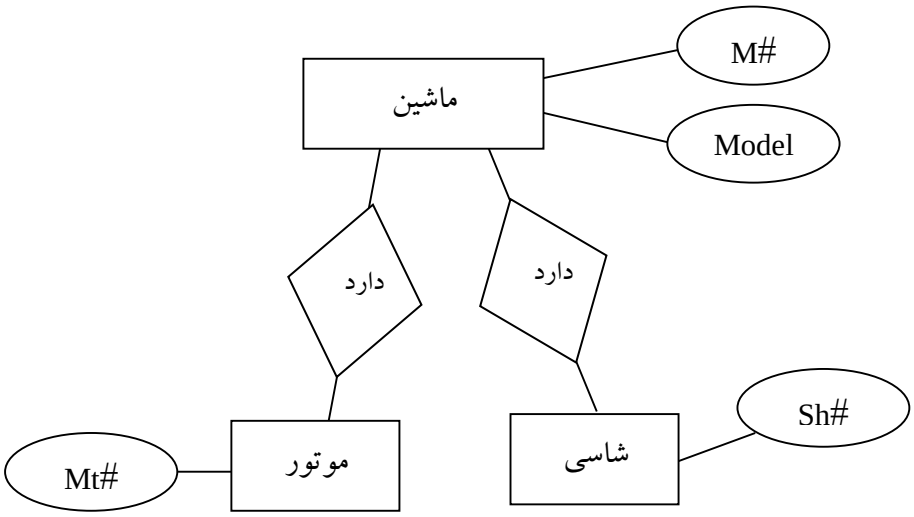
در این حالت، شیئِ جزء، صفات شیئِ کل را به ارث نمی برد.

در این حالت برای موجودیت کل یک رابطه و برای موجودیت جزء نیز یک رابطه تعریف می کنیم.

Car (M#, Model, ...)
C.K

Motor (M#, Mt#, ...)
C.K

Shasi (M#, Sh#, ...)
C.K



حالت دهم: ارتباط سلسله مراتبی بین انواع موجودیت‌ها:

در این حالت برای هر موجودیت یک رابطه تعریف می کنیم. کلید رابطه، ترکیب شناسه موجودیت با شناسه موجودیت پدر آن موجودیت در نظر گرفته می‌شود.

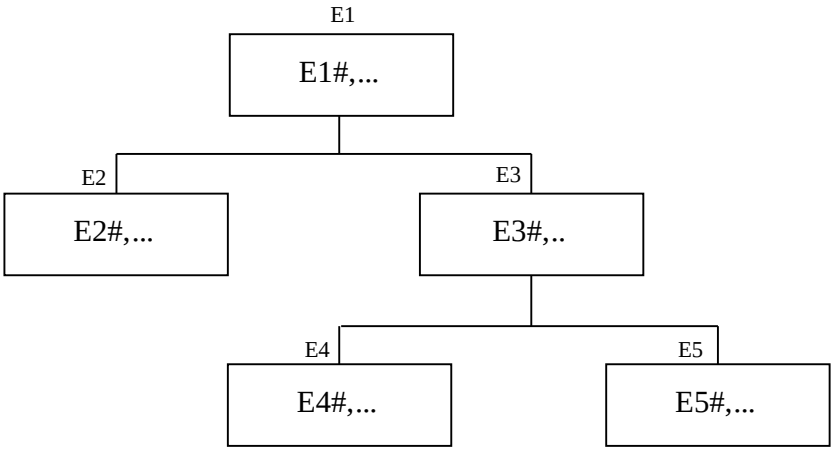
E1REL (E1#, ...)
C.K

E2REL (E2#, E1#, ...)
F.K
C.K

E3REL (E3#, E1#, ...)
F.K
C.K

E4REL (E4#, E3#, E1#, ...)
C.K

E5 (E5#, E3#, E1#, ...)
F.K F.K
C.K

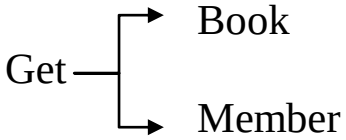


ادامه مفاهیم مدل رابطه‌ای

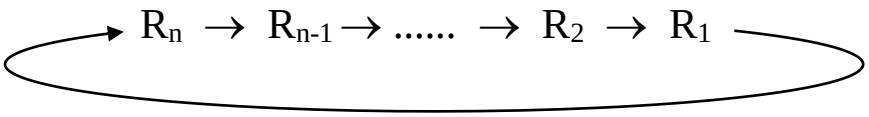
نمودار ارجاع (Reference Diagram):

همان طور که گفته شده کلید خارجی عامل ارتباط یک رابطه با رابطه دیگر است یعنی با داشتن مقدار کلید خارجی در هر سطر از رابطه R_2 می‌توانیم به یک سطر مشخص در رابطه R_1 رجوع کنیم این معنا را چنین نمایش می‌دهیم: $R_2 \rightarrow R_1$ و آن را نمودار ارجاع و یا گراف ارجاع می‌نامیم.

به طور مثال در رابطه اخذ در سیستم کتابخانه داریم:

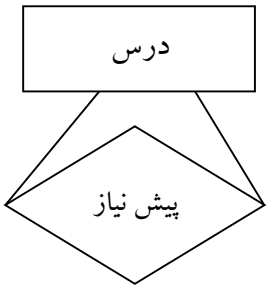


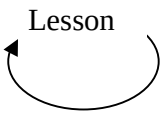
مسیر ارجاع می‌تواند حلقوی (بازگشتی یا چرخشی) باشد.



سؤال: در روابط Self-Relation نمودار ارجاع چگونه خواهد بود؟

جواب:





قوانین جامعیت در مدل رابطه‌ای:

در هر محیط عملیاتی همیشه مجموعه‌ای از قواعد و محدودیت‌ها (مثل بیشتر نبودن نمره از عدد ۲۰) وجود دارند. اگر داده‌های ذخیره شده در پایگاه داده در تمام لحظات این محدودیت‌ها را رعایت کنند در این صورت جامعیت پایگاه داده‌ها تأمین است. هر DBMS باید جامعیت پایگاه داده را کنترل و تأمین کند.

- دلیل اصلی کنترل جامعیت پایگاه داده چیست؟

چون داده‌ها مبنای کلی تصمیم‌گیری در سازمان هستند بنابراین، همیشه باید صحیح، دقیق و معتبر باشند.

انواع قواعد جامعیت:

در RM دو رده کلی محدودیت و قواعد وجود دارد:

۱- قواعد خاص

۲- قواعد عام

۱- قواعد خاص: قواعدی هستند که کاربر مجاز تعریف می‌کنند. (معمولاً این کاربر DBA است)

این قواعد وابسته به یک محیط عملیاتی مشخص هستند و خاص آن محیط هستند.

مثال: در محیط عملیاتی دانشگاه:

- قاعده ۱: نمره بین ۰ تا ۲۰ باشد.
- قاعده ۲: یک دانشجو بیشتر از ۲۰ واحد در یک ترم نمی‌تواند اخذ کند.
- قاعده ۳: اخذ یک درس بدون گذراندن پیش نیاز ممکن نیست.

۲- قواعد عام: وابسته به محیط خاصی نیستند و هر سیستم رابطه‌ای باید بتواند این قواعد را روی DB اعمال کند. (به این قاعده Meta Rules هم گفته می‌شود)

انواع قواعد عام:

۱- قاعده C_1 یا «قاعده جامعیت موجودیتی» [Entity Integrity Rule]

۲- قاعده C_2 یا «قاعده جامعیت کلید خارجی یا ارجاعی» [Referential Integrity Rule]

قاعده C_1: در مورد کلید اصلی است. طبق این قاعده هر جز تشکیل دهنده کلید اصلی نمی‌تواند «هیچ مقدار» (NULL) باشد. «هیچ مقدار» یعنی ناشناخته، تعریف نشده، ناموجود و غیرقابل اعمال.

دلیل توجیح کننده قاعده C_1: کلید اصلی، شناسه هر تاپل در رابطه است یعنی عامل تمایز یک نمونه موجودیت از نمونه دیگر، و عامل تمایز خود نمی‌تواند ناشناخته باشد.

برای اعمال قاعده C_1 باید کلید اصلی را مطابق کد زیر به سیستم معرفی کنیم:

```
CREATE TABLE R
(
     A1   DOMAIN    dom1 NOT NULL Primary Key,
     A2 ........................,
     A3 ........................,
);
```

قاعده C2: در مورد کلید خارجی است. طبق این قاعده اگر Ai در R2 کلید خارجی باشد Ai در رابطه R2 می‌تواند نامعلوم یا NULL باشد، به شرطی که جز تشکیل دهنده کلید R2 نباشد. در غیر این صورت باید حتماً مقداری داشته باشد که در R2 وجود دارد.

یعنی: مقدار کلید خارجی یک رابطه نمی‌تواند در یک رابطه مرجع وجود نداشته باشد.

نکته کنکوری: کلید خارجی می‌تواند NULL باشد.

مثال ۱: در محیط کتابخانه در جدول اخذ نمی‌توان کد کتابی را اخذ کرد که هنوز در جدول Book تعریف نشده است.

مثال ۲:

R1: S (S#, SName, Status, City)	R2: SP (S#, P#, Qty)
S1 SN1 5 Saveh S2 SN2 10 Tehran S3 SN3 2 Qom	S1 P1 10 ✓ S4 P2 100 × زیرا S4 تعریف نشده است

دلیل توجیح کننده قاعده C2: کلید خارجی عامل ارجاع است از یک نمونه موجودیت به نمونه موجودیت دیگر. بنابراین نمی‌توان به نمونه موجودیت ناموجود ارجاع کرد.

نکته: در تعریف کلید خارجی به رابطه R2 در مثال قبل، «رابطه رجوع کننده» و به رابطه R1، «رابطه مرجع» گفته می‌شود.

برای اعمال قاعده C_2 اولاً کلید یا کلیدهای خارجی باید به سیستم معرفی شوند.

ثانیاً باید مشخص شود که در عمل افزودن و حذف یا ویرایش یک تاپل سیستم چگونه عمل کند.

