CS143 Notes: Views & Authorization

Views

- What is a view?
  - A "virtual" table created on top of other "real" tables
  - Almost the same as a "real" table except that
    - the tuples are computed on the fly using "real" tables.
    - a view does not really "exist".
- Syntax and example:
  - `CREATE VIEW ViewName(`A_1`, `A_2`, ...) AS Query`
    - Attribute lists are optional
    - Example: SidNameAddr view with (sid, name, address) from Student
  - Views can be used in a query like:

```
SELECT *
FROM SidNameAddr S, Enroll E
WHERE S.sid = E.sid
```

    - The system automatically rewrites the query using the SidNameAddr view definition
  - Views can be created on top of other views

```
CREATE VIEW NameAddr AS
  SELECT name, addr
  FROM SidNameAddr
```

  - **Q:** MultiClass: View of students (sid, name) who take more than one class?
  - **Q:** Why use views?

- Three-level vision of database:
  - Virtual database: Views V := ViewQuery(R_1, R_2, ..., R_n)
    built on top of ...
  - Conceptual database: Tables (Relations)
    built on top of ...
  - Physical database: Pages on disk

Modifying Views

- Updates on views are allowed (under certain conditions)
- **Q:** How can we "update" a view when it does not exist?
  - **Q:** `UPDATE SidNameAddr SET Name = 'James' WHERE sid = 301`?

    - Modification to a view is "translated" into a modification to the underlying table
  - **Q:** `INSERT INTO SidNameAddr VALUES (305, 'Peter', '1234 Westwood')`?

    - Missing columns are filled with the DEFAULT value (or NULL)
  - **Q:** `INSERT INTO SameAddr VALUES ('Tony', 'Joshua')`?

```
CREATE VIEW SameAddr AS
  SELECT S1.name, S2.name
  FROM Student S1, Student S2
  WHERE S1.addr = S2.addr AND S1.sid > S2.sid
```

  - **Q:** `UPDATE AvgGPA SET a = 3.0`?

```
CREATE VIEW AvgGPA(a) AS SELECT avg(GPA) FROM Student;
```

  - For some views, update may not make any sense
    - Precise conditions for updatable views are very complicated
      - may involve keys, equality conditions, etc.
  - SQL2 uses very conservative conditions. View must be defined as:
    - SELECT on a single table T, without DISTINCT
    - Subqueries in WHERE must not refer to T

    - Attributes of T not projected in view allowed to be be NULL or default
    - No aggregation

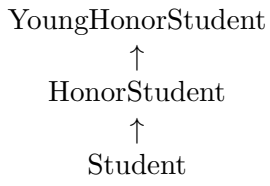
- **Q:** `INSERT INTO Student17 VALUES (403, 'Peter', '123 Olympic', 3.0, 20)` ?

```
CREATE VIEW Student17 AS
  SELECT *
  FROM Student
  WHERE age = 17
```

  - **Q:** Is the new tuple in Student17?

  - WITH CHECK OPTION
    - syntax: `CREATE VIEW ... AS ... WITH CHECK OPTION`
    - check INSERT/UPDATE to ensure the new tuple is still in the view
    - reject the statement if not

- **Q:** What will happen if we drop HonorStudent view?



  - `DROP ... [ CASCADE | RESTRICT ]`
    - CASCADE (default): drop anything that references the view
    - RESTRICT: drop statement fails if the view is referenced by other views or integrity constraints

Materialized Views

- Some DBMS allows to "precompute" or "materialize" a view
- Example: MultiClass view again. Students who take multiple classes

```
CREATE VIEW MultiClass AS
  SELECT sid, name
  FROM Student, Enroll
  WHERE Student.sid = Enroll.sid
  GROUP BY Student.sid
  HAVING COUNT(*) > 1
```

  - **Q:** Why do we want to we materialize this view?

- **Q:** Why don't we always materialize views?

    * **Q:** When should we refresh MultiClass?

        · refresh of materialized view can be costly
        · incremental refresh of materialized view is sometimes difficult

- **Q:** What views to materialize? Both? Just one? Pros and cons of each?

```
CREATE VIEW MultiClass AS
  SELECT sid, name
  FROM Student, Enroll
  WHERE Student.sid = Enroll.sid
  GROUP BY Student.sid
  HAVING COUNT(*) > 1

CREATE VIEW StudDeptCount AS
  SELECT sid, name, dept, COUNT(*)
  FROM Enroll
  GROUP BY sid, dept
```

    * Deciding views to materialize is a difficult optimization problem

- Many commercial DBMS supports materialized views
    - used for "data warehouse" for OLAP (online analytical processing) queries
    - limited support for incremental refresh
    - materialized view selection is a difficult optimization problem

Authorization

- Make sure users only see what they are allowed to see
- Do not let an unauthorized user to modify database

