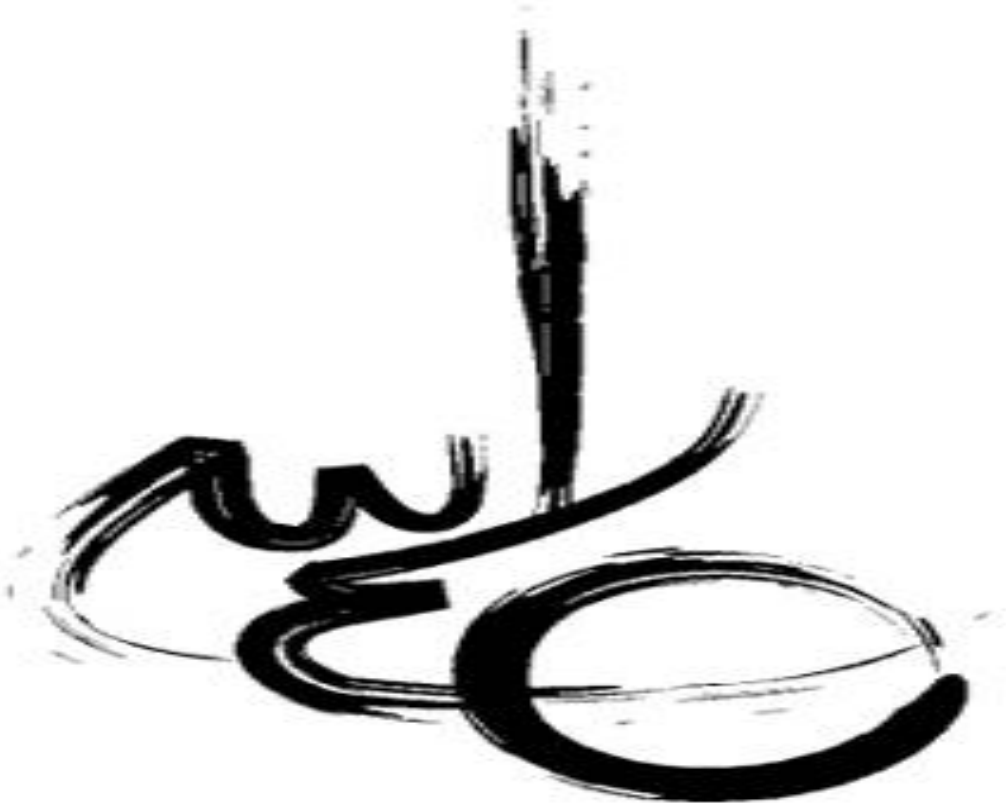


سیستم های بانک اطلاعاتی

Data Base Systems

مراجع :

1- An Introduction to Database Systems , 8th Edition , by C. J. Date , 2004 Addison-Wesley

2-مقدمه اي بر پايگاه داده ها- دكتر سيدمحمدتقي روحاني رانكوهي–ويرايش دوم 1383

3- بانك هاي اطلاعاتي – محمدرضا مقسمي

4- بانك هاي اطلاعاتي از C.J. Date مترجم : جعفرنژاد قمي ويرايش هشتم

5- Database System Concepts by Korth and Silbershatz 4th Edition 2002

عناوين:

1. عناصر اصلی سیستم بانك اطلاعاتي
2. معماری سیستم بانك اطلاعاتي
3. نمودار ER
4. نمودار EER
5. مزایای سیستم بانک اطلاعاتی
6. ساختارهای داده ای
7. دستورات SQL
8. انواع جداول
9. جامعیت بانک اطلاعاتی
10. قانون جامعیت ارجاعی
11. تراکنش
12. نرمال سازی
13. جبر رابطه ای

مقدمه:

امروزه بیش از هر زمان بحث در مورد يك سازمان و تشكيلات، وابسته به دانستن داده ها و اطلاعات، مديريت داده ها و تصميم گيری ها خواهد بود . بدون داشتن مديريت ، ارزش داده و اطلاعات نامفهوم می باشد . هدف ما در اين درس حفظ ارزش داده ها وگرفتن اطلاعات مفيد از داده ها می باشد .

سيستم ذخيره و بازيابي اطلاعات در معناي عام :

هر سيستمي كه به كاربر برنامه‌ساز يا نابرنامه‌ساز ، امكان دهد تا اطلاعات خود را ذخيره، بازيابي و پردازش كند.

تاريخچه و مفاهيم اوليه

نسلهاي ذخيره و بازيابي اطلاعات

✓ نسل اول نسل فايلهاي ساده ترتيبي

در اين نسل رسانه خارجي معمولاً نوار بوده است. اين نسل را مي‌توان نسل بي‌نرم‌افزار واسط نيز ناميد.
مشخصات كلي اين نسل عبارتند از :
1ـ ساختار فايلها ترتيبي است.
2ـ ساختار فيزيكي همان ساختار منطقي فايل است.
3ـ تنها روش پردازش فايلها، پردازش يكجا يا دسته‌اي (Batch Processing) است.
4ـ نرم‌افزار تنها عمليات ورودي/ خروجي را انجام مي‌دهد. نرم‌افزار واسطي براي مديريت پردازش فايلها وجود ندارد.
5ـ طراحي ساختار فيزيكي فايلها هم، برعهده كاربر است.
6ـ هرگونه تغيير در ساختار داده‌ها و يا رسانه‌هاي ذخيره‌سازي سبب بروز تغيير در برنامه و بازنويسي و كامپايل آن مي‌شود.
7ـ داده‌ها براي كاربرد خاصي طراحي و سازماندهي مي‌شوند.
8ـ اشتراك داده‌ها (Data Sharing) مطرح نيست.
9ـ تكرار در ذخيره‌سازي داده‌ها در بالاترين حد است.
10ـ براي انجام عمليات بهنگام‌سازي، الزاماً فايل ديگري ايجاد و تغييرات را در آن وارد كرده، نسخه قديمي را به عنوان «فايل پدر» نگهداري مي‌كنند و به اين دليل نسخه‌هاي متعددي از يك فايل نگهداري مي‌شوند.

✓ نسل دوم نسل شيوه‌هاي دستيابي

اين نسل را بايد نسل شيوه‌هاي دستيابي (Access Methods) ناميد. مهمترين ويژگي اين نسل را بايد پيدايش نرم‌افزارهاي موسوم به «شيوه‌هاي دستيابي» و همچنين ايجاد رسانه‌هاي با دستيابي مستقيم (يعني ديسك) دانست.
نرم‌افزار شيوه دستيابي، نرم‌افزاري است كه به جنبه‌هاي فيزيكي محيط ذخيره‌سازي و عمليات در اين محيط مي‌پردازد. به نحوي كه ديگر برنامه كاربر نيازي به پرداختن به اين جنبه‌ها را ندارد.
مشخصات اين نسل عبارتند از :
1ـ نرم‌افزار واسط براي ايجاد فايلها با ساختارهاي گوناگون بين برنامه‌هاي كاربردي و محيط ذخيره‌سازي وجود دارد.
2ـ امكان دستيابي ترتيبي و مستقيم به ركوردها (نه فيلدها) وجود دارد.
3ـ پردازش در محيطهاي بلادرنگ (Real Time) و برخط (On – Line) بسته به نوع سيستم عامل مي‌تواند انجام شود.
4ـ ساختار فيزيكي و ساختار منطقي فايلها از يكديگر جدا هستند ولي نه تا حدي كه برنامه‌هاي كاربردي از محيط فيزيكي ذخيره‌سازي مستقل شوند.
5ـ تغيير در رسانه‌هاي ذخيره‌سازي بر روي برنامه‌هاي كاربردي تاثير چندان ندارد.
6ـ هنوز امكان بازيابي براساس چندين كليد وجود ندارد.
7ـ ايمني و حفاظت داده‌ها مطرح بوده ولي روشهاي تامين امنيت و حفاظت ابتدايي هستند.
8ـ داده‌ها همچنان براي كاربردهاي خاص طراحي و ذخيره‌سازي مي‌شوند.
9ـ تكرار ذخيره‌سازي هنوز در حد نسبتاً بالايي وجود دارد.
10ـ براي پياده‌سازي فايل با ارتباط خاصي بين انواع ركوردها (مثلاً ارتباط سلسله مراتبي) خود برنامه‌ساز بايد ارتباطات را در برنامه‌اش بسازد.

✓ نسل سوم سيستم مديريت داده‌ها

در اين نسل نرم‌افزاري كاملتر از نرم‌افزار دستيابي به عنوان واسط بين برنامه‌هاي كاربردي و فايلهاي محيط فيزيكي طراحي و ايجاد شد. در اين نسل دريافتند كه مي‌توان برنامه‌هاي كاربردي را در قبال رشد فايلها (File Growth) مثلاً افزودن يك فيلد به يك نوع ركورد از يك فايل مصون نگاه داشت. تا قبل از اين نسل برنامه‌هاي كاربردي فقط در قبال تغييرات سخت‌افزاري و رشد كمي فايلها (يعني افزايش حجم داده‌هاي فايل) مصون بودند. مشخصات كلي اين نسل عبارتند از :

1ـ نرم‌افزار نسبتاً پيچيده‌اي به نام سيستم مديريت داده‌ها واسط بين برنامه كاربردي و محيط فيزيكي ذخيره‌سازي است.
2ـ فايلهاي منطقي متعددي مي‌توانند از داده‌هاي فيزيكي مشترك بهره‌برداري كنند و اين فايلها مي‌توانند به هم مرتبط باشند.
3ـ ميزان تكرار ذخيره‌سازي كاهش يافته است.
4ـ داده‌هاي مشترك در كاربردهاي متنوع به كار مي‌روند.
5ـ صحت داده‌هاي ذخيره شده تا حدي تامين مي‌شود.
6ـ نشاني‌دهي به داده‌ها در سطح فيلد يا گروهي از فيلدها امكان‌پذير است.
7ـ تسهيلاتي براي پردازش فايلها پيش‌بيني شده است.
8ـ بازيابي به كمك چند كليد (Multikey Retrieval) امكان‌پذير است.
9ـ تركيبي از انواع ساختارهاي فايل به كار گرفته مي‌شود.

- نسل چهارم ـ نسل DBMS

این نسل از اواخر دهه 60 آغاز شد و هم‌اکنون نیز ادامه دارد. مهمترین خصیصه این نسل مستقل شدن برنامه‌هاي کاربردي (Application Program) از جنبه‌ها و خصوصیات محیط فیزیکي ذخیره‌سازي است 0 (استقلال داده اي)

برای ایجاد سیستم بانک اطلاعاتی دو شیوه عمومی وجود دارد :

الف : شيوه سنتي يا مشي فايلينگ
ب : شيوه (مشي) پايگاهي

شيوه فايلينگ چيست ؟

آيا به نظر شما بانك اطلاعات صرفا" سيستمي است براي ذخيره داده‌ها و ايندكس‌گذاري روي ركوردها كه جستجو را تسهيل مي‌كند؟
آيا بانك اطلاعات صرفا به داده‌هايي گفته مي‌شود كه به صورت منطقي در كنار هم و با ارتباطي خاص قرار گرفته‌اند؟

تا قبل از 1970 (دهه انقلاب بانك اطلاعات) ، سيستمهايي توسعه يافته تلقي ميشدند كه كارهاي بالا را به خوبي انجام دهند و در حقيقت يك سيستم فايلينگ قوي بودند0

در سيستمهاي فايلينگ هر كاربر مي‌توانست روي كل پرونده مسلط شود و از اطلاعات آن استفاده كند.

اين مساله مشكلات زير را به همراه داشت:

1- عدم اشتراک داده ها : در سيستمهاي چند كاربره، كاربران ديگري كه با يك پرونده كار داشتند بايست منتظر مي‌ماندند تا كاربر اول كار خود را تمام كند و پرونده را آزاد كند و اين بازده و سرعت را پايين مي‌آورد.

2-عدم وجود ضوابط ايمني كارا و مطمئن : تسلط كاربر بركل پرونده يعني قرباني شدن امنيت.

3- مصرف نابهينه امكانات سخت‌افزاري و نرم‌افزاري

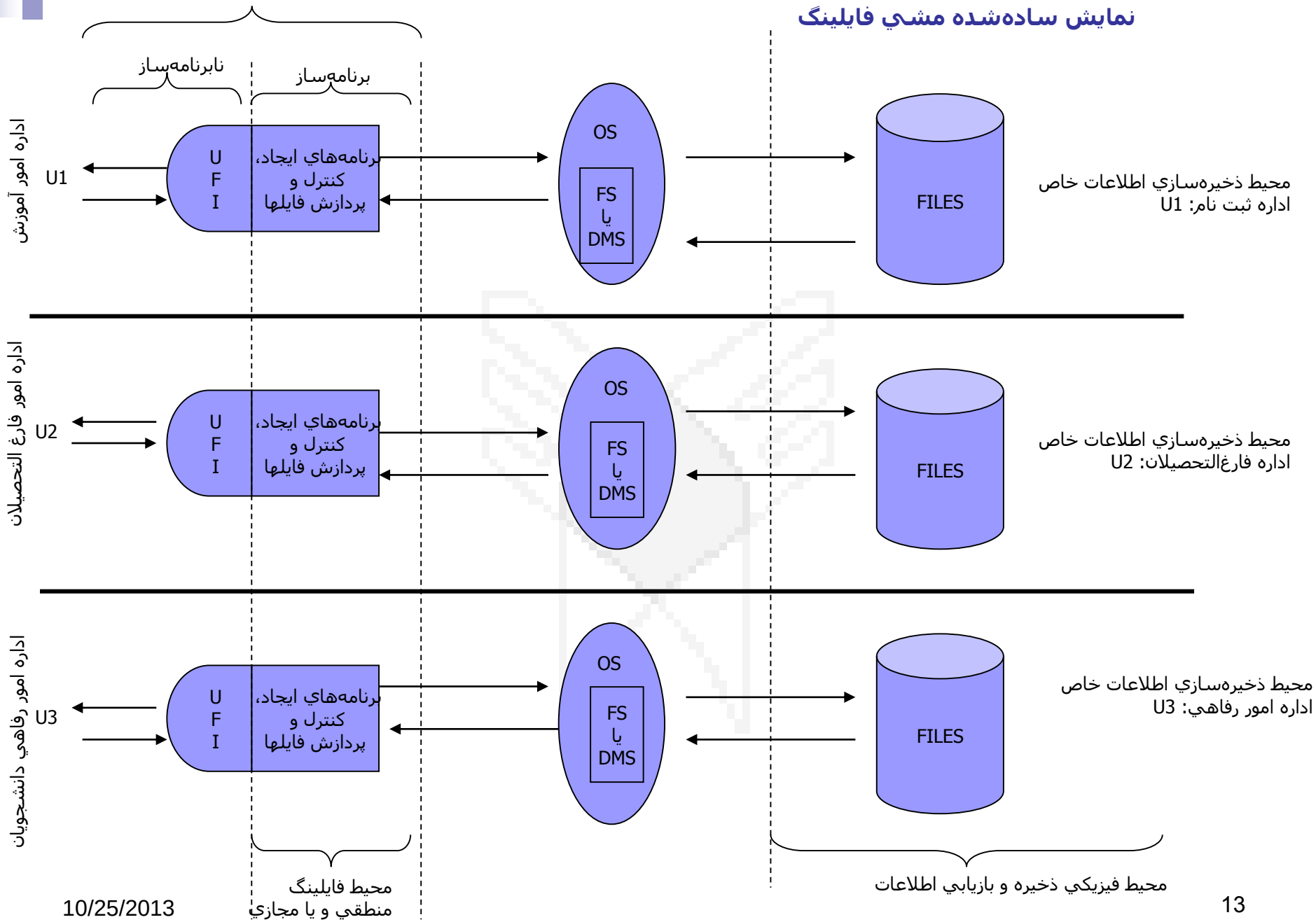
4- حجم زياد برنامه‌سازي و پيچيده

5- وابستگي برنامه‌هاي كاربردي به محيط ذخيره‌سازي داده‌ها

6- عدم وجود محيط مجتمع ذخيره‌سازي اطلاعات و عدم وجود سيستم يكپارچه

اين مشكلات و مشكلات ديگري مانند افزونگي بيش از حد داده‌ها و در نتيجه ناسازگاری داده ها باعث شدند تا سيستمهاي مديريت بانك اطلاعات *(DBMS)* ها و مفاهيم بانك اطلاعات به موازات آن رشد پيدا كنند.

نمايش ساده‌شده مشي فايلينگ



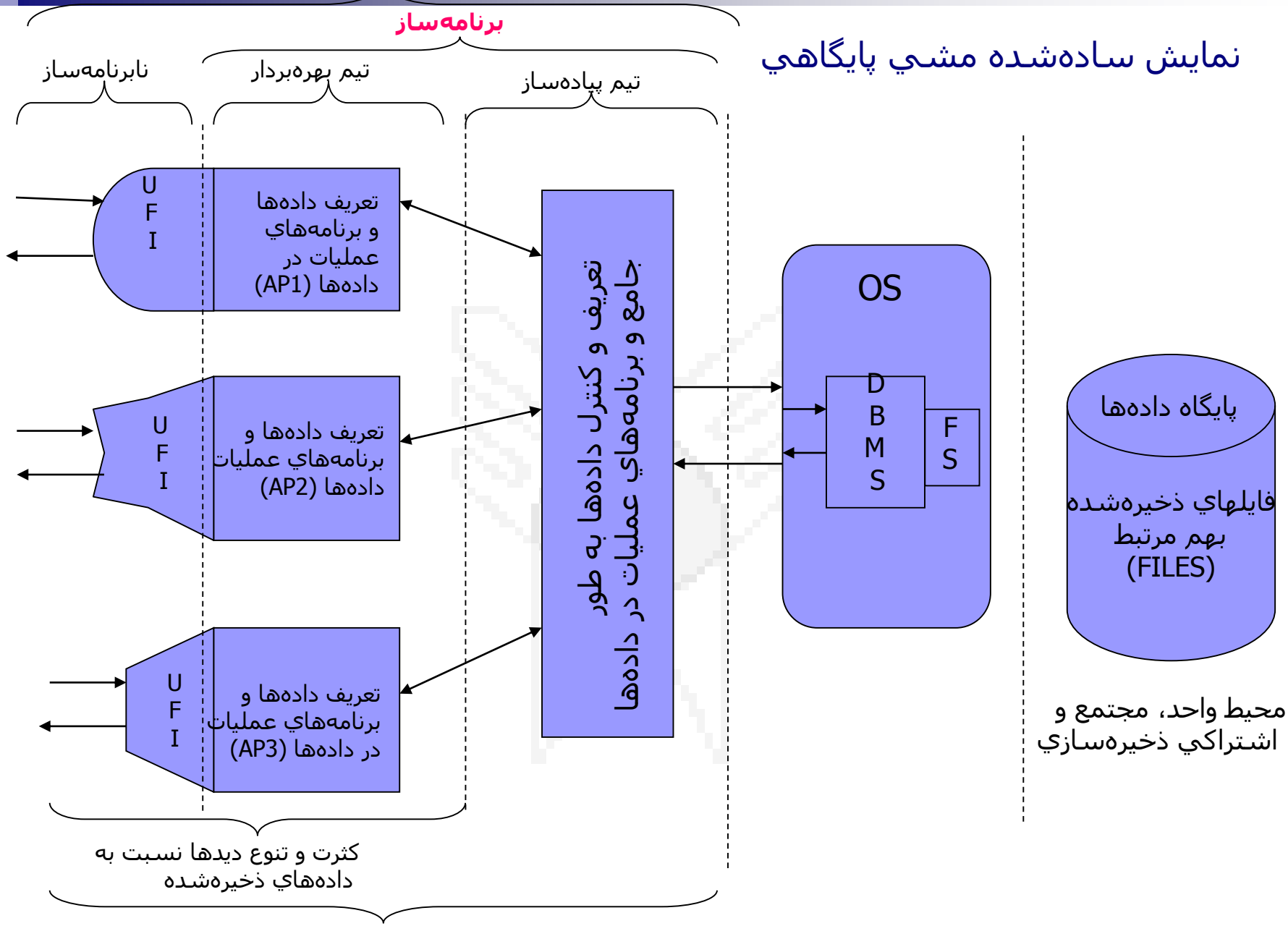
سیستم بانك اطلاعاتی چیست ؟

تعریف اول : سیستم بانك اطلاعاتی یا پایگاه داده ها به طور عمومی یك سیستم کامپیوتری نگهداری رکوردها با تضمین امنیت و جامعیت اطلاعات در محیط چند کاربره و میتوان آنرا به عنوان نوعی قفسه بایگانی الکترونیکی پیشرفته به حساب آورد0

تعریف دوم: یك سیستم بانك اطلاعاتی وظیفه دریافت اطلاعات از کاربران و نگهداري آن به شکل داده هاى خام با چیدمان صحیح در سیستم و در اختیار قراردادن آن در زمان مناسب و به شکل قابل قبول به کاربران مجاز جهت رویت و اضافه نمودن و تصحیح و حذف اطلاعات میباشد.

تعریف سوم : مجموعه‌اي است از داده‌هاي ذخیره شده و پایا، به صورت مجتمع(یکپارچه) (نه لزوما فیزیکي، بلکه حداقل به طور منطقي)، بهم مرتبط، با کمترین افزونگي، تحت مدیریت یك سیستم کنترل متمرکز، مورد استفاده یك یا چند کاربر از یك یا بیش از یك "سیستم کاربردي"، به طور همزمان و اشتراکي

نمايش ساده‌شده مشي پايگاهي

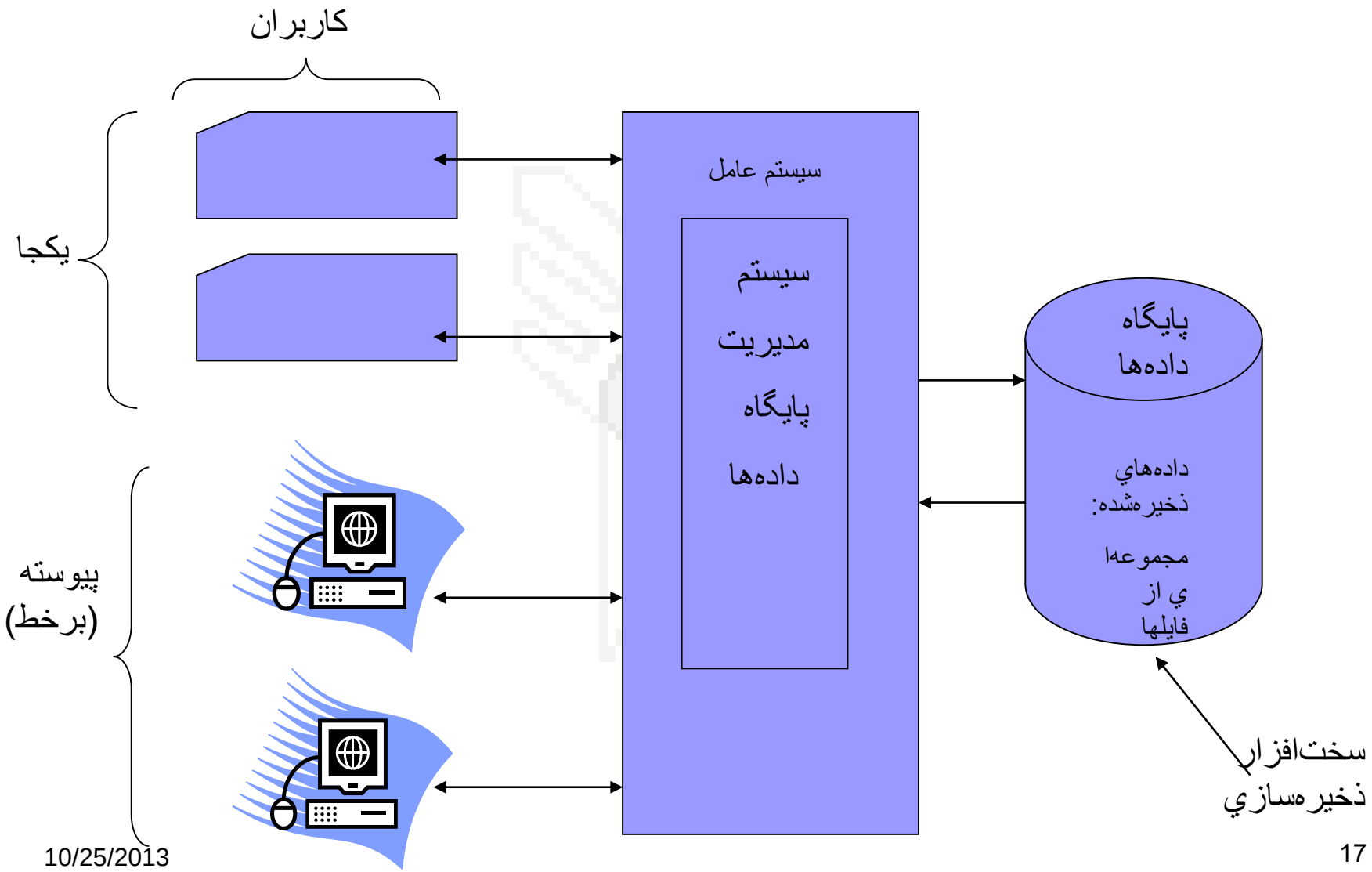


مهمترین مزایای سیستم بانك اطلاعاتی نسبت به دیگر سیستمها :

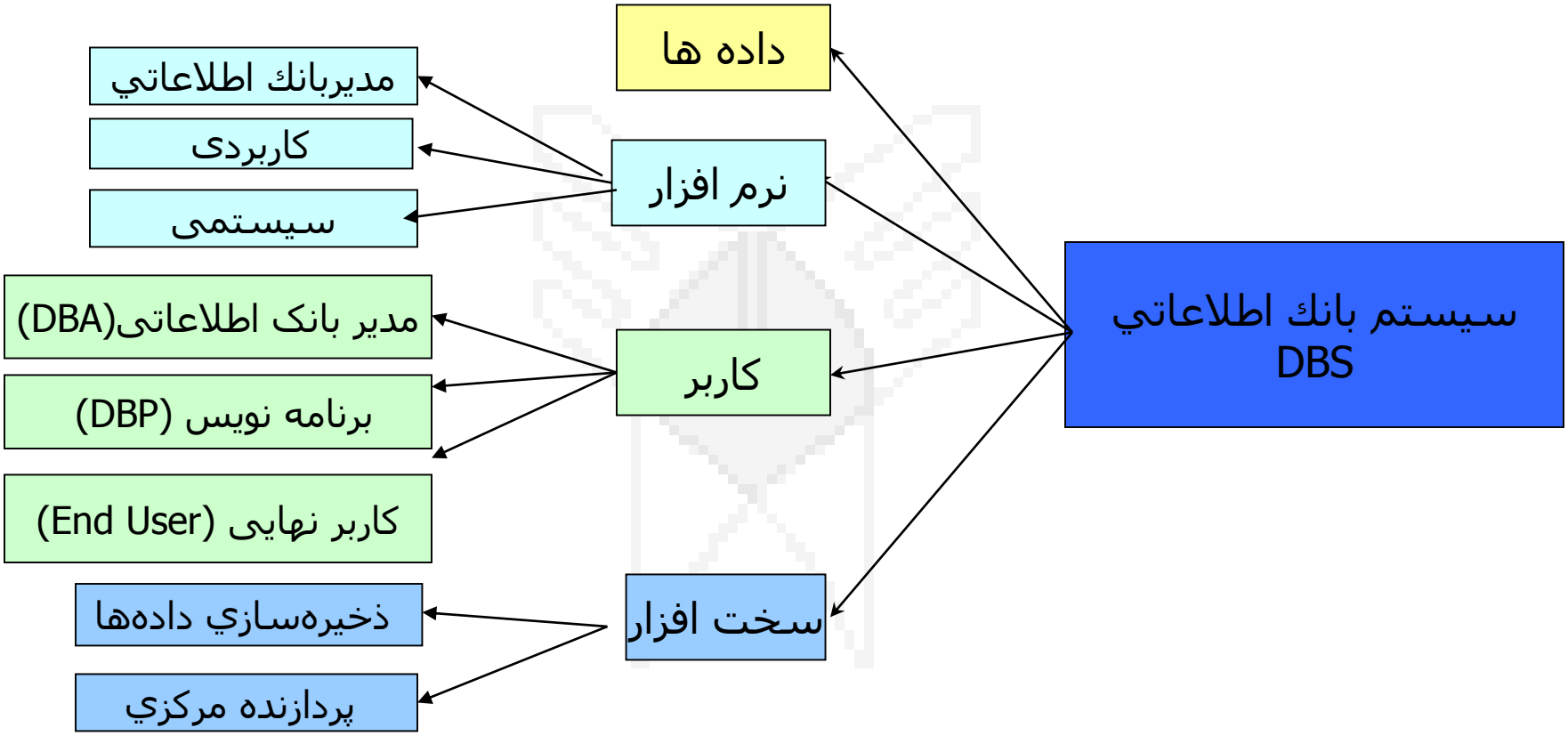
1ـ امنیت بیشتر 2ـ سرعت بیشتر 3ـ کاهش هزینه 4ـ حجم کمتر

عناصر محیط پایگاه داده‌ها

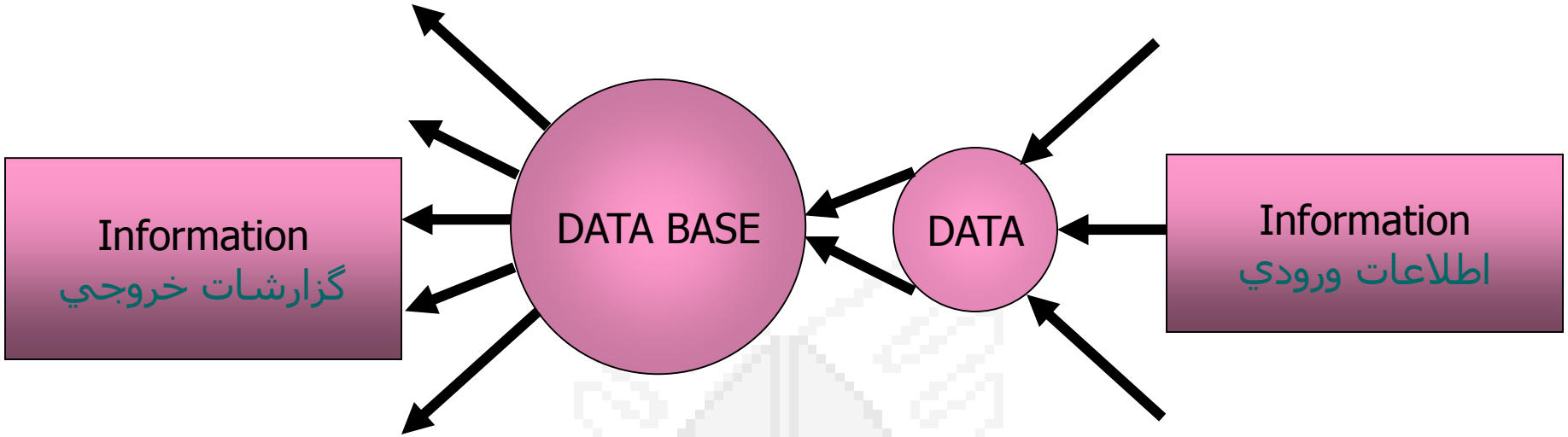
1- داده 2- نرم‌افزار 3- کاربر 4- سخت‌افزار



عناصر اصلی محیط بانك اطلاعاتی:



عنصر اول يك سيستم بانك اطلاعاتي : 1- دا ده ها Data



تعريف اول- نمايش ذخيره‌شده اشياء فيزيكي، چيزهاي مجرد، بوده‌ها، رويدادها يا چيزهاي قابل مشاهده كه در تصميم‌سازي بكار مي‌آيند.

تعريف دوم- بوده‌هاي خام كه معناي اندكي دارند مگر اينكه به صورت منطقي سازمان‌دهي شده باشند

تعريف سوم : داده عبارت است از مقادير صفحات خاصه انواع موجوديتها

- به داده های پردازش شده "اطلاعات" و به اطلاعات خام "داده" می گویند.
- اطلاع عبارت است از داده سازمان يافته‌اي كه شناختي را منتقل مي‌كند
- در يك سيستم بانك اطلاعاتی ، اطلاعات از کاربران اخذ و بشکل خلاصه و فشرده (داده) در DataBase ذخیره میشود، که در صورت نیاز کاربران مجاز بشکل اطلاعات (گزارش) در اختیار آنها قرار گیرد.
- در تبديل اطلاعات ورودي به داده هيچ اطلاعاتی نبايد از دست برود .

- داده‌های عملیاتی یا داده هاي پايدار داده‌هايي است ، در مورد موجوديتهای مختلف محيط عملياتی و نيز ارتباط بين انواع موجوديتها ، که می خواهيم ذخيره کنيم 0

- داده‌های عملیاتی با داده‌های ورودی و خروجی تفاوت دارند.
- داده‌های ورودی اطلاعاتی هستند که نخستین بار وارد سیستم شده می توانند سبب ایجاد تغییر در داده‌های عملیاتی شوند یا خود جزئي از داده‌های عملیاتی محیط گردند.
- داده‌های خروجی عبارتند از پیام‌ها، پاسخ‌ها و نتایجی که سیستم پیرو درخواست کاربر به او می‌دهد. این داده‌ها می‌توانند ازداده‌های عملیاتی استخراج شوند ولی خود بخشی از داده‌های عملیاتی تلقی نمی‌شوند.

سيستم هاي Single User , Multi User :

به سیستمی که در هر لحظه بیش از یك كاربر میتواند با آن متصل شده و كاركند ، سیستم چند كاربره یا Multi Userو گرنه تك كاربره یا Single User گویند 0 مدیریت سیستم های چندكاربره به علت اینكه در هر لحظه چند كاربرمیتوانند به یك قلم داده ای دسترسی داشته و آنرا تغییر دهند مشكلتر و مهمتر میباشد0
البته در حال حاضر اكثر سیستم هاي طراحي شده بانك اطلاعاتي چند كاربره هستند0

در رابطه با داده ها بحث اجتماع و اشتراك داده ها مورد توجه قرار می گیرد :

اجتماع داده ها : جمع آوری داده های مربوط به یك سازمان در یك محل با هرگونه تکرار را اجتماع داده می گویند0
بعنوان مثال سیستم فروش یك فروشگاه که داراي چند ایستگاه کاري مجزاست و داده هاي آن مثلا طي ابتداي هر روزکاري بروز میشود داراي عدم اطمینان و صحت اطلاعات طي روز میباشد در نتیجه با استفاده از مبحث اجتماع داده ها بیائیم داده هاي کل سیستم فروش را در یك جا (مثلا روي یك کامپیوتر سرویس دهنده Server) جمع کرده تاکاربران مجاز ایستگاههاي کاري مختلف از داده هاي متمرکز شده Server استفاده نمایندکه این امرمیتواند باعث یکپارچگي داده ها گردد0

اشتراك داده ها : جمع آوری داده های مختلف یك سازمان (مثلا سیستم فروش – سیستم انبار – سیستم حسابداري – سیستم کارگزیني – سیستم حقوق و دستمزد یك سازمان که داراي نقاط مشترك اطلاعاتي مختلفي هستند) در یك جا با حداقل تکرار (نه بدون تکرار) را اشتراك داده ها می گویند .
به اشتراك داده ها را دسترسی همزمان نیز می گویند.

عنصر دوم يك سيستم بانك اطلاعاتي : 2 - نرم افزار ها Sotware

نرم‌افزارهاي محيط بانكی را می‌توان به سه دسته تقسيم‌بندی كرد :

الف) نرم افزار بانك اطلاعاتی DBMS :

Data Base Management System

در واقع *DBMS* برنامه اي است كه همه پرونده ها را در اختيار خود مي‌گيرد و همه كاربران بدون استثناء مي‌بايست براي دسترسي به داده‌ها درخواستهاي خود را از طريق *DBMS* بدهند و ايشان اگر صلاح ديدند به درخواستها پاسخ مثبت مي‌دهند. يعني وظيفه تعريف ، ذخيره و بازيابی داده ها را انجام ميدهد . حفظ تماميت و امنيت بانك اطلاعاتی بعهده اين نرم افزار است.

ب) نرم‌افزارهای كاربردی : نرم افزارهای كمكی در يك سيستم بانك اطلاعاتی وظيفه پشتيبانی و بازيابی را به عهده دارند.

ج) نرم‌افزار سيستمی: از جمله نرم افزار سيستم عامل كه نرم افزارهای ديگر بروی آن نصب ميشوند.

عنصر سوم يك سيستم بانك اطلاعاتي : 3 - كاربران Users

در معناي عام ، هر استفاده‌كننده از پايگاه داده‌ها را كاربر گوييم.

كاربران يك سيستم بانك اطلاعاتی بترتيب اهميت و سطح دسترسی به سيستم بانك اطلاعاتي :

الف) كاربر اداره كننده بانك Data Base Administrator (DBA)

ب) كاربر برنامه‌نويس يا DataBasePrograme‌r (DBP)

ج) كاربر نهايي (End User)

تشريح انواع كاربر(Users) :

الف) Data Base Administrator (DBA) : يكی از مهمترين كاربران در سيستم بانك اطلاعاتی ، كه مسئول طراحي و تصميم گيري براي كليه موارد يك سيستم بانك اطلاعاتی است. اداره كننده بانك ، فرد يا گروهی از افراد هستند كه مسئوليت ايجاد، پياده‌سازی و نگهداری بانك را در محيط عملياتی بر عهده دارد.

ب) كاربر برنامه‌نويس يا DataBaseProgramer (DBP) : اين گروه افراد مسئول ساختن برنامه‌هايي هستند كه از يك طرف به بانك اطلاعات متصل است و از طرف ديگر به كاربر نهايي يا همان اپراتور، در واقع اين افراد تصميمات مدير را پياده سازي مي‌كنند.

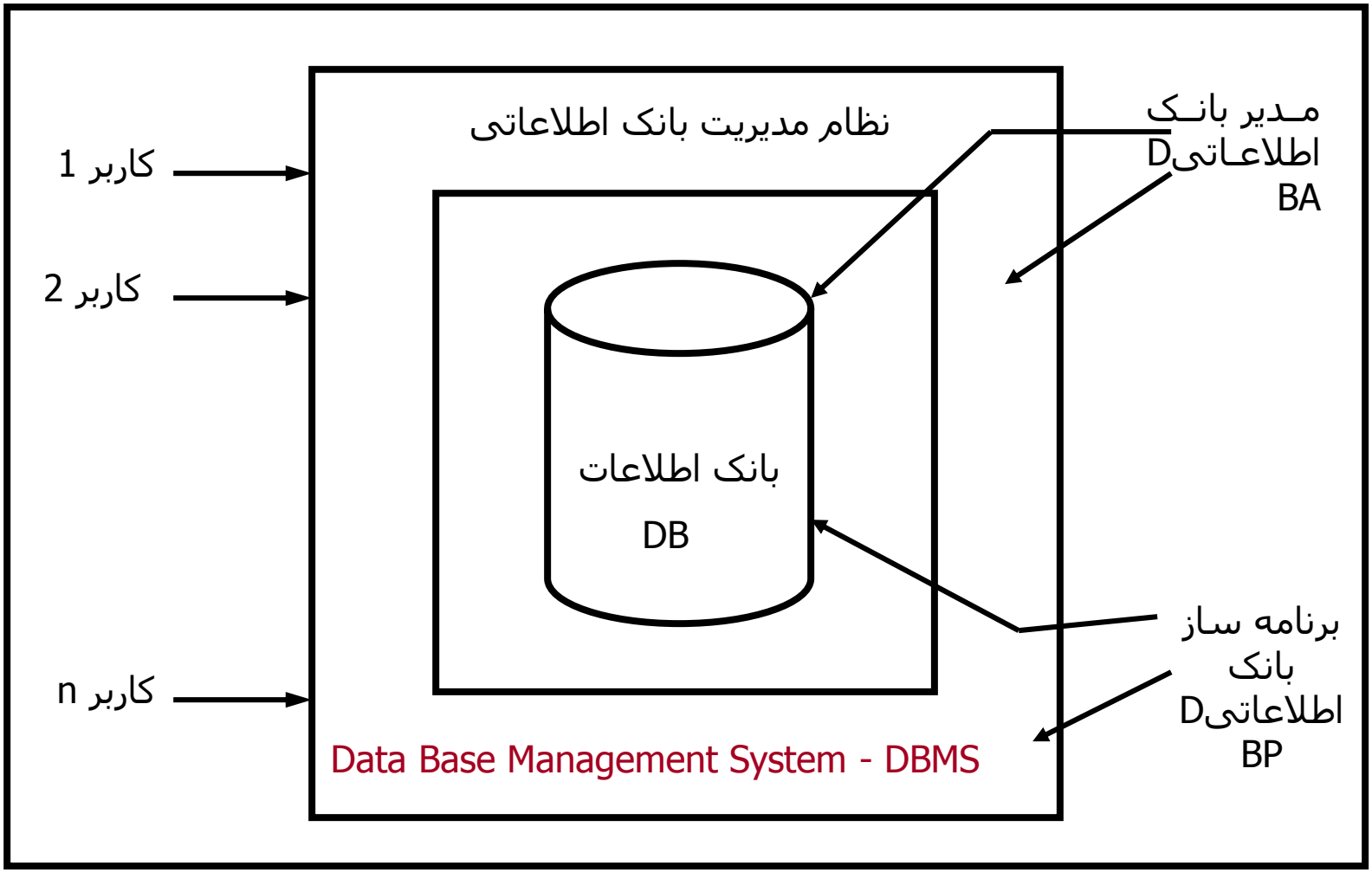
ج) كاربر نهايي (End User) :كاربران نهائي كساني هستند كه از طريق برنامه‌هاي تهيه شده داده‌ها را در حيطه نظارت *DBMS* دستكاري مي‌نمايند.
بعنوان مثال : دربانك صادرات شعبه X ، رئيس شعبه و معاونين و مديران قسمت ها و همه كارمندان و مشتريان ، كاربر نهائي ولي با دسترسي هاي مختلف هستند0

عنصر چهارم يك سيستم بانك اطلاعاتي : 4 – سخت افزار HardWare

سخت‌افزار محيط بانكي را مي‌توان به صورت زير تقسيم‌بندي كرد:

الف) سخت‌افزار ذخيره‌سازي داده‌ها
شامل رسانه هاي ذخيره سازي جهت نگهداشت و پشتيباني از اطلاعات
ب) سخت‌افزار پردازنده مركزي
انتخاب سرعت Server به نسبت داده هاي سيستم
ج) سخت‌افزار ارتباطي
جهت اشتراك داده ها در سيستم هاي يكپارچه و ارتباط بين سازمانهاي مختلف با پراگندگي مختلف نياز به سخت افزارهاي ارتباطي شامل شبكه باسيم و بي سيم ميباشد0

بانک اطلاعات علمی - کاربردی

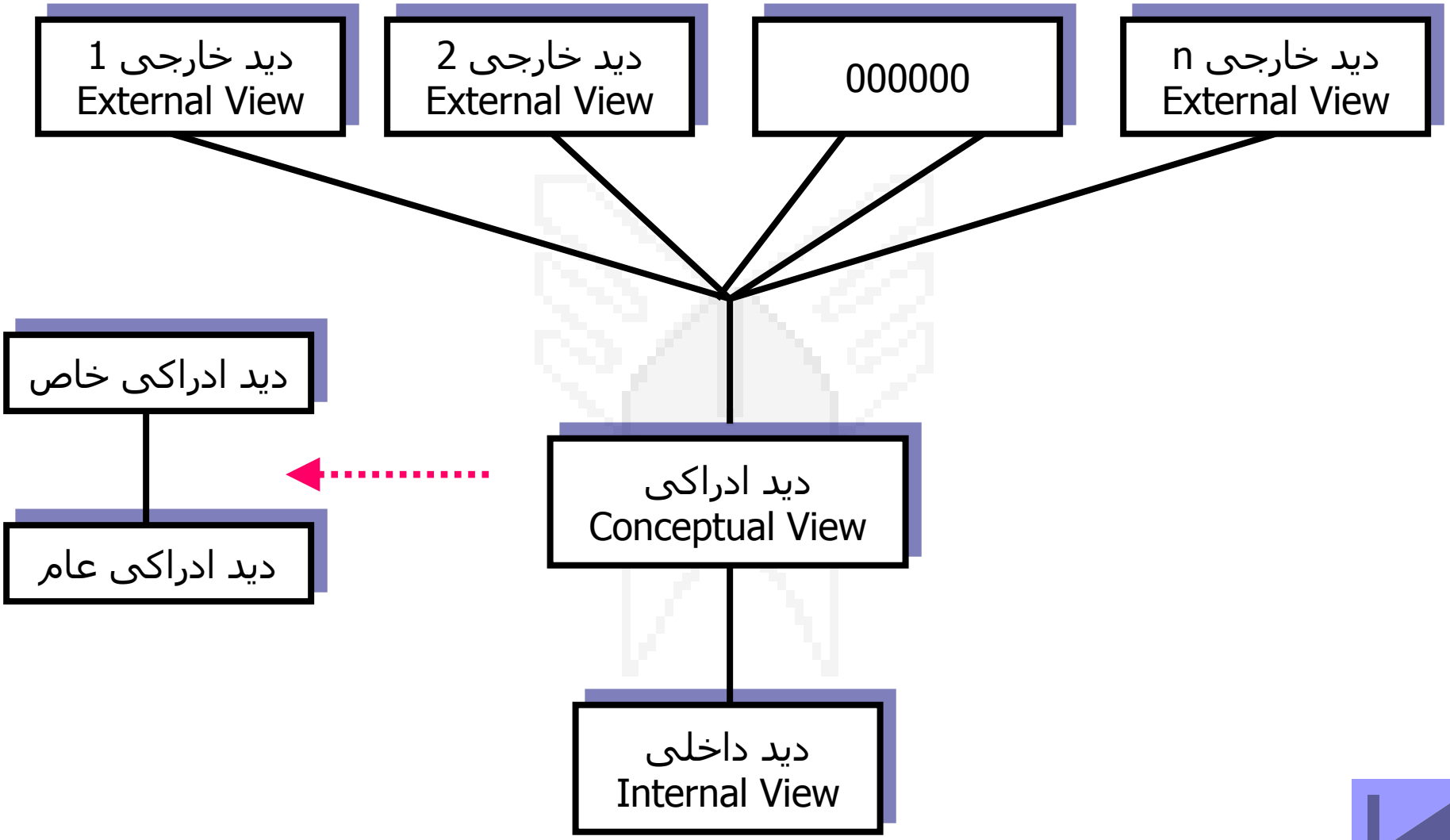


جايگاه DBMS در يك سيستم كامپيوتري

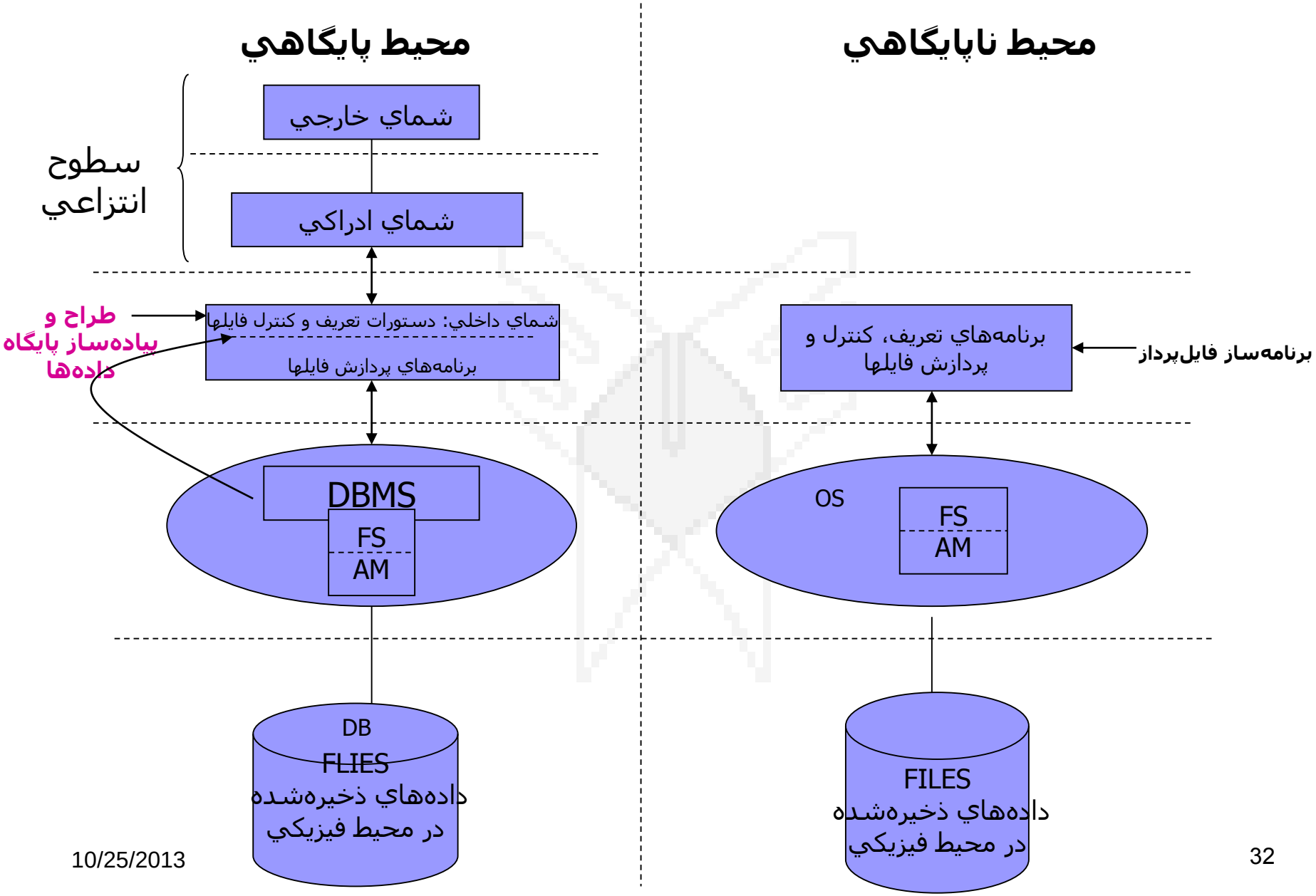
معماری سیستم بانك اطلاعاتی ANSI /Sparc :

بعد از سال‌ها بحث پیرامون اینکه ساختار بانك اطلاعات چیست، سرانجام کمیته *ANSI/SPARS* معماري 3 لايه را ارائه داد که بعدها يك لايه به آن افزودند و ما معماري 4 لايه را بررسي مي‌كنيم. اين معماري يك مدل نظري از بانك اطلاعاتي ست و به همين جهت قابل تطبيق روي انواع مدل‌هاي بانك اطلاعاتي است 0

معماری سیستم بانك اطلاعاتی ANSI /Sparc :



سطوح معماري در محيط ناپايگاهي و محيط پايگاهي و نقش DBMS در ايجاد، مديريت و پردازش فايلها



1- ديد خارجي يا External View

ديد خارجي ، ديد خاص هر گروه ازكاربران است به داده‌های ذخیره شده در بانك اطلاعاتي 0 يعني اينكه هر كاربر چه قسمتهايي از بانك اطلاعات را اجازه دارد ببيند و چه كارهايي روي آن قسمتها مي‌تواند انجام دهد.(امنیت)

اصل اول بانك اطلاعات اين اصل مي‌گويد به هر كس همان مقدار اطلاعات بده كه لازم دارد نه بيشتر

هر گروه ازکاربران دید خاص خود را دارند و همچنین چند کاربر می‌توانند دارای دید یکسانی باشند .

