Individuell uppgift 2, Sammanfattning

Yasir Riyadh Jabbar (TIDAA KTH)

Kurslitteratur: Kapitel 4.6, 5.1-5.7.1

CHAPTER 4: The Relational Model / Views

This section shows how the view model represents the subset of database and it is just virtual table (or combination of different tables) that similar to the real (base) tables.

- The attributes in view can be taken from one or more existence tables.
- View always displays recent (updated) data.
- It allows database models to be customized (for example, to make the displayed tables familiar and comfortable to the user, presents some calculated data, or gives a summary information).
- It keeps the data more secure (for example, to hide some parts of database from specific users).

CHAPTER 5: Relational DBMS and SQL

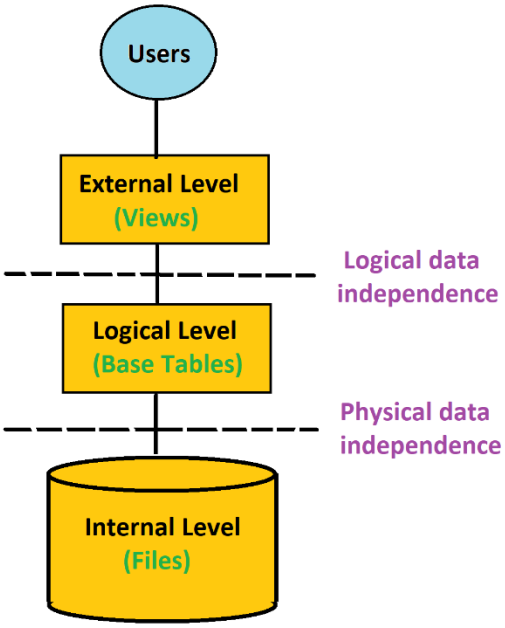
5.1 Structured Query Language (SQL)

This section shows how the standard programming language SQL for relational databases can be classified into two parts:

1. data definition language (DDL) commands that is used for design and modify DATABASE schema.
2. data manipulation language (DML) commands that is used to save and retrieve data from database.

5.2 Architecture of Relational DBMS

This section explains how the relational DBMS can support the standard 3-level architecture (external, logical, and internal level) for database as shown in the figure below:



Here is a brief explanation about the purpose of these two data independence layers:

1. Logical data independence: It contains base tables which created by DBA (**using CREATE TABLE command**). Base table can have indexes that are used to speed up loading and provide positions for data records based on the value in one or more attributes.
2. Physical data independence: indexes and base tables are represented in files. When base tables are created, DBA creates views for users (**using CREATE VIEW command**).

5.3 1. Data definition language (DDL) commands

This section describes briefly DDL commands that are used for design and modify database.

Data Types:

In MySQL, there are many data types to declare each attribute in the table:

- **INT(n)**: used for integer attributes with n is number of digits.
- **DECIMAL (m, n)**: used for floating attributes, where m is precision and n is number of digits after decimal point.
- **CHAR(n)** and **VARCHAR(n)**: used for string attributes, where n is maximum number of characters.
- **DATE**: used for date in format 'YYYY-MM-DD'.
- **TIME**: used for time in format 'hh:mm:ss'.

Constraints:

The purpose of specifying constraints (specified after the column name and data type) is to protect database system from data entry errors. For column constraints, there are many options and be used as constraints:

NULL/**NOT NULL**: used to allow the column to have null values or not.

UNIQUE: used to prevent new record that has the same value in that attribute.

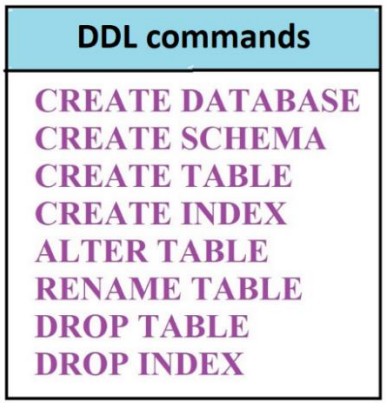
PRIMARY KEY: used to declare that primary key is not composite key.

FOREIGN KEY: used to declare referenced table where column or column combination appears.

CHECK: used to specify condition that rows are not allowed to violate.

DEFAULT: used when the inserted record does not have value, then it will be given default value automatically.

