پایگاه داده ها

SQL

SQL

- زبانی برای بازیابی و مدیریت داده‌ها در سیستم‌های مدیریت پایگاه داده‌های رابطه‌ای
- اولین نسخه از SQL توسط Chamberlin و Boyce در IBM توسعه داده شد
  - نام اولیه
  - SEQUEL (Structured English QUEry Language)

SQL

- زبان SQL شامل چندین قسمت است
  - زبان تعریف داده‌ها (DDL)
    - تعریف، اصلاح و حذف شِماهای رابطه
    - توصیف قیود صحت
    - تعریف دیدها
    - توصیف حقوق دستیابی به رابطه‌ها و دیدها
  - زبان عملیات روی داده‌ها (DML)
    - بیان پرس و جوها
    - درج، اصلاح و حذف چندگانه‌ها از پایگاه داده‌ها
  - کنترل تراکنش‌ها

تعریف داده‌ها

- نوع‌های دامنه
  - char(*n*)
  - varchar(*n*)
  - int
  - smallint
  - numeric(*p*, *d*)
  - real, double precision
  - float(*n*)

تعریف داده‌ها

- **تعریف یک رابطه**

create table r (A_1 D_1, A_2 D_2, ..., A_n D_n,
[integrity-constraint$_1$],
...,
[integrity-constraint$_k$])

- **r نام رابطه**
- **هر A_i نام یک خصیصه در شمای رابطه r**
- **هر D_i نوع مقادیر در دامنه خَصیصه A_i**
- **مثال**

create table *branch*
(*branch_name* **char**(15) **not null**,
branch_city **char**(30),
assets **numeric**(16,2))

تعریف داده‌ها

- قیود صحت
  - کلید اصلی
    - primary key (A_{j_1}, ..., A_{j_m})
  - مثال

create table *branch*
(*branch_name* **char**(15),
branch_city **char**(30),
assets **numeric**(16,2),
primary key (*branch_name*))

تعریف داده‌ها

- مثال
  - شِمای سیستم بانکی

customer **(**<u>***customer_name***</u>**,** ***customer_street***, ***customer_city***)
branch **(**<u>***branch_name***</u>**,** ***branch_city***, ***assets***)
loan **(**<u>***loan_number***</u>**,** ***branch_name***, ***amount***)
borrower **(**<u>***customer_name***</u>**,** <u>***loan_number***</u>**)**
account **(**<u>***account_number***</u>**,** ***branch_name***, ***balance***)
depositor **(**<u>***customer_name***</u>**,** <u>***account_number***</u>**)**

create table *customer*
(*customer_name* **char**(20),
customer_street **char**(30),
customer_city **char**(30),
primary key (*customer_name*))

تعریف داده‌ها

create table *branch*
(*branch_name* **char**(15),
branch_city **char**(30),
assets **numeric**(16, 2),
primary key (*branch_name*))

create table *account*
(*account_number* **char**(10),
branch_name **char**(15),
balance **numeric**(12, 2),
primary key (*account_number*))

create table *depositor*
(*customer_name* **char**(20),
account_number **char**(10),
primary key (*customer_name*, *account_number*))

تعریف داده‌ها

- **حذف یک رابطه**

  drop table *r*

- **افزودن خصیصه‌های جدید به یک رابطه**

  alter table *r* **add** *A D*

  - *A* نام خصیصه
  - *D* دامنه خصیصه

- **حذف خصیصه‌ها از یک رابطه**

  alter table *r* **drop** *A*

- بسیاری از سیستم‌های پایگاه داده‌ای از حذف خصیصه‌ها پشتیبانی نمی‌کنند

ساختار پرس و جوهای SQL

- شکل کلی پرس و جوها

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

  - هر A_i نام یک خصیصه
  - هر r_i نام یک رابطه
  - p شرطی است که باید برآورده شود

- عبارت جبر رابطه‌ای

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

- نتیجه هر پرس و جوی SQL یک رابطه است

ساختار پرس و جوهای SQL

- عبارت **select** با عمل پرتو از جبر رابطه‌ای مطابقت می‌کند
  - مثال
    - نام‌های همه شعبه‌های بانکی در رابطه ***loan*** را فهرست کنید

select *branch_name*
from *loan*

- SQL اجازه می‌دهد چندگانه‌های تکراری در رابطه‌های حاصل از ارزیابی عبارات **select** وجود داشته باشند
- مثال
  - برای حذف نام‌های تکراری از کلمه کلیدی **distinct** استفاده می‌شود

select distinct *branch_name*
from *loan*

- عبارت جبر رابطه‌ای

$$\Pi_{branch\_name}(loan)$$

ساختار پرس و جوهای SQL

- نماد * در عبارت select "همه خصیصه‌ها" را نمایش می‌دهد
  - مثال

```
select *
from loan
```

