ITS SEMANGAT BARU

ITS Institut Teknologi Sepuluh Nopember

View

What is a View?

- A *view* is a 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 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.

MySQL - **CREATE VIEW**

The format is:

```
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*.

MySQL - **`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.
- Need **`SELECT`** privilege on all tables referenced in subselect and USAGE privilege on any domains used in referenced columns.

CREATE VIEW – An Example

```
mysql> select * from Staff;
+---------+-------+-------+-----------+------+------------+----------+----------+
| staffNo | fName | lName | position  | sex  | DOB        | salary   | branchNo |
+---------+-------+-------+-----------+------+------------+----------+----------+
| SA9     | Mary  | Howe  | Assistant | f    | 1970-02-19 | 9270.00  | B007     |
| SG14    | David | Ford  | manager   | M    | 1958-03-24 | 18000.00 | B003     |
| SG16    | Alan  | Brown | Assistant | M    | 1957-05-25 | 8549.00  | B003     |
| SG37    | Ann   | Beech | Assistant | f    | 1960-11-10 | 12360.00 | B003     |
| SG44    | Anne  | Jones | Assistant | NULL | NULL       | 8343.00  | B003     |
| SG45    | Anna  | Smith | Assistant | NULL | NULL       | 8446.00  | B002     |
| SG5     | Susan | Brand | Manager   | f    | 1940-06-03 | 25956.00 | B003     |
| SL21    | John  | White | Manager   | M    | 1945-10-01 | 32445.00 | B005     |
| SL41    | Julie | Lee   | Assistant | f    | 1965-06-13 | 9270.00  | B005     |
+---------+-------+-------+-----------+------+------------+----------+----------+
9 rows in set (0.00 sec)
```

CREATE VIEW – An Example

```
mysql> CREATE VIEW Manager3Staff
    -> AS SELECT *
    -> from Staff
    -> where branchNo = 'B003';
Query OK, 0 rows affected (0.02 sec)


mysql>
```

CREATE VIEW – An Example

```
mysql> select * from Manager3Staff;
+---------+-------+-------+-----------+------+------------+----------+----------+
| staffNo | fName | lName | position  | sex  | DOB        | salary   | branchNo |
+---------+-------+-------+-----------+------+------------+----------+----------+
| SG14    | David | Ford  | manager   | M    | 1958-03-24 | 18000.00 | B003     |
| SG16    | Alan  | Brown | Assistant | M    | 1957-05-25 | 8549.00  | B003     |
| SG37    | Ann   | Beech | Assistant | f    | 1960-11-10 | 12360.00 | B003     |
| SG44    | Anne  | Jones | Assistant | NULL | NULL       | 8343.00  | B003     |
| SG5     | Susan | Brand | Manager   | f    | 1940-06-03 | 25956.00 | B003     |
+---------+-------+-------+-----------+------+------------+----------+----------+
5 rows in set (0.00 sec)

mysql>
```

CREATE VIEW – An Example

```
mysql> CREATE VIEW Staff3
    -> AS SELECT staffNo, fName, lName, position, sex
    -> from Staff
    -> where branchNo = 'B003';
Query OK, 0 rows affected (0.00 sec)


mysql> select * from Staff3;
```

CREATE VIEW – An Example

```
mysql> select * from Staff3;
+---------+-------+-------+-----------+------+
| staffNo | fName | lName | position  | sex  |
+---------+-------+-------+-----------+------+
| SG14    | David | Ford  | manager   | M    |
| SG16    | Alan  | Brown | Assistant | M    |
| SG37    | Ann   | Beech | Assistant | f    |
| SG44    | Anne  | Jones | Assistant | NULL |
| SG5     | Susan | Brand | Manager   | f    |
+---------+-------+-------+-----------+------+
5 rows in set (0.00 sec)
```

Grouped and Joined Views

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

```
mysql> 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;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

CREATE VIEW – An Example

```
mysql> select * from StaffPropCnt;
+----------+---------+-----+
| branchNo | staffNo | cnt |
+----------+---------+-----+
| B003     | SG14    |  1  |
| B003     | SG37    |  2  |
| B003     | SG5     |  1  |
| B005     | SL41    |  1  |
| B007     | SA9     |  1  |
+----------+---------+-----+
5 rows in set (0.00 sec)

mysql>
```

MySQL – **DROP VIEW**

- The Format is:

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

- Causes definition of view to be deleted from database.
- For example:

```
mysql> DROP View Manager3Staff;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

MySQL – `DROP VIEW`

- With `CASCADE`, all related dependent objects are deleted; i.e., any views defined on view being dropped.
- With `RESTRICT` (default), if any other objects depend for their existence on continued existence of view being dropped, command is rejected.

View Resolution

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

```
mysql> SELECT staffNo, cnt FROM StaffPropCnt
    -> WHERE branchNo = 'B003' ORDER BY staffNo;
+----------+-----+
| staffNo | cnt |
+----------+-----+
| SG14     |  1  |
| SG37     |  2  |
| SG5      |  1  |
+---------+-----+
3 rows in set (0.00 sec)

mysql>
```

View Resolution

- 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
```

- View names in `FROM` are replaced with corresponding `FROM` lists of defining query:

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

View Resolution

- `WHERE` from user query is combined with `WHERE` of defining query using `AND`:

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

- `GROUP BY` and `HAVING` clauses copied from defining query:

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

- `ORDER BY` copied from query with view column name translated into defining query column name

```
ORDER BY s.staffNo
```

View Resolution

- 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.
- If a column in 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, following query would fail:

```
SELECT COUNT(cnt)
FROM StaffPropCnt;
```

- Similarly, following query would also fail:

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

Restrictions on Views

- 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 we change the definition of view and replacecount 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');
```

- There is still problem, because in PropertyForRent none of the columns (except postcode/staffNo) are not allowed nulls.
- However, there is no way of giving remaining nonnull columns values.

View Updatability

- ISO specifies that a view is updatable if and only if:
  - `DISTINCT` is not specified.
  - Every element in `SELECT` list of 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.

View Updatability

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

View Updatability

- For view to be updatable, DBMS must be able to trace any row or column back to its row or column in the source 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 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
     WHERE salary > 10000
  WITH LOCAL CHECK OPTION;
```

```
CREATE VIEW Manager3Staff
  AS SELECT * FROM HighSalary
     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 `LowSalary`.
- To prevent anomalies like this, each view should be created using `WITH CASCADED CHECK OPTION`.

Advantages of Views

- Data independence – presents a consistent, unchanging picture of the database's structure even when the source tables change
- Currency – changes to the base tables are reflected immediately in the views.
- Improved security- users can be granted access to the database through a relatively small set of views.

Advantages of Views

- Reduced complexity – simplifies the writing of queries.
- Convenience – users see only what they need.
- Customization – views can be customized to the needs of individual users.
- Data integrity – CHECK OPTION clause of the CREATE VIEW command ensures that rows satisfy the WHERE clause of the defining query.

Disadvantages of Views

- Update restriction – in some cases (as we saw), a view might not be updated.
- Structure restriction – Structure is determined at the time of creation. Any columns added to the data base will not show up unless the view is dropped and redefined.
- Performance – The use of view slows down response time in some cases.

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;
```

Thank You!