پایگاه داده ها

(ایجاد بانکهای اطلاعاتی)

مدرس :

جناب آقای دکتر پورحاجی کاظم

1 . DBMS Architecture

2 . Different Concepts in DBMS

3 . ER (Entity Relationship) Model

4 . Relational Data Model

5 . Relational Algebra

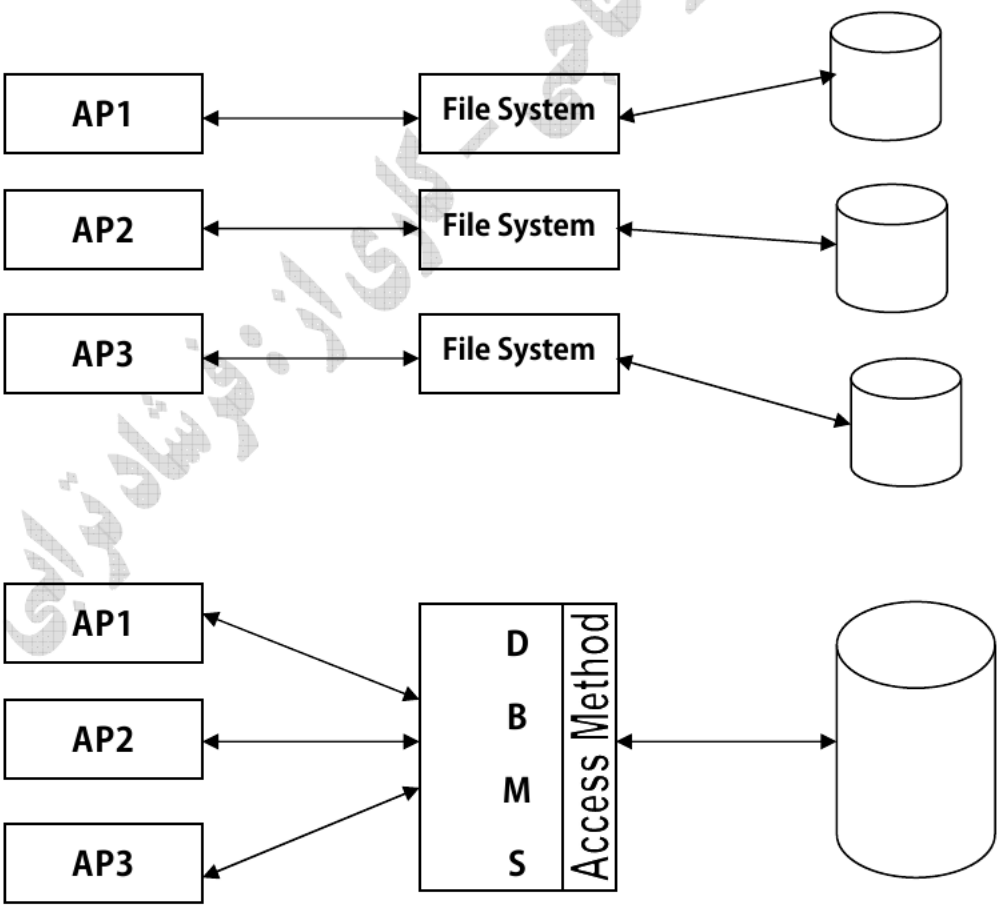
6 . SQL (Standard Query Language)

7 . Functional Dependency

8 . Normalization

File based system VS. Database system

مقایسه سیستم مبتنی بر فایل با پایگاه داده



- تفاوت ها :

۱ – افزونگی Redundancy

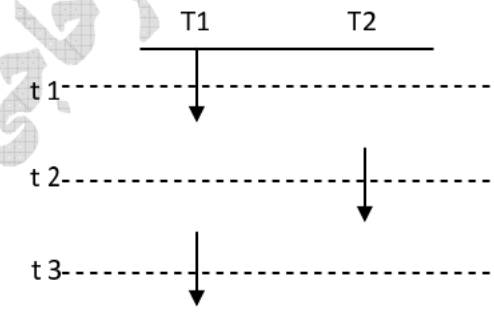
۲ – امنیت Security

۳ – جامعیت Integrity

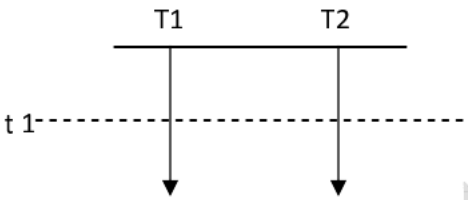
۴ – DSL (Data Sub Language)

۵ – همروندی Concurrency

۶ – ترمیم Recovery



همروند



همزمان (موازی)

- عناصر اصلی محیط پایگاه داده :

۱ . نرم افزار — سیستمی / کاربردی

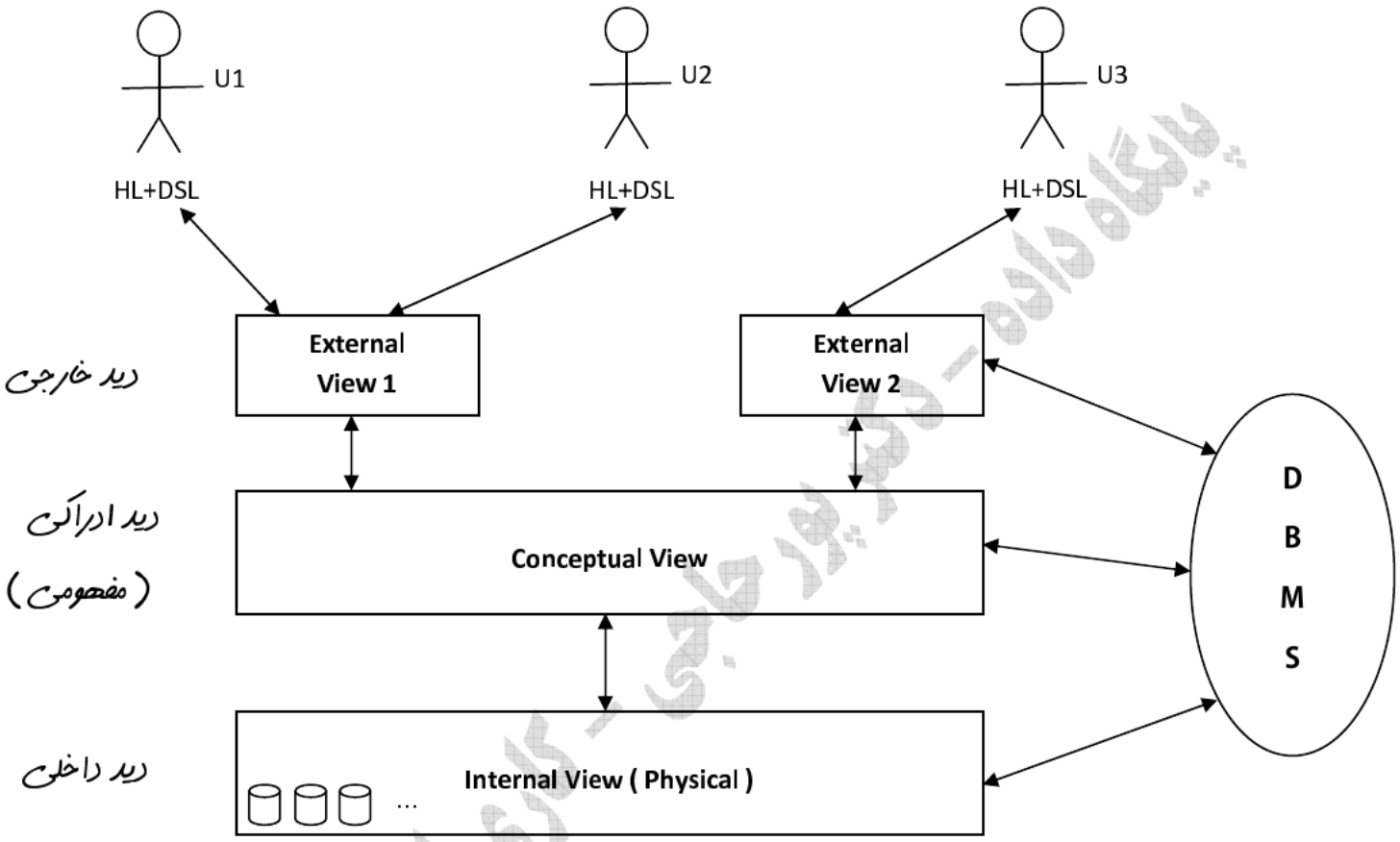
۲ . سخت افزار — کامپیوتر / وسایل ارتباطی

۳ . داده

۴ . کاربر — DBA / برنامه نویس / End User

- معماری پایگاه داده :

3 Layer (ANSI / SPARC) Architecture



External View : دید کاربران پایگاه داده نسبت به آن می باشد .