مثلاً در عمل درج چک شود که آیا در رابطه مرجع مقدار کلید خارجی وجود دارد یا خیر و یا در عمل حذف کدام یک از حالات زیر اتفاق بیفتد:

۱- **روش آبشاری (Cascading) :** دراین روش با حذف یک تاپل از رابطه مرجع (مثلاً رابطه S) تمام تاپل‌های رجوع کننده به آن در رابطه‌های رجوع کننده نیز حذف می‌شوند.

۲- **روش محدودیا مشروط(Restricted):** اگر درخواست شود که تاپلی از رابطه مرجع حذف شود این حذف تا زمانی که تاپل های رجوع کننده در رابطه‌های رجوع کننده وجود داشته باشد به تعویق می‌افتد.

۳- **روش پر کردن با NULL (Nullifing):** با حذف تاپل مرجع مقدار کلید خارجی در تاپل‌های رجوع کننده NULL می‌شود، به شرط آنکه قاعده C1 رعایت شود.

۴- **روش مقدار گذاری با مقدار پیش‌فرض (Set-To-Default Value):** با حذف تاپل مرجع مقدار کلید خارجی در تاپل‌های رجوع کننده به جای NULL به مقدار اولیه تعریف شده در شمای Database تغییر می کند.

۵- **روش عدم اقدام (No Action):** در صورت حذف تاپلی از رابطه مرجع اقدام دیگری درخواست نمی‌شود.

توجه: این روش‌ها در عمل Update(ویرایش) نیز قابل اعمال‌اند.

* بهتر است یک RBMS همه این روش‌ها را دارا باشد تا طراح، متناسب با کارش گزینه‌ی مورد نظرش را انتخاب کند.

مثال از اعمال قاعده C_1 و C_2:

```
CREATE TABLE SP (
     S# CHAR (6) NOT NULL,
     P# CHAR(6) NOT NULL,                    قاعده C2
     QTY NUMBER(2)
     )
     CHECK 0≤ QTY ≤ 100,                     قاعده خاص
     PRIMARY KEY (S#,P#),                    قاعده C1
     FOREIGN KEY S# REFERENCES S(S#),
     ON DELETE CASCADE,
     ON UPDATE CASCADE,
     FOREIGN KEY P# REFERENCES P(P#),
     ON DELETE CASCADE,
     ON UPDATE CASCADE,
```

عملیات بر روی پایگاه جدولی:

برای انجام عملیاتی مثل ایجاد و حذف جدول، افزودن، حذف و ویرایش رکورد و غیره بر روی پایگاه جدولی از زبان استاندارد سیستم‌های رابطه‌ای SQL (Structured Query Language) (زبان پرس‌وجوی ساخت‌یافته) استفاده می‌شود. که با این زبان در آزمایشگاه پایگاه‌داده‌ها آشنا خواهید شد.

عملیات بر روی پایگاه رابطه‌ای:

جبر رابطه‌ای

Relational Algebra

به مجموعه‌ای از قوانین و عملگرها که امکان پردازش رابطه‌ها را فراهم می‌کنند «جبر رابطه‌ای» می‌گویند.

در جبر رابطه‌ای حاصل اجرای هر یک از عملگرها، خود یک رابطه است. یعنی ورودی و خروجی تمام عملگرها، یک رابطه است.

خاصیت بسته بودن: این واقعیت که خروجی حاصل از هر عمل رابطه‌ای، رابطه است خاصیت بسته بودن یا بستار یا Closure نامیده می‌شود برای نمونه اجتماع دو رابطه (جدول) یک رابطه (جدول) است..

عملگرها در جبر رابطه‌ای به ۴ دسته زیر تقسیم می‌شوند.

۱- عملگرهای ساده؛ شامل: عملگر گزینش (σ , RESTRICT , SELECT) و پرتو (تصویر ، Project , π)

۲- عملگرهای مجموعه‌ای؛ شامل: اشتراک ($\cap$)، تفاضل (-)، اجتماع ($\cup$)

۳- عملگرهای پیوند؛ شامل: ضرب دکارتی ($\times$)، پیوند طبیعی (∞)، نیم پیوند($\propto$)، پیوند شرطی(θ $\times$) و فراپیوند

۴- عملگرهای دیگر مثل نامگذاری (ρ)، تقسیم($\div$) و جایگزینی ($\leftarrow$)

توجه: مثال‌های این بخش بر اساس جداول S و P و SP زیر طراحی شده است.

S

S#	SName	City
S1	کاوه	ساوه
S2	صفا	ساوه
S3	فن آور	تبریز

P

P#	Color	Type	City
P1	قرمز	آهن	تهران
P2	سبز	مس	تبریز
P3	آبی	برنج	ساوه
P4	قرمز	آهن	شیراز

SP

S#	P#	Qty
S1	P1	۳۰۰
S1	P2	۲۰۰
S1	P3	۴۰۰
S2	P1	۳۰۰
S2	P2	۴۰۰
S3	P3	۲۰۰

عملگر گزینش: این عملگر سطرهایی از جدول را بر می‌گرداند که با یک شرط مطابقت دارند.

در کتب مختلف از نمادهای σ و SELECT و RESTRICT استفاده می‌شود.

مثال: تاپل‌هایی از رابطه S را مشخص کنید که شهر آنها ساوه باشد.

الف: σ city = "ساوه" (S)

ب: SELECT S WHERE city = "ساوه"

ج: RESTRICT S WHERE city = "ساوه"

خروجی		
S1	کاوه	ساوه
S2	صفا	ساوه

ساختار کلی:

الف: $\sigma_{\underset{\text{Conditions}}{\underline{conds}}}(R)$ *ممکن است ۱ یا چند شرط باشد*

ب: SELECT R WHERE conds

ج: RESTRICT R WHERE conds

مثال: تاپل‌هایی را از رابطه SP مشخص کنید که S#=S1 و P#=P1 باشد.

جواب:

$\sigma_{S\#=S1 \wedge P\#=P1}(SP)$

SELECT SP WHERE S#=S1 AND P#=P1

خروجی		
S1	P1	300

* به جای عملگر = می‌توان از <، =<، >، =>، <>، ≠ در صورت نیاز استفاده کرد.

مثال: تاپل هایی از جدول SP را برگردانید که قطعه P2 را تولید کرده باشند و به جز شرکت S1:

$\sigma_{S\#\neq s1 \wedge P\#=P2}(SP)$

SELECT SP WHERE S#≠S1 AND P#=P2

خروجی		
S2	P2	400
S3	P2	200

مثال: تاپل‌هایی از جدول SP را برگردانید که قطعه P3 را بیش از ۵۰۰ تا تولید کرده باشند!

$\sigma_{P\#=P3 \wedge Qty>500}(SP)$

خروجی نداریم، صفر است.

عملگر پرتو: در خروجی عملگر پرتو، سطرهای تکراری حذف می‌شوند. عملگر پرتو هیچ شرطی را نمی‌پذیرد.

برای این عملگر از نمادهای π یا PROJECT استفاده می‌شود.

ستون‌های مورد نظر، زیر علامت π نوشته شده و نام جدول جلوی علامت π در پرانتز.

مثال: ستون شهر را از جدول S نمایش دهید.

نکته: شهرهای تکراری حذف می‌شود.

الف: $\pi_{city}(S)$

ب: PROJECT S [City]

ج: PROJECT S OVER City

خروجی

City
ساوه
تبریز

مثال: ستون‌های Qty , P# را از جدول SP نمایش دهید.

$\pi_{Qty\,,\,P\#}(SP)$

خروجی

P#	Qty
P1	300
P2	200
P3	400

مثال: شماره تولید کنندگان و شماره قطعات تولیدی آنها را چاپ کنید به شرط آن که قطعه تولیدی آنها بیشتر از ۲۰۰ عدد باشد.

$\pi_{S\#\,,\,P\#}(\sigma_{Qty\,\rangle\,200}(SP))$

خروجی

S#	P#
S1	P1
S2	P2

مثال: نام شرکت‌هایی را برگردانید که در شهر ساوه قرار دارند.

$\pi_{SName}(\sigma_{City\,=\,"ساوه"}(S))$

مثال: کد قطعاتی را برگردانید که از نوع آهن باشند و در شیراز تولید شده باشند.

$\pi_{P\#} (\sigma_{City = 'شیراز' \wedge Type = 'آهن'}(P))$

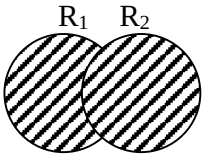
PROJECT (SELECT P WHERE City = 'شیراز' AND Type = 'آهن') [P#]

عملگرهای ∩ و ∪ و -

اجتماع ∪: اجتماع دو رابطه، رابطه‌ای است که تاپل‌هایش در یک یا هر دو رابطه وجود دارد.

یعنی رکوردهای هر دو جدول با هم ترکیب شده و تاپل‌های تکراری یک بار نوشته می‌شوند.

$R = R1 \cup R2$
R = R1 UNION R2

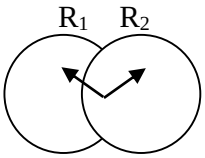


$R_1 \cup R_2$

اشتراک ∩: اشتراک دو رابطه، رابطه‌ای است که تاپل هایش در هر دو رابطه وجود داشته باشد.

به صورت زیر نمایش داده می‌شود:

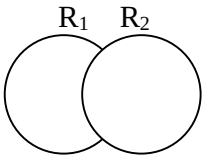
$R = R1 \cap R2$
R = R1 INTERSECT R2



$R_1 \cap R_2$

تفاضل: تفاضل دو رابطه، رابطه‌ای است که تاپل‌هایش در رابطه اول موجود است ولی در رابطه دوم وجود ندارد.

$R = R1 - R2$
R = R1 MINUS R2



$R_1 - R_2$

مثال: لیست شهرهای مربوط به تهیه کنندگان و قطعات را بدهید.

$\pi_{city} (S) \cup \pi_{city} (P)$ خروجی: ساوه، تهران، شیراز، تبریز

مثال: لیست شهرهایی که بین جدول قطعات و تهیه کنندگان مشترک اند را بدهید.

$\pi_{city} (S) \cap \pi_{city} (P)$ خروجی: ساوه و تبریز

مثال: اسامی شهرهای قطعاتی را بدهید که در رابطه S وجود ندارد.

