بسمه تعالی

اصول طراحی پایگاه داده ها

علی چوداری خسروشاهی
Akhosroshahi@iaut.ac.ir

دانشگاه آزاد اسلامی

1

مراجع

- Data Base System Concepts
  - Silberschatz
- **مفاهیم بنیادی پایگاه داده ها**
  - [illegible]
- **بانکهای اطلاعاتی علمی کاربردی** (جلد ۱)
  - [illegible]

2

فصل اول

مقدمه درس

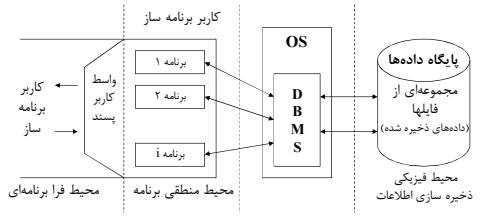
3

Database Management System (DBMS)

- مجموعه ای از داده های مرتبط
- مجموعه ای از برنامه های برای دستیابی به داده ها
- DBMS محیطی فراهم می کند که *کارا* و *راحت* برای استفاده می باشد.
- پایگاه داده تمام مفاهیم زندگی ما را تحت تاثیر قرار داده است.

4

Database Management System (DBMS)



5

DBP و DBA

- Data Base Administrator (DBA)
  - یک شخص و یا یک تیم می باشد که وظیفه ی طراحی و سیاست گذاری پایگاه داده را بر عهده دارد.
- Data Base Programmer (DBP)
  - یک شخص و یا یک تیم می باشد که وظیفه ی پیاده سازی و تصمیمات گرفته شده توسط DBA را بر عهده دارد.

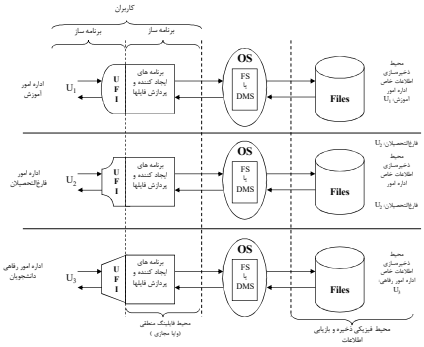
6

هدف سیستمهای پایگاه داده ای

- در سالهای قبل برنامه های پایگاه داده ای با استفاده از سیستم فایلها نوشته می شدند.
- تعریف پایگاه داده ها
  - *پایگاه داده ها مجموعه ای است از داده های ذخیره شده و پایا به صورت مجتمع به هم مرتبط ، حدالامکان با کمترین افزونگی، تحت مدیریت یک سیستم کنترل متمرکز مورد استفاده ی یک یا چند کاربر، به طور همزمان و اشتراکی.*

7

هدف سیستمهای پایگاه داده ای



8

ایرادات روش فایلینگ

- در کتاب رانکوهی
  - عدم وجود محیط مجتمع ذخیره اطلاعات و عدم وجود سیستم یکپارچه.
  - عدم وجود سیستم کنترل متمرکز روی کل داده های سازمان.
  - تکرار در ذخیره سازی اطلاعات.
  - خطر بروز پدیده نامطلوب ناسازگاری داده ها.
  - مصرف نا بهینه ی امکانات سخت افزاری و نرم افزاری ، حجم زیاد برنامه سازی ، استفاده ی نا بهینه از مهارت و وقت تیمهای برنامه سازی.
  - دشواری در گسترش سیستم های کاربردی و ایجاد کاربرد های جدید.
  - وابسته بودن برنامه های کاربردی به محیط ذخیره سازی داده ها

9

ایرادات روش فایلینگ

- Drawbacks of using file systems to store data:
  - Data redundancy and inconsistency
    - Multiple file formats, duplication of information in different files
  - Difficulty in accessing data
    - Need to write a new program to carry out each new task
  - Data isolation — multiple files and formats
  - Integrity problems
    - Integrity constraints (e.g. account balance > 0) become part of program code
    - Hard to add new constraints or change existing ones
  - Atomicity of updates
    - Failures may leave database in an inconsistent state with partial updates carried out
    - E.g. transfer of funds from one account to another should either complete or not happen at all
  - Concurrent access by multiple users
    - Concurrent accessed needed for performance
    - Uncontrolled concurrent accesses can lead to inconsistencies
      - E.g. two people reading a balance and updating it at the same time
  - Security problems

10

سطوح تجرید

- سطح فیزیکی: توضیح می دهد که یک رکورد (مانند مشتری) ذخیره سازی شده است
- سطح منطقی: داده های ذخیره سازی شده در پایگاه داده و رابطه مابین داده ها را توضیح می دهد

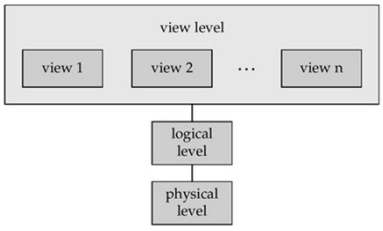
```
type customer = record
                    name : string;
                    street : string;
                    city : integer;
                end;
```

- سطح View: برنامه های کاربردی جزئیات نوع داده ای را مخفی می کنند. View ها همچنین برخی اطلاعات (مانند حقوق) را برای اهداف امنیتی مخفی می کنند.

11

View of Data

- یک معماری برای سیستم پایگاه داده



12

مدل

- **مدل (Schema)**
  - **مدل فیزیکی : پایگاه داده را در سطح فیزیکی طراحی می کند**
  - **مدل منطقی : پایگاه داده را در سطح منطقی طراحی می کند**
- **استقلال فیزیکی داده ها**
  - **توانایی تغییر در مدل فیزیکی بدون تغییر مدل منطقی**
  - **برنامه های کاربردی وابسته به مدل مدل منطقی می باشند**
  - **در کل، رابط مابین سطوح مختلف و مولفه ها تعریف خواهد شد برای اینکه تغییر در بخشهای یکسان به طور جدی در سایر بخشها تاثیر نگذارد**

13

زبانهای پایگاه داده

- به دو بخش تقسیم می شوند
- Data Definition Language (DDL)
  - ابزاری است برای تعریف شمای پایگاه داده
- Data Manipulation Language (DML)
  - ابزاری است برای دسترسی و دستکاری داده های سازمان یافته در پایگاه داده.
  - DML به عنوان زبان پرس و جو نیز شناخته می شود.
- زبانهای مجزایی نمی باشند و بخشهایی از یک زبان مانند SQL می باشند.

14

مدیریت تراکنش

- **یک تراکنش** (**transaction**) مجموعه ای از عملیاتهایی می باشد که یک عمل منطقی واحدی را در یک برنامه پایگاه داده انجام می دهد.
- **مولفه مدیریت تراکنش (Transaction-management component)** مطمئن می کند که پایگاه داده در وضعیت سازگار (صحیح) باقی خواهد ماند حتی با ایرادات سیستم (مانند power failures و operating system crashes) و ایرادات تراکنش.
- **مدیریت کنترل همروندی (Concurrency-control manager) تعاملات مابین تراکنشهای همروند را کنترل می کند تا سازگاری (consistency) پایگاه داده را تامین نماید.**

15

معماری پایگاه داده

- معماری پایگاه داده عموماً از سیستم کامپیوتری که پایگاه داده روی آن اجرا می شود تاثیر پذیرفته است.
  - معماری متمرکز
  - معماری مشتری–خدمتگذار
  - معماری توزیع شده
  - معماری با پردازش موازی
  - معماری چند پایگاهی
  - معماری موبایل

16

فصل دوم

مدل رابطه ای

Relational Model

17

نمونه ای از یک رابطه

account-number	*branch-name*	*balance*
A-101	Downtown	500
A-102	Perryridge	400
A-201	Brighton	900
A-215	Mianus	700
A-217	Brighton	750
A-222	Redwood	700
A-305	Round Hill	350

18

ساختارهای پایه ای

- **دامنه** (Domain)
  - مجموعه‌ای از مقادیر ممکن صفت (attribute) است. از نظر ریاضی، مجموعه‌ای است از مقادیر که یک یا بیش از یک صفت از آن مقدار می گیرند.
- **رابطه** (Relation)
  - رابطه R عبارت است از زیرمجموعه‌ای از ضرب کارتزین $D_1 \times D_2 \times D_3 \times \ldots \times D_n$، و می گوییم رابطه R از درجه n است.
    - هر یک از D_1، D_2 ، D_3 و ... میدان یا دامنه نامیده می شود.
- **تاپل** (Tuple)
  - ارتباط مجموعه‌ای از مقادیر در یک رابطه است.

19

ساختارهای پایه ای

- Example: if
  customer-name = {Jones, Smith, Curry, Lindsay}
  customer-street = {Main, North, Park}
  customer-city = {Harrison, Rye, Pittsfield}

  Then *r* = { (Jones, Main, Harrison),
  (Smith, North, Rye),
  (Curry, North, Rye),
  (Lindsay, Park, Pittsfield)}

  is a relation over *customer-name* × *customer-street* × *customer-city*

20

Attribute Types

- هر صفت رابطه یک نام دارد.
- مجموعه مقادیر مجاز هر صفت دامنه صفت نامیده می شود.
- مقادیر صفت باید *atomic* باشند، یعنی اینکه تجزیه نا پذیر باشند
  - نکته: مقادیر صفتهای چند مقداری atomic نیستند
- مقدار ویژه *null* عضو هر دامنه ای است.
- مقدار null موجب پیچیدگی در تعریف بسیاری از عملیاتها می شود.
  - ما باید از اثر مقادیر null در ارائه های اثلی خود صرف نظر و اثرات آنرا برای آینده در نظر داشته باشیم

21

Relation Schema

- $A_1, A_2, \ldots, A_n$ are *attributes*
- $R = (A_1, A_2, \ldots, A_n)$ is a *relation schema*

  E.g. *Customer-schema* =

  (*customer-name, customer-street, customer-city*)
- *r*(R) is a *relation* on the *relation schema* R

  E.g. *customer (Customer-schema)*

22

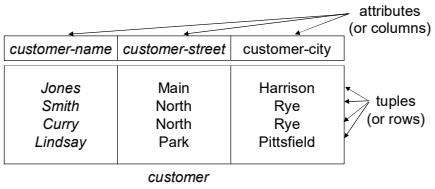
تناظر بین مفاهیم رابطه‌ای و مفاهیم جدولی

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

23

رابطه نمونه

- مقادیر صحیح از یک رابطه بوسیله یک *جدول* مشخص می شود.
- یک عنصر t از r یک *تاپل* می باشد، نماینده یک سطر در جدول می باشد.



24

ویژگیهای رابطه

- رابطه تاپل تکراری ندارد.
- تاپل مرتب نیست.
- صفات رابطه نظم مکانی ندارند (از چپ به راست).
- تمامی صفات تک مقداری هستند.

account-number	*branch-name*	*balance*
A-101	Downtown	500
A-215	Mianus	700
A-102	Perryridge	400
A-305	Round Hill	350
A-201	Brighton	900
A-222	Redwood	700
A-217	Brighton	750

25

پایگاه داده

- یک پایگاه داده از چندین رابطه تشکیل شده است.
- اطلاعات یک بنگاه اقتصادی به بخشهایی با روابط ذخیره سازی شده در یک بخش از اطلاعات تقسیم می شود.

acount : stores information about accounts
depositor : stores information about which customer owns which account
customer : stores information about customers

- ذخیره سازی تمام اطلاعات با یک رابطه مانند

bank(*account-number, balance, customer-name*, ..)

- باعث می شود
  - تکرار اطلاعات (مانند یک مشتری با دو حساب)
  - نیاز به مقادیر null (مانند نمایش مشتری که حساب ندارد)
- تئوری نورمالسازی نشان می دهد که چطور می توان رابطه های را طراحی کرد.

