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

- **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 | NULL }
    - column constraints
  - table constraints

❑ Integrity Constraint (Figure 7.1, pg. 411)

- 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

- **Trigger** (Figure 7.10, pg. 425) (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
```

```
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

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**

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.**

7.3 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