$\pi_{city} (P) - \pi_{city} (S)$ خروجی: تهران و شیراز

چند مثال کنکوری:

مثال: رابطه S و SL به صورت زیر مفروض اند.

S (S# , Sname)
SL (S# , L# , Term# , Grade)

حاصل پرسوجوی زیر چیست؟

SL [S#] Minus (S WHERE Grade < > 20) [S#]

لیست #Sها در جدول SL — #Sهایی که نمره مخالف ۲۰ دارند.

- ✓ الف: شماره دانشجویانی که حداقل یک نمره بیست دارند.
- ب: شماره دانشجویانی که تمام نمره‌های آن‌ها بیست است.
- ج: شماره دانشجویانی که نمره بیست ندارند.
- د: شماره دانشجویانی که حداکثر یک نمره بیست دارند.

مثال: اگر A و B دو رابطه با عنوان یکسان باشند عبارت زیر معادل کدام رابطه است؟

$A - (A - B) \cup (B - (B - A))$

- ✓ الف: A INTERSECT B
- ب: A DEVIDED BY B
- ج: A JOIN B
- د: A MINUS B

چند نکته کنکوری:

۱- در تفاضل (-) خاصیت شرکت پذیری نداریم.

۲- روابط ورودی برای عملگرهای ∪ و ∩ و − باید همتا (Same Arity) باشند. یعنی تعداد صفت‌های ۲ رابطه در تعداد ستون‌های ۲ جدول مساوی باشند همچنین صفت‌ها به ترتیب دارای دامنه‌های یکسان باشد.

۳- عملگرهای اجتماع و اشتراک خاصیت جابه جایی دارند ولی تفاضل این خاصیت را ندارد.

$A - B \neq B - A$

عملگرهای پیوند یا الحاق (JOIN) :

این عملگرها پرکابرد و قدرتمند می‌باشند ولی گاهی اوقات باعث مصرف و زمان زیادی می‌گردد لذا موقع استفاده از آن‌ها باید دقت کافی را داشت.

ضرب دکارتی (×): گران ترین عملگرهای عملیاتی بانک اطلاعاتی که زمان و فضای زیادی می‌خواهد. بنابراین تا حد امکان باید از آنها اجتناب کرد. حاصلضرب ۲ رابطه رابطه‌ای که تاپل هایش از اتصال هر یک از تاپل‌های دو رابطه بدست می‌آید به عبارت دیگر در R1 × R2 هر سطر R1 را قبل از تمام سطرهای R2 قرار می‌دهیم.

نحوه نمایش ← R = R1 × R2

مثال: حاصلضرب دکارتی ۲ رابطه S و P را بنویسید.

S#	Sname	S. city	P#	Color	Type	P. city
S1	کاوه	ساوه	P1	قرمز	آهن	تهران
S1	کاوه	ساوه	P2	سبز	مس	تبریز
S1	کاوه	ساوه	P3	آبی	برنج	ساوه
S1	کاوه	ساوه	P4	قرمز	آهن	شیراز
S2	صفا	ساوه	P1	قرمز	آهن	تهران
S2	صفا	ساوه	P2	سبز	مس	تبریز
S2	صفا	ساوه	P3	آبی	برنج	ساوه
S2	صفا	ساوه	P4	قرمز	آهن	شیراز
S3	فن آور	تبریز	P1	قرمز	آهن	تهران
S3	فن آور	تبریز	P2	سبز	مس	تبریز
S3	فن آور	تبریز	P3	آبی	برنج	ساوه
S3	فن آور	تبریز	P4	قرمز	آهن	شیراز

مثال: اگر جدول A دارای m سطر و N ستون و B دارای p سطر و q ستون باشد آنگاه A × B چند سطر و چند ستون خواهد داشت.

سطر = M × P

ستون = N + q

نکته: × در ریاضات مجموعه‌ها خاصیت جابه جایی ندارد یعنی B × A # A × B است ولی در جبر رابطه‌ای چون ترتیب ستون‌ها مهم نیست خاصیت جابجایی دارد.

نکته: ضرب دکارتی خاصیت شرکت پذیری دارد.

عملگر پیوند طبیعی ∞:

این عملگر ۲ رابطه را بر مبنای یک یا چند فیلد مشترک به هم پیوند می‌دهد و رکوردهایی از جدول هایی که مقدارهایی از دو فیلد مشترک یکسان است به هم می‌چسبند. به عبارت دیگر فقط سطرهایی از دو جدول را کنار هم قرار می‌دهد که ستون‌های هم نام دو جدول مقادیر مساوی داشته باشد ستون‌های هم نام فقط یکبار در خروجی ظاهر می‌شود.

مثال: خروجی S ∞ SP چیست؟

S#	Sname	City	P#	Qty
S1	کاوه	ساوه	P1	۳۰۰
S1	کاوه	ساوه	P2	۲۰۰
S1	کاوه	ساوه	P3	۴۰۰
S2	صفا	ساوه	P1	۳۰۰
S2	صفا	ساوه	P2	۴۰۰
S3	فن‌آور	تبریز	P3	۲۰۰

توضیح: ستون مشترک بین Sو SP فیلد S# است پس هر سطر جدول S را برداشته بین سطرهایی از جدول S# قرار می‌دهیم که S# آن‌ها با هم برابر باشد.

نکته: پیوند طبیعی خاصیت شرکت پذیری و جابه جایی دارد.

نکته: اگر دو جدول فیلد هم نام داشته باشند R1 join R2 یعنی تبدیل به R1 × R2 می‌شود.

نکته: اگر تمام فیلدهای دو جدول یکسان باشند آنگاه R1 join R2 معادل R1 ∩ R2

نکته: پیوند طبیعی زمان و فضای کمتری نسبت به ضرب دکارتی می گیرد.

تمرین: خروجی S ∞ P را حساب کنید.

اسامی تهیه کنندگان قطعه P2 را لیست کنید.

خروجی S ∞ #S S3 لیست؟

عملگر جایگزین ←:

اگر دستوری طولانی باشد با این عملگر مقداری از دستور در یک متغیر موقتی ذخیره می‌شود تا بعداً بتوان از آن در خطوط بعد استفاده کرد.

مثال: اسامی تهیه کنندگان قطعه P2 را لیست کنید.

$\pi_{sname}(\sigma\ P\# = P2\ ^{(S \infty SP)})$

1. S join SP Giving temp1
2. select temp1 where P#="P2" Giving temp2
3. project temp2 [sname]

با هم ترکیب می‌کنیم:

Project (select (S join SP) where P#="P2")[sname]

Temp ← یک متغیر موقتی است که در این جا از temp استفاده شده است برای نمایش از Giving

استفاده می‌کنند که نوع نمایش آن به صورت زیر است:

1. $tmeP1 \leftarrow S \infty SP$
2. $temP2 \leftarrow \sigma\ P\# = "P2"\ (temp1)$

π_{sname} (temp2)

$\pi_{sname}\ [\pi_{S\#}\ (\sigma P\# = "P2"^{(SP)}) \infty S]$

$Temp1 \leftarrow \sigma\ P\# = "P2"^{(SP)}$

$Temp2 \leftarrow \pi_{S\#}\ (temp1)$

$Temp3 \leftarrow temp2 \infty S$

عملگر پیوند شرطی: همان ضرب دکارتی با این تفاوت که شرط θ بر روی آن اعمال می‌شود به صورت X_θ نمایش داده می‌شود که شرط‌ها به جای θ می‌نشیند.

مثال: خروجی دستور مقابل چیست؟

$SX_{S.city} \geq P.city^P$

عملگر نیم پیوند یا شبه الحاق:

مشابه پیوند طبیعی است اما فقط ستون‌های جدول اول را نشان می‌دهد و نحوه نمایش آن با علامت $\propto$ که به آن semijoin

مثال: مشخصات قطعاتی را لیست کنید که در ساوه ساخته شده و تعداد آنها از ۲۰۰ بیشتر باشد.

$\sigma\ Qty > 200^{(SP)} \propto \sigma\ city = "ساوه"^{(P)}$

خروجی: ساوه برنج آبی P3

نکته: عملگر پیوند خاصیت جابه جایی ندارد. یعنی $A \propto B \neq B \propto A$ چون همیشه ستون‌های جدول اول لیست می‌شود.

مثال: مشخصات تهیه کنندگانی را بدهید که قطعه P2 را تولید کرده اند.

$S \propto \sigma\ P\#="P2"^{(SP)}$

خروجی:

S#	Sname	City
S1	کاوه	ساوه
S2	صفا	ساوه
S3	فن‌آور	تبریز

عملگر تقسیم یا (÷ یا Divide By)

اگر R1 و R2 به صورت زیر باشد.

R1

S#	P#
S1	P1
S1	P2
S1	P3
S1	P4
S1	P5
S1	P6
S2	P1
S2	P2
S3	P2
S4	P2
S4	P5
S4	P6

R2

P#
P1

مثال: نتیجه $R1 \div R2$

جواب:

S#
$S1$
$S2$

توضیح: در حقیقت به سادگی لیست تولیدکنندگانی را داریم که قطعه P1 را تولید کرده اند.

مثال: لیست کد تهیه کنندگانی را نمایش دهید که تمام قطعات را تولید کرده باشند. (با توجه به R2 و R1 مسأله قبلی)

در اینجا S1 تمام قطعات P را تولید کرده است $(R1 \div (\pi_{P\#}(P))$

تمرین: اسامی تهیه کنندگانی را بیاورید که قطعه P2 را تولید نمی کنند.

تمرین: اسامی تهیه کنندگانی را بیاورید که حداقل یک قطعه قرمز رنگ را تهیه کرده اند.

تمرین: اسامی تهیه کنندگانی را بیاورید که تمام قطعات را تهیه می‌کنند.

الف: $(\pi_{sname}[S \infty (\sigma_{P\#\neq P2}(SP))]$ چون همه شرکت‌ها P2 را تولید کردند جدول تهی است.

ب: $\pi_{sname}[S \infty (\sigma_{color=\text{"قرمز"}}(P \infty SP))]$

ج: $[S \propto [\pi_{P\#,S\#}(SP) \div \pi_{PE}(P)]]$

مقدمات نرمال سازی روابط:

تعریف آنومالی (Anomaly): به معنی غیر متعارف و غیر معمول است. هرگاه در یک رابطه مشکلی غیر عادی در درج یا حذف یا در update کردن اطلاعات وجود داشته باشد در اصطلاح گفته می‌شود رابطه آنومالی دارد.