26

رابطه مشتری (customer)

customer-name	*customer-street*	*customer-city*
Adams	Spring	Pittsfield
Brooks	Senator	Brooklyn
Curry	North	Rye
Glenn	Sand Hill	Woodside
Green	Walnut	Stamford
Hayes	Main	Harrison
Johnson	Alma	Palo Alto
Jones	Main	Harrison
Lindsay	Park	Pittsfield
Smith	North	Rye
Turner	Putnam	Stamford
Williams	Nassau	Princeton

27

رابطه سپرده گذار (depositor)

customer-name	*account-number*
Hayes	A-102
Johnson	A-101
Johnson	A-201
Jones	A-217
Lindsay	A-222
Smith	A-215
Turner	A-305

28

کلیدها

- چند مفهوم در بحث کلید وجود دارد که عبارتند از:
  - ابر کلید (Super Key)
  - کلید کاندید (Candidate Key)
  - کلید اصلی (Primary Key)
  - کلید دیگر (Alternate Key)
  - کلید خارجی (Foreign Key)

29

کلیدها

- ابر کلید (Super Key)
  - هر ترکیبی از اسامی صفات رابطه که در هیچ دو تاپل مقدار یکسان نداشته باشد.
- Example: {*customer-name, customer-street*} and {*customer-name*}

  are both superkeys of *Customer*, if no two customers can possibly have the same name.
- کلید کاندید (Candidate Key)
  - هر زیر مجموعه از مجموعه عنوان رابطه که دو خاصیت زیر را داشته باشد، کلید کاندید رابطه است:
    - یکتایی مقدار
    - کاهش ناپذیری. یعنی اگر یکی از عناصر از مجموعه را حذف کنیم زیر مجموعه باقی مانده دیگر خاصیت یکتایی مقدار نداشته باشد.
- نکته: هر S.K. لزوماً C.K. نیست، اما هر C.K. زیر مجموعه S.K. های رابطه است.

30

کلیدها

- کلید اصلی (Primary Key)
  - یکی از کلیدهای کاندید رابطه که طراح انتخاب می کند و به سیستم معرفی می کند.
    - از نظر کاربر، شناسه معمول نوع موجودیت باشد.
    - طول کوتاهتر داشته باشد.
- کلید دیگر (Alternate Key)
  - هر کلید کاندید، غیر از کلید اصلی، کلید دیگر (بدیل) نام دارد.
- کلید خارجی (Foreign Key)
  - دو رابطه R_1 و R_2 (نه لزوماً متمایز) را در نظر می گیریم. هر زیر مجموعه از صفات رابطه R_2 که هر مقدار معلومش با یک مقدار از کلید کاندید R_1 برابر باشد، کلید خارجی در رابطه R_2 است.
  - صفت (صفات) کلید خارجی باید هم میدان با صفت (صفات) کلید کاندید باشد.

31

جبر رابطه ای

- شش عملگر پایه ای دارد
  - select
  - project
  - union
  - set difference
  - Cartesian product
  - rename
- عملگرها یک یا چند رابطه را بعنوان ورودی گرفته و رابطه جدیدی را بعنوان نتیجه تولید می کنند.

32

عملگر Select

- علامت گذاری: $\sigma_p(r)$
- p گزاره انتخابی نامیده می شود.
- تعریف شده است بصورت:

$$\sigma_p(r) = \{t \mid t \in r \text{ and } p(t)\}$$

جایی که p فرمولی در حساب گزاره ای شامل مولفه های متصل شده با (**and**) $\wedge$، (**or**) $\vee$، (**not**) $\neg$ می باشد.

- هر مولفه هست یکی از:

<attribute> *op* <attribute>

<attribute> or <constant>

where *op* is one of: $=, \neq, >, \geq, <, \leq$

- نمونه ای از Select

$$\sigma_{branch\text{-}name=\text{“Perryridge”}}(account)$$

33

Select Operation – Example

- عملگر σ بر روی سطرهای جدول اعمال شده و از بین سطرها، سطرهایی که شرط مشخص شده را داشته باشند در جدول خروجی آورده خواهد شد.

- Relation r

A	B	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

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

A	B	C	D
α	α	1	7
β	β	23	10

34

Project عملگر

- علامت گذاری:

$$\Pi_{A_1, A_2, \ldots, A_k}(r)$$

- جایی که A_1، A_2 و ... نامهای خصوصیت و r نام رابطه میباشند.
- خروجی رابطه ای را با k ستون می باشد که با حذف ستونهایی که لیست شده نیستند بدست می آید.
- در رابطه خروجی سطرهای تکراری حذف می شوند.
- مثال: حذف صفت *branch-name* از *account*

$$\Pi_{account\_number,\ balance}(account)$$

35

Project Operation – Example

- عملگر Π بر روی ستونهای جدول اعمال می شود و می تواند از بین ستونها ستونهای مشخص شده را در جدول خروجی قرار دهد.

- Relation *r*:

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

36

عملگر اجتماع

- علامت گذاری: $r \cup s$
- تعریف:

$$r \cup s = \{t \mid t \in r \text{ or } t \in s\}$$

- برای $r \cup s$ باید
  - r و s باید تعداد صفتهای یکسانی داشته باشند (same arity)
  - صفتهای متناظر از دو رابطه باید سازگار (compatible) باشند.
- مثال: کلیه مشتریها اعم از سپرده گذار یا وام گیرنده

$$\prod_{customer\text{-}name}(depositor) \cup \prod_{customer\text{-}name}(borrower)$$

37

Union Operation – Example

- Relations *r, s:*

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- r ∪ s:

A	B
α	1
α	2
β	1
β	3

38

Set Difference عملگر

- علامتگذاری: $r - s$
- تعریف

$$r - s = \{t \mid t \in r \textbf{ and } t \notin s\}$$

- این عملگر باید مابین رابطه های *سازگار* اعمال شود
  - r و s باید تعداد صفتهای یکسانی داشته باشند (same arity)
  - صفتهای متناظر از دو رابطه باید سازگار (compatible) باشند.

39

Set Difference Operation – Example

- Relations *r*, *s*:

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- *r* – *s*:

A	B
α	1
β	1

40

Cartesian-Product **عملگر**

- علامت گذاری : $r \times s$
- تعریف:

$$r \times s = \{t\, q \mid t \in r \text{ and } q \in s\}$$

- فرض کنید که $r(R)$ و $s(S)$ صفتهای مجزایی باشند ($R \cap S = \varnothing$).
- اگر $r(R)$ و $s(S)$ مجزا نباشند باید از دگر نامی استفاده شود.
- معمولاً برای صفتهای مشترک از عملگر " . " استفاده می شود.

41

Cartesian-Product Operation-Example

- Relations *r*, *s*:

A	B
α	1
β	2

r

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

- *r* x *s*:

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

- در برخی از موارد زمانی که برای بدست آوردن جواب به بیش از یک جدول نیاز داشته باشیم از این عملگر استفاده خواهیم.

42

ترکیب عملگرها

- می توان عباراتی را با استفاده از ترکیب چندین عملگر ایجاد کرد.
- مثال: $\sigma_{A=C}(r \times s)$
- $r \times s$

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

- $\sigma_{A=C}(r \times s)$

A	B	C	D	E
α	1	α	10	a
β	2	β	10	a
β	2	β	20	b

43

عملگر دگر نامی (Rename)

- به ما برای نامیدن، و از این طریق برای ارجاع به نتایج عبارات جبر رابطه ای اجازه می دهد.
- به ما برای ارجاع به یک رابطه با بیش از یک نام اجازه می دهد.
- مثال:

$$\rho_{X}(E)$$

- عبارت E را با نام X بر می گرداند.
- اگر عبارت جبر رابطه ای E تعداد n عنصر داشته باشد، سپس

$$\rho_{x(A_1,A_2,\ldots,A_n)}(E)$$

- نتیجه عبارت E را با نام X ، و با صفات تغییر نام یافته به $A_1, A_2, \ldots, A_n$ بر می گرداند

44

مثال بانکداری

- بعنوان مثال سیستم بانکداری را با بصورت زیر در نظر بگیرید.

branch (branch-name, branch-city, assets)
(سرمایه ، شهر شعبه، نام شعبه) شعبه
customer (customer-name, customer-street, customer-city)
(شهر مشتری، خیابان مشتری، نام مشتری) مشتری
account (account-number, branch-name, balance)
(تراز، نام شعبه، شماره حساب) حساب
loan (loan-number, branch-name, amount)
(میزان وام، نام شعبه، شماره وام) وام
depositor (customer-name, account-number)
(شماره حساب، نام مشتری) سپرده گذار
borrower (customer-name, loan-number)
(شماره وام، نام مشتری) وام گیرنده

45

مثال بانکداری

- ما در این اسلایدها برای کوتاهتر شدن طول مثالها پایگاه داده را بصورت زیر در نظر گرفتیم

brch (BN, bcity, assets)
(سرمایه ، شهر شعبه، نام شعبه) شعبه
cust (CN, cstreet, ccity)
(شهر مشتری، خیابان مشتری، نام مشتری) مشتری
accu (AN, BN, balance)
(تراز، نام شعبه، شماره حساب) حساب
loan (LN, BN, amount)
(میزان وام، نام شعبه، شماره وام) وام
dep (CN, AN)
(شماره حساب، نام مشتری) سپرده گذار
borr (CN, LN)
(شماره وام، نام مشتری) وام گیرنده

46

پرس و جو های نمونه

- تمام وامهای بالای ۱۲۰۰$ را پیدا کنید

$\sigma_{amount > 1200}(loan)$

- شماره وام برای هر وام بالای ۱۲۰۰$ را پیدا کنید

$\Pi_{LN}(\sigma_{amount > 1200}(loan))$

47

پرس و جو های نمونه

- نام تمام مشتریانی را پیدا کنید که وام، حساب یا هر دو را از بانک داشته باشند.

$\Pi_{CN}(borr) \cup \Pi_{CN}(dep)$

- نام مشتریانی را پیدا کنید که هم وام و هم حساب داشته باشند.

$\Pi_{CN}(borr) \cap \Pi_{CN}(dep)$

48

پرس و جو های نمونه

- نام تمام مشتریانی را پیدا کنید که وامی را از شعبه Perryridge دارند.

$\Pi_{CN}(\sigma_{BN="Perryridge"}(\sigma_{borr.LN = loan.LN}(borr \times loan)))$

- نام مشتریانی را پیدا کنید که وامی را از شعبه Perryridge دارند و هیچ حسابی را در شعبه های بانک ندارند.

$\Pi_{CN}(\sigma_{BN = "Perryridge"}(\sigma_{borr.LN = loan.LN}(borr \times loan))) - \Pi_{CN}(dep)$

49

پرس و جو های نمونه

- نام تمام مشتریانی را پیدا کنید که وامی را از شعبه Perryridge دارند.

