Chapter 3

Basic SQL Query Language

Ch3 Basic SQL Query Language

3.1 Introduction

- **Structured Query Language – SQL**
  - **including update, insert, delete operations**

- **SQL History**
  - **1976: SEQUEL (IBM, SYSTEM-R)**
  - **1979: SQL (Oracle公司，第一个商用版本)**
  - **1986: SQL-86 (ANSI)**
  - **1989: SQL-89 (ANSI/ISO) (SQL1)**
  - **1992: SQL-92 (ANSI/ISO) (SQL2)**
  - **1999: SQL-99 (ANSI/ISO) (SQL3)**
  - **2003: SQL-2003 (ANSI/ISO) (SQL4)**
  - **?: ……(SQL5)**

3.1 Introduction

- SQL 2003 — ISO/IEC 9075-*
  - 1: Framework (SQL/Framework).
  - 2: Foundation (SQL/Foundation).
  - 3: Call Level Interface (SQL/CLI).
  - 4: Persistent Stored Modules (SQL/PSM).
  - 9: Management of External Data (SQL/MED).
  - 10: Object Language Bindings (SQL/OLB).
  - 11: Information and Definition Schemas (SQL/Schemata).
  - 13: Java Routines and Types Using the Java Programming Language (SQL/JRT).
  - 14: XML-Related Specifications (SQL/XML).

3.2 Setting Up the Database

- **ANSI SQL Datatype**
  - **CHARACTER DataType**
  - **NUMERIC DataType**

3.2 Setting Up the Database

- **ANSI CHARACTER Datatype**
  - **CHARACTER(n), CHAR(n)**
    - **fixed-length character strings**
  - **CHARACTER VARYING(n)**
  - **CHAR VARYING(n)**
    - **variable-length character strings**

3.2 Setting Up the Database

- **ANSI NUMERIC Datatype**
  - **NUMERIC(p, s), DECIMAL(p, s), DEC(p, s)**
    - **precision: total number of digits**
    - **scale: number of digits to the right of the decimal point**
  - **INTEGER, INT, SMALLINT**
  - **FLOAT(p)**
  - **REAL**
  - **DOUBLE PRECISION**

3.2 Setting Up the Database

- **Oracle：Character Datatype**
  - **CHAR(n)**
    - **fixed-length character strings (1≤n≤2000)**
  - **VARCHAR(n)**
    - **variable-length character strings (1≤n≤4000)**
  - **LONG**
    - **variable-length character data(text data) (maximum size 2GB)**

3.2 Setting Up the Database

- **Oracle：NUMBER Datatype**
  - **NUMBER**
    - **fixed and floating-point numbers**
  - **NUMBER(precision, scale)**
  - **NUMBER(*, scale)**
    - **precision: total number of digits**
    - **scale: number of digits to the right of the decimal point**
      - **default value is zero**
      - **negative scale: rounds the actual data to the specified number of places to the left of the decimal point**

3.2 Setting Up the Database

- **Example of NUMBER**

Input Data	Specified As	Stored As
7,456,123.89	NUMBER	7456123.89
7,456,123.89	NUMBER(*,1)	7456123.9
7,456,123.89	NUMBER(9)	7456124
7,456,123.89	NUMBER(9,2)	7456123.89
7,456,123.89	NUMBER(9,1)	7456123.9
7,456,123.89	NUMBER(6)	(not accepted)
7,456,123.89	NUMBER(7,-2)	7456100

Figure 3.2 Limited Form of Create Table Statement

```
CREATE TABLE tablename (
    colname datatype [ NOT NULL ]
    { , colname datatype [ NOT NULL ] ... }
    [ , PRIMARY KEY ( colname { , colname ... } ) ]
) ;
```

1) [...] { ... }
2) CREATE TABLE
3) NOT NULL
4) PRIMARY KEY

3.2 Setting Up the Database

- SQL statements for table creation for CAP database

```
CREATE TABLE  customers (
        cid CHAR(4) NOT NULL,
        cname VARCHAR(13),
        city VARCHAR(20),
        discnt REAL,
        PRIMARY KEY(cid) );
```

3.2 Setting Up the Database

- **SQL statements for table creation for CAP database**

```
CREATE TABLE agents (
        aid CHAR(3) NOT NULL,
        aname VARCHAR(13),
        city VARCHAR(20),
        percent SMALLINT,
        PRIMARY KEY (aid) );
```

3.2 Setting Up the Database

- SQL statements for table creation for CAP database

```
CREATE TABLE  products (
        pid CHAR(3) NOT NULL,
        pname VARCHAR(13),
        city VARCHAR(20),
        quantity INTEGER,
        price DOUBLE PRECISION,
        PRIMARY KEY(pid) );
```

3.2 Setting Up the Database

- SQL statements for table creation for CAP database

```
CREATE TABLE  orders (
        ordno INTEGER NOT NULL,
        month CHAR(3),
        cid CHAR(4),
        aid CHAR(3),
        pid CHAR(3),
        qty INTEGER,
        dollars DOUBLE PRECISION,
        PRIMARY KEY(ordno) );
```

3.3 Simple Select Statements

❑ **select statement**

```
SELECT  ......
FROM  .......
[ WHERE  ....... ]
[ GROUP BY  ....... [ HAVING  ....... ] ]
[ ORDER BY  ...... ];
```

3.3 Simple Select Statements

- select statement

```
SELECT  * | colname { , colname ... }
FROM  tablename { , tablename ... }
[ WHERE search_condition ]
[ GROUP BY colname { , colname ... }
    [ HAVING search_condition ] ]
[ ORDER BY colname [ ASC | DESC ]
        { , colname [ ASC | DESC ] ... } ];
```

3.3 Simple Select Statements

- query in relational algebra (single relation)

  (R where Condition) [A_1, A_2, ..., A_m]

- query in SQL

```
SELECT  A_1, A_2, ..., A_m
FROM    R
WHERE   Condition ;
```

3.3 Simple Select Statements

❑query in relational algebra (PRODUCT)

$((R_1 \times R_2 \times \ldots \times R_n)$ where Condition$)$ $[A_1, A_2, \ldots, A_m]$

❑query in SQL

```
SELECT  A1, A2, ..., Am
FROM    R1, R2, …, Rn
WHERE   Condition ;
```

3.3 Simple Select Statements

- **query in relational algebra (JOIN)**
  - **Head(R)={A_1, …, A_n, B_1, …, B_k}**
  - **Head(S)={B_1, …, B_k, C_1, …, C_m}**

((R ∞ S) where Condition) [A_1, A_2, ..., A_m]

- **query in SQL**

SELECT A_1, A_2, ..., A_m

FROM R, S

WHERE Condition and

$R.B_1=S.B_1$ and $R.B_2=S.B_2$ ... and $R.B_k=S.B_k$;

3.3 Simple Select Statements

❑ **query in relational algebra** **(Theta-Join)**

$(R \infty_{\text{Condition}} S) [A_1, A_2, ..., A_m]$

❑ **query in SQL**

```
SELECT  A1, A2, ..., Am
FROM    R, S
WHERE   Condition ;
```

