Chapter 7

Integrity, Views, Security, and Catalogs

- from Database Design to Physical Form
  - CREATE TABLE
    - integrity constraints (完整性约束)
  - CREATE VIEW
  - Security
    - The GRANT & REVOKE statements
  - Catalogs
    - Schemas

7.1 Integrity Constraints

- CREATE TABLE
  - Create Table Statement
  - Integrity
    - Column Constraints
    - Table Constraints
  - Referential Integrity: Foreign Key
  - Trigger
- ALTER TABLE
- DROP TABLE

CREATE TABLE statement

```
CREATE  TABLE  customers (
        cid  char(4)  not  null,
        cname  varchar(13),
        city  varchar(20),
        discnt  real,
        primary  key(cid) );
```

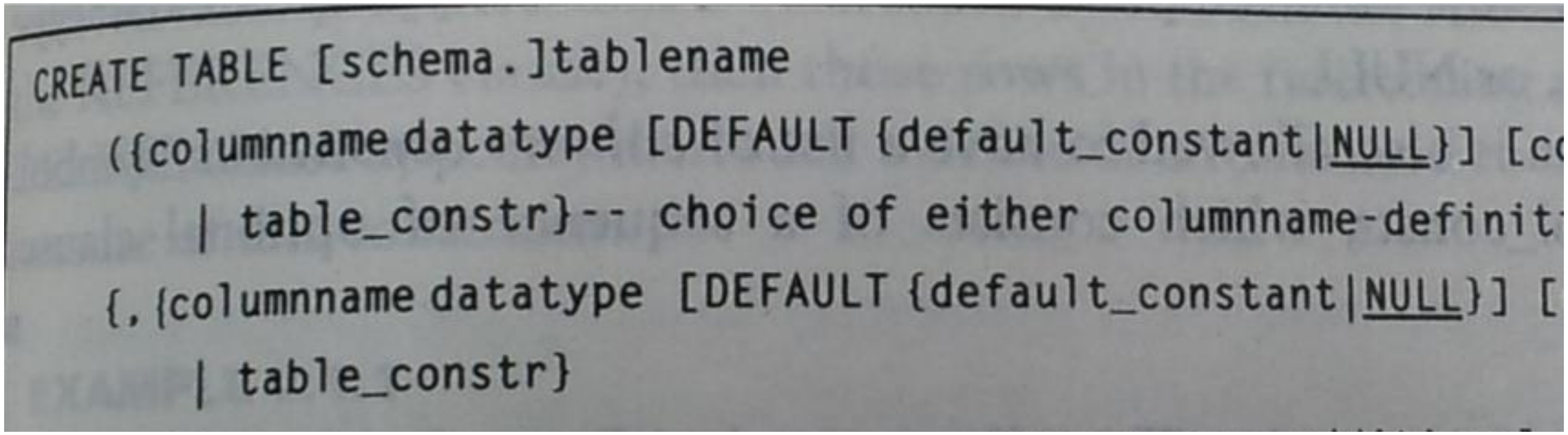
- **Integrity Constraints**
  - cid
    - primary key attribute
    - min-card is 1
  - discnt
    - min-card is 0, so can enter a tuple with no discnt value

Figure 7.1: Basic SQL Syntax for Create Table

```
    CREATE TABLE [schema.]table
     (relational_properties);
```

relational_properties:

```
{ column datatype [DEFAULT expr]
  [ col_constraint { , col_constraint } ]
}
 |
[ table_constraint { , table_constraint } ]
```



```
CREATE TABLE [schema.]tablename
    ({columnname datatype [DEFAULT {default_constant|NULL}] [co
        | table_constr}-- choice of either columnname-definit
    {, {columnname datatype [DEFAULT {default_constant|NULL}] [
        | table_constr}
```

❑ Def.7.1.1 Clauses of the Create Table Command (Figure 7.1)

- schema name & table name
- column defintion
  - column name & data type
  - an optional DEFAULT clause
    - DEFAULT { default_constant | <u>NULL</u> }
  - column constraints
- table constraints

❑Integrity Constraint (Figure 7.1, pg. 297)

- Integrity Constraints in a single column

```
{ NOT NULL   |
  [ CONSTRAINT constraint_name ]
    UNIQUE
    | PRIMARY KEY
    | CHECK ( search_condition )
    | REFERENCES table_name [ ( column_name ) ]
        [ON DELETE CASCADE|RESTRICT|SET NULL]
        [ON UPDATE CASCADE|RESTRICT|SET NULL]
}
```