دید خارجی نزديك‌ترين سطح به كاربران نهائي است.

2 – دید ادراکي خاص Spec. Conceptual View

این دید ، دید یکپارچه طراح بانك از کل بانك اطلاعاتي و داده هاي ذخیره شده درآن است 0 یعنی داده‌های انواع موجودیتها و ارتباط بین آنها ، آنگونه که طراح می بیند.
این لایه ، تصویر ادراکي خاص یا همان مدل منطقي است. یعني اینکه داده‌ها به صورت منطقي چگونه کنار هم قرار مي‌گیرند. مدل‌هاي مرسوم جدول، درخت، گراف و مانند این‌هاست.
در سطح ادراکی ارتباط موجودیتها و صفات خاصه، امنیت و جامعیت داده‌ها مطرح می‌گردد.
طراحي این لایه به عهده مدیر بانك میباشد0
فقط مدیر بانك و برنامه نویس هستند که این لایه براي آنها قابل استفاده است

3 – ديد ادراكي عام Public Conceptual View

اين لايه ، ديد منطقي يكپارچه از كل بانك اطلاعاتي است 0
لايه سوم لايه تصوير ادراكي عام است. تصوير ادراكي عام يعني طراحي بانك اطلاعات بدون وابستگي به مدل خاص و پياده سازي فيزيك خاص. اين لايه را كاربر نهايي نمي‌بيند.(بسيار مهم)
طراحي اين لايه به عهده مدير بانك ميباشد0
فقط مدير بانك است كه اين لايه بطور كامل براي او قابل استفاده است

4 – دید داخلي Internal View

در این سطح یا دید در واقع فایلهای محیط فیزیکی تعریف می‌شود , از نظر محتوا، ساختار و استراتژی دستیابی .
در شمای داخلی، انواع رکوردها، فایلها، صفات خاصه شاخص (استراتژی دستیابی)، نحوه نمایش و تشریح رکوردهای ذخیره شده در فایل، توالی رکوردها، تخصیص فضای ذخیره‌سازی برای داده‌ها، محل رکورد، فشردگی داده‌ای و تکنیکهای رمزگذاری داده‌ها تشریح می‌شوند. در یک سیستم بانک اطلاعاتی , کاربران اساسا به مسائل این سطح نمی پردازند. سطح داخلی نزدیکترین سطح به رسانه ذخیره‌سازی فیزیکی است

اجزاء دیگر معماری سیستم بانك اطلاعاتی ANSI/SPARC :

الف) تبدیلات بین سطوح (Transformation یا Mappings)

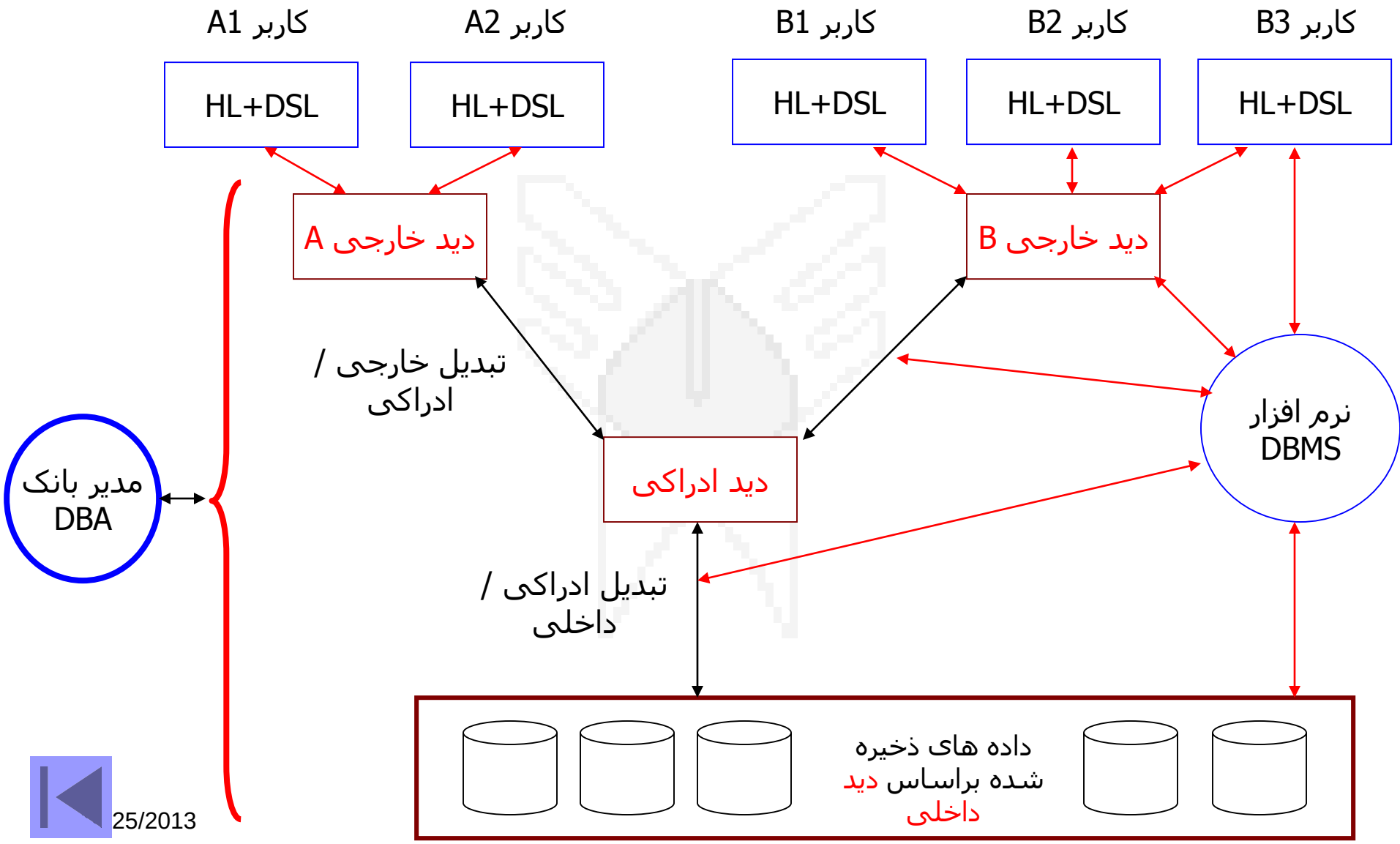
ب) زبان میزبان یا HL (Host Language)

ج) زبان فرعی داده ها یا DSL (Data Sub Language)

به علاوه در چنین سیستمی سه عنصر مهم دیگر نیز وجود دارند :

کاربر - DBMS - DBA

مدل پیشنهادی ANSI (ارائه شده در سال 1975) برای معماری سیستم بانك اطلاعاتی بطور کامل به شکل زیر است :



الف) تبدیلات بین سطوح (Transformation یا Mappings) :

در شکل استاندارد ANSI دو تبدیل وجود دارد :

تبدیل ادراکی/ داخلی :

مثلاً اگر طراح بانك ، تعدادی جدول را طراحی کرده باشد، در تبدیل ادراکی به داخل برای هر جدول می‌توان فایلی تعریف کرد بصورتی که هر سطر جدول رکوردی از این فایل باشد. تغییرات در سطح داخلی بانك همیشه ممکن است بروز کند. اینگونه تغییرات نباید در دید ادارکی تاثیر داشته باشد. در تبدیل ادراکی/داخلی از سیستم عامل نیز کمك گرفته می شود. در اینجا هدف حفظ اسقلال فیزیکي داده هاست که به مفهوم مصون نگهداشتن تغییرات رسانه ذخیره سازي یا همان محیط فیزیکي از دید کاربران است0

تبدیل خارجی/ادراکی :

این تبدیل مکانیسمی برای برقراری تناظر بین دیدهای خارجی مختلف و دید واحد ادراکی است. یك دید مشخص از یك کاربر خاص، بخشی است از دید واحد ادراکی و از نظر انواع موجودیتها، صفات خاصه هر موجودیت، نوع صفت و...لزوماً همان نیست که در دید ادراکی از نظر طراح وجود دارد. به عنوان مثال اضافه کردن یك فیلد به جدولي که قبلا داراي 3 فیلدبوده باشد0در اینجا هدف حفظ اسقلال منطقي داده هاست که به مفهوم حتي الامکان مخفي نگه داشتن جزئیات پیاده سازي از دید کاربران

ب) زبان میزبان (HL) :

منظور از زبان میزبان یکی از زبانهای سطح بالای برنامه‌سازی مثل کوبول، C,PL/1، پاسکال، بیسیك، Delphi، Visual C، Visual Basic، Java می باشد.

ج) زبان فرعی داده یی (DSL) :

در يك سيستم بانك اطلاعاتي براي ايجاد بانك اطلاعاتي خاص و اضافه و تصحيح و حذف كردن داده ها نياز به يك زبان موسوم به DSL ضروري مينمايد 0
زبان DSL زبانی است از سطح بالاتر که میهمان يك زبان سطح بالا مثل Visual C می شود هر مدل داده‌ی خاص (مثل سلسله مراتبی، شبکه‌ای، رابطه‌ای) زبان فرعی خاص خود را دارد. تعداد احکام این زبانها معمولاً کم است. برای هر سطح از معماری دستوراتی وجود دارد موسوم به : زبان فرعی داده يي خارجی، زبان فرعی داده يي ادراکی و زبان فرعی داده يي داخلی.

احکام زبان DSL را می‌توان به سه دسته زیر تقسیم کرد :

1ـ احکام تعریف داده‌ها (Data Definition Language = DDL)

2ـ احکام کار با داده‌ها (Data Manipulation Language = DML)

3ـ احکام کنترلی (Data Control Language = DCL)

به طور کلی دو دسته زبان داده يي وجود دارد :

1- یکی زبان داده يي نامستقل یا ادغام شده (Embedded)

2- دیگری زبان داده يي مستقل.

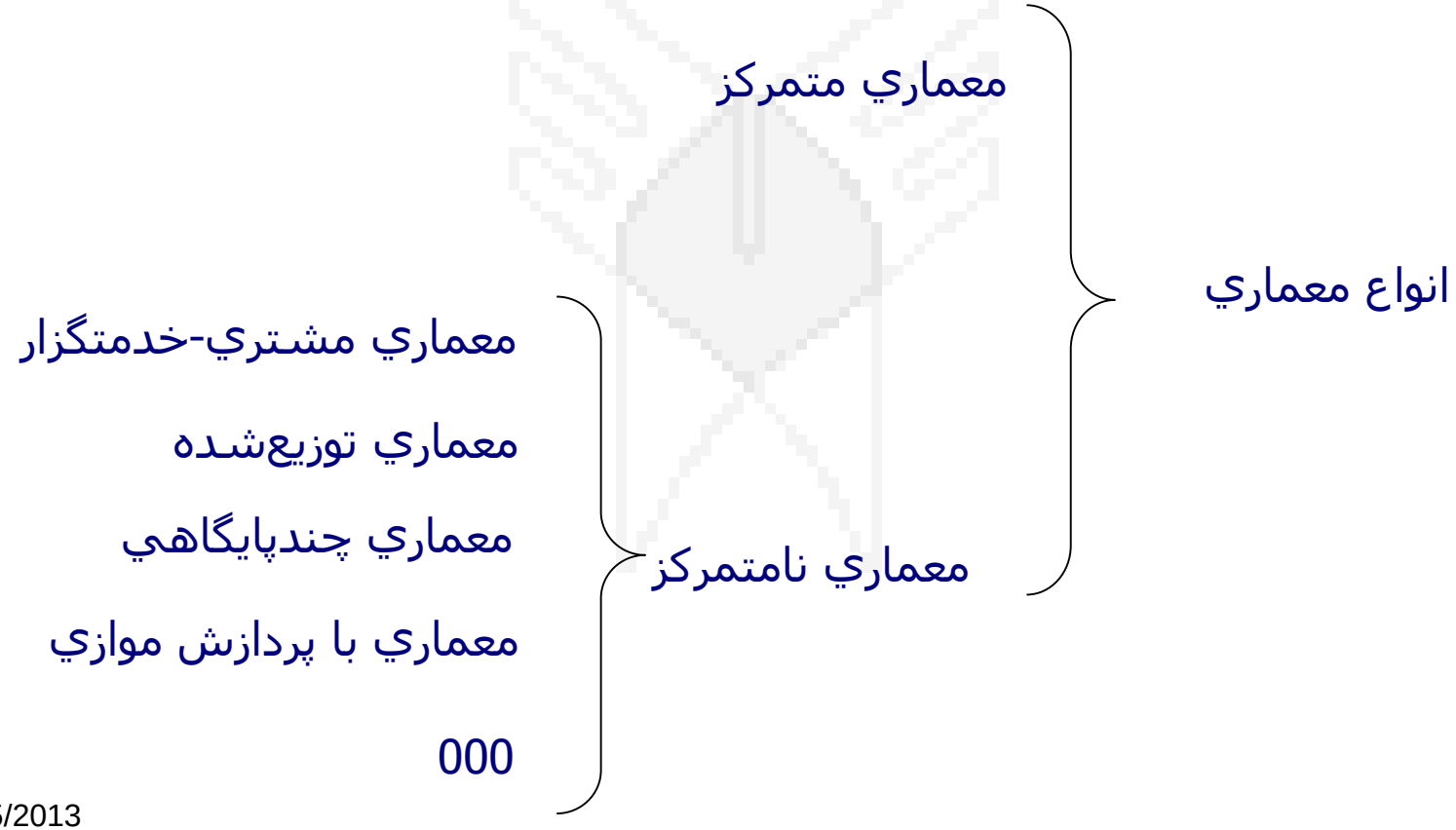
- در نوع نامستقل DSL حتماً باید میهمان يك زبان سطح بالا باشد مثل SQL که در دلفی یا ویژوال بیسيك استفاده می‌شود یا Btrieve که زبان فرعی داده‌ای برای C یا پاسکال است. در نوع مستقل DSL نیازی به زبان میزبان ندارد مثلاً Access و Foxpro نیازی به زبان میزبان ندارند.
- SQL هم به صورت مستقل و هم به صورت نامستقل وجود دارد.

معماری بانک اطلاعات به صورت خلاصه

تصویر خارجی	کاربر n ..... کاربر 2 کاربر 1	دید های کار بران مختلف(Views)
تصویر ادراکی عام	نمو دار های EER، NIAMو موجودیت 2 — ارتباط — موجودیت 1	کل بانک بدون توجه به مدل خاصی
تصویر ادراکی خاص	مدل شئی گرا مدل شبکه ای مدل سلسله مراتبی مدل جدولی A B C	کل بانک در قالب مدل انتقابی
تصویر فیزیکی	.....	کل بانک روی رسانه

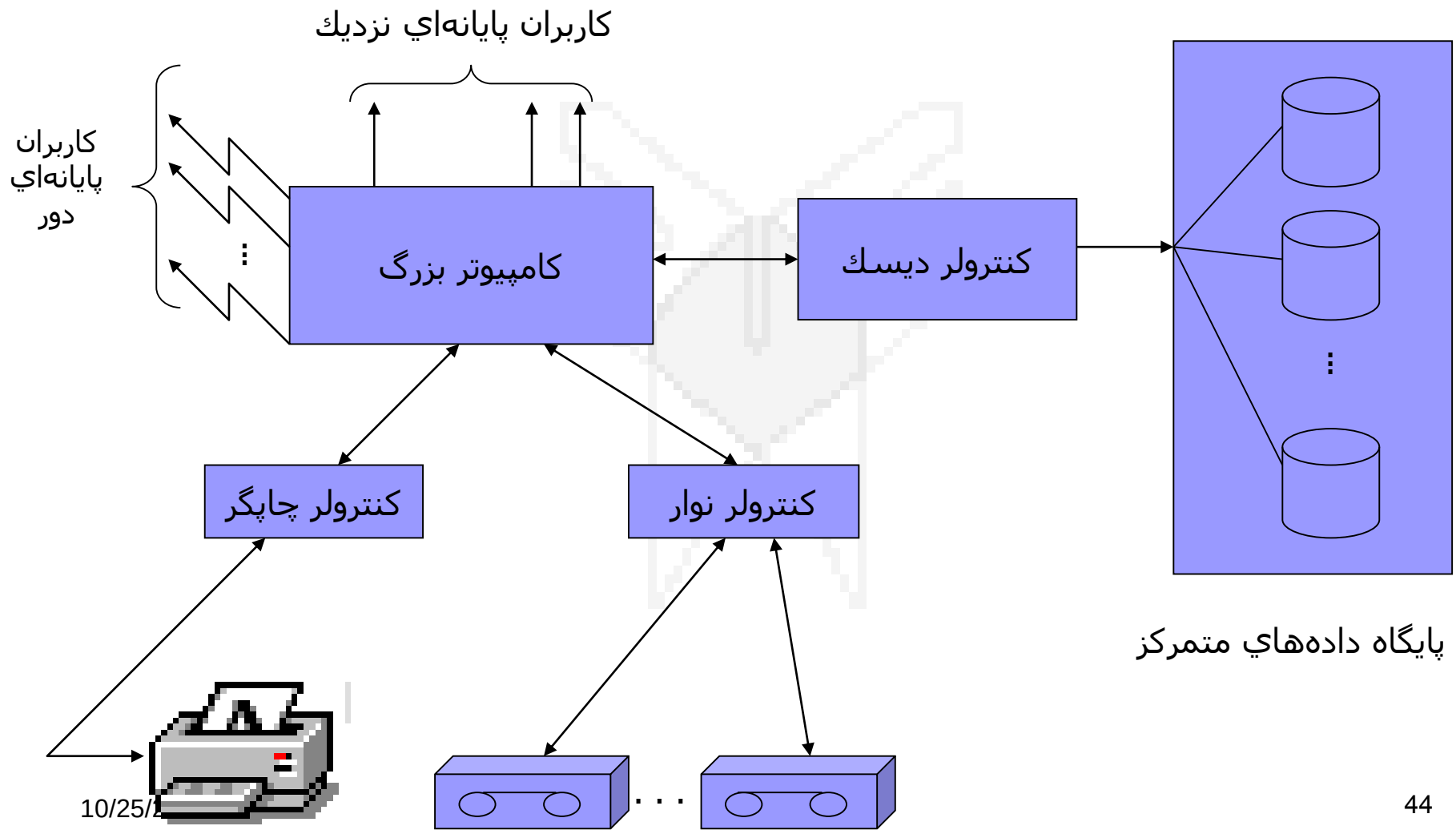
ديگر معماريهاي سيستم پايگاه داده‌ها

منظور، چندي و چوني اجزاء تشكيل‌دهنده سيستم و نيز پيكربندي يا طرز تركيب اجزاء سيستم و چگونگي تعامل اجزاء با يكديگر است. در اين معماري حداقل يك پايگاه داده‌ها، يك سيستم مديريت پايگاه داده‌ها، يك سيستم عامل، يك كامپيوتر با دستگاههاي جانبي و تعدادي برنامه كاربردي و كاربر وجود دارند.



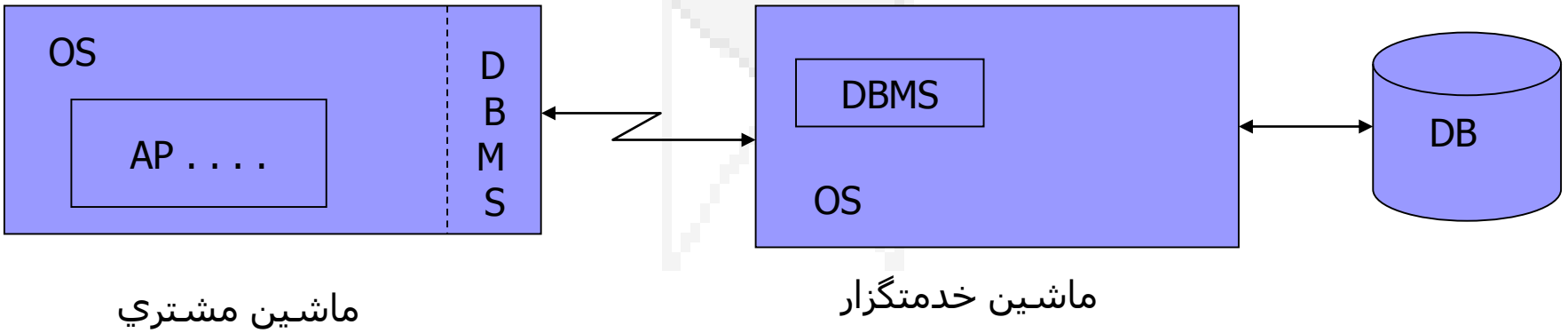
معماري متمركز

در اين معماري يک پايگاه داده روي يک سيستم كامپيوتري و بدون ارتباط با سيستم ديگر ايجادمي شود .

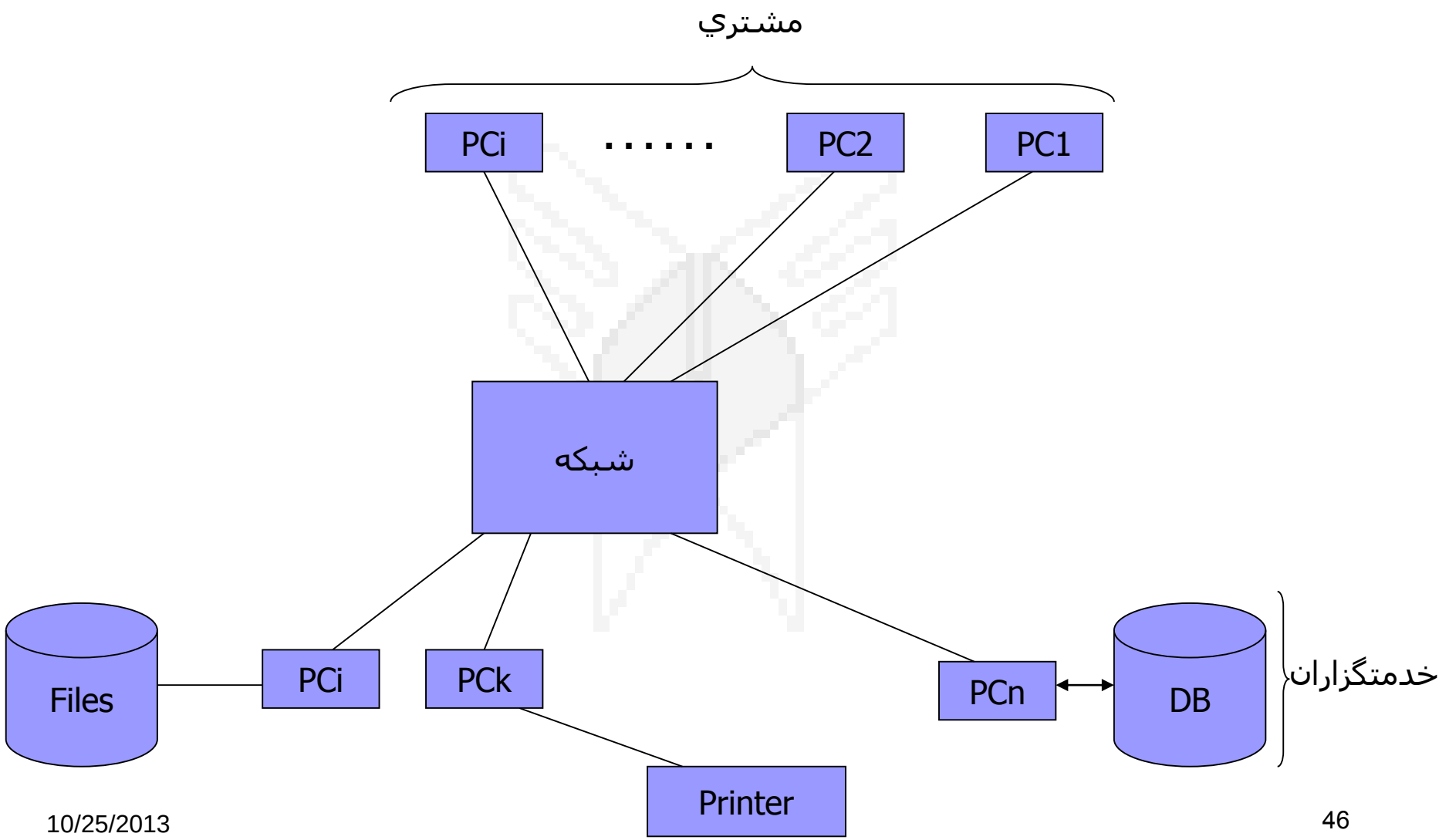


معماري مشتري – خدمتگزار Client / Server

هر معماري كه در آن قسمتي از پردازش را يک برنامه، سيستم يا ماشين انجام دهد و انجام قسمت ديگري از پردازش را از برنامه ، سيستم يا ماشين ديگر بخواهد معماري مشتري خدمتگزار ناميده مي شود



طرحهاي معماري مشتري – خدمتگزار از نظر پيكربندي سخت‌افزاري در شبكه

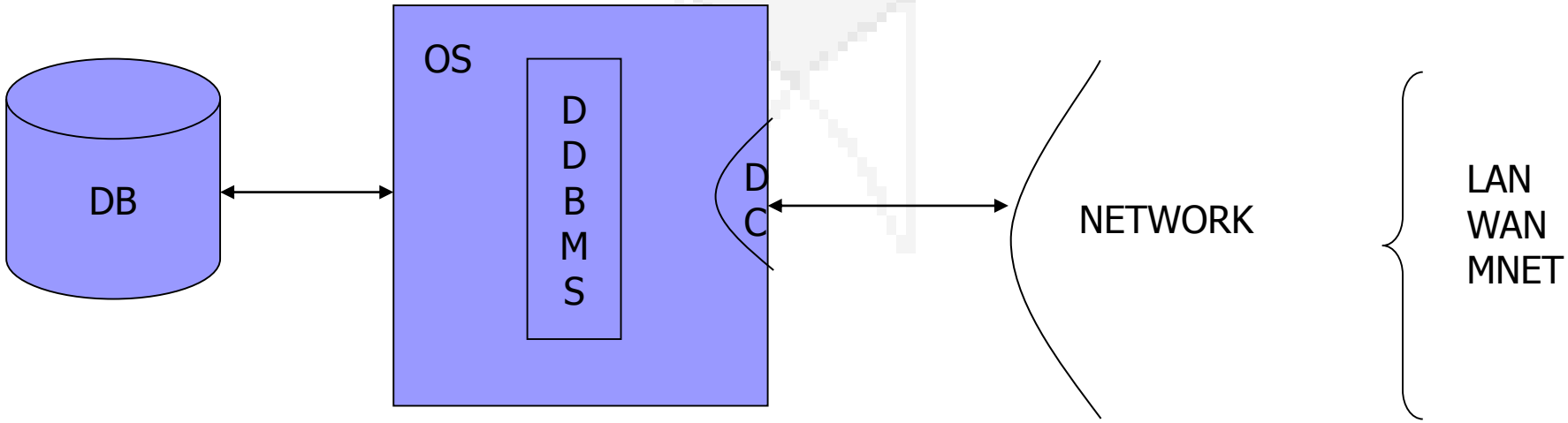


مزاياي معماري مشتري – خدمتگزار در مقايسه با معماري متمركز

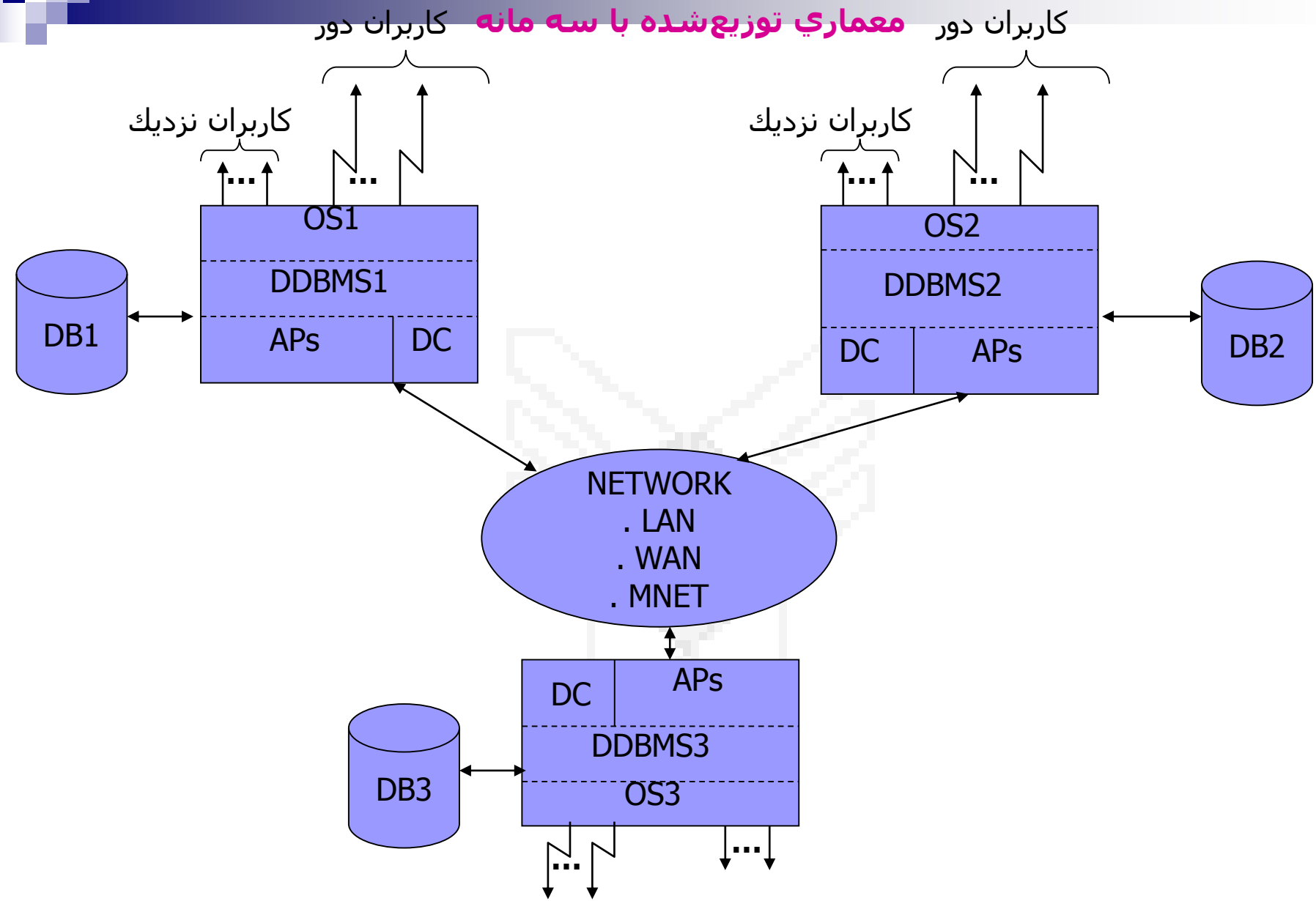
- تقسيم پردازش
- كاهش ترافيك شبكه
- استقلال ايستگاههاي كار
- اشتراك دادهها

مجموعه اي از داده هاي ذخيره شده كه منطقا به يک سيستم تعلق دارند ولي در مانه هاي مختلف يک يا بيش از يک شبکه توزيع شده اند.

نماي يك مانه در معماري توزيع‌شده



معماري توزيع‌شده با سه مانه



DDB = {DB1+DB2+DB3}

ويژگيهاي معماري توزيع‌شده

- مجموعه‌اي است از داده‌هاي منطقا مرتبط و اشتراكي
- داده‌ها به بخشهايي تقسيم و در مانه‌ها توزيع شده‌اند.
- بعضي بخشها ممكن است به طور تكراري در مانه‌ها ذخيره شده باشند.
- مانه‌ها از طريق شبكه بهم مرتبطاند.
- داده‌هاي هر مانه تحت كنترل يك DBMS است.
- DMBS هر مانه، مي‌تواند برنامه‌هاي كاربردي محلي را به طور خودكار اجرا كند.
- هر DBMS حداقل دراجراي يك برنامه كاربردي سرتاسري مشاركت دارد.

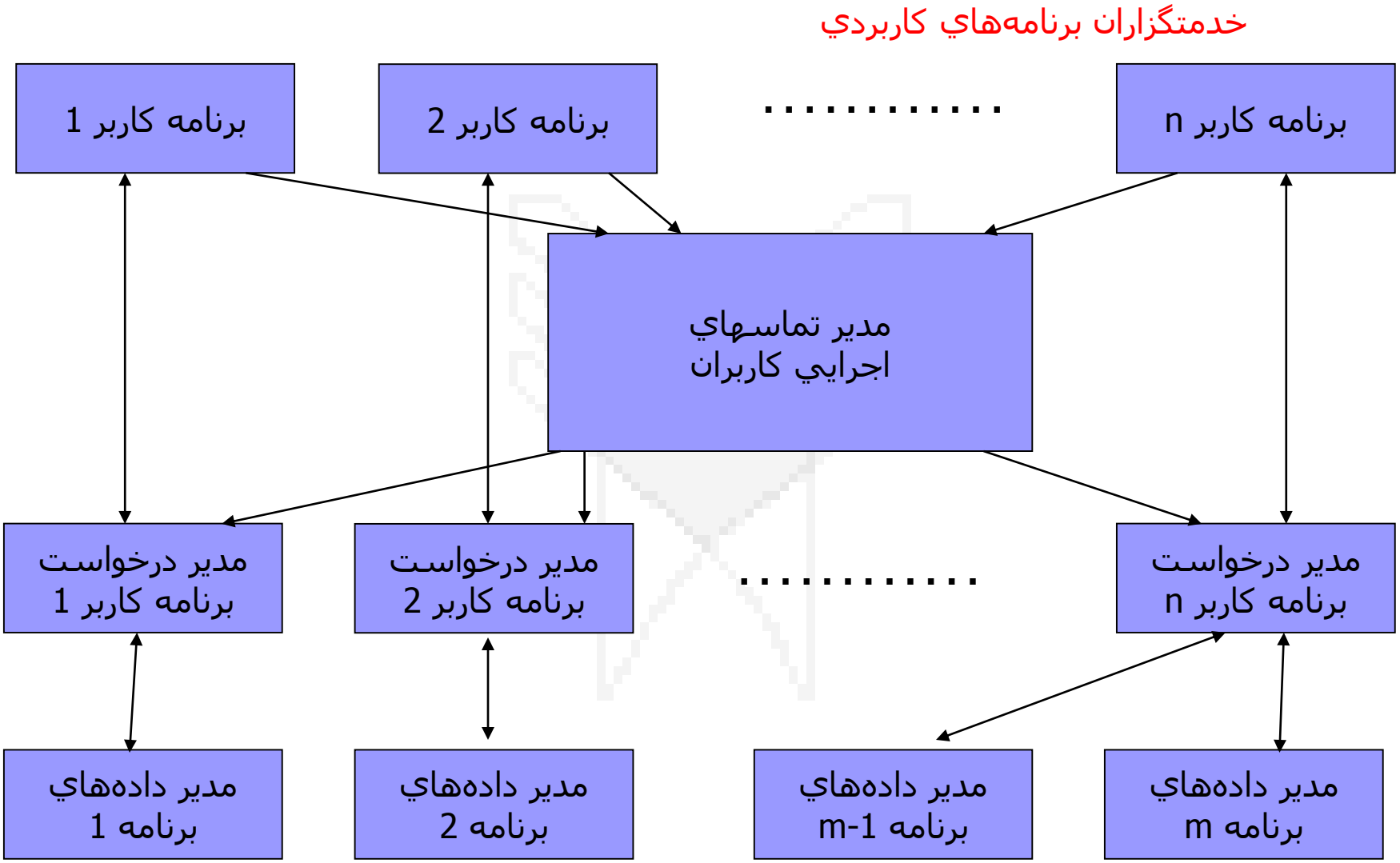
مزاياي معماري توزيع شده

- سازگاري و هماهنگي با ماهيت سازمانهاي نوين.
- كارايي بيشتر در پردازش داده‌ها.
- دستيابي بهتر به داده‌ها.
- اشتراك داده‌ها.
- افزايش پردازش موازي.
- كاهش هزينه ارتباطات.
- تسهيل گسترش سيستم.
- استفاده از پايگاه داده‌هاي از قبل موجود.

معايب معماري توزيع‌شده

- پيچيدگي طراحي سيستم.
- پيچيدگي پياده‌سازي.
- كاهش كارايي در برخي موارد.
- هزينه بيشتر.
- مصرف حافظه بيشتر.

در اين گونه سيستمها معمولا تعداد زيادي تراكنش در ثانيه و بطور موازي اجرا مي‌شود .



نمودار ER (نمودار ارتباط بين موجوديتها / Entity Relationship Diagram) :

در طراحي يك بانك اطلاعاتي ابتدا مي‌بايست مراحلي چون امكان‌سنجي، بررسي نيازها و محدوديتها، بررسي سيستم دستي موجود و غيره صورت گيرد كه اين مراحل در درس تجزيه و تحليل سيستم‌ها مطرح مي‌گردد.
پس از انجام مراحل فوق طراح بانك به كمك نمودارهايي، شماي كلي بانك را مستقل از مدل بانك (جدولي ـ شبكه‌اي ـ سلسله مراتبي) و نيز مستقل از جنبه‌هاي برنامه‌نويسي ترسيم مـــي‌كند. كه يكي از مرسوم ترين روشهاي مدلسازي بامك اطلاعاتي رسم نمودار ER است 0

اين مدل يكي از انواع گوناگون مدلسازي داده براي طراحي بانك اطلاعات كه به خاطر سادگي و كارآيي آن به صورت وسيعي به عنوان يك استاندارد پذيرفته شد و به عنوان مدلي براي تصوير ادراكي عام يك بانك اطلاعات در معماری 4 لایه كه در بخشهاي قبل آنرا مورد مطالعه قرار داديم از اهميت ويژه‌اي برخوردار است. در واقع اگر شما اين مدل را به خوبي فرا بگيريد يك طراح بانك اطلاعات خواهيد بود.

داده‌هاي ذخيره‌شدني در پايگاه داده‌ها ابتدا بايد در بالاترين سطح انتزاع ، مدلسازي معنايي شوند.

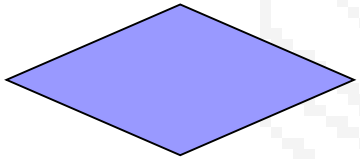
مدل E-R ، بانك اطلاعات را مجموعه‌اي از موجوديت‌ها (Entity) و ارتباطات بين اين موجوديت‌ها (Relationship) مي‌داند.

در سال 1976 چن (Chen) از دانشگاه MIT مدل ER (Entity Relation) را جهت طراحی بانك پيشنهاد كرد. اين مدل در طول زمان پيشرفت كرد و بنام EER = Extended ER معروف گرديد.