3.3 Simple Select Statements

- **Exp 3.3.1 Find aid and names of agents that are based in New York.**

- **Relational Algebra**

  (AGENTS where city=‘New York’) [aid,aname]

- **SQL**

```
SELECT  aid, aname
FROM    agents
WHERE   city = 'New York’ ;
```

3.3 Simple Select Statements

- **Exp 3.3.2 Display all values of customers in table CUSTOMERS.**

- **Relational Algebra**

  (CUSTOMERS) [cid, cname, city, discnt]

- **SQL(1)**

```
SELECT cid, cname, city, discnt
FROM     customers ;
```

- **SQL(2)**

```
SELECT  *
FROM     customers ;
```

3.3 Simple Select Statements

- Exp 3.3.3 Select product ids of products for which orders are placed.
  - Relational Algebra : (ORDERS) [pid]
  - SQL(1): SELECT pid FROM orders ;

- ALL | DISTINCT

```
SELECT distinct pid FROM orders ;
```

- 另一组例子

```
① SELECT aid, pid FROM orders ;
② SELECT distinct aid, pid FROM orders ;
```

❑**Exp 3.3.4 Retrieve all (cname, aname) pairs where the customer places an order through the agent.**

❑**Relational Algebra（2种表示方法）**

①(C[cid, cname] ∞ O) ∞ A) [cname, aname]

②((C × O × A) where C.cid=O.cid and
O.aid=A.aid) [cname, aname]

❑**SQL:**

```
SELECT  distinct  cname, aname
FROM      customers, orders, agents
WHERE   customers.cid=orders.cid and
         orders.aid=agents.aid ;
```

3.3 Simple Select Statements

- table and column alias
  - table alias in FROM clause
    - 方法1

      table_name as alias_name
    - 方法2

      table_name   alias_name

  - column alias in SELECT clause

    expression as alias_name

❑Exp 3.3.4 Retrieve all (cname, aname) pairs where the customer places an order through the agent.

❑SQL1 (no alias)

```
SELECT  distinct  cname, aname
FROM       customers, orders, agents
WHERE   customers.cid=orders.cid and
           orders.aid=agents.aid ;
```

❑SQL2 (table alias in FROM clause)

```
SELECT  distinct  cname, aname
FROM       customers  c, orders  o, agents  a
WHERE   c.cid=o.cid  and  o.aid=a.aid ;
```

❑ SQL (no alias)

```
SELECT  ordno, dollars,
              qty*price*(1-discnt*0.01)
FROM      customers, orders, products
WHERE   customers.cid=orders.cid  and
               orders.pid=products.pid ;
```

❑ column alias in SELECT clause

```
SELECT  ordno, dollars,
    o.qty*p.price*(1-c.discnt*0.01) as mydollars
FROM   customers  c, orders  o, products  p
WHERE   c.cid=o.cid and o.pid=p.pid ;
```

❑Exp 3.3.6 List all pairs of customer cids based in the same city.

```
SELECT  c1.cid, c2.cid
FROM    customers  c1, customers  c2
WHERE   c1.city = c2.city and c1.cid < c2.cid ;
```

```
FOR c1 FROM ROWS 1 TO LAST OF customers
    FOR c2 FROM ROWS 1 TO LAST OF customers
        IF (c1.city = c2.city and c1.cid < c2.cid)
        PRINT OUT pairs of (c1.cid, c2.cid)
    END FOR c2
END FOR c1
```

3.3 Simple Select Statements

- Exp 3.3.7 Find pids of products ordered by at least two customers.

Need distinct ? YES

- SQL(1):

```
SELECT  distinct  x1.pid
FROM     orders  x1,  orders  x2
WHERE   x1.pid = x2.pid and x1.cid < x2.cid ;
```

Need distinct ? YES

- SQL(2):

```
SELECT  distinct  x1.pid
FROM     orders  x1,  orders  x2
WHERE   x1.pid = x2.pid and x1.cid <> x2.cid ;
```

3.3 Simple Select Statements

- **Exp 3.3.8 Get cids and names of customers ordering a product for which an order is placed by agent a06.**

```
SELECT  distinct  c.cid, c.cname
FROM  orders  o1, orders o2, customers  c
WHERE  o1.aid='a06'  and  o1.pid=o2.pid
          and  o2.cid=c.cid ;
```

3.4 Subqueries

- The IN Predicate

  expr [NOT] IN (subquery)

- The Quantified Comparison Predicate

  expr θ SOME|ANY|ALL (subquery)

- The EXISTS Predicate

  [NOT] EXISTS (subquery)

- The BETWEEN Predicate

  expr [NOT] BETWEEN expr1 AND expr2

3.4 Subqueries

- **The IS NULL Predicate**

  column IS [NOT] NULL

- **The LIKE Predicate**

  column [NOT] LIKE val1 [ESCAPE val2]

  - **underscore (_): any single character**
  - **percent (%): any sequence of zero or more characters**

3.4 Subqueries

- **The IN Predicate**

- **Exp 3.4.1 Retrieve cids of customers who place orders with agents in Duluth or Dallas.**

  - **SQL 1**

```
select  distinct  cid
from    orders  o, agents  a
where  a.aid=o.aid  and
         (a.city='Duluth' or a.city='Dallas')
```

❑SQL 1 (SLOW)

```
select  distinct  cid
from    orders  o, agents  a
where  a.aid=o.aid  and
        (a.city='Duluth' or a.city='Dallas')
```

```
FOR $o_1$ FROM ROWS 1 TO LAST OF orders
    FOR $a_1$ FROM ROWS 1 TO LAST OF agents
        ……
        ……
    END FOR $a_1$
END FOR $o_1$
```

❑SQL 2 (<u>FAST</u>)

```
select  distinct  cid    from  orders
where  aid  IN  (
     select  aid     from  agents
     where  city= ‘Duluth’  or  city=‘Dallas’ )
```

```
FOR $a_2$ FROM ROWS 1 TO LAST OF agents
    ......(get aids of agents in Duluth or Dallas)
END FOR $a_2$       // result is the set aids-set

FOR $o_2$ FROM ROWS 1 TO LAST OF orders
    ...... ( find orders with aid IN aids-set and
   output cid of the orders )
END FOR $o_2$
```

```
FOR $o_1$ FROM ROWS 1 TO LAST OF orders
    FOR $a_1$ FROM ROWS 1 TO LAST OF agents
        ......
    END FOR $a_1$
END FOR $o_1$
```

(SQL 1)

```
FOR $a_2$ FROM ROWS 1 TO LAST OF agents
    ......
END FOR $a_2$        // result is the set aids-set

FOR $o_2$ FROM ROWS 1 TO LAST OF orders
    ...... ( find orders with aid IN aids-set )
END FOR $o_2$
```

(SQL 2)

- SQL 1 (SLOW)

```
select  distinct  cid
from    orders  o, agents  a
where  a.aid=o.aid  and
        (a.city='Duluth' or a.city='Dallas')
```

- SQL 2 (FAST)

```
select   distinct  cid
from     orders
where   aid  IN  (
    select  aid
    from  agents
    where  city= 'Duluth'  or  city='Dallas')
```

3.4 Subqueries

- **Exp 3.4.2 Get all information concerning agents based in Duluth or Dallas.**

```
SELECT  *
FROM  agents
WHERE  city  IN  ( ‘Duluth’, ‘Dallas’ ) ;
```

❑ **Exp 3.4.3 Get the names and discounts of all customers who place orders through agents in Duluth or Dallas.**

```
SELECT  cname,  discnt
FROM     customers
WHERE  cid  IN  (
    SELECT  o.cid
    FROM     orders  o
    WHERE  o.aid  IN  (
          SELECT  a.aid
          FROM     agents  a
          WHERE  a.city IN ('Duluth', 'Dallas'))) ;
```

❑Exp 3.4.4 Find the names of customers who order product p05.

