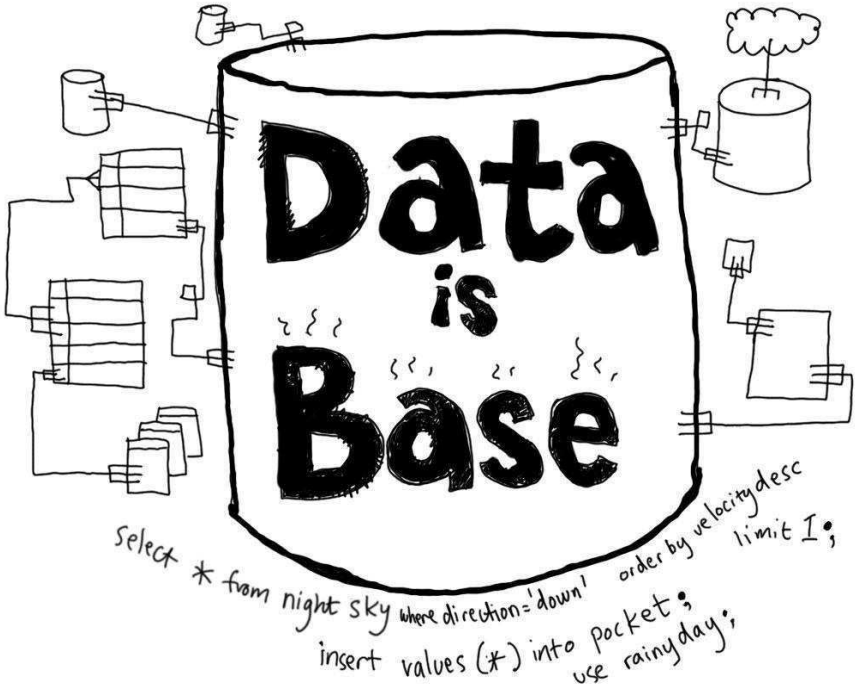
جزوه درس

پایگاه داده ها



مدرس :

حمید عرب‌نژاد

پایگاه داده‌ها یا بانک اطلاعاتی (DataBase) :

- یک سیستم بانک اطلاعاتی چیزی بیش از یک سیستم کامپیوتری نگهداری رکوردها نمی‌باشد. یعنی پایگاه داده‌ها در واقع سیستمی است که وظیفه عمومی آن ضبط و نگهداری و ارائه آنها بنا به درخواست می‌باشد.
- مجوعه‌ای از فایلها یا پرونده‌ها که به نحوی منطقاً بهم مرتبط می‌باشند، پایگاه داده‌ها یا بانک اطلاعاتی نامیده می‌شود. به بیانی دیگر مخزنی از اجتماع و اشتراک داده‌ها یا اطلاعات پایگاه داده‌ها یا بانک اطلاعاتی نام دارد.
- بانک اطلاعاتی محلی جهت نگهداری داده‌ها است که فعالیت‌های مربوط به ساختار یک تشکیلات را توضیح می‌دهد و متشکل از پرونده‌های اطلاعاتی می‌باشد.
- در تعریفی کاملتری از پایگاه داده‌ها می‌توان گفت که پایگاه داده‌ها مجوعه‌ای است از داده‌های ذخیره شده به صورت مجتمع(یکپارچه – Integrated) (نه لزوماً بطور فیزیکی،بلکه حداقل بطور منطقی) بهم مرتبط و مبتنی بر یک ساختار با کمترین افزونگی (Redundancy) تحت مدیریت یک سیستم کنترل متمرکز، مورد استفاده یک یا چند کاربر بطور همزمان و اشتراکی .

داده (Data) و اطلاعات (Information) :

دو اصطلاح داده و اطلاعات واژه‌هایی هستند که بیشتر اوقات بجای یکدیگر بکار برده می‌شوند ولی چنین تعبیری در سیستمهای اطلاعاتی صحیح نمی‌باشد.

برخی از تعاریف ارائه شده در مورد "داده" چنین‌اند:

- داده عبارتست از نمایش ذخیره شده اشیاء فیزیکی ، چیزهای مجرد ، بوده‌ها(واقعیتها) ، رویدادها یا موجودیتهای دیگر قابل مشاهده که در تصمیم‌گیری بکار می‌آیند.
- داده عبارتست از بوده(واقعیت) یا هست معلوم که می‌توان بوده یا هست دیگری را از آن استنباط کرد.

برخی از تعاریف ارائه شده در مورد "اطلاع" چنین‌اند:

- اطلاع به داده‌ای اطلاق می‌شود که توسط یک فرد یا سازمان برای تصمیم‌گیری بکار می‌رود.
- اطلاع داده پردازش شده است

در واقع می‌توان گفت که "داده" مقادیری هستند که بطور واقعی در بانک اطلاعاتی ذخیره می‌شوند و "اطلاعات" به مقادیری که توسط کاربران شناخته می‌شود (داده‌های پردازش شده) گفته می‌شود.

رابطه بین داده‌ها و اطلاعات همانند رابطه خبر با اطلاع است. هنگامی که خبری دریافت می‌گردد، پرورش می‌یابد و پس از ارزشیابی به اطلاع تبدیل می‌گردد. داده‌ها نیز پردازش می‌شوند تا اطلاعات پدید آورند. فرق بین داده و اطلاع آن است که داده صرفاً نمایشی از واقعیت‌ها، معلومات، مفاهیم، رویدادها یا پدیده‌ها می‌باشد، لیکن اطلاع دارای خاصیت ارتباط دهندگی و انتقال دهندگی می‌باشد که پس از پردازش یا تفسیر داده حاصل می‌گردد.

داده‌ها و اطلاعات در قالب‌های منطقی و فیزیکی

شرح داده‌ها و اطلاعات و روابط بین داده‌ها و اطلاعات در دوقالب منطقی و فیزیکی تجلی می‌یابند. شرح داده‌ها یا اطلاعات فیزکی به روشی که در آن داده‌ها یا اطلاعات بر روی رسانه ذخیره‌سازی، ذخیره و نگهداری می‌گردند، اشاره دارد. لیکن شرح داده‌ها یا اطلاعات منطقی به روشی که داده‌ها یا اطلاعات به برنامه‌نویس کاربردی یا کاربر ارائه می‌گردند، اشاره دارد. واژه‌های فیزیکی و منطقی برای بیان جنبه‌های مختلف داده‌ها و اطلاعات به‌کاربرده می-

شود. واژه فیزیکی نشانگر نحوه ذخیره داده‌ها و اطلاعات بر روی رسانه ذخیره‌سازی است و واژه منطقی بیانگر روشی است که برنامه‌نویس یا کاربر آنها می‌بیند. واژه‌های فیزیکی و منطقی برای بیان جنبه‌های مختلف داده و اطلاعات بکاربرده می‌شود. واژه فیزیکی نشانگر نحوه ذخیره داده‌ها و اطلاعات بری روی رسانه ذخیره‌سازی است و واژه منطقی بیانگر روشی است که برنامه‌نویس یا کاربر آنها را می‌بیند.

یک رکورد فیزیکی ممکن است حاوی چند رکورد منطقی باشد تا در فضای ذخیره سازی یا زمان دستیابی صرفه-جویی بعمل آید.

در حقیقت رکورد فیزیکی واحد اساسی داده‌هاست که در هر بار خواندن یا نوشتن با یک دستور منفرد ورودی/خروجی انتقال می‌یابد و مجموعه‌ای از داده‌هاست که مابین فاصله‌های خالی (Gap) بر روی نوار یا نشانگرهای آدرس (Address markers) بر روی دیسک ذخیره می‌گردند. یک رکورد فیزیکی غالباً حاوی چند رکورد منطقی می-باشد.

اجزای یا عناصر محیط پایگاه داده‌ها :

محیط پایگاه داده‌ها از ۴ بخش زیر تشکیل شده است:

۱- داده ۲-سخت‌افزار ۳- نرم‌افزار ۴- کاربر

۱-داده :

یکی از مهمترین اجزاء پایگاه داده‌ها است که در واقع پل ارتباطی بین اجزاء ماشین و محیط انسانی است که می‌تواند هم مجتمع و هم اشتراکی باشد.

مفهوم "**مجتمع بودن**" بدین معنی می‌باشد که بانک اطلاعاتی ممکن است به عنوان وحدت چندین فایل اطلاعاتی مجزا با هرگونه تکرار در داخل این فایل‌ها که تماماً یا قسمتی حذف شده باشد، در نظر گرفته می‌شود. به عنوان نمونه در یک سیستم پرسنلی که از دو فایل اطلاعاتی، پرسنلی و ساعات کاری تشکیل شده است لزومی ندارد که در فایل دوم (ساعات کاری) مشخصات شخص موردنظر بطور کامل آورده شود و می‌توان مشخصات پرسنلی فرد موردنظر را با مراجعه به فایل اول(مشخصات پرسنلی) بدست آورد.

مشخصات پرسنلی	ID	Name	Family	Salary	Address	……
ساعات کاری	ID	Time	Day	……		

منظور از "**مشترک بودن**" بانک اطلاعاتی این است که قسمت‌های مجزایی از داده‌ها در یک بانک اطلاعاتی می‌توانند بین چندین کاربر مختلف مشترک باشد.

داده‌های مانا یا پایدار

مرسوم است که به داده‌های یک بانک اطلاعاتی به عنوان داده‌های ثابت اشاره شود هرچند که ممکن است این داده‌ها برای مدت زیادی ثابت نباشند. منظور از "**ثابت**" این است که داده‌های بانک اطلاعاتی به لحاظ نوع از دیگر داده‌ها مانند داده‌ههای ورودی، داده‌های خروجی، دستورات کنترل، صف‌های کاری و به طور کلی هر داده‌های که ماهیتاً گذار باشد، متمایز می‌باشد . حال بطور مختصر به شرح مختصری از مفاهیم داده‌های ورودی و خروجی می‌پردازیم :

داده‌های ورودی : به اطلاعاتی اشاره می‌کند که در ابتدا به سیستم وارد می‌شود. بر روی چنین داده‌هایی برای این که به عنوان داده ثابت تلقی گردند، تغییراتی اعمال گردد ولی آنها جزئی از بانک اطلاعاتی نمی‌باشند.

داده‌های خروجی : به پیامها و نتایج حاصل از سیستم اطلاق می‌گردد. مجدداً، چنین اطلاعاتی ممکن است از داده‌های ثابت نتیجه شوند ولی خود به‌عنوان بخشی از بانک اطلاعاتی درنظر گرفته نمی‌شوند.

۲- سخت‌افزار

در محیط پایگاه داده‌ها، همانند هر محیط ذخیره و بازیابی اطلاعات سه دسته سخت‌افزار وجود دارد :

- **سخت‌افزار ذخیره‌سازی اطلاعات** : منظور همان رسانه ذخیره‌سازی خارجی است همانند دیسک که رسانه اصلی ذخیره‌سازی می‌باشد. البته نوار مغناطیسی هم در محیط‌های پایگاه داده‌ای به کار برده می‌شود ولی نه به عنوان رسانه اصلی بلکه به عنوان رسانه کمکی برای تولید نسخه‌های پشتیبان(Backup) یا فایل ثبت تراکنشها (Transactions Log Files – T.L.F) و یا فایل رویدادنگاری (Event Journaling) استفاده می‌شود . هرچند برای این منظورها هم بهتر است که از دیسک استفاده شود اما برای کاهش هزینه‌های سازمان در محیط پایگاه داده‌های بزرگ از نوار هم استفاده می‌شود.
- **سخت افزار پردازنده مرکزی** : منظور همان کامپیوتر است. می‌دانیم که برای ایجاد پایگاه داده‌ها می‌توان از کامپیوترهای معمولی استفاده کرد اما برای ایجاد پایگاه داده‌های خیلی بزرگ و بویژه توزیع شده از نوع خاصی از کامپیوترها که معماری خاص و قابلیتهای ویژه‌ای دارند، استفاده می‌گردد.
- **سخت‌افزار ارتباطی(همرسانش)** : منظور سخت‌افزارهای مورد نیاز جهت ارتباط بین کامپیوتر و دستگاههای جانبی و نیز سایر کامپیوترها می‌باشد.

۳-نرم‌افزار

بین بانک اطلاعاتی فیزیکی(یعنی داده‌ها به آن شکلی که ذخیره می‌شوند) و کاربران سیستم، لایه‌ای از نرم‌افزار، مدیر بانک اطلاعاتی(DB) و یا سیستم مدیریت بانک اطلاعاتی (DBMS) وجود دارد. تمامی درخواستها از کاربران برای دسترسی به بانک اطلاعاتی بوسیله سیستم مدیریت بانک اطلاعاتی(DBMS) پاسخ داده می‌شود. قابلیتهایی که برای اضافه کردن و حذف فایل‌ها (یا جداول)، بازیابی یا به‌هنگام‌سازی داده‌ها در چنین فایل‌ها یا جدولهایی وجود دارند، همگی قابلیتهای ارائه شده توسط DBMS هستند.

نکته : DBMS مهترین جزء نرم‌افزاری در کل سیستم می‌باشد ولی تنها نرم‌افزار موجود نمی‌باشد، دیگر نرم‌افزارهای شامل نرم‌افزارهای سودمند، ابزارهای کمکی جهت طراحی، تولیدکننده‌های گزارش و غیره می‌باشد.

۴- کاربران

در یک بانک اطلاعاتی سه دسته کاربران وجود دارند که در زیر به شرح آنها می‌پردازیم :

۱- **برنامه‌نویسان کاربردی(DataBase Programming – DBP)** : که مسئول نوشتن برنامه‌های کاربردی موردنیاز در بانک اطلاعاتی می‌باشند. این برنامه‌ها شامل اعمالی همچون بازیابی، افزودن، حذف و به‌روز رسانی اطلاعات می‌باشند که همه این وظایف توسط درخواست مناسبی از DBMS انجام می‌شود. در واقع می‌توان گفت که این برنامه‌ها وظیفه اصلی‌شان پشتیبانی از کاربران نهایی است.

۲- **کاربران نهایی(End Users)** : دسته دوم کاربران نهایی هستند که از طریق یک ترمینال با سیستم در ارتباط هستند.

۳- **اداره کننده بانک اطلاعاتی(DataBase Administrator – DBA)** : تعریف و شرح وظایف آن بعداً بطور کامل شرح داده خواهد شد.

روشهای ایجاد سیستم‌های ذخیره و بازیابی اطلاعات

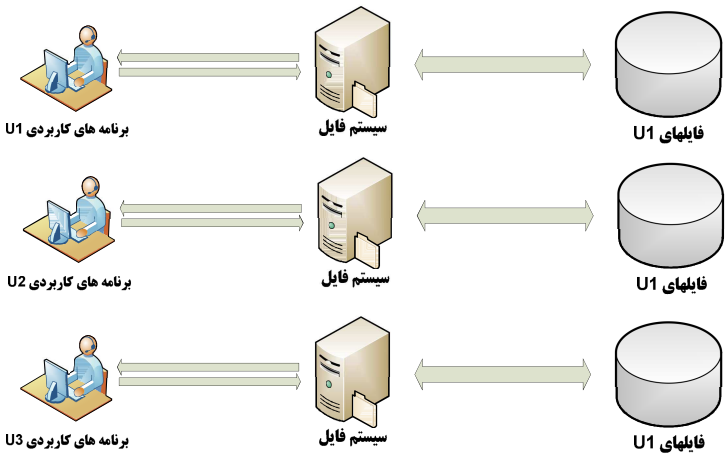
فرض کنید بخواهیم برای یک محیط عملیاتی دانشگاهی بخشهای امور آموزش، امور دانشجویی و ... را بصورت مکانیزه بوجود آوریم.

برای ایجاد یک سیتم ذخیره و بازیابی اطلاعات دو روش کلی وجود دارد

۱- روش کلاسیک (فایلینگ – ناپایگاهی - غیربانکی)

۲- روش بانکی (پایگاهی)

۱- **روش فایلینگ (غیربانکی)** : در این روش با بهرگیری از امکانات سیستم‌فایل که در اختیار سیستم عامل می‌باشد و یک زبان برنامه‌نویسی سطح بالا و با نگرش به ورودیها و خروجیهای مورد نظر، فایل یا فایلهائی تعریف می‌گردند و ایجاد می‌شوند. در هر فایل نیز رکورد یا گونه‌هائی از رکوردهای موردنظر تعریف می‌شوند و در هر رکورد نیز فیلدهای موردنظر تعریف می‌شوند. پس از تعریف فایلها عملیات پردازشی فایلها برروی آنها انجام می‌گیرد.



در روش غیربانکی بسیاری از داده‌های مورد نیاز یک کاربر همان داده‌های موردنیاز کاربر یا کاربران دیگر است ، لیکن هر کاربر فایل یا فایلهای خاص خود را در اختیار دارد. در نتیجه به علت عدم تجمع داده‌های ذخیره شده و عدم وجود وحدت در ذخیره‌سازی با پدیده افزونگی(Redundancy) داده‌ها وجود دارد. از سوی دیگر چون برنامه‌های کاربردی برای ایجاد و پردازش فایلهای خاص نوشته شده‌اند، هرگونه تغییری در ساختار رکوردها و فایلها و روش دستیابی به آنها سبب می‌گردد که برنامه‌های متناسب با آنها نیز تغییر یابند. به بیانی دیگر در یک محیط غیربانکی برنامه‌های کاربردی به خصوصیات فیزیکی ذخیره‌سازی وابسته هستند.

برخی از ویژگیهای این روش عبارتست از:

الف – فایلهای هر قسمت ، ساختار ، فرمت و استراتژی دستیابی و تا حدی تدابیر امنیتی خاص خود را دارند.

ب – برنامه‌سازی معمولاً به کمک یک(یا بیش از یک)زبان برنامه‌نویسی سطح بالا و گاه نسل چهارم انجام می‌شود.

ج – پیکربندی سخت‌افزاری و نرم‌افزاری در صورت استفاده از کامپیوترهای کوچک، معمولاً جدا هستند(گاه از طریق ایجاد شبکه‌های محلی کوچک، بعضی از منابع سیستمی به طور اشتراکی ممکن است استفاده بشود)

د – فایل‌های هر قسمت معمولاً قابل استفاده توسط قسمت‌های دیگر نیستند(به ویژه در صورت عدم وجود شبکه و عدم وجود نرم‌افزار مدیر داده‌ها)

معایب این روش عبارتست از :

الف- عدم وجود محیط ذخیره سازی اطلاعات و عدم وجود یک سیستم یکپارچه

ب- عدم وجود سیستم کنترل متمرکز روی کل داده‌های سازمان

ج- وجود تکرار در ذخیره‌سازی داده‌ها (افزونگی)

د- عدم وجود ضوابط ایمنی کارا و مطمئن

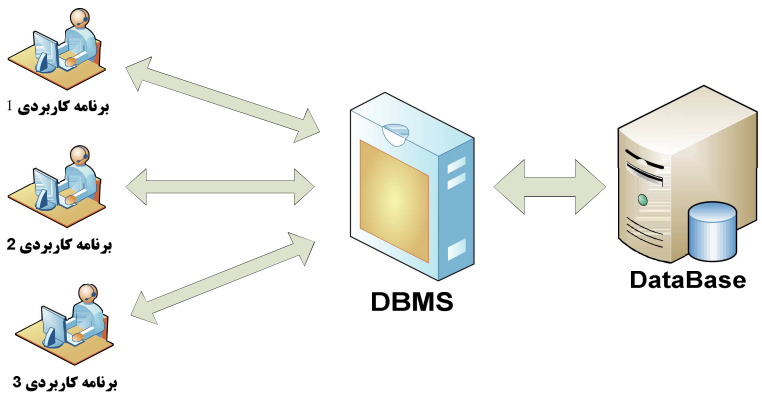
ه- خطر بروز پدیده نامطلوب ناسازگاری داده‌ها

ن- عدم امکان اشتراکی شدن داده‌ها و یا اشتراک در حد ضعیف

و- وابسته بودن برنامه‌های کاربردی به محیط ذخیره‌سازی داده‌ها

۲- **روش بانکی :** در این روش تمام داده‌ها را بصورت مجتمع ذخیره می‌کنیم. ولی هر کاربر با توجه به نیاز و نوع اطلاعاتی که می‌خواهد، می‌تواند به داده‌های موردنظر خود دسترسی داشته باشد. در اینجا وحدت ذخیره‌سازی داریم ولی هر کاربر ایده خاص خود را نسبت به داده‌ها دارد.

در واقع در اینجا داده‌ها به صورت یک DataBase ذخیره‌سازی می‌شود و یک نرم‌افزار مدیریتی DBMS در واقع رابط بین کاربران و داده‌ها می‌باشد که با این روش می‌توان امنیت را نیز بالا برد.



در روش بانکی نرم‌افزار سیستم مدیریت پایگاه داده‌ها(DBMS) نقش رابط بین برنامه‌های کاربردی کاربر(یا کاربران) و پایگاه داده‌ها یا بانک اطلاعاتی را بر عهده دارد. در این روش کنترل متمرکز وجود دارد در حالی که در راهبرد غیربانکی کنترل متمرکز وجود خارجی ندارد و از سوی دیگر در این روش داده‌ها بصورت مشترک در اختیار تمام کاربران می‌باشند. و در نتیجه داده‌های ذخیره‌شده به صورت مجتمع می-

باشند و در ذخیره‌سازی وحدت عمل وجود دارد. به بیانی دیگر محیط فیزیکی ذخیره‌سازی بصورت مجتمع، مشترک و واحد می‌باشد.

با طراحی پایگاه داده‌ها بصورت یکپارچه موجب می‌شود که افزونگی داده‌ها به حداقل ممکن تنزل پیدا کند.

مزایای و معایب سیستم‌های پایگاه داده‌ای

برخی از مهمترین مزایای سیستم‌های بانک‌های اطلاعاتی بشرح زیر می‌باشند:

۱) کاهش افزونگی داده‌ها :

درصورت استفاده از سیستمهای بدون پایگاه داده‌ای، بدلیل آنکه هر برنامه کاربردی دارای فایلهای خاص خودش می‌باشد، تکرار اطلاعات در برنامه کاربردی سبب افزونگی شده و موجب هدر رفتن فضای منبع ذخیره‌سازی می‌گردد درحالیکه با استفاده از سیستمهای پایگاه داده‌ای، تکرار داده‌ها و افزایش آن به حداقل خواهد رسید.

۲) اجتناب(پرهیز) از ناسازگاری داده‌ها:

در صورت عدم کنترل افزونگی داده‌ها، حالاتی وجود خواهد داشت که دو ورودی سازگار نیستند(متناقص یا ناسازگار) که در این حالت پایگاه داده‌ها قادر به ارائه اطلاعات صحیح به کاربران نمی‌باشد.مثلاً حالتی را در نظر بگیرید که دو فایل که اطلاعات مشابهی دارند، یکی از آنها را به روز رسانی گردد ولب دیگری بدون تغییر بماند که در این حالت گفته می‌شود که بانک اطلاعاتی ناسازگار است.

بدیهی است که با کنترل و کاهش افزونگی داده‌ها، سیستم مدیریت پایگاه داده‌ها(DBMS) می‌تواند سازگاری و یکپارچگی داده‌ها را تضمین کند.(مسلماً این امر با اطمینان از اینکه هر تغییری بر روی یکی از دو داده، بطور خودکار بر روی دیگری نیز اعمال خواهد شد، صورت می‌گیرد)

۳) اشتراک داده‌ها :

اشتراک داده‌ها بدین معنی است که نه تنها برنامه‌های کاربردی موجود بتوانند داده‌های خود را بصورت اشتراکی در اختیار سایر برنامه‌های قرار دهد ، بلکه برنامه‌های کاربردی جدید نیز بعد از ایجاد بتوانند از همان داده‌های ذخیره شده استفاده کنند.

۴) اعمال استانداردها :

در سیستمهای پایگاه داده‌ای با استاندارد کردن قالب داده‌های داده‌های ذخیره شده بخصوص بهنگام تبادل داده‌ها بین سیستمها، موجبات به اشتراک گذاری داده‌ها برای قابل درک بودن فراهم می‌شود.

۵) اعمال محدودیتهای امنیتی

در هر سیستم بانک اطلاعاتی قوانین و کنترل‌های مختلفی می‌تواند برای هر نوع دسترسی(بازیابی، اصلاح، حذف، به‌روز رسانی) برای هر بخش اطلاعاتی در بانک اطلاعاتی در نظر گرفته شود.

۶) اعمال جامعیت یا یکپارچگی داده‌ها

مسئله یکپارچگی یا جامعیت داده‌ها، مسئله اطمینان از وجود داده‌های صحیح در بانک اطلاعاتی می‌باشد. ناسازگاری بین دو داده که نشان دهنده یک واقعیت هستند، مثالی از عدم یکپارچگی و جامعیت داده-هاست. البته این موضوع فقط در هنگامی که افزونگی داده‌ها وجود دارد، پیش می‌آید. در واقع برای جلوگیری از این امر می‌توان با اعمال قوانینی که توسط مدیر داده‌ها(DA) بنام قوانین جامعیت داده‌ها تعریف می‌شود، از بروز چنین مشکلاتی جلوگیری کرد.