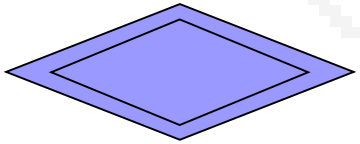
نمادهاي رسم نمودار ER

نوع موجوديت

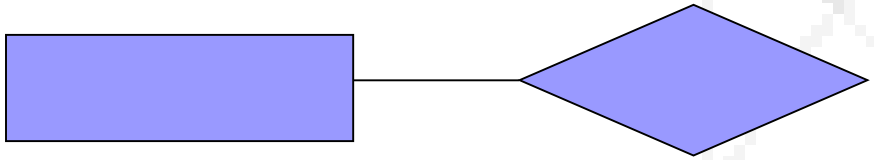
نوع موجوديت ضعيف



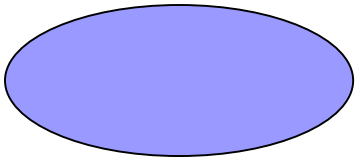
نوع ارتباط



نوع ارتباط با موجوديت ضعيف

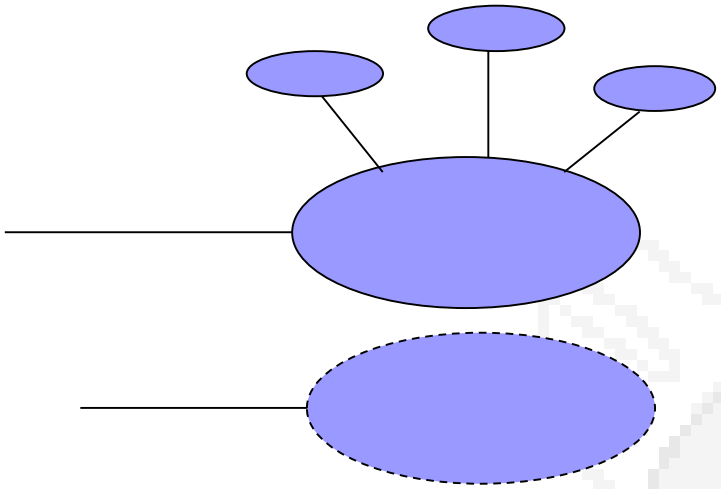


مشاركت نوع موجوديت در نوع ارتباط



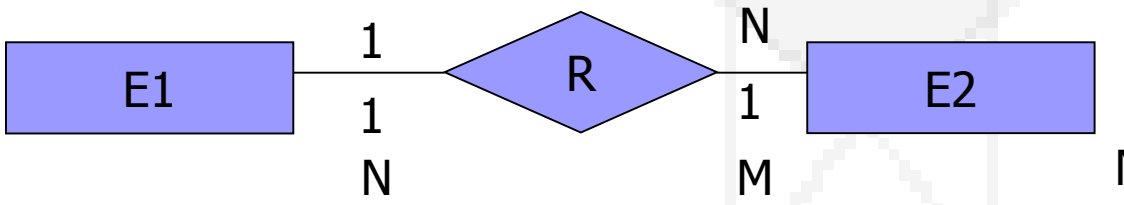
صفت

ادامه : نمادهاي رسم نمودار ER



صفت مرکب

صفت مشتق

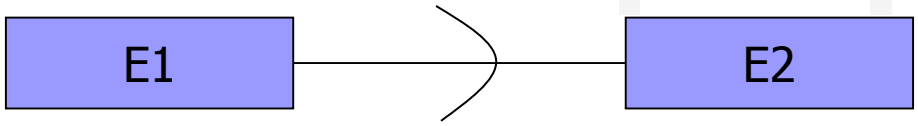


چندي ارتباط

N به 1

1 به 1

N به M



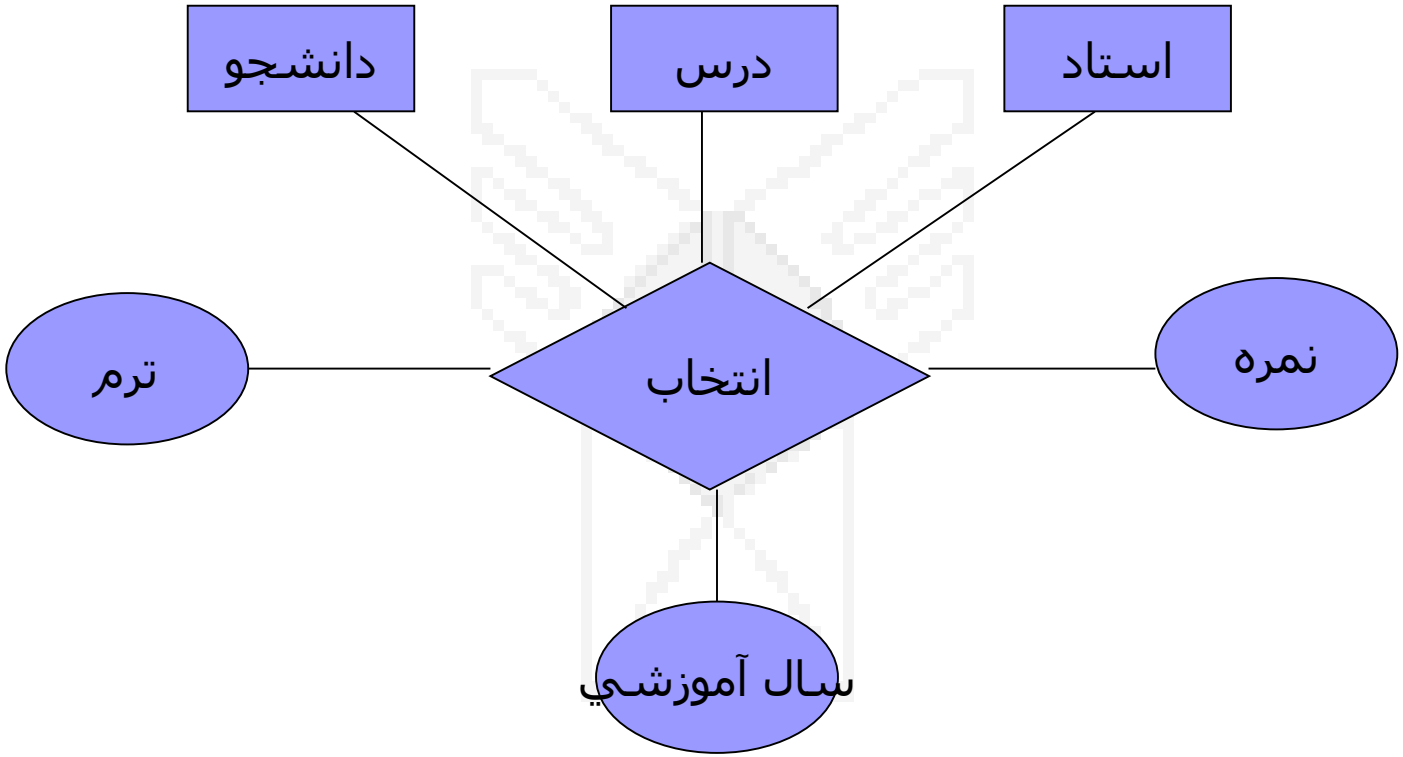
ارتباط "گونه‌اي است از"

وضع مشارکت در ارتباط

مشارکت يك نوع موجوديت در يك نوع ارتباط را الزامي گويند ، اگر تمام نمونه‌هاي آن نوع موجوديت در آن نوع ارتباط شرکت کنند. در غير اين صورت مشارکت غيرالزامي است.

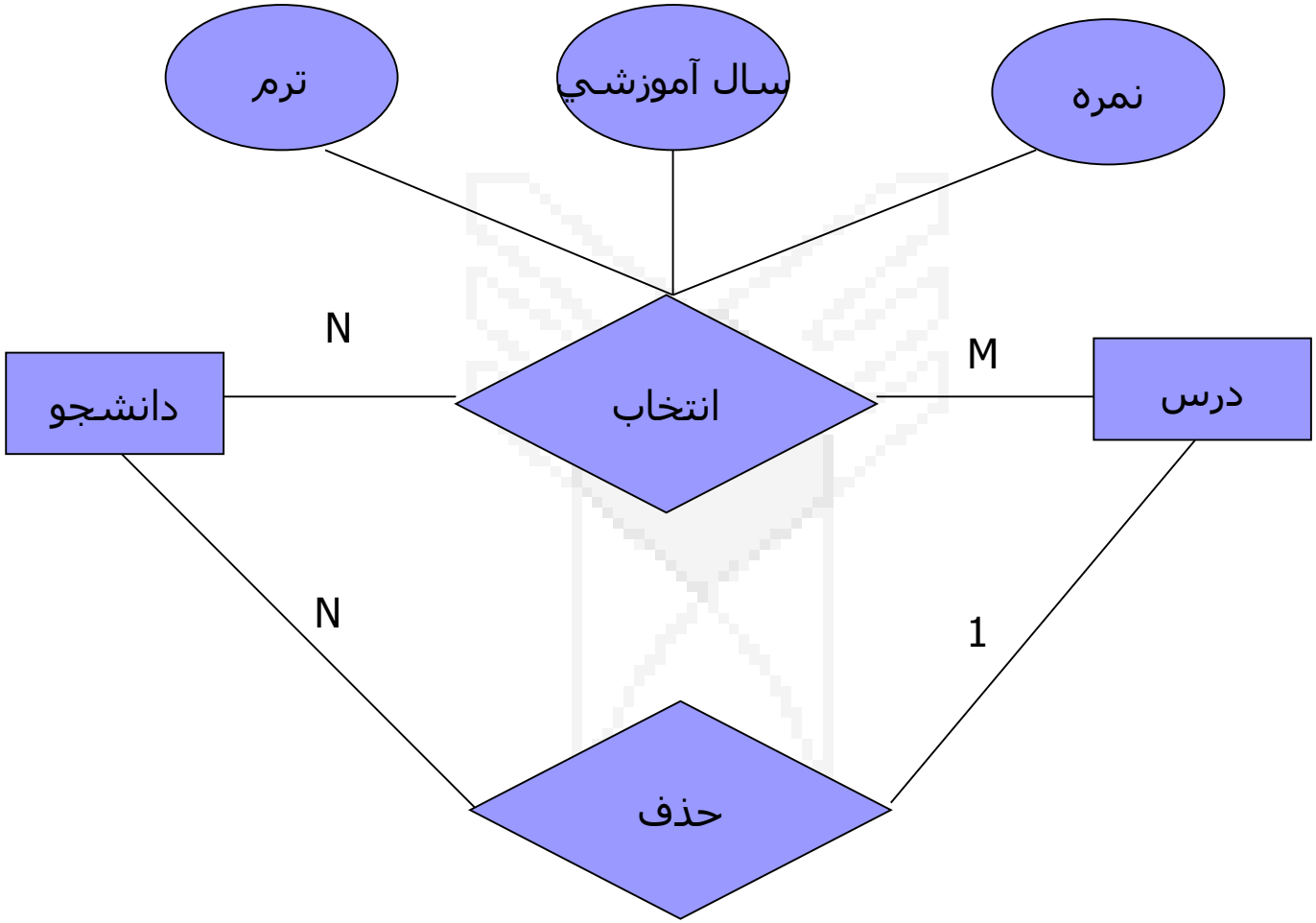
نمايش مشارکت الزامي

تعداد شرکت‌کنندگان در يك ارتباط را درجه آن ارتباط مي‌گويند.



ارتباط بين سه موجوديت

نمايش چندي ارتباط



دلايل اهميت مدل *E-R* :

1- تعريف ساختارهاي ساده، لازم و كافي و غير وابسته به پياده سازي براي طراحي بانك اطلاعات كه ويژگي‌هاي مدل ادراكي عام (Conceptual model) را به خوبي برآورده مي‌كند.

2-تعريف نمادهاي مناسب براي ارائه يك طراحي قابل فهم و ساده

مفاهيم مورد نياز در رسم نمودار ER :

1- تعريف پديده يا موجوديت يا *Entity*:

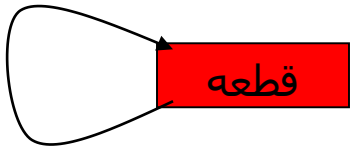
مفهوم كلي شيئ، چيز، پديده و به طور كلي هر آنچه كه مي‌خواهيم در موردش اطلاع داشته باشيم و شناخت خود را در موردش افزايش دهيم.
و يا عبارت است از همه اشياء قابل تميز از بقيه كه در دنياي واقعي وجود دارند. مانند هر فرد كه در يك سازمان كار مي‌كند. وامي كه توسط بانك به مشتريان داده مي‌شود ، مشتري بانك ، شعبه‌هاي بانك و...
هر موجوديت داراي مجموعه‌اي از ويژگي‌ها يا صفات است *(Atributes)* كه زير مجموعه‌اي از اين صفات بايد يك پديده را از ساير پديده‌هاي مشابه مشخص كند كه به اين زير مجموعه كليد مي‌گويند. به عنوان مثال شماره كارمندي براي كارمنديك سازمان آن كارمند را از ساير كارمندان متمايز مي‌كند يا شماره دانشجو در سيستم دانشگاه

توجه : در طراحي بانك اطلاعات شيءاي به عنوان پديده در نظر گرفته مي‌شود كه مجموعه‌اي از آن شيء در دامنه مساله قابل تشخيص باشد و اين يكي از راه‌هاي شناخت موجوديت واقعي است

2- ارتباط *(Relationship)*:

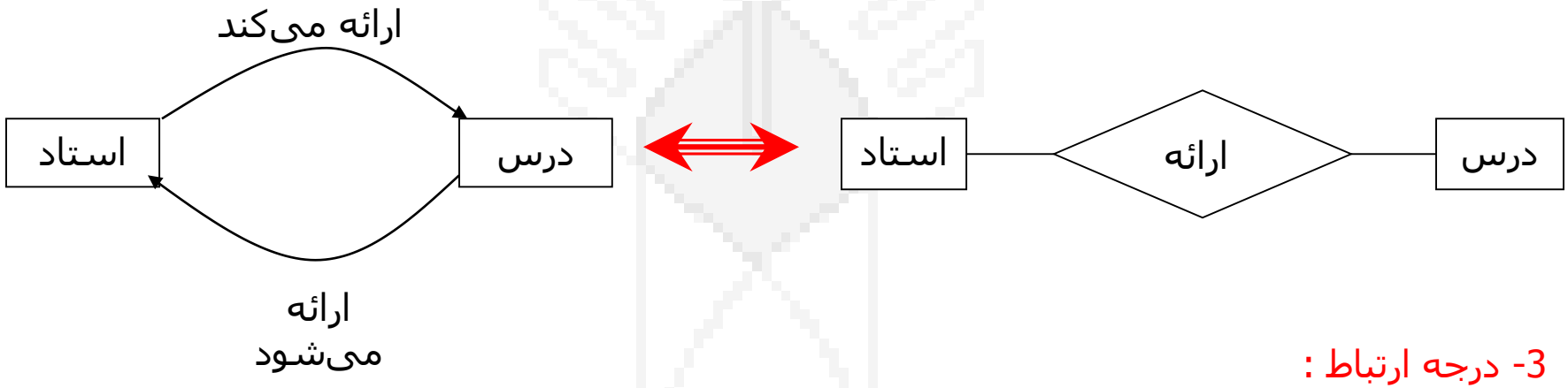
منظور از ارتباط در مدل *E-R* ارتباط موجود بين دو يا چند موجوديت است كه موجوديت‌ها را به هم پيوند مي‌دهد. ارتباط را در مدل *E-R* با يك لوزي نشان مي‌دهند.

ارتباط دو موجوديت دانشجو و درس



ممکن است یك موجودیت با خودش ارتباط داشته باشد.
این بدین معناست که«یك قطعه از قطعه یا قطعات دیگر ساخته شده است».

دو نمودار زیر معادل یکدیگرند :



3- درجه ارتباط :

در ترسیم نمودار ER درجه ارتباط (Relationship Degree) می‌تواند یك به یك (1:1)، یك به چند (1: n) چند به یك (n : 1) یا چند به چند (n : n) باشد.

ارتباط يك به يك :

در اين شكل هر استاد يك درس و هر درس فقط توسط يك استاد ارائه می‌شود. البته ممكن است استادی اصلاً درس نداشته باشد يا درسی توسط هيچ استادی اين ترم ارائه نگردد.

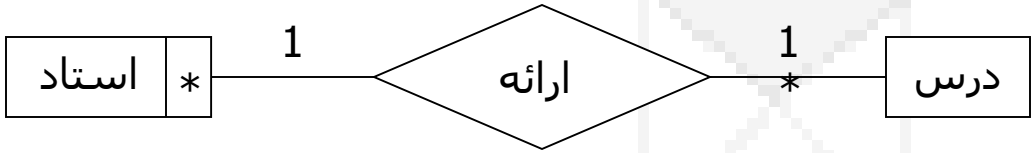
ارتباط چند به يك :

در اين شكل چند استاد ممكن است يك درس را ارائه كنند ولی هر استاد فقط يك درس را ارائه می‌كند.

ارتباط چند به چند :

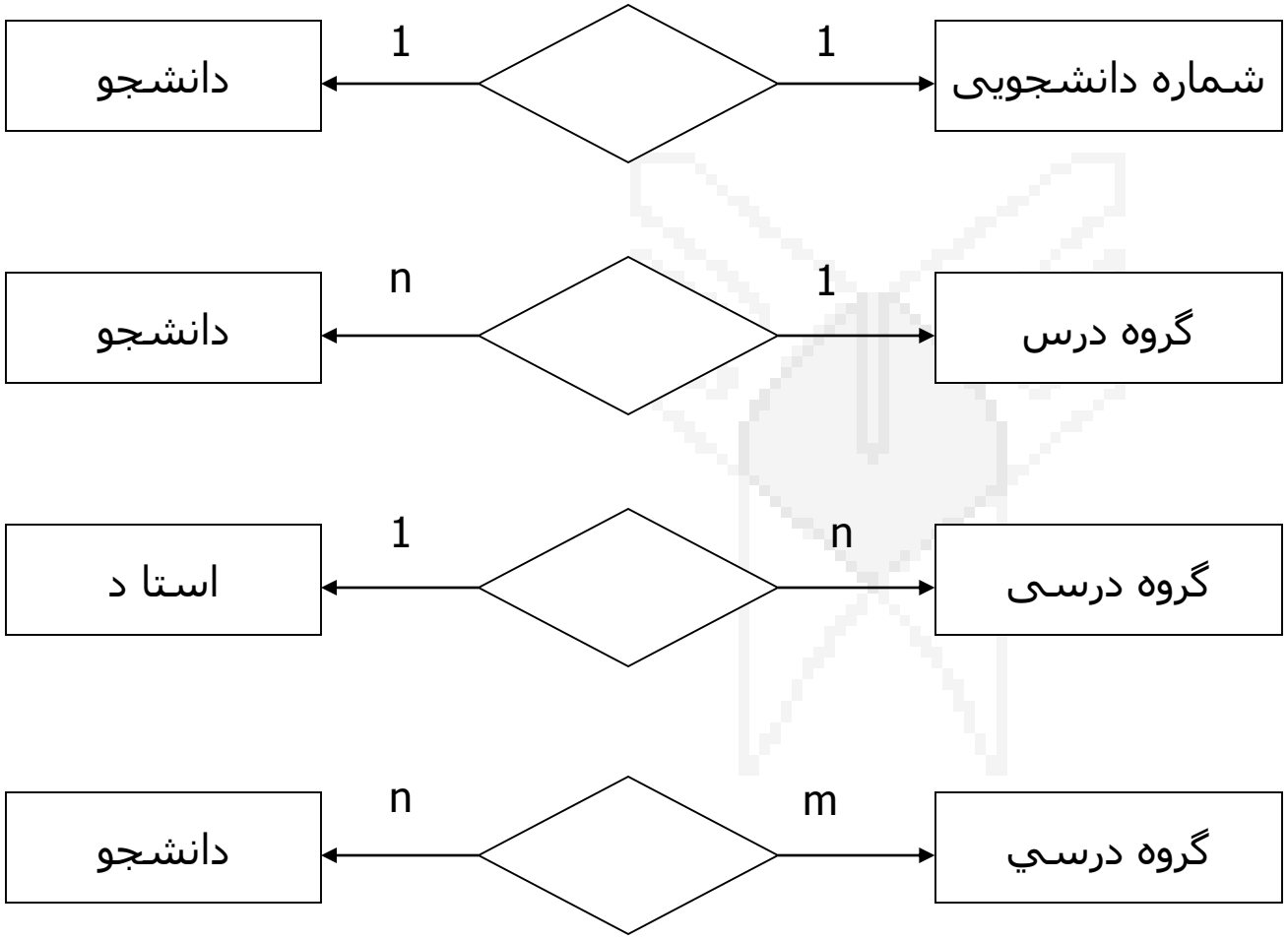
در این شکل هر درس ممکن است توسط چند استاد ارائه شود و هر استاد ممکن است چند درس مختلف را ارائه کند.

در شکلهای فوق شرکت موجودیتها در ارتباط اختیاری بود یعنی ممکن بود استادی درسی ارائه نکند و یا درسی این ترم ارائه نشود. در نمودار ER برای اینکه شرکت در ارتباط اجباری شود علامت * به جای روی خط ارتباط، داخل مستطیل موجودیت ترسیم می‌شود.

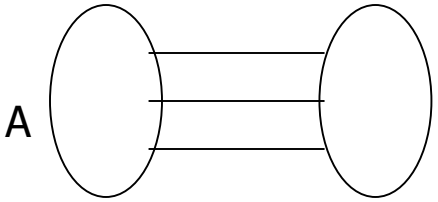


در این شکل هر استاد حتماً باید درسی ارائه کند (فقط یك درس) و هردرس فقط توسط یك استاد ارائه می‌شود (البته ممکن است درس خاصی ارائه نشود) ولی استادها نمی‌توانند درس ندهند.

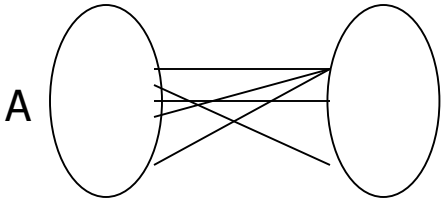
یک مثال کلی از نمودار ER :



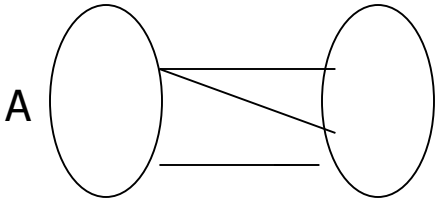
ارتباط یک به یک:



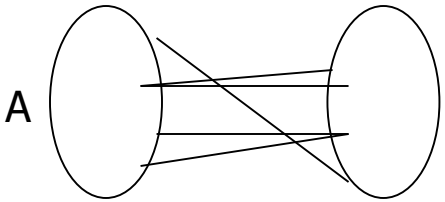
ارتباط چند به یک:

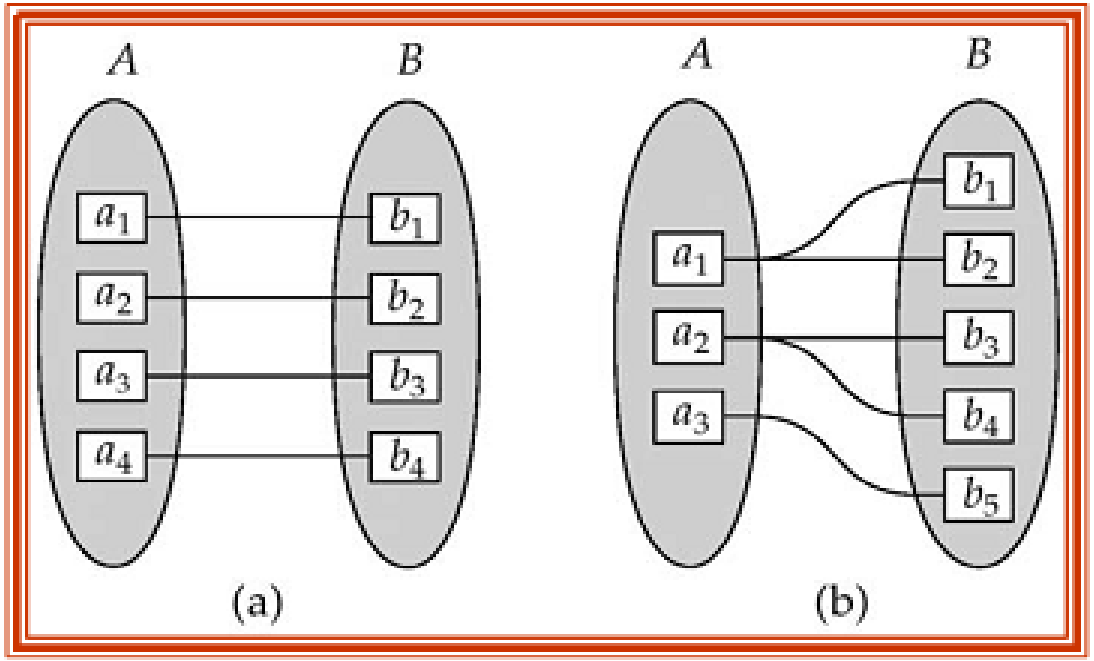


ارتباط یک به چند:



ارتباط چند به چند:



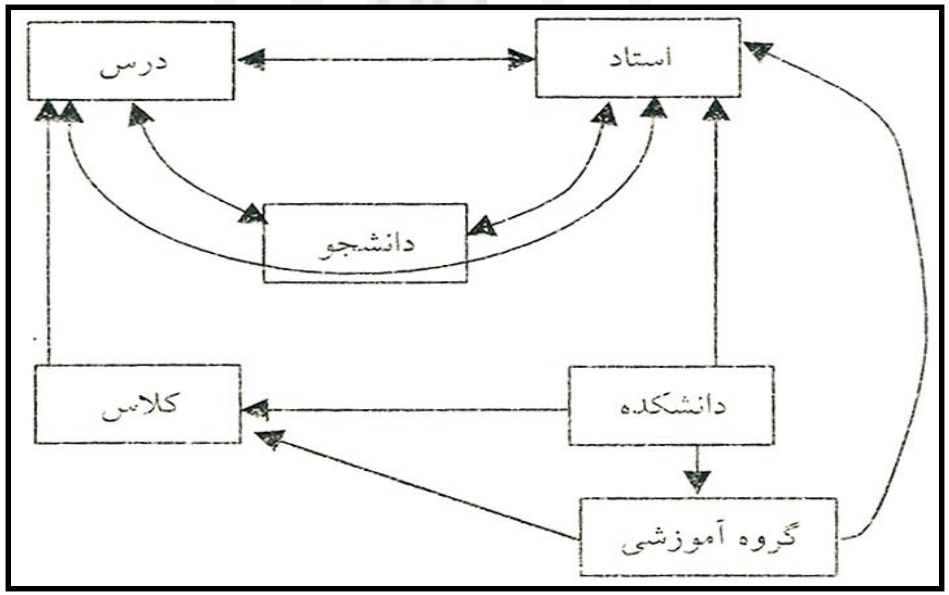


ارتباط یک به یک:

ارتباط یک به چند:

ارتباط بین موجودیتها به تعبیری خود یك نوع موجودیت است زیرا با توجه به تعریف موجودیت (پدیده شیء یا چیزی که می‌خواهیم در موردش اطلاع داشته باشیم) وجود ارتباط نیز پدیده‌ای است که باید در مورد آن اطلاعات در بانك داشته باشیم.
پس بانك اطلاعاتی به تعبیری مجموعه‌ای از اطلاعات در مورد موجودیتها یك محیط عملیاتی و ارتباط بین آنها می‌باشد.

در شکل زیر نمودارER یك دانشکده ترسیم شده است :



در سال 1976 چن (Chen) از دانشگاه MIT مدل ER (Entity Relation) را جهت طراحی بانك پیشنهاد کرد. این مدل در طول زمان پیشرفت کرد و بنام EER = Extended ER معروف گردید.

در این طراحی کلی موجودیت با مستطیل، صفتها به صورت بیضی و ارتباط (Relationship) بصورت لوزی ترسیم می‌شوند.

یک مثال از نمودار EER :

موجودیتها :

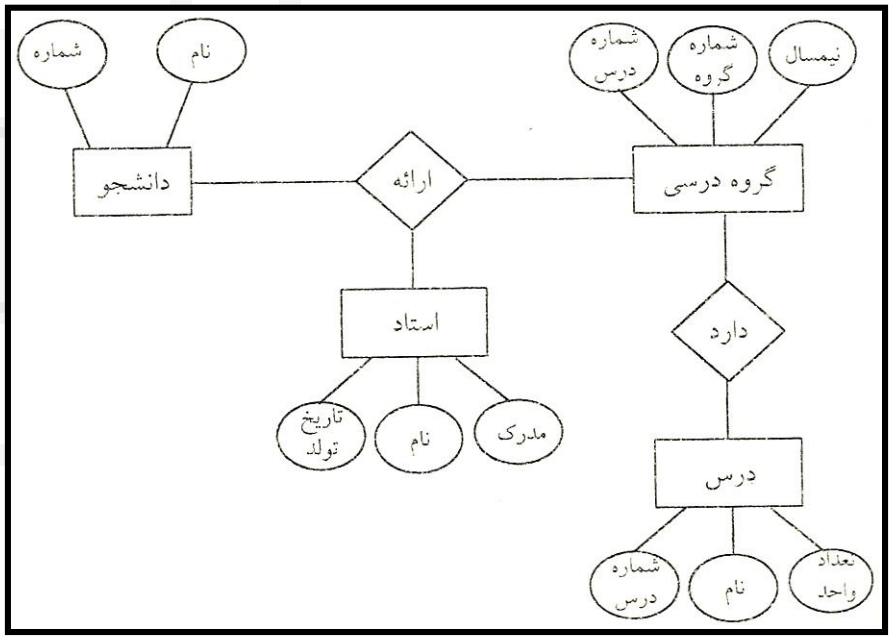
- دانشجو
- استاد
- گروه درسی
- درس

ارتباط :

- ارائه
- دارد

صفت ها :

- نام دانشجو
- شماره دانشجوئي
- نام استاد
- مدرک
- تاریخ تولد
- نیمسال
- شماره گروه
- شماره درس
- نام درس
- تعداد واحد
- شماره درس



انواع صفت در نمودار EER :

الف) صفت کلیدی

کلید عبارت است از يك يا چند صفت که در يك موجوديت منحصربه فرد باشد. مثلاً در موجوديت دانشجو شماره دانشجويي کليد است. چون هر دانشجو يك شماره يکتا دارد. ولی نام نمی‌تواند کليد باشد. گاهی اوقات يك صفت تنها نمی‌تواند کليد باشد بلکه مجموعه‌ای از دو يا چند صفت با همديگر کليد می‌شوند. مثلاً نام و شماره شناسنامه هر يك به تنهايي کليد نيست ولی هر دو با هم کليد می شوند. برای مشخص کردن کليد يك موجوديت زير آن صفت خط می‌کشيم. (نام، شماره دانشجويي,نام پدر...)

ب) صفت ساده و مرکب

بعضی از صفتها ساده هستند مثل شماره دانشجويي ولی بعضی از صفتها مرکب (تجزيه پذير) هستند مثل آدرس که خود از صفتهای شهر، خيابان، کوچه و پلاك تشکيل يافته است. در واقع صفت مرکب صفتی است که هم خودش معنی‌دار است و هم بخشهايي از آن 0

در بانك اطلاعاتی رابطه‌ای صفت مرکب نداريم.

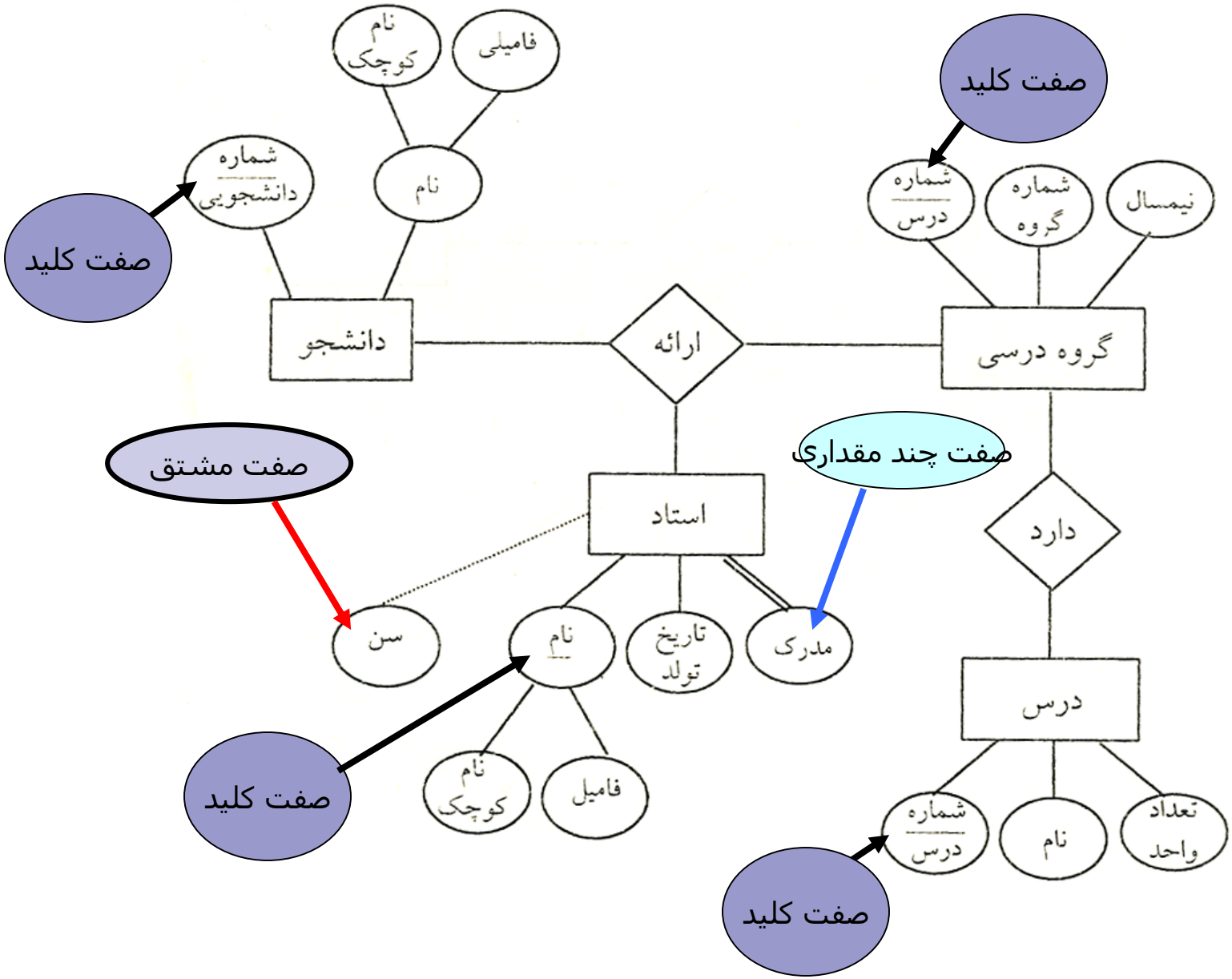
ج) صفت تك مقداری یا چند مقداری

مثلاً در موجودیت استاد نام تك مقداری است چون هر استاد فقط یك نام دارد ولی صفت مدرك چند مقداری است چون استاد ممکن است چندین مدرك داشته باشد. صفتهای چند مقداری را در مدل EER با دو خط ترسیم می‌کنیم.

در مدل رابطه‌ای صفت چند مقداری نداریم.

د) صفت مشتق

صفت مشتق ، صفتی است که به کمك صفتهای دیگر می‌توان آن را محاسبه کرد. مثل سن استاد که با توجه به تاریخ تولد قابل محاسبه می‌باشد. تصمیم‌گیری درمورد صفت مشتق به عهده طراح است مثلاً معدل کل برای دانشجو بهتراست مشتق باشد زیرا مرتباً با گذراندن دروس بیشتر عوض می‌شود ولی برای فارغ‌التحصیلان معدل کل بهتر است بخشی از پدیده باشد. صفت مشتق در نمودار EER بصورت خطچین ترسیم می‌شود.


صفت کلید
نام کوچک
فامیلی
شماره دانشجویی
نام
صفت کلید
دانشجو
ارائه
شماره درس
شماره گروه
نیمسال
گروه درسی
صفت مشتق
صفت چند مقداری
استاد
دارد
سن
نام
تاریخ تولد
مدرک
درس
صفت کلید
نام کوچک
فامیل
شماره درس
نام
تعداد واحد
صفت کلید

مزايا ي شيوه استفاده از بانك اطلاعاتی:

1) كاهش افزونگی داده ها
2) اجتناب از ناسازگاری داده ها
3) اشـــتـراك داده هـا
4) اعمال محدوديتهای امنيتی
5) اعمال جامعيت و يكپارچگی داده ها
6) متعادل نمودن نيازمندی های متضاد كه بر عهده DBA است
7) اعمال استانداردها

1ـ کاهش افزونگی داده ها :

وجود يك اطلاع در بيش از يك جا را افزونگی داده ها ميگويند. عمدتا" افزونگی قابل حذف 100% نيست ولي حتي الامكان بايد آنرا به سمت صفر ميل داد0
افزونگي هاي غير قابل حذف (طبق نظر DBA) در يك سيستم بانك اطلاعاتي بايد كنترل گردد كه به اين افزونگی ، افزونگی كنترل شده گويند . (كنترل كردن يعنی يكي نگهداشتن يك اطلاع افزونه)

انـواع افـزونگی :

1ـ افزونگی مستقيم : يك قلم داده ای عيناً در جای ديگر تكرار شود .
2ـ افزونگی غير مستقيم : يك قلم داده اي بطور مستقيم در جاي ديگر تكرار نشود و بتوان آن را از ديگرداده ها نتيجه گرفت مثل معدل يا مساحت (چون طول وعرض را داريم) يا سن 0
عمدتا" وجود افزونگی غيرمستقيم برای افزايش سرعت است .

نمونه افزونگی داده ها :

File Window Help

stud_no	name	famil	shenas	codemeli	reshteh	Expr1	term	dars	Expr2	nomreh
82400047	خديجه	حاتمي پور	98	2219117588	24090601	پرستاري	831	228673	تمرين تدريس	18.25
82400018	ليلا	زارعي	442	2092401572	24090601	پرستاري	831	228673	تمرين تدريس	18.5
82400101	فائزه	رضويان	25901	35726733	24090601	پرستاري	831	228673	تمرين تدريس	18.75
82400051	مهناز	گودرزي	7797	55544460	24090601	پرستاري	831	228673	تمرين تدريس	19
82400092	افسانه	مظاهري	1043	533141095	24090601	پرستاري	831	228673	تمرين تدريس	19
81400237	ليلا	قديانی	4827	57291756	24090601	پرستاري	821	228673	تمرين تدريس	19.25
81400183	جمال	صيدي	308		24090601	پرستاري	821	228673	تمرين تدريس	19.5
82400099	افسانه	صدوقي اصل	2786		24090601	پرستاري	831	228673	تمرين تدريس	19.5
82400074	رقيه	صائب نيا	306	2594260381	24090601	پرستاري	831	228673	تمرين تدريس	19.5
81400187	طاهره	پيمانی	6777	66994713	24090601	پرستاري	821	228673	تمرين تدريس	19.6
81400175	فرشته	قلجايی	19727	3670196102	24090601	پرستاري	821	228673	تمرين تدريس	19.6
81400201	فاطمه	منجذبی	5868		24090601	پرستاري	821	228673	تمرين تدريس	19.75
82400127	خديجه	توسلی	6560		24090601	پرستاري	831	228673	تمرين تدريس	19.75
82400028	فرناز	مرادعليزاده	672	3071084374	24090601	پرستاري	831	228673	تمرين تدريس	20
78400431	علمدار	ده نمکی کبير	15		24090601	پرستاري	782	228691	روش هاوفنون تدريس	14
82400047	خديجه	حاتمي پور	98	2219117588	24090601	پرستاري	822	228691	روش هاوفنون تدريس	15
82400035	مريم	عزيزي	122	6649568015	24090601	پرستاري	822	228691	روش هاوفنون تدريس	15.5
81400181	شيما	نظري	9859	61852074	24090601	پرستاري	812	228691	روش هاوفنون تدريس	15.7
82400018	ليلا	زارعي	442	2092401572	24090601	پرستاري	822	228691	روش هاوفنون تدريس	15.75
81400177	ويدا	قدرتی جابلو	7580	65148770	24090601	پرستاري	812	228691	روش هاوفنون تدريس	16.2
78401237	سامره	عبدلی	207		24090601	پرستاري	782	228691	روش هاوفنون تدريس	17
81400173	زهرا	طاهري ازبرمی	3606	5179056942	24090601	پرستاري	812	228691	روش هاوفنون تدريس	17.5
79400580	انسيه	جمشيدي	2114	5309581073	24090601	پرستاري	792	228691	روش هاوفنون تدريس	17.5
81400193	فريده	فرهی قهی	4416	55029523	24090601	پرستاري	812	228691	روش هاوفنون تدريس	17.6
76400521	مروت	گيوي	907	3960896212	24090601	پرستاري	762	228691	روش هاوفنون تدريس	17.75
82400028	فرناز	مرادعليزاده	672	3071084374	24090601	پرستاري	822	228691	روش هاوفنون تدريس	17.75
81400187	طاهره	پيمانی	6777	66994713	24090601	پرستاري	812	228691	روش هاوفنون تدريس	17.8
80400027	ناصر	دين محمدي	310	4284237039	24090601	پرستاري	802	228691	روش هاوفنون تدريس	18
81400201	فاطمه	منجذبی	5868		24090601	پرستاري	812	228691	روش هاوفنون تدريس	18
82400099	افسانه	صدوقي اصل	2786		24090601	پرستاري	822	228691	روش هاوفنون تدريس	18
82400051	مهناز	گودرزي	7797	55544460	24090601	پرستاري	822	228691	روش هاوفنون تدريس	18
82400101	فائزه	رضويان	25901	35726733	24090601	پرستاري	822	228691	روش هاوفنون تدريس	18.25
82400127	خديجه	توسلی	6560		24090601	پرستاري	822	228691	روش هاوفنون تدريس	18.25
81400237	ليلا	قديانی	4827	57291756	24090601	پرستاري	812	228691	روش هاوفنون تدريس	18.3
81400183	جمال	صيدي	308		24090601	پرستاري	812	228691	روش هاوفنون تدريس	18.3
81400175	فرشته	قلجايی	19727	3670196102	24090601	پرستاري	812	228691	روش هاوفنون تدريس	18.4

نمونه افزونگی داده ها :

کد دانشجويي	نام	فاميلی	ش.ش	سال تولد	نام پدر	آدرس	تلفن	نام درس	نمره
2	علی	محمدی	11	1360	محمد	رشت	2332	رياضي	13
5	محمد	ناصری	11	1362	رضا	زنجان	3251	رياضي	14
6	عليرضا	كاظمي	354	1358	حسين	انزلي	2121	فيزيك	18
5	محمد	ناصری	11	1362	رضا	زنجان	3251	فيزيك	15
6	عليرضا	كاظمي	354	1358	حسين	انزلي	2121	شيمي	17
2	علی	محمدی	11	1360	محمد	رشت	2332	فيزيك	16
6	عليرضا	كاظمي	354	1358	حسين	انزلي	2121	شيمي	19
6	عليرضا	كاظمي	354	1358	حسين	انزلي	2121	فارسي	16
3	رضا	رضايی	451	1360	احمد	تبريز	2324	فارسي	9

نمونه تصحيح شده در جهت كاهش افزونگي داده ها :

كد دانشجويي	نام	فاميلی	ش.ش	سال تولد	نام پدر	آدرس	تلفن
2	علی	محمدی	11	1360	محمد	رشت	2332
5	محمد	ناصری	11	1362	رضا	زنجان	3251
6	عليرضا	كاظمي	354	1358	حسين	انزلي	2121
3	رضا	رضايی	451	1360	احمد	تبريز	2324

كد دانشجويي	نام درس	نمره
2	رياضي	13
5	رياضي	14
6	فيزيك	18
5	فيزيك	15
6	شيمي	17
2	فيزيك	16
6	عربي	19
6	فارسي	16
3	فارسي	9

2ـ اجتناب از ناسازگاری داده ها :

در يك سيستم بانك اطلاعاتی امكان نگهداري يك اطلاع در بيش از يك جا وجود دارد (افزونگي داده ها) ، در صورتی كه به هر دليلی يكی از داده ها دچار تغيير شود و داده نگهداری شده در محل دوم را به روز رسانی نشود ، دچار ناسازگاری داده می شويم .

3ـ اشـــتـراك داده هـا :

جلوگيري از تكرار داده هاي مشترك بين بخشهاي مختلف يك سيستم يكپارچه 0
بعنوان مثال : داده هاي مشترك بين سيستم حقوق و دستمزد و گارگزيني

4ـ اعمال محدوديتهای امنيتی :

قرار دادن اطلاعات به صورت اشتراكي ، متضمن اعمال مقررات امنيتي خاص خواهد بود0

شماي بانك اطلاعاتي (Schema):

تشريح كلي پايگاه داده ها بدون درنظر گرفتن در نظر گرفتن محتويات آن را شماي بانك اطلاعاتي گويند0

بعنوان مثال : تعداد جداول و فيلدها و روابط بين جداول به شماي بانك اطلاعاتي مرتبط است ولي تعداد ركوردها و محتويات آن به شماي بانك اطلاعاتي مرتبط نيست 0

از جمله ابزارهايي كه در سيستمهاي فايلينگ وجود ندارد اما بانك اطلاعات آنها را پشتيباني مي‌كند، لغتنامه داده و كاتالوگ سيستم است.

لغتنامه (Data Dictionary) :

لغتنامه داده‌ها شبیه لغتنامه‌های معمولی، تمامی اسامی استفاده شده در سیستم و معنای آنها را در بر می‌گیرد. در مرحله طراحی بانك اطلاعات هرگاه طراح برای مفهومی نامی انتخاب می‌کند، باید آن را در لغتنامه داده‌ها همراه با معنای آن و فرمت آن وارد کند. این اسامی شامل تمامی نامهای جداول، شیء‌ها، صفتها و غیره است. در بانکهای جدید نرم‌افزار ویژه‌ای برای کار با لغتنامه‌ها وجود دارد که به کمك آن می‌توان اسامی را وارد یا جستجو کرد. این نرم‌افزارها از اشتباهاتی نظیر وارد کردن یك نام با دو معنای مختلف **(Homonym)** و یا دو نام برای یك مفهوم **(Synonym)** جلوگیری می‌کنند.