- عبارت select می‌تواند شامل عبارات حسابی باشد
  - مثال

```
select loan_number, branch_name, amount * 100
from loan
```

- رابطه حاصل مشابه با رابطه ***loan*** است با این تفاوت که مقدار خصیصه ***amount*** در ۱۰۰ ضرب شده است

ساختار پرس و جوهای SQL

- عبارت **where** با شرط انتخاب از جبر رابطه‌ای مطابقت می‌کند
- مثال
  - شماره وام‌هایی که از شعبه **Perryridge** دریافت شده‌اند و میزان آن‌ها از ۱۲۰۰ دلار بیشتر است را فهرست کنید

```
select loan_number
from loan
where branch_name = 'Perryridge' and amount>1200
```

  - شماره وام‌هایی که میزان آن‌ها بین ۹۰۰۰۰ تا ۱۰۰۰۰۰ دلار است را فهرست کنید

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

ساختار پرس و جوهای SQL

- عبارت **from** با عمل ضرب دکارتی از جبر رابطه‌ای مطابقت می‌کند
- مثال
  - نام‌های مشتریانی که وام بانکی دریافت کرده‌اند را به همراه شماره وام و میزان وام آن‌ها فهرست کنید

select *customer_name*, *borrower.loan_number*, *amount*
from *borrower*, *loan*
where *borrower.loan_number = loan.loan_number*

  - عبارت جبر رابطه‌ای

$$\Pi_{customer\_name,\ loan\_number,\ amount}\ (borrower \bowtie loan)$$

ساختار پرس و جوهای SQL

- **مثال**
  - **نام‌های مشتریانی که از شعبه Perryridge وام بانکی دریافت کرده‌اند را به همراه شماره وام و میزان وام آن‌ها فهرست کنید**

select *customer_name*, *borrower.loan_number*, *amount*
from *borrower*, *loan*
where *borrower.loan_number* = *loan.loan_number* **and**
branch_name = ‘Perryridge’

ساختار پرس و جوهای SQL

- SQL امکان نام‌گذاری مجدد رابطه‌ها و خصیصه‌ها را با استفاده از عبارت as فراهم می‌کند

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

- مثال
  - نام‌های مشتریانی که وام بانکی دریافت کرده‌اند را به همراه شماره وام و میزان وام آن‌ها فهرست کنید. نام خصیصه ***loan_number*** را به ***loan_id*** تغییر دهید

select *customer_name*, *borrower.loan_number* **as** *loan_id*, *amount*
from *borrower*, *loan*
where *borrower.loan_number = loan.loan_number*

ساختار پرس و جوهای SQL

- از عبارت as می‌توان برای تعریف متغیرهای چندگانه استفاده کرد
  - مثال
    - نام‌های مشتریانی که وام بانکی دریافت کرده‌اند را به همراه شماره وام و میزان وام آن‌ها فهرست کنید

```
select customer_name, T.loan_number, S.amount
from borrower as T, loan as S
where T.loan_number = S.loan_number
```

ساختار پرس و جوهای SQL

- مثال
  - نام‌های شعبه‌هایی را فهرست کنید که سرمایه آن‌ها از سرمایه حداقل یکی از شعبه‌های مستقر در شهر **Brooklyn** بیشتر است

select distinct *T.branch_name*
from *branch* **as** *T*, *branch* **as** *S*
where *T.assets* > *S.assets* **and** *S.branch_city* = ‘Brooklyn’

  - فرض کنید رابطه ***branch*** به صورت زیر داده شده باشد

branch_name	*branch_city*	*assets*
Brighton	Brooklyn	7100000
Downtown	Brooklyn	9000000
North Town	Rye	3700000
Perryridge	Horseneck	1700000
Round Hill	Horseneck	18000000