- SQL(1)

```
SELECT  distinct  cname
FROM    customers  c,  orders  o
WHERE  c.cid=o.cid and o.pid=‘p05’ ;
```

- SQL(2)

```
SELECT  distinct  cname
FROM    customers  c
WHERE ‘p05’  IN  ( select pid
                   from  orders  o
                   where o.cid=c.cid );
```

❑Exp 3.4.4 Find the names of customers who order product p05.

- SQL(1) 多表连接查询

```
SELECT  distinct  cname
FROM   customers  c,  orders  o
WHERE  c.cid=o.cid and o.pid=‘p05’ ;
```

- SQL(2) 相关子查询

```
SELECT distinct cname
FROM   customers  c
WHERE ‘p05’  IN  (
    select pid
    from  orders  o
    where o.cid=c.cid );
```

- SQL(3) 独立子查询

```
SELECT  distinct  cname
FROM   customers  c
WHERE  cid  IN  (
    select  cid
    from  orders  o
    where o.pid=‘p05’ );
```

3.4 Subqueries

- **The proper scope of table and column name!**
- **Exp 3.4.5 Get the names of customers who order product ‘p07’ from agent ‘a03’.**

- **SQL(ERROR)**

```
SELECT  cname
FROM  customers
WHERE  orders.aid = ‘a03’ and
        ‘p07’ IN ( select  pid  from  orders
                where cid=customers.cid ) ;
```

ERROR

CORRECT

❑Exp 3.4.6 Find ordno values for all orders placed by customers in Duluth through agents in New York.

columns in orders

```
SELECT  ordno
FROM  orders
WHERE  (cid, aid)  IN
    (select  cid, aid
     from    customers  c,  agents  a
     where  c.city='Duluth' and
                 a.city='New York') ;
```

columns in c and a

3.4 Subqueries

- **The Quantified Comparison Predicate**

  expr $\underline{\theta \{ \text{SOME} \mid \text{ANY} \mid \text{ALL} \} (\text{subquery})}$

- **Quantified Predicate vs [NOT] IN**

  IN is =SOME

  NOT IN is <>ALL

3.4 Subqueries

- ❑ **Exp 3.4.7 Find aid values of agents with a minimum percent commission.**

```
SELECT  aid
FROM  agents
WHERE  percent <= ALL (
          select  percent  from  agents ) ;
```

3.4 Subqueries

- **Exp 3.4.8 Find all customers who have the same discount as that of any of the customers in Dallas or Boston.**

```
SELECT  cid,  cname
FROM   customers
WHERE  discnt = SOME (
    select  discnt
    from   customers
    where  city='Dallas'  or  city='Boston' ) ;
```

3.4 Subqueries

- Exp 3.4.9 Get cid values of customers with discnt smaller than those of any customers who live in Duluth.

ANY or ALL

```
SELECT cid, cname
FROM   customers
WHERE  discnt < ?   (
        select discnt
        from   customers
        where  city='Duluth' ) ;
```

3.4 Subqueries

- **Exp 3.4.9 Get cid values of customers with discnt smaller than those of any customers who live in Duluth.**

```
SELECT  cid,  cname
FROM   customers
WHERE  discnt  <  ALL  (
        select  discnt
        from   customers
        where  city='Duluth' ) ;
```

3.4 Subqueries

- The EXISTS Predicate

  [NOT] EXISTS (subquery)

  - The predicate EXISTS (subquery) is TRUE if and only if the subquery returns a non-empty set.

  - The predicate NOT EXISTS (subquery) is TRUE if and only if the subquery returns a empty set.

  ▶ example 3.4.10 - 14

3.4 Subqueries

- **The BETWEEN Predicate**

  expr [NOT] BETWEEN expr1 AND expr2

- **The IS NULL Predicate**

  colname IS [NOT] NULL

3.4 Subqueries

模板(pattern)

❑ The LIKE Predicate

colname [NOT] LIKE val1 [ESCAPE val2]

转义指示字符

– Character in pattern
  – Underscore (_) : Wildcard for any single character
  – Percent (%) : Wildcard for any sequence of zero or more characters
  – Escape character : Precedes quoted literal character
  – All other characters : Represent themselves

Examples of Subqueries

[Exp 3.4.9] **Find cid values of customers with discnt smaller than those of any customers who live in Duluth.**

[Exp 3.4.10] **Retrieve all customer names where the customer places an order through agent a05.**

[Exp 3.4.11] **Get cids of customers who order both products p01 and p07.**

[Exp 3.4.12 ~ 13] **Find all customer names where the customer does not place an order through agent a05.**

[Exp 3.4.14] **Find cids of all customers who don’t place any order through agent a03.**

[Exp 3.4.15] **Retrieve the city names containing**

Exp 3.4.9 Find cid values of customers with discnt smaller than those of any customers who live in Duluth.

```
SELECT  cid
FROM    customers
WHERE   discnt  < ALL (
            SELECT  discnt
            FROM    customers
            WHERE   city = 'Duluth' ) ;
```

- **Thinking**
  - Can we do this with (not) exists predicate Yes
  - Can we do this without a subquery? No

[another example] Find cid values of customers with discnt smaller than a customer who lives in Duluth.

```
SELECT  cid
FROM    customers
WHERE   discnt  < SOME (
            SELECT  discnt
            FROM    customers
            WHERE   city = 'Duluth' ) ;
```

- **Thinking**
  - Can we do this with (not) exists predicat **Yes**
  - Can we do this without a subquery? **Yes**

Exp 3.4.10 Retrieve all customer names where the customer places an order through agent a05.

```
1. SELECT  distinct  cname
   FROM     customers c, orders o
   WHERE   c.cid = o.cid  and  o.aid = ‘a05’ ;
```

```
2. SELECT  distinct  cname
   FROM     customers
   WHERE   cid   IN
   (SELECT cid FROM orders WHERE aid=‘a05’) ;
```

```
3. SELECT  distinct  cname
   FROM     customers  c
   WHERE   EXISTS (SELECT * FROM orders o
                WHERE  o.cid=c.cid  and  o.aid=‘a05’) ;
```

Exp 3.4.11 Get cids of customers who order both products p01 and p07.

```
1. (SELECT cid FROM orders WHERE pid=‘p01’)
       INTERSECT
   (SELECT cid FROM orders WHERE pid=‘p07’) ;
```

```
2. SELECT  o1.cid  FROM  orders  o1, orders  o2
   WHERE  o1.cid=o2.cid  and
                o1.pid=‘p01’  and  o2.pid=‘p07’ ) ;
```

```
3. SELECT  o1.cid  FROM orders  o1
   WHERE  o1.pid=‘p01’  and  o1.cid  IN (
          SELECT  o2.cid   FROM  orders  o2
          WHERE  o2.pid=‘p07’ ) ;
```

Exp 3.4.12 & Exp 3.4.13 Find all customer names where the customer does not place an order through agent a05.