كاتالوگ سيستم (System catalog) :

كاتالوگ سيستم **:**علاوه بر اسامی داده‌ها ، اطلاعات دیگری باید در مورد بانك نگهداری شود مثل اطلاعات مربوط به حق دستیابی افراد به داده‌های مختلف، تاریخ ایجاد و یا تغییر داده‌ها، تعداد نسخه‌های هر پرونده، اندازه هر جدول یا شیء و غیره. اینگونه اطلاعات در کاتالوگ سیستم نگهداری می شود.

در واقع لغتنامه داده‌ها زیر مجموعه کاتالوگ سیستم است ولی بدلیل کاربرد ویژه آن مجزا شده و برای کار با آن، نرم‌افزار خاصی طراحی شده است. سیستم DBMS به طور خودکار و به کمك کاربران اطلاعات موجود در کاتالوگ سیستم را همواره به روز نگه می‌دارد.

فراداده (Meta Data)

اطلاعات موجود در دیکشنری داده‌ها اصطلاحاً فراداده (Meta Data) می‌گویند که به معنی داده هایي در مورد دیگر داده ها است.

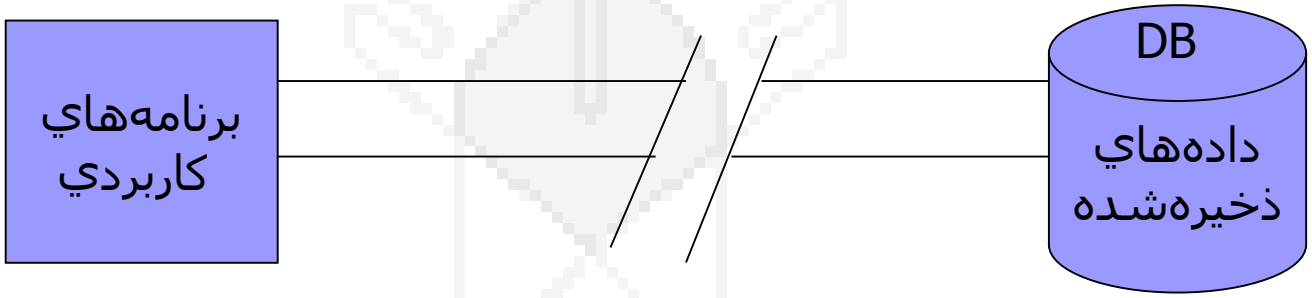
امنیت و جامعیت :

- امنیت یا Security : به معنای محافظت در برابر خطراتی از قبیل آتش‌سوزی ونیز جلوگیری از دستیابی غیرمجاز آنهاست. راه‌های مختلفی برای جلوگیری از دستیابی غیرمجاز به داده‌ها مثل استفاده ازرمزعبور(Password) وجود دارد ، ولی همواره ممکن است افرادی پیدا شوند و این رمزها را بگشایند.

- جامعیت (Integrity) : به معنای صحت داده‌ها و پردازش‌ها و پیروی از مقررات سیستم است. مثلاً موجودی واقعی حسابهای بانکی نباید منفی شود و یا شخص نتواند بیش از موجودی خود از حساب برداشت کند.

استقلال داده‌ها Data Independence :

منظور از استقلال داده‌ها ، مستقل بودن ذخیره‌سازی داده‌ها از برنامه هاي كاربردي ست.
نحوة ذخیره‌سازی داده‌ها روی رسانه‌ها از دید کاربران مخفی است.



استقلال داده‌ها به دو صورت فیزیکی و منطقی تعبیر می‌شود.

1 - استقلال فیزیکی داده‌ها (Physical Data Independence) :

عبارتست از مصونيت ديدهاي كاربران و برنامه‌هاي كاربردي در قبال تغييرات در سطح داخلي-فيزيكي پايگاه داده‌ها يعني :
اگر تغییری درنوع رسانه ذخیره‌سازی داده‌ها انجام گیرد (مثلاً نوع دیسك عوض شود) برنامه‌های کاربردی هیچ تغییری نمی‌کنند.

2 - استقلال منطقی داده‌ها (Logical Data Independence) :

يعني مصونيت ديدهاي كاربران و برنامه‌هاي كاربردي در قبال تغييرات درسطح ادراكي پايگاه داده‌ها
به عبارت دیگر:حتي الامكان تغيير تصوير ادراکی بانك ، ازدید کاربران و برنامه‌های آنها مخفی بماند.
مثلاً اگر جدولی چهار ستون (صفت خاصه) داشته و برنامه‌هايي روی آن ستونها نوشته شده باشد ، در صورتی که ستون پنجمی به آن جدول ،اضافه شود برنامه‌های سابق نیاز به دستکاری ندارند و با همان شکل قبلی قابل اجراء هستند.

تفاوت DA با DBA :

مدیر داده‌ها DA یا (Data Administrator) : شخصی است که کنترل مرکزی داده‌ها را در سازمان به عهده دارد. این فرد لازم است مفهوم داده‌ها را درك كند و نیاز موسسه به داده‌ها را در سطح مدیریت عالی قرار دهد. مدیر داده‌ها تصمیم می‌گیرد که چه داده‌هايي از همان اول در بانك اطلاعاتی قرار گیرد و پس از ذخیره آنها ، سیاستهايي را برای دستیابی به آنها تنظیم کند. توجه کنید که مسئول داده‌ها يك مدیر است نه يك نفر فنی کامپیوتري .

مدیر بانك اطلاعاتی DBA یا (Data Base Administrator) : يك شخص فنی است که مسئول پیاده‌سازی تصمیمات مدیر داده‌هاست. DBA برخلاف DA، يك فرد حرفه‌ای در تکنولوژی اطلاعات (IT) می‌باشد. وظیفه DBA ایجاد بانك اطلاعاتی ، پیاده سازی و کنترلهای فنی است که سیاستگذاری مدیر داده (DA) را اعمال کند. همچنین DBA می‌بایست تضمین کند که سیستم با کارايي قابل قبولی کار کند. DBA مجموعه‌ای از برنامه‌نویسان و سایر افراد فنی را در اختیار دارد.

تفاوت بین DBA و DBMS :

DBA : فرد یا گروهی از افراد که متخصص در امور Data Base هستند که وظیفه طراحی ساختار بانك اطلاعاتی و قوانین مربوط را بعهده دارند.
DBMS : يك نرم افزار است که کار مدیریت کل سیستم بانك اطلاعاتی انجام داده و فرامین DBA را اجراء میکند DBA 0 درخواستهايي كاربر را دريافت و اعتبارسنجی میکند و صحت آنها بررسي ، سپس دستورات كاربر (رويت ، اضافه ، حذف ، تغيير داده ها) را اجراء ميكند .
نسبت DBA به DA مثل نسبت راننده به اتومبیل است
در سيستم هاي امروزي بدنبال اين هستيم كه هر چه بيشتر كنترل را از انسان بگيريم و به ماشين بسپاريم (به علت كاهش خطا)

وظایف DBA :

1ـ تعریف شمای مفهومی(دید ادراکی) .

2ـ تعریف شمای داخلی (طراحی فیزیکی همواره بعد از طراحی منطقی انجام می‌شود) .

3ـ مرتبط بودن با کاربر ، DBA باید با کاربران ارتباط برقرار کند تا اطمینان حاصل کند که داده‌های مورد نیاز آنها وجود دارد و شمای خارجی مورد نیاز را به کمك DDL خارجی بنویسد. سایر جنبه‌های ارتباط با کاربرعبارتند از: مشورت درطراحی برنامه‌های کاربردی، تهیه آموزشهای تکنیکهای شناسایي مساله و ارائه راه‌حلهای آن .

4ـ تعریف محدودیت‌های جامعیت و امنیتی .

5ـ تعریف سیاستهای ترمیم و پشتیبانی .

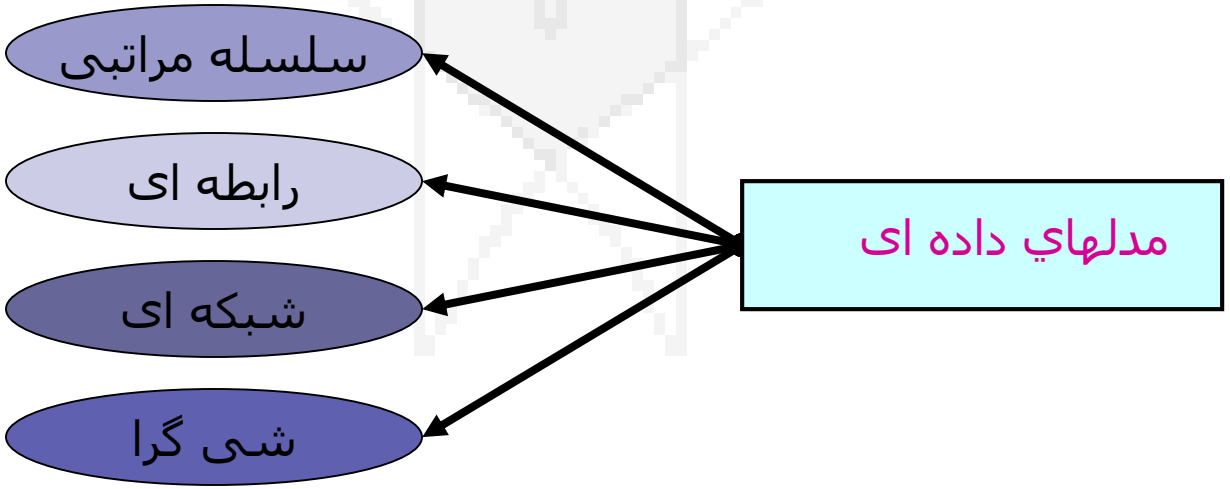
6ـ نظارت بر کارایي و پاسخ به تغییر نیازها .

وظایف DBMS :

1ـ تعریف داده ها .
2ـ دستکاری داده ها (تغییر داده ها توسط نرم افزارها نوشته شده اند)
3- بهینه سازی و اجراء .
4ـ جامعیت و امنیت دیتا ، توسط **DBA** صادر می شود و توسط**DBMS** اجرا می شود .
بررسی درخواستهای کاربر از نظردرستی می‌تواند در زمان کامپایل ، زمان اجراء یا هر دو زمان انجام شود.
5ـ فرهنگ داده ها .
6- کارایي .
7- ترمیم و سازگاری داده ها .

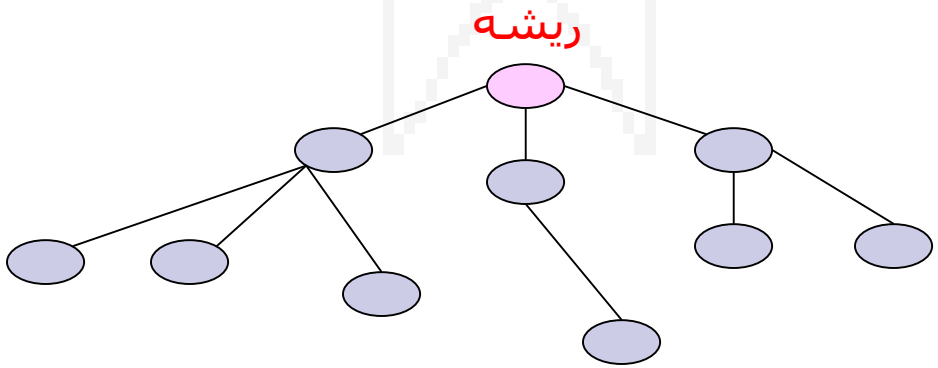
ساختارهای داده يي (مدل داده يي) :

با آنکه ساختار داده يي متعددی برای طراحی سطوح ادراکی و خارجی بانك وجود دارد ما در این فصل سه ساختار رابطه‌ای، سلسله مراتبی و شبکه‌ای را شرح می‌دهیم. البته امروزه ساختار شی‌گرا نیز معروفیت زیادی یافته است .

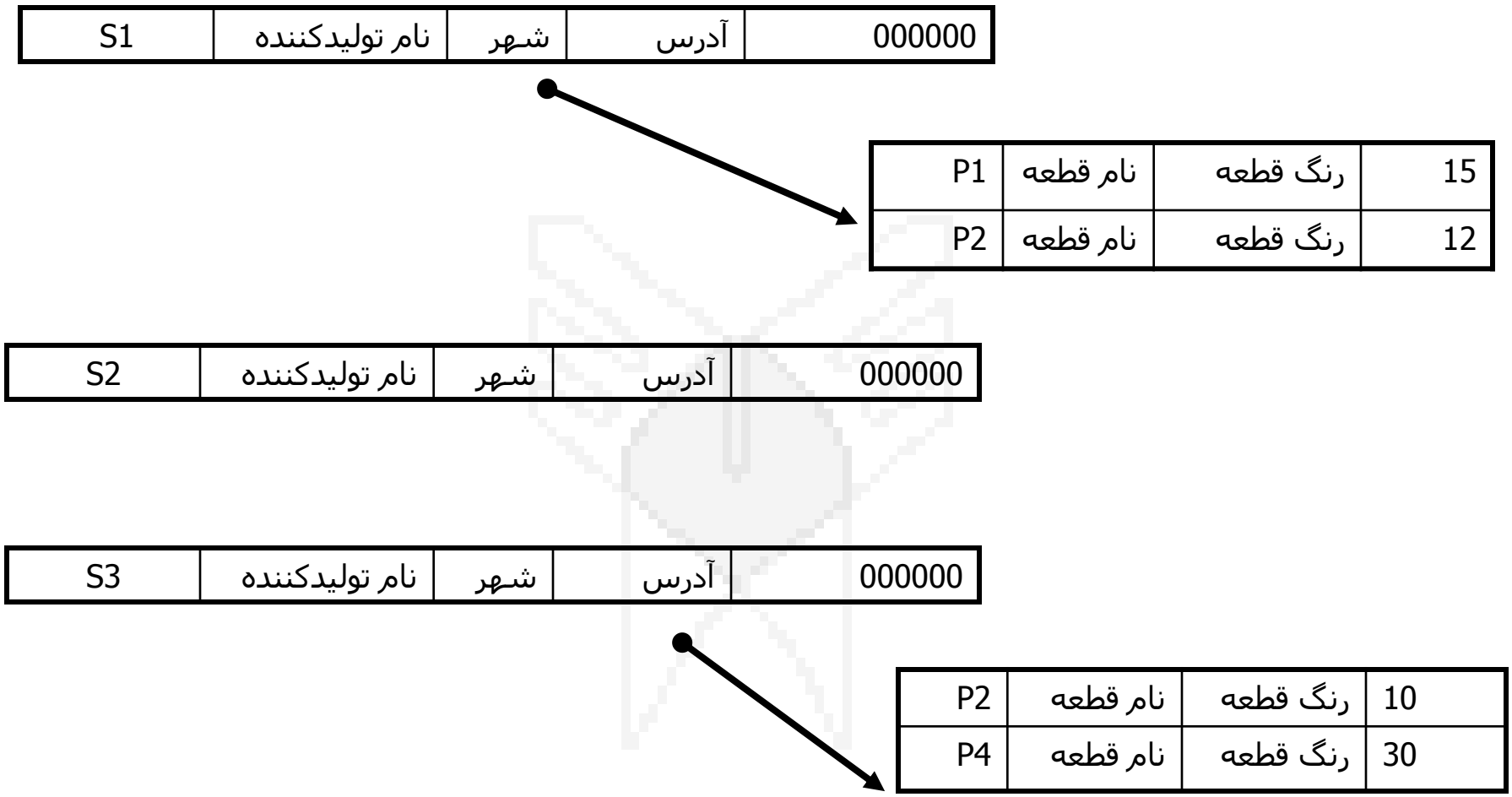


1- ساختار سلسله مراتبی :

این ساختار قدیمی‌ترین ساختار داده يي برای طرحی بانك اطلاعاتی در سطح انتزاعی است. در ساختار داده‌ها و ارتباط بین آنها به کمك يك درختواره نمایش داده می‌شوند.
درختواره گرافی است دارای يك ریشه، به هم بسته و غیر چرخشی. منظور از به هم بسته این است که بین هر دو گره پیوندی وجود دارد. غیرچرخشی یعنی مسیری از گره سطح پائین‌تر به گرهی از سطح بالاتر وجود ندارد. رابطه همواره از سطح بالاتر به سطح پائین‌تر است. هر گره پدر می‌تواند چندین فرزند داشته باشد ولی هر فرزند فقط يك پدر دارد.
در این مدل يك موجودیت بعنوان پدر(گره ریشه) و موجودیتهاي دیگر بعنوان فرزند در نظر گرفته میشوند
این رابطه برای حالت 1 : n خوب است ولی در حالت چند به چند با مشکل مواجه می شود .



مثالي از مدل ساختار سلسله مراتبی :



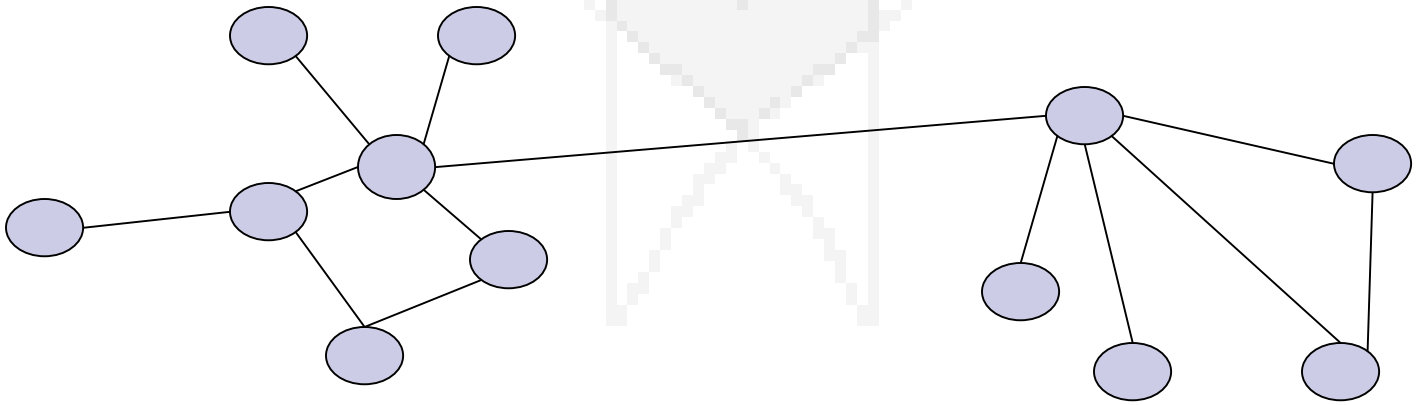
اشکالات مدل سلسله مراتبی :

1 ـ برای جستجو باید بانك اطلاعاتی را از ابتدا تا انتها پیمایش کرد و این امر زمانبر است (مثلاً وجود P2 توسط دو تولید کننده است)
2 ـ طول رکوردها متغیر است .
3 ـ افزونگی داده ها زیادتر است . (اطلاعات موجودیت فرعی تکرار می شود)
4 ـ در عملیات ذخیره سازی مشکل دارد تغییرات اعمال شده در طک موجودیت فرعی باید در تمام موجودیتهای فرعی موجود لحاظ شود . یعنی درج یا حذف سخت تر است .
5 ـ درمورد موجودیتهای فرعی که رابطه ای با موجودیت اصلی ندارند نمیتوانیم اطلاعاتی را نگهداری کنیم . مثلا از قطعه P3 (موجودیت فرعي) چون هنوز توسط هیچ تولیدکننده اي (موجودیت اصلي) ، تولید نمیشود اطلاعاتي قابل نگهداري نیست 0
6- تقارن ساختار جدولی را ندارد0

2- ساختار شبکه ای :

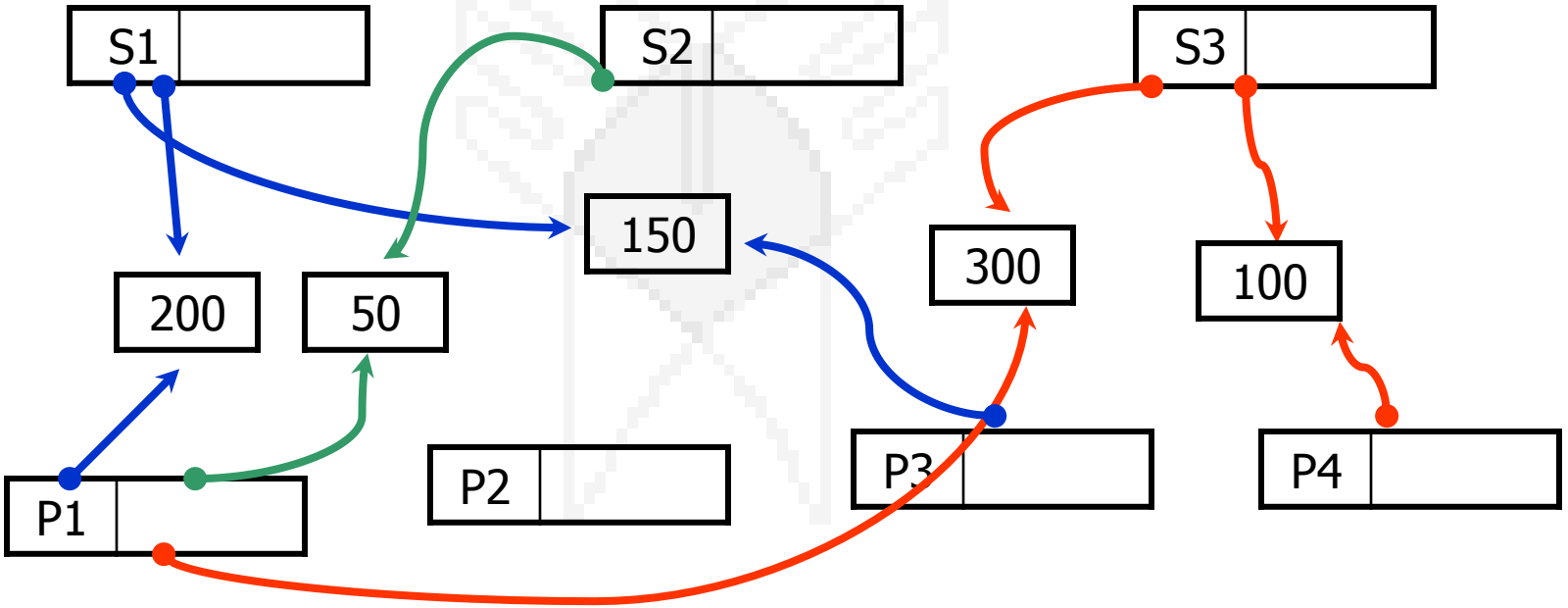
ساختار شبکه‌ای که به آن ساختار پلکس (PLEX) نیز می‌گویند.
در این ساختار هر گره فرزند می‌تواند بیش از یك گره پدر داشته باشد. این ساختار که جامع‌تر از ساختار سلسله مراتبی است برای نمایش ارتباطات یك به چند دوسویه مناسب است.
در واقع ساختار سلسله مراتبی حالت خاصی از ساختار شبکه‌ای است و ساختار شبکه‌ای را می‌توان با پذیرش مقداری افزونگی به ساختار سلسله مراتبی تبدیل کرد.
در واقع ساختار شبکه‌ای گرافی مثل زیر است :

در مدل شبکه اي اطلاعات در گره هاي يك گراف دلخواه ذخيره ميشوند. در اين گراف هر گره دلخواه ميتواند چند والد و چند فرزند داشته باشد و اين مدل براي نمايش روابط داده اي چند به چند مناسب است .



معایب روش شبکه ای :

این مدل مشکل تکرار و نیز اشکال مورد 5 سلسله مراتبی را حل کرده ولی دارای مشکلات است .
افزونگي داده ها و ناسازگاری داده ها در این روش نسبت به سلسله مراتبی کمتر است .
اما به علت حجم زیاد اشاره گر ها ممکن است سبب بروز فزونکاری و همچنین افزایش پیچیدگي در سیستم بشود .



3ـ ساختار رابطه‌ای (Relation Data Base Management System) :

در سال 1970 دكتر كاد عضو آزمايشگاه تحقيقاتي سن جوز IBM تئوري رياضي پايگاه داده اي رابطه اي را که اینکه چگونه میتوان داده ها را با استفاده از ساختار جدولي ذخيره و مديريت نمود را ارائه كرد0

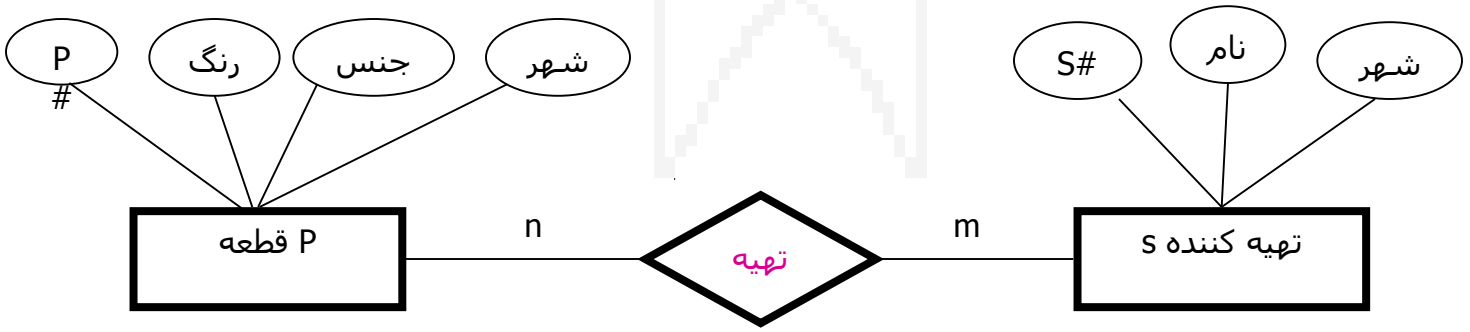
از دید کاربر بانك اطلاعاتی تشكيل شده است از تعدادی جدول .

رابطه مفهومي رياضي است. اما از ديد كاربر، رابطه نمايشي جدولي دارد.

جدول ساختاری است نامدارکه از تعدادی سطر و ستون تشكيل يافته است.هر ستون نمايشگر يک صفت خاصه از يك نوع موجوديت است و هر سطر نمايشگر يک نمونه از يک نوع موجوديت می‌باشد.دراين مدل يك سری قوانين وجود دارد که اگرجداول آن قوانين را رعايت كنند مدل رابطه ای هستند وگرنه جدول هستند .درواقع هر رابطه ای جدول است ولی هر جدولی يك رابطه نيست.

مفاهيم ساختار جدولي عبارتند از: جدول – سطر – ستون

موجودیتهای قطعه و تولید کننده را در نظر گرفته و نمودار EER آن را ترسیم می‌کنیم.



برای تشریح نمودار فوق در بانك رابطه‌ای یك جدول برای هر یك از دو موجودیت و جدولی نیز برای بیان ارتباط آنها، استفاده می‌شود :

S#	Name	City
S1	فن آوران	تهران
S2	ایران قطعه	تبریز
S3	پولادین	تبریز

جدول تهیه کنند گان (S)

P#	Name	Color	جنس	City
P1	محصول 1	قرمز	آهن	تهران
P2	محصول 2	سبز	مس	تبریز
P3	محصول 3	آبی	برنج	شیراز
P4	محصول4	قرمز	آهن	تهران

جدول قطعه (P)

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

جدول محموله (SP)

- یکی از مزایای مهم مدل رابطه‌ای سادگی زیاد و درك راحت این ساختار است. همچنین این مدل از پشتوانه تئوری ریاضی و قوی برخوردار می‌باشد.

نکاتی در باره اصطلاحات کاربردی مدل رابطه اي (Relational) :
تعاريف پايه

رابطه *(Relation)* : زير مجموعه‌اي از مجموعه ضرب دكارتي چند دامنه است. در مدل رابطه‌اي كه عمدتاً توسط جداول پياده سازي مي‌شوديك رابطه معادل يك جدول *(Table)* است (نه مساوي آن) يعني هر رابطه يك جدول است ولي هر جدول الزاما" يك رابطه نيست چون امكان دارد يك جدول قوانين مدل رابطه اي را رعايت نكند0

چند تايي Tuple : به هر سطر از هررابطه تاپل يا چندگانه می گویند و هم عرض ركورد در مدل جدولي است .
در واقع هر رابطه معادل مجموعه‌اي از تاپل‌هاست. لذا ترتيب قرارگيري تاپل‌ها در يك رابطه يا جدول مهم نيست.

نكته: هرگز رابطه را با ارتباط يكي نگيريد. ارتباط (*Relationship*) به معني ارتباط 2 موجوديت است كه در بخش قبل ديديم. در حاليكه رابطه مجموعه‌اي از تاپل‌هاست

صفت خاصه Attribute : صفت خاصه یا خصوصیت به نام هر ستون از یك رابطه گفته میشود و هم عرض فیلد در جدول است .

Domain (دامنه) : به بازهء تغییرات یك خصوصیت یا فیلد ، Domain می گویند .
دامنه فیلد نمره عددیست مثبت بین 00/00 الي 00/20

کاردینالیتی : به تعداد چندگانه ها یا تاپل های یك رابطه یا جدول کاردینالیتی می گویند .

Degree (درجه) : به تعداد ستون ها یا Attribute های یك رابطه یا جدول ، درجه یا Degree می گویند.

در رابطه ای	رابطه	تاپل	صفت خاصه	میدان یا دامنه
در جدولی	جدول	سطر(رکورد)	ستون(فیلد)	مقدار مجاز هر فیلد

بخشهاي اساسي مدل دادهاي

1- بخش ساختاري
2- بخش عملياتي (پردازشي)
3- بخش جامعيتي

بخش ساختاري ، نشاندهنده عناصر ساختاري مدل است كه همان ساختار دادهاي اصلي و مفاهيم مرتبط با آن است. مثل تعريف يك رابطه يا تغيير آن

بخش عملياتي، مجموعه امكاناتي است كه به وسيله آنها عمليات مورد نظر كاربر انجام ميشود. مثل اضافه يا تصحيح يك تاپل

بخش جامعيتي، از مجموعهاي از قواعد و محدوديتهاي جامعيتي تشكيل شده است كه به وسيله آنها سيستم مديريت پايگاه داده ميتواند صحت، دقت و سازگاري دادهها را كنترل و تضمين كند. مثل كنترل عدم مغايرت يك صفت خاصه (فيلد)

تعريف جديدتر از رابطه :

با فرض وجود n ميدان D1 تا Dn، نه لزوما متمايز، رابطه R از دو قسمت تشكيل شده است:

1- سرآيند Header : مجموعه‌اي نامدار از n صفت به صورت Ai:Di كه در آن هر Ai نام يك صفت است و هر Di نام ميدان صفت

2- پيكر (بدنه) Body : مجموعه‌اي است از m تاپل t به نحوي كه t خود مجموعه‌اي است از n عنصر هريك به صورت Ai:vi كه در آن vi مقداري است از نوع ميدان.

مقدار n را درجه (همان تعداد صفات) و مقدار m را كاردينالیتي رابطه مي‌گويند

A1	A2	A3				An

جداول نمونه مربوطه به سيستم بانك اطلاعاتي توليد كننده و محصول

S#	P#	Qty
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

جدول (SP)
تعداد توليد هر محصول توسط توليدكننده

S#	Sname	Status	City
S1	Sn1	20	C2
S2	Sn2	10	C3
S3	Sn3	30	C2
S4	Sn4	20	C2
S5	Sn5	30	C1

جدول (S)
توليدكنندگان

P#	PName	Color	Wight	City
P1	Nut	Red	12	C2
P2	Bolt	Green	17	C3
P3	Screw	Blue	17	C4
P4	Screw	Red	14	C2
P5	Cam	Blue	12	C3
P6	Coy	red	19	c2

جدول (P)
محصولات

سیستم رابطه ای دارای مشخصه هاي زیر است :

1 ـ همه اطلاعات داده فقط و فقط بصورت رابطه یا جدول قابل رویت هستند .
2 ـ برای درخواست رویت داده های جداول ازعملگرهای زیر استفاده شده و خروجی این دستورات نیز بصورت رابطه یا جدول میباشد
3- عملیات انجام گرفته در مدل رابطه ای عبارتند از : الف : درج (INSERT) ب: حذف (DELETE) ج : بهنگامسازي (UPDATE) د : بازیابي یا رویت (SELECT)

عملگر های زبان SQL برای رویت یا بازیابي داده های رابطه یا جدول بانك اطلاعاتی :

الف ـ select (محدود کننده سطری)
ب ـ project (محدود کننده ستونی)
ج ـ join (برای اتصال دو تا جدول به هم)

برای راحتی بیشترکاربران ، خدمات دستورات Project و Join را بکمك Select نیز قابل انجام است و دستورات فوق در حقیقت بخش عمومي زبان SQL را تشکیل می دهند .

مشخصه کلی دستورات زبان SQL :

1- آسانی و سادگی 2- پر قدرتی و سرعت بالا
وبه خاطر این خصایص همه DBMS ها از زبان SQL حمایت و پشتیباني میکنند SQL یك زبان پرس و جوی داده ای است .

تعدادی از انواع متغیرها در SQL :

Integer: عدد صحیح

Smallint , BigInt : عدد صحیح

Decimal(p,g): عدد هرمی دارای p رقم و q رقم اعشاری در سمت راست.

Float : برای اعداد اعشاری با نقطه شناور

Numeric(p,g) : عدد صحیح

Character(n) یا Char(n): رشته ای کاراکتری به طول n

Varchar(n): رشته ای کاراکتری به طول متغیر حداکثر n

عملگرهای موجود در SQL :

عملگرهای ریاضی عبارتند از: - , + , / , ×

عملگر || برای الصاق دو دسته کاراکتری استفاده میشود,مثلآ با دستور NAME||FAMILY دو متغیر رشته ای با هم ترکیب می شوند,در بعضی از پیاده سازی ها علامت + برای چسباندن دو رشته استفاده می شود.

عملگرهای مقایسه ای عبارتند از : =, =~,<,<=,>,=>

عملگر =~ به معنای نا مساوی می باشد.

رابط های منطقی عبارتند از: AND,OR,NOT

شکل عمومی دستورات SQL برای دیدن داده های جداول بانك اطلاعاتی :

الف : محدود کننده سطری Select

Select * From Table_Name Where شرط دلخواه

مثال : SELECT * FROM STUDENT WHERE ST_NO='8125456'

ب : محدود کننده ستونی Project

Select Field_Name1, Field_Name2,... From Table_Name

مثال : SELECT ST_NO,F_NAME,L_NAME FROM STUDENT

ج : برای join دو جدول :

```
Select * From Table_Name1, Table_Name2
    Where Table_Name1. Field_Name1= Table_Name2. Field_Name2
```

یا

```
Select * From Table_Name1
    INNER JOIN
    Table_Name2 ON
    Where Table_Name1. Field_Name1= Table_Name2. Field_Name2
```

- این نوع Join فقط رکوردهاي مشترك در 2 جدول (براساس شرط گذاشته شده در Where) را نمایش میدهد

ج : **Cross join** يك نوع خاص از Join است كه نتيجه اجراي آن درحقيقت ضرب كارتزين دو جدول است و تعداد ركوردهاي جدول نتيجه ، حاصلضرب ركوردهاي دو جدول قبل است يعني هر ركورد از جدول اول با تمام ركوردهاي جدول دوم تركيب شده و الي آخر 000

- Cross join داراي سربار زياديست و بشدت منابع سيستم را اشغال كرده و در دنياي واقعي كاربرد بسيار بسيار كمي دارد0

Select * From Table_Name1 Cross JOIN Table_Name2
يا
Select * From Table_Name1 , Table_Name2

مثال :

A	B
α	1
β	2

r

B	C	D
α	10	a
β	10	a
□	10	b

S

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	□	10	b
β	2	α	10	a
β	2	β	10	a
β	2	□	10	b

r × s :

ج : Outer join يك نوع ديگر از Join است و وقتي بخواهيم همه ركوردهاي يك جدول و ركوردهاي متناظر از جدول ديگر را ببينيم از آن استفاده ميكنيم0 و داراي 2 نوع Left و Right ميباشد0 يعني همه ركوردهاي مورد درخواست از جدول چپ باشد يا جدول راست 0

```
Select  * From Table_Name1
        Left INNER JOIN
        Table_Name2   ON
    Where Table_Name1. Field_Name1= Table_Name2. Field_Name2
```

```
Select  * From Table_Name1
        Right INNER JOIN
        Table_Name2   ON
    Where Table_Name1. Field_Name1= Table_Name2. Field_Name2
```

ج : full Outer join يك نوع ديگر از Join است و وقتي بخواهيم همه ركوردهاي متناظر و غير متناظر از دو جدول در جدول نتيجه نمايش داده شوند ، از آن استفاده ميشود0

```
Select  * From Table_Name1
        Full INNER JOIN
        Table_Name2   ON
    Where Table_Name1. Field_Name1= Table_Name2. Field_Name2
```

مثال برای جلوگیری از نمایش رکوردهای تکراری در خروجی DISTINCT :

```
SELECT  DISTINCT First_Name  FROM  Student  WHERE  job=23
```

برای نمایش رکوردها بترتیب یك یا چند فیلد در خروجی :

```
SELECT  Code,Name,Company_Name  From  Kala
        ORDER BY Company_Name,Code
```

مثال برای تصیح سایز و یا ایجاد یك فیلد جدید در جدول کالا :

```
ALTER  TABLE  Kala  MODIFY  (Name  CHAR(50))
ALTER  TABLE  Kala  ADD  (St_No CHAR(10))
```

شکلهای عمومی دستورات SQL برای تغییر در جداول بانك اطلاعاتی :

1-دستورInsert : (برای اضافه کردن رکورد)

INSERT INTO table_ name VALUES (one row)

INSERT INTO student VALUES(“771234”,”ali,”Mahmoodi”,23)

2- دستورupdate (بروز آوردن ، تغییر دادن) :

UPDATE table_ name

SET assignment _ commandlist

[WHERE Condition(s)]

UPDATE student SET field1=35 WHERE left(ST_No,2)=“77”

3- دستورDELETE :

DELETE FROM table _ name [WHERE Condition(s)]

DELETE FROM student WHERE left(ST_No,2)=“77”

فرمت کلی دستور SQL :

Select (Distinct) item(s) from table(s)

(where شرطها) (Group by Fild(s)) (Having شرطها)

(Order by Field(s))

ساده ترین شکل این دستور به شکل زیر است :

Select نام فیلد ها from نام جدول where شرط

با توجه به جداول نمونه صفحات قبل به سوالات مربوطه پاسخ خواهیم داد :

1. دستور زیر کل جدول S را چاپ می کند :

```
Select *  from s
```

2. خروجی دستور روبرو به صورت زیر می باشد :

```
Select S#,status from S where city='c2'
```

خروجی

S#	status
S1	20
S3	30
S4	20

3. دستور روبرو : select p# from sp

تمام مقادیر ستون # p از جدول sp را می دهد. (حتی با سطرهاي تکراری)

خروجی :

P#
P1
P2
P3
P4
P5
P6
P1
P2
P2
P2
P4
P5

4. دستور روبرو select Distinct p# from sp

به علت استفاده از Distinct مقادیر ستون p # از جدول sp را با حذف تکراری ها می دهد.

خروجی :

P#
P1
P2
P3
P4
P5
P6

5. شماره تهیه کنندگانی را بیابید که ساکن C2 بوده و وضعیت آنها از 20 بیشتر باشد :

```
SELECT S# FROM S WHERE CITY='C2' AND STATUS>20
```

خروجی :

S#
S3

6. SUM مجموع مقادیر دریک ستون را نشان می دهد, که در تابع select تعریف میشود.

```
Select SUM(QTY) from sp Where p#='p2'
```

خروجی : 1000

7. COUNT تعداد مقادیر در یک ستون را نشان می دهد , که در تابع select تعریف میشود.

تعداد تهیه کنندگان شهر C2 چند تاست؟

```
Select count(s#) from s where city='c2'
```

خروجی : 3

7. AVG میانگین یک ستون را نشان می دهد , که در تابع select تعریف میشود :

میانگین تعداد تولید محصول P4 چند تاست؟

```
Select AVG(p#) from sp where p#='p4'
```

خروجی : 250

7. يك مثال در مورد استفاده از توابع جمعي MIN و MAX

```
SELECT ST_No,MIN(Nomreh) , MAX(Nomreh)
        FROM   STNOMREH     Group By  ST_NO
         WHERE  TR='1'
                AND
                YEAR='85-86'
                AND
                COID='456222';
```

بالاترين و پايين ترين نمره در درس 456222 در ترم اول 86-85 از جدول STNOMREH را بازيابي مي‌كند.

مثال : استفاده از GROUP BY :

Query : شماره هر قطعه تهیه شده و کل تعداد تهیه شده از هر قطعه را بدهید .

```
SELECT P# ,SUM(QTY)
FROM SP
GROUP BY P# ;
```

GROUP BY CLAUSE جدول داده شده بعد از FROM را منطقا گروه بندی می کند به نحوی که در هر گروه مقدار ستون یا ستونهای داده شده پس از GROUP BY یکسان است .(گروه بندی بر اساس آنچه بعد از GROUP BY داده می شود) آنگاه تابع SUM عمل می شود.

(ستونهایی که حاصل اعمال توابع هستند بی نام می باشند)

S#	P#	QTY
S1	P1	300
S2	P1	300
S1	P2	200
S2	P2	400
S3	P2	200
S4	P2	200
S1	P3	400
S1	P4	200
S1	P5	100

P#	Expr1
P1	600
P2	1000
P3	400
P4	200
P5	100

مثال : این دستور یك سطر به جدول P با مشخصات : قطعهp7 در شهر c1 و وزن 24 و با اسم و رنگ در حال حاضر نامشخص اضافه میکند :

```
Insert into  p (p#,city,weight )  values (‘p7’,’c1’,24)
```

تذکر: مقدار فیلد های اسم و رنگ برابر null میشود.

مثال : دستور زیر : رنگ قطعه **p2** را به زرد تغییر داده و به وزن آن **5** مي افزاید و شهر آن را نا شناخته اعلام میکند.

```
Update  p  set  color=‘yellow’,weight=weight+5,city=null  where  p#=‘p2’
```

مثال : تهیه کننده S5 را حذف کنید:

```
Delete  from s  where s#=‘s5’
```

مثال : MIN و MAX حداقل و حداکثر یک ستون را نشان می دهد , که در تابع select تعریف میشود :

حداقل و حداکثر تعداد تولید محصول P4 چند تاست؟

```
Select  MIN(p#), MAX(p#) from  sp   where  p#='p4'
```

خروجی : 200,300

مثال : نام توليدكنندگاني كه توليدي (حداقل يك توليد) داشته باشند:

روش 1:

```
Select  Sname from SP,S
        Where   SP.S# = S.S#
```

روش 2:

```
Select  Sname from S
        Where   exists (
                        select * from SP )
```

مثال : نام شهرهايي را پيدا كنيد كه در آن قطعه اي توليد **يا** توليد كننده اي در آن وجود دارد :

جواب :

```
Select  City from P
Union
Select  City from S
```

Union براي اجتماع 2 جدول

مثال : نام شهرهايي را پيدا كنيد كه در آن قطعه اي توليد ميشود **ولي** توليد كننده اي در آن وجود ندارد :

جواب :

(Select City from P)

Except

(Select City from S)

Except براي عمل تفريق 2 جدول

مثال : نام شهرهايي را پيدا كنيد كه در آن قطعه اي توليد **و** توليد كننده اي در آن وجود دارد :

جواب :

```
Select  City from P
Intersect
Select  City from S
```

Intersect براي اشتراك 2 جدول

نكته : در هر 3 دستور Union و Except و Intersect ركوردهاي تكراري حذف ميگردد و در صورت نياز به نمايش ركوردهاي تكراري ميتوان از عبارت All همراه آنها استفاده كرد

عمليات جبر رابطه اي

مقدمه:

زبان کار با داده ها دارای مجموعه ای است از عملگرها که در کادر مدل داده ای عمل می کنند . در سیستم بانک رابطه ای عملگرهای عمل کننده روی رابطه عملگرهای جبر رابطه ای و محاسبات رابطه ای می باشند .

جبر رابطه ای :

Codd در مقاله خود 8 عملگر برای کار با رابطه ها تعریف کرده است که این عملگرها به دو دسته تقسیم می شوند :

الف : عملگرهای متعارف در مجموعه ها نظیر اجتماع ، اشتراک ، تفاضل و ضرب دکارتی .

ب : عملگرهای خاص نظیر محدودیت ، تصویر یا پرتو ، ترکیب یا پیوند و تقسیم .

عملگر گزینش یا تحدید (select)

این عملگر تاپل هایی را از یک رابطه گزینش می کند . به عبارتی زیر مجموعه ای افقی از یک رابطه را بر می دارد . گزینش تاپل یا تاپل هایی از رابطه ، معمولاً بر اساس شرط یا شرایطی صورت می پذیرد .

❋نماد تحديد : □p(r)

❋ Pشرط گزينش است و r جدول(رابطه)

❋$\square_p(r)$ **= {t | t∈r and p(t)}**

يا هر يك از جملات بفرم: not ،Or ،andفرمولي محاسباتي است شاملp

، است.≤،≥،=،#،>،< يكي از opكه<attribute>op<atrribute>or< constant>

A	B	C	D
α	α	1	7
α	β	5	17
β	β	12	10

Relation r :

A	B	C	D
α	α	1	7
β	β	23	10

$\sigma_{A = B \square D > 5}(r)$

عملگرپرتوproject :

این عملگر زیر مجموعه ای عمودی از یک رابطه را استخراج می کند که صفات خاصه (ستون های) پاسخ اعمال عملگر دارای ترتیبی هستند که در عملگر مشخص می شود .

نام رابطه است . rاسامی صفت خاصه و A_1 A_2,... که$\Pi_{A1,A2,...Ak}$ ❊نماد عملگر

ستون تعریف مي شود که باحذف ستونهايي که درليست k❊نتيجه بعنوان رابطه اي با قرار ندارند بدست آمده است .