- Query 1

$\Pi_{CN}(\sigma_{BN = "Perryridge"}(\sigma_{borr.LN = loan.LN}(borr \times loan)))$

- Query 2

$\Pi_{CN}(\sigma_{loan.LN = borr.LN}((\sigma_{BN = "Perryridge"}(loan)) \times borr))$

50

پرس و جو های نمونه

- بالاترین تراز حساب را پیدا کنید.
- رابطه *accu* را با *d* دگر نامی کنید

$\Pi_{balance}(accu) - \Pi_{accu.balance}(\sigma_{accu.balance < d.balance}(accu \times \rho_{d}(accu)))$

51

عملگرهای اضافی

- ما عملگرهای اضافی را تعریف می کنیم که هیچ قابلیتی را به جبر رابطه ای اضافه نمی کنند، بلکه آنها پرس و جو ها را ساده می کنند.
  - Set intersection
  - Natural join
  - Division
  - Assignment

52

عملگر اشتراک (Set-Intersection)

- علامتگذاری: $r \cap s$
- تعریف:

$$r \cap s = \{ t \mid t \in r \textbf{ and } t \in s \}$$

- فرض کنید
  - r و s تعداد صفتهای یکسانی دارند
  - صفتهای r و s سازگار هستند
- توجه: $r \cap s = r - (r - s)$

53

Set-Intersection Operation - Example

- Relation *r, s*:

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- $r \cap s$

A	B
α	2

54

عملگر پیوند طبیعی (Natural-Join)

- علامتگذاری: $r \bowtie s$
- r و s رابطه هایی به ترتیب در شمای R و S می باشند.
- آنگاه $r \bowtie s$ رابطه ای در شمای $R \cup S$ می باشد که از طریق زیر بدست آمده است:
  - هر زوج تاپ t_r از r و t_s از s را در نظر بگیر
  - اگر t_r و t_s مقادیر یکسانی در هر یک از صفات $R \cap S$ داشته باشند، تاپل t را به نتیجه اضافه کن جایی که:
    - t مقدار یکسانی را به عنوان t_r در r داشته باشد
    - t مقدار یکسانی را به عنوان t_s در s داشته باشد
- Example:
  $R = (A, B, C, D)$
  $S = (E, B, D)$
  - Result schema = (A, B, C, D, E)
  - $r \bowtie s$ is defined as:
    $\Pi_{r.A, r.B, r.C, r.D, s.E} (\sigma_{r.B = s.B \wedge r.D = s.D} (r \times s))$

55

Natural Join Operation – Example

- Relations r, s:

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

r

B	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	ϵ

s

- r ⋈ s

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

56

عملگر تقسیم (Division)

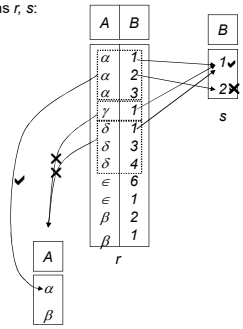
- نمادگذاری: $r \div s$
- برای پرس و جوهایی مناسب است که کلمه "همه" را داشته باشد.
- r و s رابطه هایی به ترتیب در شمای R و S می باشند. جایی که:
- $R = (A_1, \ldots, A_m, B_1, \ldots, B_n)$
- $S = (B_1, \ldots, B_n)$
- نتیجه r ÷ s رابطه ای در شمای $R - S = (A_1, \ldots, A_m)$ می باشد.

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

57

Division Operation – Example

- Relations *r, s*:



- $r \div s$:

A
α
β

58

Another Division Example

- Relations *r, s*:

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

r

D	E
a	1
b	1

s

- $r \div s$:

A	B	C
α	a	γ
γ	a	γ

59

عملگر تقسیم (ادامه)

- مشخصه
  - قرار دهید $q = r \div s$
  - سپس q بزرگترین رابطه ای است که $q \times s \subseteq r$ را برآورده می کند
- تعریف در حوزه عملگرهای پایه ای جبر رابطه ای
  - r(R) و s(S) را رابطه هایی، و $S \subseteq R$ قرار دهید

$$r \div s = \prod_{R-S}(r) - \prod_{R-S}((\prod_{R-S}(r) \times s) - \prod_{R-S,S}(r))$$

60

عملگر انتساب (assignment)

- عملگر انتساب (←) راه مناسبی را برای بیان کردن پرس و جو های پیچیده فراهم می کند.
  - پرس و جو را همانند برنامه های ترتیبی بنویس شامل:
    - سری از انتسابها
    - در پی آنها عبارتی می آید که مقدار آن بعنوان نتیجه پرسو جو می آید.
  - انتساب باید برای ساختن متغیرهای رابطه ای موقت اجازه دهد.
- Example: Write $r \div s$ as

$$temp1 \leftarrow \prod_{R\text{-}S}(r)$$

$$temp2 \leftarrow \prod_{R\text{-}S}((temp1 \times s) - \prod_{R\text{-}S,S}(r))$$

$$result = temp1 - temp2$$

- نتیجه عمارت سمت راست ← در متغیر سمت چپی ← انتساب پیدا می کند.
- ممکن است متغیر در عبارات بعدی استفاده شود.

61

پرس و جو های نمونه

- تمام مشتریانی را پیدا کنید که وام و حسابی را در بانک داشته باشند.

$$\prod_{CN}(borr) \cap \prod_{CN}(dep)$$

- نام مشتریانی که وامی را از بانک دارند همراه با میزان وام

$$\prod_{CN,\ LN,\ amount}(borr \bowtie loan)$$

62

پرس و جو های نمونه

- همه مشتریانی را پیدا کنید که حداقل یک حساب در شعبه های "Downtown" و "Uptown" دارند.

- Query 1

$$\prod_{CN}(\sigma_{BN=\text{"Downtown"}}(dep \bowtie accu)) \cap \prod_{CN}(\sigma_{BN=\text{"Uptown"}}(depo \bowtie accu))$$

- Query 2

$$\prod_{CN,\ BN}(dep \bowtie accu) \div \rho_{temp(BN)}(\{(\text{"Downtown"}), (\text{"Uptown"})\})$$

63

پرس و جو های نمونه

- مشتریانی را پیدا کنید که در همه شعبه هایی که در شهر "Brooklyn" واقعند حسابی را دارند.

$$\Pi_{CN,\ BN}\ (dep \bowtie accu) \div \Pi_{BN}\ (\sigma_{bcity\ =\ \text{"Brooklyn"}}\ (brch))$$

64

عملگرهای جبر رابطه ای گسترش یافته

- عملگر project تعمیم یافته (Generalized Projection)
- توابع جمعی

65

Generalized Projection

- عملگر project را با اجازه دادن به عملگرهای محاسباتی برای استفاده در لیست projection توسعه می دهد.

$$\Pi_{F_1, F_2, \ldots, F_n}(E)$$

- E هر عبارت جبر رابطه ای می باشد.
- هر یک از $F_1, F_2, \ldots, F_n$ عبارات محاسباتی بکار گرفته شده از ثابتها و صفات شمای E می باشند.
- رابطه *credit-info(CN, limit, credit-balance)* را در نظر بگیرید، پیدا کنید هر شخص چقدر می تواند خرج کند

$$\Pi_{CN,\ limit\ -\ credit\text{-}balance}\ (credit\text{-}info)$$

66

Aggregate Functions and Operations

- توابع جمعی (Aggregation function) مقادیری را به عنوان ورودی گرفته و یک نتیجه را به عنوان خروجی برمی گردانند.

avg: average value
min: minimum value
max: maximum value
sum: sum of values
count: number of values

- عملگر جمعی در جبر رابطه ای

$$G_1,G_2,\ldots,G_n\,\mathcal{G}_{F_1(A_1),F_2(A_2),\ldots,F_n(A_n)}(E)$$

- E یک عبارت جبر رابطه ای می باشد
- $G_1, G_2 \ldots, G_n$ لیست صفتهایی می باشند که می خواهیم برای گروه بندی استفاده کنیم (می تواند خالی باشد)
- هر یک از F_i ها یک تابع جمعی می باشند.
- هر یک از A_i ها نام صفتی می باشند.

67

Aggregate Operation – Example

- Relation *r*:

A	B	C
α	α	7
α	β	7
β	β	3
β	β	10

- $\mathcal{G}_{\text{sum(c)}}(r)$

sum-C
27

68

Aggregate Operation – Example

- Relation *accu* grouped by *BN*:

BN	AN	balance
Perryridge	A-102	400
Perryridge	A-201	900
Brighton	A-217	750
Brighton	A-215	750
Redwood	A-222	700

$_{BN}\mathcal{G}_{\textbf{sum}(balance)}(accu)$

BN	balance
Perryridge	1300
Brighton	1500
Redwood	700

69

توابع جمعی

- نتیجه تابع جمعی نام ندارد.
- می توان از عملگر دگر نامی برای دادن نام به آن استفاده کرد.
- برای راحتی، ما دگر نامی را به عنوان بخشی از aggregate operation اجازه می دهیم.

$$BN\ \mathcal{g}\ \textbf{sum}(balance)\ \textbf{as}\ sum\text{-}balance\ (accu)$$

70

Null Values

- ممکن است تاپلها برای برخی صفتهای خود مقادیر تهی داشته باشند، که با *null* مشخص می شود.
- *null* مقدار نامشخص را بیان می کند یا مقداری را بیان می کند که وجود ندارد.
- نتیجه هر عبارت ریاضی که *null* را به کار می برد برابر *null* می باشد.
- توابع جمعی به سادگی از مقادیر تهی چشم پوشی می کنند.
  - تصمیم اختیاری می باشد، می تواند *null* به عنوان نتیجه خروجی برگردانده شود.
  - ما در برخورد با مقادیر *null* از مفهوم SQL پی روی می کنیم.
- برای duplicate elimination و گروه بندی، با null همانند سایر مقادیر برخورد می شود، و دو null فرض می شوند که برابرند.
  - پیشنهاد: فرض کنید هر null با همدیگر متفاوت می باشند.
  - هر دو تصمیم اختیاری می باشند، بنابر این ما به سادگی از SQL پی روی می کنیم.

71

Null Values

- مقایسه با مقادیر *null* مقدار درستی ویژه *unknown* را برمی گرداند
  - اگر *false* بجای *unknown* استفاده شود، سپس not (A<5) برابر نخواهد بود با A>=5
- منطق سه ارزشی از مقدار درستی *unknown* استفاده می کند:
  - OR: (*unknown* **or** *true*) = *true*,
    (*unknown* **or** *false*) = *unknown*
    (*unknown* **or** *unknown*) = *unknown*
  - AND: (*true* **and** *unknown*) = *unknown*,
    (*false* **and** *unknown*) = *false*,
    (*unknown* **and** *unknown*) = *unknown*
  - NOT: (**not** *unknown*) = *unknown*
  - در SQL عبارت "*P* **is unknown**" به true ارزیابی می شود گزاره P به *unknown* ارزیابی شود.
- نتیجه گزاره select بعنوان *false* تلقی می شود اگر آن به *unknown* ارزیابی کند.

72

تغییر پایگاه داده

- محتویات پایگاه داده ممکن است با استفاده از عملگرهای زیر تغییر یابد:
  - Deletion
  - Insertion
  - Updating

73

حذف (Deletion)

- درخواست حذف شبیه یک پرس و جو می باشد با این تفاوت که به جای نمایش تاپلها به کاربر آنها را از پایگاه داده حذف می کند.
- می توان تمام تاپل را حذف کرد، و نمی توان مقادیر را از برخی صفتهای خاص حذف کرد.
- حذف در جبر رابطه ای بصورت زیر بیان می شود

$r \leftarrow r - E$

جایی که r یک رابطه و E یک پرس و جوی جبر رابطه ای می باشند.

74

مثال های حذف

- کلیه رکوردهای مربوط به حسابهای شعبه Perryridge را حذف کنید.

$accu \leftarrow accu - \sigma_{BN = \text{"Perryridge"}}(accu)$

- رکوردهای مربوط به تمام وامهایی که مقدار آنها مابین ۰ تا ۵۰ می باشد را حذف کنید.

$loan \leftarrow loan - \sigma_{amount \geq 0 \text{ and } amount \leq 50}(loan)$

- تمام حسابهایی که در شعبه های واقع در شهر Needham را حذف کنید.

$r_1 \leftarrow \sigma_{bcity = \text{"Needham"}}(accu \bowtie brch)$
$r_2 \leftarrow \Pi_{BN, AN, balance}(r_1)$
$r_3 \leftarrow \Pi_{CN, AN}(r_2 \bowtie dep)$
$accu \leftarrow accu - r_2$
$dep \leftarrow dep - r_3$

75

Insertion

- برای وارد کردن داده در یک رابطه ما:
  - تاپلی را برای وارد کردن مشخص می کنیم.
  - پرس و جویی می نویسیم که نتیجه آن تاپلهایی می باشد که باید وارد شوند.
- در جبر رابطه ای insertion بصورت زیر بیان می شود:

$$r \leftarrow r \cup E$$

جایی که r یک رابطه و E یک عبارت جبر رابطه ای می باشند.

- وارد کردن یک تاپل به اینصورت انجام می شود که شما می توانید به جای E از یک رابطه ثابت (constant relation) شامل یک تاپل استفاده کنید.

76

مثالهای Insertion

- اطلاعاتی را که مشخص می کند Smith در حساب A-937 در شعبه Perryridge ، ۱۲۰۰$ را دارد در پایگاه داده وارد کنید.