((C[cid] – (O where aid = ‘a05’) [cid]) ∞ C) [cname]

Answer 1:

```
SELECT  cname
FROM  customers
WHERE  cid  <> ALL ( SELECT  o.cid
                     FROM  orders  o
                     WHERE  o.aid = ‘a05’ ) ;
```

Exp 3.4.12 & Exp 3.4.13 Find all customer names where the customer does not place an order through agent a05.

((C[cid] – (O where aid = ‘a05’) [cid]) ∞ C) [cname]

Answer 2:

```
SELECT  cname
FROM     customers  c
WHERE   NOT EXISTS (
    SELECT  *
    FROM   orders  o
    WHERE   o.cid = c.cid  and  o.aid = ‘a05’  ) ;
```

- ❑ **There is no way to do this without a subquery !**

[Example] Find all cid, aid pairs where the customer does not place an order through the

```
1. SELECT  cid,  aid
   FROM  customers  c,  agents  a
   WHERE   NOT EXISTS (
           SELECT  *
           FROM  orders  o
           WHERE  o.cid = c.cid  and  o.aid = a.aid );
```

```
2. SELECT  cid,  aid
   FROM  customers  c,  agents  a
   WHERE   (cid,  aid)  NOT  IN  (
           SELECT  o.cid,  o.aid   FROM  orders  o );
```

Exp 3.4.14 Find cids of all customers who don't place any order through agent a03.

```
1. SELECT  cid
   FROM  customers
   WHERE   cid  NOT IN  ( SELECT  o.cid
                          FROM  orders  o
                          WHERE  o.aid = 'a03' ) ;
```

```
2. SELECT  cid
   FROM      customers  c
   WHERE   NOT EXISTS (
       SELECT  *
       FROM  orders  o
       WHERE   o.cid = c.cid  and  o.aid = 'a03'  ) ;
```

Exp 3.4.15 Retrieve the city names containing customers who order product p01.

```
1.  SELECT  city    FROM  customers  c, orders  o
    WHERE   c.cid = o.cid  and  o.pid = ‘p01’  ;
2.  SELECT  city    FROM  customers
    WHERE   cid  IN  ( SELECT  cid    FROM  orders    WHERE
        pid = ‘p01’ ) ;
3.  SELECT  city    FROM  customers
    WHERE   cid  = SOME  ( SELECT  cid    FROM  orders
        WHERE  pid = ‘p01’ ) ;
4.  SELECT  city    FROM  customers  c
    WHERE   ‘p01’  IN  ( SELECT  o.pid    FROM  orders  o
        WHERE  o.cid = c.cid ) ;
5.  SELECT  city    FROM  customers  c
    WHERE   EXISTS ( SELECT  *    FROM  orders  o
        WHERE   o.cid = c.cid  and  o.pid = ‘p01’  ) ;
```

- *[Exp 2.9.1]* **Get the names of customers who order at least one product priced at $0.50.**
- *[Exp 2.9.3]* **Retrieve customers who place orders only through agent a03.**
- *[Exp 2.9.4]* **Find products that have never been ordered by a customer based in New York through an agent based in Boston.**
- *[Exp 2.9.9]* **Get cids of customers who place an order through at least one agent who places an order for product p03.**
- *[Exp 2.9.10]* **Get cids of all customers who have the same discount as any customer in Dallas or Boston.**

Example of Simple Select Statements

- ***Exp 2.9.1***: **Get the names of customers who order at least one product priced at $0.50.**

  (((P where price=0.50)[pid]∞O)∞C) [cname]

- **SQL:**

```
SELECT  cname
FROM  products  p, orders  o, customers  c
WHERE  price=0.50 and p.pid=o.pid and
            o.cid=c.cid ;
```

Example of Simple Select Statements

- Exp 2.9.3: Retrieve customers who place orders only through agent a03.

  O[cid] – (O where aid ≠ ‘a03’) [cid]

- SQL:

```
SELECT  o1.cid
FROM    orders  o1
WHERE  o1.cid  NOT IN  (
              SELECT o2.cid
              FROM  orders  o2
              WHERE  o2.aid <> ‘a03’) ;
```

Example of Simple Select Statements

- *Exp 2.9.4*: Find products that have never been ordered by a customer based in New York through an agent based in Boston.

1) T_1 := (C where city = ‘New York’)[cid]
2) T_2 := ((($T_1 \infty$O)∞A) where city=‘Boston’)[pid]
3) T_3 := P[pid] – T_2

1) T_1 := (C where city = ‘New York’)[cid]

2) T_2 := ((($T_1 \infty$O)∞A) where city=‘Boston’) [pid]

3) T_3 := P[pid] – T_2

- **SQL:**

```
SELECT  p.pid
FROM    products  p
WHERE  p.pid  NOT IN (
    SELECT o.pid
    FROM  customers c, agents a, orders o
    WHERE  c.city=‘New York’  and
           a.city=‘Boston’  and  c.cid=o.cid
           and  o.aid=a.aid ) ;
```

Example of Simple Select Statements

- ❑ *Exp 2.9.9*: Get cids of customers who place an order through at least one agent who places an order for product p03.

  1) T_1 := (O where pid = 'p03')[aid]
  2) T_2 := ($T_1 \infty$ O) [cid]

SQL Answer 1:

```
SELECT  o2.cid
FROM     orders  o1,  orders  o2
WHERE    o1.pid='p03'  and  o1.aid=o2.aid ;
```

- *Exp 2.9.9*: **Get cids of customers who place an order through at least one agent who places an order for product p03.**

  1) **T_1 := (O where pid = ‘p03’)[aid]**

  2) **T_2 := ($T_1 \infty$ O) [cid]**

SQL Answer 2:

```
SELECT  o2.cid
FROM      orders  o2
WHERE    o2.aid  IN  ( SELECT  o1.aid
                       FROM  orders  o1
                       WHERE  o1.pid=‘p03’  ) ;
```

Example of Simple Select Statements

Exp 2.9.10: **Get cids of all customers who have the same discount as any customer in Dallas or Boston.**

1) T_1:=(C where city='Dallas' or city='Boston') [discnt]

2) T_2 := (T_1 ∞ C) [cid]

SQL Answer 1:

```
SELECT  c2.cid
FROM    customers  c1,  customers  c2
WHERE (c1.city='Dallas' or c1.city='Boston')
          and  c1.discnt = c2.discnt ;
```

Exp 2.9.10: Get cids of all customers who have the same discount as any customer in Dallas or Boston.