Privileges

- For a relation R and user U, U may be authorized for:
  - `SELECT ON R`
  - `INSERT(`A_1`, `A_2`, ..., `A_n`) ON R`
    - the rest of the columns should take NULL or DEFAULT
  - `UPDATE(`A_1`, `A_2`, ..., `A_n`) ON R`
  - `DELETE ON R`
- `GRANT` ⟨privileges⟩ `ON` ⟨R⟩ `TO` ⟨users⟩ `[ WITH GRANT OPTION ]`
  - ⟨privileges⟩: SELECT, INSERT, ... separated by commas (or ALL PRIVILEGES)
  - ⟨R⟩: table, view, ...
  - ⟨users⟩: list of users/groups, or PUBLIC
  - more about WITH GRANT OPTION later

    EXAMPLE: Grant SELECT privilege on Student to cs143

  - **Q:** When will it be useful to limit insert privileges to certain columns?

- EXAMPLES
  - ```
    UPDATE Student
    SET GPA = 4.0
    WHERE sid IN (SELECT sid FROM Enroll WHERE dept = 'CS')
    ```
    - **Q:** What privileges are needed for this statement?
  - ```
    SELETE FROM Student
    WHERE sid NOT IN (SELECT sid FROM Enroll)
    ```
    - **Q:** What privileges are needed for this statement?

Managing privileges

- **Q:** Who can grant privileges?
  - database administrator (DBA in oracle, SUPER in MySQL)
  - owner (= creator) of the table/view/...
- `GRANT ... TO` ⟨`u1`⟩ `WITH GRANT OPTION`
  - User u1 can grant the same or less privilege to others
- Authorization graph.
  - Nodes: users
  - Edges:
    - When u_i grants privilege to u_j, an edge is added from u_i to u_j.
    - When u_1 creates a table/view ..., edge is created from DBA to u_1
  - Example: What does the following authorization graph mean?

    $$\text{DBA} \to u_1 \to u_2 \to u_3$$

- Revoking privileges
  - `REVOKE` ⟨`privileges`⟩ `ON` ⟨`R`⟩ `FROM` ⟨`users`⟩ `[ CASCADE | RESTRICT ]`
    - EXAMPLE: revoke SELECT privilege on Student from u_2
  - CASCADE/REVOKE option in REVOKE statement
    - **Q:** Given the following authorization graph, what should happen when u_1 revokes privilege from u_2?

      $$\text{DBA} \to u_1 \to u_2 \to u_3$$

      - CASCADE (default): revoke all privileges passed on by the user
      - RESTRICT: reject the REVOKE command
    - **Q:** What if u_3 got privilege from u_4 as well?

      $$\begin{array}{ccccccc} \text{DBA} & \to & u_1 & \to & u_2 & \to & u_3 \\ & & & \searrow & u_4 & \nearrow & \end{array}$$

- **Q**:

$$\begin{array}{ccccc} \text{DBA} & \rightarrow & u_1 & \rightarrow & u_2 \\ & & & \searrow & \uparrow\downarrow \\ & & & & u_3 \end{array}$$

  If DBA revokes privilege from u_2, what edges should we remove? $u_1 \rightarrow u_2$? $u_3 \rightarrow u_2$?

  NOTES:

  - SQL allows u_1 to revoke privilege from u_2 only if u_1 granted the privilege to u_2. This restriction is to avoid privilege revocation ambiguity.
  - Unfortunately this restriction is too strict in practice, so most commercial systems do not enforce this. In case of ambiguity, they just do whatever is "reasonable" according to their policy.

- **Q:** What privileges for
  `CREATE TABLE test(a int, b int)`,
  `CREATE VIEW ...`?

  - Unfortunately, no standard in SQL
    - MySQL: CREATE, DROP, ALTER
    - Oracle: CREATE TABLE, CREATE VIEW, DROP ANY TABLE, ...
    - DB2: CREATETAB (table creation), CREATEALIAS (view creation), DROP, ALTER, ...

- Other privileges in later SQL standards
  - SQL92: REFERENCES (reference in constraints), USAGE (domain)
  - SQL99: TRIGGER, EXECUTE (function call), and UNDER (define subtype)

Controlling Access

- **Q:** How to allow user u_1 to see only (sid, name) of Student?

- **Q:** How to allow user u_1 to see only those students that take CS classes?

- **Q:** How to allow user u_1 modify a student record only if their age < 18?

- Authorization is one very important use of views.

Privileges and views

- **Q:** Assume user u is trying to create view V from R. User u has SELECT privilege on R (no grant option). Can u create such a view?

- **Q:** After V is created, can u grant SELECT on V to someone else?

- **Q:** If DBA revokes SELECT privilege on R from u, what should happen to V?

- Notes on views and privileges
  - To create a view, a user needs the SELECT privilege on the base tables of the view
  - To grant a privilege on a view, the user has to have GRANT OPTION privilege on the base tables of the view
  - When the privileges needed for the base tables are revoked, the affected views are automatically dropped.