البته مواردی دیگر همچون تأمین استقلال داده‌ها، تسریع در دریافت پاسخ پرس و جوها، در دسترس بودن داده‌ها و سیستم و ... نیز را می‌توان از مزایای سیستمهای پایگاه داده‌ای بیان کرد.

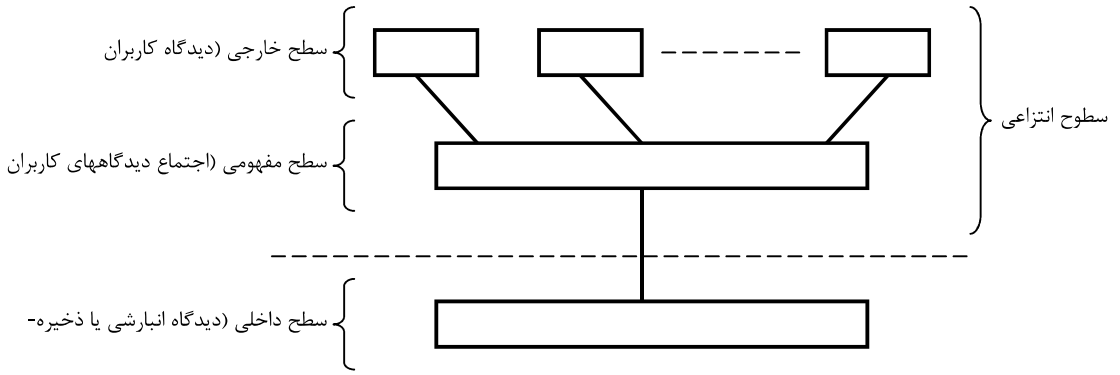
از معایب سیستمهای پایگاه داده‌ای می‌توان به موارد زیر اشاره کرد

۱) به مخاطره افتادن امنیت داده‌ها بدون کنترل‌های مناسب چرا که داده‌ها متمرکز بوده و این تمرکز آنها را آسیب‌پذیر می‌سازد.
۲) به مخاطره افتادن جامعیت داده‌ها بدون اعمال کنترل‌های لازم
۳) نیاز به سخت افزارهای اضافی
۴) نیاز به هزینه و بودجه زیاد برای اجرای DBMS
و ...

معماری پایگاه داده‌ها :

بانک اطلاعاتی نرم‌افزار پیچیده و قدرتمندی برای مدیریت اطلاعات می‌باشد. این نرم‌افزار رابط بین داده‌های ذخیره شده و کاربران می‌باشد. نرم افزارهای پیچیده معمولاً بصورت لایه‌ای طراحی می‌شوند به این ترتیب که برای نرم‌افزار چندین لایه درنظر گرفته می‌شود و هر لایه انجام بخشی از عملیات سیستم را برعهده می‌گیرد. برای بانکهای اطلاعاتی نیز یک معماری لایه‌ای در نظرگرفته شده است. در این بخش ما یک معماری برای سیستم بانک اطلاعاتی معرفی می‌کنیم. طرح چنین چهارچوبی برای توضیح مفاهیم کلی بانک‌های اطلاعاتی و همچنین توضیح اجزای مختلف سیستم بانک اطلاعاتی دارای اهمیت زیادی می‌باشد. اما نمی‌توان ادعا نمود که هر سیستم بانک اطلاعاتی در این چهارچوب خلاصه می‌شود و یا اینکه این چهارچوب تنها چهارچوب ممکن می‌باشد.

گروه مطالعاتی ANSI/SPARC مدل پیشنهادی بشرح زیر را برای پایگاه داده‌ها پیشنهاد کرده است:



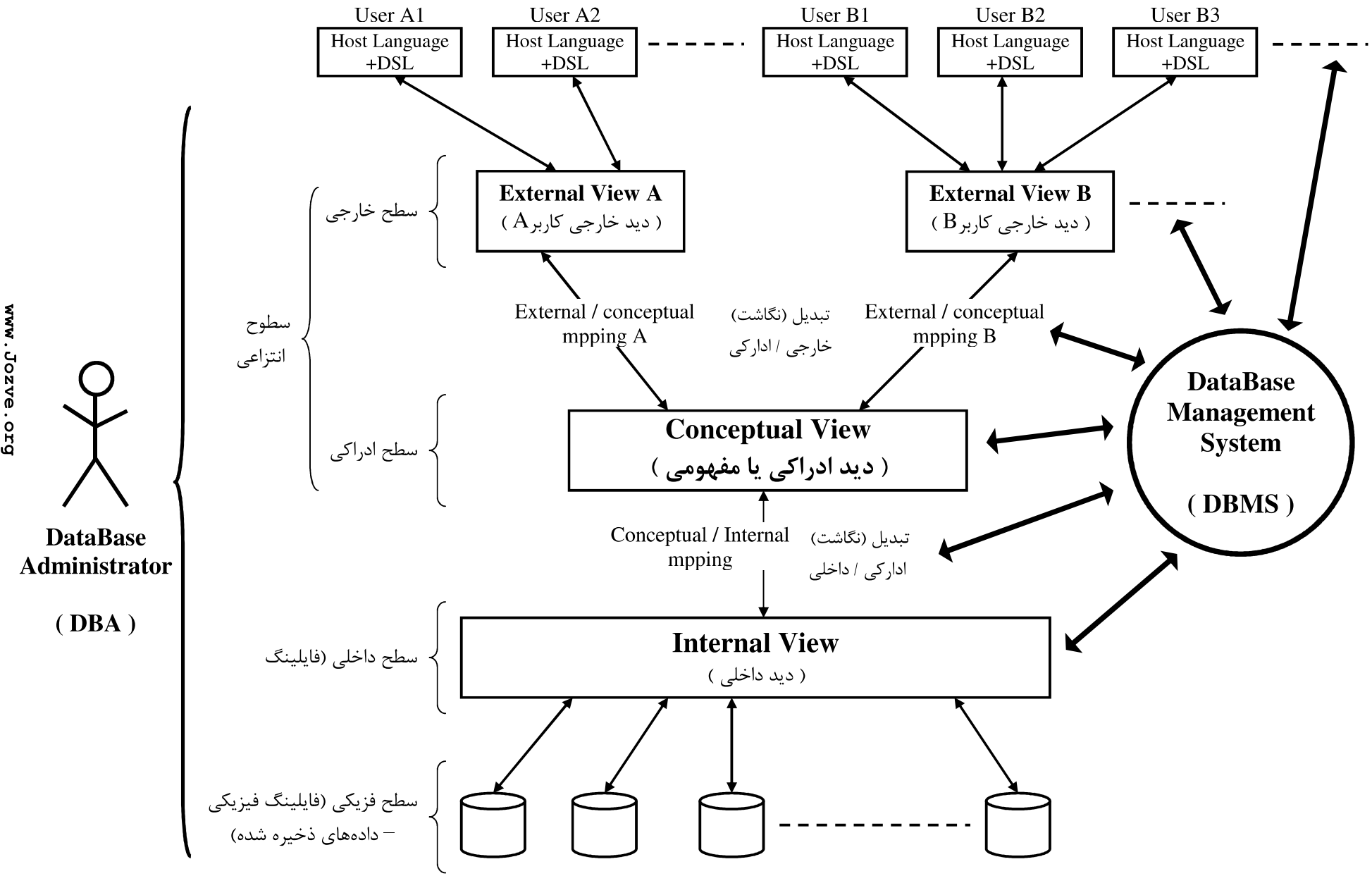
که در زیر به شرح مختصری از سطوح ارائه در طرح فوق خواهیم پرداخت

سطح داخلی : نزدیکترین سطح به حافظه اصلی یا انباره فیزیکی می‌باشد. منظور از سطح داخلی، سطحی است که ذخیره سازی عملی داده‌ها را انجام می‌دهد.

سطح خارجی : نزدیکترین سطح به استفاده کنندگان می‌باشد یعنی داده‌ها توسط استفاده کنندگان مختلف مشاهده می‌شود.لایه خارجی نزدیکترین سطح به استفاده کنندگان می‌باشد یعنی سطحی که داده توسط کاربران مختلف مشاهده می‌شود. کاربران بانک اطلاعاتی لزوماً به همه اطلاعات دسترسی ندارند و هر یک با داده‌های مربوط به خود سر و کار دارند. آن قسمت از بانک اطلاعاتی که توسط یک کاربر قابل مشاهده است، دیدگاه کاربر نسبت به بانک اطلاعاتی نامیده می‌شود.

باید توجه داشت که سه سطح، اساساً سه سطح "تعریف داده‌ها" هستند: دو سطح در محیط انتزاعی و سطح سوم در محیط فایلینگ منطقی.

حال به نمای کاملتری معماری پیشنهادی ANSI را نشان داده و به شرح هریک از سطوح معماری می‌پردازیم.



در این معماری علاوه بر سه سطح، اجزای دیگری هم دیده می‌شود که در واقع جزو "سیستم پایگاه داده‌ها" می‌باشند. بطور کلی معماری بانک اطلاعاتی از اجزای زیر تشکیل شده است :

۱- دید خارجی (External View)
۲- دید داخلی (Internal View)
۳- دید ادراکی یا مفهومی (Conceptual View)
۴- تبدیلات بین سطوح (Mapping)
۵- زبان میزبان (Host Language – HL)
۶- زبان فرعی داده‌ای (Data Sub Language – DSL)

۱- سطح خارجی (External View)

دید کاربران از داده‌های ذخیره شده در بانک اطلاعاتی می‌باشد. در واقع کاربران هر کدام می‌توانند دیدهای مختلفی نسبت به بانک‌های اطلاعاتی واحد داشته باشد.

- این دید جزئی است و نا جامع، بدین مفهوم که نشان دهنده محدوده‌ای از پایگاه داده‌ها است که به نیازهای اطلاعاتی یک کاربر خاص پاسخ می‌دهد.
- این دید در سطح انتزاعی مطرح می‌شود: بنابراین مبتنی است بر یک ساختار داده مشخص و معمولاً همان ساختار داده‌ای که دید ادراکی براساس آن طراحی و تعریف می‌شود.
- این دید بر روی دید ادراکی طراحی و تعریف می‌شود (به همین دلیل به جدولهای سطح ادارکی، جدولهای مبنا یا پایه می‌گویند)
- لایه خارجی تنها لایه‌ای است که کاربران با آن سر و کار دارند. لایه‌های دیگر به مدیر و برنامه‌سازان بانک مربوط می‌شود. این لایه برای حفظ امنیت می‌گوید : "به هر کس به همان اندازه اطلاعات بده که نیاز دارد و می‌تواند به آنها دسترسی داشته باشد و نه بیشتر".

هر مدال خارجی توسط شِمای خارجی(External schema) تعریف می‌شود که اساساً شامل شرحهائی از هر یک از انواع مختلف رکورد در آن مدل خارجی می‌باشد. به عنوان مثال نوع رکورد خارجی کارمند ممکن است با فیلد شماره کارمندی ۷ رقمی و فیلد نام ۱۴ رقمی و فیلد نام خانوادگی ۲۱ رقمی تعریف شود.

به عنوان مثال اگر جدول زیر را در نظر داشته باشیم:

ID	Name	Family	Address	Tel	Avg	Term	Year	…..

هر کاربر می‌تواند دید خاص خود را نسبت به جدول بالا داشته باشد.

ID	Name	Avg

دید کاربر A

ID	Avg	Term	Year

دید کاربر B

دقت داشته باشید که هرچند دید خارجی همانند دید ادراکی نیز یک جدول است ولیکن یک جدول مجازی می‌باشد.

۲- سطح داخلی (Internal View)

دیدگاه داخلی یک نمایش سطح پایین از تمام بانک‌های اطلاعاتی می‌باشد. در این سطح مواردی همچون انواع مختلف رکوردهای ذخیره شده، اندیسها، چگونگی نمایش رکوردهای ذخیره شده و اینکه دارای چه ترتیب فیزیکی هستند و همچنین مقدار فضایی که برای ذخیره‌سازی داده استفاده می‌شود، تشریح می‌گردد.

- این دید در سطح فایلینگ منطقی پایگاه داده‌ها مطرح است
- سطح داخلی پایگاه داده‌ها در واقع همان سطحی است که در آن فایلینگ منطقی پایگاه داده‌ها تعریف می‌شود.
- سطح داخلی نزدیکترین سطح به رسانه ذخیره‌سازی فیزیکی است

۳- سطح ادراکی (Conceptual View)

دید طراح بانک اطلاعاتی است از داده‌های ذخیره شده در بانک که دید طراح دیدی است جامع دیدهای کاربران و در عین حال متفاوت با هر یک از دیدها.

- سطح مفهومی یا ادراکی عبارتست از نمایش تمام اطلاعات موجود در یک بانک اطلاعاتی(در قالب یک دیدگاه خارجی) مجزا و مجرد از روشی که داده‌ها بطور فیزیکی ذخیره می‌شوند
- این دید، دید طراح پایگاه داده‌ها است نسبت به داده‌های ذخیره شدنی در پایگاه

مدل مفهومی توسط شِمای مفهومی تعریف می‌شود که در برگیرنده تعریف‌هایی از هر یک از انواع مختلف رکوردهای مفهومی می‌باشد.

۴- نگاشت یا تبدیلات بین سطوح (mapping)

دو سطح نگاشت در معماری بانک اطلاعاتی قابل ملاحظه می‌باشد :

۱) **نگاشت خارجی / ادراکی (External / Conceptual mapping)**

ارتباط یا تناظر بین یک دیدگاه خارجی و دیدگاه ادراکی را تعریف می‌کند.

برای مثال فیلدها می‌توانند بین مدل خارجی و مفهومی دارای انواع داده‌های مختلفی باشند و یا اسامی فیلدها ، رکوردها تغییر کند و یا چندین فیلد ادارکی می‌توانند در یک فیلد خارجی ترکیب شده باشند.

۲) **نگاشت ادارکی / داخلی (Conceptual / Internal mapping)**

این نگاشت ارتباط بین دیدگاه ادارکی و بانک اطلاعاتی ذخیره شده را تعریف می‌کند. این نگاشت مشخص می‌کند که چگونه رکوردها و فیلدها در سطح داخلی ارائه می‌گردند. اگر ساختار بانک اطلاعاتی ذخیره شده تغییر نماید (یعنی اگر تغییری در تعریف ساختار انبارشی بوجود آید)، آنگاه نگاشت ادارکی/داخلی طبعاً باید به گونه‌ای تغییر پیدا کند که شمای ادارکی بدون تغییر باقی بماند (البته کنترل چنین تغییراتی به عهده DBA می‌باشد). به عبارت دیگر، تأثیر چنین تغییراتی می‌بایست به دور و پایین‌تر از سطح ادارکی بوده بطوری که استقلال داده‌ها حفظ گردد.

۵- زبان میزبان (Host Language – HL)

یکی از زبانهای متعرف برنامه‌سازی است.بطور معمول با یک زبان پایگاه داده‌ای نمی‌توان به معنای متعارف برنامه-سازی کرد، بلکه این زبانها به صورت زبان میهمان همراه با یک زبان میزبان(همانند SQL) استفاده می‌شوند.

۶- زبان فرعی داده‌ها (Data Sub Language – DSL)

زبانی فرعی داده‌ها زبانی نظیر SQL می‌باشد که یک زبان پرس و جوی ساخت یافته است.

زبانهای فرعی داده‌ها از نظر نیاز به زبان میزبان و یا عدم نیاز به آن، به دو دسته تقسیم می‌شوند:

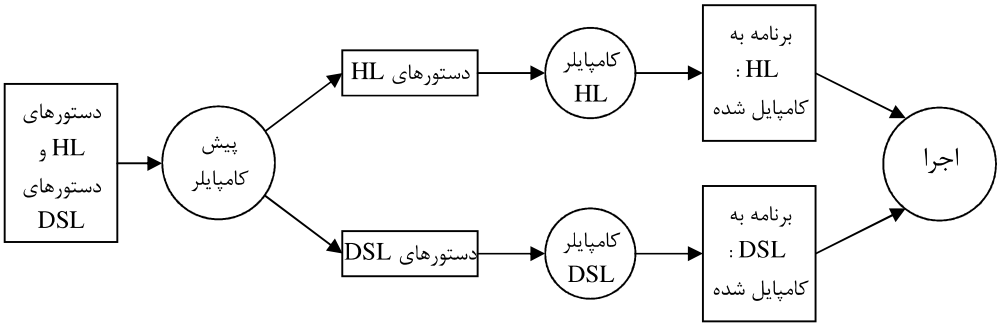
- مستقل یا خودکفا (Independent)
- ادغام شدنی یا غیرمستقل (Embedded)

اگر زبان فرعی داده‌ها غیرمستقل باشد دستورات آن در دستورات زبان میزبان ادغام می‌شوند. ولی اگر مستقل باشد نیازی به زبان میزبان ندارد. یا به صورت دیگر می‌توان گفت که

زبان مستقل زبانی است که نیاز به زبان میزبان ندارد و خود به صورت تعاملی (اندرکنشی) استفاده می‌شود و آن را I.DSL (Interactive DSL) گوییم. این گونه زبان، در واقع یک زبان پرس و جوی است یعنی با دستوراتش می‌توان انواع پرسشها را تنظیم کرد (برای انجام چهار عمل اصلی یعنی بازیابی، درج، حذف و بهنگام‌سازی).

زبان ادغام شدنی، زبانی است که دستورهایش درمتن برنامه‌ای به زبان میزبان بکار می‌روند و مستقلاً قابل استفاده نیستند و آنرا E.DSL (Embedded DSL) گوییم.البته زبان فرعی ممکن است هم مستقل و هم ادغام شدنی باشد (I/E.DSL). توجه داشته باشیم که در حالت E.DSL لزوما زبان میزبان نباید یک زبان برنامه‌سازی متعارف باشد. در حال حاضر می‌توان از زبانهای جدیدتر مثلاً جاوا و جاوا-اسکریپت نیز استفاده کرد.مثلاً می‌توان با استفاده از ASP که یک امکان تولید برنامه در محیط اینترنت است، درمحیط جاوا-اسکریپت دستورهای SQL را به صورت یک رشته داد و عملیات موردنظر را در پایگاه داده‌ای رابطه‌ای انجام داد یا از امکانات دیگر از جمله ADO (Active Data Object) استفاده نمود.

چگونگی ادغام می‌تواند صریح یا ضمنی باشد. در حالت ادغام صریح، عین دستور زبان داده‌ای فرعی در برنامه به زبان میزبان نوشته می‌شود، ولی در حالت ادغام ضمنی، دستورهای زبان داده‌ای فرعی به صورت توابع، فراخونده می‌شوند.در حالت ادغام صریح، محیط برنامه‌سازی "دو زبانی" است و برای کامپایل کردن آن به دو کامپایلر نیاز است: پیش کامپایلر برای دستورات E.DSL و کامپایلر برای زبان میزبان. در شکل زیر روند کلی مرحله کامپایل دو زبانی را مشاهده می‌کنید :

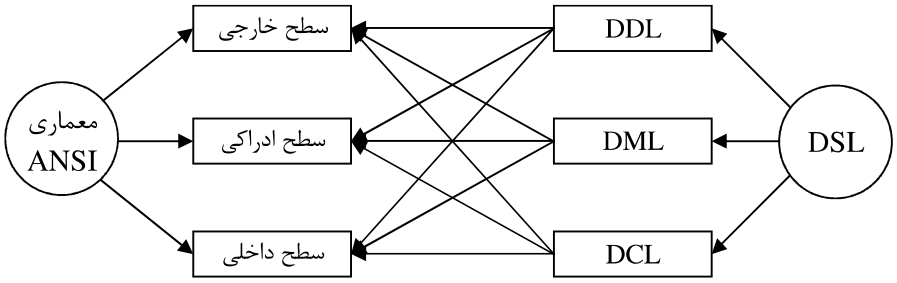


توجه داشته باشید که سیستم پس از جداسازی دو برنامه، پیوند بین آنها را به نحوی برپا می‌دارد تا در مرحله اجرا مشکلی در اجرای برنامه‌ها بروز نکند.

دستورهای زبان فرعی داده‌ها برای تعریف یا تشریح داده‌ها و دستکاری و کنترل داده‌ها به کار برده می‌شوند. هر سیستم پایگاه داده‌ها زیر نظر یک سیستم مدیریت پایگاه داده‌ای خاصی ایجاد و اداره می‌شود و زبان فرعی داده‌ها مختص خود را دارا می‌باشد و برای هر سطح از معماری پایگاه داده‌ها دستورات مشخصی دارد. با توجه به مطالب گفته شده دستورات زبان فرعی داده‌ها(DSL) را می‌توان به سه دسته زیر تقسیم بندی کرد:

- **دستورات تعریف داده‌ها یا DLL (Data Definition Language)**
- **دستورات کار با داده‌ها یا DML (Data Mainpulation Language)**
- **دستورات کنترلی داده‌ها یا DCL (Data Control Language)**

از نظر ANSI، این سه دسته دستور باید برای هریک از سه سطح معماری پایگاه داده‌ها وجود داشته باشند.



ویژگیهای زبان داده‌ای فرعی (DSL)

- ✓ تعداد دستورهایش باید کم باشد (به ویژه DDL و DML)
- ✓ یادگیری و استفاده از آن باید آسان باشد.
- ✓ در طراحی آن اصل وحدت دستور باید رعایت شده باشد، یعنی برای عمل منطقاً واحد، دستور واحد (و نه چند دستور) داشته باشد.
- ✓ بهتر است نارویه‌ای(Non Procedural) باشد و نه رویه‌ای (Procedural)

زبان رویه‌ای(Procedural) : زبانی است که در برنامه‌سازی با آن، برنامه‌ساز خود، رویه(روش) انجام عمل مورد نظرش را بنویسد.به عنوان مثال در بازیابی نه تنها باید مشخص کند که بدنبال چه داده‌ای می‌باشد بلکه باید رویه بازیابی آنرا نیز مشخص کند.

زبان نارویه‌ای(Non Procedural) : برخلاف زبان رویه‌ای، برنامه‌ساز در سطح بالاتر برنامه می‌نویسد. در واقع برنامه‌ساز تنها می‌گوید که چه داده‌ای را می‌خواهد ولی رویه انجام عمل مورد نظرش را بیان نمی‌کند.

سیستم مدیریت پایگاه داده‌ها (DBMS)

سیستم مدیریت پایگاه داده‌ها یکی از انواع نرم‌افزارهای واسط بین محیط فیزیکی ذخیره و بازیابی اطلاعات و محیط منطقی برنامه‌سازی است.

سیستم مدیریت پایگاه داده‌ها (DBMS) نرم‌افزاری است که تمام دستیابی‌ها به بانک اطلاعاتی را ممکن و انجام می‌دهد و در واقع به نوعی رابط بین کاربر و بانک اطلاعاتی می‌باشد. این نرم‌افزار به کاربر برنامه‌ساز امکان می‌دهد تا :

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

این بدان مفهوم است که علاوه بر تامین توانایی جستجوی عادی بازیابی، سیستم نیز عهده‌دار کارهای کنترل دستیابی با حفظ صحت و تمامیت می‌باشد.

یک سیستم مدیریت پایگاه داده‌ها دارای سه مولفه زیر است:

- **پایگاه داده‌ها (DataBase)** : شامل فایلهای متنوع که هرکدام از رکوردهای منفرد تشکیل می‌گردند و در برگیرنده دستیابی و عملیاتی که بر روی فایل‌ها انجام می‌گیرند، می‌باشد.
- **سیستم ارتباطات (Communication System)** : که تعامل بین کاربران سیستم و سیستم خودکار را تأمین می‌نماید و همچنین وظایف کنترل پیام، ویرایش و نمایش خروجی را بر عهده دارد.
- **سیستم مدیریت تراکنش (Transaction)** : که عهده‌دار برنامه‌ریزی کارهای دریافتی از کاربران مختلف، کنترل دستیابی به فایلها، کنترل عملیات همزمان جهت کارهایی که همزمان باهم اجرا می‌شوند و پیاده‌سازی رویه‌های ترمیم به دنبال هر خرابی سیستم می‌باشد.

آنچه که از نقطه نظر مفهومی روی می‌دهد بشرح زیر است

۱- یک کاربر با بهره‌گیری از زبان داده‌ای خاصی درخواست دستیابی می‌نماید
۲- سیستم مدیریت پایگاه داده(DBMS) درخواست را دریافت و آنرا تحلیل می‌کند
۳- سیستم مدیریت پایگاه داده(DBMS) به نوبه خود شِمای خارجی، تبدیل خارجی/مفهومی، شِمای مفهومی، تبدیل مفهومی/داخلی و تعریف ساختاری ذخیره‌سازی را مورد بررسی قرار می‌دهد.
۴- سیستم مدیریت پایگاه داده(DBMS) عملیات مورد نیاز را بر روی پایگاه داده ذخیره شده انجام می‌دهد.

