System Administration Tasks

Kathleen Durant

CS3200

Database administrator responsibilities

Maintenance

- Monitor the server
- Configure the server
- Maintain log files

Design

- Design the database
- Create the database

Security

- Maintain user accounts
- Secure the server and its databases

Database administrator responsibilities (cont.)

Backup

- Backup the database regularly
- Restore the database if necessary
- Migrate data to another server if necessary

Miscellaneous

- Start or stop the server when necessary
- Optimize the server
- Update software when necessary
- Enable and manage replication if necessary

Creating Users

- Authorization identifier is normal SQL identifier used to establish identity of a user. Usually has an associated password.
- Used to determine which objects user may reference and what operations may be performed on those objects.
- Typically, each object created in SQL has an owner, as defined in AUTHORIZATION clause of schema to which object belongs.
  - In My SQL objects do not have an owner
- Owner is only person who may know or have access to the data object.

Limiting data access to users

- **Use the GRANT command to give a user the ability to perform certain operations on the database**
- **GRANT privilege_list ON [db_name.]table**
- **TO user1 [IDENTIFIED BY 'password1']**
- **[, user2 [IDENTIFIED BY 'password2']]...**
- **[WITH GRANT OPTION]**
- **WITH GRANT OPTION allows users to grant their privileges to other users**

https://dev.mysql.com/doc/refman/5.7/en/grant.html

Granting privilege to users

Privileges for working with data

(Typically user of the database)

- **SELECT (read data)**
- **INSERT (insert data)**
- **DELETE (delete data)**
- **UPDATE (update data)**
- **EXECTUTE (stored procedure, function)**

Privileges: administers or developers

- **CREATE (create DB, table)**
- **ALTER (alter a table)**
- **DROP (drop a DB)**
- **INDEX (create or drop an index)**
- **CREATE VIEWS (create views)**
- **CREATE ROUTINE**
- **ALTER ROUTINE**
- **TRIGGER (create or drop)**
- **EVENT (CRUD events)**

Privileges

- Actions user permitted to carry out on given base table or view:

SELECT	Retrieve data from a table.
INSERT	Insert new rows into a table.
UPDATE	Modify rows of data in a table.
DELETE	Delete rows of data from a table.
REFERENCES	Reference columns of named table in integrity constraints.
USAGE	Use domains, collations, character sets, and translations.

Privileges

- **Can restrict INSERT/UPDATE/REFERENCES to named columns.**
- **Owner of table must grant other users the necessary privileges using GRANT statement.**
- **To create view, user must have SELECT privilege on all tables that make up view and REFERENCES privilege on the named columns.**

GRANT COMMAND

Give Manager full privileges to Staff table.

```
GRANT ALL PRIVILEGES
ON Staff
TO Manager WITH GRANT OPTION;
```

Give users Personnel and Director SELECT and UPDATE on column salary of Staff.

```
GRANT SELECT, UPDATE (salary)
ON Staff
TO Personnel, Director;
```

Access to Four Data Levels

- The ON clause determines the level in which the privileges are granted
  - Global – access to all databases and all tables (*.*)
  - Database – access to all tables in the specified database
  - Table – access to all columns in the specified table
  - Columns only access to the specified columns in a table
- Examples:
- GRANT ALL ON *.* to poweruser IDENTIFIED BY ‘sesame’;

- GRANT SELECT (vendor_name, vendor_stats_ ON ap.vendors TO view.user;

Viewing user privileges

- SHOW GRANTS FOR *username*;
- To find all usernames:
  - SELECT user, host from mysql.user;

REVOKE

- **REVOKE takes away privileges granted with GRANT.**

```
REVOKE [GRANT OPTION FOR]
    {PrivilegeList | ALL PRIVILEGES}
ON ObjectName
FROM {AuthorizationIdList | PUBLIC}
     [RESTRICT | CASCADE]
```

- **ALL PRIVILEGES refers to all privileges granted to a user by user revoking privileges**
- **[GRANT OPTION FOR] not implemented in My SQL**

http://dev.mysql.com/doc/refman/5.7/en/revoke.html

REVOKE Specific Privileges

Revoke privilege SELECT on Branch table from all users.

```
REVOKE SELECT
ON Branch
FROM PUBLIC;
```

Revoke all privileges given to Director on Staff table.

```
REVOKE ALL PRIVILEGES
ON Staff
FROM Director;
```

Views: another data limitation mechanism

View

Dynamic result of one or more relational operations operating on base relations to produce another relation.