SQL Answer 2:

```
SELECT  c2.cid
FROM    customers  c2
WHERE   c2.discnt  IN (
            SELECT  c1.discnt
            FROM    customers  c1
            WHERE   c1.city = ‘Dallas’ or
            c1.city = ‘Boston’ ) ;
```

- ***[Exp 2.9.11]*** **List pids of products that are ordered through agents who place orders for (possibly different) customers who order at least one product from an agent who has placed an order for customer c001.**

- ***[Exp 2.9.12]*** **Get pids of products not ordered by any customer living in a city whose name begin with the letter D.**

Exp 2.9.11: List pids of products that are ordered through agents who place orders for (possibly different) customers who order at least one product from an agent who has placed an order for customer c001.

1) T_1 := (O where cid = ‘c001’)[aid]
2) T_2 := ($T_1 \infty$ O) [cid]
3) T_3 := ($T_2 \infty$ O) [aid]
4) T_4 := ($T_3 \infty$ O) [pid]

Exp 2.9.11: List pids of products that are ordered through agents who place orders for (possibly different) customers who order at least one product from an agent who has placed an order for customer c001.

SQL Answer 1:

```
SELECT  o4.pid
FROM  orders o1, orders o2,
      orders o3, orders o4
WHERE  o1.cid=‘c001’ and o1.aid=o2.aid
   and o2.cid=o3.cid and o3.aid=o4.aid ;
```

```
SELECT  o4.pid                    (SQL Answer 1:)
FROM  orders o1, orders o2, orders o3, orders o4
WHERE  o1.cid='c001' and o1.aid=o2.aid and
    o2.cid=o3.cid and o3.aid=o4.aid ;
```

SQL Answer 2:

```
SELECT  o4.pid    FROM  orders  o4
WHERE   o4.aid  IN  (
      SELECT  o3.aid    FROM  orders  o3
      WHERE   o3.cid  IN (
         SELECT  o2.cid    FROM  orders  o2
         WHERE   o2.aid  IN (
            SELECT  o1.aid    FROM  orders  o1
            WHERE  o1.cid='c001' ) ) );
```

❑ *Exp 2.9.12*: **Get pids of products not ordered by any customer living in a city whose name begin with the letter D.**

```
SELECT  p.pid
FROM    products  p
WHERE  p.pid  NOT  IN  (
    SELECT  o.pid
    FROM      orders  o,  customers  c
    WHERE   o.cid = c.cid  and
                c.city LIKE 'D%'                 ) ;
```

3.4 Subqueries

❑summary

Relational Algebra	SQL Predicate
natural join	IN
	= SOME
	EXISTS
difference	NOT IN
	<> ALL
	NOT EXISTS

3.5 UNION Operators and FOR ALL Conditions

- **The UNION Operator**
  - **Subquery UNION [ALL] Subquery**
    - **UNION**
      - **no duplicate rows in the result**
    - **UNION ALL**
      - **may have duplicate rows in the result**

3.5 UNION Operators and FOR ALL Conditions

– Example

- R_1 := S_1 UNION S_2
- R_2 := S_1 UNION ALL S_2
- R_3 := (S_1 UNION ALL S_2) UNION S_3
- R_4 := S_1 UNION ALL (S_2 UNION S_3)

R_1 & R_3: no duplicate rows

3.5 UNION Op. and FOR ALL Cond.

- **Example 3.5.1** **Retrieve all cities where either a customer or an agent, or both, is based.**

Answer 1:

```
( select  city  from  customers ) UNION
( select  city  from  agents )
```

Answer 2:

```
( select  city  from  customers )
UNION ALL ( select  city  from  agents )
```

3.5 UNION Op. and FOR ALL Cond.

- Example Retrieve all cities where either a customer or an agent or procuct is based.

1. (select city from customers UNION select city from agents) UNION ALL (select city from product)

2. (select city from customers UNION ALL select city from agents) UNION (select city from product)

3.6 Some Advanced SQL Syntax

- **The INTERSECT and EXCEPT Operators**

```
subquery
{ UNION | INTERSECT | EXCEPT [ALL]
   subquery }
```

3.5 UNION Op. and FOR ALL Cond.

- The division operation in Relational Algebra

  [Exp 3.5.2] Find cids of customers who place orders with ALL agents based in New York.

  <u>o[cid, aid] ÷ (a where city='New York')[aid]</u>

- No *division* operator in SQL. The query means:

if row <u>*c*</u> in customers table is a customer of result set, then

for each row <u>*a*</u> in agents table which is based in New York

we can find a row <u>*o*</u> in orders table which:
o.cid = c.cid and o.aid = a.aid

3.5 UNION Op. and FOR ALL Cond.

- We can understand this query request
  - if row c in customers table is a customer of result set, then
    - no row(a) in agents table which
      - is based in New York, and
      - no row(o) in orders table which
        - o.cid = c.cid and o.aid = a.aid
  - so, we can construct first condition

cond1: NOT EXISTS (select * from orders o
where o.cid=c.cid and o.aid=a.aid)

3.5 UNION Op. and FOR ALL Cond.

- Then, We can understand this query request
  - if row <u>c</u> in customers table is a customer of result set, then
    - no row(<u>a</u>) in agents table which
      - is based in New York, and
      - cond1
  - and that, we can construct second condition with cond1:

<u>*cond2*</u>: NOT EXISTS (select * from agents a
where city = ‘New York’
and <u>*cond1*</u>)

3.5 UNION Op. and FOR ALL Cond.

- **Then, We can understand this query request**
  - **if row _c_ in customers table is a customer of result set, then**
    - **cond2**
  - **we can write this query with cond2:**

```
SELECT  c.cid
FROM    customers  c
WHERE   cond2 )
```

3.5 UNION Op. and FOR ALL Cond.

```
SELECT  c.cid
FROM     customers  c
WHERE  cond2 )
```

```
cond2: NOT EXISTS (
           select * from agents a
           where city='New York' and  cond1)
```

```
cond1: NOT EXISTS ( select * from orders o
                    where o.cid = c.cid  and
                          o.aid = a.aid )
```

```
cond1: NOT EXISTS ( select * from orders o
         where o.cid=c.cid  and o.aid=a.aid )
```

```
cond2: NOT EXISTS ( select * from agents a
         where city='New York' and  cond1 )
```

– in the end, we can write this query:

```
SELECT  c.cid   FROM   customers  c
WHERE  NOT EXISTS (
    SELECT  *  FROM  agents  a
    WHERE  a.city = 'New York'  and  NOT
    EXISTS (
        SELECT  *  FROM  orders  o
        WHERE  o.cid=c.cid and o.aid=a.aid ) )
```