Conceptual View : دید مدیر پایگاه داده (DBA) نسبت به آن می باشد (معمولا همان ER)

Internal View : نحوه ذخیره شدن پایگاه داده روی رسانه ذخیره سازی است .

HL (Host Language): برای نوشتن user interface برنامه به کار می رود

مانند : VC++ , C# , Java , ...

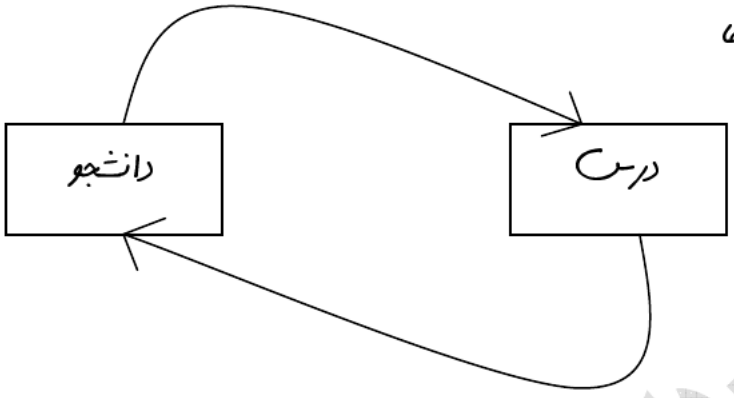
ER (Entity Relation) Model

مدل ارتباط موجودیت ها

۱ . شناسایی موجودیت ها

۲ . ارتباط بین موجودیت ها

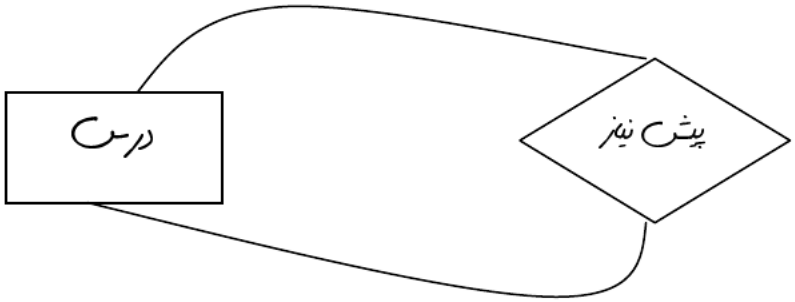
روش منسوخ



درجه رابطه: به تعداد موجودیت هایی که در یک رابطه شرکت می کنند ، درجه آن رابطه می گویند.

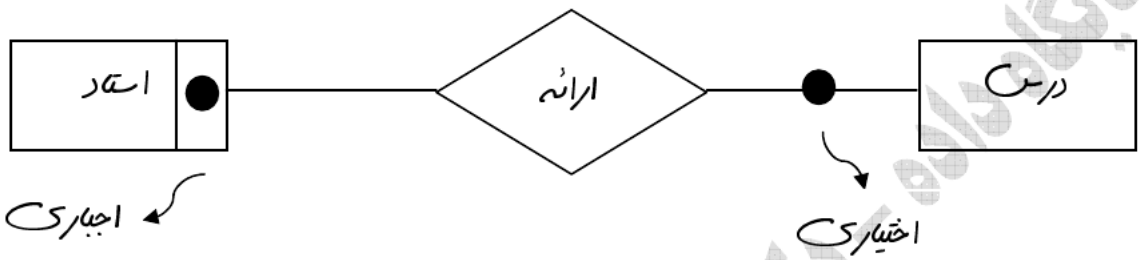
بعنوان مثال رابطه بالا یک رابطه درجه ۲ است ، درجه ی یک رابطه ۱ هم می تواند باشد ،

مانند:



- **Connectivity** (اتصال)
  - 1-1
  - 1-n
  - n-n

شرکت اجباری و اختیاری در رابطه:

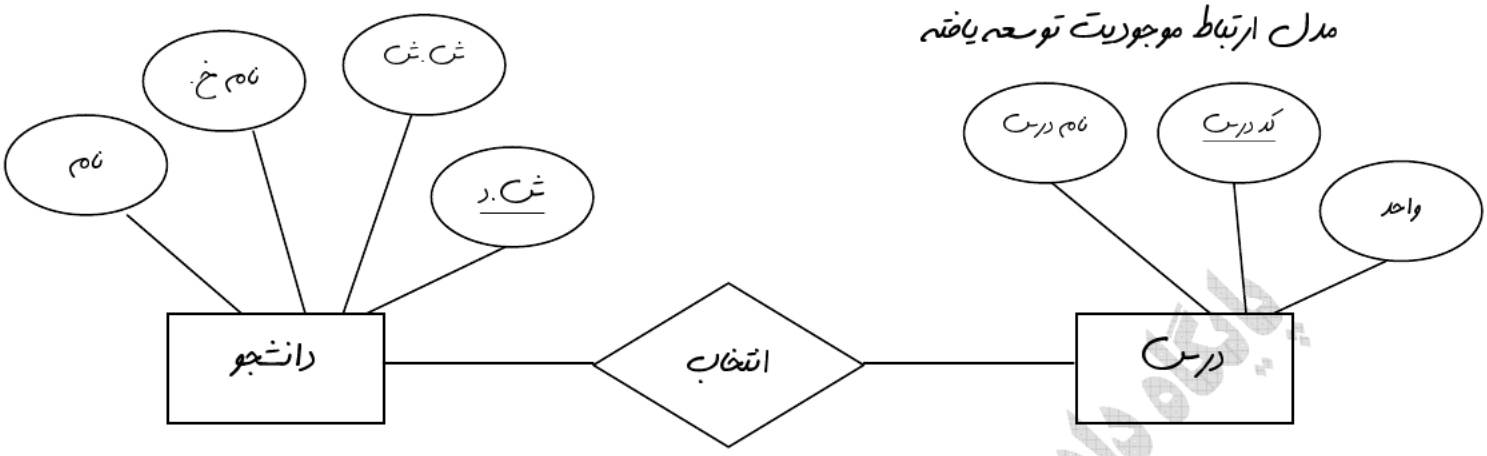


در اینجا در یک ترم می توانیم درس داشته باشیم که ارائه نشود ، ولی استاد حتما باید حداقل یک درس ارائه کند .