- **Virtual relation that does not necessarily actually exist in the database but is produced upon request, at time of request.**

Views

- **Contents of a view are defined as a query on one or more base relations.**
- **With view resolution, any operations on the view are automatically translated into operations on relations from which it is derived.**
- **With view materialization, the view is stored as a temporary table, which is maintained as the underlying base tables are updated.**

Some benefits provided by views

- **Data independence**
- **Improved security**
- **Reduced complexity**
- **Convenience**
- **Customization**
- **Data integrity**
- **Concurrency**

Disadvantages of Views

- **Update restriction**
- **Structure restriction**
- **Performance**

SQL - CREATE VIEW

CREATE VIEW ViewName [(newColumnName [,...])]

AS subselect

[WITH [CASCADED | LOCAL] CHECK OPTION]

- **Can assign a name to each column in view.**
- **If list of column names is specified, it must have same number of items as number of columns produced by *subselect*.**
- **If omitted, each column takes name of corresponding column in *subselect*.**

SQL - CREATE VIEW

- List must be specified if there is any ambiguity in a column name.
- The *subselect* is known as the defining query.
- WITH CHECK OPTION ensures that if a row fails to satisfy WHERE clause of defining query, it is not added to underlying base table.
- LOCAL and CASCADED keywords determine the scope of check testing when the view is defined in terms of another view
  - LOCAL keyword restricts the CHECK OPTION only to the view being defined
- Need SELECT privilege on all tables referenced in subselect and USAGE privilege on any domains used in referenced columns.

Create Horizontal View

Access to a subset of the records in a table

Example: Create view so that manager at branch B003 can only see details for staff who work in his or her office.

```
CREATE VIEW Manager3Staff
AS      SELECT *
        FROM Staff
        WHERE branchNo = 'B003';
```

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000.00	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000.00	B003
SG5	Susan	Brand	Manager	F	3-Jun-40	24000.00	B003

Create Vertical View

Create a view that limits the variables in the base relation

Example: Create view of staff details at branch B003 excluding salaries.

```
CREATE VIEW Staff3
AS SELECT staffNo, fName, lName, position, sex
    FROM Staff
    WHERE branchNo = 'B003';
```

staffNo	fName	lName	position	sex
SG37	Ann	Beech	Assistant	F
SG14	David	Ford	Supervisor	M
SG5	Susan	Brand	Manager	F

Grouped and Joined Views

Base query is not limited to a simple query statement

Create view of staff who manage properties for rent, including branch number they work at, staff number, and number of properties they manage.

```
CREATE VIEW StaffPropCnt (branchNo, staffNo, cnt)
 AS SELECT s.branchNo, s.staffNo, COUNT(*)
    FROM Staff s, PropertyForRent p
    WHERE s.staffNo = p.staffNo
    GROUP BY s.branchNo, s.staffNo;
```

VIEW DEFINITION IN MYSQL

CREATE [OR REPLACE]
[ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
[DEFINER = { *user* | CURRENT_USER }]
[SQL SECURITY { DEFINER | INVOKER }]
VIEW *view_name* [(*column_list*)] AS *select_statement*
[WITH [CASCADED | LOCAL] CHECK OPTION]

ALGORITHM CLAUSE:

- **MERGE:** combines the input query with the defining select statement, into a single query. Executes the combined query to return the result set.
  - LIMITATION: Query cannot have aggregates
- **TEMPTABLE**: Create a temporary table of the defining query. Executes the query against the temporary table.
  - LIMITATION: Data not UPDATEABLE through the view
- **UNDEFINED**: My SQL chooses between MERGE or TEMPTABLE

Options: My SQL View

- The DEFINER and SQL SECURITY clauses specify the security context to be used when checking access privileges at view invocation time.
  - DETERMINE which MySQL account to use when checking access privileges for the view when a statement is executed that references the view.
  - Necessary extension to the SQL standard since My SQL does not have the concept of an owner of a table
  - If a *user* value is given for the DEFINER clause, it should be a MySQL account specified as '*user_name*'@'*host_name*'
- The WITH CHECK OPTION clause can be given to constrain inserts or updates to rows in tables referenced by the view.