Integrity Constraints in multiple columns

```
[ CONSTRAINT constraint_name ]
{ UNIQUE ( colname { , colname ... } )
  | PRIMARY KEY ( colname { , colname ... } )
  | CHECK ( search_condition )
  | FOREIGN KEY ( colname { , colname ... } )
    REFERENCES tab_name [ (colname {, ...}) ]
  [ON DELETE CASCADE|RESTRICT|SET NULL]
  [ON UPDATE CASCADE|RESTRICT|SET NULL]
}
```

7.1 Integrity Constraints

- **Def.7.1.2 Column Constraints**

1. **NOT NULL vs. DEFAULT NULL**

2. **Constraint name**
   - **optional**
   - **we can later drop the constraint with an ALTER TABLE command**

3. **UNIQUE vs. NOT NULL**
   - **candidate key: UNIQUE and NOT NULL**

7.1 Integrity Constraints

- **Def.7.1.2 Column Constraints (cont.)**

4. **PRIMARY KEY vs. NOT NULL**

5. **REFERENCES**
   - **FOREIGN KEY vs. PRIMARY KEY**
   - **values of foreign key**
   - **CASCADE | <u>RESTRICT</u> | SET NULL**

6. **CHECK**

Def.7.1.3 Table Constraints

1. Constraint name
2. UNIQUE vs. NOT NULL
   - candidate key:
     - UNIQUE and all attributes is NOT NULL
3. PRIMARY KEY vs. NOT NULL
4. FOREIGN KEY
   - FOREIGN KEY vs. PRIMARY KEY
   - values of foreign key
   - CASCADE | <u>RESTRICT</u> | SET NULL
5. CHECK

7.1 Integrity Constraints

❑ **Example 7.1.2** (Figure 7.2, pg. 299)

```
create table customers (
    cid char(4) not null,
    cname varchar(13),
    city varchar(20),
    discnt real constraint discnt_max
                check(discnt <= 15.0),
    primary key(cid)
);
```

7.1 Integrity Constraints

```
create table orders(
    ordno integer not null,
    month char(3),
    c
```

```
dollars float default 0.0
                constraint dollarsck
```

```
constraint cidref
foreign key (cid) references customers,
```

```
    primary key ( ordno
    constraint cidref foreign key (cid) references customers,
    constraint aidref foreign key (aid) references agents,
    constraint pidref foreign key (pid) references products );
```

7.1 Integrity Constraints

- we can create table orders by following statement

```
create table orders (
    ordno integer not null,
    month char (3),
    cid char (4) not null references customers,
    aid char (3) not [illegible]ll references agents,
    [illegible]
    ( [illegible] 0.0 ),
    primary key (ordno) );
```

cid char (4) not null references customers,

7.1 Integrity Constraints

- **Primary Keys, Foreign Keys, and Referential Integrity**

Primary key

cid	cname	city	d
c001	...	...	...
c002	...	...	...
c003	...	...	...
c004	...	...	...
c005	...	...	...
c006	...	...	...

Foreign key

ordno	month	cid	......
1011	jan	c001	......
1019	feb	c001	......
1017	feb	c001	......
[illegible]	mar	c001	......
1022	mar	c001	......
1013	jan	c002	......
1026	may	c002	......
1015	jan	c003	......
1014	jan	c003	......
1021	feb	c004	......
1016	jan	c006	......
1020	feb	c006	......
1024	mar	c006	......

7.1 Integrity Constraints

- **Def.7.1.4 Foreign Key, Referential Integrity**

- **Foreign Key**
  - **A set of columns F in table T1 is defined as a *foreign key* of T1 if the combination of values of F in any row is required to either contain at least one null value, or else to match the value combination of a set of columns P representing a candidate or primary key of a referenced table T2.**

7.1 Integrity Constraints

- **Referential Integrity**
  - **the columns of F in any row of T1 must either (1) have null values in at least one column that allows null values, or (2) have no null values and be equal to the value combination of P on some row of T2.**

❑ Example

- primary key: red character
- foreign key: *underline*

EMPLOYEE (fname, lname, ssn, address, salary, *superssn*, dno)

DEPARTMENT (dname, dno, *ssn*, mgrstartdate)

