Database Chapter Three Outline

**SQL**

**SQL names are case insensitive** 不区分大小写

**Data Definition Language**

**Create Table Construct**

An SQL relation is defined using the **create table** command:

**create table** *r* (*A*1 *D*1, *A*2 *D*2, ..., *An Dn,* (integrity-constraint1),  
 ...,  
 (integrity-constraintk))

*r* is the name of the relation

each *Ai* is an attribute name in the schema of relation *r*

*Di* is the data type of values in the domain of attribute *Ai*

**Integrity Constraints in Create Table**

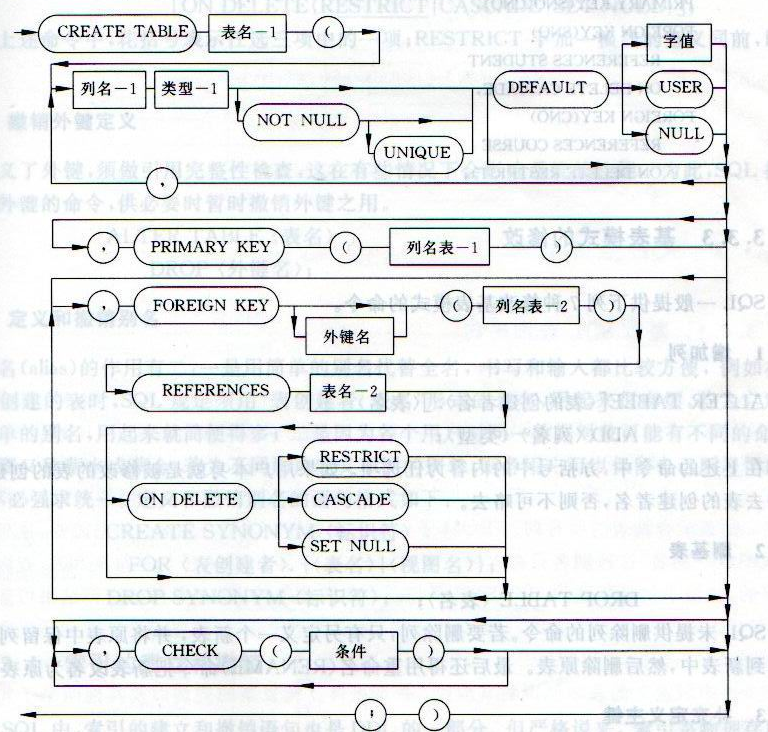
**not null**

**primary key** (*A*1, ..., *An* )

Example: Declare *branch\_name* as the primary key for *branch* and ensure that the values of *assets* are non-negative.

**create table** *branch* (*branch\_name* char(15)**,** *branch\_city* char(30),  
 *assets* integer,  
 **primary key** (*branch\_name*),

**check**(assets>=0))



PRIMARY KEY -> FOREIGN KEY -> CHECK

**Drop and Alter Table Constructs**

**alter table** *r* **add** *A D 添加属性*

**alter table** *r* **drop** *A 删除属性*

*Drop table r 删除表*

**Basic Query Structure**

A typical SQL query has the form：  
 **select** *A*1, *A*2, ..., *An*  
 **from** *r*1, *r*2, ..., *rm*  
 **where** *P*

*Ai* represents an attribute

*Ri* represents a relation

*P* is a predicate.

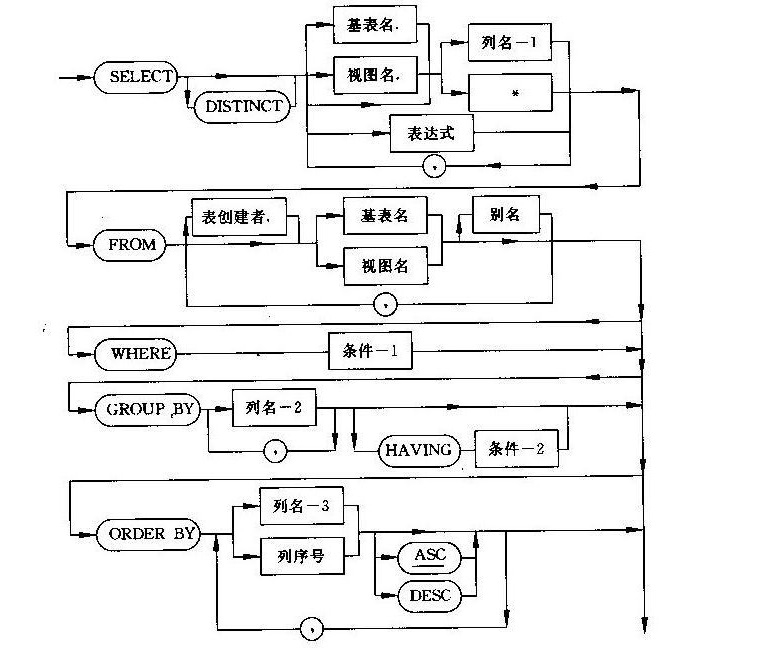
This query is equivalent to the relational algebra expression.