به عنوان مثال آنچه را که در بازیابی رکورد خارجی نهفته است را مدنظر قرار دهید. بطور کلی فیلدهائی از چند رکورد مفهومی مختلف مورد نیاز می‌باشد. هر رکورد مفهومی به نوبه خود نیاز به فیلدهائی از چند رکورد ذخیره شده دارد. در نتیجه سیستم مدیریت پایگاه داده‌ها تمام رکوردهای ذخیره شده مورد نیاز را بازیابی می‌نماید تا رکوردهای مفهومی مورد نیاز را ساخته و سپس رکوردهای خارجی مورد نیاز را ایجاد می‌نماید. در هر مرحله‌ای نوع داده‌ها یا سایر تبدیلات لازم هستند.

توجه داریم که درخواست کاربر ممکن است بازیابی، درج، حذف یا بهنگام‌سازی باشد. درصورتیکه درخواست کاربر بازیابی باشد، سیستم باید داده موردنظر کاربر را بسازد و در اختیارش قرار دهد و به بیان دیگر داده‌ها را به عینیت درآورد. "**ساختن داده**" مورد نظر کاربر ممکن است مستقیم یا غیرمستیم باشد

- **ساختن مستقیم (Direct Materialization)** : وقتی است که داده موردنظر در سطوح زیرین معماری و در نهایت در پایگاه داده‌های فیزیکی وجود داشته باشد و سیستم تنها آن‌را بازیابی می‌کند(البته جنبه‌های

داده در سطوح مختلف می‌تواند متفاوت باشد، مثلاً نام داده در سطح خارجی لزوماً همان نام در سطح ادراکی و یا در سطح داخلی نیست)

- **ساختن غیرمستقیم (Indirect Materialization)** : وقتی است که داده موردنظر کاربر، در سطوح زیرین معماری و نهایتاً در پایگاه داده فیزیکی وجود نداشته (ذخیره نشده) و سیستم باید آن‌را با پردازش بر روی تعدادی فقره داده بدست آورد.

وظایف سیستم مدیریت پایگاه داده DBMS (DataBase Management System)

- **تعریف داده‌ها :** DBMS باید قادر به تعریف داده‌ها به یکی از زبانهای تعریف داده‌ای(DDL) باشد. در واقع سیستم مدیریت اطلاعاتی باید قادر به قبول تعاریف داده‌ها از شِمای خارجی، شِمای داخلی و تمامی نگاشت‌های متناظر به شکل منبع(source) بوده و آنها را به شکل مقصد(Object) مناسب تبدیل کند. DBMS همچنین باید تعاریف DDL را درک کند به گونه‌ای که به عنوان مثال، DBMS این مسئله را که رکوردهای خارجی employee دارای فیلدی به نام salary است، درک کند.سپس آن باید قادر باشد تا این اطلاع را در تفسیر و پاسخ به درخواست‌های کاربران مورد استفاده قرار دهد.(مثلاً درخواستی برای تمام کارمندان که دارای حقوقی کمتراز 50,000 دلار دارند)
- **دستکاری داده‌ها :** DBMS باید قادر باشد که درخواستهایی از کاربران را برای بازیابی، به‌روز رسانی، اضافه کردن داده‌های جدید یا حذف داده‌های موجود در بانک اطلاعاتی را انجام دهد. بطور کلی درخواستهای DML از DBMS می‌تواند به دو صورت "برنامه‌ریزی شده" و یا "برنامه‌ریزی نشده" باشد.یک درخواست برنامه‌ریزی شده درخواستی است که نیاز مورد نظر به درستی از قبل از پیش بینی شده باشد (منظور قبل از اجرای واقعی آن است). احتمالاً DBA طراحی بانک اطلاعاتی فیزیکی را بگونه‌ای تغییر خواهد داد که روشی برای ضمانت اجرای مناسب چنین درخواستهایی وجود داشته و کارایی درخواستهای برنامه‌ریزی شده بالا رود. بر عکس درخواستهای غیربرنامه‌ریزی یا برنامه‌ریزی نشده در واقع یک پرس و جوی موردی می‌باشد یعنی درخواستی که از قبل نیاز آن مشخص نمی‌باشد بلکه در صورت نیاز و به صورت موردی مطرح می‌شود.
- **جامعیت و امنیت داده‌ها :** سیستم مدیریت بانک اطلاعاتی (DBMS) باید بر درخواستهای کاربران نظارت داشته و از هر گونه تلاشی که قوانین جامعیت و امنیت تعریف شده توسط DBA را بر هم می‌زند، جلوگیری نماید.
- **ترمیم و سازگاری داده‌ها :** DBMS باید کنترل‌های خاصی برای امکان همزمانی و ترمیم فراهم سازند.
- **فرهنگ داده‌ها (Data Dictionary – متا داده‌ها) :** DBMS باید امکان فرهنگ داده‌ها را فراهم سازد. مهمترین ابزار مدیر پایگاه داده‌ها فرهنگ داده‌ها می‌باشد. در حقیقت فرهنگ داده‌ها در جایگاه خود، پایگاه داده‌ای است که حاوی "داده‌های دربارهٔ داده‌ها" می‌باشدکه گاهی اوقات بنام "ابر داده‌ها" یا Meta Data نیز معرفی می‌گردد.بدین مفهوم که آن حاوی تعریف دیگر شی‌ها در سیستم به جای "داده‌ها خام" می‌باشد. یک فرهنگ داده‌ها شامل کلیه طرحها(شِماها)، نگاشت‌های مختلف (خارجی، ادراکی، داخلی به یکدیگر) می‌باشد. همچنین شامل اطلاعاتی دیگری نظیر اینکه
  - چه برنامه‌هایی از چه بخش‌هایی از بانک اطلاعاتی استفاده می‌کنند
  - مشخصات کاربران و حقوق دستیابی آنها به داده‌های ذخیره شده
  - چه استفاده‌کنندگانی به چه گزارشاتی احتیاج دارند

- مشخصات پایانه‌های متصل به سیستم
- چه ترمینال‌هایی به سیستم متصل هستند
- قواعد مربوط به کنترل صحت و دقت داده‌های ذخیره شده در پایگاه داده‌ها
- ضوابط کنترل ایمنی داده‌ها
- و برخی اطلاعات دیگر

دیکشنری داده‌ها معمولاً جزیی از سیستم است و به دو صورت فعال و غیرفعال تولید می‌شود. دیکشنری فعال آن است که هر بار که پایگاه داده‌ها مورد دستیابی قرار می‌گیرد، واحدهایی از سیستم بسته به نوع درخواست کاربر وارد شده و بر اساس اطلاعات موجود در آن، درخواست کاربر نهایتاً انجام می‌شود ولی دیکشنری غیر فعال فقط توسط طراح و کاربران خاصی و نیز تیم مدیریت پایگاه داده‌ها استفاده می‌شود تا اطلاعاتی از آن بدست آورند و خود از آن استفاده نمی‌کنند.

- **کارآیی :** DBMS باید تمامی عملکردهای ارائه شده قبلی را با کارآیی هر چه بیشتر انجام دهد.

رده‌بندی سیستم‌های مدیریت پایگاه داده‌ها

این نرم‌افزار (DBMS) را می‌توان از چندین نظر رده‌بندی کرد :

۱) **از نظر نوع ساختاری**
- سیستم رابطه‌ای
- سیستم سلسله مراتبی
- سیستم شبکه‌ای
- جز اینها

۲) **از نظر محیط سخت افزاری**
- وابسته به یک محیط سخت‌افزاری خاص
- نا وابسته به یک محیط سخت‌افزاری خاص

در گونه دوم می‌بینیم که سیستم از نظر سخت‌افزاری قابلیت جابجاپذیری (Portability) را دارا بوده.

۳) **از نظر رده کامپیوتر**
- مخصوص استفاده در کامپیوترهای شخصی
- مخصوص استفاده در کامپیوترهای متوسط (Mini Copmuter)
- مخصوص استفاده در کامپیوترهای بزرگ (Main Computer)
- مخصوص استفاده در کامپیوترهای خیلی بزرگ (Super Computer)
- اجرا شونده در چند رده کامپیوتر

۴) **از نظر محیط سیستم عامل**
- وابسته به یک محیط سیستم عامل خاص
- اجرا شونده در محیط چند سیستم عامل

در گونه دوم می‌گوییم که سیستم از نظر سیستم عامل، جابجاپذیری دارد.

۵) **از نظر نوع معماری سیستم پایگاه داده‌ها**

- با توانش ایجاد پایگاه متمرکز (Centrralized Database)
- با توانش ایجاد پایگاه نامتمرکز

در گونه دوم می‌گوئیم که DDBMS (Distributed DBMS) داریم به بیان دیگر سیستم دارای توانش ایجاد و مدیریت پایگاه داده‌های توزیع شده است.

۶) از نظر معماری مشتری-خدمتگذار (Cilent-Server Database)

- با توانش ایجاد معماری چند مشتری – یک خدمتگزار
- با تونش ایجاد معماری چند مشتری – چند خدمتگزار

رده بندی ۶ لزوماً همان رده بندی ۵ نیست. سیستم با معماری مشتری-خدمتگزار حالت خاصی از سیستم با معماری توزیع شده است.

در گونه اول سیستم را تک خدمتگزار (Mono-Server) و در گونه دوم چند خدمتگزار (Multi-Server) می‌گوییم. در گونه دوم تعداد خدمتگزارها قابل توجه است و باید دید که آیا سیستم فقط با معماری SSM (Single-Server Model) عمل می‌کند یا در معماری MSM (Multiple-Server Model) هم می‌تواند عمل کند.

۷) از نظر نوع زبان داده‌ای فرعی

- دارای I.DSL
- دارای E.DSL
- دارای I/E.DSL

در گونه E.DSL باید دید که آیا سیستم دارای پیش کامپایلر است یا از طریق حکم فراخوانی موجود در زبان میزبان عمل می‌کند.

۸) از نظر ماهیت زبان داده‌ای فرعی

- با زبان رویه‌ای
- با زبان نارویه‌ای

۹) از نظر سیستم فایل

- خودکفا
- وابسته به سیستم فایل محیط سیستم عامل

۱۰) از نظر نوع کاربرد

- تک منظوره (Special Purpose)
- همه منظوره (General Purpose)

تک منظوره یعنی سیستمی که برای کاربرد خاصی طراحی و تولید شده است. سیستم همه منظوره می‌تواند در کاربردهای گوناگون استفاده شود.

۱۱) از نظر قیمت

از حدود ده هزار دلار تا صد هزار دلار و گاه بیشتر. سیستمهای اجرا شونده در محیط کامپیوترهای شخصی بین صد تا سه هزار دلار قیمت دارند.

۱۲) از نظر طرز برپایی (Setup)

- با محدودیت برپایی یکپارچه
- دارای امکان برپایی گزینشی (Selective Setup)

۱۳) از نظر واسط کاربر (User Interface)

- با واسط زبانی
- با واسط غیر زبانی
- با هر دو واسط

۱۴) از نظر رفتار در قبال رویدادها

- سیستم فعال (Active DBMS)
- سیستم غیرفعال

سیستم فعال سیستمی است که در قبال رویدادهای در پایگاه داده‌ها (مثلاً انجام یک عمل بهنگام سازی یا حذف یا پدید آمدن افزونگی و ...) عکس‌العمل مناسب خودکار دارد.سیستم غیرفعال قابلیت مزبور را ندارد.

۱۵) از نظر متدولوژی زبان پایگاه داده‌ای

- بدون متدولوژی شئ‌گرایی
- دارای متدولوژی شئ‌گرایی

اداره کننده داده‌ها یا مدیر داده‌ها (Data Administrator)

شخصی است که مسئولیت کنترل اصلی بر روی داده‌ها را بر عهده دارد. این فرد یا افراد باید داده‌ها را درک نموده و نیازهای موسسه را نسبت به داده‌ها در سطح مدیریت عالی قرار دهند. وظیفه اداره کننده داده‌ها (DA) آن است که در مرحله اول تصمیم بگیرد که چه داده‌هایی باید در بانک اطلاعاتی ذخیره شود و سپس هنگامی که داده‌ها ذخیره شدند، سیاستگذاری‌های لازم را جهت نگهداری و کار با آنها تعیین کند. (مثلاً اینکه چه کسی می‌تواند چه اعمالی را بر روی چه داده‌هایی انجام دهد)

اداره کننده بانک اطلاعاتی یا مدیر پایگاه داده‌ها (DataBase Administrator)

مدیر بانک اطلاعاتی فردی است که پشتیبانی‌ها تکنیکی لازم را برای پیاده‌سازی تصمیمات اداره کننده داده‌ها (مدیر داده‌ها-DA) فراهم می‌سازد. بنابراین DBA مسئول کنترل کلی سیستم در یک سطح تکنیکی می‌باشد. این مدیر معمولاً همراه با یک تیم تخصصی کار می‌کند که به آن تیم مدیریت پایگاه داده‌ها می‌گویند. هر یک از اعضای این تیم مسئولیت خاصی دارند و در حیطه اختیارات و وظایفش می‌تواند سرپرست یک تیم اجرایی باشد.

برخی از مسئولیت‌های اصلی در این تیم تخصصی عبارتند از

- مدیر پایگاه داده‌ها
- مدیر داده‌ها
- مسئول تیم‌های برنامه‌سازی
- مسئول کنترل کارآیی DBMS
- مسئول نظارت بر عملیات روی پایگاه داده‌ها
- مسئول تنظیم مستندات

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

- **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، بهره‌برداری و گاه بهینه‌سازی و احتمالاً گسترش سیستم را بر عهده دارد
- **DBA در معنای گسترده** : با این معنا، تیم DBA خود همه مراحل لازم برای ایجاد پایگاه داده‌های سازمان را انجام می‌دهد و سپس نگهداری، بهره‌برداری و بهینه‌سازی و گسترش آن را بر عهده می‌گیرد.

وظایف مدیر بانک اطلاعاتی (DBA)

- **تعریف شمای ادراکی** : وظیفه اداره کننده داده‌ها (DA) این است که مشخص کند که چه اطلاعاتی باید در بانک اطلاعاتی قرار گیرد. به عبارت دیگر، این وظیفه شامل تشخیص نهادهای موردنظر و تعیین اطلاعاتی است که درباره این نهادها باید ثبت گردد. این فرآیند بنام طراحی بانک اطلاعاتی منطقی و گاهی اوقات ادارکی نامیده می‌شود. بنابراین، یک مرتبه که اداره‌کننده داده‌ها (DA) در خصوص محتویات بانک اطلاعاتی در یک سطح مجرد و انتزاعی تصمیم گیری نمود، آنگاه DBA با استفاده از DDLادراکی، شِمای ادراکی متناظر را ایجاد خواهد نمودو شکل مقصد یا اصطلاحاً Object این شِمای برای دسترسی به درخواست‌ها توسط DBMS مورد استفاده قرار خواهند کرفت. شکل منبع یا اصطلاحاً Source نیز به عنوان مستندات و اطلاعات قابل رجوع برای کاربران سیستم مورد استفاده قرار خواهند گرفت.
- **تعریف طرح داخلی** : همچنین DBA باید تصمیم بگیرد که داده‌های ذخیره شده در بانک اطلاعاتی، چگونه نمایش داده شوند. این فرآیند را معمولاً طراحی پایگاه داده‌های فیزیکی می‌نامند. بعد از انجام طراحی فیزیکی، DBA باید ساختار ذخیره‌سازی مربوطه را با استفاده از DDL داخلی ایجاد کند (طراحی داخلی). به علاوه باید نگاشت بین طرح داخلی و ادراکی را نیز تعریف کند.
- **ارتباط با کاربران** : این وظیفه DBA است که با کاربران ارتباط برقرار کند تا مطمئن شود که داده‌های موردنیاز آنها موجود است و طرح‌های خراجی موردنیاز با استفاده از DDL خارجی بنویسد. جنبه‌های دیگر وظیفه ارتباط با کاربران، شامل طراحی برنامه‌های کاربردی، فراهم نمودن آموزش‌های تکنیکی، کمک در شناسایی مشکل و حل آن و خدمات حرفه‌ای مربوط به سیستم می‌باشد.
- **تعریف زیر روال‌های پشتیبانی و بازیابی** : بعد از اینکه سازمانی سیستم بانک اطلاعاتی را بکار گرفت، آن سازمان به‌نحوی وابسته به عملیات موفق سیستم می‌شود. در رویداد خرابی هر قسمتی از پایگاه داده‌ها که توسط خطای انسانی، خرابی سخت‌افزار و یا عدم پشتیبانی سیستم‌عامل رخ می‌دهد، ضروری است که قادر به ترمیم داده‌ها با حداقل زمان تأخیر و حداقل تأثیر ممکن روی بقیه سیستم باشیم. DBA باید طرح بازیابی را تعریف و پیاده‌سازی کند. این شِمای ترمیم Backupگیری از بانک اطلاعاتی بر روی دیسک‌های پشتیبان و راههایی برای Restore کردن بانک اطلاعاتی از روی آخرین Backup را شامل می‌باشد.
- **نظارت بر عملکرد و پاسخ‌دهی به تغییر نیازمندیها** : همانگونه که قبلاً بیان شد، DBA مسئول سازماندهی سیستم برای بدست آوردن بهترین کارآیی برای موسسه می‌باشد و همزمان با تغییر نیازمندیها موسسه تنظیمات متناسب باید اعمال شود. هر تغییری در سطح ذخیره‌سازی فیزیکی (داخلی) سیستم باید همراه با تغییر متناسب با تعریف نگاشت از سطح ادراکی باشد، بنابراین طرح ادراکی می‌تواند ثابت باقی بماند.

تفاوت DA با DBA

توجه داشته باشید که اداره کننده داده‌ها (DA) یک مدیر است و نه ی: تکنسین (اگر چه وی نیاز دارد بعضی از قابلیتهای سیستم‌های بانک اطلاعاتی در سطح تکنیکی را بداند). شخص تکنیکی که مسئول پیاده‌سازی تصمیمات مدیر داده است، مدیر پایگاه داده‌ها (DBA) می‌نامند.

بنابراین DBA برخلاف مدیر داده‌ها، یک شخص حرفه‌ای در پردازش داده‌ها (Data Processing) می‌باشد. وظیفه DBA ایجاد یک پایگاه داده واقعی و پیاده‌سازی کنترل‌های تکنیکی موردنیاز برای اعمل تصمیمات و سیاستگذاری‌های مدیر داده‌ها می‌باشد. همچنین DBA مسئول دادن اطمینان برای اجرای کارآمد عملیات سیستم و فراهم نمودن سایر سرویس‌های تکنیکی مرتبط می‌باشد. DBA از میان برنامه‌نویسان سیستم‌ها و سایر افراد حرفه‌ای انتخاب می‌شود. در عمل، وظایف DBA توسط یک گروه چند نفری انجام می‌شود.

مدیر ارتباطات داده‌ای یا مدیر DC (Data Communication Manager)

درخواستهای یک کاربر نهایی از بانک اطلاعاتی از ایستگاه کاری کاربر به شکل پیامهای ارتباطی ارسال یا مخابره می‌شود.(از ایستگاه کاری یک کاربر که می‌تواند به لحاظ فیزیکی از خود سیستم دور باشد به بعضی از برنامه‌های کاربردی دورن خطی و یا حتی DBMS) بطور مشابه، عکس‌العمل‌های برگشتی مربوط به کاربران نیز به صورت چنین پیام‌هائی ارسال خواهند شد (از DBMS و برنامه‌های کاربردی درون خطی به ایستگاه کاری استفاده کننده). تمام این ارسال پیام‌ها تحت نظارت نرم‌افزاری دیگر به نام "**مدیر ارتباطات داده**"(مدیر DC) صورت می‌گیرد. مدیر DC بخشی از DBMS نبوده بلکه یک سیستم مستقل است. هر چند، از آنجا که مدیر DC و DBMS بطور مسلم نیازمند این هستند که با هم کار کنند، آنها گاهی اوقات به عنوان شرکا برابر، در یک مشارکت همکاری سطح بالا به نام "**سیستم بانک اطلاعاتی/ ارتباطات داده**" (سیستم DB/DC) تلقی می‌گردند به نحوی که DBMS بعد از بانک اطلاعاتی به‌نظر می‌رسد و مدیر DC تمام پیام‌ها را به/از DBMS انجام می‌دهد.

معماری سیستم پایگاه داده‌ها

منظور از معماری سیستم پایگاه داده‌ها، پیکربندی یا طرز ترکیب (Composition) اجزای سیستمی است که در آن حداقل یک پایگاه داده‌ها، یک سیستم مدیریت پایگاه داده‌ها، یک سیستم عامل، یک کامپیوتر با دستگاههای جانبی و تعدادی کاربر (برنامه‌ساز و نابرنامه‌ساز) وجود دارد و خدمات پایگاهی به کاربران ارائه می‌کند.این معماری بسیار بستگی به دو عنصر اصلی سیستم یعنی سخت‌افزار و نرم‌افزار مدیریت(DBMS) دارد. البته عوامل دیگری هم در طراحی این معماری دخالت دارند از جمله موقعیت جغرافیایی کاربران، نیازهای کاربران، ماهیت پردازشها و تراکنشها، حجم داده‌های ذخیره شدنی، موقعیت مکانی داده‌ها و ارتباطات بین آنها، ماهیت کاربردهای موردنظر و برخی عوامل دیگر.

در اساس چهار معماری برای سیستم پایگاه داده‌ها وجود دارد :

۱. پایگاه داده‌ها با معماری متمرکز (Centralized Database System)
۲. پایگاه داده‌ها با معماری مشتری–خدمتگزار (Cliented-Server Database System)
۳. پایگاه داده‌ها با معماری توزیع شده (Distributed Database System)
۴. پایگاه داده‌ها با معماری موازی (Parallel Database System)

پایگاه داده‌ها با معماری متمرکز (Centralized Database System)

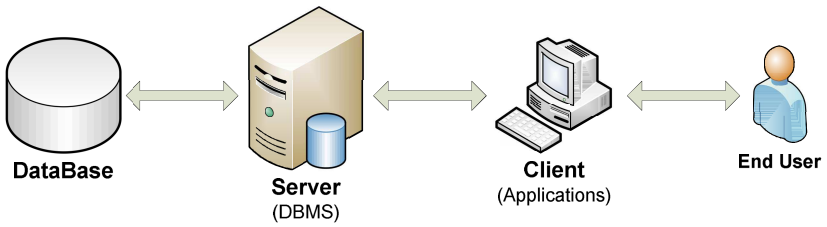
در این معماری یک پایگاه داده‌ها روی یک سیستم کامپیوتری و بدون ارتباط با سیستم کامپیوتری دیگر ایجاد می‌شود. سخت افزار این سیستم می‌تواند کامپیوتر شخصی، متوسط و یا بزرگ باشد و طبعاً قدرت، توان و کارآیی سیستم نیز متفاوت است.

سیستم با معماری متمرکز که روی یک کامپیوتر شخصی ایجاد می‌شود، تک کاربری، برای کاربردهای کوچک و با امکانات محدود است و سیستم مدیریت (DBMS) نیز توانش چندانی ندارد. اما سیستم معماری متمرکز بر روی کامپیوترهای متوسط و به ویژه بزرگ متصل به تعداد زیادی پایانه، می‌تواند سیستم کارآیی باشد.

پایگاه داده‌ها با معماری مشتری–خدمتگزار (Cliented-Server Database System)

هر معماری که در آن قسمتی از پردازش را یک برنامه، سیستم یا ماشین انجام دهد و انجام قسمت دیگری از پردازش را از برنامه، سیستم یا ماشین دیگر بخواهد، معماری مشتری–خدمتگزار نامیده می‌شود. در واقع وظایفی که باید "سیستم" انجام دهد به دو رده تقسیم می‌شوند : رده‌ای که انجام آنها برعهده خدمتگزار است و رده‌ای 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. با این تعریف، در این معماری یک ماشین (یا سیستم یا برنامه) خدمتی را به ماشین (یا سیستم یا برنامه) دیگر ارائه می‌کند از اینرو به این ماشین (یا سیستم یا برنامه) خدمتگزار می‌گویند. خدماتی که خدمتگزار ارائه می‌کند متنوع است، مثلاً خدمات چاپ، خدمات فایلی، خدمات پست الکترونیکی، خدمات پیام‌گیری و پیام‌رسانی، خدمات شبکه جهانی اطلاع رسانی، خدمات پایگاه داده‌ای و ... . و بسته به نوع خدمت، خدمتگزار را با نامی مناسب می‌نامند: مثلاً خدمتگزار چاپ، خدمتگزار فایل، خدمتگزار پست الکترونیکی، خدمتگزار پیام، خدمتگزار پایگاه داده‌ها و ... .

بنابراین منظور از معماری مشتری- خدمتگزار آن نوع از معماری است که در آن مسئولیتها بطور منطقی تقسیم شده است.

