Views

Chapter 7 - Objectives

- Data definition
- CREATE table statements
- Data types supported by SQL standard
- ALTER table statements
- Purpose of integrity enhancement feature of SQL
- Purpose of Views
- ISO transaction model
- Access Control

Chapter 7 - Objectives

- Data definition
- CREATE table statements
- Data types supported by SQL standard
- ALTER table statements
- Purpose of integrity enhancement feature of SQL
- **Purpose of Views**
- ISO transaction model
- Access Control

Views

- Purpose of Views
- Advantages of Views
- How to create and delete Views using SQL
- View Resolution
- Restrictions on Views
- Under what conditions views are updatable
- Disadvantages of views
- View Materialization and Maintenance

Views

- **Purpose of Views**
- Advantages of Views
- How to create and delete Views using SQL
- View Resolution
- Restrictions on Views
- Under what conditions views are updatable
- Disadvantages of views
- View Materialization and Maintenance

Views

- **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.

Views

- Purpose of Views
- **Advantages of Views**
- How to create and delete Views using SQL
- View Resolution
- Restrictions on Views
- Under what conditions views are updatable
- Disadvantages of views
- View Materialization and Maintenance

Advantages of Views

- Data independence
- Currency
- Improved security
- Reduced complexity
- Convenience
- Customization
- Data integrity

Views

- Purpose of Views.
- Advantages of Views
- **How to create and delete Views using SQL**
- View Resolution
- Restrictions on Views
- Under what conditions views are updatable
- Disadvantages of views
- View Materialization and Maintenance

SQL - CREATE VIEW

Not in some SQL dialects

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

- **Can assign a name to each column in view**
  - **must have same number of items as number of columns produced by *subselect***
- **List must be specified if there is any ambiguity in a column name.**
- **The subselect is known as the defining query.**

SQL - CREATE VIEW

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

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

Example 7.3 - Create Horizontal View

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';
```

Table 6.3 Data for view Manager3Staff.

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

Example 7.4 - Create Vertical View

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';
```

Table 6.4 Data for view Staff3.

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

Example 7.5 - 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.

```
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;
```

Example 7.5 - 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.

```
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;
```

Not in some SQL dialects

Use “COUNT(*) as cnt”

Example 7.5 - 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.

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

Example 7.3 - Grouped and Joined Views

branchNo	staffNo	cnt
B003	SG14	1
B003	SG37	2
B005	SL41	1
B007	SA9	1

SQL - DROP VIEW

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

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

```
DROP VIEW Manager3Staff;
```

SQL - DROP VIEW

Not in SQLite

DROP VIEW ViewName [RESTRICT | CASCADE]

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

Views

- Purpose of Views.
- Advantages of Views
- How to create and delete Views using SQL
- **View Resolution**
- Restrictions on Views
- Under what conditions views are updatable
- Disadvantages of views
- View Materialization and Maintenance

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

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

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

View Resolution

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

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

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

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

View Resolution

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

ORDER BY staffNo
ORDER BY s.staffNo

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

(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
                  s.branchNo = 'B003'
GROUP BY s.branchNo, s.staffNo
ORDER BY s.staffNo;
```

Views

- Purpose of Views.
- Advantages of Views
- How to create and delete Views using SQL
- View Resolution
- **Restrictions on Views**
- Under what conditions views are updatable
- Disadvantages of views
- View Materialization and Maintenance

Restrictions on Views

SQL imposes several restrictions on creation and use of views.

(a) If 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;
```

This is ISO, This works in SQLite!!

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.**

This is ISO, This works in SQLite!!

Views

- Purpose of Views.
- Advantages of Views
- How to create and delete Views using SQL
- View Resolution
- Restrictions on Views
- **Under what conditions views are updatable**
- Disadvantages of views
- View Materialization and Maintenance

View Updatability

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

Not in SQLite: Views are READ ONLY

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 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.

  - 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.

Not in SQLite: Views are READ ONLY

Updatable View

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*.

SQL - CREATE VIEW

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

- 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.**

Example 7.6 - 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.

Views

- Purpose of Views.
- Advantages of Views
- How to create and delete Views using SQL
- View Resolution
- Restrictions on Views
- Under what conditions views are updatable
- **Disadvantages of views**
- View Materialization and Maintenance

Disadvantages of Views

- Update restriction (not at all in SQLite)
  - Can only update under certain conditions
- Structure restriction
  - SELECT * → columns defined at creation
- Performance
  - View resolutions, complex views

Views

- Purpose of Views.
- Advantages of Views
- How to create and delete Views using SQL
- View Resolution
- Restrictions on Views
- Under what conditions views are updatable
- Disadvantages of views
- **View Materialization and Maintenance**

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

branchNo = ‘B003’ AND
rent > 400;

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).

Views

- Purpose of Views.
- Advantages of Views
- How to create and delete Views using SQL
- View Resolution
- Restrictions on Views
- Under what conditions views are updatable
- Disadvantages of views
- View Materialization and Maintenance

Chapter 7 - Objectives

- Data definition
- CREATE table statements
- Data types supported by SQL standard
- Purpose of integrity enhancement feature of SQL
- ALTER table statements
- Purpose of Views
- ISO transaction model
- Access Control