DEPT_LOCATION (*dno*, dcity)

PROJECT (pname, pno, pcity, *dno*)

WORKS_ON (*ssn*, *pno*, hours)

DEPENDENT(*ssn*, dependent-first-name, sex, bdate, relationship)

Example of foreign key

- ❑ **EMPLOYEE:**

  Emp (fname, lname, ssn, address, salary, <u>*superssn*</u>, <u>dno</u>)
- ❑ **DEPARTMENT:**

  Dept (dname, dno, <u>*ssn*</u>, mgrstartdate)

```
CREATE TABLE Dept (
    dno  char(4) PRIMARY KEY, ……);

CREATE TABLE Emp (
    ssn char(8) PRIMARY KEY,
    dno char(4) REFERENCES Dept,
    ……);
```

在对Emp表进行数据的增、删、改操作时，保证外键dno的引用完整性。

Example of foreign

```
CREATE TABLE Emp (
    ssn char(8) PRIMARY KEY,
    dno char(4) REFERENCES Dept,
    ......);
```

但在对Dept表进行数据的增、删、改操作时，不能保证Emp表中外键dno的正确性。

```
CREATE TABLE Emp (
    ssn char(8) PRIMARY KEY,
    dno char(4)
    REFERENCES Dept ON DELETE  RESTRICT,
    ......);
```

如果某部门中有职工，那么将不允许在Dept表中删除该部门元组，从而保证了Emp表中外键dno的引用完整性。

Example of foreign key

```
CREATE TABLE Emp (
    ssn char(8) PRIMARY KEY,
    dno char(4) REFERENCES Dept,
    ……);
```

```
CREATE TABLE Emp (
    ssn char(8) PRIMARY KEY,
    dno char(4)
    REFERENCES Dept ON DELETE  CASCADE,
    ……);
```

如果某部门中有职工，那么在Dept表中删除该部门元组的同时，将从Emp表中删除该部门的所有职工。

Example of foreign key

```
CREATE TABLE Emp (
    ssn char(8) PRIMARY KEY,
    dno char(4) REFERENCES Dept,
    ……);
```

```
CREATE TABLE Emp (
    ssn char(8) PRIMARY KEY,
    dno char(4)
    REFERENCES Dept ON DELETE  SET NULL,
    ……);
```

如果某部门中有职工，那么在Dept表中删除该部门元组的同时，在Emp表中将该部门所有职工的dno置为NULL。

Foreign Key Constraints: Product Variations

(Figure 7.6)

	ORACLE INFORMIX	DB2 UDB	X/Open Full SQL-99
ON DELETE ...	CASCADE SET NULL	NO ACTION CASCADE SET NULL RESTRICT	NO ACTION CASCADE SET NULL SET DEFAULT RESTRICT
Without ON DELETE	NO ACTION effect	NO ACTION effect	NO ACTION effect

7.1 Integrity Constraints

- **The Alter Table Statement**
  - **change the table structure & integrity constraints**
  - **Figure 7.7 ~ 7.9 (pg. 306)**
    - **column**
      - **Add**
      - **Drop**
      - **Modify**
    - **constraints**
      - **Add**
      - **Drop**

7.1 Integrity Constraints

❑ **Database Trigger**

- **A database trigger is procedural code that is automatically executed in response to certain events on a particular table or view in a database.**
- **The trigger is mostly used for maintaining the integrity of the information on the database. For example, when a new record (a new worker) is added to the employees table, new records should also be created in the tables of the taxes, vacations and salaries.**

7.1 Integrity Constraints

- **Trigger (Figure 7.10, pg. 307) (触发器)**

```
CREATE TRIGGER trigger_name { BEFORE|AFTER }
    {  INSERT | DELETE
        | UPDATE [ OF colname { , colname ... } ] }
    ON table_name
    [ REFERENCING corr_name_def { , ...... } ]
    [ FOR EACH ROW | FOR EACH STATEMENT ]
        [ WHEN ( search_condition ) ]
        { statement
        | BEGIN ATOMIC statement; { ... } END
```