$accu \leftarrow accu \cup \{(\text{“Perryridge”, A-973, 1200})\}$
$dep \leftarrow dep \cup \{(\text{“Smith”, A-973})\}$

- برای تمام مشتریان وام در شعبه Perryridge یک حساب ذخیره ۲۰۰$ را به عنوان جایزه در نظر بگیر. شماره وام را به عنوان شماره حساب جدید در نظر بگیر.

$r_1 \leftarrow (\sigma_{BN = \text{“Perryridge”}}(borr \bowtie loan))$
$accu \leftarrow accu \cup \Pi_{BN,\ LN,200}(r_1)$
$dep \leftarrow dep \cup \Pi_{CN,\ LN}(r_1)$

77

Updating

- مکانیسمی برای تغییر تاپلها بدون تغییر *تمام* مقادیر در تاپل می باشد.
- عملگر project تعمیم یافته برای این منظور استفاده می شود.

$$r \leftarrow \Pi_{F_1, F_2, \ldots, F_l}(r)$$

- هر F_i هست یکی از:
- صفت i ام از r ، اگر صفت i ام update شده نیست، یا،
- اگر صفت update شده باشد F_i یک عبارت می باشد که فقط ثابتها و صفات از r را به کار می برد که مقدار جدیدی برای صفت می دهد

78

مثالهای Update

- پرداخت سود را با استفاده افزایش تمام ترازها به میزان ۵٪ انجام دهید.

$$accu \leftarrow \Pi_{AN,\ BN,\ balance\ *\ 1.05}(accu)$$

- برای تمام حسابهای با تراز بالای ۱۰.۰۰۰$، ۶ درصد سود و برای بقیه حسابها ۵ درصد سود وارد کنید.

$$accu \leftarrow \Pi_{AN,\ BN,\ balance\ *\ 1.06}(\sigma_{balance\ >\ 10000}(accu)) \cup \Pi_{AN,\ BN,\ balance\ *\ 1.05}(\sigma_{balance\ \leq\ 10000}(accu))$$

79

فصل سوم

SQL

Structed Query Language

80

Domain Types in SQL

- **char(n).** Fixed length character string, with user-specified length *n*.
- **varchar(n).** Variable length character strings, with user-specified maximum length *n*.
- **int.** Integer (a finite subset of the integers that is machine-dependent).
- **smallint.** Small integer (a machine-dependent subset of the integer domain type).
- **numeric(p,d).** Fixed point number, with user-specified precision of *p* digits, with *n* digits to the right of decimal point.
- **real, double precision.** Floating point and double-precision floating point numbers, with machine-dependent precision.
- **float(n).** Floating point number, with user-specified precision of at least *n* digits.
- Null values are allowed in all the domain types. Declaring an attribute to be **not null** prohibits null values for that attribute.

81

Domain Types in SQL

- **date:** Dates, containing a (4 digit) year, month and date
  - Example: **date** '2005-7-27'
- **time:** Time of day, in hours, minutes and seconds.
  - Example: **time** '09:00:30' **time** '09:00:30.75'
- **timestamp**: date plus time of day
  - Example: **timestamp** '2005-7-27 09:00:30.75'
- **interval:** period of time
  - Example: interval '1' day
  - Subtracting a date/time/timestamp value from another gives an interval value
  - Interval values can be added to date/time/timestamp values
  - Can extract values of individual fields from date/time/timestamp
    - Example: **extract** (**year from** r.starttime)
  - Can cast string types to date/time/timestamp
    - Example: **cast** <string-valued-expression> **as date**
    - Example: **cast** <string-valued-expression> **as time**

82

دستور تعریف میدان

- برای تعریف دامنه از دستور Create Domain استفاده می شود.

Create Domain domain_name datatype
[default-definition]
[domain-constraint-definition-list]

- Default که نوشتن آن اختیاری می باشد برای مشخص کردن مقدار پیش فرض دامنه استفاده می شود.
- constraints نیز که نوشتن آن اختیاری است به منظور اعمال محدودیت به دامنه می باشد .
- مثال:

Create Domain Degree Char(3) Default '???'
Constraints Valid_Degrees
Check Value in ('bs','ms','doc','???')

83

دستور حذف میدان

Drop Domain domain-name option

- با این دستور می توان دامنه تعریف شده را حذف کرد. در این دستور optionمی تواند یکی از عباراتCascade و یاRestrict باشد .
- CASCADE
  - Automatically drop objects that depend on the domain (such as table columns).
- RESTRICT
  - Refuse to drop the domain if there are any dependent objects. This is the default.

84

Create Table دستور

- در SQL برای ایجاد کردن رابطه (جدول) از این دستور استفاده می شود :

Craete Table table_name
{ (ColumnName DataType[**Not Null**][**Unique**]
  [***Default*** defayltOption][***Check*** (search condition)][,…]) }
  [***Primary Key*** (list of Columns),]
  {[***Unique*** (listOfColumns)][,...]}
  {[***Foreign Key*** (listOfForeignKeyColumns)
 Refrences ParentTableName [(ListOfCondidateColumns)],
    [***match*** {**partial** | **full**}]
    [***on update*** RefrentialAction]
    [***on delete*** RefrentialAction]][,...]}
{[***Check*** (SearchCondition)][,…]}

85

Create Table مثال دستور

create table *customer*
   (customer-name char(20),
   customer-street char(30),
   customer-city char(30),
   primary key (*customer-name))*

create table *branch*
   (branch-name char(15),
   branch-city char(30),
   assets integer,
   primary key *(branch-name),*
   check *(assets >= 0))*

86

Create Table مثال دستور

create table *account*
   (account-number char(10),
   branch-name char(15),
   balance integer,
   primary key (*account-number),*
   foreign key (*branch-name)* **references** *branch,*
   check *(balance >= 0))*

create table *depositor*
   (customer-name char(20),
   account-number char(10),
   primary key *(customer-name, account-number),*
   foreign key *(account-number)* **references** *account,*
   foreign key *(customer-name)* **references** *customer)*

87

حذف و تغییر جدول

- **drop table**
  - دستور **drop table** تمام اطلاعات جدول مشخص شده را از پایگاه داده حذف می کند.

- DROP TABLE table_name

88

حذف و تغییر جدول

- **alter table**
  - تغییر نام جدول
    - شکل کلی alter table برای تغییر نام جدول

- ALTER TABLE table_name
  RENAME TO new_table_name;

    - به عنوان مثال

- ALTER TABLE customer
  RENAME TO tbl_cust;

  - افزودن ستون به جدول
    - حالت اول افزودن یک ستون به جدول

- ALTER TABLE table_name
  ADD column_name column-definition;

    - به عنوان مثال

- ALTER TABLE customer
  ADD bdate date;

89

حذف و تغییر جدول

- **alter table**
  - افزودن ستون به جدول
    - حالت دوم افزودن چند ستون به جدول

- ALTER TABLE table_name
- ADD (column_1 column-definition,
- column_2 column-definition,
- ...
- column_n column_definition);

    - به عنوان مثال

- ALTER TABLE customer
- ADD (bdate date,
- description varchar(60));

90

حذف و تغییر جدول

- alter table
  - تغییر ستون
    - ALTER TABLE table_name
      MODIFY column_name column_type;
  - یا
    - ALTER TABLE table_name
    - MODIFY (column_1 column_type,
    - column_2 column_type,
    - ...
    - column_n column_type);
  - مثال
    - ALTER TABLE customer
    - MODIFY (customer_name varchar(30) not null,
    - customer_city varchar(25));

91

حذف و تغییر جدول

- alter table
  - حذف ستون
    - ALTER TABLE table_name
      DROP COLUMN column_name;
  - مثال
    - ALTER TABLE customer
      DROP COLUMN customer_city;
  - تغییر نام ستون
    - ALTER TABLE table_name
      RENAME COLUMN old_name to new_name;
  - مثال
    - ALTER TABLE customer
      RENAME COLUMN customer_name to CN;

92

حذف و تغییر جدول

```
ALTER TABLE TableName1
   ADD | ALTER [COLUMN] FieldName1
      FieldType [(nFieldWidth [, nPrecision])]
      [NULL | NOT NULL]
      [CHECK lExpression1 [ERROR cMessageText1]]
      [DEFAULT eExpression1]
      [PRIMARY KEY | UNIQUE]
      [REFERENCES TableName2 [TAG TagName1]]
      [NOCPTRANS]
 - Or –
ALTER TABLE TableName1
   ALTER [COLUMN] FieldName2
      [NULL | NOT NULL]
      [SET DEFAULT eExpression2]
      [SET CHECK lExpression2 [ERROR cMessageText2]]
      [DROP DEFAULT]
      [DROP CHECK]
```

```
- Or –
ALTER TABLE TableName1
   [DROP [COLUMN] FieldName3]
   [SET CHECK lExpression3 [ERROR cMessageText3]]
   [DROP CHECK]
   [ADD PRIMARY KEY eExpression3 TAG TagName2]
   [DROP PRIMARY KEY]
   [ADD UNIQUE eExpression4 [TAG TagName3]]
   [DROP UNIQUE TAG TagName4]
   [ADD FOREIGN KEY [eExpression5] TAG TagName4
      REFERENCES TableName2 [TAG TagName5]]
   [DROP FOREIGN KEY TAG TagName6 [SAVE]]
   [RENAME COLUMN FieldName4 TO FieldName5]
   [NOVALIDATE]
```

93

ساختار دستور Select

- پرس و جوهای نمونه SQL فرم زیر را دارند

select $A_1, A_2, ..., A_n$
from $r_1, r_2, ..., r_m$
where P

  - A_i ها نشان دهنده صفات
  - r_i ها نشان دهنده رابطه ها
  - P یک گزاره شرطی می باشد
- این پرس و جو معادل یک عبارت جبر رابطه ای می باشد.
- نتیجه یک پرس و جو جبر رابطه ای یک رابطه می باشد.

$$\Pi_{A_1,A_2,...,A_n}(\sigma_P(r_1 \times r_2 \times ... \times r_m))$$

94

بخش Select

- بخش select مشخص کننده صفاتی می باشد که در حاصل پرس و جو خواهد آمد.
- همانند عملگر project جبر رابطه ای
- مثال: نام تمام شعبه ها را در رابطه loan پیدا کنید

select *BN*
from *loan*

- دستور جبر رابطه ای معادل آن عبارت زیر خواهد بود

$\Pi_{BN}(loan)$

- توجه: SQL اجازه کاراکتر '-' را در نام نمی دهد
  - به عنوان مثال هنگام پیاده سازی از عبارت branch_name به جای عبارت branch-name استفاده کنید.
- توجه: نامهای SQL حساس به حالت حروف کوچک و بزرگ نمی باشد.

95

بخش Select (ادامه)

- SQL اجازه مقادیر تکراری را در نتایج پرس و جو می دهد
- برای مجبور کردن به حذف تکرار، می توان از کلمه کلیدی *distinct* بعد ار Select استفاده کرد.
- نام تمام شعبه ها را در رابطه loan پیدا کنید، و تکرار ها را حذف کنید.

select distinct *BN*
from *loan*

- کلمه کلیدی *All* مشخص می کند که تکرار ها حذف نشوند.

select all *BN*
from *loan*

96

بخش Select (ادامه)

- یک ستار در بخش select تمام صفتها را مشخص می کند

```
select *
from loan
```

- بخش select می تواند شامل عبارات محاسباتی باشد که از عملیاتهای +، –، *، و / باشد، که بر روی مقادیر ثابت یا صفتهای تاپل اعمال می شوند.
- پرسو جوی
- رابطه ای را همانند *loan* به استثنای *amount* که در ۱۰۰ ضرب می شود.

```
select LN, BN, amount * 100
from loan
```

97

بخش where

- بخش where مشخص کننده شرطی می باشد که نتیجه باید داشته باشد، همانند عملگر select در جبر رابطه ای.
- برای پیدا کردن تمام شماره وامهایی که از شعبه Perryridge پرداخت شده است و میزان آنها بیشتر از ۱۲۰۰$ می باشد.

```
select LN
from loan
where BN = 'Perryridge' and amount > 1200
```

- نتایج جستجو می تواند با استفاده از اتصالهای منطقی and، or و not با همدیگر ترکیب شوند.