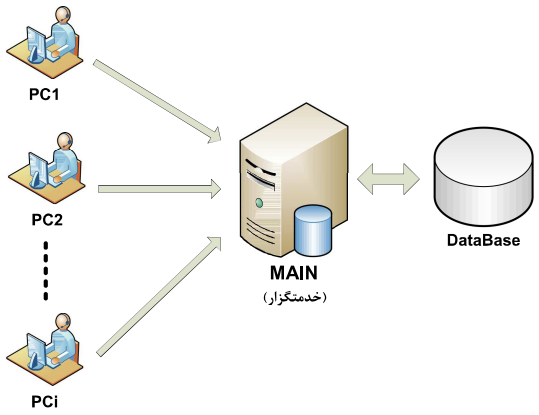


معماری مشتری–خدمتگزار

- **از نظر تعداد مشتری و خدمتگزار :** بطور کلی طرحهای زیر وجود دارند
  ۱- چند مشتری – یک خدمتگزار (Multi-Client/Server (MC/S))
  ۲- یک مشتری – چند خدمتگزار (Client/Multi-Server (C/MS))
  ۳- چند مشتری – چند خدمتگزار (Multi-Client/Multi-Server (MC/MS))
- **از نظر پیکربندی سخت افزاری :** از این نظر دو طرح وجود دارد

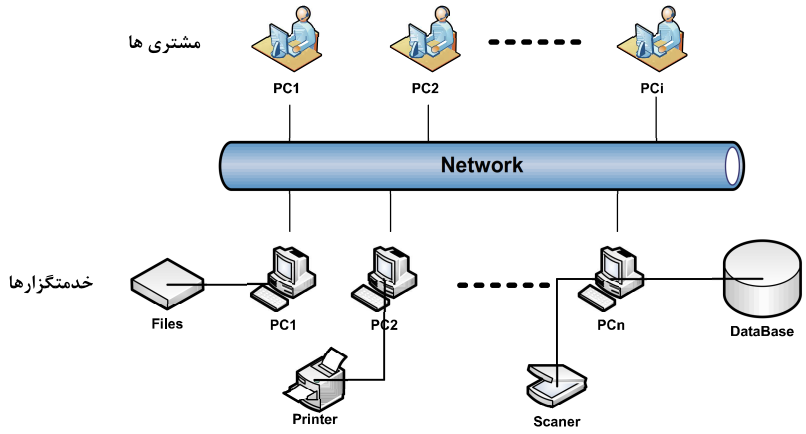
۱- معماری حول کامپیوتر بزرگ (Main Frame Centric)

در این طرح، ماشین خدمتگزار یک کامپیوتر بزرگ است و پایگاه داده‌ها روی همین کامپیوتر ایجاد و مدیریت می‌شود و تعدادی کامپیوتر شخصی از خدمات پایگاهی این کامپیوتر بزرگ استفاده می‌کنند.



۲- معماری حول شبکه (Network Centric)

در این طرح، تعدادی کامپیوتر شخصی به عنوان خدمتگزار و تعدادی دیگر به عنوان مشتری، از طریق شبکه بهم مرتبطاند. یک(یا بیش از یک) کامپیوتر شخصی، خدمتگزار پایگاه داده‌هاست و خدمتگزاران دیگری هم می‌توانند وجود داشته باشند.



مزایای معماری مشتری–خدمتگزار

در مقایسه با معماری متمرکز، این نوع معماری مزایای زیر را دارد:

- کاهش ترافیک شبکه (در معماری حول شبکه)
- استقلال ایستگاه‌های کار
- اشتراک داده‌ها

معماری سیستم پایگاه داده‌های توزیع شده

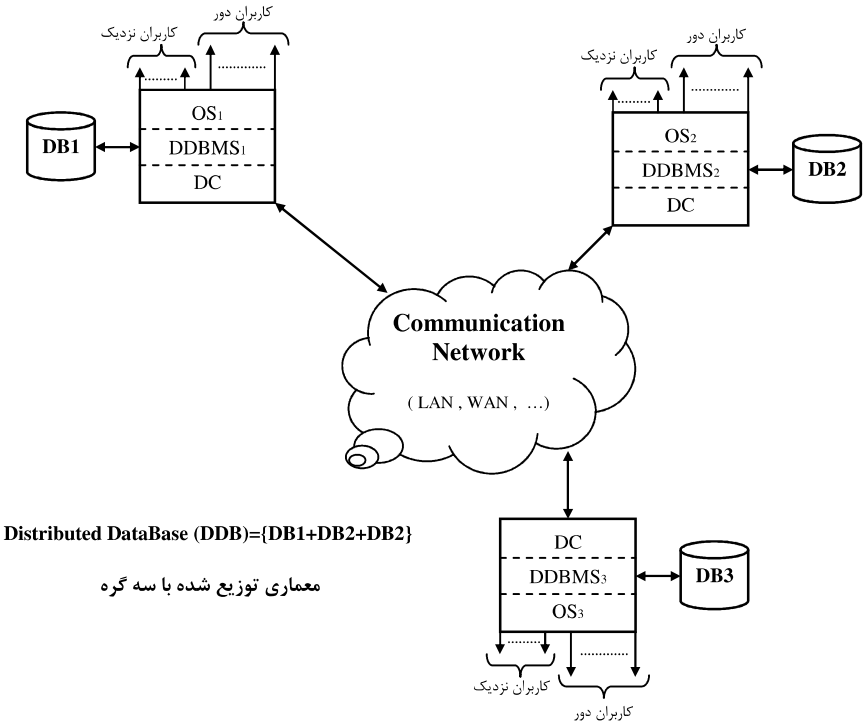
اصطلاح پردازش توزیعی بدین مفهوم است که ماشین‌های مجزا می‌توانند در یک شبکه ارتباطی به یکدیگر متصل شوند بگونه‌ای که یک عملکرد پردازش داده منفرد می‌تواند بر روی چندین ماشین شبکه پراکنده شود. در سیستم پایگاه داده‌های توزیع شده، پایگاه داده‌ها بر روی چند کامپیوتر ذخیره می‌شود. کامپیوترها در یک سیستم توزیع شده از طریق رسانه‌های مختلف ارتباطی نظیر شبکه‌های با سرعت بالا یا خطوط تلفن به یکدیگر مرتبط هستند. آنها در حافظه اصلی یا دیسکها با هم سهیم نیستند. کامپیوترها در یک سیستم توزیع شده از نظر اندازه و وظیفه متفاوتند و گستره آنها از ایستگاههای کار تا سیستمهای کامپیوتر بزرگ را در بر می‌گیرد. کامپیوترها در یک سیستم توزیع شده با نامهای گوناگونی نظیر سایت (sites) یا گره‌ها (Nodes) یا مانه (گره یا کامپیوتر) مورد اشاره قرار می‌گیرند، اساساً واژه سایت را به کار می‌بریم تا تأکیدی بر توزیع فیزیکی این سیستم‌ها داشته باشیم.

می‌توان گفت که در معماری تعدادی پایگاه داده‌ها ذخیره شده روی کامپیوترهای مختلف داریم که از نظر کاربران، پایگاه داده واحدی هستند. به بیان دیگر، مجموعه‌ای است از چند پایگاه داده منطقاً بهم مرتبط و توزیع شده روی یک شبکه کامپیوتری. ارتباطات بین ماشین‌های مختلف توسط بعضی از نرم‌افزارهای مدیریت شبکه انجام می‌شود. (احتمالاً شاخه‌ای از مدیر DC)

نکته : توجه داشته باشید که در این معماری، در سطح طراحی منطقی پایگاه، در آغاز یک پایگاه داده‌های یکپارچه داریم که طرح بر اساس یک استراتژی توزیع و یک طرح تخصیص مشخص، داده‌هایش را در چند مانه (گره یا کامپیوتر) توزیع می‌کند. گره‌ها با یکدیگر چنان همکاری دارند که هر کاربر می‌تواند به داده‌های مورد نیازش در هر گره یا کامپیوتر دستیابی داشته باشد به گونه‌ای که انگار داده‌ها در کامپیوتر خودش ذخیره شده باشند.

در این معماری هر گره یا کامپیوتر خود یک سیستم پایگاه داده‌هاست یعنی : پایگاه داده‌ها، سیستم مدیریت پایگاه داده‌ها (DBMS) و مدیر انتقال داده‌ها (DC) دارد. اصطلاحاً می‌گوییم تعدادی DBMS محلی (Local DBMS) داریم و برای ایجاد هماهنگی بین این سیستم‌های محلی، عنصر نرم‌افزار خاصی که نوعی گسترش DBMS است، لازم است.

در واقع در هر گره یا کامپیوتر یک DDBMS (Distributed DataBase Manamement System) داریم، یعنی یک DBMS با توان ایجاد و مدیریت پایگاه داده‌های 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.مثالی از طرح کلی این معماری را مشاهده می‌کنید.



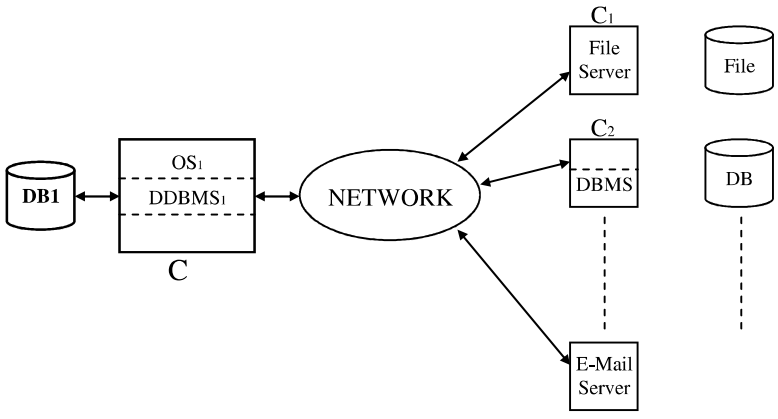
Distributed DataBase (DDB)={DB1+DB2+DB2}

معماری توزیع شده با سه گره

با توجه به تعریف این نوع معماری . طرح کلی آن (در شکل بالا) ویژگیهای این سیستم را می‌توان چنین برشمرد :

- مجموعه‌ای از داده‌ها منطقاً مرتبط و اشتراکی
- داده‌ها به بخشهایی تقسیم و در مانه‌ها توزیع شده‌اند
- بعضی از بخشها ممکن از بطور تکراری در مانه‌ها مختلف ذخیره شده باشند
- مانه‌ها از طریق یک شبکه بهم مرتبط‌اند
- داده‌های ذخیره شده در هر مانه تحت کنترل یک DBMS است
- DBMS در هر مانه می‌تواند برنامه‌های کاربردی محلی (Local Application) را بطور اتوماتیک اجرا کند
- هر DBMS حداقل در اجرای یک برنامه کاربردی سرتاسری (Global Application) مشارکت دارد

نکته : صرف وجود شبکه در یک محیط، به معنای وجود پایگاه داده توزیع شده در آن محیط نیست. مثالی از طرح یک معماری که پایگاه داده توزیع شده نیست در زیر مشاهده می‌کنید.



در این طرح تعدادی کامپیوتر از طریق شبکه، از کامپیوتر C1 خدمات پایگاهی می‌گیرند و حتی کامپیوتر C2، یک پایگاه داده‌های محلی (Local Database) خاص خود را دارد، ولی این معماری توزیع شده نیست بلکه می‌تواند یک معماری پایگاهی با چند مشتری و یک خدمتگزار باشد.

مزایای معماری توزیع شده

۱. سازگاری و هماهنگی با ماهیت سازمانهای نوین
۲. کارآیی بیشتر در پردازش داده‌ها به ویژه در پایگاه داده‌های بزرگ
۳. دستیابی بهتر به داده‌ها
۴. اشتراک داده‌ها
۵. افزایش پردازش موازی
۶. تسهیل گسترش سیستم
۷. استفاده از پایگاه داده‌های از قبل موجود

معایب معماری توزیع شده

۱. پیچیدگی طراحی سیستم
۲. پیچیدگی پیاده‌سازی
۳. کاهش کارآیی در برخی موارد
۴. هزینه بیشتر
۵. مصرف حافظه بیشتر

معماری سیستم پایگاه داده‌های موازی

سیستمهای موازی با استفاده از پردازنده‌های چندگانه و دیسکها به صورت موازی پردازش و سرعتهای ورودی / خروجی را بهبود می‌بخشند. ماشینهای موازی بطور روزافزون متداول می‌گردند و بدین لحاظ مطالعه سیستمهای پایگاه داده‌های موازی را مهمتر می‌سازند. نیروی محرکه موجود در پشت سیستمهای پایگاه داده‌های موازی تقاضاهای کاربردهائی بوده است که نیاز به پرس و جو در پایگاه داده‌های فوق‌العاده بزرگ دارند (در حد ترابایتها) یا نیاز دارند که تعداد فوق‌العاده زیادی از تراکنشها را در هر ثانیه پردازش نمایند.

در پردازش موازی تعداد زیادی عملیات بطور همزمان اجرا می‌گردند که با پردازش سریالی که در آن گامها محاسباتی بطور ترتیبی انجام می‌گیرند، در تضاد می‌باشد.

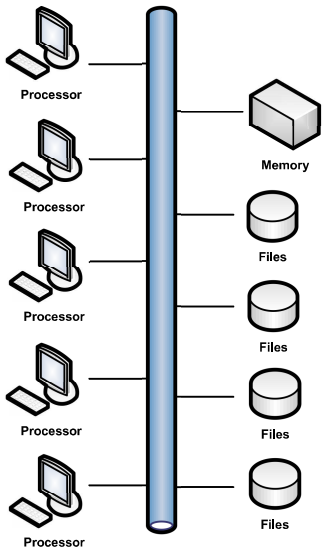
دو مقیاس اصلی عملکرد در یک سیستم پایگاه داده‌ها وجود دارند. اول توان عملیاتی است که عبارت است از تعداد کارهائی که می‌توانند در یک فاصله زمانی معینی انجام گیرند. دومی زمان پاسخگوئی است که عبارت است از مدت زمانی که طول می‌کشد تا یک کار منفرد از زمان تحویل تا به اتمام رسد. یک سیستم که تعداد زیادی از تراکشنهای کوچک را پردازش می‌کند، می‌تواند از طریق پردازش ترکنشها به صورت موازی توان عملیاتی را بهبود ببخشد. یک سیستمی که تراکنشهای بزرگی را پردازش می‌کند می‌تواند از طریق انجام کارهای فرعی هر تراکنش به صورت موازی، زمان پاسخگوئی و توان عملیاتی را بهبود ببخشد.

انواع معماری‌های پایگاه داده‌های موازی

برای ایجاد پایگاه داده‌ها با معماری پردازش موازی، بطور کلی چهار مدل یا طرح وجود دارد :

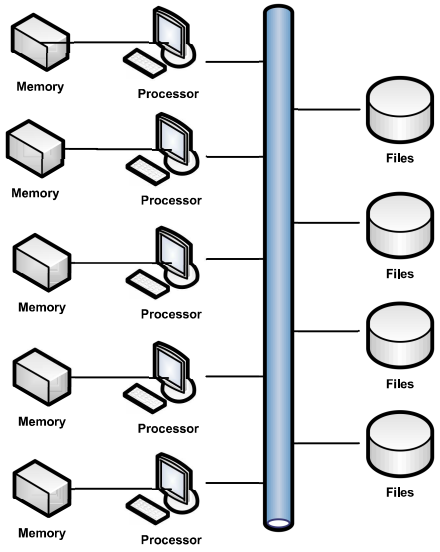
۱. معماری با حافظه مشترک
۲. معماری با دیسک مشترک
۳. معماری بی‌اجزای مشترک
۴. معماری سلسه مراتبی

معماری با حافظه مشترک : در یک معماری با حافظه مشترک، تمام پردازنده‌ها از یک حافظه مشترک استفاده می‌کنند. در این معماری تمام پردازنده‌ها و دیسکها بطور معمول از طریق یک خط حامل یا از طریق شبکه بهم مرتبط به حافظه مشترک دسترسی دارند. مزیت این طرح این است که ارتباط پردازنده‌ها بطور کارا انجام می‌شود. زیرا پیامها بین پردازنده‌ها با نوشتن در حافظه مشترک مبادله می‌شود که زمان آن کمتر از میکروثانیه است (البته در مقایسه با حالتی که پیامها توسط یک مکانیزم ارتباطی فرستاده می‌شوند). داده‌های ذخیره شده در حافظه مشترک در اختیار همه پردازنده‌ها قرار دارد و می‌تواند توسط همه مورد دستیابی قرار گیرند. (در طرح زیر و طرحهای بعدی معماری موازی، P بجای پردازنده و M به جای حافظه بکار برده می‌شود)



عیب این معماری در این است که نمی‌توان بیش از ۳۲ یا ۶۴ پردازنده داشت، زیرا احتمال بروز تنگنا در باسهای حافظه‌ای شبکه ارتباطی افزایش می‌یابد. البته اگر در هر پردازنده بافرهای نهان (Cache) با اندازه بزرگ وجود داشته باشد، دفعات مراجعه به حافظه اصلی کاهش می‌یابد، هر چند که نمی‌توان همه داده‌ها را در این بافرها جای داد.

معماری با دیسک مشترک : در مدل دیسک مشترک تمام پردازنده‌ها از طریق شبکه بهم مرتبط می‌توانند به تمام دیسکها دسترسی داشته باشند ولی پردازنده‌ها دارای حافظه‌های اختصاصی هستند.



معماری در قالب دیسک مشترک نسبت به معماری در قالب حافظه مشترک دو برتری دارد . اول اینکه چون هر پردازنده حافظه مخصوص به خود را در اختیار دارد خط حامل حافظه، گلوگاه کاهنده سرعت نیست. دوم اینکه این معماری یک روش ارزانی جهت تأمین درجه‌ای از تحمل خطا (Fault tolerance) را تأمین می‌نماید. اگر یک پردازنده (یا حافظه‌اش) با خرابی مواجه شود پردازنده‌های دیگر می‌توانند کارهای آنرا بعهده بگیرند، زیرا پایگاه داده‌ها روی دیسکها اسقرار دارد که از تمام پردازنده‌ها قابل دستیابی می‌باشد.

در این نوع معماری هرچند حافظه دیگر گلوگاه کاهنده سرعت نیست ولی ارتباط دهندگی به زیر سیستم دیسک اینک گلوگاه کاهنده سرعت است. این بویژه در مواردی که پایگاه داده‌ها تعداد زیادی دستیابی به دیسک‌ها دارد صدق می‌کند. در قیاس با سیستمهای حافظه مشترک، سیستمهای دیسک مشترک می‌توانند تا حدودی تعداد زیادتری پردازنده در اختیار داشته باشند ولی ارتباط از طریق پردازنده‌ها کندتر است زیرا باید از یک شبکه ارتباطی عبور کند.

مزایای این نوع معماری عبارتست از :

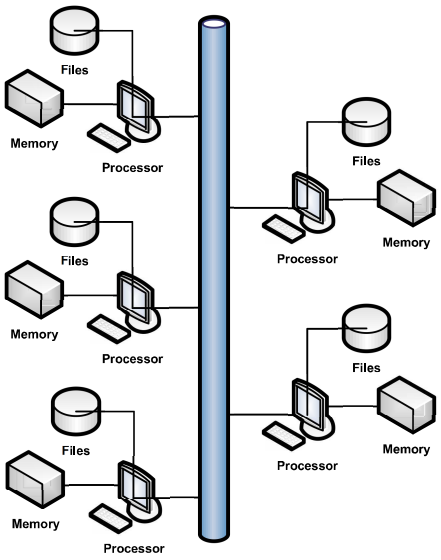
۱. عدم بروز تنگنا در باسهای حافظه

۲. تسهیل تحمل خرابی . زیرا در صورت خراب شدن یک پردازنده یا حافظه، پردازنده دیگر می‌تواند کار را ادامه دهد . البته می‌توان از سیستم RAID هم استفاده کرد.

معایب این نوع معماری عبارتست از :

دشواری در گسترش سیستم، زیرا با افزایش دیسکها و پردازنده‌ها، در ارتباط بین اجزا تنگنا ایجاد می‌آید و سرعت ارتباط بین آنها کاهش می‌یابد.

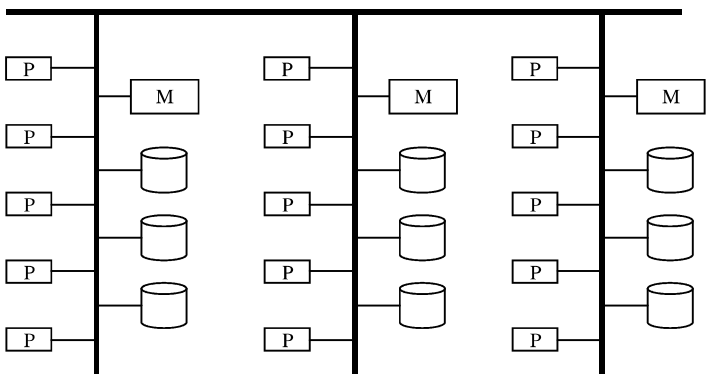
معماری بی‌اجزای مشترک (معماری غیراشتراکی) : در این معماری تمام پردازنده‌ها نه حافظه مشترکی دارند و نه در دیسک مشترک سهیمند. در واقع هر ماشین، پردازنده، حافظه و دیسک خود را دارد. پردازنده یک گره ممکن است با پردازنده گره دیگر از طریق شبکه ارتباط دهنده با سرعت بالا مرتبط گردد. یک گره به عنوان یک خدمت رسان برای داده‌های روی دیسک‌هایی که در اختیار دارد.



چون رجوعهای دیسکی محلی توسط دیسکهای محلی در هر پردازنده سرویس داده می‌شوند مدل غیراشتراکی این عیبی را که لازم است تمام ورودی‌ها/خروجی‌ها از یک شبکه ارتباط دهنده منفرد عبور داده شوند را جبران می‌نماید در عوض تنها پرس‌وجوها، دستیابی‌ها به دیسک‌های غیرمحلی و رابطه‌های منتجه از شبکه عبور داده می‌شوند. علاوه بر آن شبکه‌های ارتباط دهنده جهت سیستمهای غیراشتراکی معمولاً به گونه‌ای ساخته می‌شوند که هر چه به تعداد گره‌ها افزوده می‌شود ظرفیت انتقال آن افزایش می‌یابد. در نتیجه معماری‌های غیر اشتراکی می‌توانند به آسانی از تعداد زیادی پردازنده پشتیبانی نمایند.

عیب عمده سیستم‌های غیراشتراکی هزینه‌های ارتباطات و دستیابی دیسکی غیرمحلی است که بالاتر از معماری حافظه مشترک یا معماری دیسک مشترک می‌باشد زیرا ارسال داده‌ها دربرگیرنده تعامل نرم‌افزاری از دو طرف می‌باشد.

معماری سلسله مراتبی : معماری سلسله مراتبی ترکیبی از ویژگیهای معماری حافظه مشترک، معماری دیسک مشترک و معماری غیراشتراکی می‌باشد. در سطح بالا سیستم شامل گره‌هایی است که با شبکه ارتباط دهنده بهم متصل گردیده‌اند و دیسکها و حافظه را در اشتراک یکدیگر ندارند. بدین ترتیب سطح بالا یک معماری غیر اشتراکی است. هر گره سیستم خود می‌تواند تعداد کمی پردازنده با حافظه مشترک داشته باشد یا یک سیستم با دیسک‌های مشترک باشد. شکل زیر یک معماری سلسه مراتبی را با گره‌های حافظه مشترک که در یک معماری غیرمشترک بهم متصل شده‌اند را نشان می‌دهد.



استقلال داده‌ای

یکی از مهمترین مزایای تکنولوژی پایگاه داده‌ها، بلکه مهمترین هدف آن تأمین استقلال داده‌ای(نابستگی داده‌ای) است.

تعریف : استقلال داده‌ای یعنی وابسته نبودن برنامه‌های کاربردی به داده‌های ذخیره شده

اما به بیان دقیقتر یعنی:

مصونیت دیدهای کاربران در سطح خارجی و برنامه‌های کاربردی آنها در قبال تغییراتی که در سطح ادراکی و سطح داخلی پایگاه داده‌ها بروز می‌کند.

استقلال داده‌ای بر دو نوع می‌باشد:

الف) استقلال داده‌ای فیزیکی (Physical Data Independence – PDI)

عبارت است از مصونیت دیدهای کاربران و برنامه‌های کاربردی در قبال تغییرات در سطح داخلی-فیزیکی پایگاه داده‌ها. این نوع استقلال داده‌ای، بویژه در سیستم‌های رابطه‌ای جدید کاملاً تامین است زیرا

اولاً : کاربران سطح خارجی در محیط کاملاً انتزاعی عمل می‌کنند و برنامه‌های کاربردی در این سطح با فایلینگ پایگاه داده‌ها تماس ندارد.

ثانیاً : بین سطح خارجی و سطح داخلی، یک سطح انتزاعی دیگر واسط است و مانع تاثیرپذیری تغییرات برنامه‌های سطح خارجی در فایلینگ پایگاه داده‌ها می‌شود.