به طور مثال مشکلاتی از قبیل:

۱- عدم امکان انجام یک عمل که منطقاً قابل انجام باشد.

۲- بروز پیامد بد پس از انجام یک عمل

۳- بروز فزون کاری در سیستم در انجام یک عمل

اگر یک رابطه دارای چنین مشکلاتی باشد باید در اصطلاح نرمال تر شود.

در مبحث نرمال تر سازی رابطه به مفاهیمی از تئوری وابستگی تابعی نیاز داریم. بنابراین در ابتدا با برخی از این مفاهیم آشنا می‌شویم.

وابستگی تابعی (FD): پس فرض می‌کنیم که R یک رابطه و B و A دو زیر مجموعه دلخواه از صفات R باشد. در این صورت می‌گوییم B با A وابستگی تابعی دارد اگر و فقط اگر به ازای هر مقدار A فقط یک مقدار B متناظر باشد. ممکن است گفته شود صفت A صفت B را تعیین می‌کند و نشان داده می‌شود به این صورت $A \rightarrow B$

مثال: در این مثال نام تابعی از شماره است.

ولی صفت فامیلی تابعی از نام نیست.

نام خانوادگی	نام	شماره
حسینی	علی	۱۰۱
کریمی	علی	۱۰۲

دلایل: در مورد اول چون به ازای ۱۰۱ فقط یک نام و به ازای ۱۰۲ نیز فقط یک نام وجود دارد.

اما در مورد دوم به ازای علی بیش از ۱ فامیلی وجود دارد.

مثال: در جدول S هر یک از صفات Sname و city با صفت S# وابستگی تابعی دارد. چون به ازای هر مقدار S# فقط یک مقدار از Sname و city متناظر است.

- ✓ $S.S\# \rightarrow S.Sname$

$$\Rightarrow S.S\# (S.Sname , S.City)$$

- ✓ $S.S\# \rightarrow S.City$

تمرین: وابستگی‌های تابعی مربوط به جدول P را بنویسید.

$P\# \rightarrow color, type, city$
$Color \rightarrow type$
$City \rightarrow P\#$
$City \rightarrow type$

City → color
Type → color

نکته: واضح است که هر صفت که کلید کاندید و بویژه کلید اصلی باشد با تمام ستون‌های جدول یا صفات وابستگی تابعی دارد، چون کلید کاندید یکتایی مقدار دارد.

نکته: در A → B به سمت چپ عبارت، دترمینان و به سمت راست عبارت (وابسته) گفته شود.

نکته: اگر A → B لزوماً B تعیین می‌کند A را درست نیست.

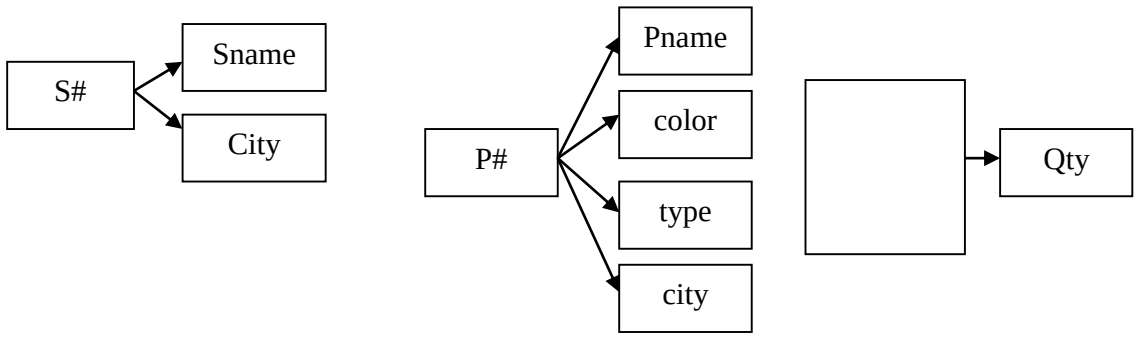
وابستگی تابعی کامل: در وابستگی A → B را اگر A را ترکیبی از چند صفت بگیریم مثلاً (S#, Sname) در رابطه S اگر B با هیچ کدام از زیر مجموعه‌های A وابستگی تابعی نداشته باشد می‌گوییم B با A وابستگی تابعی کامل دارد.

S#, Sname → city

مثال: در رابطه S صفت city با صفت مرکب Sname و S# وابستگی تابعی ندارد. یعنی ولی بستگی تابعی کامل نیست. چون S# با city وابستگی تابعی دارد. یعنی یکی از زیر مجموعه‌های S# و Sname وابستگی تابعی دارد.

مثال: در مثال SP صفت Qty با صفات مرکب (#s# , p) وابستگی تابعی کامل دارد. چون هیچ کدام از زیر مجموعه‌ها با Qty وابستگی تابعی ندارند.

نمودار وابستگی: برای نمایش وابستگی‌های تابعی یک بانک از نموداری به نام FDDigram یا نمودار وابستگی تابعی استفاده می‌شود. در این نمودار صفات در مستطیل قرار می‌گیرند. و یک فلش از آنها به هر یک از صفات وابسته وصل می‌شوند.



وابستگی تابعی بدیهی (tririal):

اگر در رابطه R (A, B, C ,...) صفت B زیر مجموعه‌ای از صفت A باشد در این صورت A→B را یک وابستگی تابعی بدیهی است.

مثال: (S#, Sname) → Sname دارد.

دقت کنید که وابستگی‌های تابعی بدیهی در مورد نظر طراح نیستند. بلکه طراح باید مجموعه وابستگی‌های مهم بین صفات راشناسایی کند. گاهی اوقات بعضی از وابستگی‌ها مخفی بوده و ممکن است در نگاه اول طراح آنها را نبیند. لذا باید بوسیله یک روش سیستماتیک وابستگی‌های دیگر را کشف کنیم.

قواعد استنتاج آرمسترانگ: طبق کشف و بیان شخصی به این نام اگر A, B, C زیر مجموعه هایی از صفات رابطه R باشند. قواعد زیر برقرارند:

۱- انعکاس یا بازتاب: $\text{If} \quad B \subseteq A \Rightarrow A \rightarrow B$

اگر B زر مجموعه A بود، آنگاه $A \rightarrow B$

۲- بسط پذیری: $\text{If} \quad A \rightarrow B \Rightarrow AC \rightarrow BC$

اگر $A \rightarrow B$,C یک صفت باشد. آنگاه، $AC \rightarrow BC$.را

$$\text{If} \quad AC \rightarrow BC \Rightarrow AC \rightarrow B,\ AC \rightarrow C$$

۳- انتقال یا تعدی: $\text{if} \quad A \rightarrow B\ ,\ B \rightarrow C \Rightarrow A \rightarrow C$

هر چند قواعد فوق برای استنتاج تمام وابستگی‌ها کامل بود اعمال آنها کار مشکلی بود. بنابراین بعدها دیگران قواعد دیگری را بیان کردند.

۴- اجتماع یا union: $\text{If} \quad A \rightarrow B\ ,\ A \rightarrow C \Rightarrow A \rightarrow BC$

۵- تجزیه decomposition: $\text{If} \quad A \rightarrow BC \Rightarrow A \rightarrow C,\ A \rightarrow B$

۶- ترکیب: $\text{If} \quad A \rightarrow B,\ C \rightarrow D \Rightarrow AC \rightarrow BD$

۷- خود تعیین: $A \rightarrow B$

۸- شبه تعدی: $\text{If} \quad A \rightarrow B,\ BC \rightarrow D \Rightarrow AC \rightarrow D$

۹- $\text{If} \quad A \rightarrow B,\ AB \rightarrow C \Rightarrow A \rightarrow C$

۱۰- اتحاد کلی یا یگانگی عمومی: $\text{If} \quad A \rightarrow B,\ C \rightarrow D \Rightarrow AV\ (C\text{-}B) \rightarrow BD$

تعریف بستار: هر رابطه‌ای یک مجمعه از وابستگی تابعی دارد. اگر این مجموعه (یعنی مجموعه وابستگی‌های تابعی یک رابطه)را F بنامیم ممکن است مجموعه‌ای دیگر از وابستگی‌های وابستگی‌های موجود در F استنتاج شود. مجموعه تمام FDهایی که از FDهای موجود در F قابل استنتاج هستند را بستار F می‌نامیم. و با F^+ نمایش می‌دهیم.

مثال: فرمول زیر را اثبات کنید.

$$A \rightarrow B,\ AB \rightarrow C \Rightarrow A \rightarrow C$$

$$A \rightarrow B \Rightarrow A \rightarrow AB$$
$$A \rightarrow AB\ ,\ AB \rightarrow C \Rightarrow A \rightarrow C$$

مثال: فرض کنید رابطه R با صفات FD , B, C, D, E, F های زیر موجودند.

$F= \{A \rightarrow BC, A \rightarrow E, CD \rightarrow EF\}$

آیا رابطه $Ad \rightarrow F$ درست است.

قانون ۵: $A \rightarrow BC \Rightarrow A \rightarrow B, A \rightarrow C$

قانون ۲: $A \rightarrow$ D, $\rightarrow CD$

قانون ۵: $CD \rightarrow EF \Rightarrow CD \rightarrow E, CD \rightarrow F$

قانون ۳: $AD \rightarrow CD, CD \rightarrow F \Rightarrow AD \rightarrow F$

توجه کنید که اگر این باشد $A\rightarrow BC$ در این صورت $A \rightarrow B, A \rightarrow C$ اما اگر $AC\rightarrow C$ را نمی توان نتیجه گرفت $A \rightarrow C, B \rightarrow C$ را.

مثال: اگر مجموعه وابستگی‌های زیر در رابطه R (A,B, C, D, E, F, G) مفروض باشد آنگاه کدام یک از وابستگی‌های زیر را می‌توان نتیجه گرفت؟

$\{A\rightarrow B, CD\rightarrow F, B\rightarrow D, B\rightarrow C, Be\rightarrow G, G\rightarrow F, AG\rightarrow B\}$

الف: $B\rightarrow G$

ب: $G\rightarrow B$

ج: $G\rightarrow D$

✓ د: $A\rightarrow F$

تمرین: از مجموعه FD زیر کدام رابطه را نمی توان اثبات کرد.