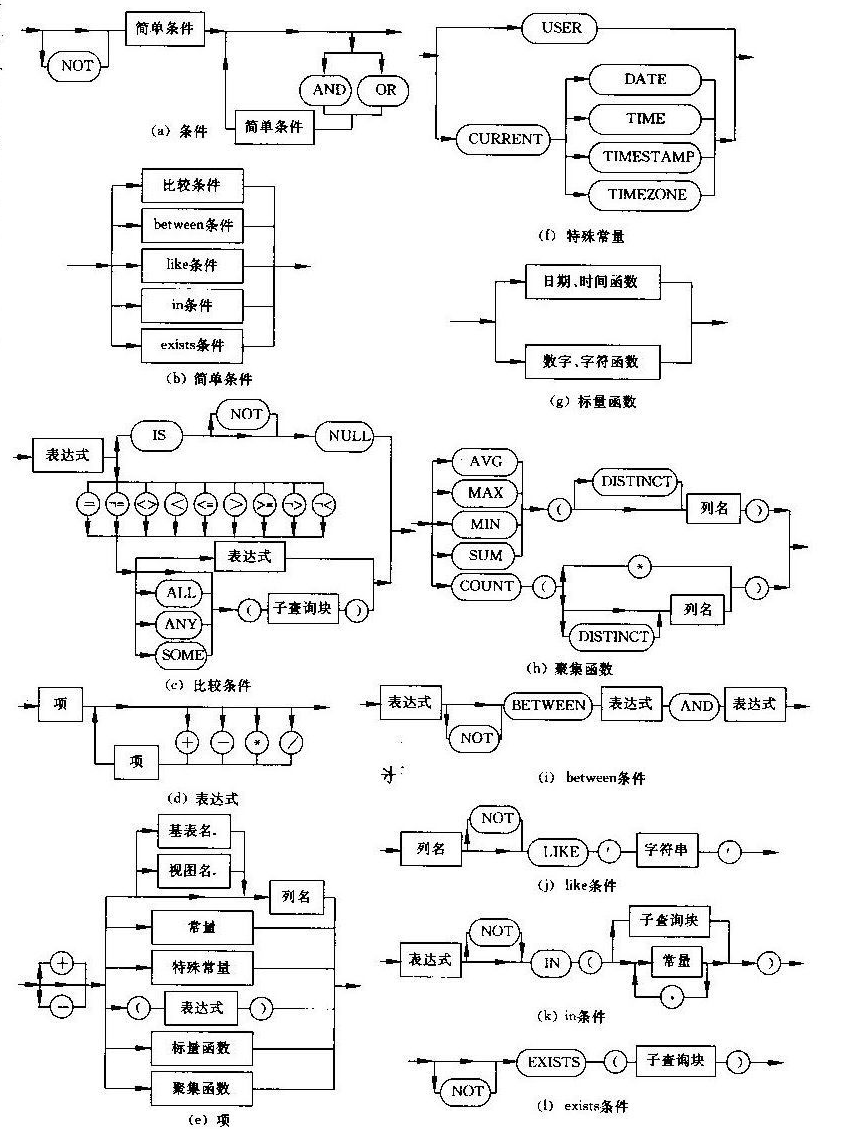


The result of an SQL query is a relation.

The meaning of *select* is different from

The **select** clause can contain arithmetic expressions involving the operation, +, –, \*, and /, and operating on constants or attributes of tuples.





DISTINCT -> FROM -> WHERE -> GROUB BY -> HAVING -> ORDER BY.