ب) استقلال داده‌ای منطقی (Logical Data Independence – LDI)

عبارت است از مصونیت دیدهای کاربران و برنامه‌های کاربردی در قبال تغییرات در سطح ادراکی پایگاه داده‌ها. با توجه به اینکه سطح خارجی روی سطح ادراکی تعریف می‌شود، بالقوه در معرض تاثیرپذیری از تغییرات در سطح ادارکی است. اما در سیستم‌های امروزی، این نوع استقلال هم تا حدی (ونه صد در صد) تامین است. برای درک این مطلب باید دید تغییرات در سطح ادارکی یعنی چه؟

تغییرات در سطح ادارکی یعنی: تغییر در طراحی منطقی پایگاه داده‌ها و تغییر در شمای ادارکی. اما تغییر در سطح ادارکی خود دو وجه دارد :

- رشد پایگاه داده در سطح ادارکی : به‌علت مطرح‌شدن نیازهای جدید برای کاربران
- سازماندهی مجدد پایگاه داده در سطح ادارکی

مدل‌سازی داده‌ای (Data Modeling) :

در طراحی یک بانک اطلاعاتی، ابتدا می‌بایست مراحلی چون امکان‌سنجی، بررسی نیازها و محدودیتها، بررسی سیستم دستی موجود و ... صورت گیرد که این مراحل در درس مهندسی نرم‌افزار مطرح می‌گردد.
پس از مراحل فوق، طراح بانک به کمک نمودارهایی، شمای کلی بانک اطلاعاتی را مستقل از مدل بانک (شبکه‌ای، سلسه مراتبی و رابطه‌ای) و نیز مستقل از جنبه‌های برنامه‌نویسی ترسیم می‌کند
برای این کار مدل‌های مختلفی وجود دارد که ما در اینجا مدل ER (Entity Relationship)که توسط آقای CHEN در سال ۱۹۶۷ پیشنهاد گردیده است را مورد بررسی قرار می‌دهیم. مدل EER در واقع مدل پیشرفته و گسترش‌یافته ER می‌باشد.

روش ER :

در روش ER، سه مفهوم معنایی وجود دارد و معنای داده‌های هر محیطی به کمک همین سه مفهوم نمایش داده می‌شود :

- موجودیت (Entity)
- صفت (Attribute)
- ارتباط (Relationship)

البته مفهوم دیگری نیز به نام زیر نوع موجودیت هم مطرح است که توضیح داده می‌شود.

موجودیت : قبلاً در درس ذخیره و بازیابی اطلاعات دیدیم که موجودیت عبارتست از مفهو کلی "شیی"، "چیز"، "پدیده" و بطور کلی هرآنچه که می‌خواهیم در موردش "اطلاع" داشته باشیم. به عنوان مثال در یک محیط عملیاتی مانند دانشکده، انواع موجودیت‌هایی که می‌توانیم نام ببریم عبارتند از : دانشجو، درس، استاد، گروه‌آموزشی و ...
موجودیت‌ها به دو دسته کلی تقسیم می‌شوند. موجدیت ضعیف (وابسته)، موجودیتی است که وجود آن وابسته به موجودیت‌های دیگر است، یعنی عدم وجود موجودیت‌های دیگر، سبب از بین رفتن یک موجودیت ضعیف می‌شود. ولی موجودیت قوی (مستقل) موجودیتی است که ضعیف نبوده و بطور مستقل می‌تواند وجود داشته باشد.

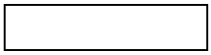
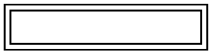
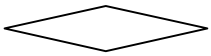
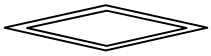
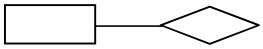
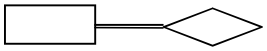
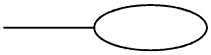
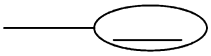
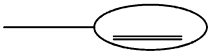
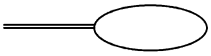
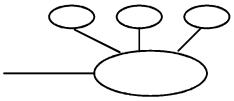
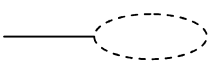
صفت (خصیصه) : صفت در واقع خصیصه یا ویژگی یک نوع موجودیت است و هر نوع موجودیت مجموعه‌ای از صفات (موسوم به مجموعه صفات موجودیت) دارد. هر صفت از نظر کاربران یک نام، یک نوع و یک معنای مشخص دارد. به عنوان مثال، موجودیت درس را در نظر بگیرید، صفات درس عبارتند از : شماره درس، عنوان درس، تعداد واحد درس، نوع درس (پایه، تخصصی، اختیاری و...)، ماهیت درس (نظری، عملی و ...)، سطح درس (کاردانی، کارشناسی و ...) را می‌توان نام برد. هر صفتی، از یک مجموعه از مقادیر معتبر و مجاز، مقدار می‌گیرد که به این مجموعه مقادیر، اصطلاحاً دامنه یا میدان (Domain) مقادیر آن صفت می‌گویند.

ارتباط : یک رابطه یک پیوند بین چند نوع موجودیت است. به عنوان مثال نوع موجودیت‌های دانشجو و درس را در نظر می‌گیریم. می‌توانیم بین این دو موجودیت روابط زیر را نام ببریم:

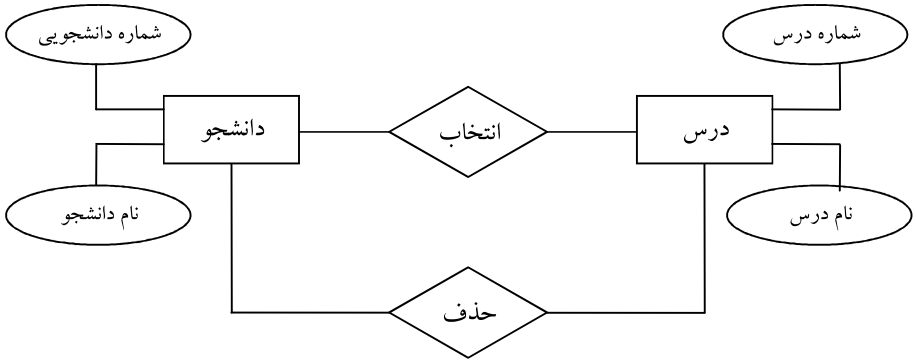
- دانشجو درس را انتخاب می‌کند
- دانشجو درس را حذف می‌کند
- دانشجو در را قبول می‌شود
- و ...

نمودار ER

نموداری است که در آن سه مفهوم اساسی مدل ER یعنی موجودیت، صفت و ارتباط نمایش داده می‌شوند.
برای رسم این نمودار به نمادهایی نیاز داریم که این نمادها را در جدول زیر مشاهده می‌کنید.

معنا	نماد
نوع موجودیت	
نوع موجودیت ضعیف (وابسته)	
نوع ارتباط	
نوع ارتباط با موجودیت ضعیف	
مشارکت موجودیت در ارتباط	
مشارکت الزامی	
صفت	
صفت کلید اول	
صفت کلید دوم (درصورت وجود)	
صفت کلیدی مرکب (مثلاً دو صفتی)	
صفت چندمقداری	
صفت مرکب	
صفت مشتق (مجازی)	

به عنوان مثال، در نمودار ER زیر روابط بین موجودیت‌های دانشجو و درس را مشاهده می‌کنید

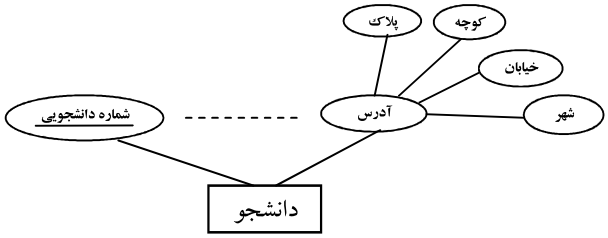


(دقت داشته باشید که در نمودار بالا تمامی صفات و روابط درنظر گفته نشده است)
در همین مثال می‌بینیم که بین دو موجودیت می‌تواند بیش از یک ارتباط وجود داشته باشد.

انواع صفت در نمودار ER

۱. **صفت کلیدی** : کلید عبارتست یک یا چند صفت که در یک موجودیت که یکتایی (Uniqueness) مقدار داشته باشد و حتی الامکان طول مقادیرش کوتاه باشد. مثلاً در موجودیت دانشجو، شماره دانشجویی کلید است زیرا که هر دانشجو یک شماره مخصوص و یکتا دانشجویی دارد ولی نام نمی‌تواند کلید باشد.
گاهی یک صفت به‌تنهایی نمی‌تواند کلید باشد بلکه مجموعه‌ای از دو یا چند صفت تشکیل یک کلید را می‌دهند. مثلاً نام و شماره‌شناسنامه هر فرد به‌تنهایی کلید نمی‌باشند بلکه هر دو با هم می‌توانند تشکیل یک کلید را بدهند.
برای مشخص‌کردن کلید یک موجودیت در نمودار، زیر صفت یا صفات مربوطه را خط می‌کشیم

۲. **صفت ساده (Single) یا مرکب (Composite)** : صفت ساده، صفتی است که مقدار آن از لحاظ معنایی ساده یا اتومیک یا تجزیه‌نشدنی باشد به این معنا که اگر مقدار آن‌را به اجزایی تجزیه کنیم، مقادیر جزیی حاصله فاقد معنا باشد. مثلاً صفت درس یک صفت‌ساده است. صفت مرکب صفتی است که از چند صفت ساده تشکیل شده باشد به گونه‌ای که تجزیه‌شدنی باشد و اجزا حاصله خود صفات ساده باشند. به عنوان مثال، صفت آدرس که از اجزاء : نام استان، نام شهر، نام خیابان، نام کوچه، شماره پلاک و کدپستی تشکیل شده است.



۳. **صفت تک‌مقداری (Single value) یا چندمقداری (Multi value)** : صفت تک‌مقداری، صفتی است که برای یک نمونه از یک نوع موجودیت، حداکثر یک مقدار از دامنه مقادیر را می‌گیرد. به بیان ساده‌تر، به ازای

یک نام صفت، حداکثر یک مقدار برای یک نمونه از موجودیت داشته باشیم. بع عنوان مثال، شماره درس یک صفت تک‌مقداری است چراکه یک نمونه درس نمی‌تواند بیش از یک شماره داشته باشد و یا شماره دانشجویی نیز یک صفت تک‌مقداری است چراکه که هر دانشجو تنها یک شماره دانشجویی دارد. صفت چندمقداری صفتی است که برای بعضی یا همه نمونه‌های موجودیت، بیش از یک مقدار از دامنه مقادیر، صفت می‌گیرد. به بیان ساده‌تر، به ازاء یک نام صفت، چندمقدار برای یک موجودیت داشته باشیم. به عنوان مثال، مدرک دانشگاهی در موجودیت استاد، یک صفت چندمقداری است چراکه یک استاد می‌تواند دارای چندین مدرک با عناوین لیسانس، فوق‌لیسانس، دکترا و ... باشد. و یا صفت شماره تلفن در موجودیت دانشکده نیز یک صفت چندمقداری است چراکه یک دانشکده می‌تواند دارای چندین شماره تلفن باشد.

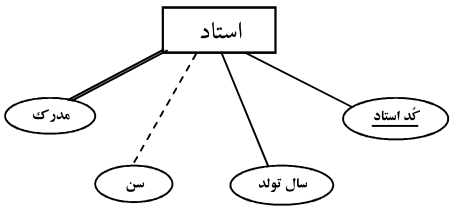
صفات چندمقداری در نمودار ER را با دو خط به موجودیت وصل می‌کنیم.

۴. **هیچ‌مقدار (Nullvalue) پذیر یا ناپذیر** : هیچ‌مقدار یعنی : مقدار ناشناخته (UnKnown)، مقدار غیرقبل اعمال (Inapplicable) ، مقدار تعریف‌نشده (Undefined) ، مقدارشناخته نشده. به عنوان مثال، شماره تلفن یک نمونه دانشجو ممکن است دردست نباشد و یا نام استاد یک درس در برنامه درسی ترم ممکن است هنوز اعلام نشده باشد. حال می‌گوییم که اگر مقدار یک صفت در یک یا بیش از یک نونه از یک نوع موجودیت برابر "هیچ‌مقدار" باشد، آن صفت هیچ‌مقدارپذیر است. دقت داشته باشید که صفت کلید موجودیت نمی‌تواند هیچ‌مقدار پذیر باشد چراکه عامل تمایز نمونه‌های یک نوع موجودیت از یکدیگر است و عامل تمایز خود نمی‌تواند ناشناخته باشد.

۵. **صفت واقعی یا مشتق** : صفت واقعی صفتی است که مقادیرش در پایگاه داده‌ها ذخیره شده باشد (موجود باشد) و البته می‌تواند هیچ‌مقدار هم باشد درصورتی که کلید نباشد.

صفت مشتق صفتی است که مقادیرش در پایگاه داده‌ها ذخیره نشده باشند بلکه حاصل از یک محاسبه یا پردازش بر روی داده‌های موجود می‌باشد. صفت معدل یک دانشجو در صورتی که جزو صفات دانشجو نباشد، یک صفت مشتق می‌باشد. به صفت مشتق گاه صفت مجازی نیز می‌گویند.

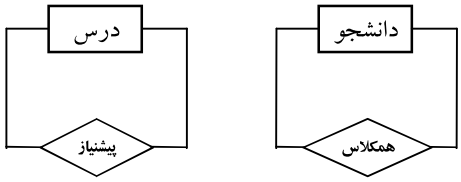
صفت مشتق را در نمودار ER با خطوط نقطه‌چین به موجودیت وصل می‌کنیم.



درجه نوع ارتباط

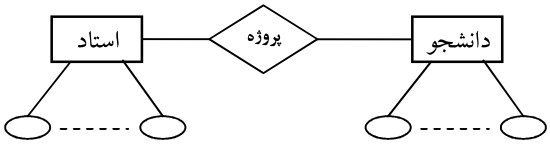
به تعداد موجودیت‌هایی که در یک ارتباط مشارکت دارند، درجه ارتباط نامیده می‌شود. معمولاً این درجه ارتباط ۱ یا ۲ و یا ۳ است و درجات بالاتر بندرت استفاده می‌شود.

مثالی از یک ارتباط درجه ۱ :

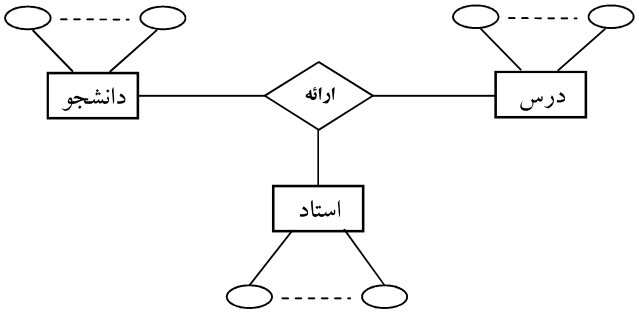


وقتی‌که یک ارتباط بین یک نوع موجودیت و خودش برقرار باشد، آن‌را ارتباط باخود (Self-Relationship) یا بازگشتی (Recursive) می‌گوییم.

مثالی از یک ارتباط درجه ۲ :



مثالی از یک ارتباط درجه ۳ :



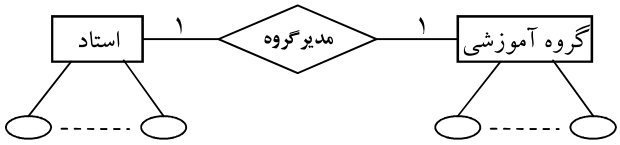
اطلاعی که از یک نوع ارتباط مثلاً سه موجودیتی بدست می‌آید طبعاً از اطلاع حاصلی از یک ارتباط دو موجودیتی، بیشتر است. بعلاوه از یک اطلاع سه موجودیتی می‌توان سه اطلاع دو موجودیتی را استنتاج کرد

اتصال (Connectivity) و حد (Cardinality)

ارتباط از نظر اتصال بر سه نوع است

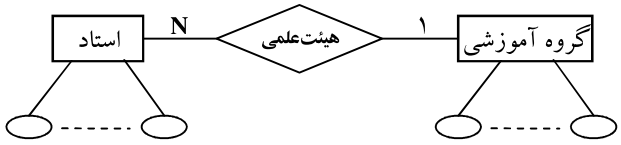
۱) ارتباط 1-1 (یک به یک)

به عنوان نمونه، در یک سیستم دانشگاهی، یک استاد می‌تواند مدیرگروه یک گروه‌آموزشی باشد و یک گروه آموزشی فقط می‌تواند یک استاد را به عنوان مدیرگروه داشته باشد.



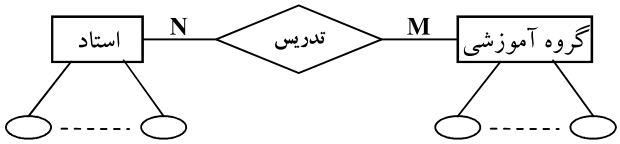
۲) ارتباط 1-M (یک به چند)

به عنوان نمونه، در یک سیستم دانشگاهی، یک استاد می‌تواند تنها در یک گروه‌آموزشی به عنوان هیئت‌علمی عضو داشته باشد ولی یک گروه آموزشی می‌تواند دارای چندین استاد به عنوان هیئت‌علمی باشد.

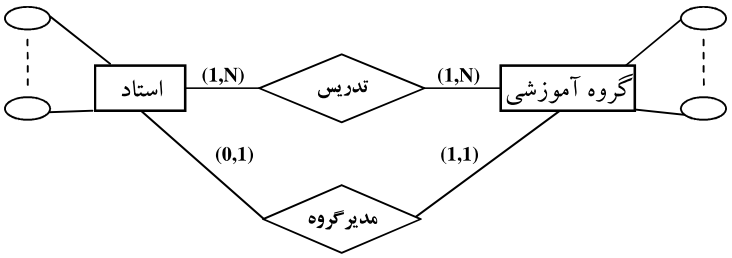


۳) ارتباط N-M (چند به چند)

به عنوان نمونه، در یک سیستم دانشگاهی، یک استاد می‌تواند در چندین گروه آموزشی تدریس داشته باشد و همچنین یک گروه‌آموزشی می‌تواند از چندین استاد چهت تدریس در گروه خود استفاده نماید.



دیگر مشخصه ارتباط، حد می‌باشد که در پایین خط ارتباط مقدار حداقل و حداکثر آن در پرانتز نوشته می‌شود.



وابستگی تابعی (Funcational Dependency)

وابستگی تابعی (یا به اختصار FD) یک رابطه چند-به-یک از یم مجموعه از صفات به مجموعه‌ای از صفات دیگر در یک رابطه معلوم و مشخص می‌باشد. به منظور مفهوم بودن مطلب، رابطه‌های عرضه‌کننده، قطعه و عرضه‌کننده-قطعه را در نظر بگیرید

S (S# , Sname , City)
P (P# , Pname , Color , Weight , City)
SP (S# , P# , QTY)

در ایجا رابطه SP را کمی تغییر می‌دهیم و بدان صفت دیگری بنام City که نشان‌دهنده شهر عرضه‌کننده می‌باشد را اضافه می‌کنیم. جدول جدید را SCP می‌نامیم

SCP :

S#	P#	QTY	City

تعریف : فرض کنید R یک رابطه و همچنین X و Y نیز زیرمجموعه دلخواهی از مجموعه صفات R باشد آنگاه گفته می‌شود که صفت Y دارای وابستگی تابعی نسبت به صفت X است و چنین نمایش می‌دهیم

$$X \to Y$$

اگر و تنها اگر هر مقدار X در رابطه R متناظر با دقیقاً یک مقدار Y در R باشد

به عنوان مثال، در رابطه SCP در بالا دارای وابستگی تابعی (FD) زیر می‌باشد

$$S\# \to City$$

زیرا هر تاپل (چندگانه) با یک مقدار S# مشخص، دارای مقدار یکسانی بای City است. رابطه SCP دارای وابستگی-های تابعی بیشتری است که در زیر به برخی از انها اشاره می‌کنیم

$$\{S\#, P\#\} \to QTY$$
$$\{S\#, P\#\} \to City$$
$$\{S\#, P\#\} \to \{QTY, City\}$$
$$\{S\#, P\#\} \to S\#$$
$$\{S\#, P\#\} \to \{S\#, P\#, QTY, City\}$$
$$S\# \to QTY$$
$$QTY \to S\#$$

نکته : اگر X کلید کاندید و به ویژه کلید اصلی رابطه R باشد، آنگاه هر صفت خاصه دیگر این رابطه الزاماً با X وابستگی تابعی دارد چراکه طبق تعریف، کلید کاندید یکتایی مقدار دارد. البته در تعریف وابستگی تابعی الزامی ندارد که صفت خاصه X کلید رابطه R باشد به بیان دیگر لزومی ندارد که مقدار X فقط در یک تاپل از رابطه R وجود داشته باشد

SCP :

S#	P#	QTY	City
S1	P1	100	London
S1	P2	100	London
S2	P1	200	Paris
S2	P2	200	Paris
S3	P2	300	Paris
S4	P2	400	London
S4	P4	400	London
S4	P5	500	London

با توجه به مثال فوق، می‌توان تعریف زیر را برای وابستگی تابعی ارائه کرد :

- صفت خاصه Y از رابطه R با صفت خاصه X از رابطه R وابستگی تابعی دارد اگر و فقط اگر هر وقت در دو تاپل از R یک مقدار X وجود داشته باشد، مقدار Y نیز در آن دو تاپل یکسان باشد.

نکته : اگر X و Y در رابطه R دارای وابستگی تابعی $X \to Y$ باشند و X کلید کاندید نباشد، آنگاه رابطه R شامل افزونگی اطلاعات خواهد بود. برای مثال در خصوص رابطه SCP، این واقعیت که یک عرضه‌کننده مشخص در شهر خاص واقع است، بطور کلی به میزان زیادی تکرار شده است.

در اینجا، ما معمولاً از اصطلاح وابستگی تابعی برای اشاره به مواردی استفاده می‌کنیم که دارای مفهوم مستقل از زمان باشند. موارد مطرح شده در مثال بالا به عنوان برخی از وابستگی‌های تابعی رابطه SCP همگی مستقل از زمان هستند به جز موارد زیر

$S\# \to QTY$

$QTY \to S\#$

به عبارت دیگر، این جمله که "هر ارسال یک عرضه‌کننده مشخص دارای مقدار عرضه یکسانی است" برای نمونه پرشده در جدول SCP بالا، معتبر است ولی برای تمام زمان‌ها معتبر نیست.

نکته : اگر در R داشته باشیم $X \to Y$
لزوماً رابطه $Y \to X$ برقرار نیست، هرچند گاه ممکن است مقادیر Y در گستره‌ای از R (یک مقدار رابطه مشخص) چنان باشند که این وضع برقرار باشد، اما در حالت کلی چنین نیست.

حال، چنانچه توجه خود را تنها معطوف وابستگی‌های تابعی کنیم که "برای تمام حالات و زمان‌ها" برقرار هستند، مجموعه وابستگی‌های تابعی (FD) تمام مقادیر مجاز و معتبر یک ربطه مشخص مانند SCP می‌تواند خیلی بزرگ گردد. آنچه که ما مایل به انجام آن هستیم پیداکردن روش‌هایی برای کاهش این مجموعه در یک اندازه قابل توجه و قابل مدیریت می‌باشد.
چرا این هدف موردنظر و مطلوب می‌باشد ؟ یکی از دلایل موجود این است که FDها محدودیت‌های جامعیت را نشان می‌دهند و لذا DBMS نیازمند کنترل آنها زمانی که بروز رسانی انجام می‌شود، می‌باشد. بنابراین با مشخص شدن مجموعه S از FDها، پیدا کردن مجموعه دیگری مانند T که (درصورت امکان) کوچکتر از S بوده و دارای این خاصیت است که هر FD واقع در S توسط FD واقع در T ارائه و نشان داده می‌شود، کاری مطلوب و مناسب می‌باشد. چنانچه یک چنین مجموعه T پیدا شود، کافی است که DBMS وابستگی‌های واقع در T را اجرا کند و سپس FDهای

واقع در S بطور خودکار اجرا می‌گردد. بنابراین مسئله پیداکردن یک چنین مجموعه T یک امر کاملاً مورد علاقه و عملی می‌باشد.

بستار (Closure) یک مجموعه از وابستگی‌ها

در یک مجموعه از FDها، وابستگی‌های تابعی خاصی وجود دارند که یکدیگر را تعریف و بیان می‌کنند. به عنوان مثال، FD زیر را در نظر بگیرید :

$$\{S\#, P\#\} \rightarrow \{QTY, City\}$$

این FD خود دو وابستگی زیر را نشان می‌دهد :

$$\{S\#, P\#\} \rightarrow City$$
$$\{S\#, P\#\} \rightarrow QTY$$

به عنوان مثال پیچیده‌تر، فرض کنید دارای رابطه‌ای به‌نام R با سه صفت A ، B و C باشیم بنحوی که وابستگی تابعی $A \rightarrow B$ و $B \rightarrow C$ هر دو در رابطه R برقرار باشند. در این‌صورت واضح است که وابستگی تابعی $A \rightarrow C$ نیز در رابطه R برقرار خواهد بود.

تعریف : مجموعه تمام FDهایی که از FDهایی موجود در F قابل استنتاج باشند، به بستار F موسوم است و آن‌را با F^+ یا S نمایش می‌دهند.

آقای آرمسترانگ ثابت کرد که با اعمال مکرر سه قاعده زیر می‌توان به تمام وابستگی‌های منتج دست یافت و هیچ وابستگی اضافی نیز تولید نمی‌شود.