$F= \{A\rightarrow D, CD\rightarrow B, AD\rightarrow C\}$

✓ الف: $C\rightarrow D$

ب: $A\rightarrow B$ $\quad A\rightarrow CD, CD\rightarrow B \Rightarrow C\Rightarrow B$

ج: $A\rightarrow C$ $\quad A\rightarrow D, AD\rightarrow C \Rightarrow A\rightarrow C$

د: $A\rightarrow CD$ $\quad A\rightarrow C, A\rightarrow D \Rightarrow A\rightarrow CD$

کدام گزینه معدال مجموعه زیر است:

$\{(x,y,z) \rightarrow A,B,C,D, A\rightarrow B, x\rightarrow y\}$

✓ الف: $\{x,y,z) \rightarrow A,B,C,D, XA\rightarrow BY)\}$

ب: $\{x,y) \rightarrow A,C,D, A\rightarrow B, x\rightarrow y)\}$

ج: $\{y,z) \rightarrow A,C,D, A\rightarrow B, x\rightarrow y)\}$

د: $\{x,z) \rightarrow A,C,D, A\rightarrow B, x\rightarrow y)\}$

نکته: اگر در یک قسمت گزینه‌ها طوری بود که باید گزینه‌ها تک تک بررسی گردند از گزینه‌های ج و د شروع کنید.

تبدیل مجموعه وابستگی‌ها به مجموعه‌های بهینه:

با اعمال قواعد آرمسترانگ وابستگی‌های زیادی به وجود می‌آید که تعدادی از آن اضافه و تکراری است در ادامه بروشی برای حذف این گونه وابستگی‌های زائد و رسیدن به مجموعه وابسته بهینه ارائه می‌کند.

قواعد سه گانه برای تبدیل یک مجموعه وابستگی به مجموعه بهینه:

۱- وابستگی‌های تکراری و اضافه حذف شود.

۲- سمت راست هر وابستگی فقط یک صفت

۳- هر صفتی که E^+ را تغییر نمی دهد از سمت چپ حذف شود.

مثال: در بانک اطلاعاتی زیر مجموعه وابستگی بهینه را بدست آورید.

$R= \{U, V, W, X, Y, Z\}$
$F= \{U\rightarrow XY, X\rightarrow Y, XY\rightarrow ZV\}$

جواب مثال قبل:

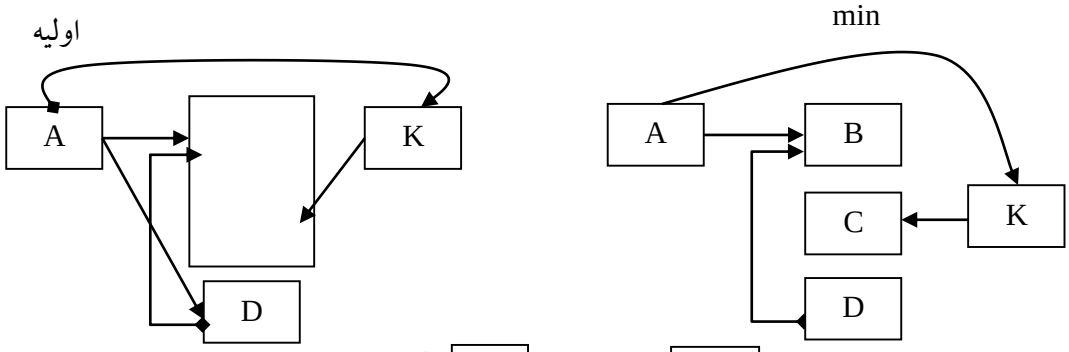
$u\rightarrow xy \Rightarrow \boxed{u \rightarrow x}, \boxed{u\rightarrow y}$ $\quad F^+= \{x\rightarrow y, u\rightarrow x, u\rightarrow y, u\rightarrow z, x\rightarrow z, u\rightarrow v, x\rightarrow y, x\rightarrow v\}$
$u\rightarrow xy, xy\rightarrow zv \Rightarrow u\rightarrow zv$
$u\rightarrow zv \Rightarrow \boxed{u\rightarrow z}, \boxed{u\rightarrow v}$
$xy\rightarrow zv \Rightarrow xy\rightarrow z , xy\rightarrow v$ $\quad F^+_{opt}= \{x\rightarrow y, u\rightarrow x, x\rightarrow y, u\rightarrow z, x\rightarrow z, u\rightarrow v, x\rightarrow v\}$
$x\rightarrow y, xy\rightarrow z \Rightarrow \boxed{x\rightarrow z}$
$x\rightarrow y, xy\rightarrow zv \Rightarrow x\rightarrow zv$ $\quad Fmin= \{x\rightarrow y, u\rightarrow x, x\rightarrow z, z\rightarrow v\}$
$x\rightarrow zv \Rightarrow \boxed{x\rightarrow z}, \boxed{x\rightarrow v}$

نکته: با توجه به این که وابستگی‌های $u\rightarrow v,\ u\rightarrow y,\ u\rightarrow z$ از طریق دیگر وابستگی‌ها بدست می‌آیند می‌توان آنها را نیز حذف کرد.

مثال: مجموعه وابستگی بهینه مجموعه وابستگی‌های زیر را بدست آورید.

$A\rightarrow(B, C)$	$A\rightarrow(B,C) \Rightarrow A\rightarrow B, A\rightarrow C$
$A\rightarrow D$	$A\rightarrow B, B\rightarrow D \Rightarrow A\rightarrow D$
$K\rightarrow C$	$A\rightarrow K, K\rightarrow C \Rightarrow A\rightarrow C$
$B\rightarrow D$	$F^+_{optm}= \{A\rightarrow B, A\rightarrow C, A\rightarrow D, A\rightarrow K, K\rightarrow G, B\rightarrow D\}$
$A\rightarrow K$	$F^+_{min}= \{A\rightarrow B, A\rightarrow K, K\rightarrow C, B\rightarrow D\}$
$(B, C) \rightarrow D$	

نمودار وابستگی‌های اولیه و min را رسم کنید.



تمرین: اگر R= [s,t,u,v,w] و F= [s→t, v→w, t→u] وابستگی بهینه را بدست آورید.

$s \to t,\ t \to u \Rightarrow \boxed{s \to u}$

$v \to sw \Rightarrow \boxed{v \to s},\ \boxed{v \to w}$

$v \to s,\ s \to u \Rightarrow \boxed{v \to u}$

$v \to s,\ s \to t \Rightarrow \boxed{v \to t}$

F^{+}_{optm} {s→t, t→u, s→u, v→s, v→w, v→u, v→t}

F^{+}_{min} {s→t, t→u, v→s, v→w, v→t}

نحوه بدست آوردن کلید کاندید: همانطور که گفته شد تمام صفات یک رابطه با صفت کلید وابستگی دارند. پس از بهینه کردن F^{+} با استفاده از قواعد زیر می‌توان کلید کاندید را شناسایی کرد.

قاعده ۱: حرکت کلید کاندید شامل مجموعه از صفت هایی است که در سمت چپ پیکان‌ها می‌آید.

قاعده ۲: کلید کاندید باید کمینه باشد. یعنی زیر مجموعه‌ای از آن خاصیت کلیدی نداشته باشد.

اگر w و s به عنوان کلید کاندید در نظر گرفته شود در حالی که خود s حالت کلیدی داشته باشد دیگر مجموعه sw کلید کاندید به حساب نمی آیند.

قاعده ۳: ممکن است چند کلید کاندید وجود داشته باشد.

قاعده ۴: کلیدهای کاندید ممکن است در یک یا چند صفت مشترک داشته باشند.

در مثال قبل در F^{+}_{optm} در مثال قبل چون v تمام صفات را تعیین می‌کند کلید کاندید است.

مثال: کدام گزینه کلید کاندید جدولی با ۵ ستون ABCDE است با توجه به وابستگی‌های زیر کلید کاندید را پیدا کنید.

{A→B, C, B→D, D→A}

۱) (D, E)

۲) (A, E) ✓

۳) (A, D)

۴) (A, ,D, E)

$A \rightarrow B, B \rightarrow D \Rightarrow A \rightarrow D$
$B \rightarrow D, D \rightarrow A \Rightarrow B \rightarrow A$
$A \rightarrow BC \Rightarrow A \rightarrow B, A \rightarrow C$

اگر یکی از صفات در هیچ یک از وابستگی‌های تابعی شرکت نکرده بود حتماًمی تواند یکی از اعضای کلید کاندید است.

نکته: اگر لیست صفات در متن سؤال آمده بود به متن سؤال آمده بود دقت کافی شود.

مثال: در بانک زیر کلید کاندید کدام است؟

$R= \{A, B, C, D, E, H\}$
$F= \{ A \rightarrow B, B \rightarrow C, D \rightarrow E, H \rightarrow A\}$

۱- EH
۲- HD
۳- ABD
۴- ADH

$A \rightarrow B, B \rightarrow C \Rightarrow A \rightarrow C$
$H \rightarrow A, A \rightarrow C \Rightarrow H \rightarrow C$
$H \rightarrow A, A \rightarrow B \Rightarrow A \rightarrow B$

$B \Rightarrow C$
$C \Rightarrow$
$D \Rightarrow E$
$E \Rightarrow$
$H \Rightarrow C, B, A$

ترکیب کنیم

$HD \rightarrow Ce, BE, EA$
$HD \rightarrow C, B, A, E$

نرمال سازی: (Normalization) مجموعه روش هایی که در طراحی بانک اطلاعاتی جهت کاهش افزونگی اطلاعات آنومالی‌ها استفاده می‌شوند و طبق آنها روابط بین موجودیت‌ها کنترل می‌شوند نرمال سازی گفته می‌شود.

در حقیقت هدف نرمال سازی ۲ چیز است.

۱- minimizing redundancy
۲- minimizing the insertion, deletion, ...

برای مثال فرض کنید یکی سوال کند چرا باید برای جدول‌های SP و S و P سه جدول جداگانه در نظر بگیریم. برای پاسخ به این سوال باید همه اطلاعات را در یک جدول در نظر بگیریم.