ASC DESC

**The Rename Operation**

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

**String Operations**

The operator “like” uses patterns that are described using two special characters:

percent (%). The % character matches any substring.

underscore (\_). The \_ character matches any character.

concatenation (using “||”)

converting from upper to lower case (and vice versa) (**upper, lower**)

finding string length, extracting substrings, etc.

**Set Operations**

**(select** *customer\_name* **from** *depositor*)  
**union  
(select** *customer\_name* **from** *borrower)*

Intersect、except;

自动去除重复。若要全部则: UNION ALL 、 INTERSECT ALL 、 EXCEPT ALL;

**Aggregate Functions**

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

**Group By**

**Having Clause**

All aggregate operations except **count(\*)** ignore tuples with null values on the aggregated attributes.

**Nested Subqueries**

**A subquery is a select-from-where expression that is nested within another query (appear in where clause or having by clause).**

Can’t use order by in subquery

set membership*: in and not in*

**Set Comparison**

**> SOME 大于子查询结果中的某个值**

**> ALL 大于子查询结果中的所有值**

**< SOME 小于子查询结果中的某个值**

**< ALL 小于子查询结果中的所有值**

**>= SOME 大于等于子查询结果中的某个值**

**>= ALL 大于等于子查询结果中的所有值**

**<= SOME 小于等于子查询结果中的某个值**

**<= ALL 小于等于子查询结果中的所有值**

**= SOME 等于子查询结果中的某个值**

**=ALL 等于子查询结果中的所有值（通常没有实际意义）**

**!=（或<>） SOME 不等于子查询结果中的某个值**

**!=（或<>）ALL 不等于子查询结果中的任何一个值**

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

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

**exists**  *r* ⇔ *r* ≠ *Ø* 如果子查询不为空，则返回true；

**not exists** *r* ⇔ *r* = *Ø*

Relevant Nested Subqueries

**NOT EXIST(B EXCEPT A) 表示关系A包含关系B。**

The **unique** construct tests whether a subquery has any duplicate tuples in its result. （专门用来处理“只有一个”、“有两个或者两个以上的”）

**Derived Relations**

SQL allows a subquery expression to be used in the **from** clause

Find the average account balance of those branches where the average account balance is greater than $1200.

**select** *branch\_name, avg\_balance* **from** (**select** *branch\_name,* **avg** (*balance*)**from** *account* **group by** *branch\_name* )**as** *branch\_avg* ( *branch\_name, avg\_balance* )**where** *avg\_balance >* 1200

Note that we do not need to use the **having** clause, since we compute the temporary (view) relation *branch\_avg* in the **from** clause, and the attributes of *branch\_avg* can be used directly in the **where** clause.

**With Clause**

The **with** clause provides a way of defining a temporary view whose definition is available only to the query in which the **with** clause occurs.

**Views (只要没有更新操作，视图可以出现在关系名出现的任何地方)**

A **view** provides a mechanism to hide certain data from the view of certain users.

**create view** *v* **as** *<* query expression >

A view definition causes the saving of an expression; the expression is substituted into queries using the view.

**Views Defined Using Other Views**

A view relation *v*1 is said to *depend directly* on a view relation *v2*  if *v*2 is used in the expression defining *v*1

A view relation *v*1 is said to *depend on* view relation *v2* if either *v*1 depends directly to *v*2  or there is a path of dependencies from *v*1 to *v*2

A view relation *v* is said to be *recursive*  if it depends on itself.

**View Expansion**

View expansion of an expression repeats the following replacement step:

**repeat** Find any view relation *vi* in *e*1  
 Replace the view relation *vi* by the expression defining *vi*   
 **until** no more view relations are present in *e*1

As long as the view definitions are not recursive, this loop will terminate

**Modification of the Database – Deletion**

Delete from *r* where *P*

**Modification of the Database – Insertion**

Insert into r(A1,A2,…,An) values(v1,v2, …,vn)

**Modification of the Database – Updates**

UPDATE *r* SET *A1*=*v1*,…,*Ai*=*vi* WHERE *P*

**update** *account*  
 **set** *balance* = **case**   
 **when** *balance* <= 10000 **then** *balance* \*1.05  
 **else** *balance* \* 1.06  
 **end**

**Update of a View**

Most SQL implementations allow updates only on simple views (without aggregates) defined on a single relation