98

بخش where (ادامه)

- SQL همچنین شامل عملگر مقایسه ای *between* می باشد.
- به عنوان مثال: شماره تمام وامهایی را پیدا کنید که میزان آنها مابین ۹۰۰۰۰$ و ۱۰۰۰۰۰$ می باشد.

```
select LN
from loan
where amount between 90000 and 100000
```

99

بخش from

- بخش *from* لیست رابطه هایی را خواهیم آورد در پرس و جو می آیند.
- همانند عملگر ضرب کارتزین جبر رابطه ای عمل می کند.
- مثال: حاصلضرب کارتزین *borr×loan*

```
select *
from borr, loan
```

- مثال: نام مشتری، شماره وام، و میزان وام تمام مشتریانی را پیدا کنید که وامی را از شعبه Perryridge داشته باشند.

```
select CN, borr.LN, amount
    from borr, loan
    where   borr.LN = loan.LN  and
            BN = 'Perryridge'
```

100

عملیات دگرنامی (The Rename Operation)

- SQL با استفاده از قید ***as*** اجازه دگر نامی رابطه ها و صفات را می دهد.

old-name **as** *new-name*

- نام مشتری، شماره وام و میزان وام تمام مشتریان را پیدا کنید، نام loan-id را برای ستون loan-number قرار دهید.

```
select CN, borr.LN as loan-id, amount
   from borr, loan
   where borr.LN = loan.LN
```

101

عملیات دگرنامی (The Rename Operation)

- تغییر نام جداول در بخش from با استفاده از قید as وجود دارد.
- نام و شماره وام را برای تمام مشتریانی پیدا کنید که وامی را از شعبه ای داشته باشند.

```
select CN, T.LN, S.amount
      from borr as T, loan as S
      where  T.LN = S.LN
```

- نام تمام شعبه هایی را پیدا کنید که سرمایه بیشتری از شعبه های واقع در Brooklen داشته باشند.

```
select distinct T.BN
      from brch as T, brch as S
      where T.assets > S.assets and S.bcity = 'Brooklyn'
            and T.bcity<>'Brooklyn'
```

102

عملیات رشته ای

- SQL شامل عملگرهای رشته ای برای مقایسه رشته های کاراکتری می باشد. عملگر "like" الگویی را استفاده می کند که با استفاده از کراکتر خاص توضیح داده شده است.
  - percent (%). The % character matches any substring.
  - underscore (_). The _ character matches any character.
- نام تمام مشتریانی را پیدا کنید که نام خیابانشان زیر رشته "Main" را داشته باشد.

```
select customer-name
from customer
where customer-street like '%Main%'
```

- Match the name "Main%"

```
like 'Main\%' escape '\'
```

- SQL تعداد متنوعی از عملگرهای رشته ای مانند:
  - concatenation (using "||")
  - converting from upper to lower case (and vice versa)
  - finding string length, extracting substrings, etc.

103

مرتب سازی نمایش تاپلها

- لیست مرتب نام مشتریانی که وامی از شعبه Perryridge دارند به ترتیب حروف الفبا

```
select distinct CN
from   borr, loan
where borr.LN = loan.LN and
       BN = 'Perryridge'
order by CN
```

- ما ممکن است ***desc*** را برای ترتیب نزولی یا ***asc*** را برای ترتیب صعودی برای هر صفت تعیین کنیم، ترتیب صعودی حالت پیش فرض می باشد.
  - Example: **order by** *CN* **desc**

104

عملگرهای مجموعه ای

- عملگرهای مجموعه ای ***union***، ***intersect*** و ***exept*** بر روی رابطه های اعمال می شوند و معادل عملگر های $-$,$\cap$,$\cup$ جبر رابطه ای می باشند.
- هر یک از عملگرهای بالایی بصورت اتوماتیک تکرار ها را حذف می کنند، برای بر گرداندن تمام تکرارها از دستورات معادل ***union all***، ***intersect all*** و ***except all*** استفاده کنید.
- فرض کنید تاپلی m بار در r و n بار در s اتفاق افتاده باشد، سپس:
  - $m + n$ times in r **union all** s
  - min(m,n) times in r **intersect all** s
  - max(0, $m – n$) times in r **except all** s

105

عملگرهای مجموعه ای

- Find all customers who have a loan, an account, or both:

  (**select** *CN* **from** *dep*)
  union
  (**select** *CN* **from** *borr)*

- Find all customers who have both a loan and an account.

  (**select** *CN* **from** *dep*)
  intersect
  (**select** *CN* **from** *borr)*

- Find all customers who have an account but no loan.

  (**select** *CN* **from** *dep*)
  except
  (**select** *CN* **from** *borr)*

106

توابع جمعی

- این توابع بر روی مقادیر ستونی از یک رابطه اعمال می شوند، و مقداری را برمی گردانند:

  avg: average value
  min: minimum value
  max: maximum value
  sum: sum of values
  count: number of values

107

توابع جمعی

- Find the average account balance at the Perryridge branch.

  select avg *(balance)*
  from *accu*
  where *BN* = 'Perryridge'

- Find the number of tuples in the *customer* relation.

  select count (*)
  from *cust*

- Find the number of depositors in the bank.

  select count (distinct *CN)*
  from *dep*

108

توابع جمعی – group by

- تعداد سپرده گذارها را در هر شعبه محاسبه کنید:

```
select BN, count (distinct CN)
        from dep, accu
        where dep.AN = accu.AN
        group by BN
```

- ***توجه***: صفات بخش group by حتماً باید در خروجی select آورده شوند.
- ***توجه***: صفات غیر از صفات گروه بندی حتماً باید با توابع جمعی آورده شوند.

109

توابع جمعی – بخش having

- نام تمام شعبه هایی را پیدا کنید که میانگین تراز حسابهای آنها بیش از ۱۲۰۰$ می باشد.

```
select BN, avg (balance)
        from accu
        group by BN
        having avg (balance) > 1200
```

- ***نکته***: عبارت بخش having بعد از شکل دهی گروها اعمال می شود و عبارت بخش where قبل از شکل دهی گروه ها اعمال می شود.

110

مقادیر null

- ممکن است برخی تاپلها مقادیر null برای برخی صفتهای خود داشته باشند، که با *null* مشخص می شود.
- Null مشخص کننده مقدار نامشخص می باشد ، یا نشان می دهد که مقدار وجود ندارد.
- گزاره is null می تواند برای چک کردن مقادیر استفاده شود،
- مثال: تمام شماره وامهایی را پیدا کنید که در رابطه loan دارای مقدار null برای amount باشد.

```
select LN
from loan
where amount is null
```

- نتیجه تمام عبارات محاسباتی که درگیر null شود برابر null خواهد بود.
- Example: 5 + null returns null
- هرچند، توابع جمعی به سادگی از مقدار null چشم پوشی می کنند.

111

مقادیر null و منطق سه سطحی

- مقایسه با *null* مقدار *unknown* را بر می گردد:
- Example: *5 < null or null <> null or null = null*
- منطق سه ارزشی از ارزش درستی unknown استفاده می کند.
- OR: (*unknown* **or** *true*) = *true*, (*unknown* **or** *false*) = *unknown*
  (*unknown* **or** *unknown*) = *unknown*
- AND: *(true* **and** *unknown)* = *unknown, (false* **and** *unknown)* = *false,*
  (unknown **and** *unknown)* = *unknown*
- NOT: *(***not** *unknown)* = *unknown*
- "*P* **is unknown"** evaluates to true if predicate *P* evaluates to *unknown*
- نتیجه بخش where به عنوان false تلقی می شود اگر به unknown ارزیابی شود.

112

مقدار null و توابع جمعی

- مجموع تمام میزان وامها

select sum (*amount)*
from *loan*

- دستور بالا از مقادیر null چشم پوشی می کند.
- مقدار برابر null خواهد بود اگر هیچ مقدار غیر null وجود نداشته باشد.
- تمام توابع جمعی به غیر از count(*) از تاپلها با مقادیر null بر روی صفات توابع جمعی چشم پوشی می کنند.

113

Subquery های تو در تو

- SQL مکانیزمی را برای استفاده از subquery ها فراهم می کند.
- یک زیر پرس جو عبارت select-from-where می باشد که در درون پرسو جوی دیگر قرار داده می شود.
- یک استفاده عمومی از subquery ها برای شکل دادن تستها عضویت مجموعه، مقایسه مجموعه ها، و cardinality مجموعه ها می باشد.

114

پرس و جوی نمونه

- تمام مشتریانی را پیدا کنید که حساب و وامی را در بانک داشته باشند.

```
select distinct CN
    from borr
    where CN in (select CN
                 from dep)
```

- تمام مشتریانی را پیدا کنید که وامی را از بانک داشته اما حسابی در بانک نداشته باشند.

```
select distinct CN
    from borr
    where CN not in (select CN
                     from dep)
```

115

پرس و جوی نمونه

- تمام مشتریانی را پیدا کنید که هم حساب و هم وام در شعبه Perryridge داشته باشند.

```
select distinct CN
    from borr, loan
    where borr.LN = loan.LN and
          BN = "Perryridge" and
          (BN, CN) in
              (select BN, CN
              from dep, accu
              where dep.AN = accu.AN)
```

116

پرس و جوی نمونه

- روش دیگر

```
select distinct CN
        from borr, loan
        where borr.LN = loan.LN and
          BN = "Perryridge" and
            CN in
              (select CN
              from dep, accu
              where dep.AN =  accu.AN and BN = "Perryridge" )
```

117

تعریف عبارت some

- F <comp> **some** $r \Leftrightarrow \exists t \in r$ *s.t. (*F <comp> *t*)
  Where <comp> can be: $<, \leq, \geq, >, =, \neq$

*(5< **some*** | 0 / 5 / 6 | *) = true*

(read: 5 < some tuple in the relation)

*(5< **some*** | 0 / 5 | *) = false*

*(5 = **some*** | 0 / 5 | *) = true*

*(5 ≠ **some*** | 0 / 5 | *) = true (since 0 ≠ 5)*

*(= **some**) ≡ **in***
*However, (≠ **some**) ≢ **not in***

مقایسه مجموعه ها

- تمام شعبه هایی را پیدا کنید که سرمایه بیشتری را از برخی شعبه های واقع در Brooklyn داشته باشند.

```
select distinct  T.BN
    from branch as T, branch as S
    where  T.assets > S.assets and
           S.bcity = 'Brooklyn' and T.bcity<>'Brooklyn'
```

- پرس و جوی یکسان بااستفاده از عبارت > some

```
select BN
    from brch
    where bcity <> 'Brooklyn' and
          assets > some (select assets
                         from brch
                         where bcity = 'Brooklyn')
```


تعریف عبارت All

- F <comp> all $r \Leftrightarrow \forall t \in r$ (F <comp> *t)*
  Where <comp> can be: $<, \leq, \geq, >, =, \neq$

*(5< **all*** | 0 / 5 / 6 | *) = false*

*(5< **all*** | 6 / 10 | *) = true*

*(5 = **all*** | 4 / 5 | *) = false*

*(5 ≠ **all*** | 4 / 6 | *) = true (since 5 ≠ 4 and 5 ≠ 6)*

*(≠ **all**) ≡ **not in***
*However, (= **all**) ≢ **in***

پرس و جوی نمونه

- شعبه هایی را پیدا کنید که سرمایه بیشتری را از تمام شعبه های واقع در Brooklyn داشته باشند.

```
select BN
    from brch
    where assets > all
            (select assets
            from brch
            where bcity = 'Brooklyn')
```

121

تست برای رابطه های خالی

- عبارت **exists** مقدار **true** را برمی گرداند اگر subquery آرگومان آن خالی نباشد.
- **exists** $r \Leftrightarrow r \neq \emptyset$
- **not exists** $r \Leftrightarrow r = \emptyset$