- **Cardinality** (حدّ) :

EER Model (Extended Entity Relation)

مدل ارتباط موجودیت توسعه یافته



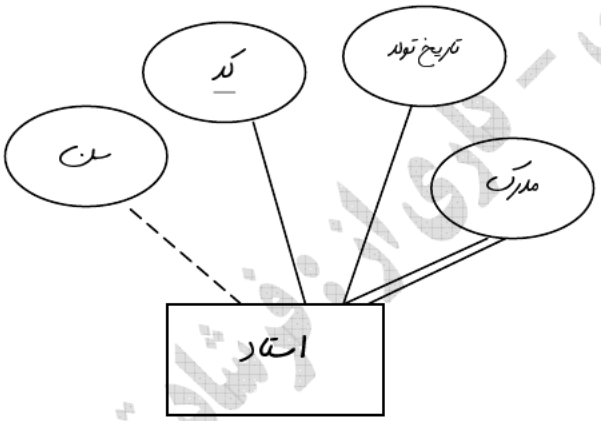
- انواع صفت خاصه:

۱. Simple / Composite (ساده / مرکب)

۲. Single Valued / Multi Valued (تک مقداری / چند مقداری)

۳. Key (کلیدی)

۴. Derived (مشتق شده)



صفت ساده / مرکب : صفتی که قابل تجزیه نباشد ساده ، و اگر قابل تجزیه باشد مرکب است .

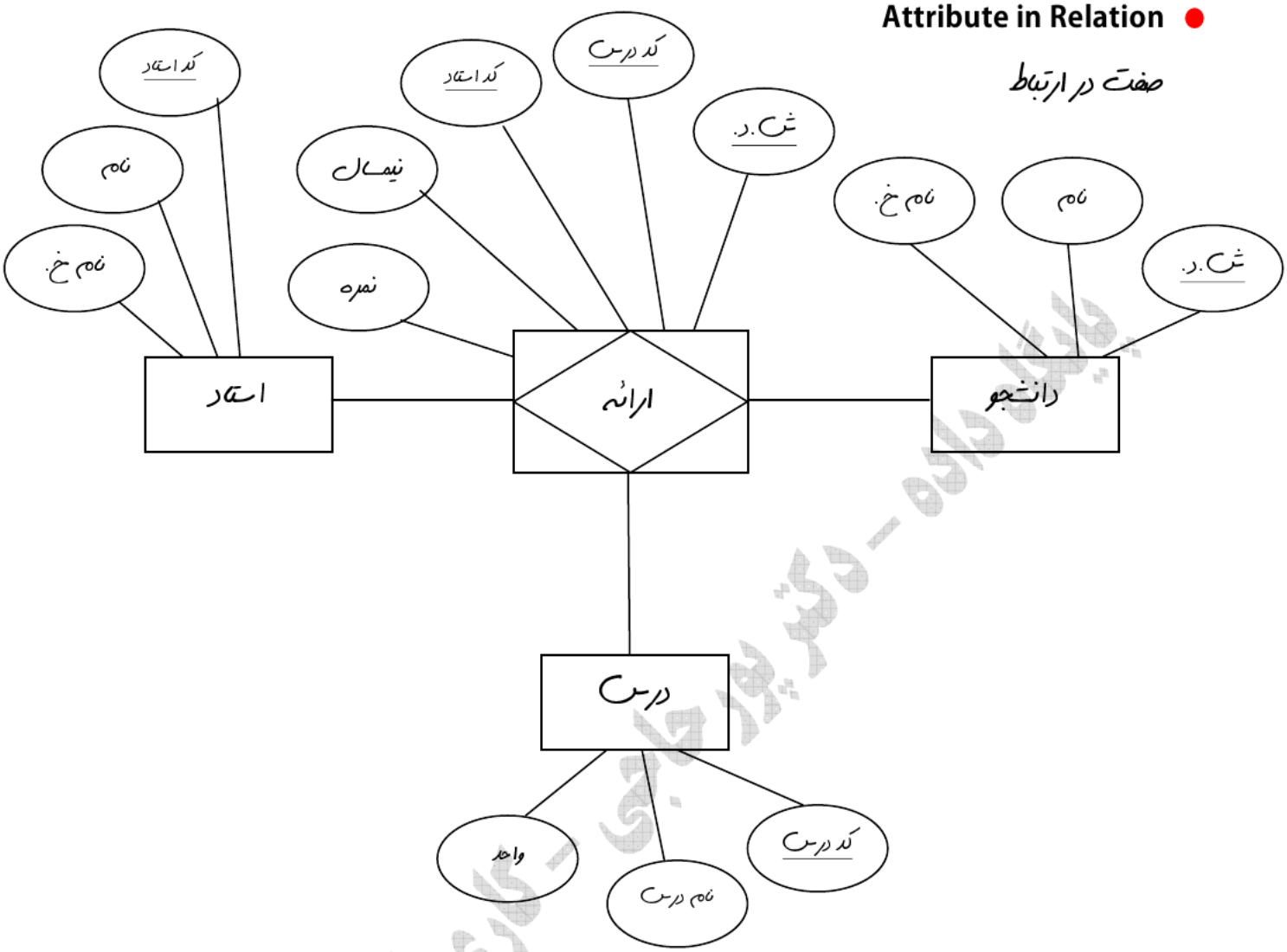
صفت تک مقداری / چند مقداری : چند مقداری مثل چند مدرک برای یک استاد

صفت کلیدی : صفتی که در بین نمونه های مختلف موجودیت، مقدار منحصر به فرد داشته باشد.

صفت مشتق شده : صفتی است که از روی صفات دیگر محاسبه شده و به دست می آید .