Identifying VIEW definition

- My SQL allows you to examine the definition of a query:
- SHOW CREATE VIEW *viewname*;

SQL - DROP VIEW

```
DROP VIEW ViewName [RESTRICT | CASCADE]
```

- Causes definition of view to be deleted from database.
- With CASCADE, all related dependent objects are deleted; i.e. any views defined on view being dropped.
- With RESTRICT , if any other objects' existence depend on the continued existence of view being dropped, then the drop command is rejected.
- For example:

```
DROP VIEW Manager3Staff;
```

View Resolution

Count number of properties managed by each member at branch B003.

```
SELECT staffNo, cnt
FROM StaffPropCnt
WHERE branchNo = 'B003'
ORDER BY staffNo;
```

View Resolution

(a) View column names in SELECT list are translated into their corresponding column names in the defining query:

```
SELECT s.staffNo As staffNo, COUNT(*) As cnt
```

(b) View names in FROM are replaced with corresponding FROM lists of defining query:

```
FROM Staff s, PropertyForRent p
```

View Resolution

(c) WHERE from user query is combined with WHERE of defining query using AND:

```
WHERE s.staffNo = p.staffNo AND branchNo = 'B003'
```

(d) GROUP BY and HAVING clauses copied from defining query:

```
GROUP BY s.branchNo, s.staffNo
```

(e) ORDER BY copied from query with view column name translated into defining query column name

```
ORDER BY s.staffNo
```

View Resolution

(f) Final merged query is now executed to produce the result:

```
SELECT s.staffNo AS staffNo, COUNT(*) AS cnt
FROM Staff s, PropertyForRent p
WHERE s.staffNo = p.staffNo AND
        branchNo = 'B003'
GROUP BY s.branchNo, s.staffNo
ORDER BY s.staffNo;
```

Restrictions on Views

SQL imposes several restrictions on creation and use of views.

(a) If a column in a view is based on an aggregate function:

- Column may appear only in SELECT and ORDER BY clauses of queries that access view.
- Column may not be used in WHERE nor be an argument to an aggregate function in any query based on view.

Restrictions on Views

- For example, the following query would fail:

```
SELECT COUNT(cnt) – cnt is an aggregated value
FROM StaffPropCnt;
```

- Similarly, the following query would also fail:

```
SELECT *
FROM StaffPropCnt
WHERE cnt > 2;
```

Restrictions on Views

(b) Grouped view may never be joined with a base table or a view.

- **For example, StaffPropCnt view is a grouped view, so any attempt to join this view with another table or view fails.**

View Updatability

- **All updates to base table reflected in all views that encompass base table.**
- **Similarly, may expect that if view is updated then base table(s) will reflect change.**

View Updatability

- However, consider again view StaffPropCnt.
- If we tried to insert record showing that at branch B003, SG5 manages 2 properties:

```
INSERT INTO StaffPropCnt
VALUES ('B003', 'SG5', 2);
```

- Have to insert 2 records into PropertyForRent showing which properties SG5 manages. However, do not know which properties they are; i.e. do not know primary keys!

View Updatability

- If change definition of view and replace count with actual property numbers:

```
CREATE VIEW StaffPropList (branchNo,
                           staffNo, propertyNo)
AS SELECT s.branchNo, s.staffNo, p.propertyNo
     FROM Staff s, PropertyForRent p
     WHERE s.staffNo = p.staffNo;
```

View Updatability

- **Now try to insert the record:**

```
INSERT INTO StaffPropList
VALUES ('B003', 'SG5', 'PG19');
```

- **Still problem, because in PropertyForRent all columns except postcode/staffNo are not allowed nulls.**
- **However, have no way of giving remaining non-null columns values.**

View Updatability

- **ISO specifies that a view is updatable if and only if:**

  - DISTINCT is not specified.