7.1 Integrity Constraints

The ***corr_name_def*** that defines a correlation name follows:

```
{       OLD [ ROW ] [ AS ] old_row_corr_name
      | NEW [ ROW ] [ AS ] new_row_corr_name
      | OLD TABLE [ AS ] old_table_corr_name
      | NEW TABLE [ AS ] new_table_corr_name    }
```

```
CREATE TRIGGER trigger_name { BEFORE | AFTER }
    { INSERT | DELETE
     | UPDATE [ OF colname { , colname ... } ] }
      ON table_name
    [ REFERENCING corr_name_def { , ...... } ]
    [ FOR EACH ROW | FOR EACH STATEMENT ]
    [ WHEN ( search_condition ) ]
        { statement
        | BEGIN ATOMIC statement; { statement; ... } END
```

触发事件

触发方式

结果事件

two types of UPDATE triggers

①**Row Level Trigger**

- **This gets executed before or after any column value of a row changes**

②**Column Level Trigger**

- **This gets executed before or after the specified column changes**

four types of triggers

①For Each Row Type

- This trigger gets executed once for each row of the result set affected by an insert/update/delete
- before/after

②For Each Statement Type

- This trigger gets executed only once for the entire result set, but fires each time the statement is executed.
- before/after

❑ Example 7.1.5

- Use a trigger to CHECK that the discnt value of a new customers row does not exceed 15.0.

```
create  trigger  discnt_max
   after  insert  on  customers
   referencing  new  as  x
   for  each  row  when  ( x.discnt > 15.0 )
   begin
      raise_application_error(-20003, “invalid
      discount attempted on insert”);
   end;
```

❑ Example 7.1.6: 删除一个客户元组时，需要将该客户所有订单上的cid置为空值(set null)

```
create  trigger  foreign_cid
    after  delete  on  customers
    referencing  old  as  old_custom
    for  each  row
    begin
        update  orders
            set  cid = null
            where  cid = :old_custom.cid ;
    end;
```

7.2 Creating Views

- **View**
  - **idea ?**
    - **The data retrieved by any SQL SELECT statement is in the form of a table.**
    - **We want to use this TABLE in FROM clause of other Select statement.**
  - **Method?**
    - **Subquery in the FROM clause (Fig 3.11)**
    - **Creating Views**

7.2 Creating Views

- **View Table (or View)**
  - **Definition**
    - **It is a table that results from a subquery, but which has its own name**
      - **table name & attributes name**
    - **It can be used in most ways as a Base Table created by SQL CREATE TABLE statement**

7.2 Creating Views

- **View Table (or View)**
  - **Property**
    - **no data storage in its own right, just window on data it selects from**
    - **so, it is regarded as a Virtual Table**
  - **Weakness**
    - **limits to View Updates**

❑ Create View statement (Figure 7.13, pg. 314)

```
CREATE VIEW view_name [(col_name {, ……})]
    AS  subquery
    [ WITH CHECK OPTION ]
```

1. List of column names
   - Can leave out list of colnames if
     - ➢ target list (Select clause) of subquery has colnames that require no qualifiers.
   - Need to give names for expressions or for ambiguous colnames.
   - Can rename column in any way at all.

Create View statement (cont.)

```
CREATE VIEW view_name [(col_name {, ……})]
    AS  subquery
    [ WITH CHECK OPTION ]
```

2. subquery
   - Figure 3.14
   - no ORDER BY clause

3. WITH CHECK OPTION
   - will not permit an update or insert through a updatable view if
     - ➢ they result in rows that would be invisible to the view Subquery.

Create View statement (cont.)

```
CREATE VIEW view_name [(col_name {, ……})]
    AS  subquery
    [ WITH CHECK OPTION ]
```

- **Nested View definition**
  - create a view based on other views

7.2 Creating Views

```
CREATE VIEW view_name [(col_name {, ……})]
    AS  subquery
    [ WITH CHECK OPTION ]
```

- **Executing of Create View statement**
  - **The definition of the view is placed in the system catalogs as a <u>distinct object</u> of the database.**
  - **No data is retrieved or stored.**

7.2 Creating Views

```
CREATE VIEW view_name [(col_name {, ……})]
    AS  subquery
    [ WITH CHECK OPTION ]
```

- **Operations through Views**
  - **query modification**
    - **A query or update statement that accesses a view can be modified.**
    - **So that the modified query or update actually performs accesses on base tables.**

7.2 Creating Views

- **Privileges on Views**

  1) **The user creating the view becomes the owner of the view**

  2) **The owner is given update privileges on the view, assume:**
     - **the view is *updatable***
     - **the user has *the needed update privileges on the base table* on which the view is defined.**