- **قوانین استنتاجی آرمسترانگ :** فرض کنید A ، B و C زیر مجموعه اختیاری از مجموعه صفات یک رابطه معلوم مانند R بوده و فرض کنید AB اجتماع A و B باشد.

۱. انعکاسی یا بازتابی(Reflexivity) : اگر B زیر مجموعه A باشد ($B \subseteq A$) آنگاه : $A \rightarrow B$

۲. بسط‌پذیری یا افزایش (Augmentation) : اگر $A \rightarrow B$ و C صفت باشد، آنگاه : $AC \rightarrow BC$

۳. تعدی یا انتقال (transitivity) : اگر $A \rightarrow B$ و $B \rightarrow C$ آنگاه: $A \rightarrow C$

این سه قانون، قوانین خودکامل هستند، بدین مفهوم که در خصوص یک مجموعه مشخصی از FDها، تمام FDهای ارائه شده توسط S می‌تواند توسط قوانین فوق گرفته و اخذ گردد. انها همچنین معتبر هستند بدین مفهوم که هیچگونه FD اضافی نمی‌تواند گرفته و اخذ گردد (یعنی FDهایی که توسط S ارائه نمی‌شوند) به عبارت دیگر، قوانین می‌توانند برای بدست آوردن بستار S^+ مورد استفاده قرار گیرند. از سوی دیگر اگر مجموعه از وابستگی‌های تابعی در R داشته باشیم، ممکن است بتوان با استفاده از قواعد آرمسترانگ این مجموعه را کاهش داد، یعنی بعضی از وابستگی‌های موجود در مجموعه وابستگی‌ها را از بعضی دیگر منطقاً استنتاج کرد و بنابراین افزونه‌اند و می‌توان انها را حذف کرد. در این کار هم از قواعد آرمسترانگ استفاده می‌شود.

چندین قانون دیگر می‌تواند از این سه قانون ارائه شده مشتق گردد، موارد زیر بعضی از این قوانین هستند. این قوانین اضافی می‌تواند برای ساده‌کردن یک مسئله عملی در محاسبه S^+ از S بکار رود.

۴. خود تعیینی یا خودوابستگی (Self-Dependency) : $A \rightarrow A$

۵. تجزیه : اگر داشته باشیم $A \rightarrow BC$ آنگاه وخواهیم داشت $A \rightarrow B$ و $A \rightarrow C$

۶. اجتماع (Union) : اگر $A \rightarrow B$ و $A \rightarrow C$ برقرار باشند آنگاه : $A \rightarrow BC$

۷. ترکیب (Composition) : اگر $A \to B$ و $C \to D$ سپس خواهیم داشت $AC \to BD$
۸. شبه تعدی (Pesudotransitivity) : اگر $A \to B$ و $BC \to D$ آنگاه $AC \to D$
۹. یکسانی عمومی (General Unification) : اگر $A \to B$ و $C \to D$ آنگاه :
$A \cup (C - B) \to BD$
۱۰. اگر $A \to B$ و $AB \to C$ آنگاه $A \to C$

نکته : دو مجموعه وابستگی‌های تابعی F و G معادلند اگر F^+ مساوی با G^+ باشد.

نکته : اگر F و G دو مجموعه از FD باشند، F را پوش (Cover) G گوییم اگر هر FD در G در F^+ هم وجود داشته باشد. یعنی هر FD در G را می‌توان از FDهای موجود در F استنتاج کرد. اصطلاحاً می‌گوییم که F ، G را می‌پوشاند. بنابراین اگر سیستم مدیریت پایگاه داده‌ها بتواند مجموعه وابستگی‌های تابعی F را اعمال کند، مجموعه وابستگی‌های تابعی G نیز خودبه‌خود اعمال می‌شود.

مثال : اگر در رابطه R وابستگی‌های تابعی روبه‌رو صادق باشد، نشان دهید که وابستگی تابعی $AD \to F$ در R برقرار است.

$A \to BC \ , \ B \to E \ , \ CD \to EF$

حل :

$A \to BC \Rightarrow A \to C , A \to B \ \ (Decompostion)$
$A \to C \Rightarrow AD \to AC \ \ (Augentation)$
$AD \to AC , CD \to EF \Rightarrow AD \to EF \ \ (Transitivity)$
$AD \to F \Rightarrow AD \to F \ \ (Decompostion)$

نکته : از $A \to BC$ دو وابستگی $A \to B$ ، $A \to C$ نتیجه می‌شود ولی در حالت کلی از $AB \to C$ نمی‌توان نتیجه گرفت که $A \to C$ و یا $B \to C$

مثال : اگر در رابطه R وابستگی‌های تابعی روبه‌رو صادق باشد، نشان دهید که وابستگی تابعی $A \to C$ در R برقرار است.

$A \to B \ , \ \mathrm{A}B \to C$

حل :

$A \to B \Rightarrow A \to AB$
$A \to AB \ , \ \mathrm{AB} \to \mathrm{C} \Rightarrow A \to C$

مثال : فرض کنید $R = \{S,T,U,V,W\}$ و $F = \{S \to T, V \to SW, T \to U\}$ وابستگی‌های تابعی دیگر را بدست آورید

حل :

$S \to T, T \to U \Rightarrow \mathrm{S} \to U$
$V \to SW \Rightarrow \mathrm{V} \to S, V \to W$
$V \to S, S \to T \Rightarrow \mathrm{V} \to T$
$V \to S, S \to U \Rightarrow \mathrm{V} \to U$

پس داریم $V \to S, V \to T, V \to U, V \to W$

بستار (Closure) مجموعه‌ای از صفات

در اینجا به ارائه روشی برای محاسبه زیرمجموعه‌ای از بستار نشان می‌دهیم. یعنی زیرمجموعه‌ای که حاوی تمام FDهایی است که مجموعه مشخص‌شده S از صفات به عنوان بخش سمت چپ آنها تعیین شده است.

$S^+ := S;$

Do (S^+ did not change)

For each FD $X \to Y$ in S

if $X \subseteq Z^+$ then $Z^+ := Z^+ \cup Y$

مثال : اگر $R(A,B,C,D)$ و $F = \{A \to B, D \to AE, B \to C\}$ باشد، آنگاه الف) $\{A,D\}^+$ و ب) $\{D\}^+$ را بدست آورید.

حل :

$S^+ = \{A,D\}$

$S^+ = \{A,D\}$, $A \to B \Rightarrow S^+ = \{A,D,B\}$

$S^+ = \{A,D,B\}$, $D \to AE \Rightarrow S^+ = \{A,D,B,E\}$

$S^+ = \{A,D,B,E\}$, $B \to C \Rightarrow S^+ = \{A,B,C,D,E\}$

$\{A,D\}^+ = \{A,B,C,D,E\}$

با تکرار حلقه دیگر Z^+ تغییری پیدا نمی‌کند.

$S^+ = \{D\}$

$S^+ = \{D\}$, $A \to B \Rightarrow S^+ = \{D\}$

$S^+ = \{D\}$, $D \to AE \Rightarrow S^+ = \{A,D,E\}$

$S^+ = \{A,D,E\}$, $B \to C \Rightarrow S^+ = \{A,D,E\}$

حلقه را دوباره اجرا می‌کنیم:

$S^+ = \{A,D,E\}$, $A \to B \Rightarrow S^+ = \{A,B,D,E\}$

$S^+ = \{A,B,D,E\}$, $D \to AE \Rightarrow S^+ = \{A,B,D,E\}$

$S^+ = \{A,B,D,E\}$, $B \to C \Rightarrow S^+ = \{A,B,C,D,E\}$

در نتیجه خواهیم داشت : $\{D\}^+ = \{A,B,C,D,E\}$

مثال : در رابطه $R(A,B,C,D)$ داریم :

$A \to D \ , \ CD \to B \ , \ AD \to C$

می‌خواهیم $\{A,D\}^+$ پیدا کنیم.

حل :

$S^+ = \{A,D\}$

$S^+ = \{A,D\} \ , \ \mathrm{A} \to D \Rightarrow S^+ = \{A,D\}$

$S^+ = \{A,D\} \ , \ CD \to B \Rightarrow S^+ = \{A,D\}$

$S^+ = \{A,D\} \ , \ AD \to C \Rightarrow S^+ = \{A,D,C\}$

$S^+ = \{A,D,C\} \ , \ \mathrm{A} \to D \Rightarrow S^+ = \{A,D,C\}$

$S^+ = \{A,D,C\} \ , \ CD \to B \Rightarrow S^+ = \{A,D,C,B\}$

$S^+ = \{A,D,C,B\} \ , \ AD \to C \Rightarrow S^+ = \{A,D,C,B\}$

در نتیجه خواهیم داشت : $\{A,D\}^+ = \{A,D,C,B\}$

الگوریتم یافتن سوپرکلید : برای اینکه صفت A سوپر کلید رابطه R باشد، باید $A \to R(H)$ که در آن H مجموعه صفات رابطه R است. بنابراین باید مجموعه صفاتی را پیدا کرد که با صفت A وابستگی تابعی دارد. این مجموعه صفات را با A^+ نشان می‌دهیم و آنرا بستار A می‌نامیدیم. اگر A^+ همان H باشد، در این‌صورت A سوپرکلید است.

مثال : فرض کنید ما دارای رابطه R با صفات A ، B ، C ، D ، E ، F و FDهای زیر هستیم

$A \to BC$

$E \to CF$

$B \to E$

$CD \to EF$

مطلوبست پیدا نمودن $\{A,B\}^+$

مثال : در اینجا مجموعه‌ای از FDها برای رابطه $R = (A,B,C,D,E,F,G)$ ارائه شده است

$A \to B$

$BC \to DE$

$AEF \to G$

مطلوبست پیدا نمودن $\{A,C\}^+$

الگوریتم حذف وابستگی تابعی افزونه

با فرض وجود یک مجموعه از وابستگی‌های تابعی F در رابطه R، لازم است که وابستگی‌های تابعی افزونه (یعنی قابل استنتاج از وابستگی‌های تابعی دیگر) حذف شوند.

الگوریتم حذف چنین است:

1) choose a FD, say $X \to Y$, and remove if from F
2) RESULT := X
   while (RESULT changes and Y is not in RESULT) do
     for each FD , $A \to B$, remaining in F , if A is a subset of RESULT
       Then RESULT := RESULT Union B
   end
3) if Y is a subset of RESULT , then FD $X \to Y$ is redendent

بنابراین می‌توان $X \to Y$ را از F خارج کرد زیرا می‌توان آن‌را از FDهای دیگر استنتاج کرد. با اجـرای ایـن الگـوریتم برای هر وابستگی تابعی در F، می‌توان به مجموعه وابستگی‌های تابعی ناافزونه از مجموعه F رسید.

مثال : در رابطه R مجموعه FDهای زیر مفروضند :

$$F = \begin{cases} 1)\ A \to D \\ 2)\ A \to C \\ 3)\ CD \to B \\ 4)\ AD \to C \end{cases}$$

ابتدا $A \to D$ را تست می‌کنیم . قرار می‌دهیم RESULT := A
حال بررسی می‌کنیم که آیا FD دیگری وجود دارد که سمت چپ آن A باشـد. وابسـتگی $A \to C$ چنـین است. پس قرار می‌دهیم RESULT := A ∪ C
دوباره بررسی می‌کنیم که آیا FD دیگری وجود دارد که سمت چـپ ان زیـر مجموعـه‌ای از (A,C) باشـد؟ چنین وابستگی وجود ندارد. پس می‌بینیم که D در RESULT نیست، بنـابراین وابسـتگی $A \to D$ افزونـه نیست.

حال وابستگی $A \to C$ تست می‌کنیم . قرار می‌دهیم RESULT := A
چون یک وابستگی وجود دارد که سمت چپ آن زیر مجموعه‌ای از A (یا خـود A) اسـت، یعنـی $A \to D$ پس RESULT := A ∪ D
حال می‌بینیم آیا یـک وابسـتگی تـابعی وجـود دارد کـه سـمت چـپ آن زیرمجموعـه‌ای از (A,D) باشـد؟ وابستگی (۴) چنین است، پس : RESULT := (A, D) ∪ C
چون C زیر مجموعه‌ای از (A,D,C) است، پس $A \to C$ افزونه است و می‌توان آن‌را حذف کرد.

مثال : در هر یک از دو مجموعه FDهای زیر، FDهای افزونه را حذف کنید.

$$a = \begin{cases} B \to D \\ E \to C \\ AC \to D \\ CD \to A \\ BE \to A \end{cases}$$

$$b = \begin{cases} A \to CDE \\ B \to CE \\ AD \to E \\ CD \to F \\ BD \to A \\ CED \to ABD \end{cases}$$

مجموعه کاهش‌ناپذیر (کهینه) وابستگی‌های تابعی

با اعمال قواعد آرمسترانگ وابستگی‌های زیادی به دست می‌آیند که تعدادی از آنها اضافی و تکراری هستند. در زیر روشی را برای حذف این‌گونه وابستگی‌های زائد و رسیدن به مجموعه وابسته کاهش‌ناپذیر مطرح می‌کنیم.

مجموعه‌ای از وابستگی‌های تابعی رابطه R به نام F را کاهش‌ناپذیر گوییم اگر :

۱. در F وابستگی تابعی افزونه نباشد (اگر یک FD از F حذف شود و F^+ تغییر نکند، F کاهش‌پذیر است)
۲. در سمت راست هر FD از F فقط یک صفت وجود داشته باشد
۳. هیچ صفتی در سمت چپ FDهای F افزونه نباشد (سمت چپ هر FD، در صورت مرکب بودن، کاهش ناپذیرباشد). یعنی اگر در F داشته باشیم $X \to Y$ ، هیچ زیرمجموعه‌ای از X، مثلاً Z وجود نداشته باشد بنحوی که $Z \to Y$ را بتوان جایگزین $X \to Y$ کرد.

مثال : فرض کنید رابطه R با صفات A ، B ، C ، D و FDهای زیر ارئه گردد :

$A \to BC$
$B \to C$
$A \to B$
$AB \to C$
$AC \to D$

اکنون ما مجموعه غیر قابل کاهش FDها را که هم ارز با این مجموعه است، محاسبه می‌کنیم

۱- حال ما مجموعه‌ای از FDهای که هر کدام دارای یک عبارت سمت راست تنها می‌باشند را ارائه می‌کنیم

$A \to B$
$A \to C$
$B \to C$
$A \to B$
$AB \to C$
$AC \to D$

بلافاصله مشاهده می‌شود که وابستگی تابعی $A \to B$ دو مرتبه اتفاق می‌افتد. بنابراین می‌توان یکی را حذف کرد

۲- سپس، صفت C می‌تواند از سمت چپ وابستگی تابعی $AC \to D$ حذف گردد. دلیل این مسئله این است که داریم $A \to C$ و لذا بر اساس بسط‌پذیری $A \to AC$ خواهد بود و چنانچه $AC \to D$ باشد، آنگاه بر اساس اصل تعدی خواهیم داشت $A \to D$ و بنابراین، C در سمت چپ $AC \to D$ دارای افزونگی است

۳- سپس مشاهده می‌شود که وابستگی تابعی $AB \to C$ می‌تواند حذف گردد زیرا مجدداً ما دارای $A \to C$ هستیم و در نتیجه $AB \to CB$ برقرار می‌باشد (بر اساس بسط‌پذیری) و $AB \to C$ نیز برقرار خواهد بود (بر اساس تجزیه)

۴- بالاخره، وابستگی تابعی $A \to C$ توسط وابستگی‌های تابعی $A \to B$ و $B \to C$ ارائه می‌شود و لذا می‌تواند حذف گردد، در این‌صورت داریم

$A \to B$
$B \to C$
$A \to D$

این مجموعه غیر قابل کاهش است

مثال : در رابطه R با FDهای مفروض ارائه شده در زیر، مجموعه غیرقابل کاهش وابستگی‌های تابعی را بدست آورید.

$R = \{u, v, w, x, y\}$ $F = \{u \to xy, x \to y, xy \to zv\}$

حل :

$u \to xy, xy \to zv \Rightarrow u \to zv \Rightarrow u \to z, u \to v$

$u \to xy \Rightarrow u \to x, u \to y$

$x \to y, xy \to zv \Rightarrow x \to zv \Rightarrow x \to z, x \to v$

پس

$F = \{u \to x, u \to y, x \to y, x \to z, x \to v, u \to z, u \to v\}$

توجه کنید با توجه به خاصیت انتقال $A \to C$ و $A \to C$ و $A \to C$ بدست می‌آیند، پس وابستگی بهینه برابر خواهد بود با $F = \{u \to x, x \to y, x \to z, x \to v\}$

مثال : در اینجا مجموعه‌ای از وابستگی‌های تابعی برای رابطه R(A,B,C,D,E,F) مطرح شده است :

$AB \to C$
$C \to A$
$BC \to D$
$ACD \to B$
$BE \to C$
$CE \to FA$
$CF \to BD$
$D \to EF$

مجموعه غیرقابل کاهش وابستگی‌ها را برای این مجموعه FD پیدا کنید

حل : اولین مرحله بازنویسی مجموعه فوق است به نحوی که هر وابستگی تابعی دارای یک کمیت منفرد در سمت راست باشد

1 – $AB \to C$
2 – $C \to A$
3 – $BC \to D$
4 – $ACD \to B$
5 – $BE \to C$
6 – $CE \to A$
7 – $CE \to F$
8 – $CF \to B$
9 – $CF \to D$
10 – $D \to E$
11 – $D \to F$

اکنون :

- مورد ۲، عبارت ۶ را دلالت می‌کند و لذا می‌توانیم عبارت ۶ را حذف کنیم

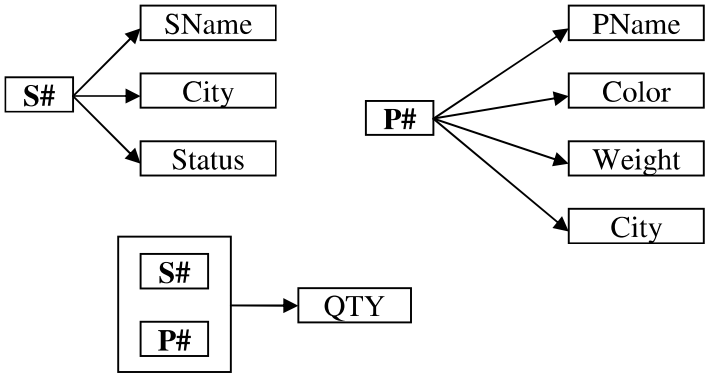
- عبارت ۸، را می‌توان توسط بسط به‌صورت $CF \rightarrow BC$ نوشت و با توجه به رابطه ۳ می‌توان با استفاده از مفهوم تعدی یا تراگذاری نوشت: $CF \rightarrow D$ بنابراین عبارت ۹ می‌تواند حذف گردد
- عبارت ۸ را می‌توان با استفاده از بسط به‌صورت $ACF \rightarrow AB$ بازنویسی نمود و همچنین عبارت ۱۱ را می‌توان به‌صورت $ACD \rightarrow ACF$ نوشت با توجه به مفهوم تعدی $ACD \rightarrow AB$ و درنتیجه $ACD \rightarrow B$ است و درنتیجه می‌توان عبارت ۴ را حذف کرد

در اینجا دیگر امکان کاهش‌پذیری دیگر وجود نداشته و بنابراین مجموعه‌های غیرقابل کاهش برابر عبارات زیر هستند

$AB \rightarrow C$
$C \rightarrow A$
$BC \rightarrow D$
$BE \rightarrow C$
$CE \rightarrow F$
$CF \rightarrow B$
$D \rightarrow E$
$D \rightarrow F$

نمودار وابستگی‌های تابعی (FD Diagram)

نموداری است که وابستگی‌های تابعی یک رابطه را نشان می‌دهد. در این نمودار صفات در داخل مستطیل‌ها قرار می‌گیرند و پیکان بین آنها نشان دهنده وابستگی تابعی صفات می‌باشد



مثال : در یک رابطه وابستگی‌ها به‌صورت زیر است :

$A \rightarrow BC$
$A \rightarrow D$
$A \rightarrow K$
$K \rightarrow C$
$B \rightarrow D$
$BC \rightarrow D$

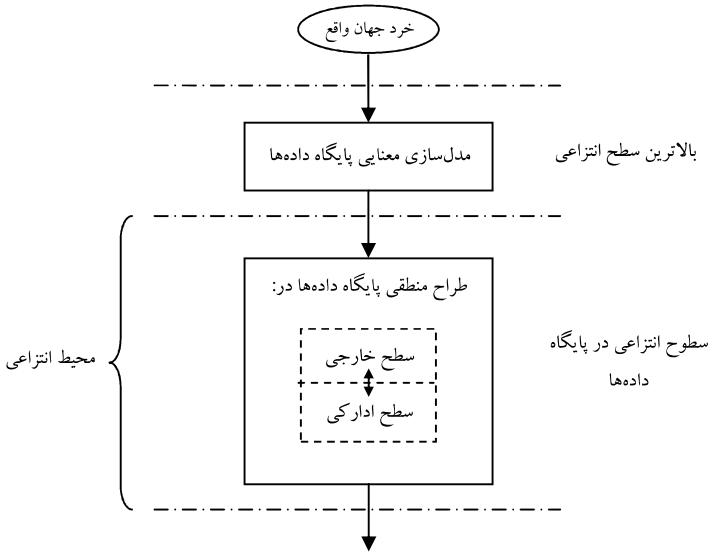
نمودار وابستگی را ترسیم کنید. سپس مجموعه کهینه این وابستگی‌ها را بدست آورده و نمودار آن را ترسیم کنید.

ساختار داده‌ای

در درس ساختمان داده‌ها با گونه‌هایی از ساختارهای داده‌ای آشنا شدیم. اما در حیطه پایگاه داده‌ها، مفهوم ساختار داده‌ای وضعی دیگر و گونه‌های خاص خود را دارد.

در پایگاه داده‌های منظور از محیط انتزاعی، محیطی فراتر از محیط فایلینگ منطقی و فیزیکی است و طبیعتاً مباحثی که در این نوع محیط مطرح می‌شوند باید از جنبه‌های فایلینگ پایگاه داده مستقل و اساساً انتزاعی باشند. خود همین محیط انتزاعی می‌تواند شامل سطوح مختلفی باشد. مدل‌سازی داده‌ها که در بالا به آن پرداختیم در واقع امکانی برای نمایش داده‌هایی بود که باید در پایگاه داده‌ها ذخیره شوند البته در بالاترین سطح انتزاعی.

محیط انتزاعی شامل سطوح پایین‌تر از مدل‌سازی داده‌ها است که همان سطح ادارکی منطقی پایگاه داده‌هاست. که خود این سطوح پایین‌تر را می‌توان به دو سطح خارجی و سطح ادارکی تقسیم نمود.



سطوح محیط انتزاعی

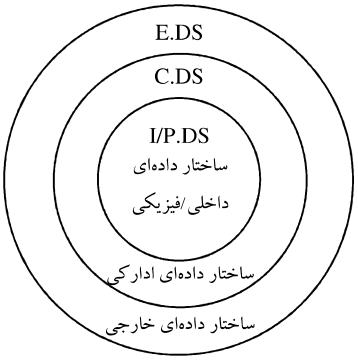
به کمک ساختار داده‌ای، مهمترین سطح بانک اطلاعاتی یعنی سطح ادارکی در حالت انتزاعی تعریف و تشریح می‌گردد.

در عمل انواع مختلف موجود ساختار داده‌ای عبارتند از :

- ساختار داده‌ای رابطه‌ای (Relation Data Structure – RDS)
- ساختار داده‌ای سلسله مراتبی (Hierarchycal Data Structure – HDS)
- ساختار داده‌ای شبکه‌ای (Network Data Structure – NDS)

بر اساس این ساختارها به ویژه ساختار رابطه‌ای، سیستم‌های مدیریت پایگاه داده‌ها طراحی و تولید می‌شود.

نکته : با توجه به سطوح مختلف تعریف پایگاه داده‌ها، باید گفت که پایگاه داده‌ها در هر سطح، مبتنی بر یک ساختار داده‌ای خاص همان سطح است. (هرچند که ساختار داده‌ای در دو سطح خارجی و ادارکی یکسان است)

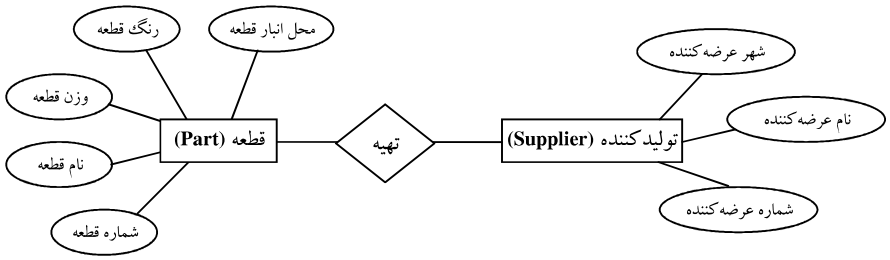


ساختار داده‌ای رابطه‌ای :

در حال حاضر متداول‌ترین مدل پایگاه داده‌ها، مدل رابطه‌ای می‌باشد. سیستم‌های مدیریت پایگاه داده‌ها که بر مبتنی بر مدل رابطه‌ای داده‌ها هستند نقش برجسته‌ای را در بازار نرم‌افزار بازی می‌کنند. برآوردها حاکی از آن است که ۸۰ درصد سیستم‌های مدیریت پایگاه داده‌ای فروخته شده مبتنی بر مدل رابطه‌ای می‌باشد. این نشانگر آن است که سیستم‌های پایگاه‌های داده‌ای که به‌صورت رابطه‌ای سازماندهی می‌شوند جای دو مدل سنتی به نام مدل سلسه مراتبی و مدل شبکه‌ای را گرفته است.