122

پرس و جوی نمونه

- تمام مشتریانی را پیدا کنید که حسابی را در تمام شعبه های شهر Brooklen داشته باشند.

```
select distinct S.CN
    from dep as S
    where not exists (
            (select BN
            from brch
            where bcity = 'Brooklyn')
        except
            (select R.BN
            from dep as T, accu as R
            where T.AN = R.AN and
                S.CN = T.CN))
```

- Note that $X - Y = \emptyset \Leftrightarrow X \subseteq Y$
- *Note:* Cannot write this query using = **all** and its variants

123

تست کردن برای عدم وجود تاپلهای تکراری

- عبارت **unique** تست می کند که آیا subquery تاپل تکراری در نتایج خود دارد یا نه.
- تمام مشتریانی را پیدا کنید که حداکثر یک حساب در شعبه Perryridge داشته باشند.

```
select T.CN
from dep as T
where unique (
    select R.CN
    from accu, dep as R
    where T.CN = R.CN and
          R.AN = accu.AN and
          accu.BN = 'Perryridge')
```

124

پرس و جوی نمونه

- تمام مشتریانی را پیدا کنید که بیش از یک حساب در شعبه Perryridge داشته باشند.

```
select T.CN
from dep as T
where not unique (
    select R.CN
    from accu, dep as R
    where T.CN = R.CN and
          R.AN = accu.AN and
          accu.BN = 'Perryridge')
```

125

تغییر داده های پایگاه داده – insertion

- یک تاپل جدید به account اضافه کنید

```
insert into accu
    values ('A-9732', 'Perryridge',1200)
```

یا حالت معادل

```
insert into accu (BN, balance, AN)
    values ('Perryridge', 1200, 'A-9732')
```

- یک تاپل جدید به account با balance برابر با null اضافه کنید

```
insert into account
    values ('A-777','Perryridge', null)
```

126

تغییر داده های پایگاه داده – insertion

- یک حساب ذخیره ۲۰۰$ برای تمام مشتریان وام شعبه Perryridge به عنوان جایزه باز کنید. شماره وام را به عنوان شماره حساب ذخیره جدید در نظر بگیرید.

```
insert into accu
    select LN, BN, 200
    from loan
    where BN = 'Perryridge'
insert into dep
    select CN, loan.LN
    from loan, borr
    where BN = 'Perryridge'
        and loan.LN = borr.LN
```

- بخش select from where بصورت کامل قبل از وارد کردن مقادیر به رابطه محاسبه می شود (در غیر اینصورت پرس و جوهایی شبیه
- **insert into** *table1* **select** * **from** *table1*
- ممکن است مسئله ایجاد کند).

127

تغییر داده های پایگاه داده – update

- برای تمام حسابها با تراز بالای $10000 ، ۶٪ و برای سایر حسابها ۵٪ سود اضافه کنید.

```
update accu
set balance = balance * 1.06
where balance > 10000

update accu
set balance = balance * 1.05
where balance ≤ 10000
```

- The order is important

128

تغییر داده های پایگاه داده – Delete

- تمام حسابهای شعبه Perryridge را حذف کنید

```
delete from accu
where BN = 'Perryridge'
```

- تمام حسابهایی که در شعبه های شهر Needham واقع شده اند را حذف کنید.

```
delete from accu
where BN in (select BN
             from brch
             where bcity = 'Needham')
```

129

فصل چهارم

Entity-Relationship Model

(ER)

130

مدلسازی

- در روش ER سه مفهوم معنایی وجود دارد و معنای داده های هر محیطی به کمک همین سه مفهوم نمایش داده می شود که عبارتند از:
  - نوع موجودیت
  - صفت
  - نوع ارتباط
- یک پایگاه داده می تواند مدل شود بصورت:
  - مجموعه ای از موجودیتها (entities)
  - رابطه مابین موجودیتها

131

موجودیت (entity)

- عبارتست از مفهوم کلی ‘‘شی’’ ، ‘‘چیز’’ ، ‘‘پدیده’’ و بطور کلی هر آن چه که می خواهیم در موردش ‘‘اطلاع’’ داشته باشیم و شناخت خود را در موردش افزایش دهیم ، اعم از اینکه وجود فیزیکی یا ذهنی داشته باشد. هر نوع موجودیت از نظر کاربر نام و معنای مشخصی دارد.
- باید سه ضابطه ی زیر را در تشخیص موجودیت ها در نظر بگیریم.
  - یک نوع موجودیت از یک محیط، معمولاً نمونه هایی (بیش از یک نمونه) متمایز از یکدیگر دارد.
  - یک نوع موجودیت معمولاً بیش از یک صفت دارد و کاربر به مجموعه ای از اطلاعات در مورد آن نیاز دارد.
  - معمولاً حالت کنشگری (فاعلیت) یا کنش پذیری (مفعولیت) دارد.
- یک *entity set* یک مجموعه از entity های هم نوع می باشد که صفات یکسانی دارند.
  - Example: set of all persons, companies, trees, holidays

132

Entity Sets *customer* and *loan*

customer_id	customer_name	customer_street	customer_city
321-12-3123	Jones	Main	Harrison
019-28-3746	Smith	North	Rye
677-89-9011	Hayes	Main	Harrison
555-55-5555	Jackson	Dupont	Woodside
244-66-8800	Curry	North	Rye
963-96-3963	Williams	Nassau	Princeton
335-57-7991	Adams	Spring	Pittsfield

loan_number	amount
L-17	1000
L-23	2000
L-15	1500
L-14	1500
L-19	500
L-11	900
L-16	1300

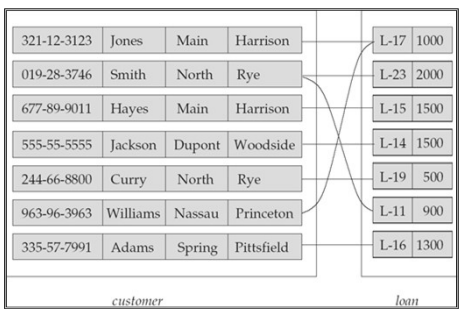
133

ارتباط (Relationship)

- معمولا ما بین موجودیت ها ارتباطاتی وجود دارد نوع ارتباط عبارتست از تعامل ما بین n موجودیت(n>=۱) و ماهیتا نوعی بستگی بین انواع موجودیت ها می باشد.
- هر نوع ارتباط یک معنای مشخص دارد و با یک نام بیان می شود، همچنین می توان گفت نوع ارتباط عملی است که بین انواع موجودیت های جاری بوده، است و یا خواهد بود.

134

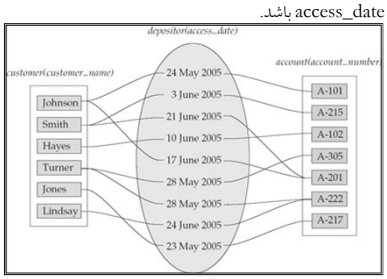
ارتباط *borrower*



135

ارتباط

- می توان برای ارتباط ها نیز صفت در نظر گرفت.
- به عنوان مثال ارتباط depositor مابین customer و account می تواند شامل صفت access_date باشد.



136

درجه ارتباط

- معرف تعداد موجودیهای شرکت کننده در ارتباط می باشد.
- ارتباطی که در آن دو موجودیت شرکت داشته باشند از نوع باینری (درجه دو) خواهد بود.
- اکثر ارتباطات از درجه دو می باشند.

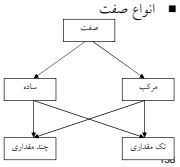
137

صفت (Attribute)

- صفت در واقع خصیصه یا ویژگی یک نوع موجودیت است و هر نوع موجودیت مجموعه ای از صفات دارد که حالت یا وضعیت آن را توصیف میکند.
- یک موجودیت با استفاده از مجموعه ی از صفات نشان داده می شود، که توصیف کننده خصوصیاتی می باشد که تمام موجودیتها دارا می باشند.

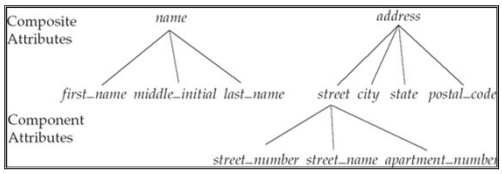
- Example:
- *customer* = (*customer_id, customer_name, customer_street, customer_city*)
  loan = (*loan_number, amount*)

- دامنه (domain) مجموعهای از مقادیر مجاز برای هر صفت می باشد.
- انواع صفت
  - *Simple* and *composite* attributes.
  - *Single-valued* and *multi-valued* attributes
    - Example: multivalued attribute: *phone_numbers*
  - *Derived* attributes
    - Can be computed from other attributes
    - Example: age, given date_of_birth



138

صفت مرکب



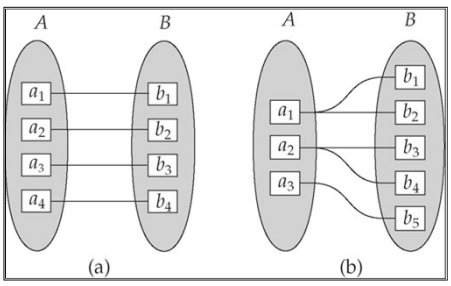
139

Cardinality ارتباط

- بیان کننده تعداد موجودیتهایی می باشد که می تواند با استفاده از ارتباط با موجودیت دیگر در ارتباط باشد.
- اکثراً در ارتباط باینری مفید می باشد.
- برای ارتباط باینری cardinality می تواند یکی از موارد زیر باشد
  - One to one
  - One to many
  - Many to one
  - Many to many

140

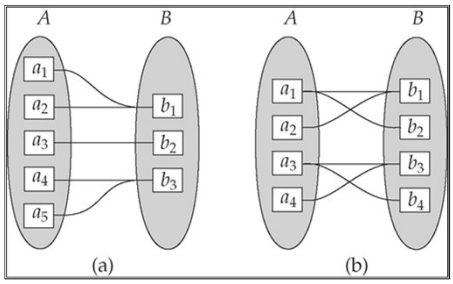
Mapping Cardinalities



One to one One to many

141

Mapping Cardinalities



Many to one Many to many

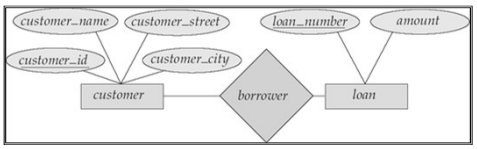
142

کلیدها

- **super key**
  - هر زیر مجموعه از صفات رابطه که یکتایی مقدار بدنه رابطه داشته باشد.
- **candidate key**
  - هر زیر مجموعه از مجموعه عنوان رابطه که دو خاصیت زیر را داشته باشد، کلید کاندید رابطه است:
    - یکتایی مقدار
    - کاهش ناپذیری
- **primary key**
  - یکی از کلیدهای کاندید رابطه که طراح انتخاب می کند
    - ظابطه های انتخاب عبارتند از:
      - از نظر کاربر، شناسه معمول نوع موجودیت باشد.
      - طول کوتاهتر داشته باشد.

143

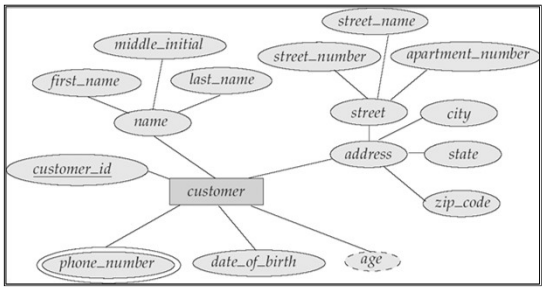
نمودار ER



- مستطیل نشان دهنده موجودیت (entity) می باشد.
- لوزی نشان دهنده ارتباط (relationship) می باشد.
- خطوط صفات را به موجودیتها و موجودیتها را به ارتباط ها متصل می کند.
- بیضی نشان دهنده صفت ها (Attributes) می باشد.
  - بیضی تودرتو نشان دهنده صفت چند مقداری می باشد.
  - بیضی خطچین نشاندهنده صفت مشتق (derived) می باشد.
- صفت زیر خط دار نشان دهنده کلید اصلی می باشد.