$\underline{o}$ [cid, aid] ÷ ($\underline{a}$ where city=‘New York’)[aid]

```
SELECT  c.cid
FROM    customers  c
WHERE  NOT EXISTS (
    SELECT  *   FROM  agents  a
    WHERE  a.city = ‘New York’  and  NOT
    EXISTS (
        SELECT  *   FROM  orders  o
        WHERE  o.cid=c.cid  and  o.aid=a.aid ) )
```

3.5 UNION Op. and FOR ALL Cond.

- **Relational Algebra**

  $R(x, y) \div S(y)$

- **Relational Calculus**

  $\forall z\ (\ \exists y\ (\ P(z, y)\ ))\ \leftrightarrow\ \neg\exists z\ (\ \neg\exists y\ (\ P(z, y)\ ))$

Exp 3.5.3: **Get the aid values of agents in New York or Duluth who place orders for all products costing more than a dollar.**

```
SELECT  aid
FROM    agents  a
WHERE  (city='New York' or city='Duluth') and
    NOT EXISTS (
        SELECT  *
        FROM  products  p
        WHERE  p.price > 1  and  NOT EXISTS (
            SELECT  *
            FROM  orders  o
            WHERE  o.aid=a.aid and o.pid=p.pid))
```

Exp 3.5.4: **Find aid values of agents who place orders for product p01 as well as for all products costing more than a dollar.**

Answer 1:

```
SELECT  aid    FROM  agents  a
WHERE   aid IN (  select aid from orders
                   where pid='p01' )
      and  NOT EXISTS (
   SELECT  *    FROM  products  p
      WHERE  p.price > 1  and  NOT EXISTS (
         SELECT  *    FROM  orders  o
         WHERE  o.aid=a.aid and o.pid=p.pid))
```

Exp 3.5.4: Find aid values of agents who place orders for product p01 as well as for all products costing more than a dollar.

Answer 2:

```
SELECT  aid
FROM    orders  y
WHERE   y.pid = ‘p01’  and  NOT EXISTS (
        SELECT  *
        FROM    products  p
        WHERE  p.price > 1  and  NOT EXISTS (
            SELECT  *
            FROM    orders  x
            WHERE  x.aid=y.aid and x.pid=p.pid))
```

Exp 3.5.5: Find cids for customers who order all products ordered by customer c006.

```
SELECT  cid
FROM      customers  c
WHERE   NOT EXISTS (
    SELECT  *
    FROM      orders  z
    WHERE   z.cid=‘c006’  and  NOT EXISTS (
          SELECT  *
          FROM  orders  y
          WHERE  y.cid=c.cid and y.pid=z.pid))
```

Exp 3.5.6: Find pid values of products supplied to all customers in Duluth.

```
SELECT  pid
FROM      products  p
WHERE   NOT EXISTS (
    SELECT  *
    FROM      customers  c
    WHERE   c.city='Duluth'  and  NOT EXISTS (
          SELECT  *
          FROM  orders  x
          WHERE  x.cid=c.cid and x.pid=p.pid))
```

Example of Simple Select Statements

- *[Exp 2.9.5]* **Get names of customers who order all products priced at $0.50.**

- *[Exp 2.9.6]* **Get cids of customers who order all products that anybody orders.**

- *[Exp 2.9.7]* **Get aids of agents who take orders on at least that set of products ordered by c004.**

- *Exp 2.9.5*: Get names of customers who order all products priced at \$0.50.
  1) T_1 := O[cid, pid] ÷ (P where price=0.50) [pid]
  2) T_2 := (T_1 ∞ C) [cname]

- SQL:

```
SELECT  cname    FROM    customers  c
WHERE  NOT EXISTS (
    SELECT  *  FROM  products  p
    WHERE  p.price=0.50  and  NOT EXISTS (
        SELECT  *  FROM  orders  o
        WHERE  o.cid=c.cid  and  o.pid=p.pid ) )
```

- *Exp 2.9.6*: **Get cids of customers who order all products that anybody orders.**

  O[cid, pid] ÷ O[pid]

- **SQL:**

```
SELECT  c.cid  FROM  customers  c
WHERE  NOT EXISTS (
    SELECT  *  FROM  orders  o1
    WHERE  NOT EXISTS (
        SELECT  *  FROM  orders  o2
        WHERE  o2.cid=c.cid  and
            o2.pid=o1.pid ))
```

- ***Exp 2.9.7***: **Get aids of agents who take orders on at least that set of products ordered by c004.**

  O[aid, pid] ÷ (O where cid = 'c004')[pid]

- **SQL:**

```
SELECT  a.aid
FROM  agents  a
WHERE  NOT EXISTS (
    SELECT  *
    FROM  orders  o1
    WHERE  o1.cid=‘c004’  and  NOT EXISTS (
        SELECT  *
        FROM  orders  o2
        WHERE  o2.aid=a.aid  and  o2.pid=o1.pid ))
```

- **Question**
  1. **Get aids of agents who place orders for all customers who have discount greater than 8.**
  2. **Get cids for customers with the following property: if customer c006 orders a product x through agent y, so the customer orders the product x through the agent y.**
  3. **Get aids of agents who place orders for all customers who place orders for all products costing more than a dollar through the agent.**
  4. **Get aids of agents who place orders for all customers who place orders for all products.**

1. **Get aids of agents who place orders for all customers who have discount greater than 8.**

```
SELECT  aid
FROM  agents  a
WHERE  not exists (
    SELECT  *
    FROM  customers  c
    WHERE  c.discnt > 8  and  not exists (
        SELECT  *
        FROM  orders  x
        WHERE  x.cid=c.cid  and  x.aid=a.aid ) );
```

2. Get cids for customers with the following property: if customer c006 orders a product x through agent y, so the customer orders the product x through the agent y.

```
SELECT  cid
FROM      customers  c
WHERE   NOT EXISTS (
    SELECT  *
    FROM      orders  o1
    WHERE   o1.cid='c006'  and  NOT EXISTS (
        SELECT  *
        FROM  orders  o2
        WHERE   o2.cid = c.cid  and
            o2.pid = o1.pid and  o2.aid = o1.aid  ) )
```

3. **Get aids of agents who place orders for all customers who place orders for all products costing more than a dollar through the agent.**

```
SELECT  aid
FROM  agents  a
WHERE  not exists (
    SELECT  *
    FROM  customers  c,  products  p
    WHERE  p.price > 1  and  not exists (
        SELECT  *
        FROM  orders  x
        WHERE   x.cid = c.cid  and  x.pid = p.pid
                    and  x.aid = a.aid ) );
```

4. Get aids of agents who place orders for all customers who place orders for all products.

```
SELECT  aid   FROM  agents  a
WHERE  not exists (
    SELECT  *   FROM  customers  c
    WHERE  not exists (
        select  *    from  products  p
        where  not exists (
            select  *    from  orders  y
            where  y.cid=c.cid  and  y.pid=p.pid ) )
        and not exists (
            SELECT  *   FROM  orders  x
            WHERE   x.cid=c.cid and x.aid=a.aid));
```

C place orders for all products.