❊ سطرهای تکراری از نتیجه حذف می شوند .

مثال:

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

$\Pi_{A,C}$ (r)

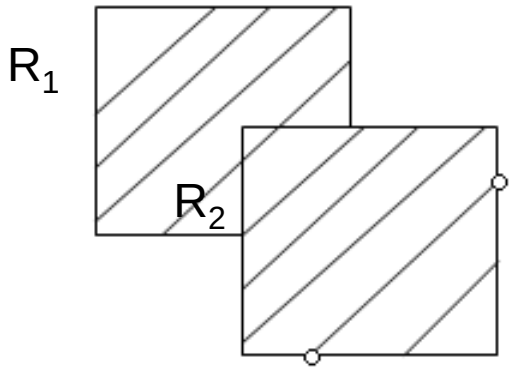
A	C
α	1
α	1
β	1
β	2

=

A	C
α	1
β	1
β	2

Union : عملگر اجتماع

اجتماع دو رابطه رابطه ایست که تاپلهایش در یك یا هر دو رابطه وجود دارند .



R = R1 union R2

- نماد عملگر : $r \cup s$
- $r \cup s = \{t \mid t \in r \text{ or } t \in s\}$
- $r \cup s$ موقعي معتبر است كه :

1 - r,s بايستي تعداد صفات خاصه برابر داشته باشند .

2 - حوزه (ميدان) صفات خاصه دو مجموعه بايستي سازگار باشند . بعنوان مثال صفت خاصه i ام از هر دو رابطه يكسان باشند.

مثال :

Relation r,s

A	B
α	1
α	2
β	1

A	B
α	2
β	3

r ∪ s :

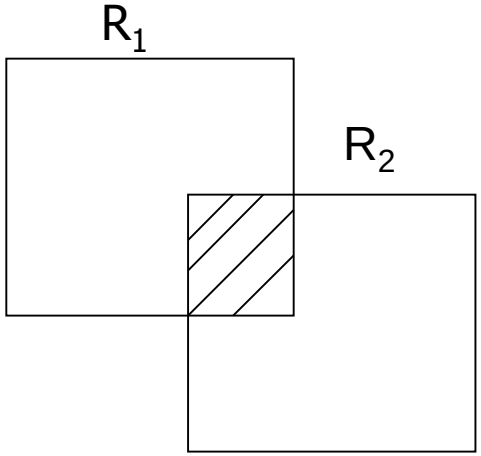
A	B
α	1
α	2
β	1
β	3

❊اشتراك Intersect

اشتراك دو رابطه،رابطه ایست که تاپلهایش در هر دو رابطه وجود داشته باشند .

❊ نماد r ∩ s

❊ **r ∩ s = {t | t ∈ r and t ∈ s }**



R = R1 ∩ R2

❊ r و s باید از نظر صفات خاصه برابر و صفات خاصه متناظر ازیک میدان برگرفته شوند .

A	B
α	1
α	2
β	1

r1

A	B
α	1
β	3

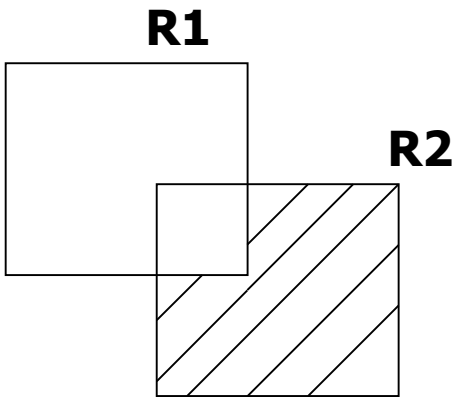
r2

A	B
α	1

r1 ∩ r2

عملگر تفاضل set difference

تفاضل دو رابطه ، رابطه ایست که تاپلهایش در رابطه اول موجود باشند و در رابطه دوم وجود نداشته باشند .



* نماد : **r – s**

* تعریف عملگر : **r - s = {t | t ∈ r and t ∉ s }**

* اختلاف دو مجموعه بین دو مجموعه سازگار انجام مي شود . r و sباید دارای ستون های یکسان باشند .

مثال :

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

A	B
α	1
β	1

r – s

حاصلضرب کارتزین Cartesian Product

حاصل رابطه ایست حاوي ترکیب هاي ممکن تاپلهاي دو رابطه که باید در هم ضرب شوند .

- نماد عملگر : $\mathbf{r \times s}$
- تعریف عملگر : $\mathbf{r \times s = \{(t,q) \mid t \in r \text{ and } q \in s\}}$
- فرض مي شود صفات خاصه r(R) و s(S) مجزا باشند . $R \cap S = \emptyset$
- اگر صفات خاصه r(R) و s(S) مجزا نباشند تغییر نام بایستي صورت پذیرد .

مثال :

A	B
α	1
β	2

r

B	C	D
α	1	a
β	0	a
□	1	b
	0	
	1	
	0	

s

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	□	10	b
β	2	α	10	a
β	2	β	10	a
β	2	□	10	b

r × s :

ترکیب (پیوند) join (Natural join)

- حاصل رابطه اي است كه تاپلهاي آن از پيوند (تركيب) تاپلهايي از دو رابطه بدست مي آيد بشرط تساوي مقادير يك يابيش از يك صفت خاصه .

- نماد عملگر : $r \infty s$

- فرض r رابطه روي اسكيماي R و s رابطه اي روي اسكيماي s باشد .نتيجه رابطه اي در اسكيماي $R \cup S$ است كه با در نظر گرفتن هر جفت تاپل tr از r و ts از s بدست مي آيد .

- اگر tr و ts داراي مقادير يكسان در هر يك از صفات خاصه $R \cap S$ باشند يك تاپل t به نتيجه اضافه مي شود بطوريكه t همان مقادير tr در r و ts در s را دارد.

مثال : R = (A,B,C,D)

S = (E,B,D)

(A,B,C,D,E) == اسکیماي نتیجه

* **r ∞ s** بصورت زیر تعریف مي شود :

A	B	C	D
α	1	α	a
β	2	□	a
□	4	β	b
α	1	□	a
δ	2	β	b

r

B	D	E
1	a	α
3	a	β
1	a	□
2	b	δ
3	b	ε

S

A	B	C	D	E
α	1	α	a	α
α	1	α	a	□
α	1	□	a	α
α	1	□	a	□
δ	2	β	b	δ

r ∞ s

عملگر تقسيم Division

دورابطه يكي از درجه m + n و ديگري از درجه n را برهم تقسيم مي كند . حاصل رابطه ايست حاوي مقاديري از صفات خاصه رابطه از درجه m + n كه مقادير صفت خاصه ديگر به تمامي در رابطه درجه n وجود داشته باشند .

- نماد ÷
- اين عملگر براي براي پرس و جوهايي كه عبارت براي تمام (for all) را دارند مناسب است .

- فرض كنيد r و s رابطه هايي روي اسكيماي R و S باشند بطوريكه :

$$R = (A1,\ldots,Am,B1,\ldots,Bn)$$
$$S = (B1,\ldots,Bn)$$

نتيجه تقسيم r بر s رابطه ايست در اسكيماي R – S

$$R - S = (A1,\ldots,Am)$$

$$\mathbf{r \div s = \{t \mid t \in \square R - S(r) \wedge \forall\, u \in s(tu \in r)\}}$$

مثال :

A	B	C	D	E
α	A	α	a	1
α	A	□	a	1
α	A	□	b	1
β	A	□	a	1
β	A	□	b	3
□	A	□	a	1
□	A	□	b	1
□	A	β	b	1

r

D	E
a	1
b	1

s

A	B	C
α	a	□
□	a	□

r ÷ s

عملگرهاي اضافه شده و عمليات ديگر جبر رابطه اي

عملگر تغيير نام Rename

اين امكان را مي دهد كه به يک رابطه با بيش از يك اسم رجوع شود .

* نماد عملگر $P_x(E)$

عبارت E را تحت نام X بر مي گرداند .

* اگر يك عبارت جبررابطه اي E ، n ستون داشته باشد $Px(A1, \ldots, An)E$.

به معني آن است كه نتيجه عبارت E تحت نام X با صفات خاصه تغيير نام يافته A1, ... ,An بر می گرداند .

عملگر بسط Extend

عملگری است که براي گسترش عنوان يك رابطه بكار مي رود كه معمولا صفت خاصه اضافه شده مقادير آن با ارزيابي يك عبارت محاسباتي مشخص بدست مي آيد .

Extend P Add (weight * 454) As GMWT

GMWT وزن قطعات بر حسب گرم مي باشد كه در واقع تبديل شده پوند به گرم است .

عملگر هاي جمعي Aggregate Operator

عملگرهايي كه براي شمارش، مجموع، ميانگين و مي نيمم و ماكزيمم بكار مي روند. در واقع مجموعه اي از مقادير را گرفته و يک مقدار تكي را بعنوان خروجي باز مي گردانند.

Min , Max , Avg , Sum , Count

مثال : مجموع قطعات تهيه شده توسط تهيه كننده s1. 'g sum(Qty) (sp)' = #s6
gsum(Qty) (sp)

عملگر انتساب Assignment

- نماد عملگر : ←
- اين عملگر يك روش مناسب براي بيان پرس و جوهاي پيچيده است .
- انتساب بايستي به يك متغيير رابطه اي موقت نسبت داده شود .

عملگر نيم پيوند Semi Join

اين عملگر گونه اي ديگر از پيوند طبيعي است كه درآن ، تنها تاپلهاي پيوند شدني از رابطه سمت چپ در رابطه جواب وارد مي شوند. اين عملگر در پايگاه داده هاي توزيع شده كاربرد دارد .

$$R1 \propto R2 = \Pi \text{ Attribute(R1) } (R1 \infty R2)$$

عملگر نیم تفاضل Semi Minus

این عملگر بصورت زیر تعریف شده است :

$$R1 \text{ SEMIMINUS } R2 = R1 \text{ MINUS } (R1 \propto R2)$$

مجموعه کامل عملگرها در جبر رابطه اي

از عملگرهاي مطرح شده ، برخي مبنايي هستند به اين معنا كه مجموعه آنها از نظر عملياتي كامل است و هر عملگر ديگر را مي توان بر حسب عملگرهاي اين مجموعه بيان كرد . اين مجموعه كامل بصورت زير است :

{Select , Project , Union , Minus, Time }

برخي عملگرهاي تعريف شده با اين مجموعه عبارتند از :

$$R1 \cap R2 = R1 - (R1 - R2) = R2 - (R2 - R1)$$
$$R1 \cap R2 = (R1 \cup R2) - (R1 - R2) \cup (R2 - R1)$$
$$R1(Y,X) \div R2(X) = R1[Y] - ((R1[Y] \times R2) - R1)[Y]$$

برخي خواص عملگرها :

اگر R,S,T رابطه باشند داريم :

1- R ∞ S = S ∞ R
2-(R ∞ S) ∞ T = R ∞ (S ∞ T)
3-Π A1..Ap (Π Ai..Aj) (R) = Π A1..Ap (R)
4- б Ai=’a’ (R) U бAi=’a’(S) = бAi=’a’ (RUS)
5- б Ai=’a’ (R) ∩ бAi=’a’(S) = бAi=’a’ (R∩S)
6- б Ai=’a’ (R) - бAi=’a’(S) = бAi=’a’ (R-S)
7- Π A1..Ap (R) U Π A1..Ap (S) = Π A1..Ap (R U S)

مثال : بانك اطلاعاتي تهيه كنندگان، قطعات، پروژه را در نظر بگيريد. ساختار آن در زير ارائه شده است با استفاده از جبر رابطه اي به پرس و جوهاي زير پاسخ دهيد.

S (s# , sname , status , city)
P (p# , pname , color , weight , city)
J (j# , jname , city)
SPJ (s# , p# , j# , Qty)

۱- جزئيات كامل تمام پروژه هاي شهر "تهران" را مشخص كنيد .

6city = 'تهران' (J) يا J where city = 'تهران'

۲- اسامي تهيه كنندگان قطعه P2 را بدهيد .

ΠSname (6P# = 'P2'(S ∞ SPJ)) يا ΠSname (S join SPJ where P# = 'P2')

يا temp1 ← S join SPJ

temp2 ← temp1 where P# ='P2'
result ← ΠSname (temp2)

۳ -اسامي تهيه كنندگاني كه اقلا يك قطعه آبي را تهيه مي كنند .

((P where color = 'آبی') ∞ SPJ) ∞ S)[sname]
ΠSname ((бcolor = 'آبی' P) ∞ SPJ) ∞ S

٤- اسامي تهيه كنندگاني را بدهيد كه تمام قطعات را تهيه مي كنند .

Π_{Sname} ((Π(S# , P#) (SPJ) ÷ $\Pi_{P\#}$ (P) ∞ S)

٥- اسامي تهيه كنندگاني كه قطعه p2 را تهيه نمي كنند.

(Π$_{S\#}$ (S) – (Π$_{S\#}$ (**б**P# = 'P2'(SPJ)) ∞ S [sname]
(S[S#] – (SPJ where P# = 'p2') [S#]) ∞ S [sname] يا

٦- شماره قطعاتي را مشخص كنيد كه توسط يك تهيه كننده در شهر تهران يا پروژه اي در شهر تهران عرضه مي شود .

SPJ ∞ (S Where city = 'تهران')) [P#] ∪ SPJ ∞ (J Where city = 'تهران')) [P#]

$\Pi_{P\#}$ (SPJ ∞ (бcity = 'تهران' (S)) ∪ $\Pi_{P\#}$ (SPJ ∞ (бcity = 'تهران' J)

مثال ۲ : يك بانك اطلاعاتي در محيط عملياتي بانك را در نظر بگيريد كه داراي جداول زير مي باشد .
شعبه بانك (نام شعبه، شهر شعبه، داراييها)
مشتري (نام مشتري، آدرس)
حساب (شماره حساب ، نام شعبه، موجودي)
وام (شماره وام، نام شعبه، مقدار وام)
سپرده گذاري (نام مشتري، شماره حساب)
وام گيرنده (نام مشتري، شماره وام)

branch (branch_name , branch_city , assests)
customer (customer_name , customer_street , customer_city)
account (account number , branch_name , balance)
loan (loan_number , branch_name , amount)
depositor (customer_name , account number)
borrower (customer_name , loan_number)

ادامه مبحث مدل رابطه ای

چند نکته در مورد مدل رابطه ای :

1 ـ ساختار جدولی ساختار منطقی است . یعنی در سطح داخلی وسطح سیستم عامل الزاما" غیر جدولی است . ولی در سطح ادراکی و خارجی جدولی است .

2 ـ اطلاعات موجود در سیستم بانك اطلاعاتی فقط وفقط به یك روش بیان می شود .

3 ـ تمام مقادیر در مدل رابطه ای اسکالریا اتمیك هستند .

قواعد اولیه در مدل رابطه ای :

1-کاربران درتعریف ساختار دادهای موظف نیستندکه فیلدها را طبق نظم خاصی از چپ به راست و یا از راست به چپ بچینند زیرا کاربران مختلف متقاضي نظم دلخواه خود هستند و به همین جهت با دستور * Select میتوان بجای ستاره اسامی فیلدها را با نظم دلخواه جهت نمایش چید.

Select Field1 , Field2 , Field3 From Table_Name

2- کاربران در ثبت رکوردها (سطرها) موظف نیستند رکوردها را بر اساس يك نظم اجباری ثبت نماید, مثلا بر حسب شماره سند یا شماره فاکتور زیرا میتوان اینکارراادرگزارشات درخواستی با دستورات SQL و باکمك order by در انتهای دستور select انجام داد .

3- دو رکورد عین هم نبایدوجودداشته باشد. (اگر دو رکورد عین هم باشند دومی هیچ اطلاعات جدیدی به ما نمی دهند پس دومی قابل حذف است .)

نکته : در DBMSها ما بظاهر شاید بتوانیم 2رکورد عین هم ثبت کنیم درحالی که یک فیلد پنهان بنام " شماره رکورد" همه رکوردها را از هم متمایز میکند

4- در مدل رابطه ای هر خصوصیت یا فیلد باید اتمیك و غیر قابل تجزیه باشد .
یعني تعریف خصوصیت نام و سال تولد بعنوان يك فیلد غیر قابل قبول است 0

انواع جداول (رابطه) در يك سيستم بانك اطلاعاتی :

1-جداول مبنا Base Tables :

جداولی هستند که بطورواقعی وجود دارند واطلاعات ماندگارهرسیستم بانك اطلاعاتی درداخل آنها نگهداری میشود و هیچ کدام ازاین جداول درسیستم بانك اطلاعاتی قابل حذف نیستند .اگرهرکدام از این جداول را حذف کنیم اطلاعات آن ازطریق دیگر قابل حصول نیست .

2-جدول دیدگاه View Table :

الف : غیرواقعی است 2- افزونگی ندارد 3ـ جدول وابسته است(مشتق شده)

ایجاد جدول دید : CREATE VIEW ViewName AS دستورsql

جدول دیدگاه فقط شکل دستور را ذخیره میکند و در حقیقت يك ميانبر سريع براي استفاده از دستورات SQL است 0 مثال :

```
CREATE VIEW MAPHSTUD AS
        SELECT STID, STDEG, STMJR FROM STT
                WHERE STMJR='Math' OR STMJR='Phys'
```

3- جدول تصویر لحظه ای Snapshot Table :

مشخصات : 1-جدول واقعی است 2- افزونه است 3- مشتق شده است

برای بروزکردن جدول تصویرلحظه ای ازدستور زیر استفاده می شود که در این مثال هر روز جدول Update می شود .

CREATE SNAPSHOT اسم دلخواه AS دستورSQL REFRESH EVERY DAY

دلایل استفاده از snapshot :

1- به عنوان backup و پشتیبانی از داده ها در مواقع آسیب دیدگی و دستکاری اطلاعات .

2- وقتی ما نیاز به اطلاعات بروزآوری شده آنی نداشته باشیم و شبکه پر کاربر و شلوغ باشد .

معایب : یکی از ایرادات جدول SNAPSHOT این است که وقتی درجدول اصلی تغییرپیدا شود تا زمان REFRESH بعدی این تغییردرجدولSNAPSHOT ظاهرنخواهد شد. و همچنین این جدول در سیستم افزونگي ایجاد میکند0

مزیت : مزیت این جدول سرعت بیشتر دسترسی به آن و استفاده از آن به عنوان پشتیبان است .

نکته : جدول دیدگاه مثل آینه در برابر واقعیت است ولی جدول SNAPSHOT مثل عکسبرداری است از یك واقعیت .

جامعيت بانك اطلاعاتی :

يعني صحت، دقت و سازگاري داده‌هاي ذخيره‌شده در پايگاه در تمام لحظات 0

جامعیت درادامه حفظ امنیت سیستم بانک اطلاعاتی است. یک سیستم بانک اطلاعاتی ممکن است بظاهر از جهت امنیت به مخاطره نیافتد ولی جامعیت آن دچار آسیب شده باشد. مثلا ثبت عدد 23 بعنوان نمره یا معدل دانشجو یا انتخاب درسی که اصلا در سیستم کدینگ دروس ثبت نشده است .

انواع قواعد جامعيت :

1- قواعد كاربري User Rules (قواعد خاص)
2- متا قواعد Meta Rules (قواعد عام)

انواع متا قواعد Meta Rules :

الف : قانون جامعيت نهاد Entity Intigriry Rule :

ب : قانون جامعيت ارجاعى : Referentinal Intigriry Rule

1- قواعد كاربري User Rules (قواعد خاص)

قواعدي هستند كه توسط كاربرمجاز تعريف مي‌شوند. وابسته به داده‌هاي جهان خرد هستند، به اين معنا كه در مورد يك پايگاه داده خاص مطرح مي‌شوند و عموميت ندارند. به اين قواعد، قواعد محيطي يا وابسته به داده و يا محدوديتهاي جامعيت معنايي مي‌گويند. به عنوان مثال محدويت فيلد نمره بين 0 تا 20 كه در DDL قابل تعريف است ازجمله شيوه هاي اعمال قواعد جامعيت كاربري ، استفاده از Trigger يا Stored Procedured است .

Trigger : قاعده يا قواعدي (به صورت چند دستور) است كه در پي بروز تغييراتي در بانك اطلاعاتي ، بصورت خودكار اجراء ميشود كه تا سيستم بانك اطلاعاتي دچار ناسازگاري نشود . بعنوان مثال در صورت تغيير نمره يك درس فيلد معدل بطور خودكار بروزآوري شود.

و اما قواعد کاربردی در مدل رابطه ای خود بر چهار دسته اند:

•**قاعده میانی:** قاعده ای است ناظر به یک میدان و مقادیر مجاز آن را مشخص می کند ، مثلاً مقادیر میدان GRADE (نمره) اعداد از صفر تا بیست است.

• **قاعده صفتی(ستونی):** قاعده ای است ناظر به یک صفت(ستون) و بیان کننده نوع آن صفت است،مثلاً STID (شماره دانشجویی) از نوع کاراکتر است.

•**قاعده رابطه ای:** قاعده ای است ناظر به یک رابطه و مقادیر مجاز یک متغییر رابطه ای را مشخص می کند . مثلاً در رابطه COT (درس ها) درس عملی از گروه آموزشی D111 و D444 نمی تواند بیش از یک واحد داشته باشد.

•**قاعده پایگاهی:** قاعده ای است ناظر به دو یا بیش از دو متغییر رابطه ای که به نحوی با یکدیگر مرتبط هستند . مثلاً در رابطه های COT (درس ها) و PROF (اساتید) و STCOPR (رابطه میان درس و استاد) محدودیت زیر می تواند وجود داشته باشد :

استاد با مرتبه دانشیار به بالا نیاید درسی از دوره کاردانی را تدریس کند . (فرض می کنیم که در رابطه COT صفت COLVL به معنای سطح درس را هم داریم) .

نکته: قاعده میدانی طبعاً در مورد صفت (صفاتی) که از یک میدان مقدار می گیرند ، اعمال می شود ولی قاعده صفتی ، قاعده ای است که قبل از هر چیز محدودیتی است که نوع صفت را مشخص می کند.

مثال:

قاعده ۱: شماره دانشجو به صورت dddddddd است که در آن دو رقم اول عددی بزرگتر از ۶۵ است.

قاعده ۲: مقادیر میدان STDEGR: 'bs'، 'ms' و 'doc' است.

قاعده ۳: دانشجوی با معدل کمتر از ۱۲ در یک ترم، نمی‌تواند در ترم بعد بیش از ۱۴ واحد انتخاب کند.

قاعده ۴: مدیر گروه آموزشی حداکثر می‌تواند دو دوره دو ساله، مدیر گروه باشد.

قاعده ۵: هر درس حتماً باید یک منبع اصلی و دو منبع فرعی داشته باشد.

قاعده ۶: رعایت پیش نیاز (ها) در انتخاب درس، الزامی است.

قاعده ۷: در میدان مقادیر GRADE، نمره «ناتمام» وجود ندارد.

قاعده ۸: دانشجوی دوره کارشناسی باید ۱۴۲ واحد بگذراند تا فارغ التحصیل شود.

قاعده ۹: دانشجو نمی‌تواند درس آزمایشگاه را حذف کند.

قاعده ۱۰: تعداد دانشجویان درس‌های تخصصی در هر گروه نباید بیش از ۳۰ نفر باشد.

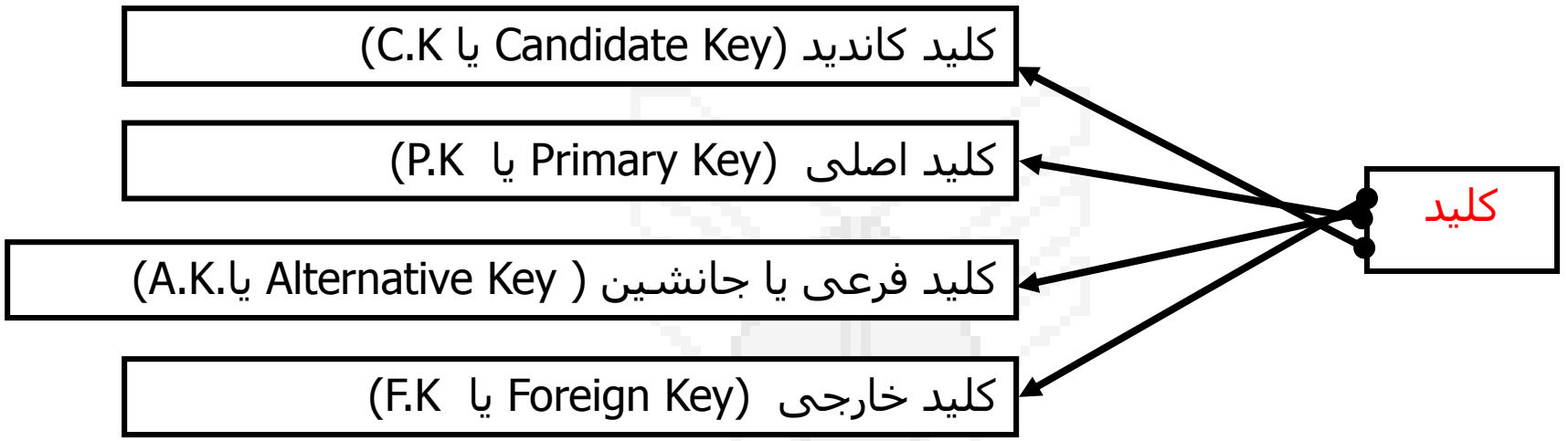
قاعده ۱۱: حداقل نمره قبولی برای دانشجوی دوره کارشناسی ارشد ۱۲ است.

قاعده ۱۲: تعداد واحدهای اخذ شده دانشجوی فعال نمی‌تواند «هیچمقدار» باشد.

2- متا قواعد Meta Rules (قواعد عام)

قواعدي هستند كه بايد توسط هرسيستم رابطه‌اي در هر پايگاه داده رابطه‌اي اعمال شود، وابسته به داده‌هاي خاص نيستند و عموميت دارند.
مثل كليد اصلي ، كليد خارجي ، يا NULL نبودن يك فيلد
متا قواعد ها بعد از تعاريف اوليه كليدها مورد بررسي قرار خواهند گرفت.

از مهمترین ابزارها درحفظ جامعیت بانك اطلاعاتی کلیدها Keys هستند .



كليد كانديد (Candidate Key) :

يك فيلد يا مجموعه اى از فيلد ها كليد كانديد را تشكيل مى دهند (يك Atribute يا مجموعه اى از Atributeها كه تشكيل كليد كانديد را مى دهند) با رعايت دو شرط :
1ـ منحصر به فرد باشد و تكرارى نباشد
2ـ خاصيت غير كاهشى داشته باشد (يعنى اگريك كليد كانديد را بشكنيم كليد كانديد ديگرى از آن بدست نيايد) 0

هررابطه اى درسيستم بانك اطلاعاتى رابطه اى حتما" حداقل يک كليد كانديد دارد

كليد اصلي (Primary Key)

يكي از كليدهاي كانديد كه طراح بانك به صلاحديد خود به عنوان كليد اصلي انتخاب ميكند0

در هر رابطه (جدول) ، فقط و فقط يك كليد اصلي وجود دارد0

كليد جانشين (Alternate Key)

در يك رابطه (جدول) امكان وجود بيش از كليدكانديد وجود دارد كه يكي از آنها حتما" كليد اصلي و باقيمانده كليدهاي كانديد ، كليد جانشين هستند 0
به عبارت ديگر : كليد جانشين = كليد كانديد منهاى كليد اصلى0

كليد خارجي (Foreign Key)

تعريف 1- دورابطه R1 و R2 را در نظر مي‌گيريم . هرزيرمجموعه ازصفات رابطه R2 كه هرمقدار معلومش با يك مقدار از كليد كانديد R1 برابر باشد ، كليد خارجي در رابطه R2 است.

در 2 جدول که با هم رابطه والد و فرزندي دارند ، کلید خارجی کلیدی ست که درجدول فرزند میتواند تکرار شود ولی درجدول والدبعنوان کلید اصلی یا کاندید بوده و تکرار ناپذیر است 0

– در حقیقت برای حفظ قوانین جامعیت و مدیریت ارتباط بین جداول بکار میرود0

مثالی از کلید ها

جدول : student

شماره دانشجوئي	ش پرونده	نام	فاميلی	ش.ش	نام پدر	سال تولد
2	2041	علی	محمدی	11	محمد	1363
5	1275	محمد	ناصری	19	رضا	1362
14	1102	علي	محمدي	456	احمد	1365
6	1590	حميد	كاظمي	867	حامد	1366
3	1561	رضا	رضايی	456	احمد	1363

در جدول بالا " شماره دانشجوئي " و " ش پرونده " هر كدام به تنهائي كليد كانديد و ش دانشجوئی به تنهايي كليد اصلي است 0

در جدول بالا " شماره دانشجوئي " به عنوان كليد اصلي انتخاب شده است 0

كد درس	نام درس	-------	---------
1	فيزيك		
2	مباني كامپيوتر		
3	رياضي 1		
4	معادلات		
6	شيمي		

جدول : dars

ش.دانشجوئي و كد درس در جدول term هر كدام به تنهائي كليد خارجی هستند.

كد ترم	ش دانشجويي	كد درس	نمره
821	14	4	12
821	5	2	17
821	14	2	16
821	5	4	9
821	2	1	16
821	14	6	14
822	6	6	13
822	5	4	11
822	3	1	10
822	5	3	19
822	3	2	18

ش.دانشجوئي و كد درس و كدترم در جدول term با يكديگر كليد كانديد و كليداصلي هستند.

جدول : term

جدول S

S#	S.name	status	City
S1	Sn1	20	C2
S2	Sn2	10	C3
S3	Sn3	30	C2
S4	Sn4	20	C2
S5	sn5	30	C1

جدول SP

S#	P#	Qty
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

P#	P.Name	Color	Wight	City
P1	Nut	Red	12	C2
P2	Bolt	Green	17	C3
P3	Screw	Blue	17	C4
P4	Screw	Red	14	C2
P5	Cam	Blue	12	C3
P6	coy	red	19	c2

جدول P

در جدولS ،کلید اصلی s# است و در جدولP کلید اصلی p# است و درجدول SP ، فیلدهاي S#+P#تواما" کلید اصلي هستند

جدول SP داراي 2 کلید خارجي است :

فیلد s# در جدول SP به عنوان کلید خارجي است 0

فیلد p# در جدول SP به عنوان کلید خارجي است 0

نکته : در هر رابطه حتماً حداقل یک کلید کاندید وجود دارد , زیرا در بدترین حالت کل فیلدها با هم به عنوان کلید کاندید رابطه است.

به عنوان مثال در جدول زیر (s#,p#,j#) کلید کاندید رابطه است .زیرا هیچ یک از صفات خاصه به تنهایی یا دو به دو خاصیت یکتایی مقدار ندارند.

S#	P#	J#
S1	P1	J1
S1	P2	J1
S2	P1	J1
S1	P1	J2

تعریف: رابطه ای که مجموعه عنوانش کلید کاندید آن باشد , اصطلاحاً به رابطه "تمام کلید" یا Super Key موسوم است.

قانون جامعیت نهاد Entity Intigriry Rule :

این قانون میگوید : هیچ جزء تشکیل‌دهنده کلید اصلي نمي‌تواند NULL باشد.
به عبارت دیگر در يك بانك اطلاعات رابطه ای , اطلاعات مربوط به آنچه را که نمی توانیم شناسایی کنیم هرگز ثبت نخواهیم کرد. که کلید اصلي عامل شناسايي يك تاپل (رکورد) است .
یعنی اطلاعات مبهم در بانك وارد نمی کنیم .
(بین اطلاعات **Null** (پوچ) و Blank (خالی) یا صفرفرق وجوددارد
NULL = بيمقداري

نکته: دکتر Codd در کتابش، همان‌طور که اشاره شد، بر «مقدار» بودن این مفهوم تأکید دارد. اما Date (که در حیطه دانش و تکنولوژی سیستم‌های رابطه‌ای، مرجع مسلم است) پس از سال‌ها دفاع از نظر کاد در این مورد، بالاخره به این نتیجه رسید که مفهوم هیچمقدار در مدل رابطه‌ای لازم نیست و حتی اعلام کرد که این مفهوم «مخرب» مدل رابطه‌ای است و به جای آن باید همان مفهوم کلاسیک «مقدار پیش نهاده[1]» را به کار برد تا نشان دهنده «اطلاع نهست» باشد. با این نظر، دیگر نیازی به تکنیک خاص برای نمایش هیچمقدار، شبیه تکنیک متابایت نیست. کافی است «مقدار پیش نهاده» به طور مناسب، در سیستم انتخاب شود.

سایر مؤلفین فقط مفهوم هیچمقدار را مطرح کرده‌اند و به بحث‌های نظری در این مورد نپرداخته‌اند.

نکته: محدودیت هیچمقدارناپذیری، فقط ناظر به کلید اصلی نیست بلکه طراح می‌تواند این محدودیت را در مورد هر صفت دیگر رابطه نیز اعمال کند و در این صورت، جزء قواعد کاربری محسوب می‌شود.

قانون جامعیت ارجاعی : Referentinal Intigriry Rule

این قانون ناظر بر کلید خارجي است و چنین است:
اگر صفت خاصه Ai در رابطه R2 کلید خارجي باشد در این صورت:
Ai در R2 مي‌تواند NULL باشد یا اینکه باید حتما مقداري داشته باشد که در رابطه مرجع R1 وجود دارد. به عبارت دیگرمقدار کلید خارجي يك رابطه نمي‌تواند در رابطه مرجع وجود نداشته باشد.

رعایت قانون جامعیت ارجاعي باعث :

1-عدم نقض روابط کلید خارجی (اگر در رکوردهاي جدول پدر تغییر بدهیم در جدوال فرزند نیز تغییر اعمال شود)

2-عدم وجود رکورد یتیم (رکورد یتیم(Orphan) یعنی يك رکوردی که در جدول فرزند وجود دارد ولی در جدول پدر وجود ندارد)

ترجیحا" قانون جامعیت ارجاعی توسط **DBMS** اعمال میشود چون حتي **DBA** امکان نقض آن را بر اثر فراموشي یا اشتباه دارد0

روشهای کنترل قانون جامعیت ارجاعی :

براي رعايت اين قانون در روابط كليد خارجي ، بنا به نظر طراح بانك از يكي از روشهاي زير استفاده كرد :

1- قانون انتشار یا تسری Cascade

الف : اگر در جدول Parent یا والد ، رکوردی تصحیح یا حذف شود که کلید اصلی آن درجداول فرزند بعنوان کلید خارجی است از استراتژی انتشار (سرایت) استفاده کنیم يعني درجدول والد رکوردي حذف شود که رکورد يا رکوردهاي متناظر آن در جدول Child يا فرزند وجود دارد ، اين رکوردها در جدول فرزند هم حذف می شود) تا قانون جامعيت ارجاعي نقض نشود0

ب : اگر در جدول فرزند ، رکوردی اضافه یا تصحیح شود که کلید خارجي آن درجدول والد بعنوان کلید اصلي است از استراتژی انتشار (سرایت) استفاده کنیم يعني تغييرات رکورد جدول فرزند به جدول والد نيز اعمال شود0

2- قانون محدودیت یا Restrict

الف : اگر در جدول Parent یا والد ، رکوردی تصحیح یا حذف شود که کلید اصلی آن درجداول فرزند بعنوان کلید خارجی است از استراتژی محدودیت (جلوگيري) استفاده کنیم يعني درجدول والد رکوردي بخواهد حذف شود که رکورد يا رکوردهاي متناظر آن در جدول Child يا فرزند وجود دارد ، از اين کار جلوگيري شود تا قانون جامعيت ارجاعي نقض نشود0

ب : اگر در جدول فرزند ، بخواهيم رکوردی اضافه یا تصحیح شود که کليدخارجي آن درجدول والد بعنوان کليد اصلي است از استراتژی جلوگيري يا محدوديت استفاده کنيم 0

3- روش هيچ مقداري NullLifies

در اين روش با حذف رکورد مرجع (در جدول والد) ، رکوردهاي رجوع شونده در جداول فرزند به Null مقدارگذاري شوند بشرط آنکه کليد خارجي جزئي از کليد اصلي در جدول فرزند نباشد. (اين روش توسط Date مطرح نميشود).

4- روش عدم اقدام No Action

در اين روش فقط همان عمل خواسته شده انجام ميشود و اقدام ديگري صورت نميگيرد.

5- روش مقدارگذاري با مقدار پيش فرض Default

در اين روش با حذف رکورد مرجع (در جدول والد) ، رکوردهاي رجوع شونده در جداول فرزند به مقدار پيش فرض مقدارگذاري شوند

FOREIGN KEY (Attribute(s)) REFERENCES Relation - name

DELETE **Option**

UPDATE **Option**

مقدار Option يکي از 5 حالت زير است :

Cascade

Restricted

NullLifies

No Action

Set To Default

مثال ايجاد يك جدول

```
CREATE  TABLE   Stud
  (STID    CHAR(8)  NOTNULL
  COID    CHAR(6)    NOTNULL
  TR     CHAR(1)
  YRYR  CHAR(5)
  GRADE  DECIMAL(2,2)
PRIMARY  KEY (STID, COID)
FOREIGN  KEY (STID)  REFERENCES STT
      DELETE    CASCADE
      UPDATE    CASCADE
FOREIGN  KEY (COID) REFERENCES COT
     DELETE CASCADE
     UPDATE CASCADE )
```

راههاي كلي اعمال قوانين جامعيت در سيستم هاي بانك اطلاعاتي :

1- معرفي كليد اصلي در هر رابطه (جدول) Primary Key

2- اعلام صفت (فيلد) غير پوچ (Not Null)

3- معرفي كليد خارجي Foriegn Key

4- اعلام محدويت هاي مورد نظر از طريق شماي بانك اطلاعاتي (Diagrams)

5- نوشتن راه انداز (Trigger)

6- معريف ميدان – مثلا" دامنه تغيير فيلد احتمال وقوع يك حادثه : 0 الي 100 درصد

7- معرفي وابستگي بين صفات خاصه

تراکنش (Transaction) :

مجموعه عمل واحدی که یا باهم انجام میشود یا باهم انجام نمیشود.

تراکنش یك واحد منطقی از کار است و معمولاً شامل چندین عمل بانك اطلاعاتی است. تفاوت اصلی یك تراکنش با یك برنامه معمولی در محیط غیربانکی این است که تراکنش همواره به DBMS تسلیم می‌شود و DBMS در اعمال هرگونه کنترل و حتی به تعویق انداختن و ساقط کردن آن آزادی عمل دارد.

هدف اصلی این کنترلها حفظ جامعیت و صحت بانك اطلاعاتی است. چرا که در بانك اطلاعاتی آنچه در درجه اول اهمیت دارد داده است نه برنامه. داده‌های بانك اطلاعاتی را مانا (Persistent) می‌نامند زیرا برنامه‌ها می ایند و می‌روند اما داده‌ها می‌مانند.

مثلاً برنامه‌ای که پولی را به حسابی می‌ریزد یا برداشت می‌کند در درجه دوم اهمیت است . مهم این است که داده های یك سیستم بانك اطلاعاتي دچاره نقض یا خطا نشود.

تضمین جامعیت بانك اطلاعاتی در مدیریت تراکنش ها :

آقای جیم‌گری (Jim Gray) در سال 1981 ثابت کرد که چهار کنترل زیر لازم است روی تمامی تراکنش‌ها در بانك اطلاعات اعمال گردد تا صحت و جامعیت آن تضمین شود این کنترلها به خواص ACID معروفند.

1ـ یکپارچگی (Atomicity)

2ـ همخوانی (Consistency)

3ـ انزوا (Isolation)

4ـ پایائی (Durability)

1ـ یکپارچگی (Atomicity) :

این خاصیت به همه یا هیچ موسوم است. منظور این است که یا تمام دستورات یك تراکنش باید اجراء شود یا هیچکدام از آنها نباید اجراء شود. مثلاً تراکنشی می‌خواهد مبلغی را از حسابی به حساب دیگر منتقل کند. فرض کنید بخش اول کار (برداشت پول) در یك ماشین و بخش دوم کار (واریز پول) در ماشینی دیگر اجراء می‌شود. حال در نظر بگیرید پس از انجام بخش اول (برداشت پول) ارتباط با ماشین دوم ناگهان قطع شود بدیهی است که در این حالت باید پول برداشت شده دوباره به همان حساب اول بازگردانده شود.

2ـ همخوانی (Consistency) :

این خاصیت به این صورت بیان می‌گردد که :«هر تراکنش اگر به تنهایی اجراء شود بانك اطلاعات را از حالتی صحیح به حالت صحیح دیگری منتقل می‌کند» یعنی این تراکنش ممکن است دو نوع پایان داشته باشد : الف) پایان ناموفق که آنرا سقوط (Abort) می‌نامند ب) پایان موفق که آنرا انجام (Commit) می‌نامند.

3ـ انزوا (Isolation) :

در بانك اطلاعاتی ممکن است تراکنش‌های همروند وجود داشته باشد (مثل Multitasking در سیستم عامل Windows که چند برنامه همزمان اجراء می‌شوند). بر طبق خاصیت انزوا همروندی تراکنش‌ها باید کنترل شود تا اثر مخرب بر روی هم نداشته باشند به عبارتی دیگر اثر تراکنش‌های همروند روی یکدیگر چنان است که گویا هر کدام در انزوا انجام می‌شود. این کنترل توسط بخشی از DBMS به نام واحد کنترل همروندی (Concurrency Control) انجام می‌شد.

4ـ پایایی (Durability) :

براساس این خاصیت تراکنش‌هایی که به مرحله انجام (Commit) برسند اثرشان ماندنی است و هرگز به طور تصادفی از بین نمی رود. مثلاً اگر مبلغی به حسابی واریز شود تراکنش مربوطه انجام یافته اعلام شود حتی در صورت وقوع آتش‌سوزی در آن شعبه بانك، مشتری نباید متضرر شود، یعنی مثلاً عمل واریز قبل از اعلام انجام موفق باید در جای دیگری نیز ثبت شده باشد.

هر تراکنش با تعیین محدویت زیر در هر برنامه مشخص و تعریف میگردد :

Begin Transaction

....

....

....

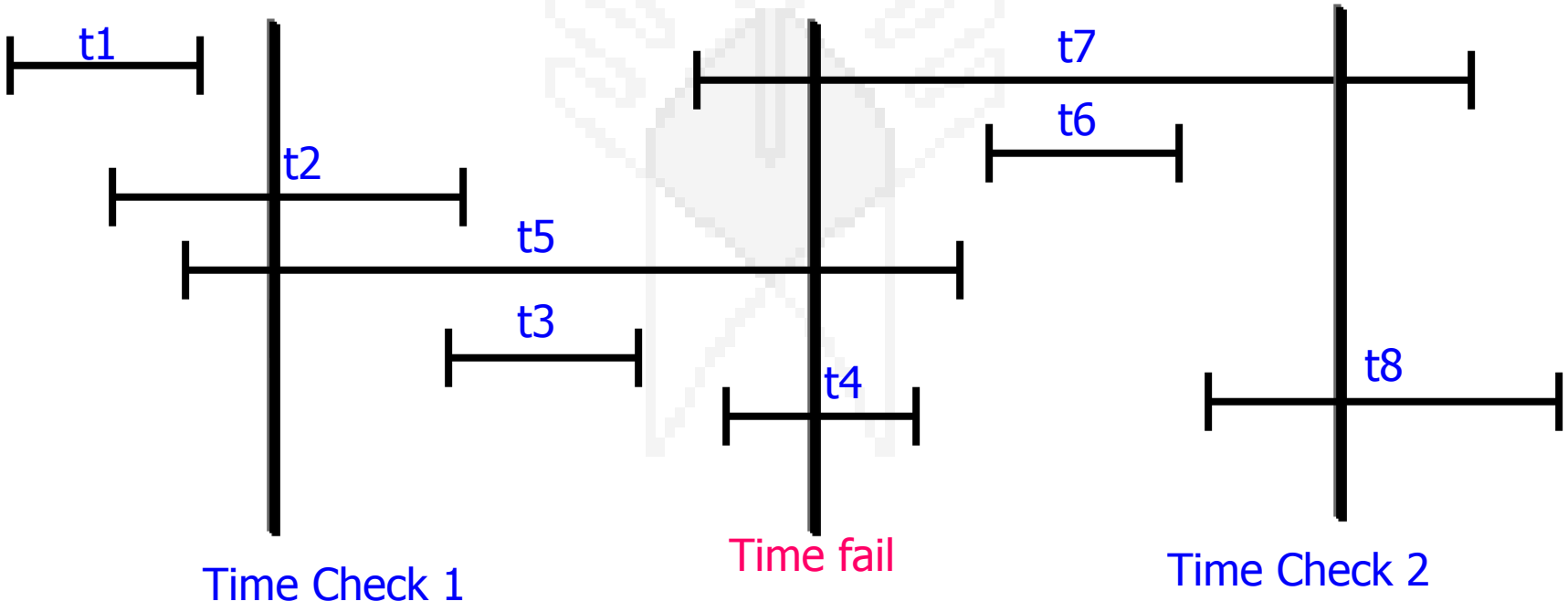
End Transaction

هرتراکنش به دو صورت خاتمه میابد : در صورت موفقیت تائید یا Commit میشودو درغیراینصورت Abort میگردد.

برای Abort تراکنش های ناموفق ، سیستم بانك اطلاعاتی باید به وضعیت قبل ازشروع تراکنش برگردد که برای این منظور برنامه RollBack اجراء میگردد. برای اینکار سیستم یك امکان "ثبت وقایع" (Journal یا LogFile) که شامل جزئیات مربوط به عمل بهنگام رسانی ست ، ضبط میشود که در صورت خطا یا هرگونه اشکال درسیستم ، تراکنش لغو که سیستم بانك اطلاعاتی را به وضعیت قبل بر میگرداند.