7.2 Creating Views

- **Example 7.2.1 Create View**

```
CREATE VIEW  agentorders (ordno, month, cid, aid,
        pid, qty, charge, aname, acity, percent)
    AS SELECT  o.ordno, o.month, o.cid, o.aid, o.pid,
            o.qty, o.dollars, a.aname, a.city, a.percent
        FROM  orders o, agents a
        WHERE  o.aid = a.aid;
```

7.2 Creating Views

```
CREATE VIEW  agentorders (ordno, month, cid, aid,
        pid, qty, charge, aname, acity, percent)
    AS SELECT  o.ordno, o.month, o.cid, o.aid, o.pid,
            o.qty, o.dollars, a.aname, a.city, a.percent
        FROM  orders o, agents a
        WHERE  o.aid = a.aid;
```

- **Example 7.2.2 Query on View**

```
SELECT  sum (charge)
FROM    agentorders
WHERE  acity = 'Toledo';
```

7.2 Creating Views

❑ **Example 7.2.2 (cont.) Query Modification**

```
SELECT  sum (charge)
FROM    agentorders
WHERE   acity = 'Toledo';
```

```
SELECT  sum(o.dollars)
FROM    orders o, agents a
WHERE   o.aid = a.aid and a.city = 'Toledo';
```

```
CREATE VIEW  agentorders (ordno, month, cid, aid,
        pid, qty, charge, aname, acity, percent)
    AS SELECT  o.ordno, o.month, o.cid, o.aid, o.pid,
            o.qty, o.dollars, a.aname, a.city, a.percent
        FROM  orders o, agents a
        WHERE  o.aid = a.aid;
```

- **Example:** Agentorders view can be created by following statement.

```
CREATE VIEW  agentorders
    AS SELECT  o.ordno,  o.month,  o.cid,  o.aid,
                o.pid, o.qty,  o.dollars,  a.aname,
                a.city,  a.percent
        FROM  orders o, agents a
        WHERE  o.aid = a.aid;
```

❑Example 7.2.3: illegal view definition

```
CREATE VIEW  cacities  AS
      SELECT  c.city,  a.city
      FROM  customers c, agents a, orders o
      WHERE  c.cid = o.cid and a.aid = o.aid;
```

– legal view definition

```
CREATE VIEW  cacities ( ccity, acity )  AS
   SELECT  c.city,  a.city
   FROM  customers c, agents a, orders o
   WHERE  c.cid = o.cid and a.aid = o.aid;
```

❑ Exp 7.2.4 Create Views with 'WITH CHECK OPTION'

Assume: No CHECK clause on discnt of customers

```
CREATE VIEW  custs  AS
        SELECT  *
        FROM  customers
        WHERE  discnt <= 15.0
        WITH CHECK OPTION;
```

❑ Now cannot insert/update a row into custs with discnt > 15.0:

```
insert into custs values('c009', 'AM', 'Kyoto', 16.0);
```

This insert will fails.

- Consider the update:

```
update  custs  set  discnt = discnt + 4.0;
```

- This update fails for customer c002 on the basis of the values of Figure 2.2

cid	cname	city	discnt
c001	TipTop	Duluth	10.00
c002	Basics	Dallas	12.00
c003	Allied	Dallas	8.00
c004	ACME	Duluth	8.00
c006	ACME	Kyoto	0.00

- Note, this insert and update statement can execute on the base table customers.

7.2 Creating Views

❑Example 7.2.5 Nested Views

```
CREATE VIEW  acorders (ordno, month, cid, aid,
        pid, qty, dollars, aname, cname)
    AS SELECT  ao.ordno, ao.month, ao.cid, ao.aid,
            ao.pid, ao.qty, ao.charge, ao.aname,
            c.cname
        FROM  agentorders  ao,  customers  c
        WHERE  ao.cid = c.cid;
```

7.2 Creating Views

- **Listing Defined Views**
  - **In ORACLE**

```
select view_name from user_views;
DESCRIBE  { view_name | table_name };
```

- **Dropping Tables and Views**

```
DROP { TABLE table_name | VIEW view_name }
      { CASCADE | RESTRICT };
```

  - **When a view table is dropped, no rows are dropped.**

7.2 Creating Views

- **Updatable and Read-Only Views**
  - **The problem**
    - **How do we translate updates on the View into changes on the base tables?**
  - **Figure 7.15**
    - **Restrictions on the Subquery Clause for an Updatable View**

❑ A view table is said to be updatable when the following conditions hold for its Subquery clause.