3.6 Some Advanced SQL Syntax

- **The INTERSECT and EXCEPT Operators**

subquery { UNION [ALL] subquery }

subquery { INTERSECT [ALL] subquery }

subquery { EXCEPT [ALL] subquery }

❖Definition of FROM clause

```
FROM  tableref {, tableref … }
```

❖Definition of <join>

```
INNER JOIN
| [LEFT | RIGHT | FULL] [OUTER] JOIN
```

❖Figure 3.11: Definition of tableref

```
tablename [[AS] corr_name [(colname {,…})]]
| (subquery) [AS] corr_name [(colname {,…})]
| tableref1 <join> tableref2 on condition
```

❖Figure 3.11: Definition of tableref

tablename [[AS] corr_name [(colname {,…})]]

example 1

```
SELECT  c.cname,  a.agent_name

FROM  orders  o,   customers  AS  c,

      agents  a(agent_id, agent_name, city, per)

WHERE  o.cid=c.cid and o.aid=a.agent_id ;
```

❖Figure 3.11: Definition of tableref

```
(subquery) [AS] corr_name [(colname {,…})]
```

example 2

```
SELECT  c.cid,  c.cname
FROM  customers  c,
    ( select  AVG(discnt)  AS  avg_dis
      from customers )  AS  w
WHERE  c.discnt > w.avg_dis ;
```

example 2

```
SELECT  c.cid,  c.cname
FROM  customers  c,
    ( select  AVG(discnt)  AS  avg_dis
      from customers )  AS  w
WHERE  c.discnt > w.avg_dis ;
```

```
SELECT  c.cid,  c.cname
FROM  customers  c,
    ( select  AVG(discnt)
      from customers )  AS  w ( avg_dis )
WHERE  c.discnt > w.avg_dis ;
```

❖Figure 3.11: Definition of tableref

```
tableref1 <join> tableref2 on condition
```

example 3

```
SELECT  cname,  aname

FROM  customers  JOIN  agents

      ON customers.city=agents.city ;
```

❑**Example 3.6.3:** **Retrieve all customer names where the customer places at least two orders for the same product.**

```
select  cname
from  (select  o.cid  as  spcid
        from  orders  o, orders  x
        where  o.cid = x.cid  and  o.pid = x.pid
                and  o.ordno <> x.ordno)  y,
        customers  c
where  y.spcid = c.cid;
```

❑**Other ways**

3.7 Set Functions in SQL

- **Set Functions**
  - **COUNT, MAX, MIN, SUM, AVG**

Name	Argument type	Result type	Description
COUNT	any (can be *)	numeric	count of rows
SUM	numeric	numeric	sum of arg
AVG	numeric	numeric	average of arg
MAX	char or numeric	same as arg	maximum value
MIN	char or numeric	same as arg	minimum value

3.7 Set Functions in SQL

Exp 3.7.1 Get the total dollar amount of all orders.

```
select  sum ( dollars )  as  totaldollars
from  orders
```

Exp 3.7.2 Get the total quantity of product p03 that has been ordered.

```
select  sum ( qty )  as  TOTAL
from  orders
where  pid = ‘p03’
```

3.7 Set Functions in SQL

- **Exp 3.7.3 Find the total number of customers.**

```
select  count ( cid )
from  customers
```

```
select  count ( * )
from  customers
```

- **Exp 3.7.4 Get the number of cities where customers are based.**

```
select  count ( distinct  city )
from  customers
```

```
select  count ( city )        ✗
from  customers
```

3.7 Set Functions in SQL

Exp 3.7.5 List the cid values of all customers who have a discount less than the maximum discount.

– Invalid SQL syntax

```
select  cid
from  customers
where  discnt < max ( discnt )
```

– Valid SQL statement

```
select  cid
from  customers  c1
where  discnt < all ( select  max(c2.discnt)
                      from  customers  c2 )
```

3.7 Set Functions in SQL

Exp 3.7.6 Find products ordered by at least two customers.

```
select   p.pid
from     products  p
where  2<=ALL ( select  count(distinct cid)
                from     orders  o
                where  o.pid = p.pid )
```

Example 3.6.3: Retrieve all customer names where the customer places at least two orders for the same product ?

3.7 Set Functions in SQL

- **Handling Null Values**
  - **a null value is not equal to any values (including a null value)**
  - **Set Functions must also ignore null values (including the COUNT function)**
    - **Count (*) 不存在空值问题！**
  - **the value returned by a set function acting on an empty set of values is**
    - **count() return 0**
    - **others return the null value**

3.8 Groups of Rows in SQL

- **GROUP BY clause & HAVING clause**

3.8 Groups of Rows in SQL

- **Example:** Get the total quantity for each products that has been ordered.

```
select  pid,  sum(qty) as total
from  orders
group by  pid ;
```

– the query were being performed:

```
for each distinct value V of pid in orders:
    select pid, sum(qty) as total
    from orders
    where pid=V;
end for;
```

3.8 Groups of Rows in SQL

- Invalid SQL syntax

```
select pid, cid, sum(qty) as total
from orders
group by pid;
```

- Invalid SQL syntax

```
select pid, sum(qty) as total
from orders
group by pid, cid ;
```

- the orders of SQL clause being performed
  1. First the relational products of all tables in the FROM clause are formed.
  2. From this, rows not satisfying the WHERE clause are eliminated.
  3. The remaining rows are grouped in accordance with the GROUP BY clause.
  4. The groups not satisfying the HAVING clause are eliminated.
  5. Finally, expressions in the select list are evaluated.
     - a group → a result row

3.8 Groups of Rows in SQL

- **Exp 3.8.1 Find the total product quantity ordered of each individual product by each individual agent.**

```
SELECT  pid, aid, sum(qty)
FROM  orders  o
GROUP  BY  pid, aid
```

3.8 Groups of Rows in SQL

- **Exp 3.8.2 Find the agent name and aid, and the product name and pid, together with the total quantity each agent supplies of that product to customers c002 and c003.**

```
SELECT  a.aid, a.aname, p.pid, p.pname, sum(qty)
FROM  agents  a, products  p, orders  o
WHERE  a.aid = o.aid and p.pid = o.pid and
        (o.cid = ‘c002’ or o.cid = ‘c003’)
GROUP  BY  a.aid, a.aname, p.pid, p.pname
```

3.8 Groups of Rows in SQL

- **Exp 3.8.3** **Find all product and agent IDs and the total quantity ordered of the product by the agent, when this quantity exceeds 1000.**

```
SELECT  pid,  aid,  sum(qty)  as  total
FROM        orders
GROUP BY  pid,  aid
HAVING   sum(qty) > 1000 ;
```

3.8 Groups of Rows in SQL

- **Exp 3.8.4** **Provide pid values of all products purchased by at least two customers.**

```
SELECT  pid
FROM      orders
GROUP BY  pid
HAVING   count ( distinct  cid ) >= 2 ;
```