دقت شود که Commit و RollBack به تراکنش خاتمه میدهد نه به برنامه

به مثال زیر توجه کنید :
سیستم بانك اطلاعاتی در زمان Time Check 1 چك شده (شامل پشتیبانی و نگهداری وضعیت کلیه تراکنشهای در حال اجراء) ، سپس سیستم در زمان TF دچار خطا شده است . در صورت شروع مجدد سیستم مشخص کنید کدام تراکنش t دوبار اجراء (ReDo) و کدامیك Abort یا (UnDo) میگردد :



- ✓ قانون کلي : کلیه تراکنشهایی که قبل از خطا Commit شده و تائیدیه گرفته اند ، در شروع مجدد ReDo و بقیه Undo میشوند.
- ✓کلیه تراکنش هایی که در بازه tc1 تا tf تائیدیه گرفته اند یا به اتمام رسیده اند بایدRedo شوند .
- ✓ کلیه تراکنش هایی که درزمان time check 1 تا time fail در حال انجام بودند بایدundo (cancel) شوند .
- ✓ به تراکنش های که بعد از tf شروع شده اند کاری نداریم (چون به طور واقعي وجود ندارند).
- ✓ T2,T3 باید redo شوند .
- ✓ T5,T4 ,T7 باید undo شوند .
- ✓ T6 ,T8 بی موردند .

مقدمه نرمال سازي

اگر در يك سيستم بانك اطلاعاتي همه جداول به هم پيوند داده شود ما فقط با يك جدول سروكار ميداشتيم و اينكار چه نتيجه اي در بر ميداشت ؟

جواب : مزيت عمده آن ساده تر شدن درخواست ها و پرس و جوها از بانك اطلاعاتي و عدم نياز به پيوندها (Join) و در نتيجه افزايش سرعت پاسخ گوئي

به عنوان مثال : به سه جدول معروف SP,P,S (فصل های قبل) نگاه كنيد. فرض كنيد جدول SP را به صورت (S#,P#,Qty,Status) تعريف و جدول S را به صورت (S#,Sname,City) تعريف مي كرديم، يعني فيلد Status را از جدول S به جدول SP مي‌برديم. در اينصورت جدول به صورت روبرو مي‌شد:
همانطور كه مشاهده مي‌گردد فيلد Status براي S1 و S2 همواره ثابت و مشخص است(عدد 20 و 10) و بي جهت در جدول تكرار شده و بدين جهت افزونگي اطلاعات داريم

SP'

S#	P#	Qty	Status
S1	P1	300	20
S1	P2	200	20
S1	P3	400	20
S1	P4	200	20
S1	P5	100	20
S1	P6	100	20
S2	P1	300	10
S2	P2	400	10

معايب عمده :

1- افزونگي داده ها (Data Redundancy)

کد دانشجويي	نام	فاميلی	ش. ش	سال تولد	نام پدر	آدرس	تلفن	نام درس	نمره
2	علی	محمدی	11	1360	محمد	رشت	2332	رياضي	13
5	محمد	ناصری	11	1362	رضا	زنجان	3251	رياضي	14
6	عليرضا	كاظمي	354	1358	حسين	انزلي	2121	فيزيك	18
5	محمد	ناصری	11	1362	رضا	زنجان	3251	فيزيك	15
6	عليرضا	كاظمي	354	1358	حسين	انزلي	2121	شيمي	17
2	علی	محمدی	11	1360	محمد	رشت	2332	فيزيك	16
6	عليرضا	كاظمي	354	1358	حسين	انزلي	2121	شيمي	19
6	عليرضا	كاظمي	354	1358	حسين	انزلي	2121	فارسي	16
3	رضا	رضايی	451	1360	احمد	تبريز	2324	فارسي	9

2- بي‌نظمي (Anomaly) :

طراحي نامناسب يا ناقص رابطه هاي (جداول) يك سيستم بانك اطلاعاتي ، باعث ايجاد بي‌نظمي يا آنومالي (Anomaly) ميگردد 0

آنومالي (Anomaly) بر مبناي 3 حالت زير اتفاق مي افتد :

الف : آنومالي در درج - عدم امكان عمل درج (Insert) در جدول به علت عدم تعريف تمام يا بخشي از كليد اصلي كه باعث نقص قانون جامعيت ميشود0

به عنوان مثال اضافه كردن نام درس و نمره در جدول قبل غير ممكن است ، چون شماره دانشجويي (كليد اصلي) معلوم نيست

ب : آنومالي در حذف - بروز تبعات نامطلوب روی انجام عمل حذف (Delete) روي يك ركورد 0

- حذف اطلاعات يك يا چند دانشجو ميتواند باعث حذف مشخصات درس نيز گردد0

ج : آنومالي در بروزآوري - بروز فزونكاری در سيستم در انجام عمل بروزآوري (Update)

- اگر اسم يك درس تغيير يابد و در همه ركوردهاي دانشجوياني كه اين درس را انتخاب كرده اند ، بروزآوري نشود ، باعث ناسازگاري اطلاعات و باعث نقض جامعيت ميشود0

رابطه SPC مفروض است

جدول SPC

S#	P#	QTY	CITY
S1	P1	100	C2
S1	P2	200	C2
S1	P3	80	C2
S2	P1	70	C1
S2	P2	120	C1
S3	P1	90	C1
S4	P1	80	C3
S4	P3	90	C3

مثال آنومالي نوع الف :

-براي درج يا اضافه كردن (Insert) اطلاع <S7, C7> در جدول ، دچار آنومالی میشویم . چون تا ندانیم چه قطعه ای تهیه کرده است زیرا کلید اصلي رابطه SPC ، (S# و P#) است و درج تاپل بدون داشتن آنها ناممکن است .

-مثال آنومالي نوع ب :

- حذف اطلاع < S3 , P1 , 90 > انجام شدنی است ، اما اطلاع ناخواسته حذف می شود (اینکه S3 ساکن C1 است) .

-مثال آنومالي نوع ج :

بهنگام سازی : شهر S1 را عوض کن .

```
UPDATE   SPC
SET CITY = 'C1'
WHERE S# = 'S1'
```

این عمل ، منطقا" تاپلی (مربوط به يك ركورد) ، تبدیل به عمل منطقا" مجموعه ای (مربوط به گروهي از ركوردها) و نهایتا" منجر به بروز فزونکاری در سیستم می شود.

نتيجه اوليه :

مشخص شد كه رابطه SPC داراي آنومالی است و دليل غير تئوريک اين آنومالي اينست كه در رابطه SPC پديده اختلاط اطلاعاتي وجود دارد .

يعني اطلاعات در مورد موجوديت محصول (P#) با اطلاعات تهيه كننده (S#) مخلوط شده است . شهر(City) از صفات خاصه تهيه كننده (S#) است و با صفات خاصه محصول (P#) تركيب شده است ، همين اختلاط اطلاعاتي سبب بروز پديده افزونگي هم شده است .

رابطه SPC خوش طرح (Well Design) نيست و در مراحل بعد خواهيم ديد كه اين طراحي بايد تغيير كند .

3- مشكل مقادير Null :

اگر فرض كنيد در مثال قبل بخواهيم ثبت نام اوليه دانشجو (ثبت اطلاعات پايه اي) را انجام دهيم كه هنوز انتخاب واحدي را انجام نداده باشد بايد مقدار *NULL* به جاي اسم درس و نمره وارد كنيم 0

تعاريف اوليه در نرمالسازي :

شرح صور نرماليتي :

ابتدا سطوح کلاسیک کادی را بررسی می کنیم و برای این منظور نیاز به دو مفهوم از تئوری وابستگی داریم :

1- وابستگي تابعي - Functional Dependency (FD)

صفت خاصه Y به صفت خاصه X از رابطه R وابستگی تابعی دارد اگر به ازای هر مقدار از X در R فقط و فقط یک مقدار برای Y وجود داشته باشد .

مثال : رابطه R مفروض است :

رابطه R

X	Y	Z
X1	Y1	Z1
X1	Y1	Z2
X2	Y1	Z2
X2	Y1	Z3
X3	Y2	Z3

$$R.X \longrightarrow R.Y$$

آیا $X \rightarrow Y$ ؟ بله

آیا $Y \rightarrow X$ ؟ خیر

آیا $X \rightarrow Z$ ؟ خیر

نكات مهم :

1- وابستگي تابعي براي همه اطلاع (ركوردها) ي ممكن (ركوردهاي موجود و ممكن آينده) بايد مصداق داشته باشد 0 يعني وابستگي به معني و ذات صفتها (فيلدها) مربوط است نه به موارد خاص يا صرفا" اطلاعات موجود0

2- وابستگي تابعي ميتواند براي هر بانك اطلاعاتي متفاوت باشد0

امكان يا عدم امكان تدريس يك درس در يك ترم توسط بيش از يك استاد

استاد <---- (ترم ، درس)

3- DBMS بايد امكان بدهد تا مجموعه FD های محيط معرفي شوند ، بعنوان مجموعه قواعد Semantic ناظر برمحيط كه خود نوعي قواعد جامعيتي هستند (Dependency Integrity Rules) .
عملا اين قواعد در يک طراحي خوب از طريق مفهوم كليد اصلي و كليد كانديد به يک سيستم انتقال می يابد.

نکته : FD ها در " خرد جهان واقع " (محیط عملیاتي) تفسیر دارند . در واقع بیانگر قواعد معنایی در محیط عملیاتی هستند (قوانین Semantic ناظر بر محیط عملیاتي) .

S# → CITY

قاعده Semantic : هر تهیه کننده در یک شهر دفتر دارد .

S# → STATUS

قاعده Semantic : هر تهیه کننده یک مقدار وضعیت دارد .

مثال : در یک محیط آموزشی :

PR # → CO#

یعنی یک استاد فقط یک درس را تدریس می کند .

CO# --/→ PR#

یک درس توسط اساتید مختلف تدریس می شود.

A	B	C
A1	B1	C1
A2	B2	C3
A1	B1	C2
A3	B4	C2
A5	B1	C1

مثال : در رابطه R1 وابستگي تابعي A ---> B و C ---> B را بررسي كنيد

وابستگي A ---> B برقرار است زيرا به ازاي هيچ مقدار مساوي از A ، دو مقدار متفاوت از B وجود ندارد0

وابستگي B ---> C برقرار نيست زيرا به ازاي B1 دو مقدار C1 و C2 در رابطه R1 وجود دارد

مثال : رابطه S را در نظر می گیریم

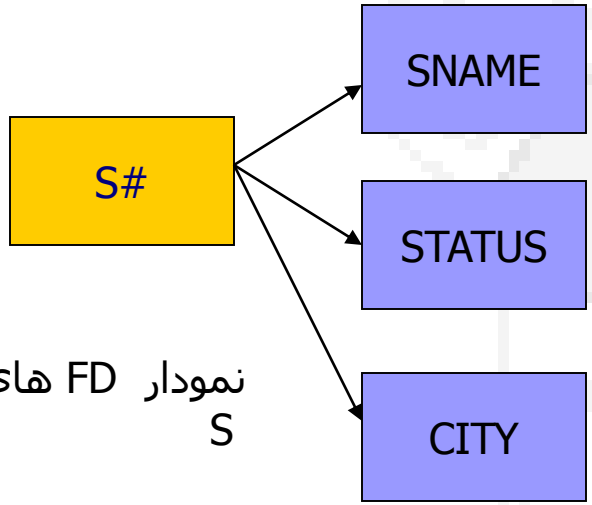
S (S# , SNAME , STATUS , CITY)

کلید کاندید است S#.

S # → SNAME

S # → STATUS

S # → CITY



نمودار FD های رابطه S

چون S# کلید کاندید است این وابستگیها محرز است .

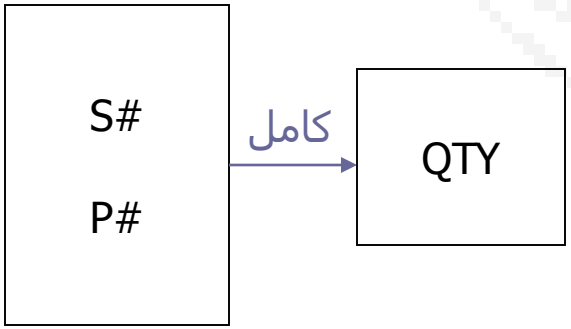
نتیجه : تمام صفات خاصه یک رابطه در طول حیات آن با کلید کاندید آن FD دارند .

2- مفهوم وابستگي تابعي كامل (Full Functional Dependency) (FFD)

صفت خاصه Y از رابطه R به صفت خاصه مركب X از R وابستگي تابعي كامل دارد

با نمايش R.Y <== R.X اگر و فقط اگر Y با X وابستگي تابعي داشته باشد اما با هيچ يک از اجزای تشكيل دهنده X وابستگي تابعي نداشته باشد .

مثال : در رابطه SP (S# , P# , QTY) ، (S# و P#) كليد اصلي است .



(S# , P#) → QTY

P# ↛ QTY

S# ↛ QTY

مثال : قواعد زير در يك سيستم بانك اطلاعاتي برقرار است نمودار FD مربوطه به آن را ، بهمراه يك جدول با مقادير دلخواه متناطر با آن را ايجاد كنيد :

قاعده 1 : هر ناشر تعدادي كتاب منتشر ميكند0

قاعده 2 : هر كتاب به تعداد خاصي منتشر ميشود0

قاعده 3 : هر ناشر در يك شهر دفتر دارد0

قاعده 4 : هر ناشر داراي يك رتبه انتشاراتي است0

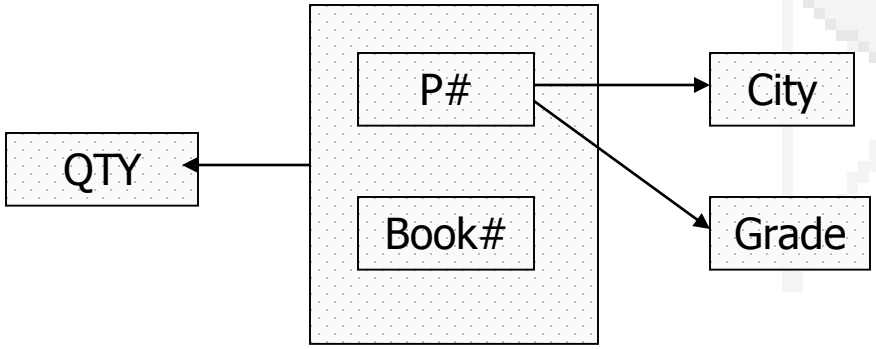
قاعده 5 : ناشران يك شهر داراي يك رتبه انتشاراتي هستند0

جواب :

P#شماره ناشر - BOOK# شماره كتاب - QTY تعداد انتشار - City شهر - Grade رتبه

1) P# --/→ Book# 2) (P# , Book#) --→ QTY 3) P# --→ City

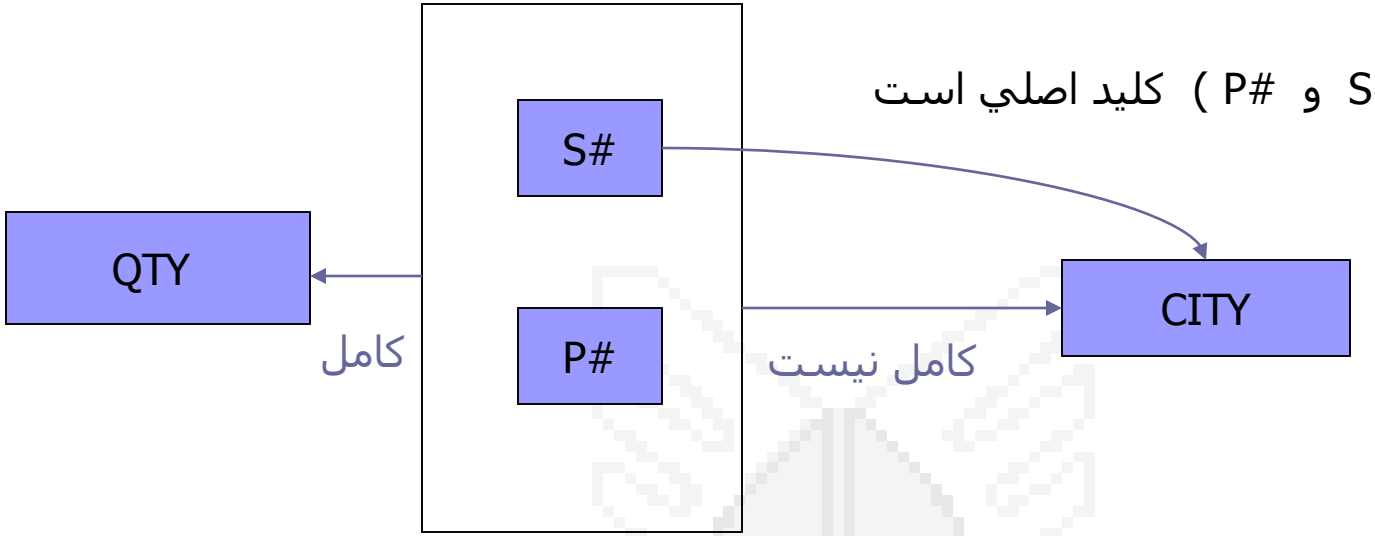
4) P# --→ Grade 5) City --→ Grade



P#	Book#	QTY	City	Grade
P1	B1	5000	C1	40
P2	B4	3000	C2	60
P3	B3	2000	C1	40
P1	B2	5000	C1	40
P4	B3	4000	C3	50
P4	B5	3500	C3	50

SPC (S# , P# , QTY , CITY)

در رابطه فوق ، (S# و P#) کلید اصلي است



مشخص است که اگر سمت چپ FD کامل ، صفت خاصه ساده باشد همان FD معمولي است و کامل نبودن موضوعیت نخواهد داشت .

A	B	C
A1	B1	C1
A2	B1	C3
A1	B2	C2

مثال : در رابطه R وابستگي تابعي كامل (A,B) ==> C را بررسي كنيد

- رابطه (A,B) --> C برقرار است .
- رابطه (A,B) ==> C برقرار است چون C به تنهايي به A و يا B وابسته نيست 0

تعريف : اگر براي تمام صفتهاي B در رابطه R داشته باشيم A→B آنگاه A را ابر كليد (Super Key) رابطه R مينامند و بصورت A → R نمايش ميدهند

اگر اين وابستگي از نوع FFD باشد آنگاه A كليد كانديد رابطه R است 0

تعريف : اگر B زير مجموعه اي از A باشد آنگاه A→B

اين وابستگي تابعي را بديهي (Trivial FD) گويند 0

نكته : در يك بانك اطلاعاتي ما بدنبال اطلاع از همه وابستگي ها هستيم 0 تعدادي از وابستگي هاي بين صفتها بشكل ساده تري قابل فهم و شناسايي هستند ولي بايد ساير وابستگي ها را بايد بدست آورد 0

هر گاه تعدادي وابستگي داشته باشيم ممكن است بتوانيم وابستگي هاي ديگري نيز از آنها بدست آوريم و يا بعضي از آنها را كه اطلاعات جديدتري به ما نمي دهند ، حذف كنيم.

مثال : رابطه مفروض Rرا داريم :

R=(S , T , U , V , W)

F=(S→T , V→SW , T→U)

آنگاه منطقا" ميتوان وابستگي هاي زير را نيز بدست آورد :

S→U (S→T , T→U زيرا)

V→W

V→S

V→T (V→S , S→T زيرا)

V→U (V→S , S→U زيرا)

پس نتيجه ميگيريم كه V كليد كانديد است .

مثال: در بانک اطلاعاتی زیر ابتدا کلید کاندیدا را یافته و سپس آن را نرمال سازی کنید.

R={A,B,C,D,E,F,G}

F={AF ⟶ BE,FC ⟶ DE,F ⟶ CD,D ⟶ E,C ⟶ A}

حل: ابتدا سمت راست وابستگی ها را به صفت تبدیل می کنیم

AF ⟶ B AF ⟶ E FC ⟶ D

FC ⟶ E F ⟶ C F ⟶ D

D ⟶ E C ⟶ A

از شماره 5 برای ساده کرن سمت چپ 3 و 4 استفاده می کنیم . در نتیجه:

3)F ⟶ D 4)F ⟶ E

از شماره 5 و 8 می توان نتیجه گرفت A F این نتیجه را بر روی 1 و 2 اعمال می کنیم:

1)F ⟶ B 2)F ⟶ E

در نتیجه داریم: F={F ⟶ A,F ⟶ B,F ⟶ C,F ⟶ D,F ⟶ E,D ⟶ E,C ⟶ A}

در نتیجه F همه صفت ها دیگر به جز G را می دهد.پس (F,G) کلید کاندیدا است .

1NF : F , G , A , B , C , D , E

2NF : (F , G)

F , A , B , C , D , E

3NF : (F , G)

(C , A)

(D , E)

(F , B , C , A)

مجموعه پوششي وابسته :

از يك مجموعه وابستگي ، ميتوان وابستگي هاي ديگر استخراج نمود و به آن افزود.

تعريف :

اگر F يك مجموعه از وابستگي هاي تابعي باشد آنگاه مجموعه تمام وابستگي هاي تابعي كه از آن منتج ميشود را مجموعه پوششي F ميناميم و با F+ نمايش ميدهيم

آقاي آرمسترانگ در سال 1974روشي براي استخراج پوششي ارائه داد و اثبات كرد كه سه قاعده زير براي براي استخراج مجموع پوششي كافي است ، يعني با اعمال مكرر اين سه قاعده ميتوان به تمام وابستگي هاي منتج دست يافت و هيچ وابستگي اضافي نيز توليد نميشود :

1- بازتاب : اگر B زير مجموعه A باشد آنگاه A→B

2- افزايش : اگر A→B و C صفت خاصه باشد آنگاه : AC→BC

3- انتقال : اگر A→B و B→C آنگاه A→C

هر چند قوانين براي استخراج F+ كافي بود ولي اعمال آنها مشكل بود و بعدها قوانين ديگري به آن اضافه كه كار را تسهيل ميكرد كه مهمترين آنها :

4- اجتماع (Union) : اگر A→B و B→C آنگاه A→BC

5- تجزيه (Decomposition) : اگر A→BC آنگاه A→B و A→C

6- تركيب (composition) : اگر A→B و C→D آنگاه AC→BD

مجموعه وابستگي بهينه :

با اعمال قواعد فوق وابستگي زيادي بدست مي آيد كه بعضا" تكراري و اضافي هستند و در اينجا روشي براي حذف اينگونه وابستگي ها و رسيدن به مجموعه وابستگي بهينه ارائه ميشود.

تعريف :

دومجموعه وابستگي تابعي F1 و F2 را معادل (Equivalant) ميناميم اگر مجموعه پوششي آنها برابر باشند يعني F1+ = F2+

با استفاده از قواعد 3 گانه زير ميتوانيد مجموعه وابستگي را به مجموعه بهينه معادل آن تبديل كرد :

1- سمت راست هر وابستگي فقط يك صفت باشد.

2- هر صفتي كه F+ را تغيير نمي دهد از سمت چپ حذف شود.

3- وابستگي هاي تكراري و اضافي حذف شود.

بطور خلاصه بايد گفت : كه براي يافتن وابستگي هاي تابعي در يك بانك اطلاعاتي ميبايست ابتدا مجموعه پوششي وابستگي ها را تعيين كرد و سپس آنرا بهينه نمود.

مثال :

در بانک زیر ، مجموعه وابستگی پوششی بهینه را بیابید .

R=(U , V , W , X , Y , Z)

F={U $\longrightarrow$ XY , X $\longrightarrow$ Y , XY $\longrightarrow$ ZV , U $\longrightarrow$ ZV}

حل: ابتدا F^+ را می یابیم و سپس آن را بهینه می کنیم.

F^+={U $\longrightarrow$ XY , X $\longrightarrow$ Y , XY $\longrightarrow$ ZV , U $\longrightarrow$ ZV}

قاعده 1 روی وابستگی اول:

F^+={U $\longrightarrow$ X , U $\longrightarrow$ Y , X $\longrightarrow$ Y, XY $\longrightarrow$ ZV , U $\longrightarrow$ ZV}

قاعده 2 روی ZV $\longrightarrow$ XY با توجه به Y $\longrightarrow$ X :

F^+={U $\longrightarrow$ X , U $\longrightarrow$ Y , X $\longrightarrow$ Y, XY $\longrightarrow$ ZV , U $\longrightarrow$ ZV}

حال قاعده 1 روی سایر وابستگی ها :

F_{opt}={U $\longrightarrow$ X , U $\longrightarrow$ Y , U $\longrightarrow$ Z , U $\longrightarrow$ V, X $\longrightarrow$ Y, X $\longrightarrow$ Y , X $\longrightarrow$ Z , X $\longrightarrow$ V}

كليدهاي كانديد :

اگر مجموعه اي از صفتها را ATTR و مجموعه وابستگي تابعي آنها و تعدادي صفت ديگر را F بناميم آنگاه الگوريتم زير ، مجموعه تمام صفت هاي وابسته به ATTR را ميدهد (بهتر است ابتدا F را بهينه كنيم) .

$ATTR^+ = \text{ATTR}$

تكرار كن

براى هر Y ⟵ X در F

اگر X زير مجموعه $ATTR^+$ باشد آنگاه:

$ATTR^+ = \text{ATTR}^+ \text{UV}$

تا زمانيكه $ATTR^+$ ديگر تغيير نكند .

با استفاده از این الگوریتم می توان کلیدهای کاندیدا را بدست آورد . به طور کلی باید به نکات زیر توجه کرد:

1-هر کلید کاندیدا شامل مجموعه ای از صفت هایی است که در سمت چپ پیکان ها می آیند .

2- کلید کاندیدا باید کمینه باشد ، یعنی زیر مجموعه ای از آن خاصیت کلیدیی نداشته باشد.

3- بانک اطلاعات ممکن است چند کلید کاندیدا داشته باشد.

4- کلید کاندیدا ممکن است در یک یا چند صفت مشترک باشند .

•مثال: اگر R = (S , TU , V , W) و F={ S ⟶ T , V ⟶ SW , T ⟶ U }

آنگاه : { S , V, T , W , U } {S , V , T , W } $\{S,V\}^+$ = { S , V }

می توان این الگوریتم را برای مجموعه صفت های دیگر سمت چپ پیکان مرتباً بکار برد و وابستگی های مختلف و نیز کلیدها را بدست آورد .

اگر در مثال فوق V^+ را محاسبه کنیم خواهیم دید که { S , V } کاهش پذیر است:

V^+ ={ V } ⟹ { V , S , W } ⟹ { V , S , W , T } ⟹ { V , S , W , T , U }

در نتیجه {S,V} کلید کاندیدا نیست ولی V هست .

5-هرگاه تعداد صفت ها در بانک زیاد و وابستگی آنها گسترده باشد ، می توان با رسم نمودار وابستگی به فهم بهتر آن بانک اطلاعات کمک کرد . در این نمودار ، صفت ها در مستطیل قرار می گیرند و پیکانی از آنها به هر یک از صفت های وابسته به آن رسم می شود . برای مجموعه صفت ها و وابستگان آنها نیز مستطیل های دیگری رسم می گردد .

معمولاً پیکان هایی که وابستگی به کلید اصلی را نشان می دهند در بالای صفت ها و سایر پیکان ها زیر آنها کشیده می شوند . ابتدا باید مجموعه بهینه وابستگی ها را بدست آوریم و آنگاه به رسم نمودار وابستگی مبادرت کنیم .

مثال: در بانک اطلاعات زیر ابتدا همه کلیدهای کاندیدا را بیابید و سپس نمودار وابستگی را رسم کنید.

R=(U , V , W , X , Y , Z , O , P , Q)

F={U⟶ VXQ , UVP⟶ O , OQ ⟶ YZ , UP ⟶ XY}

حل: ابتدا مجموعه بهینه معادل F را پیدا می کنیم(تمرین):

F={U⟶V , U ⟶ X , U⟶ Q , OQ⟶Y , OQ ⟶Z , UP⟶ Y , UP ⟶O , UP⟶ Z}

سپس مجموعه صفت های وابسته به تمام مجموعه صفت های چپ پیکان ها را می یابیم :

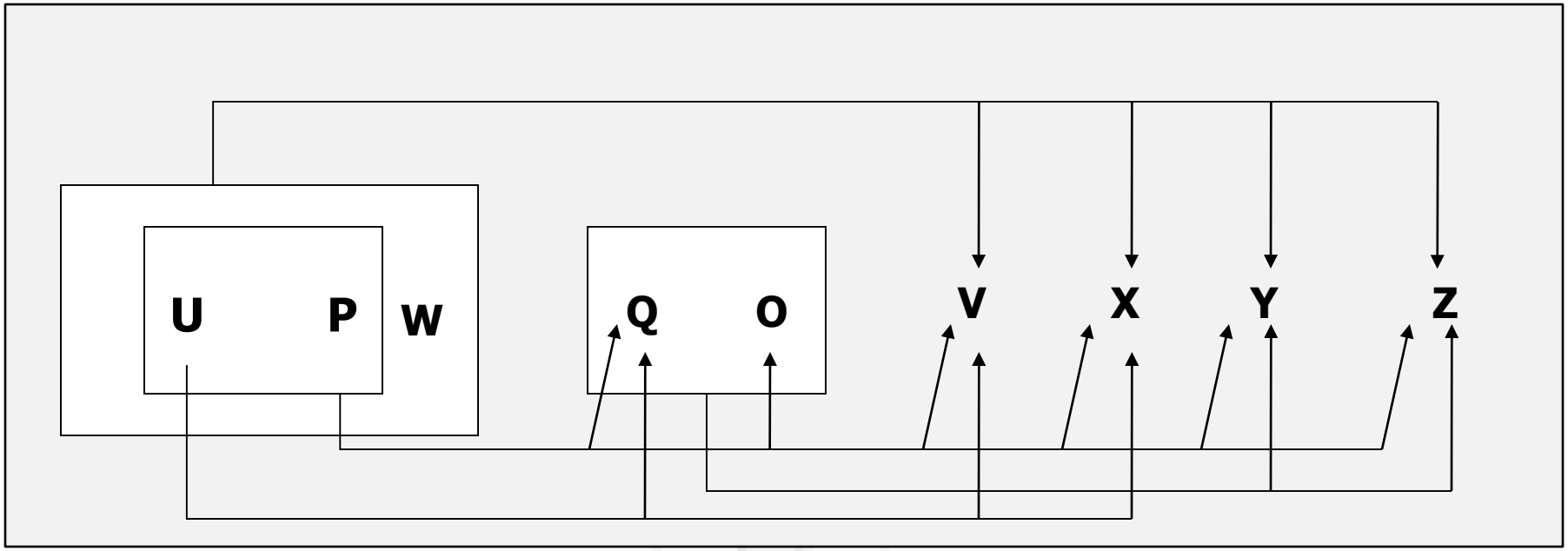
U^+ = { U } ⟶ { U , V , X , Q }

UP^+ = { U , P } ⟶ { U , P , V , X , Q , O , Y } ⟶ { U , P , V , X , O , Q , Y , Z }

OQ^+ = { O , Q } ⟶ { O , Q , Y , Z }

نتیجه می گیریم که بیشترین صفت ها را UP^+ می دهد . .تنها صفتی که از UP^+ خارج است W است . پس UPW کلید کاندیدا می باشد . یعنی UPW ⟶ R کلید کاندید دیگری به دست نمی آید زیرا از O و Q نمی توان به U و P رسید (یاد آوری می شود که کلید کاندیدا باید کمینه باشد) .

نمودار وابستگی این بانک اطلاعات در شکل زیر مشاهده می شود.



شکل - مثالی از نمودار وابستگی تابعی

تمرین: وابستگی P ⟶ OQ را به مثال فوق اضافه کنید . آیا کلید کاندید دیگری خواهیم داشت؟

تعريف نرمال سازي

تجزيه يك رابطه يا يك جدول به روابط كوچكتر با رعايت دو شرط زير را نرمال سازی می گویند :

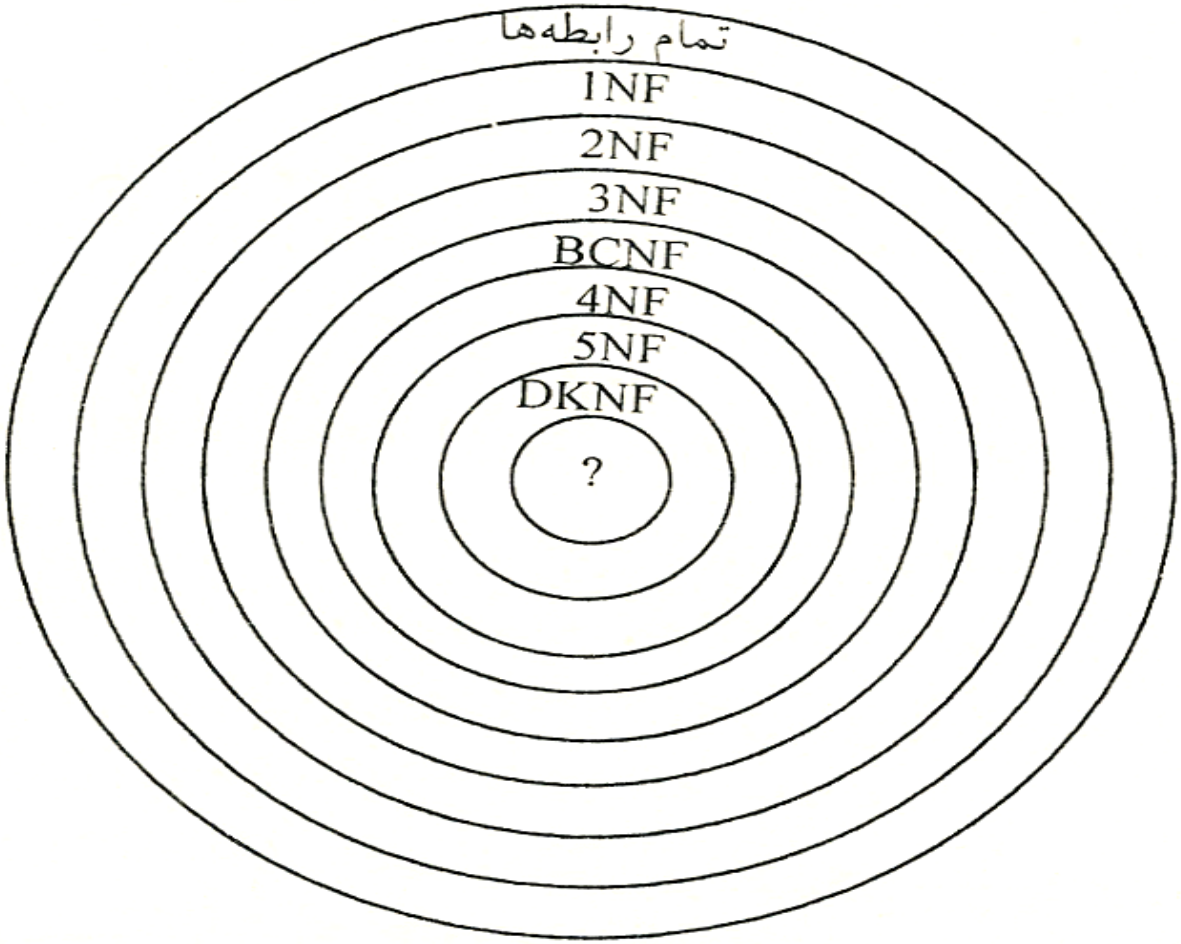
1 ـ خاصيت برگشت پذيری (اتصال جداول تجزيه شده به حالت اول)

2 ـ خاصيت بدون از دست دادن اطلاعات 0(با اتصال مجدد جداول اطلاعاتي از دست نرفته باشد)

نكته : هرچه رابطه نرمالتر باشد آنومالی كمتر می شود.

ملاك تجزيه رابطه : تجزيه رابطه R به رابطه هاي R1 و R2 ميبايست به نحوي باشد كه پيوند دو رابطه R1 و R2 ، رابطه R را ايجاد كند و تاپلي (ركورد يا سطري) كم و يا زياد نشود و از طرف ديگر تجزيه R ميبايد ، وابستگي هاي تابعي را حفظ كند.

درجات مختلف نرمالسازي :



فرم نرمال درجه اول NF1 (First Normal From) :

رابطه R به فرم 1NF (First Normal From) است اگر و فقط اگر :

1- همه كليدهاي آن تعريف شده باشند0
2- تمام صفات خاصه آن روي ميدانهاي اتميك تعريف شده باشند به عبارتي ديگر صفت هاي آن از دامنه تودرتو (Nested Domain) نباشد. يعني صفت تركيبي نداشته باشيم. به عبارت ساده تر هر سلول از جدول فقط وفقط يك مقدار معين داشته باشد . مثل فيلد نام شامل (نام و نام فاميلي) نباشد يا فيلد آدرس بدون توجه به اجزاء شهر – خيابان و ... نباشد

مثال از رابطه از درجه يك نرمال :

First (S#	P#	QTY	CITY)
S1	P1	100	C1
S1	P2	80	C1
S2	P3	90	C2
S2	P1	60	C2
S2	P2	80	C2
S3	P1	70	C2
S4	P1	120	C1

مثال: رابطه کارمند را در نظر بگیرید ، آیا این رابطه در سطح 1NF است؟

کد پرسنلی	میزان تحصیلات	شماره شناسنامه	نام
110	دیپلم	243	علی علوی
140	کارشناسی	2719	ستایش یمقانی
130	کاردانی	593	رضا قاسمیان

پاسخ: فیلد نام قابل تجزیه به دو فیلد نام و نام خانوادگی است . اگر بخواهیم نام خانوادگی کارمندان را از رابطه بدست آوریم ، فیلد نام را تجزیه کرده ایم بنابراین رابطه غیر نرمال است و در سطح 1NF قرار ندارد . شکل نرمال 1NF آن به صورت زیر است:

کد پرسنلی	میزان تحصیلات	شماره شناسنامه	نام خانوادگی	نام
110	دیپلم	243	علوی	علی
140	کارشناسی	2719	یمقانی	ستایش
130	کاردانی	593	قاسمیان	رضا

مثال: جدول ST را در نظر بگیرید:

کلید اصلی جدول ، #ST می باشد .

St

St#	Name	میزان تحصیلات				
7801	علی	1400	پایگاه داده	3	79-2	20
		1500	ریاضی 1	80-1	10	3
		1600	تجزیه و تحلیل	3	80-1	20
7902	آرش	1400	پایگاه داده	3	80-1	7
		1700	تربیت بدنی	1	80-1	20

برای تبدیل این جدول به حالت نرمال 1 ، باید آن را به فرم زیر تبدیل کنیم :

کلید اصلی جدول St# + Crs# + term می باشد .

St

St#	Name	Crs#	Course	unit	Grade	term
7801	علی	1400	پایگاه داده	3	20	79-2
7801	علی	1500	ریاضی1	3	10	80-1
7801	علی	1600	تجزیه و تحلیل	3	20	80-1
7902	آرش	1400	پایگاه داده	3	7	80-1
7902	آرش	1700	تربیت بدنی	1	20	80-1

نرمال سازی درجه دوم :

يك رابطه از درجه دوم است اگر و فقط اگر از :

1- درجه اول نرمال باشد .

2- تمام صفتهاي خاصه ي(فيلدهاي) آن به كليد اصلي وابستگي تابعي (FD) داشته باشند0

3- كليه صفات غير كليدی (غير كليد اصلی و كليد كانديد) به زير مجموعه هاي كليد اصلی وابستگي نداشته باشند. (وابستگي جزئي Partial dependency)

جدول 1

کد دانشجو	نام	فامیلی	ش.ش	سال تولد	نام پدر	آدرس	تلفن
2	علی	محمدی	11	1360	محمد	رشت	2332
5	محمد	ناصری	11	1362	رضا	زنجان	3251
3	رضا	رضایی	451	1360	احمد	قزوین	2324

در جدول بالا کد دانشجو کلید اصلی است و نام ، فامیلی ، ش.ش، سال تولد ، نام پدر ، آدرس ، تلفن صفات غیر کلیدي و هر کدام به تنهايي به کلید اصلي وابسته اند . پس جدول بالا از درجه دوم نرمال است 0

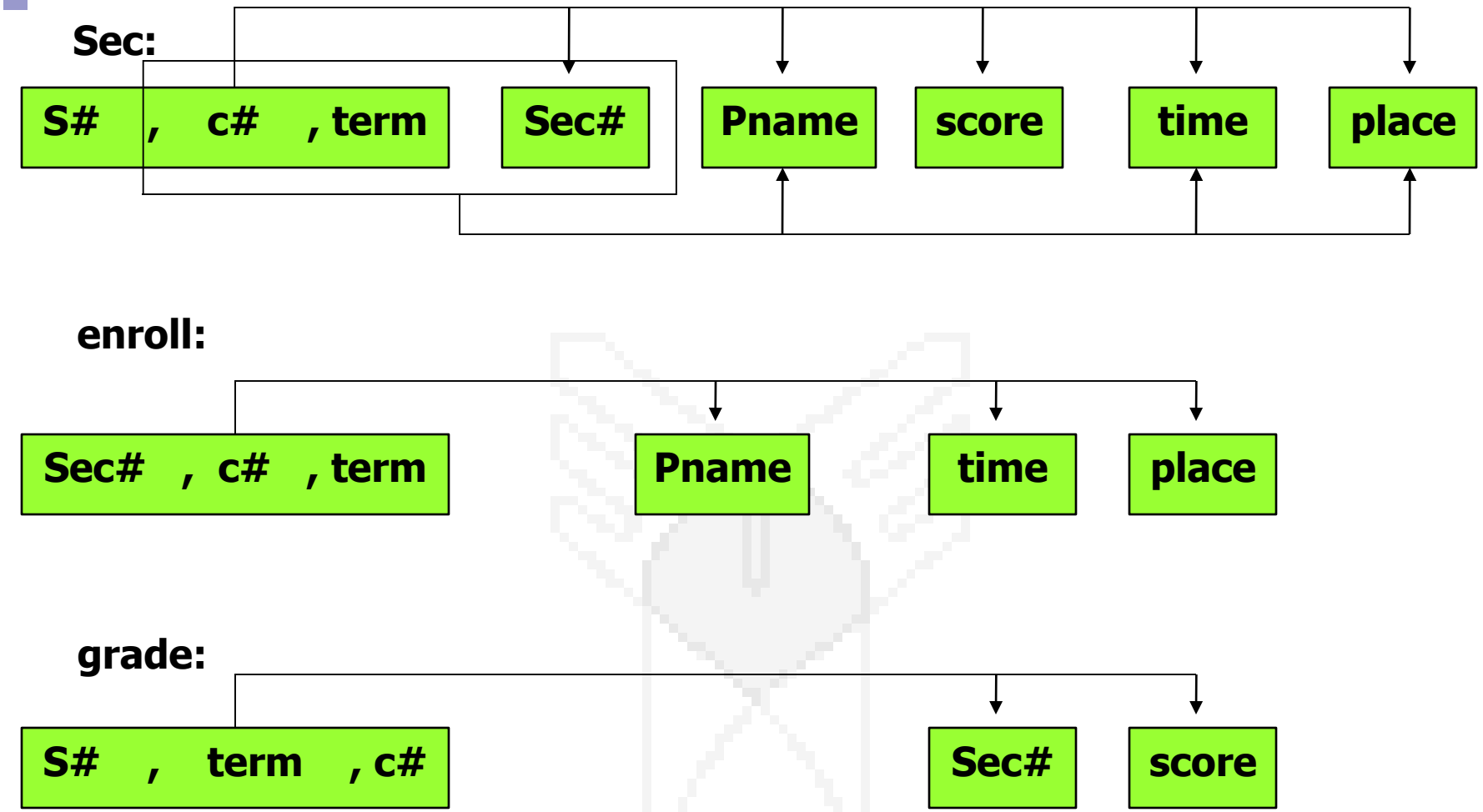
الگوریتم زیر ، جدول 1NF را به جدول 2NF تجزیه می کند :

1- هر بخش از کلید اصلی را که ایجاد وابستگی جزئی کرده است ، با همه وابسته های آن کنار بگذارد .

2- کل کلید اصلی را با صفت های باقیمانده کنار هم بگذارد .

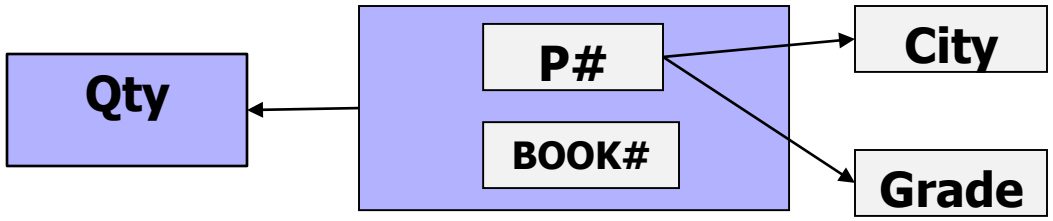
3- صفت های کلیدی را به عنوان کلید خارجی به 2 اضافه کن .

از سه جدول شکل صفحه بعد فقط جدول SEC در 2NF نیست . با پیروی از الگوریتم بالا جدول SEC به دو جدول enroll و grade تجزیه می شود (شکل صفحه بعد) .



شکل – مراحل تبدیل بانک اطلاعات ثبت نام دانشگاه از 1NF به 2NF

مثال: رابطه وقتی در سطح 1NF باقی می ماند که وابستگی تابعی کامل بین حداقل یک فیلد با کلید اصلی نقص شود . به عنوان مثال نمودار FIRST را در نظر بگیرید:



کلید اصلی در این رابطه صفت مرکب (BOOK# , p#) است . از آنجایی که تمام فیلدها کلید FD دارند ؛ داریم:

(p# , BOOK#) ⟶ CITY

برای برقراری وابستگی تابعی کامل باید داشته باشیم: BOOK# ↛ CITY , P# ↛ CITY

اما همانطور که در نمودار FD قابل مشاهده است داریم: P# ⟶ CITY

در نتیجه FD کامل بین CITY و (BOOK# , p#) نقض شده است .