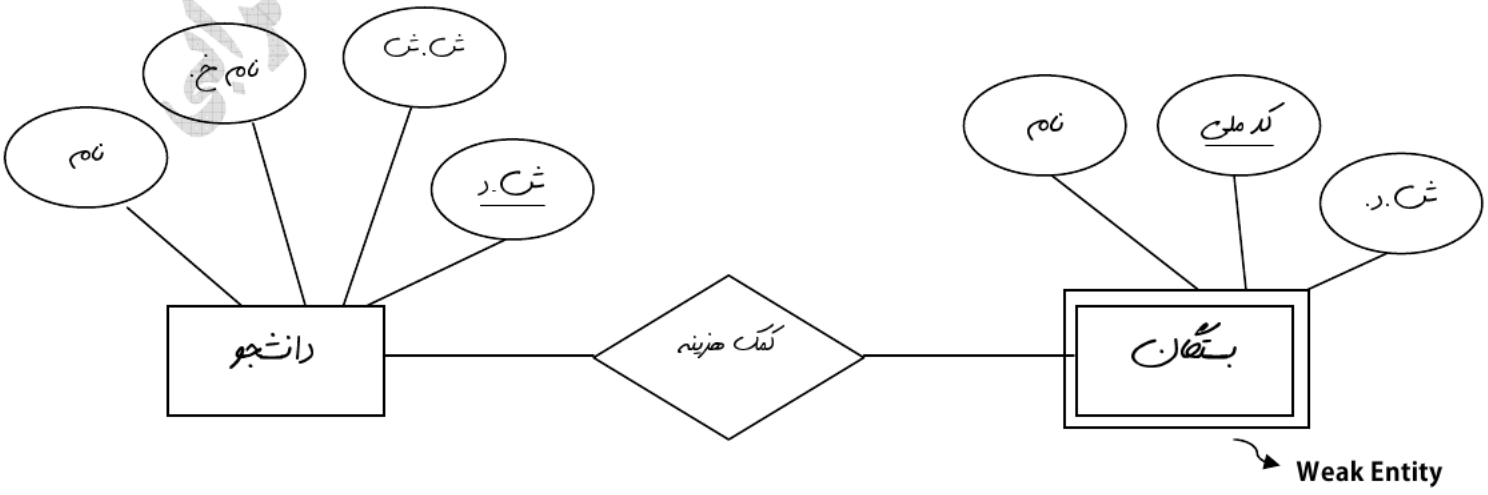
Attribute in Relation

صفت در ارتباط



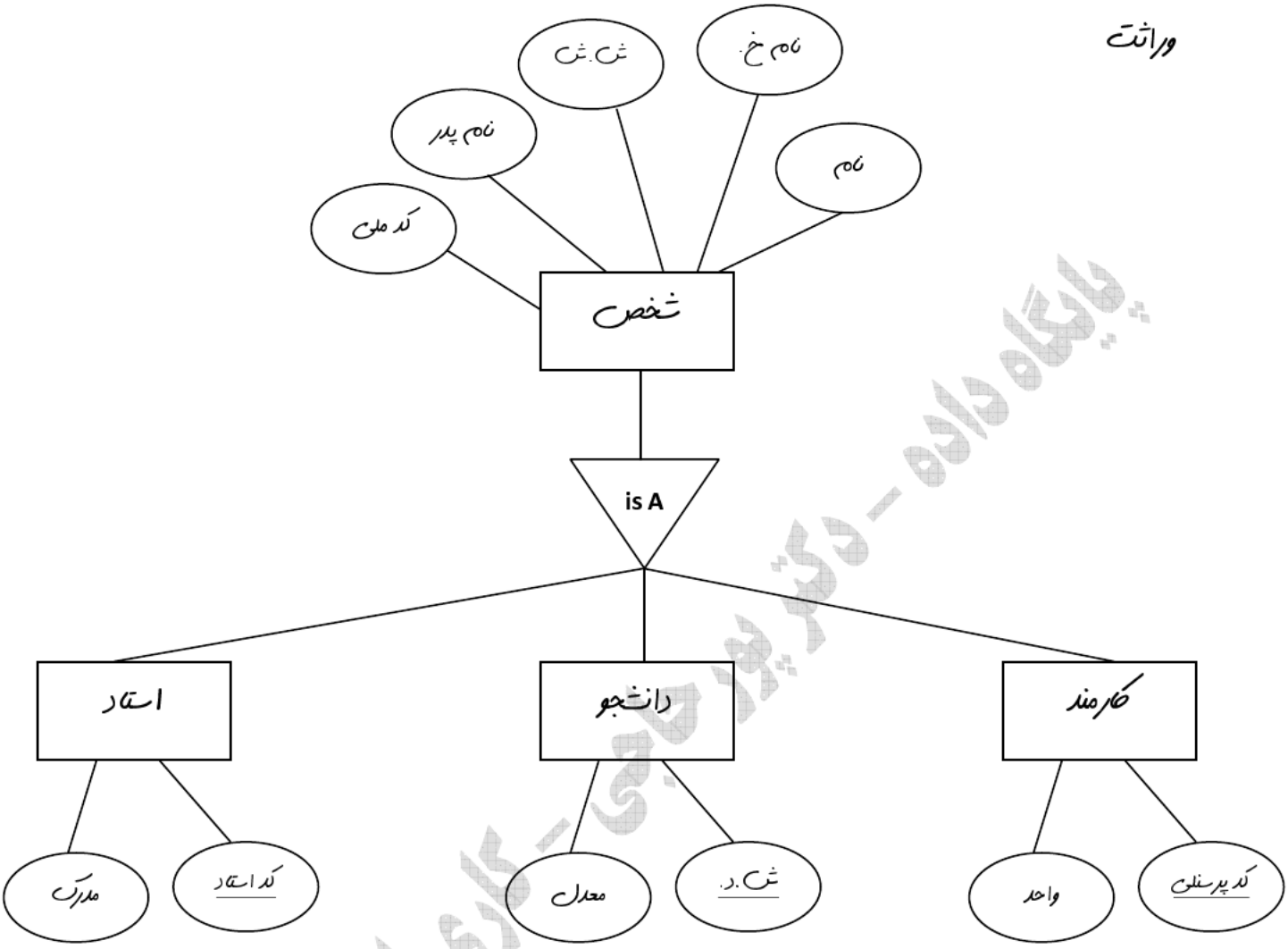
Existence Dependency

وابستگی وجودی



Attribute Sharing (Inheritance)

وراثت



- دید ادراکی
  - عام ← ER Model
  - خاص ← Data Model
    - Hierarchical سلسله مراتبی
    - Network شبکه
    - Relational رابطه‌ای (جدولی)
    - Object Oriented شی گرا

- استقلال داده ها

۱ . استقلال فیزیکی داده ها : منظور این است که هرگونه تغییر در لایه داخلی از دید کاربران پنهان می ماند ، مثلا اگر ساختار فایل عوض شود یا نوع رسانه تغییر کند ، کاربران هیچ اطلاعی از این موضوع نخواهند داشت .

۲ . استقلال منطقی داده ها : هرگونه تغییر در لایه ادراکی از دید کاربران پنهان است .

نکته : تفاوت DBA و DA در این است که DA فرد غیرفنی در امر پایگاه داده است که تنها نیازمندی های داده ای و جریان داده ها و ... را مشخص می کند ، ولی DBA متخصص پایگاه داده است و مدل ER را ترسیم می کند .

- Transaction (تراکنش)

تعریف : مجموعه ای از عملیات است که روی پایگاه داده اجرا می شود .

یک تراکنش معمولا از دستورات زیر تشکیل می شود :

Read (a) ← خلم داده a را از پایگاه داده می خواند .

Write (a) ← خلم داده a را در پایگاه داده می نویسد .

Abort یا Rollback ← زمانی که این دستور در بدنه تراکنش اجرا شود، اجرای آن تراکنش با شکست مواجه شده و تمامی تغییرات انجام شده توسط این تراکنش در پایگاه داده باید خنثی گردد.

Commit ← با اجرا شدن آن ، اجرای یک تراکنش با موفقیت به پایان رسیده و تمامی تغییراتی که توسط آن اعمال شده ، تثبیت می شود .

خصوصیات یک تراکنش :

Atomicity **C**onsistency **I**solation **D**urability

Atomicity (یکپارچگی) : یک تراکنش ، یا باید همه دستورات آن اجرا شده و با موفقیت **commit**کند و یا اینکه هیچ یک از دستورات آن اجرا نشود ، بنابراین اگر در وسط اجرای تراکنش ، دستور **abort** اجرا شود ، باید اثر تمامی دستورات اجرا شده خنثی گردد.

Consistency (سازگاری) : با اجرای یک تراکنش ، پایگاه داده از یک وضعیت سازگار به وضعیت سازگار دیگر می رود .

Isolation (انزوا) : اگر چند تراکنش بصورت همروند اجرا شوند ، اثر اجرای آنها باید مانند حالتی باشد که آن تراکنش در انزوا و بصورت تکی اجرا می شود .

Durability (پایایی) : اگر تراکنشی شروع به اجرا کرده و با موفقیت **commit**کند، اثرات آن، تحت هیچ شرایطی نباید از پایگاه داده پاک شود.

Data Dictionary یا فرهنگ داده : شامل توضیحاتی در رابطه با هر اسم استفاده شده در پایگاه داده می باشد ، مثلا اسم یک جدول یا **view**

System Catalog : اطلاعات دیگری وسیع تر از فرهنگ داده است ، و خود فرهنگ داده هم زیر مجموعه ای از کاتالوگ سیستم است .

Relational Data Model

مدل داده ای رابطه‌ای (جدولی)

رابطه:

$x R y = (x,y)$

$R1 = \{ (2,4),(5,6),(3,9) \}$

یادآوری ضرب دکارتی:

$A = \{ 1,2 \} \qquad B = \{ 3,4 \}$

$A \times B = \{ (1,3),(1,4),(2,3),(2,4) \}$

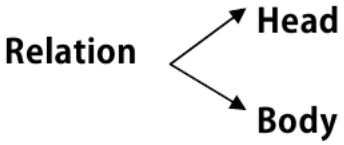
مثلا **R1** زوج مرتب هایی که جمع آنها زوج است:

$R1 = \{ (1,3),(2,4) \}$

می توان رابطه‌ای مانند **R2** بصورت زیر داشت:

$R2 = R1 \times \{5,6\} = \{ (1,3,5),(1,3,6),(2,4,5),(2,4,6) \}$

به یک رابطه در ریاضی **Relation** و در بحث پایگاه داده **Table** گویند و به همین ترتیب ، به هر چندتایی مرتب در ریاضی **Tuple** و در پایگاه داده **Record** گویند.