3.8 Groups of Rows in SQL

- **Exp 3.8.5 Find the average, over all agents, of the maximum dollar sales made by each agent.**

- **Invalid SQL syntax**

```
SELECT avg(select max(dollars) from orders)
FROM      orders
GROUP BY  aid ;
```

- **Valid SQL syntax**

```
    SELECT  avg ( t.x )
    FROM   ( select  aid, max(dollars)  as  x
             from   orders
             group by  aid )   t  ;
```

3.9 A Complete Description of SQL Select

- **Reading at home**

3.10 Insert, Update, and Delete Statements

❑ **The Insert Statement**

```
INSERT INTO tablename [ ( colname, ...... ) ]
VALUES ( expr|NULL, ...... ) | subquery
```

– Example 3.10.1: insert a tuple

```
INSERT INTO orders(ordno,month,cid,aid,pid)
VALUES (1107, ‘aug’, ‘c006’, ‘a04’, ‘p01’);
```

– OR (*等价的另一条INSERT命令*)

```
INSERT INTO orders(cid,aid,pid,month,ordno)
VALUES (‘c006’, ‘a04’, ‘p01’, ‘aug’, 1107);
```

3.10 Insert, Update, and Delete Statements

– **Example 3.10.2: insert by subquery**

```
create table swcusts (
        cid char(4) not null,
        cname varchar(13),
        city varchar(20),
        discnt real );
```

```
INSERT  INTO  swcusts
    select  *
    from  customers
    where  city  in  (‘Dallas’, ‘Austin’);
```

3.10 Insert, Update, and Delete Statements

❑The Update Statement

```
UPDATE  tablename
SET colname=expr|NULL|subquery, ......
[ WHERE  search-condition ] ;
```

- **Example 3.10.3:** Give all agents in New York a 10% raise in the percent commission they earn on an order.

```
UPDATE  agents
SET  percent = 1.1 * percent
WHERE  city = ‘New York’
```

3.10 Insert, Update, and Delete Statements

- **Example 3.10.4:** **Give all customers who have total orders of more than $1000 a 10% increase in the discnt they receive.**

```
UPDATE  customers
SET  discnt = 1.1 * discnt
WHERE  cid  in  ( select  cid
                  from  orders
                  group  by  cid
                  having  sum(dollars) > 1000 );
```

3.10 Insert, Update, and Delete Statements

- **Example 3.10.5: update the discnt values in rows of the swcusts table created in Example 3.10.2 with more up-to-date discnt values from the customers table.**

```
UPDATE  swcusts
SET  discnt = ( select  discnt
                from  customers
                where cid = swcusts.cid );
```

3.10 Insert, Update, and Delete Statements

❑The Delete Statement

```
DELETE
FROM  tablename
[ WHERE  search-condition ] ;
```

Exp. 3.10.6: Delete all agents in New York.

```
DELETE  FROM  agents
WHERE  city = ‘New York’
```

3.10 Insert, Update, and Delete Statements

- **Example 3.10.7:** **Delete all agents who have total orders of less than $600.**

```
DELETE  FROM  agents
WHERE  aid  IN  (
            select  aid
            from  orders
            group  by  aid
            having  sum(dollars) < 600 );
```

Homework

- 课后作业
  - homework_2_ch3.doc

Examples of Subquery

- **Example 3.4.10: Retrieve all customer names where the customer places an order through agent a05.**

```
Select  distinct  c.cname
From  customers c
Where  EXISTS  (
    Select  *
    From  orders x
    Where  c.cid=x.cid  and  x.aid=‘a05’ );
```

Examples of Subquery

- **Example 3.4.11: Get cid values of customers who order both products p01 and p07.**

```
Select  distinct  x.cid
From  orders  x
Where  pid = ‘p01’  and  EXISTS  (
    Select  *
    From  orders  y
    Where  y.cid=x.cid  and  y.pid=‘p07’ );
```

- **Example 3.4.12:** **Retrieve all customer names where the customer *<u>does not</u>* place an order through agent a05.**

```
Select  distinct  c.cname
From  customers c
Where  NOT EXISTS  (
     Select  *
     From  orders x
     Where  c.cid=x.cid  and  x.aid=‘a05’ );
```

- **<u>NOT EXISTS can be used to implement the *MINUS* operator from relational algebra.</u>**

```
Select  distinct  c.cname
From  customers c
Where  NOT EXISTS  (
        Select  *
        From  orders x
        Where  c.cid=x.cid  and  x.aid='a05' );
```

相同的查询请求，
不同的表示方式！

```
Select distinct c.cname
From  customers c
Where  c.cid  NOT IN  (
    Select  x.cid
    From  orders  x
    Where  x.aid='a05' );
```

```
Select distinct c.cname
From  customers c
Where  c.cid  <> ALL  (
    Select  x.cid
    From  orders  x
    Where  x.aid = 'a05' );
```

Examples of Subquery

- **Example 3.9.4: Retrieve all data about customers whose cname begins with the letter ‘A’.**

```
Select  *
From  customers
Where  cname  LIKE  ‘A%’ ;
```

Examples of Subquery

- **Example 3.9.5:** Retrieve cid values of customers whose cname *does not* have a third letter equal to ‘%’.

```
Select  cid
From  customers
Where  cname  NOT LIKE  ‘_ _ \ % %’
          ESCAPE  ‘ \ ’ ;
```

Examples of Subquery

- **Example 3.9.6:** **Retrieve cid values of customers whose cname begins "Tip_" and has an arbitrary number of characters following.**

```
Select  cid
From  customers
Where  cname  LIKE  'Tip\_%'
            ESCAPE  '\' ;
```

Examples of Subquery

❑**Example 3.9.7:** Retrieve cid values of customers whose cname starts with the sequence “ab\”.

```
   Select  *
   From  customers
   Where  cname  LIKE  ‘ab
```

连续的两个转义指示字符表示‘转义指示符’自己。

```
   Select  *
   From  customers
   Where  cname LIKE ‘ab\\%’ ESCAPE ‘\’ ;
```

Review of class 4

- Data Type in SQL
- SQL statement
  - CREATE TABLE
  - Simple Select Statement
    - Relational Algebra vs SQL Query Statement
      - single table, product, natural join, theta join
    - table & column alias

Review of class 4 (cont.)

- predicate
  - distinct
  - The IN Predicate
    - expr [NOT] IN (subquery)
  - The Quantified Comparison Predicate
    - expr θ SOME|ANY|ALL (subquery)
  - The EXISTS Predicate
    - [NOT] EXISTS (subquery)
  - The BETWEEN Predicate
    - expr [NOT] BETWEEN expr1 AND expr2

Review of class 4 (cont.)

- predicate (cont.)
  - The IS NULL Predicate
    - column IS [NOT] NULL
  - The LIKE Predicate
    - column [NOT] LIKE val1 [ESCAPE val2]
      - underscore (_): any single character
      - percent (%): any sequence of zero or more characters