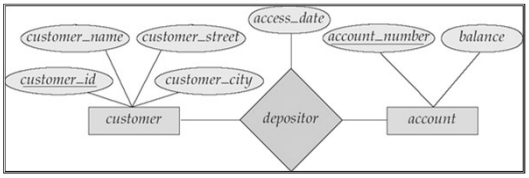
144

E-R Diagram With Composite, Multivalued, and Derived Attributes



145

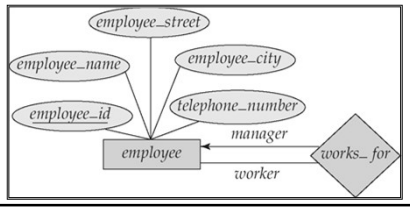
Relationship Sets with Attributes



146

نقش ها (Roles)

- موجودیتهای مربوط به یک ارتباط لزوماً نباید از هم مجزا باشند.
- برچسبهای "manager" و "worker" نقش نامیده می شوند. آنها مشخص می کنند که موجودیت کارمند چه نقشی را در رابطه works_for بازی می کنند.
- نقشها در نمودار ER بوسیله برچسب زدن بر خطی که لوزی را به مستطیل وصل می کند مشخص می شوند.
- برچسب نقشها اختیاری می باشند و برای توضیح دادن مفهوم ارتباط استفاده می شوند.



147

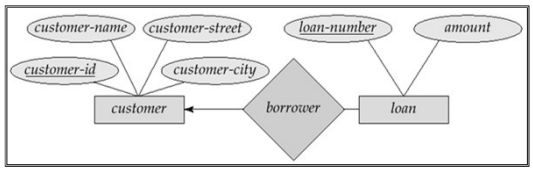
Cardinality Constraints

- ما می توانیم قید cardinality را با رسم خط جهت دار (→) که مشخص کننده "one" یا خط غیر جهتدار (—) که مشخص کننده "many" می باشد بین ارتباط و موجودیت نشان دهیم.
- ارتباط one-to-one
  - A customer is associated with at most one loan via the relationship borrower
  - A loan is associated with at most one customer via borrower

148

One-To-Many Relationship

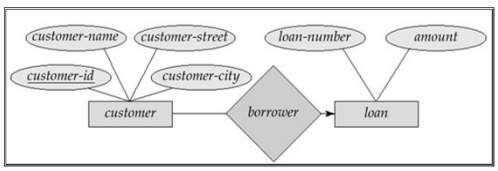
- در ارتباط یک به چند یک وام حداکثر با یک مشتری در ارتباط می باشد، و یک مشتری با چندین وام از طریق *borrower* در ارتباط می باشد.



149

Many-To-One Relationships

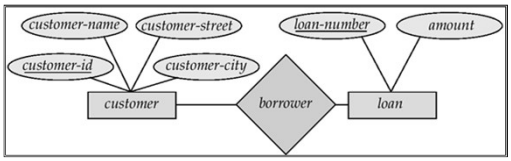
- در ارتباط چند به یک، یک وام با چندین مشتری از طریق *borrower* در ارتباط می باشد، و یک مشتری حداکثر با یک وام در ارتباط می باشد.



150

Many-To-Many Relationship

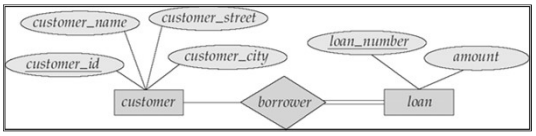
- یک وام با چندین مشتری از طریق *borrower* در ارتباط می باشد،
- یک مشتری با چندین وام از طریق *borrower* در ارتباط می باشد.



151

شرکت موجودیت در ارتباط

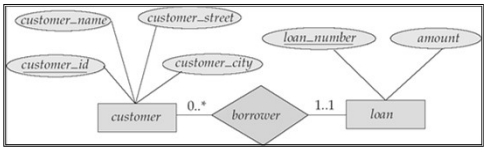
- شرکت الزامی (با دوخط مشخص می شود): هر نمونه از موجودیت مورد نظر باید در حداقل یک ارتباط شرکت کند.
  - بعنوان مثال شرکت *loan* در *borrower* الزامی می باشد.
- شرکت اختیاری: بعضی نمونه از موجودیت می تواند در هیچ ارتباطی شرکت نکند.
  - بعنوان مثال شرکت *customer* در *borrower* اختیاری می باشد.



152

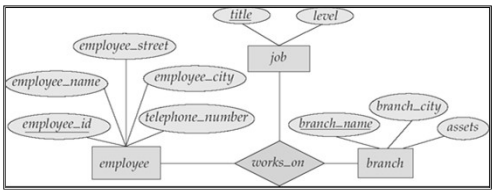
نماد جایگزین برای نمایش محدودیتهای cardinality

- محدودیتهای کاردینالیتی همچنین می تواند با محدودیتهای مشارکت نشان داده شود.



153

نمودار E-R با ارتباط درجه سه



154

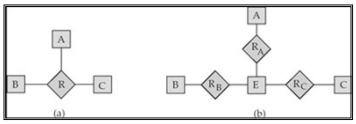
محدودیتهای cardinality ارتباط درجه سه

- ما حداکثر یک پیکان را اجازه می دهیم که از ارتباط درجه سه (یا درجه های بالاتر) برای نمایش محدودیتهای کاردینالیتی خارج شود.
- بعوان مثال یک پیکان از *work_on* به *job* نشان می دهد که هر *employee* در حداکثر یک *job* در هر شعبه کار می کند.
- اگر در آنجا بیش از یک پیکان باشد، آنجا دو راه برای تعریف معنای آن وجود دارد.
  - بعنوان مثال، یک ارتباط درجه سه R بین A، B و C با پیکانهایی به B و C می تواند معنا دهد:
    1. هر موجودیت A با موجودیت واحدی از B و C در ارتباط است یا
    2. هر زوج از موجودیتهای (A, B) با موجودیت واحدی از موجودیت C در ارتباط می باشد، و هر زوج (A, C) با موجودیت واحدی از B در ارتباط می باشد.

155

تبدیل ارتباطات۳ به درجه ۲

- به طور کلی هر ارتباط غیر درجه ۲ با ایجاد موجودیت ساختگی قابل تبدیل به ارتباط درجه ۲ می باشد
  - R مابین A، B و C را با موجودیت E و سه ارتباط جایگزین کنید.
    - 1. R_A, relating E and A
    - 2. R_B, relating E and B
    - 3. R_C, relating E and C
  - صفتهای شاخصی را برای E ایجاد کنید.
  - برای هر ارتباط (a_i, b_j, c_j) در R ایجاد کن
    ۱. نمونه جدید e_i از موجودیت E ۲. اضافه کن (e_i, a_i) را به R_A
    ۳. اضافه کن (e_i, b_j) را به R_B ٤. اضافه کن (e_i, c_j) را به R_C



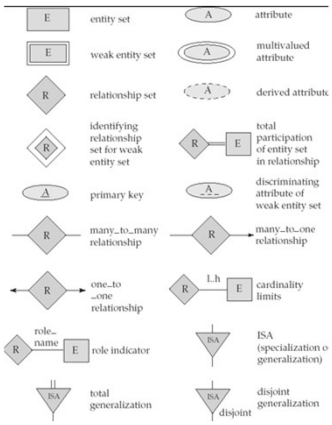
156

موجودیت ضعیف

- موجودیتی که کلید اصلی نداشته باشد

- An entity set that does not have a primary key is referred to as a **weak entity set**.
- The existence of a weak entity set depends on the existence of a **identifying entity set**
  - it must relate to the identifying entity set via a total, one-to-many relationship set from the identifying to the weak entity set
  - Identifying relationship depicted using a double diamond
- The **discriminator** *(or partial key)* of a weak entity set is the set of attributes that distinguishes among all the entities of a weak entity set.
- The primary key of a weak entity set is formed by the primary key of the strong entity set on which the weak entity set is existence dependent, plus the weak entity set's discriminator.

157



نمادهای مورد استفاده در نمودار ER

158

فصل پنجم

طراحی پایگاه داده های رابطه ای

(روش بالا به پایین)

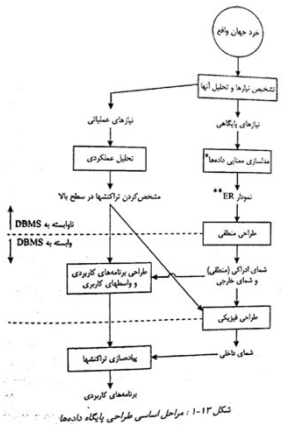
159

مقدمه

- مراحل اساسی طراحی پایگاه داده ها
  1. مطالعه و شناخت خرد جهان واقع
  2. انجام مهندسی نیازها
  3. مدلسازی معنایی داده ها
  4. **طراحی منطقی پایگاه داده ها**
  5. طراحی فیزیکی پایگاه داده ها
  6. انجام تحلیل عملکرد : تعیین تراکنشها
  7. طراحی برنامه های کاربردی

160

مقدمه (ادامه)



شکل ۱۳-۱ : مراحل اساسی طراحی پایگاه داده‌ها

161

مقدمه (ادامه)

- برای طراحی منطقی پایگاه داده ها دو روش وجود دارد:
  - روش بالا به پایین (top-down design method)
  - روش سنتز رابطه ای (Relational synthesis)
  - روش ترکیبی

162

تبدیل مدلسازی به طراحی منطقی
(روش بالا به پایین)

- فرض ما بر این است که مراحل قبل انجام شده است یعنی
  - شناخت خرد جهان واقع و انجام مهندسی نیازها
  - مدلسازی معنایی داده ها
- نکته: کار طراحی ماهیت هنری هم دارد و طراحی نوعی هنر است. در این کار ای بسا ظرافت و دقایق فنی وجود دارد

163

تبدیل مدلسازی به طراحی منطقی
(روش بالا به پایین)

روش تبدیل نمودار ER به رابطه ها

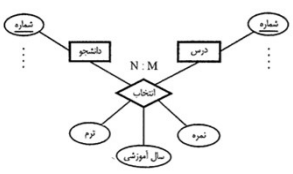
- حالت اول
  - تعداد نوع موجودیت : $n \geq 2$
  - وضع موجودیت ها : مستقل
  - چندی ارتباط : N : M
- $n+1$ ارتباط نیاز است
  - یک رابطه برای هر یک از n موجودیت مستقل
  - یک رابطه برای نمایش ارتباط بین آنها

164

تبدیل مدلسازی به طراحی منطقی
(روش بالا به پایین)

روش تبدیل نمودار ER به رابطه ها

- مثال حالت اول
- رابطه های این مثال
  - STT
  - COT
  - STCOT



STT($\underset{C.K}{\underline{STID}}$, ……)

COT($\underset{C.K}{\underline{COID}}$, ……)

STCOT($\underset{F.K}{\underline{STID}}$, $\underset{F.K}{\underline{COID}}$, ……)

$\underset{C.K}{\quad}$

165

تبدیل مدلسازی به طراحی منطقی
(روش بالا به پایین)

روش تبدیل نمودار ER به رابطه ها

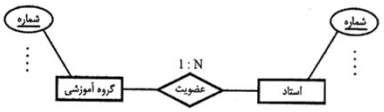
- حالت دوم
  - تعداد نوع موجودیت : دو
  - وضع موجودیت ها : مستقل
  - چندی ارتباط : 1 : N
- دو رابطه کفایت می کند
- یک رابطه برای نمایش نوع موجودیت طرف یک
- یک رابطه برای نمایش نوع موجودیت طرف N و نیز ارتباط 1:N
- کلید اصلی رابطه اول به عنوان کلید خارجی به این رابطه اضافه می شود.