برای رفع آنومالی و افزایش سطح نرمال باید رابطه FIRST به صورت زیر تجزیه شود :

SECOND (P# , CITY , GRADE)

PB (P# , CITY , GRADE)

P#	CITY	GRADE
P1	C1	40
P2	C2	60
P3	C1	40
P1	C1	40
P4	C3	50
P4	C3	50

P#	BOOK#	QTY
P1	B1	1000
P2	B4	3000
P3	B3	2000
P1	B2	3000
P4	B3	4000
P4	B5	3500

رابطه SECOND در سطح دوم نرمال قرار دارد زیرا : 1- در 1NF است 2- وابستگی تابعی کامل نقض نشده اما این رابطه نیز دارای آنومالی است .

St

St#	Name	Crs#	Course	Unit	Grade	Term
7801	علی	1400	پایگاه داده	3	20	79-2
7801	علی	1500	ریاضی1	3	10	80-1
7801	علی	1600	تجزیه و تحلیل	3	20	80-1
7902	آرش	1400	پایگاه داده	3	7	80-1
7902	آرش	1700	تربیت بدنی	1	20	80-1

برخی از جداول در حالت نرمال1 هستند ولی هنوز ناهنجاری دارند . مثلاً با آن که جدول St به نرمال 1 تبدیل شد ، ولی هنوز دارای ناهنجاری های زیر است :

1- ناهنجاری در حذف:

فرض کنید علی تنها دانشجویی باشد که درس ریاضی 1 را گرفته است . اگر لازم باشد اطلاعات علی را حذف کنیم ، به طور ناخواسته اطلاعات ریاضی1 را نیز از دست می دهیم .

2- ناهنجاری در درج :

فرض کنید بخواهیم اطلاعات دانشجوی جدیدی را که هنوز هیچ درسی را نگرفته است درج کنیم . از آنجا که Course و Term جزیی از کلید اصلی هستند نمی توانند مقدار تهی داشته باشند . انجام این عمل ممکن نیست .

3- ناهنجاری در اصلاح :

فرض کنید بخواهیم تعداد واحد درس پایگاه داده را از 3 به 4 تغییر دهیم . در این صورت برای حفظ سازگاری داده ها لازم است این تغییر را به دفعات مکرر و در تاپل های مختلف اعمال کنیم (اصلاح منتشر شونده) .

مثال : جدول St که در مثال قبل تبدیل به نرمال1 شد ، نرمال2 نیست چرا که کلید اصلی این جدول ، St# + Crs# + Term است در حالی که تنها با داشتن St# می توان به Name رسید و یا تنها با داشتن Crs# می توان به Cname و Unit رسید .

برای تبدیل یک جدول نرمال1 به نرمال 2 مراحل زیر را دنبال کنید :

1- کلیه ترکیبات ممکن میان اجزا کلید اصلی را به ترتیب در سطرهای مجزا بنویسید (اول ترکیبات یک جزیی ، سپس ترکیبات دو جزیی ، سپس ترکیبات سه جزیی و . . .) :

St#
Crs#
Term
St# + Crs#
St# + Term
Crs# + Term
St# + Crs# + Term

2- در کنار هر یک از ردیف ها ، ویژگی هایی را که در تشکیل کلید اصلی نقشی ندارند و به ویژگی های مورد نظر وابستگی تابعی دارند و در سطرهای بالاتر نوشته نشده اند بنویسید . مثلاً با داشتن یک St# تنها به یک Name می رسیم ، پس Name به St# وابستگی تابعی دارد . Name را در ردیف اول اضافه کرده و در ردیف های بعدی ، آن را در نظر نمی گیریم :

St# , Name
Crs# , Cname , Unit
Term
St# +Crs#
St# + Term
Crs# + Term
St# + Crs# + Term , Grade

3- کلیه ردیف هایی را که هیچ ویژگی غیر کلیدی در آنها وجود ندارد و شامل هیچ اطلاعات مفیدی که در جداول دیگر نیز وجود ندارد حذف کرده و سایر سطرها را به جدول مجزا تبدیل کنید .

Student (St# , Name)
Course (Crs# , Cname , Unit)
SC (St# , Crs# , Term , Grade)

به این ترتیب جدول St به جداول زیر تبدیل خواهد شد :

Student

St#	Name
7801	علی
7902	آرش

Course

Crs#	Cname	Unit
1400	پایگاه داده	3
1500	ریاضی1	3
1600	تجزیه و تحلیل	3
1700	تربیت بدنی	1

SC

St#	Crs#	Term	Grade
7801	1400	79-2	20
7801	1500	80-1	10
7801	1600	80-1	20
7902	1400	80-1	7
7902	1700	80-1	20

مثال: در بانک اطلاعاتی زیر ابتدا کلید کاندیدا را یافته و سپس ان را نرمال سازی کنید .

R={A,B,C,D,E,F,G}

F={ AF ⟶ BE , FC ⟶ DE , F ⟶ CD , D ⟶ E , C ⟶ A}

حل: ابتدا سمت راست وابستگی ها را به صفت تبدیل می کنیم :

AF ⟶ B

AF ⟶ E

AF ⟶ D

F ⟶ C

F ⟶ D

D ⟶ E

C ⟶ A

از شماره5 برای ساده کردن سمت چپ 3 و 4 استفاده می کنیم . در نتیجه خواهیم داشت :

3) F ⟶ D 4) F ⟶ E

از شماره 5 و 8 می توان نتیجه گرفت A ⟶ F و این نتیجه را بر روی 1 و 2 اعمال می کنیم :

1)F ⟶ B 2) F ⟶ E

در نتیجه داریم :

F={ F ⟶ A , F ⟶ B , F ⟶ C , F ⟶ D , F ⟶ E , D ⟶ E , C ⟶ A}

در نتیجه F همه صفت ها دیگر به جز G را می دهد . پس (F , G) کلید کاندیدا است .

1NF : F , G , A , B , C , D , E

2NF : (F , G)

F , A , B , C , D , E

3NF : (F , G)

(C , A)

(D , E)

(F , B , C , A)

مثال :

کد دانشجو	کد درس	نام درس	تعداد واحد	نمره قبولی	نمره
2	251	عربی	2	10	16
5	363	ریاضی	3	10	15
1	256	شیمي	2	10	9
2	363	رياضي	3	10	17
5	251	عربي	2	10	16
3	251	عربي	2	10	15
1	363	رياضي	3	10	13
3	256	شيمي	2	10	18

در جدول بالا کددانشجو و کددرس (هر دو با هم) کلید اصلی هستند 0
نام درس ، تعداد واحد و نمره قبولی به کددرس وابسته است و نمره به کلیداصلي (کد درس و کد دانشجو) وابسته است.
پس جدول بالا از درجه دوم نرمال نیست 0

جدول دوم بصورت زیر نرمال شده وبه دو جدول تقسیم میشود:

2A

کد درس	نام درس	تعداد واحد	نمره قبولی
251	عربی	2	10
363	ریاضی	3	10
256	شیمي	2	10

2B

کد دانشجو	کد درس	نمره
2	251	16
5	363	15
1	256	9
2	363	17
5	251	16
3	251	15
1	363	13
3	256	18

در جدول 2A ، کددرس ، کلید اصلی است و در جدول 2B کد دانشجو و کد درس باهم کلید اصلی هستند (کد درس در این جدول کلید خارجي ست) بنابر این حالا هر دو جدول ما نرمال هستند .

نرمال سازی درجه سوم :

يك رابطه به شكل سوم نرمال است اگر و فقط اگر اولا از درجه دوم باشد و ثانيا تمامی صفات غيركليدي به يكديگر وابسته نباشد(يعني وابستگي انتقالي نداشته باشند) .

مثال:

كد كالا	كدفنی	شرح كالا	قيمت	كدرنگ	نام رنگ
1	3AB	مداد	150	4	قرمز
2	5BF	خودكار	250	3	آبی
4	7RF	صندلي	350	1	مشكي
6	4PO	مداد	75	1	مشكي
5	3KJ	ميز	850	4	قرمز
3	7NS	تراش	400	2	سبز

در جدول بالا كد فنی ، شرح كالا ، قيمت و كد رنگ همه به كد كالا وابسته هستند ولی نام رنگ هم به كد كالا وابسته است وهم به كد رنگ ، بنابراين ، اين جدول از درجه دوم نرمال ولی از درجه سوم نرمال نيست بخاطراينكه نام رنگ هم به كليد اصلی و هم به صفت غيركليد يعنی همان كد رنگ وابسته است
بنابر اين جدول بالا را به فرم درجه 3 ميتوان نرمال كرد 0

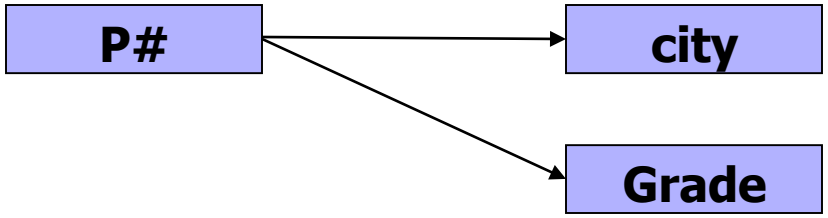
کد کالا	کدفنی	شرح کالا	قیمت	کدرنگ
1	3AB	مداد	150	4
2	5BF	خودکار	250	3
4	7RF	صندلي	350	1
6	4PO	مداد	75	1
5	3KJ	ميز	850	4
3	7NS	تراش	400	2

کدرنگ	نام رنگ
4	قرمز
3	آبی
1	مشکي
2	سبز

تذکر: در جدول قبل کد کالا و کد فنی کلید های کاندید هستند,که می توان یکی از آنها را به دلخواه کلید اصلی انتخاب کرد,که ما در اینجا کد کالا را کلید اصلی در نظر می گیریم.

مثال: رابطه SECOND را در نظر می گیریم . نمودار FD آن به صورت زیر است :

1) p# ⟶ city
2) city# ⟶ Grade
3) P# ⟶ Grade



از دو وابستگی تابه اول و دوم می توان وابستگی تابعی سوم را نتیجه گرفت ، یعنی Grade وابستگی تابعی با واسطه P# از طریق city دارد .
علت بروز آنومالی Second نیز همین عامل یعنی وابستگی تابعی با واسطه است .
برای رفع آنومالی آن می باید این رابطه تجزیه شود . بنابراین Second را به رابطه های PC و CS تجزیه می کنیم :

PC (P# , City)
CS (City , Grade)

PS

P#	City
P1	C1
P2	C2
P3	C1
p4	c3

CS

City	Grade
C1	40
C2	60
C3	50

با حذف آنونالی های Second ، روابط جدید نا تنها در سطح 3NF قرار می گیرند ؛ بلکه در سطح BCNF قرار می گیرند .

الگوريتم تبديل روابط نرمال از درجه 2 به نرمال درجه 3 :

1- صفتي كه وابستگي انتقالي ايجاد كرده با ديگر صفات وابسته آنرا در كنار هم بگذار.

2-كليد اصلي را با صفتهاي باقيمانده كنار هم بگذار.

3- صفتهاي كليدي را به عنوان كليد خارجي به 2 اضافه كن.

برخی از جداول ، در حالت2 نرمال هستند ولی هنوز مشکلاتی دارند . به عنوان مثال جدول Professor که شامل شماره، نام ، کد آخرین مدرک تحصیلی و نان آخرین مدرک تحصیلی است ؛ در نظر بگیرید :

Prof

Prof#	Pname	LastDegree#	LastDegreeName
100	علی راد	2	کارشناسی ارشد
101	آرش رضایی	2	کارشناسی ارشد
102	عسل شاملو	3	دکترا

این جدول ، نرمال2 است زیرا نرمال1 است و آز انجا که کلید اصلی تنها یک جز دارد ، بدون شک هیچ وابستگی جزیی در آنوجود ندارد . با این جود این جدول هنوز دارای ناهنجاری های زیر است:

2- ناهنجاری در حذف:

فرض کنید عسل شاملو تنها کسی باشد که مدرک دکترا دارد . در این صورت اگر بخواهیم اطلاعات عسل شاملو را حذف کنیم ، اطلاعات مربوط به مدرک دکترا (کد مدرک و نام مدرک) را نیز از دست می دهیم .

2- ناهنجاری در اصلاح:

اگر بخواهیم نام مدریک 2 را از کارشناسی ارشد به کارشناسی تغییر دهیم ، در این صورت مجبوریم این تغییر را در تاپل های مختلف تکرار کنیم (اصلاح منتشر شونده) .

جدول Professor نرمال 3 نیست چون :

LastDegree# ⟶ LastDegreeName

برای تبدیل جداول نرمال 2 به نرمال 3 ، مراحل زیر را دنبال کنید :

1- ویژگی هایی را که در وابستگی متعددی شرکت دارند را در یک جدول مجزا قرار داده و ویژگی هایی را که در طرف چپ این وابستگی قرار دارند ، به عنوان کلید اصلی معرفی کنید .

Degree (LastDegree# , LastDegreeName)

2- بقیه ویزگی های جدول اولیه را در جدول مجزای دیگری قرار داده (Pname,prof#) ، ویژگی های تشکیل دهنده کلید اصلی در جدول اول (lastDegree#) را به آنها اضافه کنید .

Prof(Prof# , Pname , LastDegree#)

Degree

Lastdegree#	LastDegreeNmae
2	کارشناسی ارشد
3	دکترا

Prof

Prof#	Pname	LastDegree#
100	علی راد	2
101	آرش رضایی	2
102	عسل شاملو	3

در دیدگاه آقای کاد، اینجا پایان راه است . آقای بویس روی یک نکته انگشت گذاشت و فرم نرمال خود BCNF (BOYCE_CODD NORMAL FORM) را تعریف کرد . این نکته در زیر بررسی می شود .

3NF با جداولي كه هر سه شرط زير را دارند دچار مشكل ميگردد :

1- جدول حداقل داراي دو كليد كانديد باشد.

2- اين كليدهاي كانديد تركيبي باشند.

3-اين كليدهاي تركيبي ، صفت هاي مشتركي داشته باشند.

مثال1 : جدول دانشجو در یک موسسه کوچک که دانشجوی همنام ندارد (در این صورت هم شماره دانشجویی کلید کاندیدا است و هم نام دانشجو) :

Stud (s# , Sname , address , avg)

Candidate key (S#)

Candidate key (Sname)

وابستگی ها :

S# ⟶ Stud

Sname ⟶ Stud

این جدول در 3NF هست . از سه شرط بلا فقط شرط 1 را داراست . بنابراین نیاز به نرمال سازی بیشتر ندارد . با در نظر گرفتن داده های مناسب می توان دید که این جدول افزونگی ندارد .

مثال2: جدول "درس – دانشجو " در همین موسسه :

Crs_stud (s# , sname , c# , score)

Candidate key (s# , c#)

Candidate key (sname , c#)

وابستگی ها:

(s# , c#) ⟶ crs_stud

(sname , c#) ⟶ crs stud

S# ⟶ sname

Sname ⟶ s#

این جدول هر سه شرط بلا را دراست . بنابراین ممکن است نیاز به نرمال سازی بیشتر داشته باشد . داده های زیر نشان می دهد که این جدوا دارای افزونگی است و برای برطرف نمودن آن نیازی به نرمال سازی بیشتر دارد و می توان این جداول را به دو جدول st(s# , sname) و grade (s# , c# , score) شکست (در جدول grade می توان به جای s# از sname نیز استفاده کرد .

Crs_stud

S#	Sname	C#	Score
S1	علی	C1	17
S1	علی	C2	12
S1	علی	C3	19
S1	علی	C4	20
S1	علی	C5	10

افزونگی

مثال3: جدول امتحان شامل شماره دانشجویی ، شماره درس ، رتبه دانشجو در درس (غیر تکراری) :

Exam (s# , subj# , rank)

Candidate key (s# , subj#)

Candidate key (subj# , rank)

وابستگی ها:

Key (s# , subj#) ⟶ Exam

Key (subj# , rank) ⟶ Exam

این جدول در 3NF هست و هر سه شرط را نیز داراست ، ولی نیاز به نرمال سازی بیشتر ندارد (با در نظر گرفتن داده هایی می توان دید که این جدول افزونگی ندارد) .

چگونه باید فهمید که چنین جداولی نیاز به نرمال سازی دارند یا خیر ؟ پاسخ این سوال را باید در تعریف BCNF جستجو کرد . اگر جدولی در فرم BCNF باشد ، نیاز به شکستن ندارد .

•تعریف: جدولی BCNF است که ستون های آن فقط به کلید های کاندیدش وابستگی تابعی داشته باشد .

در مثال 1 و 3 ، وابستگی به غیر کلید های کاندیدا وجود ندار ولی در مثال2 این نوع وابستگی وجود دارد (sname ⟶ s# , s# ⟶ sname) .

توجه به نکات زیر در مورد BCNF ضروری است :

1- برخلاف فرم های نرمال دیگر ، BCNF بدون استفاده از 3NF و فرم های قبلی نرمال تعریف می شود . غالباً می توان از بانک اطلاعات را با استفاده از تعریف BCNF در یک قدم نرمال کرد و نیازی به تعریف وابستگی هایی از قبیل وابستگی انتقالی نیست .

بنابراین ، BCNF جامع ترین تعریف نرمال سازی بر مبنای وابستگی تابعی را به طور مستقل ارائه می دهد .

2- در مواردی ، نرمال سازی تا BCNF لازم نیست و بهتر است از تبدیل جدول فرم 3NF به BCNF خودداری کرد .

به عنوان مثال ، برای ثبت آدرس فعلی و آدرس خانواده دانشجویان می توان جدول زیر را تعریف کرد و برای هر دانشجو بیش از یک آدرس درظر گرفت :

S_addr (s# , city , no , zip)

Candidate key(s# , city)

Candidate key(s# , zip)

کد پستی برای هر شهر منحصر به فرد است و می تواند به جای نام شهر قرار بگیرد . این جدول در فرم 3NF هست ولی در فرم BCNF نیست زیرا داریم : $Zip \longrightarrow city$

با این همه نمی توان با اطمینان خاطر این جدول را شکست زیرا کد پستی ، بخش جدایی ناپذیر است و جدا کردن آن باعث پیچیده شدن پرس و جو های مربوط به آدرس می شود .

چون وابستگی $zip \longrightarrow city$ نیز برقرار است ، همین مطلب در مورد 2NF هم صادق است . به طور خلاصه باید گفت که پرس و جو نیز می تواند در طراحی جداول نقش داشته باشد . اگر طراح تشخیص دهد که تجزیه یک جدول ، هرچند افزونگی هم داشته باشد ؛ باعث پایین آمدن سرعت اکثر پرس و جوها می شود ، مجاز است از نرمال تر سازی آن صرف نظر کند .

برخی از جداول در حالت نرمال3 هستند ولی هنوز ناهنجاری دارند . به عنوان مثال دانشگاهی با قوانین زیر را در نظر بگیرید :

1- هر دانشجو می تواند در چند رشته تحصیلی ، تحصیل کند ولی در هر یک از رشته های تحصیلی ، تنها یک استاد راهنما دارد .

2- هر استاد تنها می تواند در یک رشته تحصیلی تدریس کند ولی در هر رشته تحصیلی چندین استاد وجود دارند .

جدول Project را که شامل شماره دانشجویی ، رشته تحصیلی و استاد راهنمای دانشجو در رشته مربوطه است را در نظر بگیرید :

Project

St#	Field	Tutor
7801	مهندسی کامپیوتر	مجید رضایی
7801	ریاضی محض	آرش ریاضی دان
7801	هنر	گلناز هنردوست
7902	مهندسی کامپیوتر	مجید رضایی
8001	مهندسی کامپیوتر	پروین صبا

در جدول Project ، دو کلید کاندیدا وجود دارد :

کلید کاندیدای 1 : St# + field

کلید کاندیدای 2 : St# + Tutor

فرض کنید کلید کاندیدای اول را به عنوان کلید اصلی انتخاب کنیم .

جدول Project ، نرمال2 است چون در آن هیچ ویژگی غیر کلیدی به قسمتی از کلید اصلی ، وابستگی ندارد (تنها ویژگی غیر کلیدی این جدول ، Tutor است که نه به St# تنها وابسته است و نه به Field تنها) .

این جدول نرمال3 نیز هست چون در آن هیچ ویژگی کلیدی به ویژگی غیر کلیدی دیگر ، وابستگی ندارد (چون این جدول تنها یک ویژگی غیر کلیدی دارد ، این مساله بدیهی است) .

با وجود آن که این جدول نرمال3 است ، ولی هنوز ناهنجاری هایی دارد :

1- ناهنجاری در درج :

به عنوان مثال ، نمی توان استاد جدیدی را که هنوز هیچ دانشجویی با وی پروژه نگرفته است ، در جدول درج کرد چون St# جزیی از کلید اصلی است و نمی تواند تهی باشد .

2- ناهنجاری در حذف :

فرض کنید دانشجوی شماره 80001 تنها دانشجویی باشد که با پروین صبا پروژه گرفته است ؛ در این صورت اگر این دانشجو را حذف کنیم ، اطلاعات مربوط به این استاد نیز از بین می رود .

جدول Project نرمال BCNF نیست چون : Tutor ⟶ Field

پس Tutor ، یک تعیین کننده است در حالی که کلید کاندیدا نیست .

تبدیل جداول نرمال2 به نرمال BCNF کاملاً مشابه روند تبدیل جداول نرمال2 به نرمال3 است .

برای تبدیل جدول Project به جدول نرمال BCNF :

1- Tutor و Field را در یک جدول مجزا قرار داده ، Tutor را به عنوان کلید اصلی این جدول معرفی می کنیم .

TC (Totur , Field)

2- بقیه ویژگی ها (ST#) را در جدول دیگری قرار داده ، کلید اصلی جدول اول یعنی Tutor را به آن ها اضافه می کنیم .

ST (St# , Tutor)

TC

Tutor	Field
مجید رضایی	مهندسی کامپیوتر
آرش ریاضی دان	ریاضی محض
گلناز هنردوست	هنر
مجید رضایی	مهندسی کامپیوتر
پروین صبا	مهندسی کامپیوتر

ST

St#	Tutor
7801	مجید رضایی
7801	آرش ریاضی دان
7801	گلناز هنردوست
7902	مجید رضایی
8001	پروین صبا

تذکر: حالت های نرمال 3 و نرمال BCNF معمولاً با هم معادلند . تنها در صورتی که جدول بیش از یک کلید کاندیدا داشته باشد و میان کلید کاندیدا ، ویژگی مشترک وجود داشته باشد ، این دو حالت با هم معادل نخواهند بود . مثلاً در جدول Project ، دو کلید کاندیدا وجود دارد و #ST میان دو کلید کاندیدا مشترک است .

به همین دلیل حالت نرمال 3 و BCNF این جدول معادل نیستند.

مثال : در بانک اطلاعاتی زیر ابتدا کلید های کاندید را بیابید و سپس آنرا بطور کامل نرمال سازی نمایید :

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

$F = \{ AF \rightarrow BE, FC \rightarrow DE , F \rightarrow CD , D \rightarrow E , C \rightarrow A \}$

حل :

سمت راست وابستگی ها را به یک صفت تبدیل می کنیم .

1) $B \rightarrow AF$
2) $AF \rightarrow E$
3) $FC \rightarrow D$
4) $FC \rightarrow E$
5) $F \rightarrow C$
6) $F \rightarrow D$
7) $D \rightarrow E$
8) $C \rightarrow A$

از شماره 5 می توانیم برای ساده کردن سمت چپ شماره های 3 و 4 استفاده کنیم . در نتیجه : 3 ($F \rightarrow D$ و 4) $F \rightarrow E$

از شماره های 5 و 8 می توان نتیجه گرفت که $F \rightarrow A$

این نتیجه را روی شماره های 1 و 2 اعمال می کنیم :

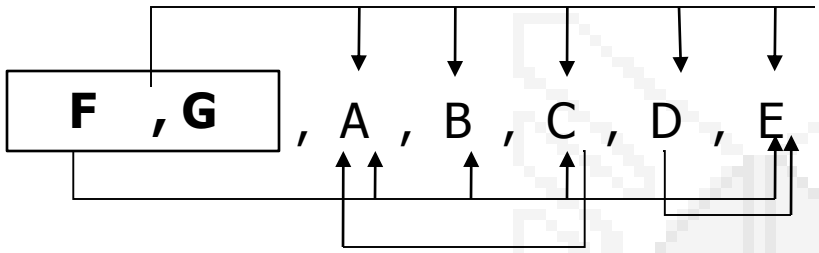
1) $F \longrightarrow B$, 2) $F \longrightarrow E$

پس $F = \{ F \longrightarrow A, F \longrightarrow B , F \longrightarrow C , F \longrightarrow D, F \longrightarrow E , D \longrightarrow E , C \longrightarrow A \}$

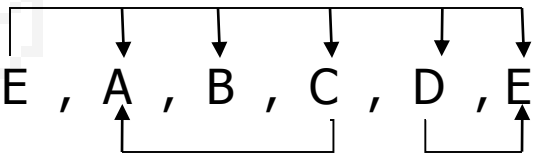
در نتیجه صفت F همه صفت های دیگر ، بجز G ، را می دهد ، پس(F , G) کلید کاندید است . این کلید کاندید منحصر به فرد است زیرا هیچ صفتی F و G را نمی دهد
(یعنی در هر کلید کاندید ، این دو صفت لازم هستند) .

نرمال سازی :

1NF : **F , G** , A , B , C , D , E



2NF : (F , G) E , A , B , C , D , E



3NF : (F , G)
(C , A)
(D , E)
(F , B , C , D)

نیازی به BCNF نیست زیرا بیش از یک کلید کاندید نداریم .

مثال : رابطه R={ X, Y, Z, S, T, U, W }را با **FD** های زیر در نظر بگیرید :

F = { S⟶ X, T⟶ Y, X ⟶Y, XY ⟶TUZ }

ب) بطور کامل نرمال سازی نمایید :

حل :

الف :

$S \longrightarrow X , X \longrightarrow Y \Longrightarrow S \longrightarrow XY$

$S \longrightarrow XY , XY \longrightarrow TUZ \Longrightarrow S \longrightarrow T , S \longrightarrow U , S \longrightarrow Z \Longrightarrow (S , W)$

$FOPT = \{S \longrightarrow X, S \longrightarrow Y, S \longrightarrow Z, S \longrightarrow T, S \longrightarrow U, T \longrightarrow Y, X \longrightarrow Y, X \longrightarrow T, X \longrightarrow U, X \longrightarrow Z \}$

2NF : ($\underline{S , W}$)
($\underline{S}$, X , Y , Z , T , U)

3NF : ($\underline{S, W}$)
(T, Y)
($\underline{X}$, Z , T , U)
($\underline{S}$, X)

نیازی به BCNFنیست زیرا بیش از یک کلید کاندید نداریم .

باید تاکید کرد که مجموعه وابستگی پوششی بهینه نقش مهمی در نرمال سازی دارد .

اگر بعضی از وابستگی های آنرا دور بداریم به نتیجه غلطی می رسیم .

به مثال زیر توجه کنید:

مثال : بانک اطلاعاتی زیر را نرمال سازی کنید .

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

$F = \{ (A, B) \longrightarrow R , A \longrightarrow G , B \longrightarrow EF , G \longrightarrow DF \}$

راه حل غلط : بدون یافتن $\mathbf{F^+}$

2NF (A , G)
(B , E , F)
(A , B , C , D , F , G)
3NF (A , G)
(B , E , F)
(A , B , C)
3NF (G , D , F)
(A , G)
(B , E , F)
(A , B , C)

راه حل صحیح با افزودن DF ⟶ A به وابستگی های تابعی

2NF (A , D , F , G)
(B , E , F)
(A , B , C)
3NF (G , D , F)
(A , G)
(B , E , F)
(A , B , C)

مشاهده می شود که در راه حل غلط ، آخرین جدول دارای افزونگی است .

شکل چهارم نرمال

تئوری فاگین : فرض کنید مجموعه R{A,B,C} یک متغیر رابطه ای باشد که در آن A,B,Cمجموعه هایي ازصفات هستند . آنگاه R برابر با الحاق دو تصویرش بر روی {A,B}و {A,C} است اگر و فقط اگر R وابستگی چند مقداری A → → B | C را برآورده کند.)

• متغیر رابطه ای R به شکل چهارم نرمال است اگر و فقط اگر , هرگاه زیر مجموعه های A و B از صفات R موجود باشند که غیر جزئی باشند , وابستگی چند مقداری A → → B برآورده شود.در این صورت تمام صفات R نیز به طور تابعی به A وابستگی دارند .
به عبارت دیگر , تنها وابستگی های غیر جزئی در R به شکل K → X هستند (یعنی وابستگی تابعی از یک فوق کلید به صفت دیگر X) .

وابستگی چند مقداری :

فرض کنید **R** یك متغیر رابطه ای و**A, B** و **C** زیر مجموعه ای از صفات **R** باشند . آنگاه وابستگی چند مقداری **B** بر روی **A** را به صورت زیر نمایش می دهیم :

A → → B

(این نماد به این صورت خوانده می شود **A''** به طور چند گانه **B** را تعیین می کند 0 یا **A''** با دو فلش به **B** مراجعه میکند .

این حالت برقرار است اگر و فقط اگر در هر مقدار ممکن معتبری برای **R** , مجموعه ای از مقادیر**B** که با جفت (مقدار **C** و مقدار **A**) مطابقت دارد, فقط به مقدار **A** بستگی داشته باشد و مستقل از مقدار **C** باشد.

برخی از جداول، در حالت نرمال BCNFهستند ولی هنوز ناهنجاری هایی دارند . جدول employee را در نظر بگیرید . این جدول شامل شماره کارمندی و مهارت ها و زبان هایی است که دهر کارمند بر آن مسلط است:

Employee

EMP≠≠	Skills	LANGUAGES
100	برنامه نویسی جاوا تجزیه و تحلیل شی گرا	انگلیسی
101	برنامه نویسی دلفی تجزیه و تحلیل شی گرا طراحی وب سایت	انگلیسی آلمانی

در این جدول ، ستون های Skills و LANGUAGES دارای گروه های اطلاع تکرار شونده هستند. نمودار وابستگی این جدول به شکل زیر است :

employee (EMP≠, Skills , LANGUAGES)

فرض کنید این جدول را بصورت زیر ، تبدیل به نرمال 1 کنیم :

2

Employee

EMP≠≠	SKILLS	LANGUAGES
100	برنامه نویسی جاوا	انگلیسی
100	تجزیه و تحلیل شی گرا	انگلیسی
101	برنامه نویسی دلفی	انگلیسی
101	تجزیه و تحلیل شی گرا	انگلیسی
101	طراحی وب سایت	انگلیسی
101	برنامه نویسی دلفی	آلمانی
101	تجزیه و تحلیل شی گرا	آلمانی
101	طراحی وب سایت	آلمانی

نمودار وابستگی جدول Employee 2،به شکل زیر خواهد بود :

Employee 2 (EMP≠, Skills , LANGUAGES)

جدول Employee 2، نرمال BCNF است (چون یک جدول تمام کلید است و این یک مسئله بدیهی است) ولی در درج، حذف و اصلاح، ناهنجاری دارد . مثلاً اگر کارمند 101 ، مهارت « برنامه نویسی جاوا » را نیز کسب کند ، به جای اضافه کردن یک تاپل ، باید دو تاپل جدید به جدول اضافه شود :

101	برنامه نویسی جاوا	انگلیسی
101	برنامه نویسی جاوا	آلمانی

و یا اگر بخواهیم زبان آلمانی را از اطلاعات کارمند 101 حذف کنیم، به جای حذف یک تاپل ، باید سه تاپل زیر را حذف کنیم :

101	برنامه نویسی دلفی	آلمانی
101	تجزیه و تحلیل شی گرا	آلمانی
101	طراحی وب سایت	آلمانی

برای رفع این معایب،جدول را به نرمال 4 تبدیل مکنیم.

یک جدول نرمال 4 است اگر :

1 – نرمال BCNF باشد .

2 – هیچ وابستگی چند مقداری در آن وجود نداشته باشد .

تعریف وابستگی چند مقداری : اگر در جدول T(A , B , C)، به ازاء هر مقدار برای ویژگی A، مجموعه ای از مقادیر برای ویژگی C وجود داشته باشد و این مجموعه، مستقل از مقادیر ویژگی Bباشد، ویژگی Cبه ویژگی A وابستگی چند مقداری دارد (A $\longrightarrow\!\!\!\!\longrightarrow$ B).

مثال :جدول (employee 2 (EMP≠, Skills , LANGUAGES را در نظر بگیرید :

(Emp#, Skills)= (101 , (برنامه نویسی دلفی ⟹ Languages = {انگلیسی و آلمانی }
(Emp#, Skills)= (101 , (تجزیه و تحلیل شی گرا ⟹ Languages = {انگلیسی و آلمانی }
(Emp#, Skills)= (101 , (طراحی وب سایت ⟹ Languages = {انگلیسی و آلمانی }

مشاهده می کنید که تنها Emp# ، مجموعه Languages را تعیین می کند و تغییر Skills هیچ نقشی در تغییر مجموعه Languagesندارد. دلیل این امر نیز آن است که مهارت های یک شخص و زبان هایی که شخص بر آن تسلط دارد هیچ ارتباطی با یکدیگر ندارند . پس Languages به Emp# وابستگی چند مقداری دارد (Emp# ↠ Languages) . از طرف دیگر :

(**Emp#, Languages**)= (101 , (انگلیسی ⟹ Skills={ طراحی وب سایت
{برنامه نویسی دلفی ، تجزیه و تحلیل شی گرا،

(Emp#, Languages)= (101, (انگلیسی ⟹ Skills={ طراحی وب سایت
{برنامه نویسی دلفی ، تجزیه و تحلیل شی گرا،

در واقع تنها Emp#مجموعه Skills را تعیین می کند و Languages هیچ نقشی در تغییر مجموعه Skills ندارد پس Skills نیز به Emp# وایستگی چند مقداری دارد (Emp# ↠ skills).
برای جلوگیری از ایجاد وابستگی چند مقداری، در همان مرحله اول و هنگام تبدیل جدول آنرمال به نرمال 1، به ازای هر گروه اطلاع تکرار شونده ، باید جدول مجزایی در نظر بگیریم :

Emp_ Skills

Emp#	Skills
100	برنامه نویسی جاوا
100	تجزیه و تحلیل شی گرا
101	برنامه نویسی دلفی
101	تجزیه و تحلیل شی گرا
101	طراحی وب سایت

Emp_ Languages

Emp#	Languages
100	انگلیسی
101	انگلیسی
101	آلمانی

emp_skills (Emp #, Skills)

emp_languages(Emp#, Languages)

هردو جدول emp_languages وemp_skills ، نرمال 4 هستند.

SPJ :

s#	p#	j#	Qty
S1	P1	J1	12000
S1	P2	J1	20000
S1	P3	J1	3000
S1	P4	J1	8000
S1	P1	J2	10000
S1	P1	J3	9000
S2	P1	J1	2000
S2	P1	J3	5000
S2	P3	J1	8000
S3	P1	J1	9000
S3	P2	J3	9000
S3	P1	J3	4000

مثال: جدول معروف SPJرا در نظر بگیرید:

هر تولید کننده ممکن است برای چندین پروژه، محصولاتی تولید کند . بنابراین ، به ازای هر#S، مجموعه ای از #J ها و مجموعه ای از #P ها وجود دارند اما :

1) ویژگی #j به #s ، وابستگی چند مقداری ندارد چون :

(s# , p#)= (S2 , P1) ⟹ J# = { J1 , J3 }

(s# , p#)= (S2 , P3) ⟹ J# = { J1 }

پس تنها #S، مجموعه #j را تعیین نمی کند بلکه با تغییر مقدار #p نیز احتمال تغییر مجموعه مقادیر #j وجود دارد.

2) ویژگی #p به #s وابستگی چند مقداری ندارد چون :

(s# , j#)= (S1 , j1) ⟹ J# = { p1 ,p2 , J3 }

(s# , j#)= (S1 , j1) ⟹ J# = { p1 }

پس تنها #S، مجموعه #p را تعیین نمی کند بلکه با تغییر مقدار #j نیز احتمال تغییر مجموعه مقادیر #p وجود دارد .

می توان نتیجه گرفت در جدول SPJ ، وابستگی چند مقداری وجود ندارد و این جدول ، نرمال 4 است.

شکل 5 نرمال:

یک متغیر رابطه ای R از درجه 5 نرمال است ,اگر وفقط اگر هر وابستگی الحاق غیر جزئی که برای R برقرار است,توسط کلیدهای R ارائه شود.به طوری که :

الف: وابستگی الحاق {A,B,...Z}* بر روی R جزئی است اگر وفقط اگر حداقل یکی از A,B,...Zمجموعه ای از صفات R باشند.

ب: وابستگی الحاق {A,B,...Z}* بر روی R توسط کلید کاندید R ارائه شوداگر وفقط اگر هر یک از A,B,...Z فوق کلید R باشد.

نکته: R{A,B,C} وابستگی الحاق {AB,AC}* را برآورده می کند اگر و فقط اگر وابستگی چند مقداری A →→ B | C را برآورده سازد.

چون این تئوری می تواند به عنوان تعریفی از وابستگی چند مقداری بیان شود , نتیجه می شود که وابستگی چند مقداری فقط حالت خاصی از وابستگی الحاق است, یا وابستگی های الحاق حالت کلی از وابستگی های چند مقداری است . به طور رسمی , داریم : A →→ B | C = *{AB,AC}

یک جدول در حالت نرمال 5 است اگر :
1- نرمال 4 باشد .
2 – نتوان آنرا به جداول های کوچک تر تقسیم کرد بطوری که حداقل یکی از جداول، شامل هیچ یک از کلید های کاندیدای جدول اولیه نباشد .
مثال : طبق قوانین یک دانشگاه :
-هر دانشجو می تواند هر درس را چند بار ولی تنها یک بار با هر استاد بگیرد .
- هر استاد می تواندرس های مختلفی را تدریس کند و هر درس توسط اساتید مختلف تدریس می شود .

جدول enroll را در نظر بگیرید :

enroll:

St#	Crs#	Prof#
S1	C1	P1
S1	C2	P2
S2	C2	P1

تنها کلید کاندیدای این جدول ، St#+Crs#+Prop# و نمودار وابستگی این جدول به شکل زیر است :
Enroll (St#+Crs#+ Prop#)
این جدول، نرمال BCNF است و چون وابسگتی چند مقداری ندارد، نرمان 4 نیز هست. برای تشخیص نرمال 5 بودن این جدول، باید بررسی کنیم آیا این جدول قابل تجزیه به چند جدول کوچک تر که پیوند آنها، جدول اولیه را نتیجه دهد، هست یا خیر ؟
این جدول را به دو جدول enroll1 , enroll2 می شکنیم :

enroll1

St#	Crs#
S1	C1
S1	C2
S2	C2

enroll2

St#	Prof#
S1	p1
S1	p2
S2	P1

Enroll1 join enroll2

St#	Crs#	Prof#
S1	C1	P1
S1	C1	P2
S1	C2	P1
S1	C2	P2
S2	C2	P1

نتیجه پیوند این دو جدول، بصورت زیر خواهد بود :

همانطور که مشاهده می کنید، پیوند دو جدول، دو تاپل اضافی تولید کرد که در جدول enroll اولیه وجود نداشت. پس این تجزیه بی فایده است .

به همین ترتیب، اگر این جدول را به دو جدول enroll3 و enroll4 بشکنیم، داریم :

Enroll3

St#	Crs#
S1	C1
S1	C2
S2	C2

Enroll4

St#	Prof#
S1	P1
S1	P1
S2	P2

enroll3 join enroll4

St#	Crs#	Prof#
S1	C1	P1
S1	C2	P1
S1	C2	P2
S2	C2	P1
S2	C2	P2

نتیجه دو پیوند بصورت زیر خواهد بود :

همانطور که مشاهده می کنید، پیوند دو جدول، دو تاپل اضافی تولید کرد که در جدول enroll اولیه وجود نداشت. پس این تجزیه نیز بی فایده است .

حال روی جدول enroll، سه پرتو (از این عملگر که در جبر رابطه ای تعریف شده و بصورت Π نشان داده می شود ، برای گزینش عمودی در یک جدول استفاده می ود یعنی با استفاده از این عملگر می توان ستون های مورد نظر از یک جدول را انتخاب کرد) می گیریم :

SC

St#	Crs#
S1	C1
S1	C2
S2	C2

CP

Crs#	Prof#
C1	P1
C2	P2
C2	P1

SP

St#	Prof#
S1	P1
S1	P2
S2	P1

ویژگی مشترک بین دو جدول SC و CP، CRS# است. نتیجه پیوند دو جدول ، جدول زیر خواهدبود :

CS join CP

St#	Crs#	Prof#
S1	C1	P1
S1	C2	P2
S1	C2	P1
S2	C2	P2
S2	C2	P1

ویژگی مشترک بین دو جدول (SC join CP)و SP، Prof# + St# است. نتیجه پیوند این دو جدول، جدول مقابل خواهد بود :

(Sc join Cp) join Sp

St#	Crs#	Prof#
S1	C1	P1
S1	C2	P2
S1	C2	P1
S2	C2	P1

همانگونه که مشاهده می کنید، نتیجه (Sc join Cp) join Sp ، با جدول enroll برابر نیست و شامل تاپل اضافی (S1 , C2 , P1) است که در جدول enroll وجود ندارد . این مشکل از آنجا ناشی می شودکه :

1 – دانشجوی S1 ، درس C2 را گرفته است .

2 – دانشجوی S1 با استاد P1 درس گرفته است .

3 – درس C2 جز درس هایی است که استاد P1 تدریس می کند.

چنین نتیجه شده است که دانشجوی S1، درس C2 را با استاد P1 گرفته است در حالی که واقعیت ندارد. جدول enroll، سه ستون دارد پس نمی توان آنرا بصورت ستونی به بیش از 3 جدول تجزیه کرد و چون پیوند هیچ یک از تجزیه های دوتایی و سه تایی، ما را به جدول اصلی نرساندند، می توان نتیجه گرفت جدول enroll قابل تجزیه به جداول کوچکتر که پیوند آنها جدول اولیه را نتیجه می دهد، نیست پس جدول enroll نرمال 5 است.

مثال : مشخصات كلي بانك اطلاعاتي ثبت نام يك دانشگاه در زير آمده است . اين بانك را طراحي و نرمال كنيد:

1- كد درسها غير تكراري و نام و تعداد واحد دروس مشخص است.

2- هر دانشكده درس هاي مشخصي را ارائه ميكند و مسئول دانشكده و شهر آن مشخص است . دروس همنام در دانشكده هاي مختلف واحد متفاوتي دارند.

3- هر درس در چند گروه در زمان و مكان مشخص توسط يك استاد ارائه مي شودو شماره گروه در درسها و نيمسال هاي مختلف تكرار ميگردد.

4- دانشجويان در گرو ه هاي مختلف ثبت نام ميكنند و نمره مي گيرند.

جواب : ابتدا هر جداول مشخص و وابستگي تابعي آن تعيين مي گردد

Crs = { c# , cname , unit , clg# }

1=f{c# $\longrightarrow$ cname , unit , clg# , (cname , unit) $\longrightarrow$ clg#

Clg = (clg# , Clg name , city , Pname)

2= {clg# $\longrightarrow$ Clg name , city , Pname }f

Sec = (s# , sec# , s# , pname , term , score , time , place)

3= { (s# , c# , term) $\longrightarrow$ sec# , score , (sec# , c# , term) $\longrightarrow$ pname , time , placce }f

2 – 3 تمرین حل شده :

فرض کنید سیستم پایگاه داده ها در یک دانشگاه شامل جداول زیر باشد :

Field (field# , field Name)

در این جدول اطلاعات مربوط به رشته های تحصیلی ذخیره می شو د :

field# : کد رشته تحصیلی یک کد منحصر بفرد در نظر گرفته شده است)

Field Name : نام رشته تحصیلی است

Type (type# , type Name , fee)

در این جدول اطلاعات مربوط به نوع درس ذخیره می شود :

type# : کد نوع درس (به هر نوع درس یک کد منحصر بفرد داده می شود) .

type Name : نوع درس

Fee : قیمت هر واحد

Student (st# , sname , start year , field#)

در این جدول اطلاعات مربوط به دانشجویان نگهداری می شود :

st# : شماره دانشجویی

Sname : نام دانشجو

start year : سال ورود به دانشگاه

field# : کد رشته تحصیلی دانشجو

Course (crs# , cname , unit , type#)

در این جدول اطلاعات دروس ذخیره می شود :

crs# : شماره درس

cname : نام درس

unit : تعداد واحد درس

type# : کد نوع درس (نظری ، عملی ، و ...)