ساختار پرس و جوهای SQL

- پرس و جوی زیر را در نظر بگیرید

select *
from *branch* **as** *T*, *branch* **as** *S*

T.branch_name	*T.branch_city*	*T.assets*	*S.branch_name*	*S.branch_city*	*S.assets*
Brighton	Brooklyn	7100000	Brighton	Brooklyn	7100000
Downtown	Brooklyn	9000000	Brighton	Brooklyn	7100000
North Town	Rye	3700000	Brighton	Brooklyn	7100000
Perryridge	Horseneck	1700000	Brighton	Brooklyn	7100000
Round Hill	Horseneck	18000000	Brighton	Brooklyn	7100000
Brighton	Brooklyn	7100000	Downtown	Brooklyn	9000000
Downtown	Brooklyn	9000000	Downtown	Brooklyn	9000000
North Town	Rye	3700000	Downtown	Brooklyn	9000000
Perryridge	Horseneck	1700000	Downtown	Brooklyn	9000000
Round Hill	Horseneck	18000000	Downtown	Brooklyn	9000000
Brighton	Brooklyn	7100000	North Town	Rye	3700000
Downtown	Brooklyn	9000000	North Town	Rye	3700000
North Town	Rye	3700000	North Town	Rye	3700000
Perryridge	Horseneck	1700000	North Town	Rye	3700000
Round Hill	Horseneck	18000000	North Town	Rye	3700000
Brighton	Brooklyn	7100000	Perryridge	Horseneck	1700000
Downtown	Brooklyn	9000000	Perryridge	Horseneck	1700000
North Town	Rye	3700000	Perryridge	Horseneck	1700000
Perryridge	Horseneck	1700000	Perryridge	Horseneck	1700000
Round Hill	Horseneck	18000000	Perryridge	Horseneck	1700000
Brighton	Brooklyn	7100000	Round Hill	Horseneck	18000000
Downtown	Brooklyn	9000000	Round Hill	Horseneck	18000000
North Town	Rye	3700000	Round Hill	Horseneck	18000000
Perryridge	Horseneck	1700000	Round Hill	Horseneck	18000000
Round Hill	Horseneck	18000000	Round Hill	Horseneck	18000000

ساختار پرس و جوهای SQL

- پرس و جوهای زیر را در نظر بگیرید

select *
from *branch* **as** *T*, *branch* **as** *S*
where *T.assets* > *S.assets* **and** *S.branch_city* = ‘Brooklyn’

T.branch_name	*T.branch_city*	*T.assets*	*S.branch_name*	*S.branch_city*	*S.assets*
Downtown	Brooklyn	9000000	Brighton	Brooklyn	7100000
Round Hill	Horseneck	18000000	Brighton	Brooklyn	7100000
Round Hill	Horseneck	18000000	Downtown	Brooklyn	9000000

select *T.branch_name*
from *branch* **as** *T*, *branch* **as** *S*
where *T.assets* > *S.assets* **and** *S.branch_city* = ‘Brooklyn’

branch_name
Downtown
Round Hill
Round Hill

ساختار پرس و جوهای SQL

- پرس و جوی زیر را در نظر بگیرید

select distinct *T.branch_name*
from *branch* **as** *T*, *branch* **as** *S*
where *T.assets* > *S.assets* **and** *S.branch_city* = ‘Brooklyn’

branch_name
Downtown
Round Hill

ساختار پرس و جوهای SQL

- از عملگر like می‌توان برای مقایسه بین دو رشته کاراکتری استفاده کرد
  - الگوهای مورد استفاده توسط این عملگر با استفاده از دو کاراکتر خاص توصیف می‌شوند
    - کاراکتر % با هر زیررشته دلخواه مطابقت می‌کند
    - کاراکتر _ با هر کاراکتر دلخواه مطابقت می‌کند
  - مثال
    - 'Perry%'
      - با رشته‌هایی که با Perry آغاز می‌شوند مطابقت می‌کند
    - '___'
      - با رشته‌هایی به طول دقیقاً سه کاراکتر مطابقت می‌کند

ساختار پرس و جوهای SQL

- مثال
  - نام‌های مشتریانی را فهرست کنید که نام خیابان محل سکونت آن‌ها شامل زیررشته Main باشد

```
select customer_name
from customer
where customer_street like '% Main%'
```