S#	Sname	S.city	P#	Color	type	p. city	Qty
S1	کاوه	ساوه	P1	قرمز	آهن	تهران	300

S1	کاوه	ساوه	P2	سبز	مس	تبریز	200
S1	کاوه	ساوه	P3	آبی	برنج	ساوه	400
S2	صفا	ساوه	P1	قرمز	آهن	تهران	300
S2	صفا	ساوه	P2	سبز	مس	تبریز	400
S3	فن آور	تبریز	P3	سبز	مس	تبریز	200

در این صورت همانطور که می‌بینید افزونگی داده داریم.

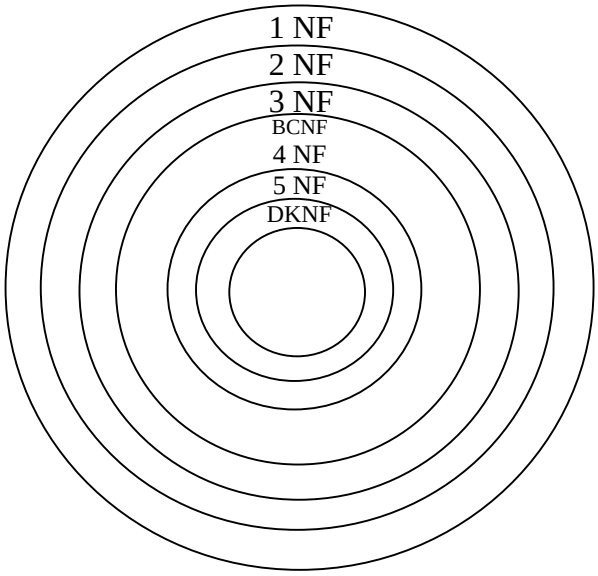
نکته: در پایگاه داده رابطه‌ای تنها افزونگی و تکرار داده که داریم و راهی جز آن نیست در مورد کلید اصلی است که در جدول دیگری به عنوان کلید خارجی عنوان می‌شود. تکرارهای بیش از کلید خارجی افزودگی داده نامیده می‌شود.

سطوح نرمال سازی: روند نرمال سازی که در سال ۱۹۷۰ توسط آقای کاد برای اولین بار ارائه شد شامل یکسری تست بود تا مشخص کند آیا یک رابطه به فرم نرمال رسیده است.

فرم نرمال یعنی حالت تجزیه شده جداول به طوریکه کمترین افزونگی انومال را داشته باشد.

کاد تا ۳ سطح نرمال سازی معرفی کرد و بعدها طراحان دیگر ستونی به اینها اضافه کردند تاکنون این صورت‌های نرمال معرفی شده اند.

First normal form ⇒ 1 NF
Second normal form ⇒ 2 NF
Third normal form ⇒ 3 NF
Boyce/GDD normal form ⇒ BCNF
Fourth normal form ⇒ 4 NF
Fifth normal form ⇒ 5 NF
Domain key normal form ⇒ DKNF



سطوح نرمال‌سازی در عمل: هر چند با ارائه سطوح نرمال‌سازی پایگاه داده‌های امروز بسیار بهینه تر از پایگاه داده‌های گذشته‌اند اما طراحان و کاربران بیش از سطوح فعلی را قابل درک نمی‌دانند. و در عمل نرمال سازی بیش از سطوح BCNF و ۴ NF و... کاربرد ندارد.

نکته: ممکن است یک رابطه نرمال باشد یا نباشد. اگر نرمال باشد حداقل NF ۱ است.

نکته: هر سطح نسبت به سطح بالایی خود نرمال تر است. یعنی اگر یک رابطه در سطح NF ۳ باشد حتماً سطوح NF ۱و NF ۲ را موفقیت طی کرده است.

نکته: در کل نرمال سازی یعنی پیروی از یکسری فرم‌های نرمال که منجر به تجزیه جداول می‌شود.

فرم NF ۱: اگر در یک رابطه یا جدول صفت ترکیبی نداشته باشیم آن رابطه در سطح NF ۱ است. به طور مثال اگر جدولی شامل فیلد آدرس باشد و خود فیلد آدرس از فیلدهای کوچکتری مثل نام استان و نام شهر و... تشکیل شده باشد جدول مورد نظر NF ۱ است.

دلیل دیگری یک رابطه را غیر NF ۱ می‌کنداین است که یک فیلد با کلیدهای اصلی رابطه وابستگی تابعی کامل نداشه باشد.

تعریف صفت عمده Prime ATTR: صفتی است که کلید اصلی یا جز از کلید اصلی باشد.

فرم NF۲: رابطه R در حالت فرم دوم نرمال است اگر (اولاً NF۱ باشد و ثانیاً) هر صفت غیر کلید با کلید اصلی رابطه R وابستگی تابعی کامل داشته باشد.

مثال: فرض کنید جدول S و SP را یک جدول در نظر بگیریم.

S #	status	city	P #	Qty
S1	20	C2	P1	300
S1	20	C2	P2	200
S1	20	C2	P3	400
S2	10	C3	P1	100
S2	10	C3	P2	200
S3	10	C3	P3	200

در این جدول کلید اصلی ترکیب P# و S# است. درست است که این به خوبی City را تعیین می‌کند اما S# نیز به تنهایی می‌توان City را تعیین کند. یعنی City که یک صفت غیر کلید است با کلید اصلی رابطه وابستگی تابعی کامل ندارد. وجود این مشکل باعث ایجاد آنرمالی هایی در درج و افزونگی داده‌ها می‌شود.

برای رفع این مشکل جدول به دو جدول SP و S تجزیه می‌کنیم.

S

S#	status	city
S1	20	C2
S2	10	C3
S3	10	C3

P

S#	P#	Qty
S1	P1	300
S1	P2	200
S1	P3	400
S2	P1	100
S2	P2	200
S3	P3	200

فرم NF۳: رابطه A در فرم NF۳ است اگر (اولاً در فرم NF۲ باشد و ثانیاً) هیچ صفت غیر کلید از R با R وابستگی با واسطه یا انتقالی نداشته باشد.

تعریف وابستگی با واسطه انتقالی:

اگر صفت B از رابطه R با صفت A از همین رابطه وابستگی $A \rightarrow B \rightarrow C$ تابعی داشته باشد و صفت C از این رابطه با صفت B در این رابطه وابستگی تابعی داشته باشد (و A با B وابستگی تابعی نداشته باشد) در این صورت گفته می‌شود C با A وابستگی انتقالی یا با واسطه دارد.

مثال: به جدول P دقت کنید.

P#	PName	color	weight	city
P1	PN1	Red	100	C1
P2	PN2	Blue	200	C2
P3	PN1	Blue	100	C3
P4	PN2	Blue	200	C1
P5	PN3	Red	100	C2

این در سطح NF۳ است اما جدول S در مثال قبل بدست آمد در سطح NF۳ نیست چون فیلد غیر کلید status وابسته به فیلد غیر کلید است. (یعنی city، Status را تعیین می‌کند.)

چون فقط P# بقیه را تعیین می‌کند و بقیه اصلاً همدیگر را تعیین نمی کنند.

برای رفع این مشکل می‌توان جدول S را به دو جدول تبدیل کرد.

S (S# , city)

S#	city
S1	C2
S2	C3
S3	C3
S4	C2
S5	C1

CS (city , status)

city	status
C1	30
C2	20
C3	10

نکاتی در مورد تجزیه رابطه:

۱- تجزیه مطلوب: تجزیه‌ای که در آن با پیوند رابطه‌های حاصل از تجزیه همان محتوای اطلاعاتی رابطه اولیه بدست آید و اطلاعات زاید و ناموجود در رابطه اول پدید نیاید و همچنین تمام وابستگی‌های تابعی رابطه اولیه محفوظ بماند. تجزیه مطلوب است.

۲- بر اساس ضوابط رسانن Ressanen اگر در رابطه (RCABC) اگر رابطه A→B و B→C در این صورت تجزیه مطلوب به صورت زیر است.

$$(R\ (A,B,C) \Rightarrow R1\ (A,B)\ R2(B,C$$

۳- قضیه heath: اگر با توجه به این رابطه (R(A,B,C، A→B آنگاه می‌توان R را به هر رابطه تجزیه کرد (R1 =(A,B و (R2 =(A,C و این تجزیه مطلوب است (البته اگر ضوابط رسانی برقرار نباشد.)

الگوریتم‌های تبدیل NF۱ به NF۲ و NF۲ به NF۳:

مثال: رابطه (R=(ABCDEFG با کلید اصلی BCD با مجموعه وابستگی‌های زیر مفروض است؟

FD={ BCD→E , BCD→B , BCD→F , BD→A , F→G , E→F , C→E}

در صورت تبدیل آن به NF۳ چند رابطه‌ایجاد خواهد شد؟

الف) ۶ ب) ۴ ج)۳ د) ۵

برای تبدیل NF۱ به NF۲ به صورت زیر عمل می‌کنیم.

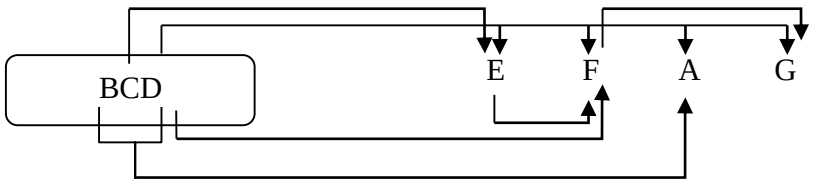
۱- هر بخش از کلید اصلی را که صفت وابسته دارد با آن صفت‌ها کنار هم قرار می‌دهیم.

۲- کل کلید اصلی را با صفت‌های باقیمانده کنار هم قرار می‌دهیم.

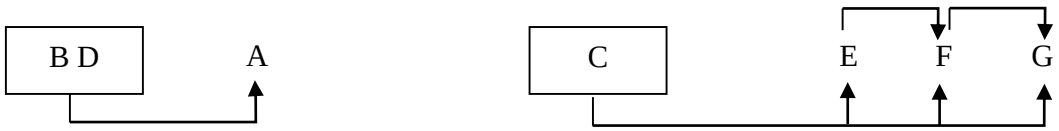
۳- سایر وابستگی‌ها را ترسیم می‌کنیم.