Name	Lname	STNO
Farshad	Torabi	896901
Reza	Hasani	906952
Ali	Modiri	906941

Head (R) = { Name,Lname,STNO }

Body (R) = { (Farshad,Torabi,896901),(Reza,Hasani,906952),(Ali,Modiri,906941) }

Degree (درجه) رابطه : تعداد ستون های یک رابطه را درجه رابطه گویند .

Cardinality (حدّ) رابطه : تعداد سطرها (**records**) یک رابطه را **Cardinality** آن گویند .

Degree (R) = n(Head(R))

Cardinality (R) = n(Body(R))

خصوصیات مدل داده ای رابطه ای :

۱ . در یک رابطه یک ستون نمی تواند نمایانگر یک صفت مرکب باشد .

۲ . در یک رابطه ترتیب ستون ها مهم نیست .

۳ . در یک رابطه **record**ها ترتیب خاصی ندارند .

۴ . در یک رابطه **record** تکراری وجود ندارد .

- انواع **Key** :

1 – Super Key

2 – Candidate Key

3 – Primary Key

4 – Secondary Key (Alternative Key)

5 – Foreign Key

Super Key (ابر کلید) : مجموعه صفات **A** در یک رابطه ابر کلید است اگر در نمونه های مختلف **record** های آن رابطه ، مقدار منحصر به فردی داشته باشد .

Candidate Key (کلید کاندید) : مجموعه صفات **A** در یک رابطه کلید کاندید است اگر اولا ابرکلید باشد ، ثانیا با حذف هرکدام از صفات آن ، خاصیت منحصر به فرد بودنش از بین برود.

Primary Key (کلید اصلی) : کلید کاندیدی است که از طرف مدیر پایگاه داده انتخاب می شود . معیار مدیر پایگاه داده برای انتخاب کلید اصلی از بین کلیدهای کاندید ، به شرح زیر است :

الف) درجه اهمیت کلید کاندید مربوطه (مثلا اکثر جستجو ها براساس آن باشد)

ب) کوچک بودن از نظر سایز

Secondary Key (کلید ثانویه یا فرعی) : کلید کاندیدی است که بعنوان کلید اصلی انتخاب نشده است .

Foreign Key (کلید خارجی) : صفت یا صفاتی است که در یک جدول دیگر بعنوان کلید اصلی تعریف شده اند .

Integrity Rules in Relational Model

قوانین جامعیت در مدل رابطه ای

1 – Intra-Relation integrity rule درون رابطه ای

2 – Entity integrity rule موجودیتی

3 – Referential integrity rule ارجاعی

قانون جامعیت درون رابطه ای : یک رابطه به تنهایی درست تعریف شده باشد (مثلا شامل صفت مرکب نباشد) .

قانون جامعیت موجودیتی : این قانون می گوید که مقدار هریک از ستون های کلید اصلی ، نمی تواند **NULL** باشد .

قانون جامعیت ارجاعی : بیانگر این موضوع است که مقدار کلید خارجی یا باید **NULL** باشدو یا همان مقداری را داشته باشد که در جدول اصلی دارد .

Relational Algebra

جبر رابطه ای

نکته مهم : مجموعه رابطه ها نسبت به همه عملگر های جبر رابطه ای بسته است ، بعبارت دیگر نتیجه اعمال هر عملگر جبر رابطه ای روی رابطه ها ، باز هم یک رابطه است .

عملگر انتخاب (گزینش یا **Select**) : برای انتخاب **record**هایی از یک رابطه بکار می رود که شرط یا شروطی را ارضا کنند .

$$\delta_{\text{شرط}}(R)$$

جدول زیر را که **S** متعلق به تهیه کنندگان و **P** قطعات می باشد را در نظر بگیرید :

S (S# , Sname , City , Status)

P (P# , Pname , Color , Type , Weight , City)

SP (S# , P# , Qty)

از این پس در هر سئوال **Query** (پرسش) از پایگاه داده مطرح خواهد شد.

*مشخصات تهیه کنندگانی را بدهد که در شهر تبریز هستند .

$$\delta_{City='TBZ'}(S)$$

*مشخصات قطعاتی را بدهد که به رنگ آبی هستند و جنس آنها از آهن است .

$$\delta_{Color='Blue' \text{ and } Type='iron'}(P)$$

عملگر پرتو (**Project**) : برای انتخاب ستون یا ستون هایی از یک رابطه بکار می رود.

$$\Pi (R)$$

نام ستون ها

*نام و شهر همه تهیه کنندگان را بدهد .

$$\Pi_{Sname,City}(S)$$

*شماره قطعاتی را بدهد که توسط تهیه کننده **S2** تهیه شده اند.

$$\Pi_{P\#}\left(\delta_{S\#='S2'}(SP)\right)$$

*نام تهیه کنندگانی را بدهد که در شهر اصفهان هستند .

$$\Pi_{Sname}\left(\delta_{City='ISF'}(S)\right)$$

*شماره قطعاتی را بدهد که توسط تهیه کنندگان S2 و S5 تهیه شده اند.

$$\Pi_{P\#}\left(\delta_{S\#='S2' \text{ OR } 'S5'}(SP)\right)$$

● عملگرهای مجموعه ای

شرط :

R1 ∪ R2

R1 ∩ R2

R1 – R2

head (R1) = head (R2)

*نام شهرهایی را بدهد که هم محل انبار حداقل یک قطعه باشند و هم اینکه حداقل یک تهیه کننده در آن واقع شده باشد.

$$\Pi_{City}(S) \cap \Pi_{City}(P)$$

Cartesian Product

ضرب دکارتی

R1 x R2

Head(R1 x R2) = Head(R1) ∪ Head(R2)

Card(R1 x R2) = Card(R1) * Card (R2)

جدول S

S#	Sname	City	Status
S1	A	TBZ	20
S2	B	ISF	14
S3	C	THR	18

جدول P

P#	Pname	Color	Type	Weight	City
P1	D	Blue	Iron	250	THR
P2	E	Red	Plastic	350	TBZ
P3	F	Yellow	Wood	150	TBZ

S x P

S#	Sname	S.City	Status	P#	Pname	Color	Type	Weight	P.City
S1	A	TBZ	20	P1	D	Blue	Iron	250	THR
S1	A	TBZ	20	P2	E	Red	Plastic	350	TBZ
S1	A	TBZ	20	P3	F	Yellow	Wood	150	TBZ
S2	B	ISF	14	P1	D	Blue	Iron	250	THR
S2	B	ISF	14	P2	E	Red	Plastic	350	TBZ
S2	B	ISF	14	P3	F	Yellow	Wood	150	TBZ
S3	C	THR	18	P1	D	Blue	Iron	250	THR
S3	C	THR	18	P2	E	Red	Plastic	350	TBZ
S3	C	THR	18	P3	F	Yellow	Wood	150	TBZ

در مثال فوق ، همانگونه که مشاهده می کنید ، هر سطر از جدول S با تک تک سطرهای جدول P آمده است.