  - Every element in the SELECT list of the defining query is a column name and no column appears more than once.

  - FROM clause specifies only one table, excluding any views based on a join, union, intersection or difference.

  - No nested SELECT referencing outer table.

  - No GROUP BY or HAVING clause.

  - Also, every row added through view must not violate integrity constraints of base table.

Updatable View

For a view to be updatable, DBMS must be able to trace any row or column back to its row or column in the source table.

For a view to be updatable, there must be a one-to-one relationship between the rows in the view and the rows in the underlying table.

WITH CHECK OPTION

- **Rows exist in a view because they satisfy WHERE condition of defining query.**
- **If a row changes and no longer satisfies condition, it disappears from the view.**
- **New rows appear within view when insert/update on view cause them to satisfy WHERE condition.**
- **Rows that enter or leave a view are called *migrating rows*.**
- **"WITH CHECK OPTION" prohibits a row migrating out of the view.**

WITH CHECK OPTION

- **LOCAL/CASCADED apply to view hierarchies.**
- **With LOCAL, any row insert/update on view and any view directly or indirectly defined on this view must not cause row to disappear from view unless row also disappears from derived view/table.**
- **With CASCADED (default), any row insert/ update on this view and on any view directly or indirectly defined on this view must not cause row to disappear from the view.**

WITH CHECK OPTION

```
CREATE VIEW Manager3Staff
    AS  SELECT *
    FROM Staff
    WHERE branchNo = 'B003'
WITH CHECK OPTION;
```

- Cannot update branch number of row B003 to B002 as this would cause row to migrate from view.
- Also cannot insert a row into the view with a branch number that does not equal B003.

WITH CHECK OPTION

- **Now consider the following:**

```
CREATE VIEW LowSalary
 AS   SELECT * FROM Staff WHERE salary > 9000;
CREATE VIEW HighSalary
 AS   SELECT * FROM LowSalary  --view on a view
      WHERE salary > 10000
 WITH LOCAL CHECK OPTION;
CREATE VIEW Manager3Staff
 AS   SELECT * FROM HighSalary –view on a view on a view
      WHERE branchNo = 'B003';
```

WITH CHECK OPTION

```
UPDATE Manager3Staff
SET salary = 9500
WHERE staffNo = 'SG37';
```

- This update would fail: although update would cause row to disappear from HighSalary, row would not disappear from LowSalary.
- However, if update tried to set salary to 8000, update would succeed as row would no longer be part of LowSalary.

WITH CHECK OPTION

- **If HighSalary had specified WITH CASCADED CHECK OPTION, setting salary to 9500 or 8000 would be rejected because row would disappear from HighSalary.**
- **To prevent anomalies like this, each view should be created using WITH CASCADED CHECK OPTION.**

View Materialization

- View resolution mechanism may be slow, particularly if view is accessed frequently.
- View materialization stores view as temporary table when view is first queried.
- Thereafter, queries based on materialized view can be faster than recomputing view each time.
- Difficulty is maintaining the currency of view while base tables(s) are being updated.

View Maintenance

- **View maintenance aims to apply only those changes necessary to keep view current.**
- **Consider following view:**

```
CREATE VIEW StaffPropRent(staffNo)
AS    SELECT DISTINCT staffNo
      FROM PropertyForRent
      WHERE branchNo = 'B003' AND
                  rent > 400;
```

staffNo
SG37
SG14

View Materialization

- If insert row into PropertyForRent with rent ≤400 then view would be unchanged.
- If insert row for property PG24 at branch B003 with staffNo = SG19 and rent = 550, then row would appear in materialized view.
- If insert row for property PG54 at branch B003 with staffNo = SG37 and rent = 450, then no new row would need to be added to materialized view.
- If delete property PG24, row should be deleted from materialized view.
- If delete property PG54, then row for PG37 should not be deleted (because of existing property PG21).