166

تبدیل مدلسازی به طراحی منطقی
(روش بالا به پایین)

روش تبدیل نمودار ER به رابطه ها

- مثال حالت دوم



DEPT(DEID , DTITLE , ...)
PROF(PRID , PRNAME , ... , DEID)

167

تبدیل مدلسازی به طراحی منطقی
(روش بالا به پایین)

روش تبدیل نمودار ER به رابطه ها

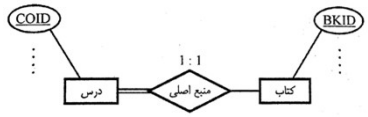
- حالت سوم
  - تعداد نوع موجودیت : دو
  - وضع موجودیت ها : مستقل
  - چندی ارتباط : 1 : 1
- دو رابطه لازم است
  - یک رابطه برای نمایش یکی از دو نوع موجودیت
  - یک رابطه برای نمایش موجودیت دیگر و ارتباط بین دو نوع موجودیت
- می توان با یک رابطه نیز نشان داد وقتی مشارکت هر دو نوع موجودیت در ارتباط الزامی باشد و تعداد صفات موجودیت های شرکت کننده در ارتباط زیاد نباشد.

168

تبدیل مدلسازی به طراحی منطقی
(روش بالا به پایین)

روش تبدیل نمودار ER به رابطه ها

- مثال حالت سوم



BOOK(BKID , BKTITLE , ...)

COT(COID , COTITLE , ... , BKID)

OR

COBK(COID , COTITLE , ... , BKID , BKTITLE , ...)

169

تبدیل مدلسازی به طراحی منطقی
(روش بالا به پایین)

روش تبدیل نمودار ER به رابطه ها

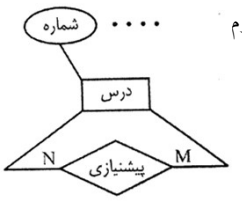
- حالت چهارم
  - تعداد نوع موجودیت : یک
  - وضع موجودیت ها : مستقل
  - چندی ارتباط : N : M
- حالت خاص حالت اول است
- دو رابطه لازم است
  - یک رابطه برای نمایش خود موجودیت
  - یک رابطه برای نمایش ارتباط (الزامی باشد یا نباشد)

170

تبدیل مدلسازی به طراحی منطقی
(روش بالا به پایین)

روش تبدیل نمودار ER به رابطه ها

- مثال حالت چهارم



COT(COID , TITLE , ...)

PREREQ(COID , PRCOID)

171

تبدیل مدلسازی به طراحی منطقی
(روش بالا به پایین)

روش تبدیل نمودار ER به رابطه ها

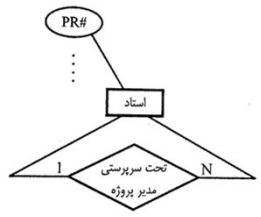
- حالت پنجم
  - تعداد نوع موجودیت : یک
  - وضع موجودیت ها : مستقل
  - چندی ارتباط : 1 : N
- حالت خاص حالت دوم است
- یک رابطه کفایت می کند

172

تبدیل مدلسازی به طراحی منطقی
(روش بالا به پایین)

روش تبدیل نمودار ER به رابطه ها

- مثال حالت پنجم



PRJMGR *(PRID , ... , PRMGRID)*
PK FK

173

تبدیل مدلسازی به طراحی منطقی
(روش بالا به پایین)

روش تبدیل نمودار ER به رابطه ها

- حالت ششم
  - تعداد نوع موجودیت : یک
  - وضع موجودیت ها : مستقل
  - چندی ارتباط : 1 : 1
- حالت خاص حالت سوم است
- یک رابطه کفایت می کند، به شرط اینکه مشارکت در ارتباط الزامی باشد
- می توان با دو رابطه هم طراحی کرد

174

تبدیل مدلسازی به طراحی منطقی
(روش بالا به پایین)

روش تبدیل نمودار ER به رابطه ها

- مثال حالت ششم

شماره دانشجویی هم تیم

- طراحی با یک رابطه

STUD(STID , STNAME , ... , PJSTID)
C.K. F.K.

- و یا

STPJST(STID , STNAME , ... , PJSTID , SSTNAME , ...)
C.K. F.K.

- طراحی با دو رابطه

STUD(STID , STNAME , ...)
C.K.

STOJCOST(STID , PJSTID)
F.K. F.K.
C.K.

175

تبدیل مدلسازی به طراحی منطقی
(روش بالا به پایین)

روش تبدیل نمودار ER به رابطه ها

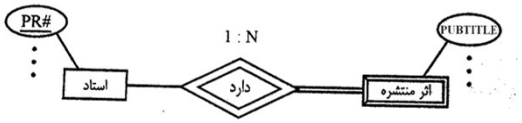
- حالت هفتم : نمایش موجودیت ضعیف
  - موجودیت ضعیف دارای شناسه یکتا نیست بلکه صفت ممیزه دارد
- برای نمایش این نوع موجودیت یک رابطه طراحی می کنیم
  - کلید کاندید موجودیت قوی به عنوان کلید خارجی اضافه می شود
  - کلید کاندید این رابطه از ترکیب کلید کاندید موجودیت قوی و صفت ممیزه بدست می آید

176

تبدیل مدلسازی به طراحی منطقی
(روش بالا به پایین)

روش تبدیل نمودار ER به رابطه ها

- مثال حالت هفتم



PRPUB(PRID , PUBTITLE , ...)
F.K.
C.K.

177

تبدیل مدلسازی به طراحی منطقی
(روش بالا به پایین)

روش تبدیل نمودار ER به رابطه ها

- حالت هشتم : وجود صفت چند مقداری
- اگر MVA ، یک صفت چند مقداری ، EID شناسه نوع موجودیت E و A_1 ، A_2 ، ... ، A_i سایر صفات تک مقداری (ساده یا مرکب) موجودیت E باشند
- برای نمایش این موجودیت دو رابطه نیاز است

R1(EID , A_1 , A_2 , ... , A_i)
C.K

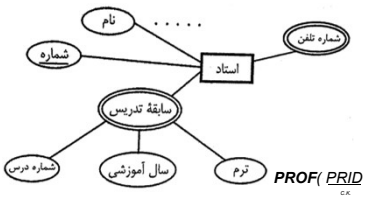
R2(EID , MVA)
F.K. C.K

178

تبدیل مدلسازی به طراحی منطقی
(روش بالا به پایین)

روش تبدیل نمودار ER به رابطه ها

- مثال حالت هشتم



PROF(PRID , PRNAME , ...)
C.K

PRTEACHIS(PRID , COID , TR , YRYR)
F.K. C.K

PRTEL(PRID , TEL)
F.K. C.K

179

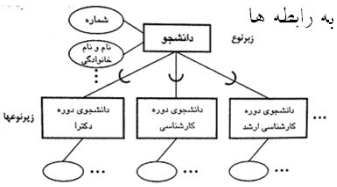
تبدیل مدلسازی به طراحی منطقی
(روش بالا به پایین)

روش تبدیل نمودار ER به رابطه ها

- حالت نهم : ارتباط IS-A یا " گونه ای است از ..."
- یک روش این است که موجودیت سطح بالاتر با یک رابطه نشان داده می شود و هر نوع موجودیت سطح پایین تر با صفات خاص خودش همراه با شناسه اصلی موجودیت سطح بالاتر، با رابطه دیگر نمایش داده می شود.

180

تبدیل مدلسازی به طراحی منطقی
(روش بالا به پایین)



روش تبدیل نمودار ER به رابطه ها

- مثال حالت نهم

STUD(*STID* , *STNAME* , ...)
C.K.

BSSTUD(*STID* , ...)
C.K. ,F.K.

MSSTUD(*STID* , ...)
C.K. ,F.K.

DOCSTUD(*STID* , ...)
C.K. ,F.K.

181

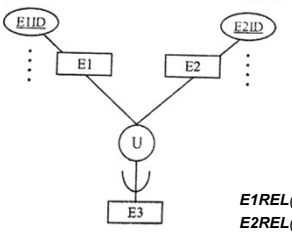
تبدیل مدلسازی به طراحی منطقی
(روش بالا به پایین)

روش تبدیل نمودار ER به رابطه ها

- حالت دهم : وجود دسته (طبقه) در ارتباط IS-A
  - وقتی صفات شناسه زیر نوعها متفاوت باشند، زیر نوع را با رابطه نشان می دهیم.
    - کلید این رابطه یک کلید ساختگی است.
    - بعلاوه یک صفت دیگر در آن در نظر می گیریم که نشان دهنده نوع آن است (از نوع کدام زیر نوع است).
  - هر زیر نوع هم با یک رابطه نشان می دهیم و صفت کلید زیر نوع را به هر یک از رابطه های نشان دهنده زیر نوعها اضافه می کنیم.

182

تبدیل مدلسازی به طراحی منطقی
(روش بالا به پایین)



روش تبدیل نمودار ER به رابطه ها

- مثال حالت دهم

E1REL(*E1ID* , ... , *E3ID*)
E2REL(*E2ID* , ... , *E3ID*)
E3REL(*E3ID* , *E3TYPE*)

183

تبدیل مدلسازی به طراحی منطقی
(روش بالا به پایین)

روش تبدیل نمودار ER به رابطه ها

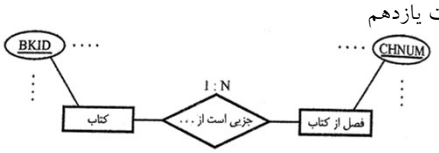
- حالت یازدهم : ارتباط IS-A PART OF یا "جزئی است از ..."
- برای موجود کل یک رابطه های و برای هر یک از موجودیتهای جزء نیز یک رابطه طراحی می کنیم.
  - در رابطه نشان دهنده موجودیت جزء ، کلید کاندید رابطه نشان دهنده موجودیت کل، آورده می شود.

184

تبدیل مدلسازی به طراحی منطقی
(روش بالا به پایین)

روش تبدیل نمودار ER به رابطه ها

- مثال حالت یازدهم



BOOK(BKID , *BKTITLE* , ...)
C.K.

CHAP(BKID , *CHNUM* , *CHAPTITLE* , *NO-OF-PAGE* , ...)
C.K.

185

تبدیل مدلسازی به طراحی منطقی
(روش بالا به پایین)

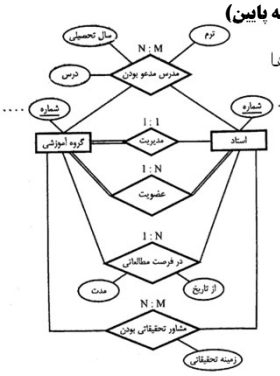
روش تبدیل نمودار ER به رابطه ها

- حالت دوازدهم : بیش از یک ارتباط بین دو موجودیت
  - تعداد رابطه ها بستگی به چندی هر ارتباط دارد
  - بصورت زیر عمل می کنیم
    - هر نوع موجودیت مستقل شرکت کننده در یک ارتباط با چندی N:M با یک رابطه نشان داده می شود.
    - هر نوع ارتباط N:M را با یک رابطه نشان می دهیم.
    - هر یک از ارتباطهای با چندی 1:N، اگر مشارکت دو نوع موجودیت در همه ارتباطها الزامی باشد، را می توان با یک کلید خارجی در رابطه نشان دهنده نوع موجودیت طرف (N) ، نشان داد.
    - اگر مشارکت دو موجودیت در یک ارتباط 1:N، الزامی نباشد بهتر است برای چنین ارتباطی یک رابطه جداگانه طراحی شود که صفات آن، شناسه دو نوع موجودیت و صفات خود ارتباط ، در صورت وجود ، هستند.

186

تبدیل مدلسازی به طراحی منطقی

(روش بالا به پایین)



روش تبدیل نمودار ER به رابطه ها

- مثال حالت دوازدهم

برای ارتباط "مدیریت"

DEPT(DEID , DETITLE , ... , PRID)
C.K. F.K.

برای ارتباط "عضویت"

PROF(PRID , PRNAME , ... , DEID)
C.K. F.K.

برای ارتباط "فرصت مطالعاتی"

PROF-IN-SAB(PRID , BEGDATE , DURATION , DEID)
C.K. F.K.

PROF-INVIT(PRID , DEID , TR , YRYR , COID)
C.K.

187

188