اگر وابستگی به بخشی از کلید به صورت تو در تو بود الگوریتم فوق باید تکرار شود.

جواب: در مورد مثال قبل فرم NF۱ به صورت زیر است.



در حقیقت هیچ صفتی نباید به قسمتی از کلید اصلی وابسته باشد. در سطح NF۲ داریم:



سپس الگوریتم جدول NF۲ به NF۳ تبدیل می‌کنیم.

الگوریتم تبدیل جدول NF۲ به NF۳ به صورت زیر است.

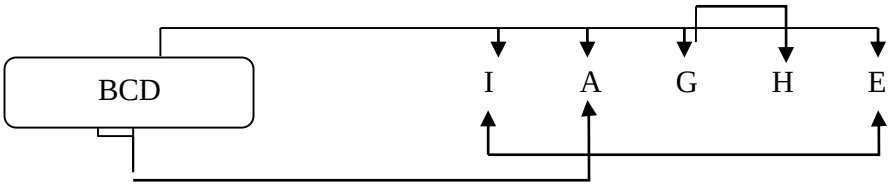
۱- صفت هایی را که وابستگی انتقالی ایجاد کرده اند با وابستگی آنها کنار هم قرار می‌دهیم.

۲- کلید اصلی را با صفت‌های باقیمانده کنار هم قرار می‌دهیم.

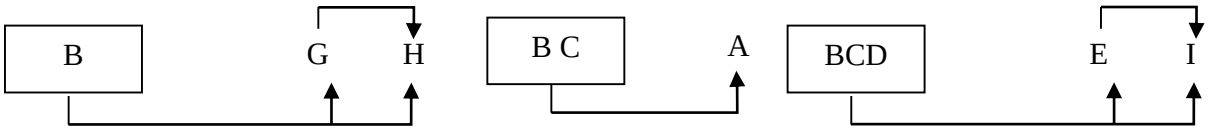
۳- صفت‌های کلیدی را به عنوان کلید خارجی را در رابطه دوم تکرار می‌کنیم.

مثال: با نرمال سازی رابطه R تا سطح NF۳ چند جدول به جا می‌ماند؟

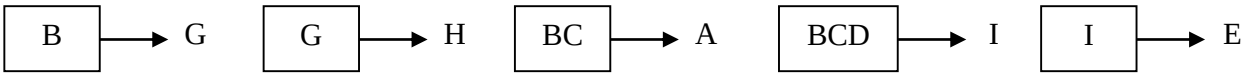
R (A, B, C, D, E, F, G, H, I)
FD={ BC→A , B→G , BCD→I , G→H , E→I , BCD→E , BCD→A}



تبدیل NF۱ به NF۲:



تبدیل NF۲ به NF۳:



مثال: اگر رابطه (R(A B C D E F و کلید اصلی BCD باشد آنگاه کدام وابستگی تابعی زیر شرط NF۲ بودن رابطه R را نقص می‌کند.

الف) BC → E ب) DCB → A ج) E → F د) A → EF

فرم BCNF: گاهی اوقات یک رابطه هر چند در فرم NF۳ است اما باز هم شاهد افزونگی داده‌ها هستیم در این شرایط باید تجزیه را بیشتر ادامه دهیم. به طور مثال در مواقعی که:

۱- رابطه دارای چند کلید کاندید باشد.

۲- کلیدهای کاندید رابطه مرکب باشند.

۳- کلیدهای کاندید در یک صفت اشتراک داشته باشند.

مثال: جدول و رابطه زیر را در نظر بگیرید.

S#	Sname	P#	Qty
S1	Sn1	P1	300
S1	Sn1	P2	200
S1	Sn1	P3	400
S1	Sn1	P4	200

کلیدهای کاندید: (S#,P#) ، (Sname ,P#)

هر چند رابطه در فرم [illegible] رنگی داریم. برای رفع مشکل فوق باید جدول را به دو جدول زیر تعیین کنیم.

S (S# , Sname) SP (S# , P# , Qty)

نکته: رابطه‌ای که BCNF باشد حتماً NF۳ است ولی هر رابطه NF۳ لزوماً BCNF نیست.

S#	Sname
S1	Sn1
S1	Sn1
S1	Sn1
S1	Sn

نکته: در عمل باید نرمال سازی را تا حد BCNF ادامه داد مگر اینکه با این کار وابستگی‌ها حذف شوند. در این صورت تا سطح NF۳ کافی است.

S#	P#	Qty
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200

مثال: اگر رابطه R(ABCDED) و کلید اصلی BCD باشد آنگاه کدام وابستگی تابعی زیر شرط nf۲ بودن رابطه R را نقص می‌کند.

الف: $BC \rightarrow E$

ب: $DCB \rightarrow A$

ج: $E \rightarrow F$

د: $A \rightarrow EF$

فرم BCNF: گاهی اوقات یک رابطه هر چند در فرم NF۳ است اما باز هم شاهد افزونگر داده‌ها هستیم در این شرایط باید تجزیه را بیشتر ادامه دهیم به طور مثال در مواقعی که:

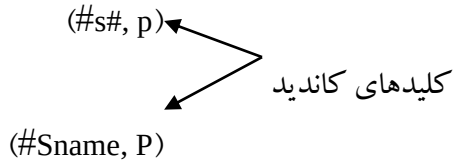
۱- رابطه دارای چند کلید کاندید باشد.

۲- کلیدهای کاندید رابطه مرکب باشند.

۳- کلیدهای کاندید در یک صفت اشتراک داشته باشند.

مثال: جدول و رابطه زیر را در نظر بگیرید.

S#	Sname	P#	Q
S1	Sn1	P1	3
S1	Sn1	P2	2
S1	Sn1	P3	4
S1	Sn1	P4	2



هر چند رفع مشکل فوق

باید جدول را به دو جدول زیر تعیین کنیم.

S (S#, Sname)
SP (S#, P#, Qty)

نکته: رابطه‌ای که BCNF باشد حتماً NF۳ است ولی هر رابطه NF۳ لزوماً BCNF نیست.

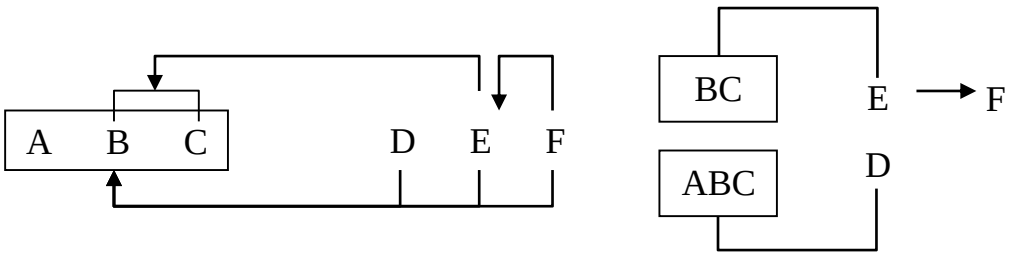
S#	Sname
S1	Sn1
S1	Sn1
S1	Sn1
S1	Sn1

نکته: در عمل باید نرمال سازی را تا حد BCNF ادامه داد مگر انیکه با این کار وابستگی‌ها حذف شوند. در این صورت تا سطح NF۳ کافی است.

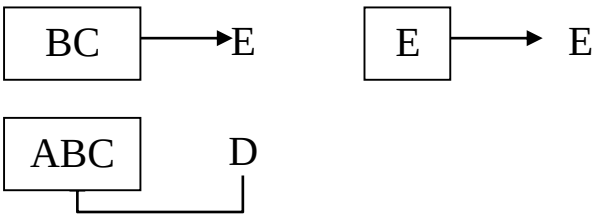
S#	P#	Qty
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200

مثال: رابط زیر در حالت NF۱ است آن را به حالت NF۳ ببرید.

NF۲



NF۳



معماری پایگاه داده: Database Architect

این معماری به معماری سه سطحی نیز مشهور است.

اجرای معماری db:

۱- external view دید خارجی

۲- conceptual view دید مفهومی یا ادراکی

۳- internal view دید داخلی

۴- mapping تبدیلات بین سطوح

۵- DSL (data sub-language) زبان داده‌ای فرعی

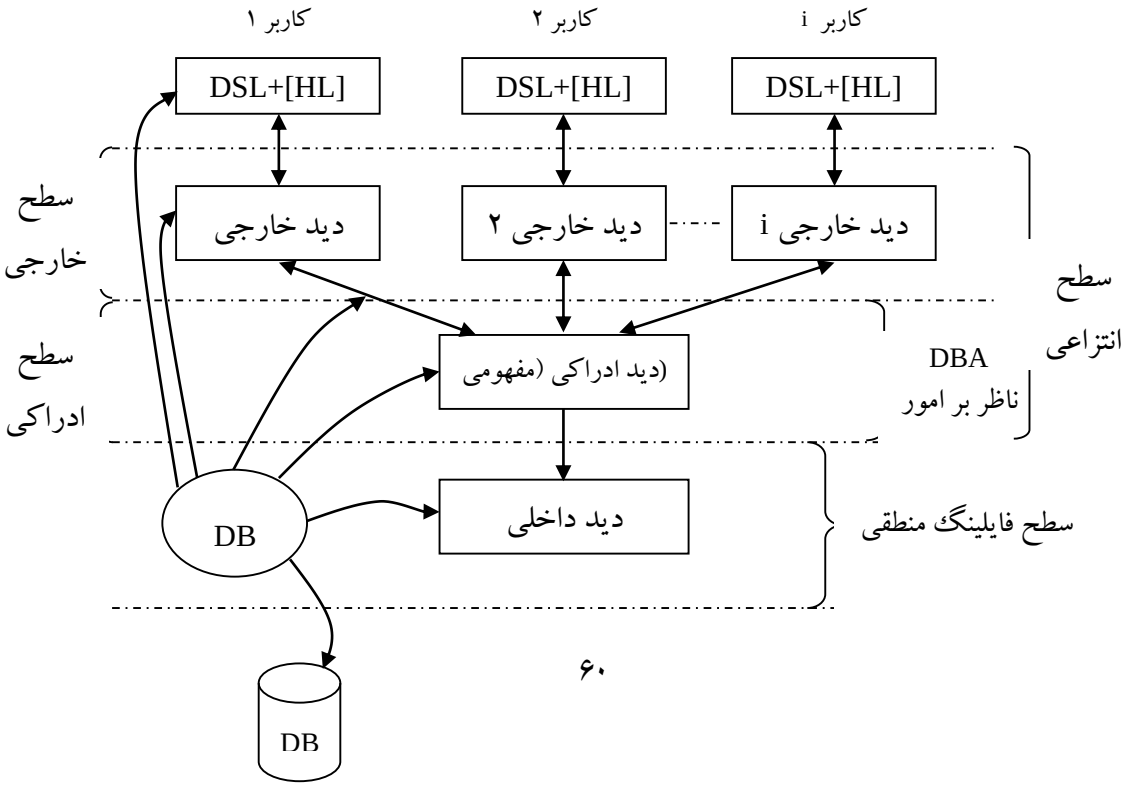
۶- HL (Host language) زبان میزبان

۷- user کاربر

۸- DBMS سیستم مدیریت پایگاه داده ای

۹- DBA مدیر پایگاه داده

ممکن است به سطح داخلی فیزیکی نیز گفته شود.