CF (crs# , field# , Kind)

در این جدول مشخص می شود هر درس مربوط به کدام رشته های تحصیلی است:

crs# : شماره درس

field# : کد رشته تحصیلی

Kind : این ویژگی مشخص می کند درس مورد نظر برای رشته مورد نظر چه حالتی دارد (' **P** ' برای پیش نیاز ، ' **A** ' برای پایه ، ' **T** ' برای تخصصی ، ' **O** ' برای عمومی و ' **E** ' برای اختیاری)

Grades (st# , crs# , term , grade)

در این جدول اطلاعات مربوط به نمرات دانشجویان ذخیره می شود :

st# : شماره دانشجویی

crs# : شماره درس

Term : نیمسال اخذ درس توسط دانشجو (مثلاً 831 به معنی ترم اول سال 83 است)

Grade : نمره اخذ شده

Per (crs# , per#)

در این درس پیش نیازهای هر درس مشخص می شوند :

crs# : شماره درس

per# : شماره درس پیش نیاز

Prof (Prof# , pname , degree)

در این جدول اطلاعات مربوط به اساتید ذخیره می شود :

Prof# : شماره استاد (به هر استاد یک شماره منحصربفرد داده شده است)

Pname : نام استاد

Degree : آخرین مدرک تحصیلی استاد (' D 'برای دکترا، ' F ' برای فوق لیسانس، ' L 'برای لیسانس)

PC (Prof# , crs# , term)

در این جدول مشخص می شود هر استاد در هر نیمسال چه دروسی را تدریس کرده است :

Prof# : شماره استاد

crs# : شماره درس

Term : نیمسال تحصیلی

Tuition (field#, start year , const Tuition)

در این جدول بر اساس سال ورود به دانشگاه، شهریه ثابت هر رشته تحصلی مشخص می شود :

field# : کد رشته تحصیلی

start year : سال ورود به دانشگاه

const Tuition : شهریه ثابت

برای درک بهتر مساله به نمونه هایی از اطلاعات ذخیره شده در پایگاه داده توجه کنید :

Field

field#	field Name
1	مهندسی کامپیوتر
2	مهندسی الکترونیک
3	ریاضی محض

Type

type#	type Name	fee
1	نظری	5000
2	عملی	20000
3	آزمایشگاه	30000

Tuition

field#	start year	const Tuition
1	80	75000
1	81	80000
1	82	90000
2	81	80000
2	82	90000
3	82	75000

Course

crs#	cname	unit	type#
1100	ادبیات	2	1
1105	تربیت بدنی	1	2
1400	برنامه سازی1	3	1
1402	برنامه سازی 2	3	1
1403	ذخیره و بازیابی	3	1
1407	پایگاه داده	3	1
1500	ریاضی 1	3	1
1600	زبان پیش نیاز	2	1

Student

st#	sname	start year	field#
8001	آرش رادمهر	80	1
8002	عسل شاملو	80	3
8003	سیاوش آزاد	80	1
8101	ساغر راد	81	1
8102	علی نیکی	81	1
8111	فرامرز نیکی	81	1
8112	علی رضایی	81	2

Prof

Prof#	pname	degree
101	فرانک شایسته	L
102	علی پیامی	F
106	آزاده نیکوکار	F
107	سیامک فرزانه	D
107	علی نادر نژاد	F

Grades

st#	crs#	term	grade
8001	1400	811	9
8001	1400	812	13
8001	1401	811	13
8101	1400	821	19
8101	1500	821	14
8111	1400	811	12
8111	1401	811	20

PC

Prof#	crs#	term
101	1400	801
101	1400	802
101	1401	801
102	1400	801
108	1500	811

Per

crs#	per#
1400	1500
1402	1400
1407	1402
1407	1403

CF

crs#	field#	Kind
1100	1	O
1100	2	O
1100	3	O
1105	1	O
1105	2	O
1105	3	O
1400	1	T
1400	2	E
1400	3	E
1402	1	T
1403	1	T
1407	1	T
1500	1	P
1500	2	P
1500	3	P

الف) صحت روابط زیر را در جدول field بررسی کرده، نمودار وابستگی این جدول را رسم کنید :

field# ⟶ field Name

این رابطه صحیح است چون با داشتن یک کد رشته، تنها به یک نام رشته می رسیم .

field# —FFD→ field Name

این رابطه صحیح است چون با داشتن یک کد رشته، تنها به یک نام رشته می رسیم و #field قابل خلاصه شدن نیست .

نمودار وابستگی :

Field (field# , field Name)

تذکر: در نمودار های وابستگی، برای مشخص کردن کلید اصلی از خط زیر استفاده می شود .

ب) صحت روابط زیر را در جدول type بررسی کرد، نمودار وابستگی این جدول را رسم کنید :

type# —FFD→ type Name , fee

این رابطه صحیح است چون با داشتن یک نوع کد درس، تنها به یک نام نوع درس و تنها به یک قیمت واحد می رسیم.

type# + type Name —FFD→ fee

این رابطه صحیح نیست چون type ⟶ fee صحت دارد. پس طرف چپ قابل خلاصه شدن است .

type# —FFD→ type Name , fee

این رابطه صحیح است چون type# ⟶ type Name , fee و از طرف دیگر، #type قابل خلاصه شدن نیست.

نمودار وابستگی :

Type (type# , type Name , fee)

ج) صحت روابط زیر را در جدول Tuition بررسی کرده ، نمودار وابستگی این جدول را رسم کنید.

Field ⟶ const Tuition

این رابطه صحیح نیست چون با داشتن یک کد رشته، لزوماً به یک شهریه ثابت واحد نمی رسیم، چون نرخ شهریه برای ورودی های هر سال در رشته های مختلف، متفاوت است. مثلاً برای کد رشته کامپیوتر ممکن است چندین نرخ شهریه وجود داشته باشد.

start year ⟶ const Tuition

این رابطه صحیح نیست چون با داشتن یک سال ورود، لزوماً به یک شهریه ثابت واحد نمی رسیم، چون نرخ شهریه برای ورودی های هر سال در رشته های مختلف، متفاوت است. مثلاً برای ورودی های سال 83 ممکن است چندین نرخ شهریه وجود داشته باشد .

field#+ start yea ⟶ const Tuition

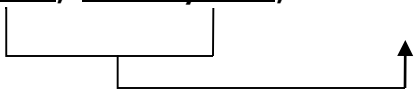
این رابطه صحیح است چون با داشتن یک کد رشته و سال ورود، لزوماً به یک شهریه ثابت می رسیم .

field#+ start yea $\xrightarrow{FFD}$ const Tuition

این رابطه صحیح است چون const Tuition ⟶ field#+ start yea صحیح است و از طرف دیگر، field#+ start yea، قابل خلاصه شدن نیست.

نمودار وابستگی :

Tuition (field#, start year , const Tuition)



د) صحت روابط زیر را در جدول st بررسی کرده و نمودار وابستگی این جدول را رسم کنید :

sname + start year ⟶ field#

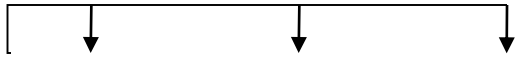
این رابطه صحیح است چون با داشتن نام یک دانشجو و سال ورود، لزوماً به کد رشته واحد نمی رسیم. مثلاً ممکن است دو دانشجوی ورودی 80 با نام مسعود اسماعیلی در دانشگاه وجود داشته باشند که یکی در رشته عمران و دیگری در رشته کامپیوتر تحصیل می کند.

st# ⟶ sname , start year , field#

این رابطه صحیح است چون با داشتن یک شماره دانشجویی، تنها ؛ به یک نام ، یک سال ورود و به یک کد رشته می رسیم .

st# $\xrightarrow{FFD}$ sname , start year , field#

این رابطه صحیح است چون st# ⟶ sname ، start year ، field# صحت دارد و از طرف دیگر ،st# قابل خلاصه شدن نیست.



Student (st# , sname , start year , field#)

نمودار وابستگی :

ر) صحت روابط زیر را در جدول Course بررسی کرده، نمودار وابستگی این جدول را رسم کنید:

crs# ⟶ cname , unit , type#

این رابطه صحیح است چون با داشتن یک شماره درس،تنهابه یک نام درس، تنها به یک تعداد واحد و تنها به یک کد نوع درس می رسیم.

crs# +cname ⟶ unit , type#

این رابطه صحیح است چون با داشتن یک شماره درس و نام درس، تنها به یک تعداد واحد و تنها به یک کد نوع درس می رسیم.

crs# $\xrightarrow{FFD}$ cname , unit , type#

این رابطه صحیح است چون crs# ⟶ cname , unit , type# برقرار است و از طرف دیگر، crs# قابل خلاصه شدن نیست.

crs# +cname $\xrightarrow{FFD}$ unit , type#

این رابطه صحیح نیست چون unit , type# ⟶ crs# صحیح است. پس سمت چپ قابل خلاصه شدن است.

نمودار وابستگی :

ز) صحت روابط زیر را در جدول CF بررسی کرده، نمودار وابستگی این جدول را رسم کنید:

crs# ⟶ #field

این رابطه صحیح نیست چون با داشتن یک شماره درس، لزوماً به یک کد رشته نمی رسیم. مثلاً ممکن است درس ریاضی1 ، هم مربوط به رشته ریاضی و هم مربوط به کامپیوتر باشد.

crs# ⟶ Kind

این رابطه صحیح نیست چون با داشتن یک شماره درس، لزوماً تنهابه یک حالت نمی رسیم. مثلاً ممکن است درس برنامه سازی2 ، برای رشته ریاضی،اختیاری و برای رشته کامپیوتر ،تخصصی باشد.

field# ⟶ Kind

این رابطه صحیح نیست چون با داشتن یک کد رشته، لزوماً تنها به یک حالت درس نمی رسیم چون مسلماً درس های زیادی در یک رشته وجود دارند که هر کدام از آنها، حالت خاص خود را دارد. مثلاً در رشته کامپیوتر هم دروس پایه و هم دروس تخصصی و هم دروس اختیاری وجود دارند.

field# ⟶ FFD ⟶ Kind

این رابطه صحیح نیست چون رابطه Kind ⟶ #field صحیح نیست.

crs#+ field# ⟶ Kind

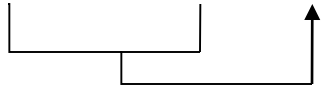
این رابطه صحیح است چون هر درس برای هر رشته، تنها یک حالت می تواند داشته باشد مثلاً درس برنامه سازی 2 ، برای رشته کامپیوتر تنها تخصصی است .

crs#+ field# ⟶ FFD ⟶ Kind

این رابطه صحیح است چون اولاً Kind ⟶ #crs#+ field برقرار است و ثانیاً #crs#+ field قابل خلاصه شدن نیست.

نمودار وابستگی :

CF (crs# , field# , Kind)



س) صحت روابط زیر را در جدول Grades بررسی کرده، نمودار وابستگی این جدول را رسم کنید :

st# ⟶ grade

این رابطه صحیح نیست چون یک دانشجو ممکن است نمرات زیادی داشته باشد، پس داشتن یک شماره دانشجویی لزوماً ما را به یک نمره واحد نمی رساند.

st# + crs# ⟶ grade

این رابطه صحیح نیست چون ممکن است یک دانشجودر یک درس چندین نمره گرفته باشد. مثلاً ممکن است دانشجوی 7801 بار اول درس 1400 نمره 8 و بار دوم نمره 11 گرفته باشد. پس با داشتن یک شماره دانشجویی و یک شماره درس، لزوماً به یک نمره واحد نمی رسیم.

st# + crs# ⟶ term

این رابطه درست نیست چون ممکن است یک دانشجو یک درس را در چند ترم گرفته باشد، ممکن است دانشجوی 7801 درس 1400 را هم در ترم 821 و هم در ترم 822 گرفته باشد. پس با داشتن یک شماره دانشجویی و یک شماره درس، لزوماً به یک ترم واحد نمی رسیم .

st# + crs# + term ⟶ grade

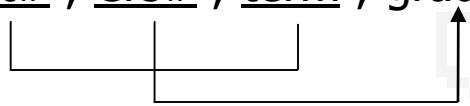
این رابطه صحیح است چون با داشتن یک شماره دانشجویی، یک شماره درس و یک ترم، تنها به یک نمره می رسیم، چون در هر درس در یک ترم برای هر دانشجو تنها یک نمره ثبت می شود.

st# + Crs# +term $\xrightarrow{FFD}$ grade

این رابطه صحیح است چون اولاً st# + crs# +term ⟶ grade برقرار است و ثانیاً طرف چپ رابطه قابل خلاصه شدن نیست.

نمودار وابستگی :

Grades (st# , Crs# , term , grade)



ش) صحت روابط زیر را در جدول **Per** بررسی کرده، نمودار وابستگی این جدول را رسم کنید :

crs# ⟶ PerCrs#

این رابطه صحیح نیست چون با داشتن یک شماره درس، لزوماً به یک شماره درس پیش نیاز نمی رسیم. مثلاً ممکن است درس پایگاه داده بیش از یک پیش نیاز داشته باشد.

Per crs# ⟶ crs#

این رابطه صحیح نیست چون با داشتن یک شماره درس پیش نیاز، لزوماً به یک شماره درس واحد نمی رسیم. مثلاً ممکن است درس برنامه سازی1 ، پیش نیاز چند درس مختلف باشد.

crs# + Per crs# ⟶ crs# , Per crs#

صحت این رابطه بدیهی است .

crs# + Per crs# $\xrightarrow{FFD}$ crs# , Per crs#

این رابطه صحیح است چون اولاً crs# + Per crs# $\longrightarrow$ crs# , Per crs# برقرار است و ثانیاً طرف چپ رابطه قابل خلاصه شدن نیست.

نمودار وابستگی :

Per (crs# , Per crs#)

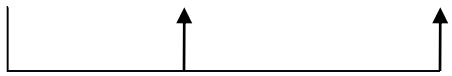
و) صحت روابط زیر را در جدول **Prof** بررسی کرده، نمودار وابستگی این جدول را رسم کنید :

Prof# $\longrightarrow$ Pname , degree

این رابطه صحیح است چون اولاً Prof# $\longrightarrow$ Pname , degree برقرار است و ثانیاً طرف چپ رابطه قابل خلاصه شدن نیست.

نمودار وابستگی :

Prof (Prof# , Pname , degree)



crs# + Per crs# $\xrightarrow{FFD}$ crs# , Per crs#

این رابطه صحیح است چون اولاً crs# + Per crs# $\longrightarrow$ crs# , Per crs# برقرار است و ثانیاً طرف چپ رابطه قابل خلاصه شدن نیست.

نمودار وابستگی :

Per (crs# , Per crs#)

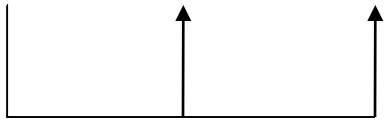
و) صحت روابط زیر را در جدول Prof بررسی کرده، نمودار وابستگی این جدول را رسم کنید :

Prof# $\longrightarrow$ Pname , degree

این رابطه صحیح است چون اولاً Prof# $\longrightarrow$ Pname , degree برقرار است و ثانیاً طرف چپ رابطه قابل خلاصه شدن نیست.

نمودار وابستگی :

Prof (Prof# , Pname , degree)



ی) کلید های خارجی کلیه جداول را مشخص کنید:

نام جدول	کلید خارجی
Field	ندارد
Type	ندارد
Student	field# نسبت به جدول field
Course	type# نسبت به جدول type
CF	crs# نسبت به جدول Course و field# نسبت به field
Grades	st# نسبت به جدول Student و crs# نسبت به جدول Course
Per	crs# نسبت به جدول Course و per#نسبت به جدول Course (تشابه اسمی کلید اصلی و خارجی اهمیتی ندارد)
Prof	ندارد
PC	Prof# نسبت به جدول Prof و crs# نسبت به جدول
Tuition	field# نسبت به field

اکنون به مثال دیگری توجه کنید :

مثال : فرض کنید قوانین زیر در یک دانشگاه حاکم باشد:

- هر استاد می تواند درس های مختلفی را تدریس کند و هر در توسط اساتید مختلف تدریس می شود .

- هر استاد برای هر درس ، می تواند چند کتاب معرفی کند .

- اگر استادی یک درس را تدریس کند، علاوه بر کتابهای که خود معرفی می کند، مجبور است کلیه کتابهایی که سایر مدرسین آن درس برای دانشجویان معرفی کرده اند را نیز معرفی کند . !!!

جدول PCB را در نظر بگیرید :

PCB

Prof#	Crs#	BOOK#
P1	C1	B1
P2	C2	B2
P3	C7	B3

تنها کلید کاندیدای این جدول، **Prof# + Crs# + BOOK#** است و نمودار وابستگی این جدول، به شکل مقابل است :

PCB (Prof# + Crs# + BOOK#)

این جدول، نرمال 4 است ولی یک ناهنجاری بسیار عجیب دارد .مثلاً اگر بخواهیم تاپل (P2 , C2 , B5) را در جدول درج کنیم، برای رعایت محدودیت اعمال شده از طرف دانشگاه، باید تاپل (P2 , C2 , B2) و تاپل (P1 , C2 , B5) را نیز درج کنیم !!!

حال باید تشخیص داد که با توجه به محدودیت اعمال شده از طرف دانشگاه، این جدول قابل تجزیه به جداول کوچکتر است یا خیر ؟ تجزیه به دو جدول نتیجه ای ندارد پی تجزیه سه تایی را مورد بررسی قرار می دهیم :

PC

Prof#	Crs#
P1	C1
P1	C2
P3	C7

CB

Crs#	Book#
C1	B1
C2	B2
C7	B8

PB

Prof#	Book#
P1	B1
P1	B2
P3	B8

پیوند سه جدول فوق ، ما را به جدول اولیه می رساند. به عبارت دیگر:

Pc JOIN CB JOIN PB=PCB

از طرف دیگر هیچ یک از جداول PC یا CB و یا PB شامل کلید کاندیدای جدول اصلی نیستند پس جدول PCB ، نرمال5 نیست و باید به سه جدول PC و CB و PB که هر سه نرمال5 هستند شکسته شوند.

با توجه به مثال صفحه بعد داریم:

Crs#

مثال: جدول Course **را در نظر بگیرید:**

Crs#	Cname	Unit
1100	ادبیات	2
1105	تربیت بدنی	1
1400	برنامه سازی1	3
1402	برنامه سازی2	3

کلیدهای کاندیدای این جدول عبارتند از:
کلیدهای کاندیدای1: Crs#
کلیدهای کاندیدای2: Cname
اگر Crs# را به عنوان کلید اصلی معرفی کنیم نمودار وابستگی این جدول به شکل زیر خواهد بود:
Course (Crs# , Cname , Unit)

این جدول را می توان:
1- به دو جدول { (Cname , Unit) , (Crs# , Cname) } تجزیه کرد.در این صورت نتیجه پیوند دو جدول با جدول Course اولیه یکسان خواهد بود ولی انجام این تقسیم کاملاً بی مورد است چون در هر دو جدول حداقل یکی از کلیدهای کاندیدا حضور دارند.

2- به دو جدول { (Crs# , Unit) , (Crs# , Cname) } تجزیه کرد.در این صورت نتیجه پیوند دو جدول با جدول Course اولیه یکسان خواهد بود ولی انجام این تقسیم کاملاً بی مورد است چون در هر دو جداول حداقل یکی از کلیدهای کاندیدا حضور دارند.
پس می توان نتیجه گرفت که جدول Course نرمال 5 است و لازم نیست به جداول کوچکتر تجزیه شود.

تذکر: نرمال5 در عمل کاربردی ندارد و اکثر جداول تنها تا مرحله 3NF نرمال می شوند چون در حالت نرمال3 هیچ مشکلی ندارند.

سیستم کتابخانه یک دانشگاه را در نظر بگیرید.در این کتابخانه به هر کتاب یک شماره منحصر به فرد داده شده است مثلاً اگر در کتابخانه ، سه نسخه از یک کتاب وجود داشته باشد،هر یک از آنها یه شماره مجزا خواهد داشت.در هر امانت دانشجو می تواند حداکثر سه کتاب را به امانت بگیرد (مثلاً در امانت شماره 1001 ممکن است دو کتاب 1403 و 1809 به امانت گرفته شوند) و باید همه کتاب هایی را که باهم از کتابخانه به امانت گرفته است ، با هم به کتابخانه برگرداند.با توجه به این قوانین ، نمودار وابستگی جدول loan را رسم کرده ، آن را به جداول نرمال 3 تبدیل کنید.

Loan (loan# , book# , bname , author , st# , sname , field , loanDate , returnDate)

Loan#: **شماره امانت**

Book#: شماره کتاب

Bname: نام کتاب

Author: نام نویسنده

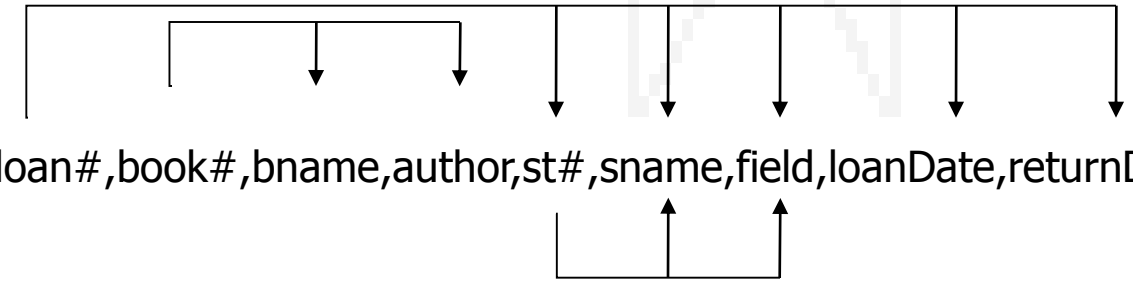
St#: شماره دانشجویی که کتاب را به امانت گرفته

Sname: نام دانشجو

Field: رشته تحصیلی دانشجو(هر دانشجو تنها در رشته تحصیل می کند)

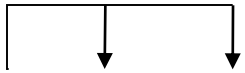
loanDate: تاریخ امانت گرفتن کتاب

ReturnDate: تاریخ برگشت کتاب به کتابخانه توسط دانشجو



تبدیل به نرمال2:

این جدول،نرمال2 نیست چون bname و author به book# که جزیی از کلید اصلی است،وابستگی تابعی دارند.همچنین، st# و sname و field و loanDate و returnDate به loan# که جزیی از کلید اصلی است وابستگی تابعی دارند.پس این جدول را به جداول نرمال 2 می شکنیم:



L (loan# , s# , sname , field , loanDate , returnDate)

Book (book# , bname , author)

LB (loan# , book#)

نکته: توجه کنید اگر چه هیچ ویژگی غیر کلیدی که به book# و loan# وابستگی داشته باشد و در سطرهای قبل نیامده باشد،وجود ندارد،ولی این ردیف را به جدول LB تبدیل کرده ایم.دلیل این امر آن است که تنها در این جدول مشخص می شود در هر امانت ، چه کتاب هایی امانت کرفته شده اند.پس این جدول اطلاعات مفیدی در اختیار می گذارد که در جداول دیگر موجود نیست.

تبدیل به نرمال3:

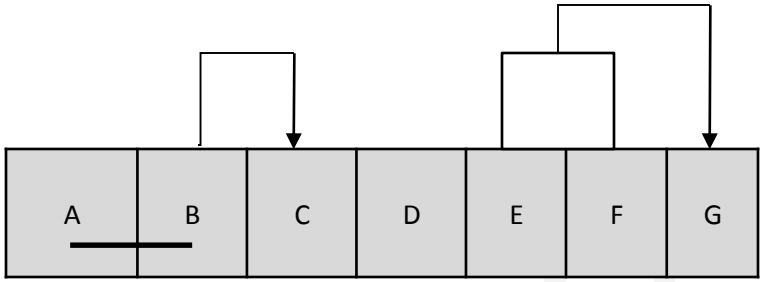
جداول B و LB نرمال 3 هستند چون در آنها هیچ ویژگی غیر کلیدی به ویژگی غیر کلیدی دیگر وابستگی تابعی ندارد ولی جدول L نرمال3 نیست چون در آن sname و field به s# که غیر کلیدی است وابستگی دارد.پس جدول L را به جداول زیر که هر یک نرمال3 هستند می شکنیم:

L1 (s# , sname , field)

L2 (loan# , loanDate , returnDate)

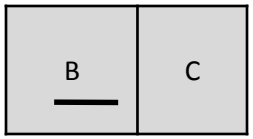
پس جدول loan به چهار جدول LB و Book و L1 و L2 که همگی نرمال3 هستند،شکسته شد.

2-7-4)تمرین2
جدول زیر را با ذکر کلیه مراحل،به جداول نرمال3 تبدیل کنید.

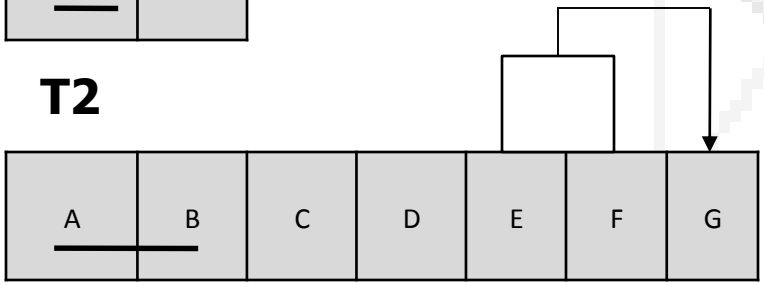


تبدیل به نرمال 2: از آنجا که ویژگی C به ویژگی B که بخشی از کلید اصلی است،وابستگی تابعی دارد،در این جدول ، وابستگی جزیی وجود دارد.پس نرمال 2 نیست.برای تبدیل به نرمال 2:

T1



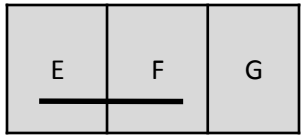
T2



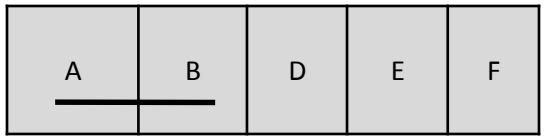
تبدیل به نرمال 3: جدول T1 ، نرمال 3 است ولی در جدول T2 ، ویژگی G که یک ویژگی غیر کلیدی است،به ویژگی های E و F که غیر کلیدی می باشند وابستگی تابعی دارد،پس جدول T2 نرمال 3 نیست.

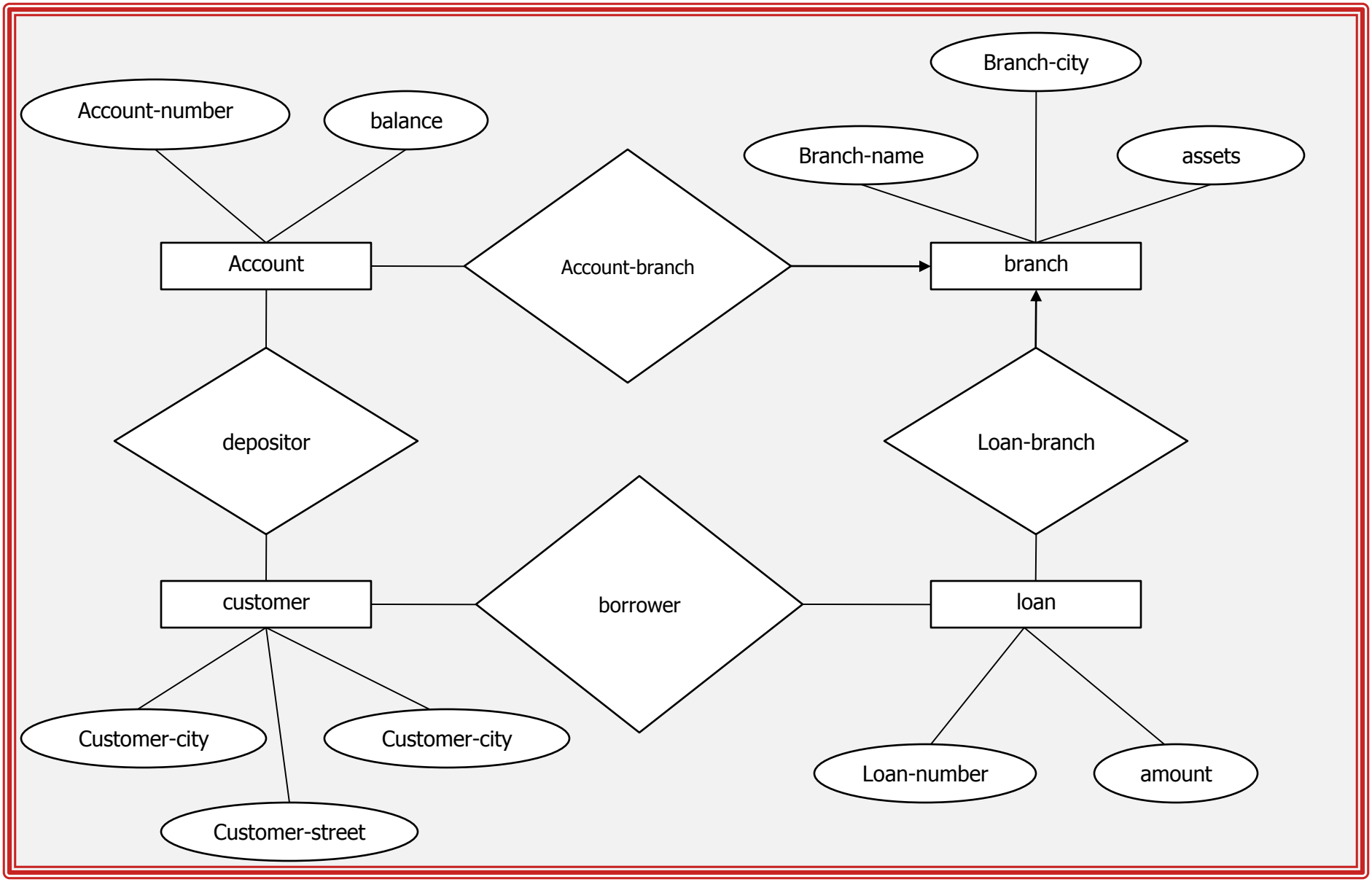
برای تبدیل به نرمال 3 ، این جدول را به دو جدول T21 و T22 می شکنیم:

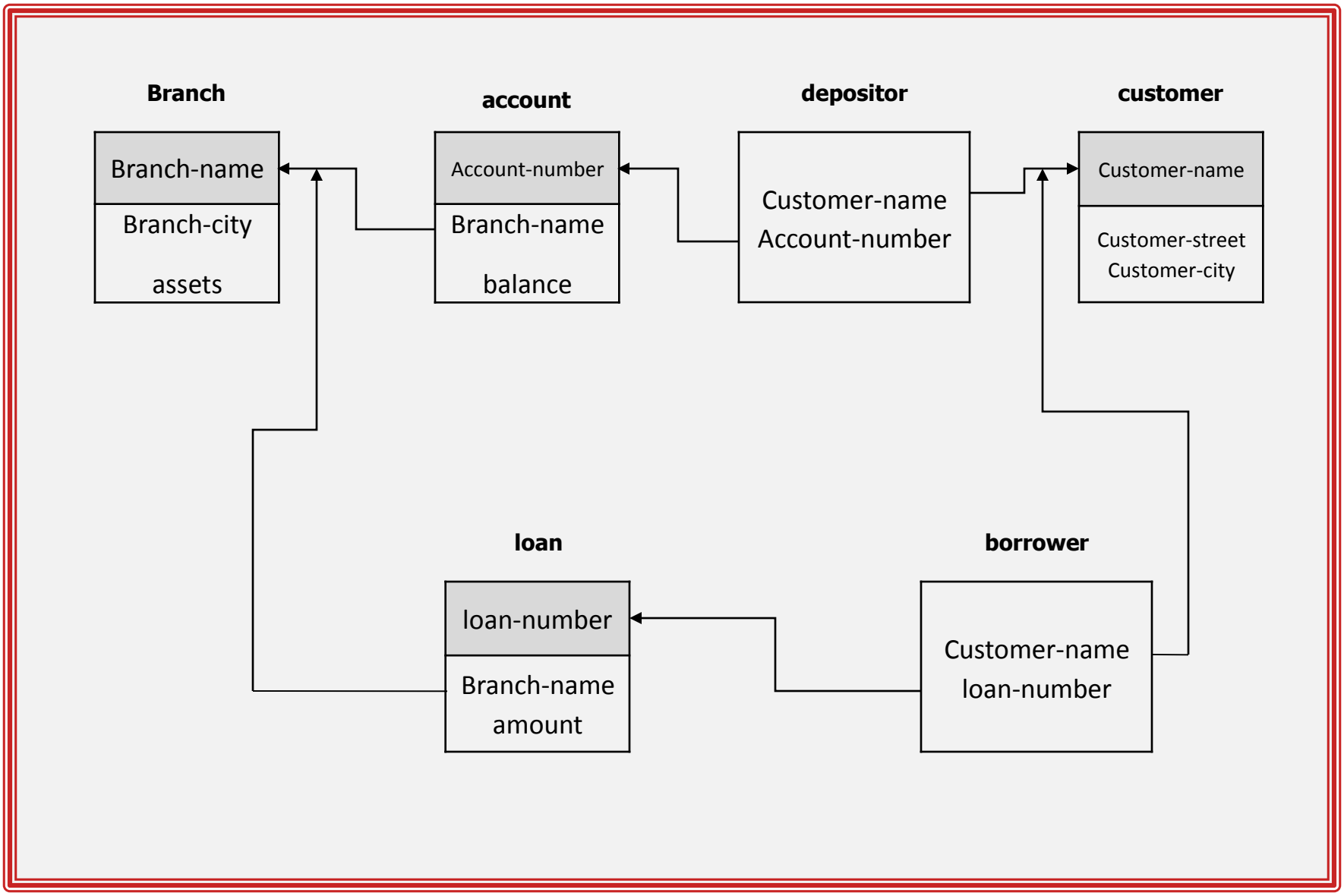
T21



T22




Branch-city
Account-number
balance
Branch-name
assets
Account
Account-branch
branch
depositor
Loan-branch
customer
borrower
loan
Customer-city
Customer-city
Loan-number
amount
Customer-street


Branch
Branch-name
Branch-city
assets
account
Account-number
Branch-name
balance
depositor
Customer-name
Account-number
customer
Customer-name
Customer-street
Customer-city
loan
loan-number
Branch-name
amount
borrower
Customer-name
loan-number

query ها :

١- تمامي وامهاي بيش از ١٠٠٠٠٠ ريال را بدهيد .

$$б_{amount > 100000}(loan)$$

٢- شماره وامهايي را كه بيش از ١٠٠٠٠٠ ريال مي باشند بدهيد .

$$П_{loan\_number}(б_{amount > 100000}(loan))$$

٣- اسامي تمامي مشتريانى كه وام گرفته و شماره حساب دارند را بدهيد .

$$П_{customer\_name}(borrower) \cap П_{customer\_name}(depositor)$$

٤- اسامي تمام مشتريانى را كه در شعبه مركزي وام گرفته اند بدهيد .

$$П_{customer\_name}(б_{branch\_name = 'مرکزی'}(б_{borrower.loan\_number = loan.loan\_number}(borrower \times loan)))$$

۵ -اسامي تمام مشتريانی که در شعبه نادري وام گرفته اند را بدهيد .

Q1: Пcustomer_name (бbranch_name = ‘نادری’)(бborrower.loan_number = loan .loan_number (borrower × loan)))

Q2: Пcustomer_name (бloan .loan_number = borrower.loan_number (бbranch_name = ‘نادری’ (loan)) × borrower))

٦- اسامي تمام مشتريانی را که يك حساب در تمام شعبه هاي شهر قزوين دارند را بدهيد .

Пcustomer_name , branch_name (depositor ∞ account) ÷Пbranch_name (бbranch_city = “قزوين” (branch))

❊نكات مهم جبر رابطه اي

۱- جبر رابطه اي زباني است روشمند يعني برنامه ساز نه تنها به سيستم مي گويد چه مي خواهد بلكه نحوه بدست آوردن آنچه را كه مي خواهد نيز بيان مي كند .

۲- معادل است با محاسبات رابطه اي يعني هرتواني كه مجموعه امكانات محاسباتي دارد جبررابطه اي نيز
دارد .

۳- جبر رابطه اي فقط براي بازيابي نيست بلكه مي توان در عمليات ذخيره سازي و به هنگام سازي نيز
استفاده كرد .

مثال : درج كنيد < S7,S n 7...> .

در جبر رابطه اي مي توان از عملگر اجتماع براي درج استفاده نمود :

r U E ⟵ و r رابطه و E رابطه ثابت و داراي يك تاپلي است .

S U {s# = s7, sname =sn7,...}

مثال : حذف کنيد <S4,Sn4,...> .

S Minus {s# = s4 }

از عملگر تفريق براي حذف از يك رابطه استفاده مي شود . r – E ⟵ r

r رابطه و E رابطه اي با شرايط حذف مي باشد .

*محاسبات رابطه اي

محاسبات رابطه اي امكان ديگري است براي انجام عمليات روي رابطه ها و كار با بانک رابطه اي محاسبات رابطه اي حالت ناروشمند (غير رويه اي) دارد يعني در آن عمليات لازم براي اشتقاق رابطه اي از رابطه هاي ديگرتصريح نمي شود . به عبارتي تنها آنچه كه بايد بازيابي شود تشريح مي شود و چگونگي بازيابي آنها بيان نمي شود .

محاسبات رابطه اي در دو شاخه بيان مي شود :

١- محاسبات روي تاپل ها Tuple Relational Calculus

٢- محاسبات روي ميدان ها Domain Relational calculus

ايـده اولـيه اسـتفـاده از محـاسـبات رابـطه اي نخـسـتـين بار توسط Knuth مطرح شـد و سـپس كاد نحـوه استفـاده از آن را بـراي انجـام عـمليات روي رابطه هاي بانك رابطه اي بيان كرد.بر اساس كارهاي كاد زبان Alpha طراحي شد كه البتـه به مرحله پياده سازي نرسـيد . بعدها زباني بنام Quel برگرفته از Alpha طراحي شد كه به نوعي مي توان آن را نمونه پياده سازي Alpha دانسـت .

يكي از مهـمترين جنـبه هاي محـاسبات رابطه اي ، مفهوم متغير طيفي يا متغير تاپلي است . متغـير تاپلي متغـيري است كه مقـاديرش از يـک رابـطـه برگرفته مي شـود و به عبارت ديگ ر ، طيف مقادير مجازش ،همان تاپلـهاي مجمـوعه بدنه رابطه است . متغير تاپلي را بايستي تعريف كرد كه در زبان Quel بصورت زير است :

```
Range of SX is S ;
RETRIEVE ( SX.S#) WHERE SX.CITY = `TEHRAN`
```

در اينجا متغير SX همان متغير تاپلي است كه مقاديرش تاپلهاي S هستند .

در محاسبات رابطه اي با متغير ميداني نيز بجاي متغير تاپلي ، متغير ميداني وجـود دارد و زبانهاي متعددي مبتني بر اينگونه از محاسبات رابطه اي طراحي شده اند كه از جمله مي توان به زبانهاي زير اشاره كرد .

ILL -
Formal Query Language FQL -
Query By Example QBE -

❊محاسبات رابطه اي تاپلي

گرامر زبان مبتني بر محاسبات رابطه اي را مي توان در كتابهاي مختل ف بانك اطلاعاتي از جمله كتاب DATE مشاهده كرد كه خارج از بحث اين درس مي باشد .در اينجا بطور خلاصه برخي مفاهيم اساسي موجود در محاسبات رابطه اي بيان مي شود و سپس به ذكر چند مثال بسنده مي كنيم .

❊تعريف متغير تاپلي :

متغير تاپلي به كمك حكم زير تعريف مي شود :

$$\text{Range of T is } x_1 ; x_2 ; x_3 ; \dots ; x_n$$

كه در آن T متغير تاپلي و x1 و X2 و . . . Xn عبارات محاسبات رابطه اي هستند و نمايانگر رابطه هايي مثل R1 و R2 و . . . Rn مي باشند . بديهي است اگر درليست رابطه ها تنها يک رابطه داشته باشيم در اينصورت متغير تاپلي مقاديرش را از همان رابطه R می گيرد .

مثال :

RANGE OF SX IS S
RANGEVAR SX RANGES OVER S ; يا
RANGE OF SPX IS SP ;
RANGEVAR SPX RANGES OVER SP ; يا

❊عملگر ها

دو عملگر در محاسبات رابطه اي تعريف شده اند كه به آنها سور گويند . كه عبارتند از :

- سور وجودي Exist Quantifier به معناي وجود دارد .
- سور همگاني FOR.ALL Quantifire به معناي براي همه .

به كمك اين دو سور عبارات محاسبات رابطه اي نوشته مي شوند و در آن گزاره هايي بر نهاده مي شود . حاصل ارزيابي اين عبارات ممكن است true و يا false باشد .

يادآوري : فرض X □ N

EXISTS X (X>10) ⟶ **True**

EXISTS X (X<-5) ⟶ **False**

N1	N2	N3
1	2	3
1	2	4
1	3	4

1) EXISTS T (T(N1)=1) : True
2) FORALL T (T(N1)=1) : True
3)EXISTS T (T(N1)=1 AND T(N3)>5) : False

متغير تاپلي روي جدول TNumB

* **توجه :** سور همگاني FORALL را مي توان به كمك سور وجودي EXISTS بيان نمود .

FORALL X(f) =NOT EXISTS X (NOT f)

❊ **استفاده در حساب محمولات در فرموله کردن پرس و جو ها :**

سوال ۱ : شماره تهیه کنندگان ساکن c2 با وضعیت بیشتر از ۲۰ را بیابید .

Sx.s# WHERE sx.city = `c2` AND sx.STATUS > 20

سوال ۲ : شماره جفت تهیه کنندگان ساکن یك شهر را بدهید .

Sx.s# , Sy. s# WHERE sx.city =sy.city AND sx.s# < sy.s#

سوال ۳ : اسامي تهیه کنندگان قطعه P2 را بدهید .

Sx.SNAME WHERE EXISTS Spx (Spx.s# = Sx. s# AND Spx.p# = ` p2`)

سـوال ٤ : نام تهـيه كنـندگاني را بـدهيد كه دست كم يك قطـعه قرمز را تهيه مي كنند .

SX.SNAME
WHERE EXISTS SPX (SX.S# =SPX.S# AND EXISTS PX (PX.P# =SPX.P# AND PX.COLOR = `RED`))

سوال ٥ : نام تهيه كنندگاني را مشخص كنيد كه حداقل يك قطعه از قطعات عرضه شده توسط تهيه كننده s2 را تامين می كنند .

SX.SNAME
WHERE EXISTS Spx (EXISTS SPY (SX.S# =SPX.S# AND SPX.P# =SPY.P#AND SPY.S#=`S2`))

سوال ٦ : اسامي تهيه كنندگاني را بدهيد كه تمام قطعات را تهيه مي كنند .

SX.SNAME WHERE FORALL PX (EXISTS SPX (SPX.S# =SX.S# AND SPX.P#=PX.P#))

Sx.SNAME WHERE NOT EXISTS PX (NOT EXISTS SPX (SPX.S# = SX.S# AND SPX.P# = PX.P#))

سوال ۷ : شماره و وزن قطعاتي را تهيه كنيد كه وزن هر قطعه بر حسب گرم بيشتر از ۱۰۰۰ باشد .

(px.p# , px.WEIGHT * 454 AS GWMT)
WHERE (px.WEIGHT * 454> 1000)

سوال ۸ : شهر قطعاتي را مشخص كنيد كه بيش از سه قطعه آبي در آن انبار شده باشد .

Px.city
WHERE COUNT (PY WHERE PY.city =px.city AND py.color=`BLUE`)>3

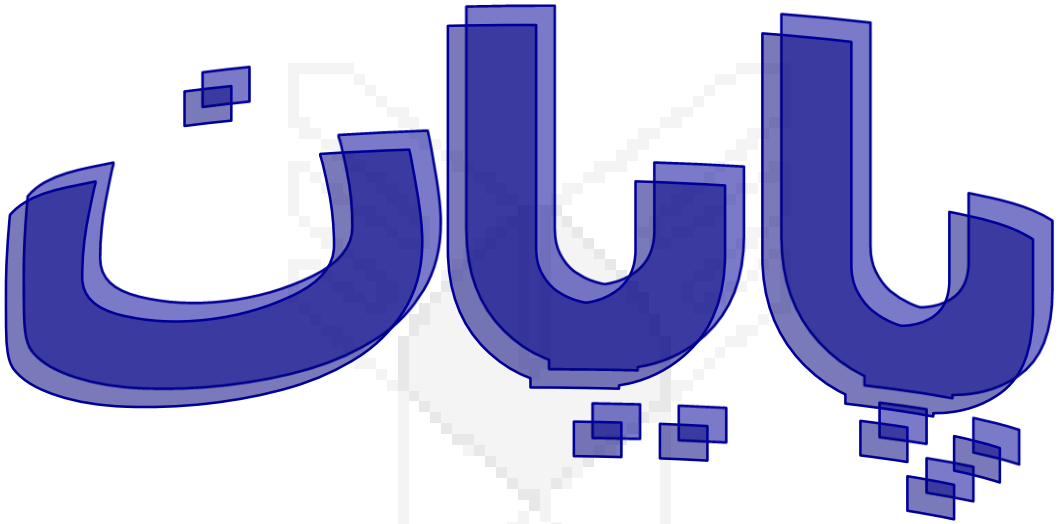
* **نكات مهم در خصوص آناليز رابطه اي :**

١- محاسبات رابطه اي ناروشمند است و تفاوت اصلي اش با جبر رابطه اي همين است .
٢- جبر رابطه اي و محاسبات رابطه اي معادلند .
٣- هر دو اكمال رابطه اي دارند يعني هر رابطه متصور از رابطه هاي ممكن قابل اشتقاق به كمك هر يک از اين دو امكان مي باشد .
٤- زبانهاي مبتني بر محاسبات رابطه اي به دليل ناروشمند بودن سطحشان بالاتر است .
٥- هر پرس و جو مبتني بر محاسبات رابطه اي را مي توان بفرم { t /p (t) } نيز نمايش داد بطوريكه :

- مجموعه اي از تمام تاپلهاي t است كه شرط p براي آن درست است .
- t يك متغير تاپلي است و t[A] نشان دهنده مقدار تاپل t روي صفت خا صه A مي باشد.
- t∈ r به اين معناست كه تاپل t در رابطه r است .
- p يك فرمول است كه شرط محاسبات مي باشد .

مثال : مشخصات تهيه كنندگان با وضعيت بيش از ١٠ را بدهيد .

S ^ t[status] > 10 {t / t }


پایان