θ - Join

$$R1 \times R2 \equiv \delta_{\text{شرط: } \theta} (R1 \times R2)$$

Natural Join

پیوند طبیعی

$$R1 \infty R2 \equiv R1 \times_{R1.Col = R2.Col} R2$$

کلید اصلی — محمولا : کلید خارجی

*نام تهیه کنندگانی را بدهد که قطعه P2 را تهیه کرده اند .

$$\prod_{Sname} (\delta_{P\#='P2'}(S \infty SP))$$

*نام ، رنگ و وزن قطعاتی را بدهد که توسط S3 تهیه شده اند .

$$\prod_{Pname,Weight,Color} (\delta_{S\#='S3'}(P \infty SP))$$

*نام تهیه کنندگانی را بدهد که قطعات به رنگ قرمز تولید کرده اند.

$$\prod_{Sname} ((\delta_{Color='Red'}(P) \infty SP) \infty S)$$

$\propto$ Semi-Join

$$R1 \propto R2 \equiv \prod_{\text{ستون های R1}} (R1 \infty R2)$$

*مشخصات قطعاتی را بدهد که در تهران انبار شده و تعداد آنها بیشتر از ۲۰۰ باشد .

$$\delta_{City='THR'}(P) \propto \delta_{Qty > 200}(SP)$$

Outer Join
- Left Outer Join (L.O.J)
- Full Outer Join (F.O.J)
- Right Outer Join (R.O.J)

فرا پیوند

نتیجه فراپیوند چپ ۲ رابطه (L.O.J)، همان نتیجه join طبیعی است ، با این تفاوت که recordهای join نشدنی رابطه سمت چپ نیز به نتیجه اضافه می شود و به جای ستون های رابطه دوم NULL گذاشته می شود . R.O.J هم به همین ترتیب است و به جای رابطه سمت چپ ، بر رابطه سمت راست عمل می کند و F.O.J رکوردهای join نشدنی هر دو طرف را شامل می شود .

*نام و دپارتمان تمام اساتید را بدهد ، حتی اگر دپارتمان استادی هنوز مشخص نشده است.

Prof (Prof# , Pname , Dept#)

Dept (Dept#,title,off#)

$$\prod_{Pname,title} (Prof\ L.O.J\ Dept)$$

Semi-Minus

عملگر شبه تفاضل

$$R1\ Semi\text{-}Minus\ R2 \equiv R1 - (R1 \propto R2)$$

*نام تهیه کنندگانی را بدهد که قطعه P2 را تهیه نکرده اند .

$$\prod_{Sname} (S\ \ Semi\text{-}Minus\ \ \delta_{P\#='P2'}(SP))$$

عملگر جایگزینی

$$R1 \leftarrow R2$$

مثال :

$$R1 \leftarrow (S \infty SP) \infty P$$

$$R2 \leftarrow \delta_{P\#='P2'}(SP)$$

$$\Pi\ (R2)$$

عملگر نامگذاری

$$\rho_{K}\ (R)$$

به رابطه R در Query مربوطه اسم مستعار K می دهد .

مثال : نام و شهر جفت تهیه کنندگانی را بدهد که همشهری هستند (در خروجی اطلاعات زاید دیده نشود)

$$\Pi_{S.Sname,K.Sname,S.City}\ (S \times \rho_{K}\ (S))_{S.City = K.City \text{ and } S.S\# < K.S\#}$$

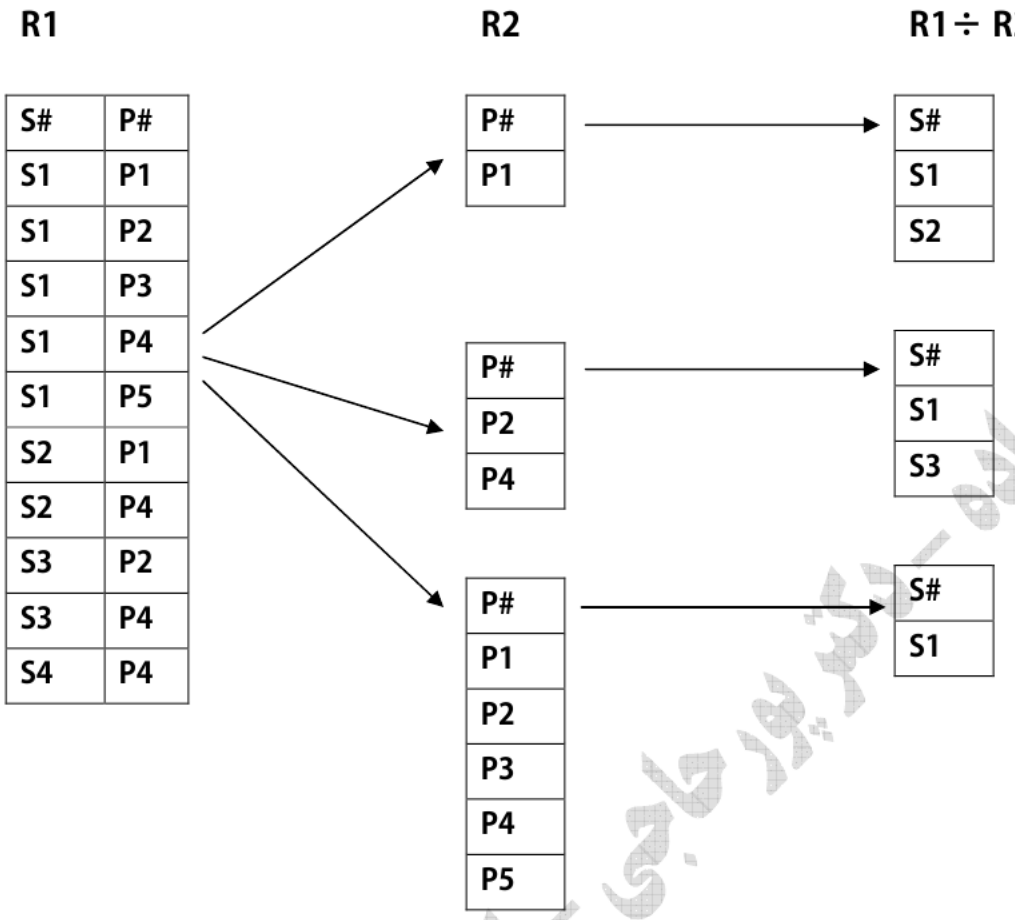
Division

تقسیم

$$R1 \div R2$$

شرایط : $Head(R2) \subset Head(R1)$

جواب : $Head(R1 \div R2) = Head(R1) - Head(R2)$



مثال : شماره تهیه کنندگانی را بدهد که همه قطعات را تهیه کرده اند.

$$\prod_{S\#,P\#}(SP) \div \prod_{P\#}(P)$$

SQL (Standard Query Language)

Data Types :

Small int , integer , float , decimal(p,q) , numeric(p,q) , char(n) , varchar(n) ,

Time , Date , Logical , TimeStamp , Bit(n)

ایجاد یک جدول :

بعنوان مثال جدول تهیه کنندگان را با SQL بدین صورت ایجاد می کنیم :

```
Create Table S
( SNO char(6),
Sname char(20) Not NULL,
City char(3),
Status small int,
Primary Key (STNO),
Check (Status>0) ) ;
```

و حال جدول SP

```
Create Table SP
(SNO char(6),
PNO char(6),
Qty integer,
Primary Key (SNO,PNO),
Foreign Key (SNO) References S
                On Delete Cascade
                On Update Cascade ,
Foreign Key (SNO) References S
                On Delete Cascade
                On Update Cascade  );
```

نکته: برای تعریف مقدار پیش فرض از **default** استفاده می کنیم ، مثلا :

Status small int default 17

ویرایش جدول :

Alter Table S

Add Address varchar(300);

Alter Table S

Modify SNO char(10);

نکته: بعد از تعریف کلید اصلی ، برای کلیدهای فرعی می توان منحصر به فرد بودن آنها را بصورت زیر حفظ کرد :

Unique (Nation_Code)

مثال برای کد ملی دانشجو در سیستم انتخاب واحد

ایجاد **index** :

Create Index SnIdx on S(Sname) ASC

نکته: در **SQL** برای نشان دادن حالت صعودی از **ASC** و حالت نزولی از **DESC** استفاده می کنیم .