ساختار پرس و جوهای SQL

- از عبارت **order by** می‌توان برای مرتب‌سازی چندگانه‌های حاصل از یک پرس و جو استفاده کرد
  - مثال
    - نام‌های مشتریانی که از شعبه **Perryridge** وام بانکی دریافت کرده‌اند را به ترتیب الفبایی فهرست کنید

```
select distinct customer_name
from borrower, loan
where borrower.loan_number = loan.loan_number and
      branch_name = 'Perryridge'
order by customer_name
```

- برای مشخص کردن ترتیب مرتب‌سازی به صورت صعودی یا نزولی از **asc** یا **desc** استفاده می‌شود

ساختار پرس و جوهای SQL

- عمل اجتماع
  - مثال
    - نام‌های مشتریانی را فهرست کنید که حساب بانکی و یا وام بانکی دارند

(**select** *customer_name*
from *depositor*)
union
(**select** *customer_name*
from *borrower*)

- عمل **union** به طور خودکار چندگانه‌های تکراری را حذف می‌کند
- برای حفظ چندگانه‌های تکراری از **union all** استفاده می‌شود
- مشابه با عمل اجتماع (**union**) می‌توان از عمل اشتراک (**intersect**) و عمل تفاضل (**except**) استفاده کرد

ساختار پرس و جوهای SQL

- توابع تجمیع
  - مجموعه‌ای از مقادیر را به عنوان ورودی دریافت می‌کنند و یک مقدار واحد را به عنوان خروجی برگشت می‌دهند
  - در SQL می‌توان از توابع تجمیع avg، min، max، sum و count استفاده کرد
  - مثال
    - متوسط موجودی حساب‌های شعبه Perryridge را مشخص کنید

select avg (*balance*)
from *account*
where *branch_name* = ‘Perryridge’

ساختار پرس و جوهای SQL

- برای ایجاد گروه‌هایی از چندگانه‌ها و اعمال توابع تجمیع برروی هر گروه از عبارت **group by** استفاده می‌شود
- مثال
  - متوسط موجودی حساب‌های هرکدام از شعبه‌های بانک را مشخص کنید

select *branch_name*, **avg** (*balance*)
from *account*
group by *branch_name*

- تنها خصیصه‌های موجود در فهرست **group by** و خصیصه‌های حاصل از توابع تجمیع می‌توانند در عبارت **select** ظاهر شوند

ساختار پرس و جوهای SQL

- مثال
  - تعداد سپرده‌گذاران هرکدام از شعبه‌های بانک را مشخص کنید

```
select branch_name, count (distinct customer_name)
from depositor, account
where depositor.account_number = account.account_number
group by branch_name
```

- گاهی اوقات لازم است شرطی را روی گروه‌های ساخته شده توسط عبارت group by اعمال کرد
  - مثال

```
select branch_name, avg (balance)
from account
group by branch_name
having avg (balance) > 1200
```

ساختار پرس و جوهای SQL

- در صورتی که هر دو عبارت where و having در پرس و جوی یکسانی ظاهر شوند
  - ابتدا چندگانه‌هایی که شرط مشخص شده توسط where را برآورده می‌کنند توسط عبارت group by گروه‌بندی می‌شوند
  - سپس عبارت having روی هرکدام از گروه‌ها اعمال می‌شود
    - گروه‌هایی که شرط مشخص شده توسط having را برآورده نمی‌کنند حذف می‌شوند
  - در نهایت عبارت select از گروه‌های باقیمانده برای تولید چندگانه‌های حاصل از پرس و جو استفاده می‌کند

ساختار پرس و جوهای SQL

- مثال
  - متوسط موجودی حساب‌های هرکدام از مشتریانی که در شهر Harrison زندگی می‌کنند و حداقل سه شماره حساب دارند را مشخص کنید

```
select depositor.customer_name, avg (balance)
from depositor, account, customer
where depositor.account_number = account.account_number and
      depositor.customer_name = customer.customer_name and
      customer_city = 'Harrison'
group by depositor.customer_name
having count (distinct depositor.account_number) >= 3
```

ساختار پرس و جوهای SQL

- مثال
  - تعداد چندگانه‌های رابطه ***customer*** را مشخص کنید

select count (*)
from *customer*

  - متوسط موجودی همه حساب‌های بانکی را مشخص کنید

select avg (*balance*)
from *account*