شرح اجزای معماری DB:

دید ادراکی یا مفهومی:

- دید ادراکی دید طراح است نسبت به داده‌های ذخیره شده در Database
- این دید مبتنی است بر یک DS مشخص (جدولی یا رابطه‌ای...)
- طراح سیستم به کمک دستورات باید این دید را توصیف کند.
- در این سطح ساختار داده ها. ارتباط بین موجودیت ها، امنیت و جامعیت داده‌ها و اطلاعات معنایی داده‌ها مطرح می‌گردد.
- نام‌های دیگری چون سطح مفهومی (سطح منطقی اجتماع) یا سطح منطقی نیز به این سطح گفته می‌شود.

مثال: پس از بررسی محیط عملیاتی دانشگاه موجودیت‌های درس و دانشجو مشخص می‌شود، همین که طراح پایگاه داده دو جدول برای موجودیت‌ها و به طور مثال یک جدول برای رابطه بین آنها در نظر می‌گیرد و تعریف می‌کند در حقیقت با دید ادراکی خود در اصطلاح (شمای ادراکی) را تعریف کرده است.

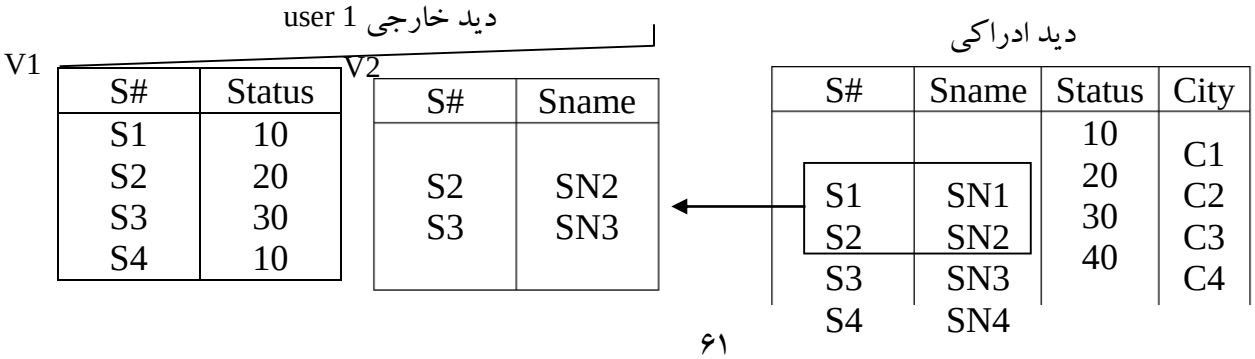
دید خارجی:

- دید کابر است نسبت به داده‌های ذخیره شده در Database
- این دید معمولاً مبتنی بر BS دید ادراکی است. و اگر مبتنی بر BS دید ادراکی نباشد می‌گوییم سیستم دو ساختاره است.
- این دید جزئی است و فقط در حد نیازهای اطلاعاتی یک کاربر است.
- چند کاربر می‌توانند در یک دید اشتراک داشته باشند.
- DBA دید مربوط به هر کاربر را تعیین می‌کند و می‌تواند بیش از یک دید به کاربر بدهد.

(منظور از دید یا view در حقیقت سطح دسترسی است)

مثال: DBA پایگاه داده‌های بانک ملی مرکزی به یک بانک شهرستانی فقط اجازه دیدن شماره حساب و نام کاربران را بدهد نه اجازه مشاهده شماره تلفن و آدرس و ....

مثال: در مورد جدول S می‌توان ۲ کاربر با viewهای مختلف زیر را تعیین کنید.



ایجاد View در SQL:

به طور مثال برای ساخت v1 برای کاربر usser1 داریم

S#	Status
S1	10
S2	20
S3	30
S4	10

```
Creat view V1 (S#, Status)
AS Select S#, Status From S;
```

برای ساخت V2

```
me)
AS Select S#, Sname From S where S.City = 'C2';
```

برای ساخت V3

```
Creat view V3 (S#, Status)
AS Select S#, Status From S;
```

نکته: نزدیکترین سطح به کاربر سطح خارجی است.

دید داخلی

- دید DBMS است نسبت به داده‌های ذخیره شده در DB
- DBMS در سطح داخلی پایگاه داده
- فایل‌ها در سطح داخلی باید تعریف و ایجاد شود.
- به تعریف داده‌ها و اطلاعات کنترلی شمای داخلی گفته می‌شود.
- در شمای داخلی انواع رکوردها، فایل ها، صفات خاصه، تخصیص فضای ذخیره سازی برای داده ها، محل رکوردها، فشردگی داده ای، تکنیک‌های رمزگزاری داده‌ها وغیره تشریح می‌شود.
- در حقیقت مباحث این سطح مربوط به درس ذخیره و بازیابی اطلاعات است.

مثال: در سطح ادراکی جدول‌های P و S و SP را داریم اطلاعات این جدول‌ها در سطح فیزیکی و داخلی در فایل هایی به نام Sfile-Pfile و SPfile نگه داری می‌شوند.

نکته: معمولاً بین سطح داخلی و ادراکی تناظر ۱ به ۱ وجود دارد یعنی هر جدول ۱ فایل اما ممکن است گاهی برای یک جدول بزرگ بیش از یک فایل در نظر گرفته شود.

کاربر: هر استفاده کننده از محیط DB یک کاربر در نظر گرفته می‌شود.

کاربر
- برنامه ساز
  - برنامه ساز سیستمی: برنامه‌های ایجاد کنترل DBرا می‌نویسند.
  - برنامه ساز کاربردی: برنامه‌های بهره برداری از DB را می‌نویسند.
- نابرنامه ساز

UFi: کاربر از طریق یک UFi از سیستم استفاده می‌کنند که این UFI می‌تواند:

- منو محور
- صوت محور
- فرم محور
- تصویر محور
- خط فرمان محور

باشد.

HL: یک زبان سطح بالا و گاهی زبان اسمبلی است و در مواردی زبان میزبان مکمل زبان پایگاهی است. زبان هایی مثل php و sp برای اتصال به سیستم پایگاه داده Mysql و CH و VB برای اتصال DBMS SqL سرور.

DSL: یک نام عمومی برای اشاره به زبان‌های پایگاه داده‌ای مثل SqL است

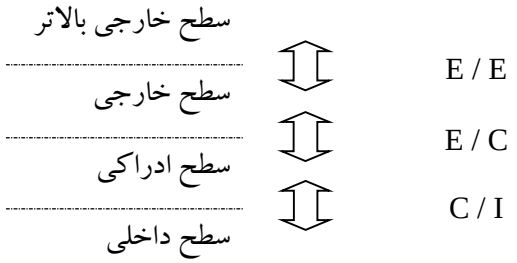
از نظر استاندارد ANSI یک DSL باید سه دسته دستور را پشتیبانی کند.

DDL و DML و DCL

برخی خصوصیات یک DSL خوب:

- باید نزدیک به زبان محاوره‌ای باشد تعداد دستورات اصلی کم باشد.
- یادگیری آن آسان باشد.
- کمترین افزونگی را در دستورات داشته باشیم،
- بهتر است نارویه‌ای باشد نه رویه ای.

تبدیلات بین سطوح:



ایده اصلی در Mapping: هر حکم و یا دستور عمل کننده در سطح خارجی باید تبدیل شود به ادراکی و سپس هر حکم یا دستور عمل کننده در سطح ادراکی باید تبدیل شو به دستور عمل شونده در سطح داخلی و نهایتاً در DB فیزیکی.

این تبدیلات توسط DBMS انجام می‌شود و از دلائل مهم Overhead در DBMSها عملیاتی است که سیستم برای این Mappingها انجام شود.

مثال: دستور Select * From S در سطح خارجی است اما در سطح فیزیکی یعنی اطلاعات فایل Sfile نمایش داده شود.

مثال:

سطح خارجی

Scode	Sta	City
S1	10	C1
S4	16	C4

سطح ادراکی

S#	Sname	Status	City
S1	SN1	10	C1
S2	SN2	17	C2
S3	SN3	14	C3
S4	SN4	16	C4

سطح داخلی Sfile Pfile SPfile

دستور:

```
Create view my view (Scode, Sta, city)
As select S#, Status, city From S where
City! = 'C2';
```

اگر کاربری دستور زیر را صادر کند در سطح ادراکی مفهوم view معنی ندارد پس باید تبدیل

```
Select Scode From my view where Sta < 15;
Select S# From S where Status < 15 and city ! = 'C2'
```

در حقیقت دستور سطح خارجی با دستور ایجاد دید my view، and می‌گردند تا یک Query قابل اجرا در سطح ادراکی تولید می‌شوند. همین دستور درسطح داخلی چیزی شبیه به دستورات زیر است.

```
Open Sfile (Read, ...)
Lread Sfile (LBuf, ...)
On status. Index value < 15;
If Sfile. Rec. city! = 'C2'
More Sfile. Rec into LBUF;
Move Sfile. S# into use;
.
.
.
.
.
Close Sfile;
```