دستور **Select** :

Select نام ستون ها

From نام جدول

Where شرط

Order By نام ستون ها

Group By نام ستون ها

Having شرط

*نام تهیه کنندگانی را بدهد که در شهر تبریز ساکن اند .

```
Select Sname
From S
Where City='TBZ'
```

*شماره قطعاتی را بدهد که توسط S2 تهیه شده اند .

```
Select P#
From SP
Where S#='S2'
```

عملگر in :

*شماره تهیه کنندگانی را بدهد که قطعات P2 یا P4 یا P7 را تهیه کرده اند .

```
Select S#
From SP
Where P# in ('P2','P4','P7')
```

عملگر between :

```
Select *
From S
Where Status between 15 and 19
```

عملگر like :

```
Select Sname,City
From S
Where Sname like '%Sazi' یا '_ _ t' یا '%Sazi%'
```

% نشانگر چند کاراکتر مجهول و _ یک کاراکتر مجهول است .

نکته: جهت جلوگیری از نمایش داده تکراری می توان از **Distinct** استفاده کرد:

```
Select Distinct S#
From SP
```

مرتب سازی نتیجه بر اساس ستون (ها)

```
Select Sname,City
From S
Order By Sname ASC یا DESC
```

عملگر **is NULL**:

```
Select Sname
From S
Where City is NULL
```

پیوند جدول ها:

ضرب دکارتی **SxP** ⟶

```
Select *
From S,P
```

θ - Join ⟶

```
Select *
From S,P
Where شرط
```

∞ ⟶

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

*نام تهیه کنندگانی را بدهد که قطعه P6 را تهیه کرده اند .

```
Select Sname
From S,SP
Where  S.S#=SP.S# and P#='P6'
```

*نام ، رنگ و جنس قطعاتی را بدهد که توسط S3 تهیه شده اند .

```
Select Pname,Color,Type
From P,SP
Where  P.P#=SP.P# and S#='S3'
```

*نام ووزن قطعاتی را بدهد که توسط ماشین سازی تبریز تهیه شده اند .

```
Select Pname,Weight
From S,SP,P
Where  S.S#=SP.S# and P.P#=SP.P# and S.City='TBZ' and Sname='MashinSazi'
```

*نام و شهر جفت تهیه کنندگانی را بدهد که همشهری هستند (اطلاعات زاید و تکراری ندهد)

```
Select First.Sname,Second.Sname,First.City
From S as First,S as Second
Where  First.City=Second.City and First.S#<Second.S#
```

Select های متداخل (تو در تو)

*نام تهیه کنندگانی را بدهد که قطعه P2 را تهیه کرده اند .

```
Select Sname
From S
Where  S# in (
               Select S#
               From SP
               Where P#='P2' )
```

*نام تهیه کنندگانی را بدهد که قطعات به رنگ قرمز تهیه کرده اند .

```
Select Sname
From S
Where  S# in (
              Select S#
              From SP
              Where P# in (
                           Select P#
                           From P
                           Where Color='Red' ) )
```

- جداول زیر مربوط به یک دانشگاه است :

Stud (S#,Sname,City,Avg,Clg#)

CRS (C#,Cname,Unit,Clg#)

Prof (P#,Pname,Deg,Clg#)

SEC (S#,C#,P#,Term,Score)

Clg (Clg#,Clgname,P#)

*نام دانشجویانی را بدهد که در دانشکده کامپیوتر تحصیل می کنند .

```
Select Sname
From S tud
Where  Clg# in (
                Select Clg#
                From Clg
                Where Clgname='Computer' )
```

*نام درس هایی را بدهد که استاد پورحاجی در نیمسال اول ۹۰-۸۹ ارائه کرده است .

```
Select Cname
From CRS
Where  C# in (
            Select C#
            From SEC
            Where Term=891 and P# in (
                                    Select P#
                                    From Prof
                                    Where Pname='Pourhaji' ) )
```

*نام دانشجویانی را بدهد که درس ریاضی ۱ را با استاد علوی که در دانشکده مکانیک ارائه شده را در نیمسال دوم ۹۱-۹۰ انتخاب کرده اند .

```
Select Sname
From Stud
Where  S# in (
        Select S#
        From SEC
        Where Term=902 and P# in (
                            Select P#
                            From Prof
                            Where Pname='Alavi' ) and C# in (
                                                Select C#
                                                From CRS
                                                Where Cname='Riazi1' and Clg# in (
                                                        Select Clg#
                                                        From Clg
                                                        Where Clgname='Mechanic' ) ) )
```

حال مثال بالا را به روش join می نویسیم :

```
Select Sname
From Stud,CRS,Prof,SEC,Clg
Where Stud.S#=SEC.S# and
        CRS.C#=SEC.C# and
        Prof.P#=SEC.P# and
        CRS.Clg#=Clg.Clg# and
        Cname='Riazi1' and Term=902 and Clgname='Mechanic' and Pname='Alavi'
```

توابع ستونی (Min,Max,Sum,Avg,Count,…)

مثال : جمع کل تعداد قطعات تهیه شده توسط تهیه کننده S1 را بدهد .

```
Select Sum(Qty) as TotQty
From SP
Where S#='S1'
```

نکته: تعداد دقیق کل رکورد ها با Select Count(*) قابل محاسبه است .

این دو Query اشتباه نوشته شده اند در اولی یک Qty با تک تک S# ها تکرار خواهد شد که منطقی نخواهد بود و دومی نیز به این شکل صحیح نیست .

```
Select S#,Min(Qty) as MinQ
From SP
```

```
Select S#
From SP
Where Qty < Avg(Qty)
```

```
Select S#
From SP
Where Qty <  (
            Select Avg(Qty)
            From SP )
```

عملگرهای مجموعه‌ای

Query 1 { ∪ : Union ; ∩ : Intersect ; − : Except } Query 2

مثلا

```
Select City From S
Union
Select City From P
```

Group By

مثال :

```
Select S#,Avg(Qty) as AvgQ
From SP
Where S# in ('S2','S7','S9')
Group By S#
```

شرط روی گروه (Having)

*شماره تهیه کنندگانی را بدهد که بیش از ۲ نوع قطعه تهیه کرده اند .

```
Select S#
From SP
Group By S#  having Count(P#) > 2
```

پیاده سازی عملگر ÷ در SQL

*شماره تحصیل کنندگانی را بدهد که همه قطعات را تحصیل کرده اند

```
Select S#
From SP
Group By S# having Count(P#) = (
                                Select Count(P#)
                                From P )
```

عملگر Contains

*شماره دانشجویانی را بدهد که همه درس های ارائه شده توسط استاد پورحاجی در نیمسال دوم ۸۹ را انتخاب کرده باشند .

```
Select S#
From SEC
Where Term=892
Group By S# having C# Contains (
                Select C#
                From SEC,Prof
                Where SEC.P#=Prof.P# and Term=892 and Pname='Pourhaji' )
```

عملگر Exists

*نام تحصیل کنندگانی را بدهد که قطعه P2 را تحصیل کرده اند.

```
Select Sname
From S
Where Exists (Select *
            From SP
            Where SP.SP#=S.S# and P#='P2' )
```

انواع جدول در پایگاه داده

۱ . پایه (Base Table)

۲ . میانی (Intermediate Table)

۳ . مجازی (Virtual Table)

جدول پایه همان جدول های رایج پایگاه داده است ، جدول میانی را خود **DBMS** برای اعمالی چون **Select** های تو در تو می سازد و پس از اتمام کار ، ازبین می برد و جداول مجازی ، همان **View** ها هستند که نحوه ایجاد آن را می بینیم :

```
Create View V1
AS Select S#,Sum(Qty) as TotQty
From SP
Group By S#
```

حال می توان از **View** ایجاد شده ، به این شکل استفاده کرد :

```
Select S#
From V1
Where TotQty > 500
```

مثالی دیگر :

```
Create View SSP
AS Select *
From S,SP
Where S.S#=SP.S#
```

و هر بار که **Join** طبیعی **S** و **SP** مورد نیاز بود ، این **View** را جایگزین سطرهای فوق نمود.