یک پایگاه داده‌ای رابطه‌ای شامل مجموعه‌ای از جداول که به هر کدام یک‌نام منحصربفرد اختصاص داده می‌شود، می‌باشد. جدول ساختاری است دارای نام که از تعدادی سطر و ستون تشکیل شده است. هر ستون نشان‌دهنده یک صفت خاص (فیلد) از یک نوع موجودیت و هر سطر نمایانگر یک نمونه از یک موجودیت (رکورد) می‌باشد.

مثال : موجودیت‌های قطعه (Part) و تولیدکننده (Supplier) را درنظر بگیرید



برای تشریح نمودار فوق در پایگاه رابطه‌ای، یک جدول برای هر موجودیت و یک جدول برای رابطه بین آنها درنظر می‌گیریم.

S :

S#	SName	City
S1	Smith	London
S2	Jones	Paris
S3	Blake	Paris
S4	Clark	London
S5	Adams	Athens

SP :

S#	P#	QTY
S1	P1	300
S1	P2	200
S1	P3	400
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	300
S4	P5	400

P :

P#	PName	Color	Weight	City
P1	Nut	Red	12	London
P2	Bolt	Green	17	Paris
P3	Screw	Blue	17	Rome
P4	Screw	Red	14	London
P5	Cam	Blue	12	Paris
P6	Cog	Red	19	London

همانگونه که مشاهده می‌کنید در مدل رابطه‌ای برای نمایش روابط بین موجودیت‌ها نیز همانند نمایش موجودیت‌ها از جدول استفاده می‌کنیم که این امر یکی از مزایای این مدل نسبت به دیگر مدل‌ها می‌باشد.

عملیات در پایگاه داده‌ای جدولی :

در صورتی که بخواهیم بر روی پایگاه جدولی خود (در محیط انتزاعی) عملیاتی انجام دهیم، این کار را با چهار عمل اصلی زیر انجام خواهیم داد :

- INSERT : دستور درج سطر(ها) در یک جدول (سطر کامل یا ناقص : با تمام ستون‌ها یا با بعضی از ستون-ها)

```
Insert
Into S
Values (S6,David,Arizona)
```

- DELETE : دستور حذف سطر(ها) از یک جدول

```
Delete
From P
Where City='London'

Delete
From SP
Where P#=P2
```

- UPDATE : دستور بهنگام‌سازی ستون(ها) از سطر یا سطرهایی از جدول

```
Update S
Set City='Darwin'
Where City='Paris'
```

- SELECT : دستور بازیابی سطر(ها) از جدول(ها)

```
Select S#
From SP
Where P#=P3
```

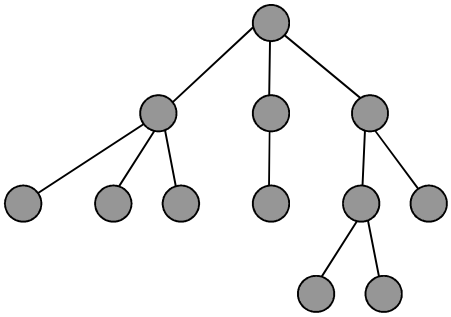
ویژگیهای ساختار داده‌ای جدولی:

- ✓ از نظر کاربر، نمایش ساده‌ای دارد
- ✓ محیطش مسطح است (شبیه یک فایل ترتیبی)
- ✓ همه چیز شامل داده‌ها و ارتباط بین آنها را با سطرها نمایش می‌دهیم
- ✓ منطق بازیابی آن ساده است.
- ✓ ارتباطات با کاردینالتی‌های متفاوت (یک به یک – یک به چند و چند به چند) را می‌توان با آن نمایش داد

- ✓ برای پرسش‌های قرینه، رویه پاسخگوی قرینه دارد

ساختار داده‌ای سلسله مراتبی :

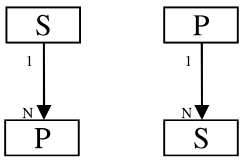
این ساختار، قدیمی‌ترین ساختار داده‌ای برای طراحی منطقی پایگاه داده‌ها در سطح انتزاعی می‌باشد.
در این ساختار، داده‌ها و ارتباط بین آنها به کمک یک درخت نمایش داده می‌شود. در این درخت، هر گره به عنوان یک نوع موجودیت و ارتباط بین گره‌ها در واقع ارتباط بین موجودیت‌ها می‌باشد.



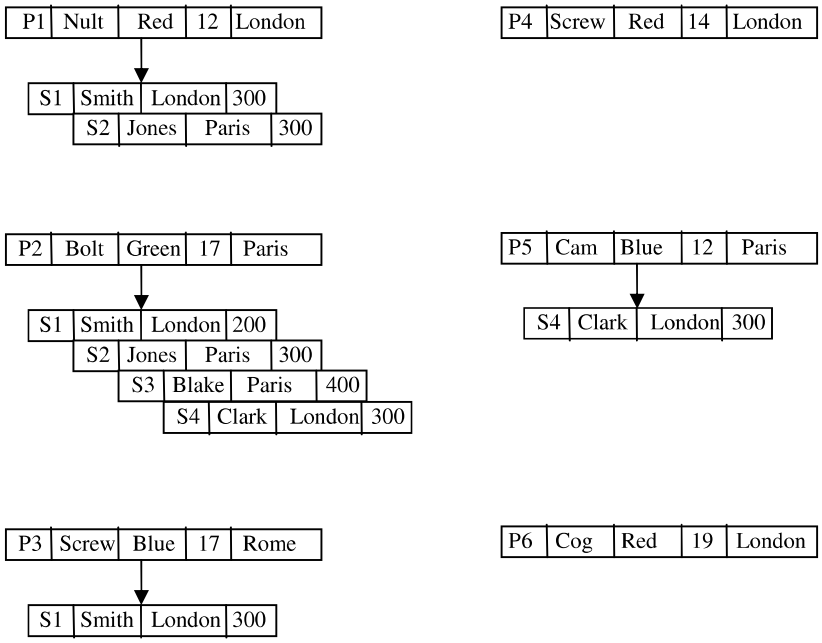
ویژگی‌های این ساختار عبارتند از :

- ❑ هر نوع رکورد (موجودیت) می‌تواند از صفر تا n نوع رکورد فرزند (درسطح پایین‌تر) داشته باشد. در واقع ساختار سلسله مراتبی برای مدلینگ ارتباطات یک به چند یکسویه بین انواع موجودیت‌ها مناسب است. همچنین در این مدل رابطه‌ها از بالا به پایین، یک-به-یک ویا یک-به-چند است. در این ساختار با حذف گره پدر، گره‌های فرزند نیز حذف می‌شوند.
- ❑ هر نوع رکورد فرزند از یک سطح خود می‌تواند پدر انواعی از رکوردها در سطح بلافاصله پایین‌تر و در چند مسیر باشد. بدینسان، سلسله مراتبی از انواع رکوردها در سطوح و مسیرهای مختلف ایجاد می‌شود.
- ❑ هر نوع رکورد فرزند، تنها یک نوع رکورد پدر دارد.

به عنوان مثال، موجودیت‌های قطعه و تولیدکننده را درنظر بگیرید. بعلت اینکه بین این دونوع موجودیت یک ارتباط چند-به-چند وجود دارد، برای نمایش این روابط با ساختار سلسله مراتبی باید آن‌را به دو رابطه یک-به-چند تبدیل کرد تا قابل نمایش شود.



حال یکی از روابط را نمایش می‌دهیم.



بررسی رفتار مدل سلسله مراتبی :

۱. عمل بازیابی : برای عمل بازیابی در این ساختار، ابتدا باید ریشه را پیدا نمود و سپس بر روی فرزندان مربوط به آن عمل جستجو را انجام داد.

به عنوان مثال، در صورتیکه بخواهیم بدانیم که چه تهیه‌کننده‌گانی قطعه P3 را تولید می‌کنند، باید ابتدا ریشه مربوط به قطعه P3 را پیدا نموده و سپس فرزاندان آن را پیمایش کنیم. ولی در صورتیکه بخواهیم پرس‌وجوی قرینه را انجام دهیم، یعنی اینکه تولیدکننده S3 چه قطعاتی را تولید می‌کنند، باید تمام پایگاه را جستجو کنیم که ببینیم کدام قطعات توسط تولیدکننده S3 تولید می‌شوند و یا اینکه رابطه‌ای را که در آن تولیدکننده‌گان به عنوان ریشه و قطعات به عنوان فرزندان است، را به پایگاه اضافه کنیم، ولی این امر موجب افزونگی داده‌ها در پایگاه می‌گردد.

۲. عمل درج : با توجه به ماهیت این ساختار، محدودیت‌هایی در عملیات ذخیره‌سازی وجود دارد: درج نمونه فرزند بدون مشخص بودن پدر آن ناممکن است. به عنوان نمونه، در صورتیکه بخواهیم اطلاعات یک تولیدکننده جدید را اضافه کنیم، تا هنگامی که ندانیم چه قطعه‌ای را تولید می‌کند، امکان‌پذیر نخواهد بود.

۳. عمل حذف : در این ساختار، حذف نمونه پدر منجر به حذف نمونه فرزندان آن می‌شود. به عنوان نمونه، در صورتیکه بخواهیم قطعه P3 را از پایگاه داده‌ها حذف کنیم، بطور ناخواسته موجب حذف اطلاعات مربوط به تولیدکننده‌گانی قطعه مربوطه، خواهد شد.(البته این امر درصورتیکه است که اطلاعات آن تولیدکننده در فرزندان سایر قطعات موجود نباشد یعنی تولید کننده فقط قطعه P3 را تولید کند)

۴. عمل بهنگام‌سازی : در این عمل ممکن است فزونکاری در سیستم پیش آید، زیرا ممکن است که بهنگام-سازی منتشر شوند لازم باشد. به عنوان نمونه، اگر بخواهیم شهر تولیدکننده S1 را تغییر دهیم، بعلت اینکه امکان تکرار اطلاعات این تولیدکننده در چندین قطعه وجود داشته باشد، مجبور خواهیم بود عمل بهنگام-سازی را در تمام نمونه‌های آن انجام دهیم که طبعاً حجم عملیات را افزایش می‌دهد.

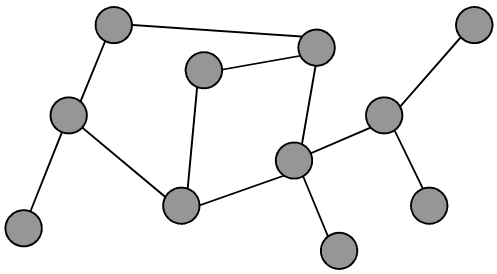
نکته: این مدل در عملیات ذخیره‌سازی (یعنی در سه عمل حذف، درج و بهنگام‌سازی) دارای آنومالی (Anomaly) است. آنومالی یعنی وجود دشواری در انجام یک عمل خاص و یا عدم امکان انجام عمل و یا بروز عوارض نامطلوب در پی انجام یک عمل.

ویژگیهای ساختار داده‌ای سلسله مراتبی :

- ✓ از نظر کاربر به اندازه نمایش جدولی، سادگی ندارد
- ✓ به علت ساختار درختی آن، خاص محیطهایی است که در آنها ارتباط‌های یک-به-چند یک طرفه وجود دارد. نمایش ارتباط چند-به-چند در آن دشوار است
- ✓ دستور بازیابی آن سادگی دستور بازیابی در ساختار جدولی را ندارد
- ✓ برای پاسخ‌گویی به پرسش‌های قرینه، رویه‌های پاسخ‌گویی قرینه ندارد

ساختار داده‌ای شبکه‌ای :

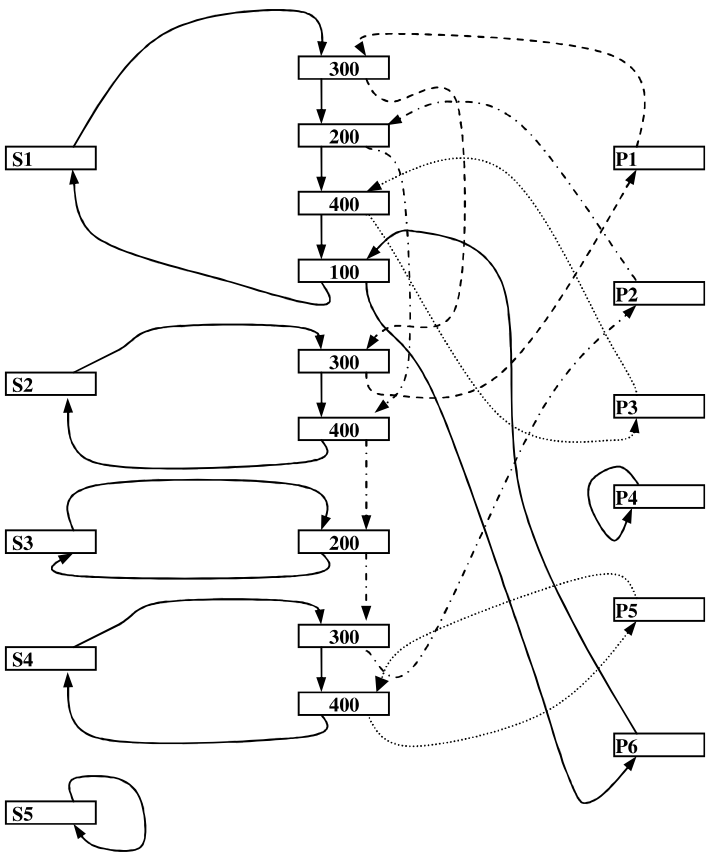
این ساختار که گاهی به آن ساختار پلکس (Plex) نیز گفته می‌شود، را می‌توان شکل توسعه یافته ساختار داده‌ای سلسله مراتبی در نظر گرفت. تفاوت اصلی بین این دو عبارتست از: در ساختار سلسله مراتبی، یک رکورد فرزند دارای یک پدر است در حالی‌که در سیستم شبکه‌ای، یک فرزند می‌تواند چندین پدر داشته باشد (و یا احتمالاً هیچی).



بدلیل ساختار گرافی مدل شبکه‌ای، برای نمایش ارتباطات دوسویه از این مدل استفاده می‌گردد.
به عنوان مثال،

SP :

S#	P#	QTY
S1	P1	300
S1	P2	200
S1	P3	400
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	300
S4	P5	400



بررسی رفتار مدل شبکه‌ای :

۱. عمل بازیابی : عمل بازیابی در این مدل پیچیده‌تر از مدل سلسله مراتبی است. به عنوان نمونه، برای پیدا کردن تولیدکنندگانی که قطعه P2 را تولید کرده‌اند، ابتدا به سراغ گره P2 رفته و سپس از طریق فلش‌ها به رکوردهای واسط (پیوند دهنده) 200 و 400 و 200 و 300 می‌رسیم، هنگامی که به هر کدام از رکوردهای واسط رسیدیم، باید آنها را از طریق فلش‌های دیگر که مربوط به حلقه تولیدکننده‌گان است را دنبال کنیم تا به مشخصات تولید کننده دست پیدا کنیم. در این ساختار پرس‌وجوی قرینه نیز با حفظ همین ساختار قابل پاسخ‌گویی است. مثلاً، برای یافتن اینکه تولیدکننده S3 چه قطعاتی را تولید می‌کند باید روال مشابهی همانند بالا را دنبال کنیم.

۲. عمل درج : در این ساختار بر خلاف ساختار سلسله مراتبی، این امکان که تولیدکننده‌ای را بدون اینکه بدانیم چه قطعه‌ای را تولید می‌کند، وجود دارد.

۳. عمل حذف : در این ساختار بر خلاف ساختار سلسله مراتبی، عمل حذف یک نمونه موجب حذف ناخواسته دیگر اطلاعات نخواهد شد.

۴. عمل بهنگام‌سازی : در این ساختار بر خلاف ساختار سلسله مراتبی، عمل بهنگام‌سازی دارای تکرار نخواهد بود.

مفاهیم اساسی مدل رابطه‌ای :

مدل رابطه در سال ۱۹۷۰ میلادی توسط کاد (E.F.CODD) ابداع و معرفی شد. در واقع کاد در جستجوی یک ساختار داده‌ای بود که دارای انتزاع قوی باشد تا محیط انتزاعی قوی‌تر و از آنجا استقلال داده‌ای بهتر تأمین شود. می‌دانیم که ساخت‌های ریاضی، اساساً انتزاعی هستند و کاد هم به سراغ ریاضیات رفت. ساختار داده‌ای در مدل رابطه‌ای یک مفهوم ریاضی به‌نام "رابطه" است که در ادامه بحث به آن خواهیم پرداخت.

در اینجا قبل از شروع به شرح مفاهیم اساسی مدل رابطه‌ای، مفهوم مدل داده‌ای را توضیح می‌دهیم. مدل داده‌ای امکانی است برای طراحی منطقی پایگاه داده‌ها، تعریف و کنترل آن و نیز انجام عملیات در آن، و امکان می‌دهد تا هر سه عمل اساسی در محیطی انتزاعی انجام شوند. بنابراین می‌توان گفت مدل داده‌ای تامین کننده محیط انتزاعی پایگاه داده‌ها است و از سه بخش اساسی تشکیل شده است :

- بخش ساختاری (Structural) : نشان دهنده عناصر ساختار مدل است که همان ساختار داده‌ای اصلی ئ مفاهیم مرتبط با آن است.
- بخش عملیاتی (پردازشی – Manipulative) : مجموعه امکاناتی است که بوسیله آنها عملیات موردنظر کاربر، از جمله بازیابی و ذخیره‌سازی انجام می‌شود
- بخش جامعیتی (Integrity) : از مجموعه قواعد و محدودیت‌های جامعیتی تشکیل شده است. با استفاده از این قوانین، سیستم مدیریت پایگاه داده‌ها می‌تواند صحت، دقت و سازگاری داده‌ها را در هر لحظه از حیات پایگاه، کنترل و تضمین کند.

با توجه به آنچه گفته شد، مفهوم مدل داده‌ای گسترده‌تر از مفهوم ساختار داده‌ای است و ساختار داده‌ای فقط بخشی از مدل داده‌ای است و نه همه آن. تأکید می‌کنیم که در واقع مدل داده‌ای است که تأمین کننده محیط انتزاعی پایگاه داده‌ها است و هر مدل داده‌ای حتماً باید سه بخش اساسی مذبور را داشته باشد وگرنه کامل و جامع نیست.

تعریف دامنه (Domain) : میدان مجموعه‌ای است نامدار از مقادیر همنوع که یک یا بیش از یک صفت از آن مقدار می‌گیرند. به عنوان مثال، دامنه شماره عرضه‌کنندگان، مجموعه تمام شماره عرضه‌کنندگان ممکن است. همچنین، دامنه مقادیر ارسال کالا فرضاً شامل مجموعه‌ای از اعداد صحیح بزرگتر از صفر و کوچکتر از 100000 می‌باشد.

در نتیجه، دامنه‌ها مجموعه مقادیری هستند که از آن، مقادیر واقعی صفات گرفته و اخذ می‌گردد. بطور دقیق‌تر می‌توان گفت که هر صفت باید براساس یک دامنه مربوطه تعریف شود، بدین مفهوم که مقادیر آن صفت باید از آن دامنه گرفته شود.

با توجه به مطالب ارائه شده، اهمیت دامنه‌ها چیست؟ پاسخ دقیق و مهم با این سوال این است که دامنه‌ها، مقایسه‌ها و محدودیت‌ها را محدود می‌کند. برای روشن شدن مطلب، دو پرس‌وجوی SQL زیر را برای بانک اطلاعاتی عرضه‌کنندگان و قطعات درنظر بگیرید

```
Select …
From S , SP
Where  P.P# = SP.P#
```

```
Select …
From S , SP
Where  P.weight = SP.QTY
```

از بین این دو پرس‌وجو، پرس‌وجوی سمت چپ مطمئناً با معنی است چراکه در پرس‌وجوی سمت چپ مقایسه‌ای بین دو صفت P.P# و SP.P# است که هر دو در دامنه یکسانی تعریف شده‌اند. ولی پرس‌وجوی سمت راست احتمالاً به‌این گونه نیست چراکه شامل مقایسه‌ای است بین دو صفت P.weight و SP.QTY که قطعاً در دامنه‌های متفاوتی تعریف شده‌اند.

بنابراین همانگونه که از بحث فوق مشخص می‌گردد، اگر دو صفت مقادیر خود را از یک دامنه واحد بگیرند، درنتیجه مقایسه‌ها (و مسلماً الحاق، اجتماع و بسیاری از اعمال دیگر) شامل آن دو صفت می‌توانند دارای مفهوم و معنی درستی باشند، چراکه عمل مقایسه بر روی دامنه‌های همسان انجام می‌گیرد.

رابطه (Relation) :

ساختاری است شامل صفات و تاپل‌ها (tuple) در یک جدول متناظر و یا بطور خلاصه می‌توات گفت یک رابطه جدولی است شامل سطرها و ستون‌ها.

اصطلاحاتی در مدل‌های رابطه‌ای وجود دارد که عبارتند از:

۱. یک رابطه که همان جدول می‌باشد مشخص کننده یک فایل است
۲. یک تاپل که همان سطر جدول می‌باشد مشخص کننده یک رکورد است
۳. یک صفت خاصه که همان ستون جدول است مشخص کننده یک فیلد است

در جدول زیر تناظری بین مفاهیم رابطه‌ای و مفاهیم جدولی نشان داده شده است

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

ویژگی‌های رابطه :

- ✓ رابطه تاپل (چندگانه) تکراری ندارد. بدلیل اینکه بدنه رابطه مجموعه است و مجموعه نمی‌تواند دارای عنصر تکرار باشد.
- ✓ تاپل‌ها نظم ندارند. بدلیل اینکه بدنه رابطه مجموعه است و مجموعه در حالت کلی فاقد نظم است. هرچند که معمولاً با یک نظم خاص نشان داده می‌شوند
- ✓ صفات رابطه نظم ندارند (ترتیب قرار گرفتن آنها در جدول نظم ندارد)
- ✓ تمام مقادیر صفات خاصه تجزیه‌نشدنی (اتمیک – Atomic) هستند. مقدار تجزیه نشدنی، مقداری است ساده که قابل تجزیه نباشد، به عبارت دیگر از یک میدان ساده گرفته شده‌باشد. مثلاً صفت آدرس تجزیه شدنی است.

کلید در مدل رابطه‌ای

در بررسی بانک‌های اطلاعاتی رابطه‌ای دانستن شناسه‌های منحصر بفرد هر تاپل از نیازهای بانک محسوب می‌شود. در مدل رابطه‌ای ،کلید مجموعه‌ای از صفت‌ها می‌باشد.

دقت داشته باشید که مقدار کلید در یک رابطه نمی‌تواند تهی (Null) باشد و درصورتی‌که برای رابطه‌ای کلید تعیین نشود، بصورت خودکار همه صفت‌های آن رابطه روی‌هم کلید محسوب می‌شوند.

انواع کلیدهای در مدل رابطه‌ای عبارتند از

- ابرکلید (Super Key – S.K) : یعنی هر ترکیبی از صفت‌ها که خاصیت کلید داشته باشد. این نوع کلید، تنها نوع کلید است که الزاماً کمینه نیست، یعنی زیر مجموعه‌ای از آن هم ممکن است کلید باشد.
- کلید کاندید (Candidate Key – C.K) : هر زیر مجموعه‌ای از مجموعه صفات رابطه که دو خاصیت زیر را داشته باشد، کلید کاندید است
  ۱- یکتایی مقدار : مقدار کلید در هر سطر از مجموعه منحصر بفرد باشد
  ۲- کاهش ناپذیری : یعنی اگر یکی از عناصر این مجموعه را حذف کنیم، زیر مجموعه باقیمانده خاصیت یکتایی مقدار نداشته باشد

  توجه داشته باشید که هر ابر کلید لزوماً کلید کاندید نیست، اما هر کلید کاندید، جزء مجموعه ابر کلیدهای رابطه هست.