1) The FROM clause of the Subquery must contain only a single table, and if that table is a view table it must also be an updatable view table.
2) Neither the GROUP BY nor HAVING clause is present.
3) The DISTINCT keyword is not specified.
4) The WHERE clause does not contain a Subquery that references any table in the FROM clause, directly or indirectly via views.
5) All result columns of the Subquery are simple column names: no expressions, no column name appears more than once.

7.2 Creating Views

- Example 7.2.6, 7.2.7

```
    CREATE VIEW colocated AS
       select cid, cname, aid, aname,
                  a.city as acity
       from customers c, agents a
       where c.city = a.city;
```

```
CREATE VIEW agentsales (aid, totsales)
   AS  select aid, sum(dollars)
       from orders
       group by aid;
```

7.2 Creating Views

- **Updatable Views in ORACLE**
  - **can update join views if**
    - **join is N-1 (many-to-one), and**
    - **columns of the view contains the primary key of the table on N side**

❑ example 7.2.1:

the view contains the primary key of the table on N side (orders)

```
CREATE VIEW  agentorders (ordno, month, cid, aid,
        pid, qty, charge, aname, acity, percent)
AS SELECT  o.ordno, o.month, o.cid, o.aid, o.pid,
        o.qty, o.dollars, a.aname, a.city, a.percent
    FROM  orders o, agents a
    WHERE  o.aid = a.aid;
```

join is many to one (orders to agents)

❑ example 7.2.1:

```
CREATE VIEW  agentorders (ordno, month, cid, aid,
        pid, qty, charge, aname, acity, percent)
AS SELECT  o.ordno, o.month, o.cid, o.aid, o.pid,
        o.qty, o.dollars, a.aname, a.city, a.percent
    FROM  orders o, agents a
    WHERE  o.aid = a.aid;
```

a) we can only update the columns that map one-to-one with the orders table (on N side)

b) not the aname, acity, or percent columns (on 1 side), and also not the aid column

7.2 Creating Views

– Display updatable column of this view

```
select  column_name, updatable
from  user_updatable_columns
where table_name = ‘AGENTORDERS’
```

7.2 Creating Views

- **The Value of Views**

1. **Views provide a way to make complex, commonly issued queries easier to compose.**
2. **Views allow obsolete tables, and programs that reference them, to survive reorganization.**
    - **program-data independence (数据独立性)**
3. **Views add a security aspect to allow different users to see the same data in different ways.**

7.3 Security

- **The Grant Statement in SQL**

```
GRANT {ALL PRIVILEGES|privilege {, privilege … }}
    ON [ TABLE ] tablename | viewname
    TO { PUBLIC | user-name { , user-name … } }
    [ WITH GRANT OPTION ]
```

- **used by the owner of a table**
  - **the owner of a table has ALL PRIVILEGES on the table.**
  - **other user can not access the table if it does not have the PRIVILEGES on the table.**
- **column privileges can be implemented through views.**

❑ The Grant Statement in SQL (cont.)

```
GRANT {ALL PRIVILEGES|privilege {, privilege …}}
    ON [ TABLE ] tablename | viewname
    TO { PUBLIC | user-name { , user-name … } }
    [ WITH GRANT OPTION ]
```

- **privileges**
  - SELECT, DELETE, INSERT
  - UPDATE [col_name {, col_name ...}]
  - REFERENCES [col_name {, col_name ...}]
- **PUBLIC**
- **WITH GRANT OPTION**

- **Example 7.3.1, 7.3.2**
  - **grant select on customers to <u>eoneil</u>;**
  - **grant select, update, insert on orders to <u>eoneil</u>;**
  - **grant all privileges on products to <u>eoneil</u>;**

- **Example 7.3.3**
  - **create view custview as**
    select cid, cname, city from customers;

  - **grant select, delete, insert,**
    update(cname, city) on custview
    to <u>eoneil</u>;

- **<u>The user eoneil does not need these privileges on the base table customres.</u>**

7.3 Security

❑ The SQL statement to revoke privileges

```
REVOKE { ALL PRIVILEGES
                | privilege {, privilege …} }
    ON  tablename | viewname
    FROM { PUBLIC | user-name {, user-name … }}
    { CASCADE | RESTRICT }
```