دستورات دستکاری داده ها

۱ . درج رکورد

مثال: **INSERT INTO S (S#,Sname,City,Status) VALUES ('S10','TractorSazi','TBZ',20)**

مثال ۲:

```
Create Table Temp1 (
S# char(5),
TotQty integer,
Primary Key (S#) );

INSERT INTO Temp1
Select S#,Sum(Qty)
From SP
Group By S#
```

۲ . حذف رکورد

DELETE From نام جدول **Where** شرط

مثال : مشخصات تهیه کنندگانی را که قطعه **P1** را تهیه کرده اند ، از جدول پاک کند .

```
 DELETE From S
Where S# in (
        Select S#
        From SP
        Where P#='P1' )
```

۳ . بروز رسانی جدول

UPDATE نام جدول

Set . . . مقدار جدید = نام ستون ۱ ، نام ستون۲=مقدار جدید و

Where شرط

مثال : شهر تهیه کننده به شماره **S1** را به تهران تغییر دهد .

UPDATE S

Set City='THR'

Where S#='S1'

Functional Dependency

وابستگی تابعی

$$\left.\begin{matrix}(x,y) \in R \\ (x,z) \in R\end{matrix}\right\} \Rightarrow y=z$$

مجموعه صفات **B** به مجموع صفات **A** وابستگی تابعی دارد ، اگر و تنها اگر به ازای هر مجموعه مقادیر برای صفات **A** فقط یک مجموعه مقادیر از صفات **B** وجود داشته باشد.

تعریف ابرکلید : اگر در یک رابطه تمامی صفات رابطه به مجموعه صفات **A** وابستگی تابعی داشته باشد، آنگاه **A** ابرکلید (**Super Key**) است .

Full Functional Dependency

وابستگی تابعی کامل :

مجموعه صفات **B** به مجموع صفات **A** وابستگی تابعی کامل دارد ، اگر **B** به **A** وابستگی تابعی داشته باشد، ولی به هیچ زیرمجموعه محض از مجموعه صفات **A** وابستگی نداشته باشد.

(S#,City) ⟶ Sname

وابستگی دارد ، ولی کامل نیست

(S#,P#) ⟶ Qty

وابستگی کامل دارد

تعریف کلید کاندید : در یک مجموعه صفات A کلید کاندید (Candidate Key) خواهد بود ، اگر همه صفات رابطه به A وابستگی تابعی کامل داشته باشند .

Normalization

سطوح مختلف دارد که در سه سطح بررسی خواهیم کرد :

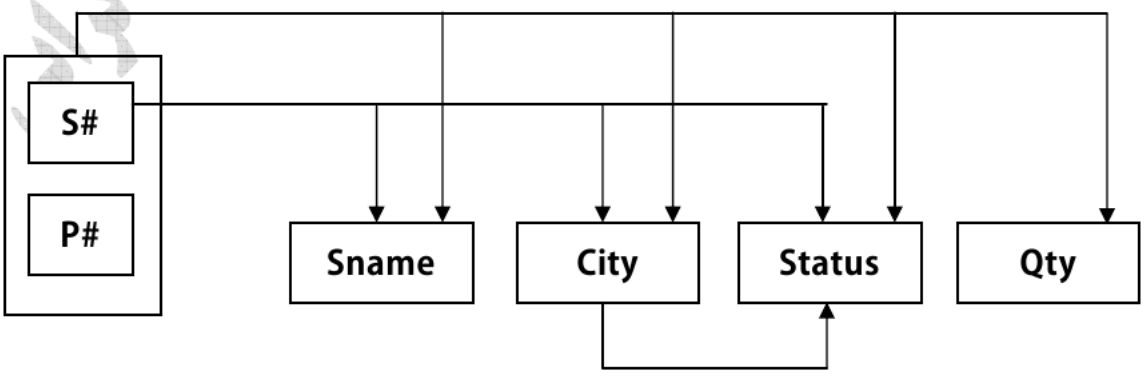
1NF,2NF,3NF

1NF : رابطه R در سطح اول نرمال قرار دارد اگر درست تعریف شده باشد و شامل صفات مرکب نباشد .

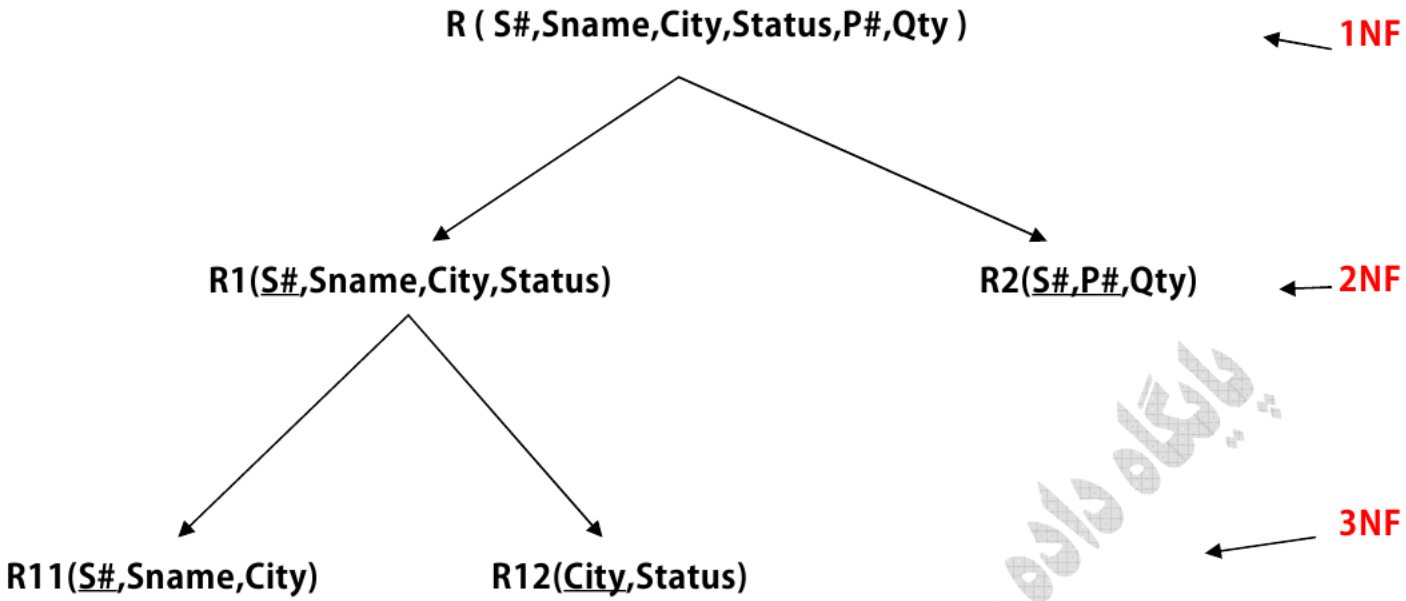
R (S#,Sname,City,Status,P#,Qty)

2NF : رابطه R در سطح دوم نرمال قرار دارد اگر اولا در سطح اول نرمال قرار داشته و ثانیاً تمامی صفات غیرکلیدی به کلید اصلی وابستگی تابعی کامل داشته باشند.

3NF : رابطه R در سطح سوم نرمال قرار دارد اگر اولا در سطح دوم نرمال قرار داشته و ثانیاً هیچ دو صفت غیرکلیدی از آن ، بهمدیگر وابستگی تابعی نداشته باشند .



نمودار وابستگی کامل



*تا سطح ۳ نرمال سازی کنید