I will summarize the DDL commands with simple example giving for each command:

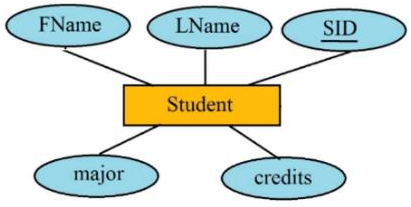


CREATE DATABASE and **USE** commands used to create a new database and set it as default database, for example:

```
CREATE DATABASE Company;
USE Company;
```

CREATE TABLE This command used to create the base tables that consist of column names and data types, for example:

```
CREATE TABLE Student (
SID VARCHAR(6) PRIMARY KEY,
LName VARCHAR(20) NOT NULL,
```



```
FName VARCHAR(20) NOT NULL,
major VARCHAR(10),
credits INT DEFAULT 0,
```

CREATE INDEX: The purpose of the index is to keep track of values of indexed column and the records that contain those values. For example, to create index on the LName column of the student table:

```
CREATE INDEX Student_LName_ix ON Student (LName);
```

ALTER TABLE: This command used to add, drops or rename columns, data types or constrains. For examples:

```
ALTER TABLE Student ADD address VARCHAR(30);    -- to add new column
ALTER TABLE Student DROP COLUMN address;        -- to delete column
ALTER TABLE Student RENAME COLUMN SID TO studentNo;    -- to rename column
```

RENAME TABLE: This command used to change name of a table. For example:

```
RENAME TABLE Faculty TO Collage;
```

DROP TABLE and **DROP INDEX**: These commands used to delete table and destroy index.

5.4 Data manipulation language (DML) commands

This section describes briefly DML commands that are used for manipulating database

SELECT (for one table): used to retrieve data from table. Examples:

```
1)  SELECT LName, FName, credits        -- select without condition
    FROM Student

2)  SELECT *                            -- select with all columns (*)
    FROM Student

3)  SELECT LName, FName, credits        -- select with simple condition
    FROM Student
    WHERE major = 'Math';

4)  SELECT DISTINCT classNo             -- DISTINCT used here to eliminate the duplicates
    FROM Enroll;

5)  SELECT FName AS name    -- to sort FName in ascending order ASC (or descending DESC)
    FROM Student
    ORDER BY FName;

6)  SELECT LName, FName                 -- select with multiple conditions
    FROM Student
    WHERE major = 'Math' AND credits > 30;
```

SELECT (for multiple tables):

```
1)  SELECT SID, grade               -- natural Join in ascending order
    FROM Class, Enroll
    WHERE FID = 'F10' AND Class.classNo = Enroll.classNo
    ORDER BY SID ASC;

2)  SELECT classNo FROM Class   -- for subquery with equality
```

```
        WHERE FID = (SELECT FID FROM Faculty
                        WHERE name = 'Sam' AND department = 'Math');
3)      SELECT name, FID FROM Faculty       -- using IN (set of values rather than single value)
        WHERE FID IN (SELECT FID
                        FROM Class   WHERE room = 'A21');
4)      SELECT LName, FName FROM Student        -- set of values not exist in condition
        WHERE NOT EXISTS (SELECT * FROM Enroll
                            WHERE Student.SID = Enroll.SID
                            AND classNo = 'C101A');
```

SELECT (with union):

```
SELECT FID FROM Faculty WHERE department = 'Math'
UNION           -- Get IDs of lecturers who are in Math dep or who teach in room A21
SELECT FID FROM Class WHERE room ='A21';
```

SELECT (with aggregate Functions):

Aggregate Functions
COUNT
SUM
AVG
MAX
MIN

```
1) SELECT COUNT (DISTINCT SID) FROM Enroll
   WHERE classNo = 'A103';    --get No of values in column SID
2) SELECT SID   FROM Student
   WHERE credits < (SELECT AVG(credits)
                    FROM Student);  --get IDs who has less than AVG of credits
```

SELECT (with LIKE, NULL): '%' stands for any length of characters and '_' for one character

1) SELECT * FROM Class --get only details how classNo start with 'MTH'

WHERE classNo LIKE 'MTH%';

2) SELECT SID FROM Enroll -- gets IDs of all students whose grades are missing

WHERE grade IS NULL;

Note: we can also use NOT LIKE and NOT NULL

UPDATE / SET: used to change values in rows in one table. For examples:

1) UPDATE Student SET major = 'Math' --update one value in one row

WHERE SID = 'S102';

2) UPDATE Faculty SET department = 'CE', rank = 'Lec' --update many values in only one row

WHERE name = 'John';

