Views

Leo Mark

5/18/2020

Views – definition

Create AtlantaUserInterests view with Email, Interest and SinceAge of RegularUsers in Atlanta

```
CREATE VIEW AtlantaUserInterests AS
SELECT U.Email, Interest, SinceAge
FROM UserInterests I, RegularUser U
WHERE I.Email = U.Email AND HomeTown='Atlanta';
```

- a view is a **virtual*** table
- the definition is stored in the catalog
- column names are inherited from the base tables
- columns may be explicitly named in the definition
- columns must be explicitly named if ambiguous
- computed columns must be explicitly named

* *Unless otherwise specified, this is our definition of a view*

RegularUser

Email	Birth Year	Sex	HomeTown
user1@gt.edu	1985	M	Atlanta
user2@gt.edu	1969	M	Austin
user3@gt.edu	1967	M	Portland
user4@gt.edu	1988	M	Atlanta

UserInterests

Email	Interest	SinceAge
user1@gt.edu	Music	10
user1@gt.edu	Reading	5
user1@gt.edu	Tennis	14
user2@gt.edu	Blogging	13
user3@gt.edu	Music	11
user4@gt.edu	DIY	18

AtlantaUserInterests

Email	Interest	SinceAge
user1@gt.edu	Music	10
user1@gt.edu	Reading	5
user1@gt.edu	Tennis	14
user4@gt.edu	DIY	18

Views – use in queries

Find Email and Interest from AtlantaUserInterests where the user had the Interest SinceAge 12 or older

How a query on a view is written:

```
SELECT Email, Interest
FROM AtlantaUserInterests
WHERE SinceAge >= 12;
```

How a query is computed by query modification:

```
SELECT U.Email, Interest
FROM UserInterests I, RegularUser U
WHERE I.Email = U.Email AND HomeTown='Atlanta' and
SinceAge >= 12;
```

How a view is dropped:

```
DROP VIEW AtlantaUserInterests [RESTRICT|CASCADE];
```

RegularUser

Email	Birth Year	Sex	HomeTown
user1@gt.edu	1985	M	Atlanta
user2@gt.edu	1969	M	Austin
user3@gt.edu	1967	M	Portland
user4@gt.edu	1988	M	Atlanta

UserInterests

Email	Interest	SinceAge
user1@gt.edu	Music	10
user1@gt.edu	Reading	5
user1@gt.edu	Tennis	14
user2@gt.edu	Blogging	13
user3@gt.edu	Music	11
user4@gt.edu	DIY	18

AtlantaUserInterests

Email	Interest	SinceAge
user1@gt.edu	Music	10
user1@gt.edu	Reading	5
user1@gt.edu	Tennis	14
user4@gt.edu	DIY	18

Views – updatability

Since a view is a virtual table, “updates” to a view can only be done by the DBMS updating the base tables from which the view is defined

A view is **updatable if and only if**:

- it does not contain any of the keywords JOIN, UNION, INTERSECT, EXCEPT
- it does not contain the keyword DISTINCT
- every column in the view corresponds to a uniquely identifiable base table column
- the FROM clause references exactly one table which must be a base table or an updatable view
- the table referenced in the FROM clause cannot be referenced in the FROM clause of a nested WHERE clause
- it does not have a GROUP BY clause
- it does not have a HAVING clause
- updatable means: insert, delete, update are ok

Views – not updatable

```
CREATE VIEW SexHomeTown AS
SELECT Sex, HomeTown
FROM RegularUser
WHERE BirthYear >= 1969;
```

not unique

```
CREATE VIEW AvgUserInterestSinceAge AS
SELECT Email, AVG(SinceAge)
FROM UserInterests
GROUP BY Email;
```

computed attribute

```
CREATE VIEW AtlantaUserInterests AS
SELECT U.Email, Interest, SinceAge
FROM UserInterests I, RegularUser U
WHERE I.Email = U.Email AND HomeTown='Atlanta';
```

join of tables

RegularUser

Email	Birth Year	Sex	HomeTown
user1@gt.edu	1985	M	Atlanta
user2@gt.edu	1969	M	Austin
user3@gt.edu	1967	M	Portland
user4@gt.edu	1988	M	Atlanta

UserInterests

Email	Interest	SinceAge
user1@gt.edu	Music	10
user1@gt.edu	Reading	5
user1@gt.edu	Tennis	14
user2@gt.edu	Blogging	13
user3@gt.edu	Music	11
user4@gt.edu	DIY	18

AtlantaUserInterests

Email	Interest	SinceAge
user1@gt.edu	Music	10
user1@gt.edu	Reading	5
user1@gt.edu	Tennis	14
user4@gt.edu	DIY	18

Views – updatable, but …

```
CREATE VIEW UserInterestsSinceTeen AS  /*an updatable view*/
SELECT *
FROM UserInterests
WHERE SinceAge >= 13;

UPDATE UserInterestsSinceTeen      /*moves row(s) outside the view*/
SET SinceAge = 12
WHERE Interest = ‘DIY’;

INSERT INTO UserInterestsSinceTeen      /*creates row outside the view*/
VALUES (‘user7@gt.edu’, ‘Soccer’, 8);

CREATE VIEW UserInterestsSinceTeen AS
SELECT *
FROM UserInterests
WHERE SinceAge >= 13
WITH CHECK OPTION;                /*prevents updates outside the view */
```

UserInterests

Email	**Interest**	**SinceAge**
user1@gt.edu	Music	10
user1@gt.edu	Reading	5
user1@gt.edu	Tennis	14
user2@gt.edu	Blogging	13
user3@gt.edu	Music	11
user4@gt.edu	DIY	18

Materialized Views

To create a materialized view, we use the following syntax:

```
CREATE MATERIALIZED VIEW ViewName
[REFRESH [FAST|COMPLETE|FORCE] [ON DEMAND|ON COMMIT]] [BUILD IMMEDIATE|BUILD DEFERRED]
AS Select Query;
```

- **The materialized view definition is stored in the catalog.**
- **The Select Query is run and the results stored in the materialized view table.**

Options includes:

- **REFRESH FAST**: uses an incremental refresh method which uses changes made to the underlying tables in a log file.
- **REFRESH COMPLETE**: a complete refresh by re-running the query in the materialized view.
- **REFRESH FORCE**: a fast refresh should be performed if possible, but if not, a complete refresh is performed.
- **REFRESH ON DEMAND**: a refresh will occur manually whenever specific package functions are called.
- **REFRESH ON COMMIT**: a fast refresh occurs whenever a transaction commits that makes changes to any of the underlying tables.
- **BUILD IMMEDIATE**: the materialized view will be populated immediately. This is the default.
- **BUILD DEFERRED**: the materialized view is populated on the next refresh operation.

Source: https://www.databasestar.com/sql-views
Remember that the syntax and semantics may vary in different commercial DBMSs.