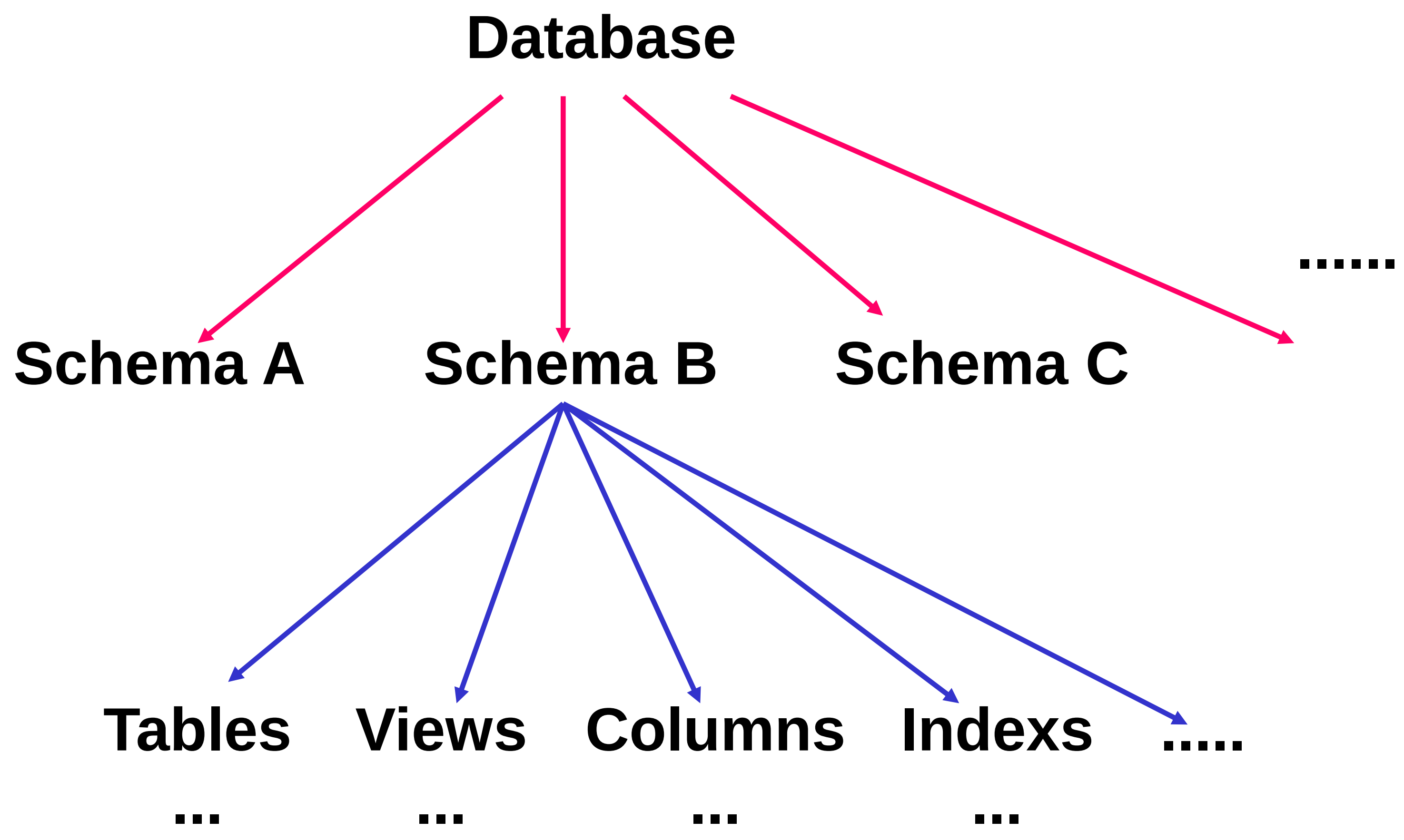
- revoke privileges earlier granted to a user.

7.4 System Catalogs and Schemas

- **Idea**
  - **data dictionary**
    - **all objects created by SQL commands are listed as objects in tables maintained by the system.**
  - **DBA visiting another site would look at catalog tables (meta-data) to find out**
    - **what tables exist**
    - **what columns in them**
    - **……**

7.4 System Catalogs and Schemas

- **Schemas**


Database
Schema A
Schema B
Schema C
......
Tables
Views
Columns
Indexs
.....
...
...
...
...