3) UPDATE Enroll SET grade = 'A' --update one value in many rows

WHERE classNo = 'CSC201A';

4) UPDATE Student SET credits = credits + 2; --update only one value in all rows

INSERT / INTO / VALUES: used to add one or more rows. For examples:

1) INSERT INTO Student -- add one row

VALUES ('S103', 'Sam', 'John', 'Math', 15);

2) INSERT INTO Faculty (FID, name, department, rank) -- add one row with specified attributes

VALUES ('F30', 'SAM', 'CSC', 'Lec');

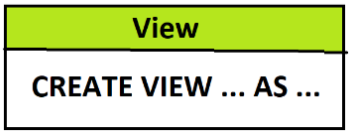
Note: we cannot add value for IDENTITY attribute

DELETE: used to delete one or more rows. Examples:

```
1) DELETE FROM Student   --delete only one row
   WHERE SID = 'S102';
2) DELETE FROM Student   --delete all rows
3)   DELETE FROM Enroll
     WHERE SID = (SELECT SID   --delete row with sub query
                  FROM Student
                  WHERE LName = 'John' AND FName = 'Sam');
```

Creating and Using Views:

Views in general provide simple and customized tables with hiding some data (creates virtual table by selecting some rows and columns to show). For example:




```
-- to create new view to show class numbers and rooms without any condition
CREATE VIEW NewClass AS
        SELECT classNo, room
        FROM Class
-- to create new view to show IDs and names for all Math majors (with condition)
CREATE VIEW Myview (LN, FN, id) AS
        SELECT LName, FName, SID FROM Student
```

```
    WHERE major = 'Math';

-- to create new view for class from two tables

CREATE VIEW ListView AS

    SELECT Student.SID,LName, FName

    FROM Enroll, Student

    WHERE classNo = 'CSC101' AND Enroll.SID = Student.SID;
```

5.5 Active Databases

database is called active database when the content is monitored to prevent illegal operations and this is done with the help of constraints and triggers.

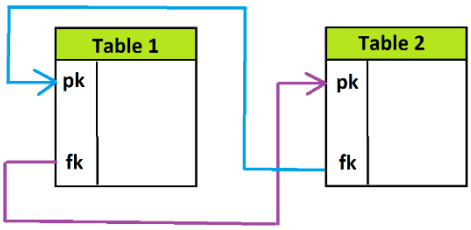
Using Constraints:

They are checked after each statements insert, delete, or update to ensure that new operation is correct. Also, we can enable or disable constraints for some situations.

ALTER TABLE … DISABLE CONSTRAINT …;

ALTER TABLE … ENABLE CONSTRAINT …;

Problem:



It is difficult to create two tables and to insert data if both tables contain foreign keys that reference each other's primary keys as shown in figure above.

Solution:

1) creating table-2 without foreign key constraint

2) creating table-2 referencing the table-1
3) adding the foreign key constraint to table-2 with ALTER TABLE command

Using SQL Triggers:

Also used to control the contents of database for valid operations. Triger in general is stored procedure and reacts to event or some operations that have been made in database. It consists of three parts: **event** (type of change that made in database contents), **condition** and **action** (procedure that invokes when event occurred and condition became true).

5.6 COMMIT and ROLLBACK Statements

There are two statements make the SQL transaction ends:

1) COMMIT statement: is used to make the changes that made in database (from beginning of current transaction until current point) permanent.

2) ROLLBACK statement: is used to undo any changes that made by the current transaction.

5.7 Temporal Databases and SQL

It contains data varying corresponding to current time. So, any change in the time or changing in the old values, this leads to update rows and reflect this change in time.

As example, in university database its important to record the time of student actions (time of register, drop, and de-enroll). The time from registration date to dropped date represents the valid time (real time) but it differs from database changed time. These times are called valid start time and valid end time and they are supported with datatypes (DATE, TIMESTAMP, TIMESTAMP WITH TIME ZONE, and TIMESTAMP WITH LOCAL TIME ZONE).

For simple example, the following fields can be used for manipulation of the valid time data in the creation of a table-1:

```
CREATE TABLE table-1 (
        …
        start_time TIMESTAMP,
        end_time TIMESTAMP,
```

```
    PERIOD FOR user_valid_time (start_time, end_time));
    …
    )
```

Here table-1 includes two attributes that store the dates, start_time and end_time. The PERIOD FOR statment uses these attributes values as beginning and ending time for user_valid_time.