  کلید کاندید می‌تواند ساده یا مرکب باشد. حال این سوال مهم مطرح می‌شود: آیا وجود حداقل یک کلید کاندید در هر ربطه‌ای تضمین است؟ پاسخ مثبت است. زیرا در بدترین حالت، با ترکیب تمام صفات رابطه به همان مجموعه عنوان می‌رسیم که طبق تعریف رابطه، در مجموعه بدنه، یکتایی مقدار دارد. به چنین رابطه‌ای، رابطه تمام کلید می‌گوییم.
- کلید اصلی (Primary Key – P.K) : رابطه‌ها می‌توانند بیش از یک کلید کاندید داشته باشند. چنانچه تنها یک کلید داشته باشد به آن کلید، کلید اصلی گفته می‌شود و چنانچه چند کلید وجود داشته باشد طراح خود بایستی یک کلید اصلی را درنظر بگیرد. در تعیین کلید اصلی از بین کلیدهای کاندید باید دو ضابطه زیر را درنظر گرفت :
  ۱- کلیدی به عنوان کلید اصلی درنظر گرفته می‌شود که نسبت به سایر کلیدها در پاسخ‌گویی به نیازهای کاربران اهمیت بیشتری داشته باشد
  ۲- کوتاه‌تر بودن طول کلید کاندید درنظر گرفته شده به عنوان کلید اصلی
- کلید خارجی (Foreign Key – F.K) : دو رابطه R1 و R2 (نه لزوماً متمایز) را درنظر می‌گیریم. هر زیر مجموعه از صفات رابطه R2 که در رابطه R1 کلید کاندید باشد، کلید خارجی در رابطه R2 است.

  نقش یک کلید خارجی چیست؟ کلید خارجی برای نمایش ارتباطات بین انواع موجودیت‌ها بکار می‌رود

قواعد جامعیت در مدل رابطه‌ای

در این قسمت به قواعد جامعیت (محدودیت‌های جامعیتی) در مدل رابطه‌ای می‌پردازیم.

تعریف : جامعیت پایگاه داده‌ها یعنی: صحت، دقت و سازگاری داده‌های ذخیره شده در پایگاه داده‌ها در تمام لحظات.

هر سیستم مدیریت پایگاه داده‌ها باید بتواند جامعیت پایگاه داده‌ها را کنترل و تضمین‌کند، زیرا همیشه ممکن است عواملی سبب نقص جامعیت شوند. برخی از عوامل عبارتند از:

- اشتباه در برنامه‌های کاربردی
- اشتباه در وارد کردن داده‌ها
- وجود افزونگی کنترل نشده
- توارد تراکنش‌ها به گونه‌ای که داده نامعتبر ایجاد شود
- خرابی‌های سخت افزاری و نرم‌افزاری

این عوامل بطور مستقیم یا غیرمستقیم شرایطی را پدید می‌آورند که نهایتاً منجز به نقض جامعیت داده‌ها می‌شوند.

انواع قواعد جامعیت :

قواعد جامعیت در مدل رابطه‌ای به دو رده کلی تقسیم می‌شوند :

- قواعد کاربری (قواعد خاص – User Defined Rule)
- متا قواعد (قواعد عام – Meta Rule)

قواعد کاربری : قواعدی هستند که توسط کاربر مجاز، تعریف می‌شوند. در واقع در یک پایگاه داده‌های خاص (مورد نظر کاربر) مطرح گردیده و عمومیت ندارند. گاه به این قواعد، قواعد محیطی یا وابسته به داده‌ها و یا محدودیت‌های جامعیت معنایی می‌گویند. مشخص کردن قواعد یا محدودیت‌های جامعیتی یکی از وظایف مدل‌ساز و طراح پایگاه داده‌ها است. بخشی از این محدودیت‌ها در مرحله مدلسازی قابل اعمال هستند و بخش دیگری باید در مرحله طراحی منطقی پایگاه داده‌ها منظور شوند.

بدیهی است که تعداد قواعد کاربری بستگی به محیطی دارد که پایگاه داده‌ها برای آن ایجاد می‌شود. توجه داشته باشیم که مجموعه قواعد کاربری یک محیط عملیاتی باید رسماً به تأیید مدیر داده‌های سازمان (DA) برسد و سپس مدیر پایگاه داده‌ها آنها را در مراحل طراحی و پیاده‌سازی پایگاه داده‌ها منظور نماید.

هرچه تعداد قواعد جامعیت کاربری بیشتر باشد، فزونکاری در سیستم برای اعمال آنها بیشتر است.

توجه داریم که اگر یک قاعده جامعیتی جدید به سیستم داده شود، سیستم باید ابتدا مطمئن شود که این قاعده با وضعیت جاری پایگاه داده‌ها همخوانی دارد. اگر چنین نباشد، سیستم باید آن قاعده را رد کند. در غیر این‌صورت قاعده پذیرفته می‌شود (در گاتالوگ سیستم پذیرفته می‌شود) و از این لحظه به بعد اعمال می‌شود.

متا قواعد : قواعدی هستند که باید توسط هر سیستم رابطه‌ای در هر پایگاه داده‌ها رابطه‌ای اعمال شوند، ناوابسته به داده‌های خاص هستند و عمومیت دارند. این قواعد که به آنها متا قواعد محدودیت نیز می‌گویند عبارتند از:

- قاعده C_1 : قاعده جامعیت موجودیتی (Entity Integrity Rule) : یعنی هیچ جزء تشکیل دهنده کلید اصلی نباید دارای هیچ مقدار باشد
- قاعده C_2 : قاعده جامعیت ارجاعی (Refrencial Integrity Rule) : اگر صفت خاصه A1 (ساده یا مرکب) در رابطه R2 کلید خارجی باشد در این‌صورت : A1 در R2 می‌تواند هیچ‌مقدار داشته باشد، یا اینکه باید حتماً مقداری داشته باشد که در رابطه مرجع (R1) وجود دارد. به‌عبارت دیگر، مقدار کلید خارجی یک رابطه نمی‌تواند در رابطه مرجع وجود نداشته باشد.

نرمال‌سازی (Normalization)

موضوع اصلی مورد توجه بسیاری از طراحان پایگاه داده‌ها این است که چه اقلام داده‌هائی به‌صورت رکورد یا تاپل‌ها گروه‌بندی گردند. روش‌های زیادی وجود دارند که بر اساس آنها صدها یا هزاران فقره داده می‌توانند گروه‌بندی شوند و بعضی از روش از روش‌های دیگر بهتر است.

بسیاری از پایگاه داده‌ها پیوسته تغییر می‌کنند. اقلام جدید، داده‌ها و روابط جدید بین داده‌ها مداوم افزوده می‌شوند و کاربردهای جدیدی شکل می‌گیرند. هنگامی که پایگاه داده‌ها را تغییر می‌دهیم باید دیدهای قدیمی کاربران نسبت به داده‌ها را حفظ نماییم تا از بازنویسی برنامه‌ها خودداری بعمل آید. با این وجو تغییرات مشخصی در روابط داده‌ها یا کاربرد آنها وجود دارند که تغییر برنامه‌ها را الزامی می‌سازند، به عنوان مثال ممکن است یک رکورد را به دو بخش تقسیم کنیم و یا کلیدی را که جهت اقلام داده‌های مشخصی مورد استفاده قرار می‌گیرد، تغییر دهیم. چنین تغییراتی فوق‌العاده مخرب هستند. اگر در گروه بندی اقلام داده‌ها و کلیدها در ابتدا خوب اندیشیده شده باشد می‌توان چنین تخریبی را غیرمحتمل ساخت.

گروه بندی نامناسب اقلام داده‌ها منجر به ظهور مشکلاتی ناشی از عدم جامعیت معنائی (Semantic Integrity) می‌گردد. واژه عدم جامعیت معنائی در مورد عملیاتی بکار می‌رود که از این طریق ترکیبات غیرمجاز اقلام داده‌ها را بوجود نمی‌آورند. برای نیل به جامعیت معنائی قواعدی مورد نیاز هستند که بر مبنای آن‌ها مشخص شود که چه عملیاتی مجاز هستند و این قواعد باید توسط روابط بین اقلام داده‌ها اداره شوند.

با نگرش به آنچه که بیان گردید، فنی به نام نرمال‌سازی از اهمیت ویژه‌ای برخوردار است. تنها رابطه‌هائی که در مدل رابطه‌ای پایگاه داده‌ها مجاز هستند آنهایی هستند که شرط زیر را تحقق می‌بخشند :

- هر مقدار در رابطه یعنی هر مقدار صفت خاصه در هر تاپل، یکتا یا غیرقابل تجزیه است

به بیانی دیگر، در محل برخورد سطر و ستون جدول رابطه دقیقاً یک مقدار وجود دارد و هرگز مجموعه‌ای از مقادیر وجود ندارد. البته وجود مقادیر تهی محتمل می‌باشد.

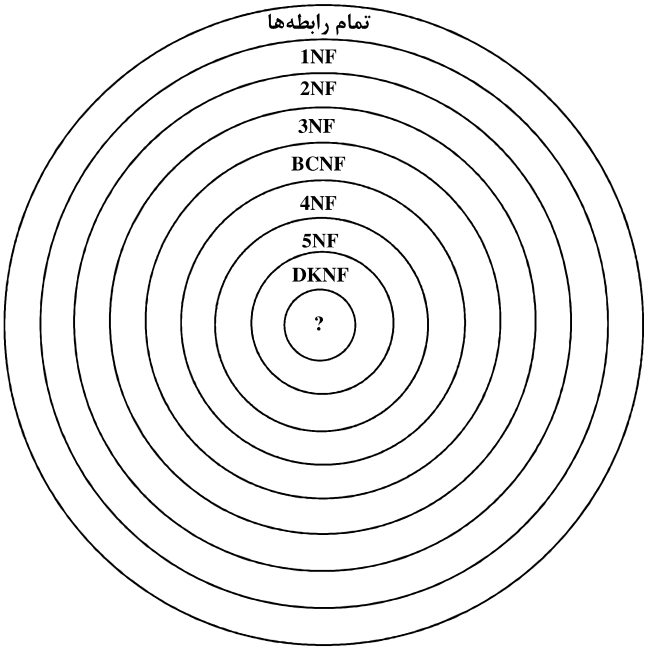
سطوح نرمال

کاد، واضع مدل رابطه‌ای، در آغاز سه صورت (سطح) نرمال تعریف کرد اما بعداً پژوهشگران صورت‌های دیگری هم تعریف نمودند.

صورت‌های نرمال عبارتند از

- اولین شکل نرمال یا 1NF (First Normal From)
- دومین شکل نرمال یا 2NF (Second Normal From)
- سومین شکل نرمال یآ 3NF (Third Normal From)
- شکل نرمال بایس-کاد یا BCNF (Boyce-Codd Normal From)
- چهارمین شکل نرمال یا 4NF (Fourth Normal From)
- پنجمین شکل نرمال یا 5NF/PJNF (Fifth Normal From – Projection-Join Normal From)
- شکل نرمال کلید-دامنه یا DKNF (Domail-Key Normal From)

در واقع صورت‌های نام‌برده هریک از قبلی خود نرمال‌تر است، اما در عمل، وضعی که در آن یک رابطه BCNF باشد اما 4NF نباشد و یا رابطه‌ای 4NF باشد اما 3NF نباشد، به‌ندرت پیش می‌آید. از این‌رو می‌گوییم که در عمل اگر رابطه را تا سطح BCNF (وحتی گاه تا 3NF) نرمال کنیم، کفایت می‌کند.



صورت‌های نرمال

در حال حاضر به درستی دانشته نیست که آیا ممکن است صورت‌های دیگری مثلاً 6NF و 7NF و ... نیز تعریف شوند.

در جمع‌بندی بحث نرمال‌سازی خواهیم دید که در نرمال‌سازی اساساً فرض بر این است که رابطه‌ای که با آن شروع می‌کنیم، اختلاط اطلاعات (و در نتیجه افزونگی و آنومالی) دارد.

اولین شکل نرمال یا 1NF

تعریف : رابطه‌ای 1NF است اگر هر صفت خاصه آن در هر تاپل، تک‌مقداری باشد به بیان دیگر، صفت چند مقداری نداشته باشد.

مثلاً اگر در جدولی شامل فیلد تاریخ باشد که خود فیلد تاریخ از سه فیلد کوچکتر (سال-ماه-روز) تشکیل شده باشد، آنگاه جدول 1NF نیست.

مثال : به جای دو جدول مجزای S و SP جدولی بنام TEST را به‌صورت زیر تعریف می‌کنیم

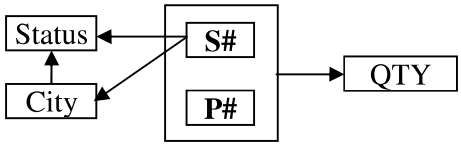
TEST (S# , Status , City , P# , QTY)

فرض کنید در طراحی بانک این قاعده وجود دارد که وضعیت یک تهیه‌کننده از روی شهر او تعیین می‌شود

TEST

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

نمودار وابستگی جدول به‌صورت زیر می‌باشد (کلید اصلی این رابطه (S#,P#) است)



با مشاهده جدول فوق و بررسی روابط مشخص می‌شود که جدول TEST دارای مشکل افزونگی اطلاعات است که این افزونگی در رابطه آنامولی‌ها زیر را ایجاد می‌نماید

- آنامولی درج : در جدول فوق نمی‌توان واقعیت "S4 در شهر C3 قرار دارد" را تا زمانی که ندانیم چه قطعه‌ای را تهیه می‌کند، درج کرد. دلیل این امر این است که P# جزو کلید اصلی است و طبق قوانین جامعیت نمی‌تواند NULL باشد
- آنامولی حذف : اگر بخواهیم واقعیت "S2 از P2 به تعداد 200 عدد تولید می‌کند" را حذف کنیم، بطور ناخواسته اطلاع "S2 در شهر C3 قرار دارد" نیز حذف خواهد شد.
- آنامولی بهنگام‌سازی : در این جدول در صورتیکه شهر یک تولید کننده را بخواهیم بهنگام‌سازی کنیم، بدلیل تکرار آن، باید این بهنگام سازی در تمامی موارد تکرار انجام شود و درصورتیکه بدرستی انجام نشود داده‌های بانک دچار ناسازگاری می‌شوند

دو واقع دلیل مشکلات مطرح شده در بالا این است که فیلدهای City و Status در رابطه بالا با کلید اصلی وابستگی تابعی کامل ندارند و تنها با S# که بخشی از کلید اصلی است وابستگی دارند. پس باید سعی گردد که این مورد را برطرف سازیم.

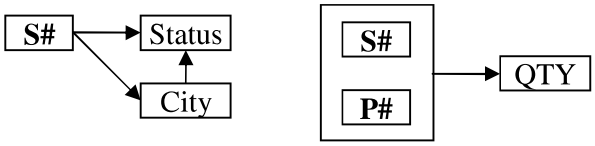
دومین شکل نرمال یا 2NF

تعریف : رابطه‌ای 2NF است اگر اولاً 1NF باشد و ثانیاً تمام صفات غیرکلید (کلید نباشد و یا جزء تشکیل دهنده کلید هم نباشد) با کلید اصلی وابستگی تابعی کامل (تام) داشته باشند. به عبارت دیگر هر صفت غیر کلید با کلید اصلی بطور کاهش‌ناپذیر وابسته باشد.

در مثال قبل، رابطه TEST یک رابطه 2NF نیست چراکه صفات غیرکلید City و Status با کلید اصلی (S#,P#) وابستگی تابعی کامل ندارند.
رابطه TEST در مثال قبل را به دو رابطه زیر تجزیه می‌کنیم

$$TEST(S\#, Status, City, P\#, QTY) \begin{cases} Second(S\#, Status, City) \\ SP(S\#, P\#, QTY) \end{cases}$$

در اینجا رابطه‌های Second و SP هر دو 2NF هستند و افزونگی اطلاعات به این طریق کاهش می‌یابد



Second **SP**

با توجه به تجزیه رابطه، جداول زیر را برای روابط جدید خواهیم داشت

Second :

S#	Status	City
S1	20	C2
S1	20	C2
S1	20	C2
S2	10	C3
S2	10	C3

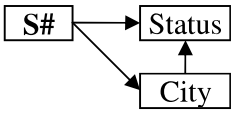
SP :

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

جداول فوق مشکلات رابطه TEST را ندارند. مثلا می‌توان این واقعیت را که "S4 در شهر C3 قرار دارد" در بانک درج کرد بدون انکه نیازی باشد که بدانیم چه قطعه‌ای را تولید می‌کند. همچنین می‌توان اطلاع "S2 از P2 به تعداد 200 عدد تولید می‌کند" را بدون آنکه اطلاعات دیگری بطور ناخواسته از بین روند را حذف نمود و به‌طور مشابه در عمل بهنگام‌سازی، می‌توان شهر هر تولید کننده را به آسانی به‌روز رسانی نمود چراکه این صفت در رابطه Second تنها یک بار تکرار شده است. که این مشکلات را با حذف وابستگی تابعی غیرکامل و تبدیل به وابستگی کامل از بین رفته‌اند.

اما رابطه Second هنوز هم آنامولی‌هائی دارد. به عنوان نمونه در عمل درج نمی‌توان اطلاع " شهر C4 دارای وضعیت 30 است" را تا هنگامی که ندانیم که در شهر C4 چه تهیه‌کننده‌ای وجود دارد، را درج نمود. در عمل حذف، نیز اگر تولیدکننده‌ای را حذف کنیم موجب حذف این اطلاع که شهری مربوط به تولیدکننده حذف شونده دارای چه وضعیتی است، می‌گردد. در عمل بهنگام‌سازی نیز در صورتیکه بخواهیم وضعیت یک شهر مشخص را بهنگام‌سازی کنیم، بعلت امکان تکرار چندین بار از شهر موردنظر در جدول، باید عمل بهنگام‌سازی بر روی تمام نمونه‌های تکرار شده شهر در جدول انجام گیرد.

علت بوجود آمدن این‌چنین مشکلاتی، کلید اصلی (S#) با فیلد Status هم به‌صورت مستقیم وابستگی دارد و هم از طریق یک واسطه (تراگذاری-تعدی) یعنی صفت City وابستگی دارد



در نتیجه برای رفع این‌چنین مشکلاتی، باید این نوع وابستگی‌ها (یعنی اینکه دو فیلد غیرکلید با یدکدیگر وابستگی داشته باشند) را برطرف کنیم.

سومین شکل نرمال یا 3NF

بی‌نظمی‌هائی نظیر آنچه بیآن گردید گاهی در رابطه‌ای که در دومین شکل نرمال است، بروز می‌کند. برای حذف آنها گام دیگری در راستای نرمال‌تر سازی برداشته می‌شود که دومین شکل نرمال را به سومین شکل نرمال تبدیل می‌کند.

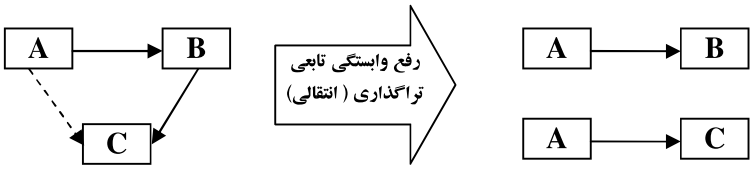
تعریف وابستگی انتقالی (با واسطه – Transitive) : در رابطه R(A,B,C,…) اگر

$$A \to B, B \to C, B \not\to A$$

می‌گوییم که C با واسطه B با A وابستگی دارد

تعریف : رابطه‌ای 3NF است اگر 2NF باشد و هر صفت غیرکلید با کلیداصلی وابستگی تابعی بی‌واسطه داشته باشد.

در مثال قبل، بعلت اینکه در رابطه Second صفت Status با کلید اصلی S# از طریق واسطه City وابستگی انتقالی دارد، بنابراین در فرم 3NF نمی‌باشد، برای تبدیل این رابطه به 3NF باید این وابستگی برطرف گردد.
در راستای تبدیل‌شدن به سومین شکل نرمال این وابستگی انتقالی از طریق شکافتن رابطه و تقسیم‌شدن آن به دو رابطه با وابستگی‌های تابعی زیر از بین می‌رود



در مورد مثال، رابطه‌های به‌شکل زیر تبدیل می‌شوند

S#	Status
S1	20
S1	20
S1	20
S2	10
S2	10

S#	City
S1	C2
S1	C2
S1	C2
S2	C3
S2	C3

با تبدیل به شکل نرمال 3NF دیگر آنومالی‌های مطرح شده در مورد رابطه Second از بین می‌رود.به عنوان نمونه در این حالت می‌توان اطلاع " شهر C4 دارای وضعیت 30 است" را به راحتی بدون نیاز به اینکه بدانیم در شهر C4 چه تهیه‌کننده وجود دارد، درج نمود و یا در عمل حذف، در صورتیکه بخواهیم تولیدکننده‌ای را حذف کنیم، دیگر مشخصات مربوط به شهری که تولیدکننده موردنظر ما در آن قرار دارد از بین نخواهد رفت. در عمل بهنگام‌سازی نیز در صورتیکه بخواهیم وضعیت یک شهر مشخص را بهنگام‌سازی کنیم، دیگر لازم نیست که عمل بهنگام‌سازی تکرار شود، چرا که مشخصات شهر فقط یکبار تکرار شده‌است.

تجزیه مطلوب

در تجزیه یک رابطه به دو رابطه، ممکن است مشکلاتی بروز کند از جمله :

۳- عدم امکان بدست آوردن رابطه اولیه با پیوند دو رابطه حاصل از تجزیه، اگر رابطه R را به دو رابطه R1 و R2 تجزیه کنیم، ممکن است وضعی پیش آید که در آن: R=R1 JOIN R2 برقرار نباشد. اگر R از پیوند R1 و R2 بدست آید، اصطلاحاً تجزیه را بی‌گمشدگی (Non-Loss) می‌گویند. با گمشدگی بودن یک تجزیه، به معنای از دست رفتن اطلاعات اولیه است و نه به معنای از دست رفتن تاپلها. اما اصطلاح بی‌گمشدگی از دقت کافی برخوردار نیست و باید گفت: تجزیه

بی‌حشو (بدون افزایش – Non Additive) به این معنی که از پیوند R1 و R2 تاپل‌های اضافی (ساختگی–جعلی) که در رابطه اولیه وجود ندارد، پدید نیاید. در واقع اگر از پیوند R1 و R2 تاپل‌های حشو پدید آیند، دیگر اطمینان نداریم که چه تاپل‌هایی واقعی هستند، بنابراین اطلاعات از دست می‌روند.

به عنوان مثال، رابطه (PRID , COID , DEID) PRCODE را در نظر می‌گیریم.

PRCODE :

PRID	COID	DEID
Pr11	Com111	D444
Pr11	Com333	D111
Pr22	Com111	D111
Pr11	Com111	D111

معنای این رابطه چنین است: "استاد p درس c را در گروه آموزشی d تدریس می‌کند." فرض می‌کنیم که بخواهیم آن‌را به دو رابطه تجزیه کنیم، به‌صورت زیر

PRCO :

PRID	COID
Pr11	Com111
Pr11	Com333
Pr22	Com111

CODE :

COID	DEID
Com111	D444
Com333	D111
Com111	D111

حال این دو قسمت را با هم پیوند می‌کنیم:

PRCO JOIN CODE :

PRID	COID	DEID
Pr11	Com111	D444
Pr11	Com111	D111
Pr11	Com333	D111
Pr22	Com111	D444
Pr11	Com111	D111

تاپل حشو (ساختگی) ←

استاد Pr22 درس Com111 را در گروه آموزشی D444 تدریس نمی‌کند، حال آنکه با این پیوند، تاپلی در رابطه پدید می‌آید که اصلاً وجود نداشته است. یعنی اطلاع حشو یا ساختگی بوجود آمده است. البته این رابطه در فرم 3NF است و لازم به تجزیه نیست، فقط بیانگر این مطلب است که باید در عمل تجزیه یک رابطه دقت داشته باشیم که تاپل‌های حشو پدید نیایند.

۴- امکان دارد بعضی از وابستگی‌های تابعی موجود در رابطه R از دست بروند. اگر رابطه‌های R1 و R2 حاصل از تجزیه رابطه R باشند و F مجموعه وابستگی‌های تابعی R باشد، با تجزیه R به دو رابطه، مجموعه‌ای از وابستگی‌های تابعی در R1 و R2 باقی می‌مانند که آن‌را F' می‌نامیم. ممکن است F' لزوماً مساوی F نباشد. به تجزیه‌ای که در آن همه وابستگی‌های تابعی R برجا می‌ماند، تجزیه حافظ وابستگی‌ها می‌گوییم.

ضوابط ریسانن (Rissanen)

تجزیه رابطه R به دو رابطه R1 و R2 مطلوبست اگر R1 و R2 مستقل از یکدیگر باشند.

R1 و R2 مستقل از یکدیگرند اگر و فقط اگر :

۱) صفت مشترک در دو رابطه، حداقل در یکی از آنها کلید کاندید باشد

۲) تمام FDهای موجود در R ، یا در مجموعه FDهای R1 و R2 موجود باشند و یا از آنها منطقاً قابل استنتاج باشند.

با بررسی این دو رابطه در می‌یابیم که : ضابطه اول تضمین می‌کند که با پیوند R1 و R2 تاپل حشو بروز نکند و ضابطه دوم تضمین می‌کند که تجزیه ، حافظ وابستگی‌ها باشد یعنی تمام FDهای رابطه اول محفوظ بماند.

قضیه هیث (Heath)

رابطه R(A,B,C) که در آن A و B و C سه مجموعه از صفات هستند، مفروض است. اگر $A \rightarrow B$ آنگاه می‌توان R را به دو رابطه R1(A,B) و R2(A,C) تجزیه کرد و این تجزیه خوب است.