More SQL

Database Systems (CSCI 440)

Fall 2014

Patrick Donnelly

Montana State University

Administrivia

Midterm #1:

in-class Friday, October 3, 2014

Project Proposal:

by 5pm: Monday, October 8, 2014

Reading:

Chapter 5.2-5.5

Assertions

CREATE ASSERTION

`CREATE ASSERTION` allows specification of additional types of constraints outside scope of built-in relational model constraints.

- Specify query that selects any tuples that violate the condition
- Use only in cases where it is not possible to use CHECK on attributes and domains

```
CREATE ASSERTION SALARY_CONSTRAINT
CHECK ( NOT EXISTS   ( SELECT      *
                       FROM        EMPLOYEE E, EMPLOYEE M,
                                   DEPARTMENT D
                       WHERE       E.Salary>M.Salary
                                   AND E.Dno=D.Dnumber
                                   AND D.Mgr_ssn=M.Ssn ) );
```

Triggers

CREATE TRIGGER

`CREATE TRIGGER` allows specification of automatic actions that database system will perform when certain events and conditions occur.

- Used to monitor the database

Typical trigger has three components:

- Event(s)
- Condition
- Action

Trigger Example

Example

```
CREATE TRIGGER uocheck BEFORE UPDATE ON account
FOR EACH ROW
BEGIN
    IF NEW.amount < 0 THEN
        SET NEW.amount = 0;
    ELSEIF NEW.amount > 100 THEN
        SET NEW.amount = 100;
    END IF;
END;
```

View Example

Definition

Concept of a **view** in SQL is a single virtual table derived from other tables.

`CREATE VIEW`

`CREATE VIEW` command requires a (virtual) table name, list of attribute names, and a query to specify the contents of the view.

Views

```
V1:     CREATE VIEW     WORKS_ON1
        AS SELECT       Fname, Lname, Pname, Hours
           FROM         EMPLOYEE, PROJECT, WORKS_ON
           WHERE        Ssn=Essn AND Pno=Pnumber;
```

WORKS_ON1

Fname	Lname	Pname	Hours

Views

```
V1:   CREATE VIEW   WORKS_ON1
      AS SELECT     Fname, Lname, Pname, Hours
         FROM       EMPLOYEE, PROJECT, WORKS_ON
         WHERE      Ssn=Essn AND Pno=Pnumber;
```

WORKS_ON1

Fname	Lname	Pname	Hours

```
V2:   CREATE VIEW   DEPT_INFO(Dept_name, No_of_emps, Total_sal)
      AS SELECT     Dname, COUNT (*), SUM (Salary)
         FROM       DEPARTMENT, EMPLOYEE
         WHERE      Dnumber=Dno
         GROUP BY   Dname;
```

DEPT_INFO

Dept_name	No_of_emps	Total_sal

Specification of Views

Specify SQL queries on a view in same manner as on base tables.

Views are always up-to-date

- Responsibility of the DBMS and not the user

Specification of Views

Specify SQL queries on a view in same manner as on base tables.

Views are always up-to-date

- Responsibility of the DBMS and not the user

`DROP VIEW`

`DROP VIEW` command dispose of a view.

Example

```
DROP VIEW WORKS_ON;
```

Querying Views

The problem problem of efficiently implementing a view for querying is complex. Two main approaches have been suggested.

Querying Views

The problem problem of efficiently implementing a view for querying is complex. Two main approaches have been suggested.

Query Modification Approach

- Modify view query into a query on underlying base tables
- *Disadvantage*: inefficient for views defined via complex queries that are time-consuming to execute

Querying Views

View Materialization Approach

- Physically create a temporary view table when the view is first queried
- Keep that table on the assumption that other queries on the view will follow
- Requires efficient strategy for automatically updating the view table when the base tables are updated
- Incremental update strategies:
    - DBMS determines what new tuples must be inserted, deleted, or modified in a materialized view table

View Update

Update on a view defined on a single table without any aggregate functions can be mapped to an update on underlying base table.

View involving grouping and aggregate functions are not updateable.

View involving joins often not possible for DBMS to determine which of the updates is intended.

`WITH CHECK OPTION`

`WITH CHECK OPTION` must be added at the end of the view definition if a view is to be updated.

Inline View

Definition

An **Inline View** is defined in the FROM clause of an SQL query.

Example

```
SELECT height
FROM (SELECT height
      FROM test
      WHERE id = :b1
      ORDER BY id DESC,
               acc_date DESC,
               height DESC)
WHERE ROWNUM = 1;
```

Schema Change Statements

Schema evolution commands

- Can be done while the database is operational
- Does not require recompilation of the database schema

Schema Change Statements

Schema evolution commands

- Can be done while the database is operational
- Does not require recompilation of the database schema

DROP SCHEMA

DROP SCHEMA command is used to drop named schema elements, such as tables, domains, or constraint.

Example

```
DROP SCHEMA COMPANY CASCADE;
```

ALTER Command

ALTER TABLE

ALTER TABLE command permits:

- Adding or dropping a column (attribute)
- Changing a column definition
- Adding or dropping table constraints

Example

```
ALTER TABLE COMPANY.EMPLOYEE
      ADD COLUMN Job VARCHAR(12);
```

ALTER Command

ALTER command can change constraints specified on a table

- Add or drop a named constraint

```
ALTER TABLE COMPANY.EMPLOYEE
DROP CONSTRAINT EMPSUPERFK CASCADE;
```

Drop behavior options:

- CASCADE
- RESTRICT