ICT1222 Database Management Systems Practicum

SQL Views

SQL Views

- A view is a virtual table based on the result-set of an SQL statement.
- A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.
- You can add SQL functions, WHERE, and JOIN statements to a view and present the data as if the data were coming from one single table
- A view always shows up-to-date data! The database engine recreates the data, using the view's SQL statement, every time a user queries a view.

Advantages of database view

- A database view allows you to simplify complex queries
- A database view helps limit data access to specific users
- A database view provides extra security layer
- A database view enables computed columns
- A database view enables backward compatibility

Disadvantages of database view

- Performance:
  - Querying data from a database view can be slow especially if the view is created based on other views.
- Tables dependency:
  - You create a view based on underlying tables of the a database.
  - Whenever you change the structure of those tables that view associated with, you have to change the view as well.

Creating a VIEW

```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name/view_name
WHERE condition;
```

- In the SELECT statement, you can query data from any table or view that exists in the database.
- There are several rules that the SELECT statement must follow:
- The SELECT statement can contain a subquery in WHERE claus but not in the FROM clause.
- The SELECT statement cannot refer to any variables including local variables, user variables, and session variables.
- The SELECT statement cannot refer to the parameters of prepared statements

Updating a VIEW

```
CREATE OR REPLACE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name/view_name
WHERE condition;
```

- In "CREATE OR RPLACE
  - If view exists it modifies the view unless it creates a new one
- Or you can use "ALTER" command, like in tables

Dropping a VIEW

```
DROP VIEW view_name;
```

- Proper way
  - DROP VIEW [IF EXISTS] [database_name].[view_name]

View WITH CHECK OPTION

- The WITH CHECK OPTION clause can be given for an updatable view to prevent inserts to rows for which the WHERE clause in the *select_statement* is not true.
- It also prevents updates to rows for which the WHERE clause is true but the update would cause it to be not true (in other words, it prevents visible rows from being updated to nonvisible rows).
- In a WITH CHECK OPTION clause for an updatable view, the LOCAL and CASCADED keywords determine the scope of check testing when the view is defined in terms of another view.
- When neither keyword is given, the default is CASCADED.

View WITH CHECK OPTION

- As of MySQL 5.7.6, WITH CHECK OPTION testing is standard-compliant
- With LOCAL, the view WHERE clause is checked, then checking recurses to underlying views and applies the same rules.
- With CASCADED, the view WHERE clause is checked, then checking recurses to underlying views, adds WITH CASCADED CHECK OPTION to them (for purposes of the check; their definitions remain unchanged), and applies the same rules.
- With no check option, the view WHERE clause is not checked, then checking recurses to underlying views, and applies the same rules.

View WITH CHECK OPTION

```
CREATE TABLE t1 (a INT);

CREATE VIEW v1 AS SELECT * FROM t1
WHERE a < 2
WITH CHECK OPTION;

CREATE VIEW v2 AS SELECT * FROM v1
WHERE a > 0
WITH LOCAL CHECK OPTION;
CREATE VIEW v3 AS SELECT * FROM v1
WHERE a > 0
WITH CASCADED CHECK OPTION;
```

View WITH CHECK OPTION

- INSERT INTO v2 VALUES (1);
- INSERT INTO v2 VALUES (2);

- INSERT INTO v3 VALUES (1);
- INSERT INTO v3 VALUES (2);