Database Systems

SQL Views

Objectives

- Purpose of views
- Advantages and disadvantages of views
- Create and drop views using SQL
- View materialization
- Under what conditions are views updatable

Simple vs Materialized Views

- Dynamic result of one or more relational operations operating on base tables
- Normally stored as a query definition, no data
- Virtual table produced upon request, not necessarily exists in the database
- Defined as a query on one or more base tables
- View materialization is relatively new, common in data warehousing applications
- With view materialization, the view is stored as a temporary table, and maintained as the underlying base tables are updated

Simple Views : Pros and Cons

- Advantages :
  - Improved Security
  - Reduced Complexity
  - Customization
  - Data Independence
  - Data integrity
  - No extra storage required
- Disadvantages :
  - Performance
  - Update restriction
  - Structure restriction (e.g.constraints not supported, cannot reference itself)

SQL - CREATE VIEW

```
CREATE VIEW ViewName [(ColumnName [,...])]
AS subselect
[WITH [CASCADED | LOCAL] CHECK OPTION]
```

- If list of column names is specified, it must have same number of items as produced by subselect
- If omitted, each column takes name of corresponding column in subselect
- List must be specified if there is any ambiguity in a column name
- The subselect is known as the defining query
- Some DBMSs support only WITH CHECK OPTION

SQL - CREATE VIEW

- Need SELECT privilege on all tables referenced in subselect and USAGE privilege on any domains used in referenced columns
- WITH CHECK OPTION – applicable only for updatable views
  - If a modified row fails to satisfy WHERE clause of defining query, ensure it is not done to base table
  - Forces all data modification statements executed against the view to adhere to the criteria set within the defining query
  - Ensures the data remains visible through the view after the modification is committed

CREATE VIEW Examples

```
CREATE VIEW au_names
  AS
  SELECT au_id, au_fname, au_lname
    FROM authors;
```

```
CREATE VIEW cities
  (au_id, au_city, pub_id, pub_city)
  AS
  SELECT a.au_id, a.city, p.pub_id, p.city
    FROM authors a
    INNER JOIN publishers p
      ON a.city = p.city;
```


CREATE VIEW Examples

- View that lists total revenue grouped by book type within publishers

```
CREATE VIEW revenues
  (Publisher, BookType, Revenue)
  AS
  SELECT pub_id, type, SUM(price * sales)
    FROM titles
    GROUP BY pub_id, type;
```


DROP VIEW

- **DROP VIEW ViewName [RESTRICT | CASCADE];**
  - CASCADE : all dependent objects are deleted; i.e. any views defined on the view being dropped
  - RESTRICT (default): if any other objects depend for their existence on continued existence of the view being dropped, command is rejected
- For example:

```
DROP VIEW ny_authors;
```

- Many DBMSs do not support [RESTRICT | CASCADE]
- Oracle supports CREATE OR REPLACE ...

View Materialization

- Newer DBMS feature, and commonly used in data warehousing application
- Materialization means the view definition query is executed and the data result is stored
- Speeds up performance in handling large volume of data (over terabytes)
- Data currency is maintained by DBMS when underlying base tables are updated
- SQL syntax varies between DBMSs

View Updatability

- An updateable view is one to which one can apply INSERT, UPDATE and DELETE statements.
- 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.
- View is not updateable if its SELECT definition uses GROUP BY, HAVING, DISTINCT or aggregate functions.
- Single-table views are almost always updateable if the primary key is included in the view.
- Can use INSTEAD OF triggers

View Updatability

- SQL-92 : 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 each column appear only once
  - FROM clause specifies only one table, excluding any views based on JOIN, UNION, INTERSECT or EXCEPT
  - 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
- SQL:1999 relaxed some restrictions

View Updatability

```
CREATE VIEW ny_authors
  AS
  SELECT au_id, au_fname, au_lname, state
    FROM authors
    WHERE state = 'NY';
```

```
UPDATE ny_authors
  SET au_fname = 'Yasmin',
      au_lname = 'Howcomely'
  WHERE au_id = 'A01';
```

```
INSERT INTO ny_authors
  VALUES('A08','Don','Dawson','NY');
```


View Updatability

```
CREATE VIEW ny_authors
  AS
  SELECT au_id, au_fname, au_lname, state
    FROM authors
    WHERE state = 'NY';
```

```
UPDATE ny_authors
  SET au_fname = 'Yasmin',
      au_lname = 'Howcomely',
      state = 'CA'
  WHERE au_id = 'A01';
```

```
INSERT INTO ny_authors
  VALUES('A09','Jill','LeFlore','CA');
```


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 in or out of the view

WITH CHECK OPTION

```
CREATE VIEW ny_authors
  AS
  SELECT au_id, au_fname, au_lname,
      state
    FROM authors
    WHERE state = 'NY'
  WITH CHECK OPTION;
```

- The following statements will generate an error:

```
UPDATE ny_authors
  SET au_fname = 'Yasmin',
      au_lname = 'Howcomely',
      state = 'CA'
  WHERE au_id = 'A01';
```

```
INSERT INTO ny_authors
  VALUES('A09','Jill','LeFlore','CA');
```