Views

Agenda

- Virtual Tables
- View
- Types of Views
  - Simple View
  - Complex View
  - Inline View
  - Horizontal View
  - Vertical View
  - Joined View
  - Drop View

Virtual Tables

- A virtual table is a ***derived form of data*** from a physical database tables.
- Virtual tables are logically represented using physical data but ***do not store*** any data.
- A virtual table is a construct of record set that are not actually tables.

  so that these record sets are referenced in SQL statements like normal tables.
- Virtual tables otherwise called as views and are stored as database objects in MYSQL

  Information_schema, and can be retrieved without re-writing it again.

What is a view?

What is view?

- A view is an object in a database that is created using a Select Query in combination with complex logic.
- Views are therefore logical representations of physical data.
- The views behave like a physical table, so they can be used as database objects in all parts of SQL queries.
- DML operations can be performed on base tables via views.
- Since they are stored in the data dictionary , they can be retrieved easily.

Types of views: Simple View

- With simple views, a select query is written using a single table.
- DML operations are easily performed just like performing on tables.

Create VIEW *Simple_view* as
Select * from CUSTOMER ;

Complex View:

- Complex view is created using a Select query written with multiple tables using JOINS, sub-queries, with clauses and conditionally filtered.

```
Create VIEW Complex_view as
SELECT  c.Cust_Id ,
        a.ACCOUNT
From CUSTOMER c
JOIN ACCOUNT  a
Where c.Cust_Id = a.Cust_Id
And   a.balance > 300000
```

Inline View:

- A subquery is an inline view only if it is used in the FROM clause of a SELECT query.
- It is not stored in any data dictionary like other views. It is dynamically defined in queries

```
SELECT * FROM

( SELECT  c.Cust_Id , ba.ACCOUNT
  From    CUSTOMER bc
JOIN ACCOUNT   ba
Where c.Cust_Id = ba.Cust_Id
And ba.balance > 300000)
```

Refresh Methods

- Fast & Complete.
- A fast refresh uses a log that compares the history and recent changes to its database tables and captures only the changes.
- Complete refresh truncates the whole materialized view data and refreshes with new changed data.
- Complete refresh is slow when compared with fast refresh method.

Advantages and Disadvantages of views

Advantages:

- Views are stored queries and re-used.
- They are accessible by multiple users in different sessions.
- Scalability is high.
- They hide the confidential columns data like SSN, date of birth, Address , telephone.

Advantages:

- Avoids direct access to physical data.
- Views protects the data when views are invoked by downstream users to build web applications that are prone to hackers.
- Materialized views are useful in a centralized environment where multiple users access it in batches of data rather streaming data.

Disadvantages:

- When a base table of an associated view is dropped, it becomes obsolete.
- Queries using views are bit slow compared to accessing data directly from physical tables.
- Because , SQL engine will take an additional processing of semantics checking of view each time when queries are running on it.

Disadvantages:

- Views created using aggregate functions are restricted to use in WHERE clauses.
- Materialized views stores physical data hence it holds redundant data and results in consumption of additional space for its Mviews and Logs.
- Views are in-efficient when joined with remote database tables as they consume lot of I/O transactions between local and remote database tables.

Horizontal View

Horizontal View

- A Horizontal view dynamically represents the data without aware of the columns.
- The view does not necessary to know about the number of columns and its data types of the respective base tables.
- However the base tables are explicitly defined in the view.

Example

```
Create view Horizontal_view as
Select ba.*
From ACCOUNT  where balance > 99999
```

Example

- In this previous example, the wildcard - “*” references all of the columns belongs to base tables mentioned in the view.
- Any changes to the column names , data types doesn’t affect the view creation.

Vertical View

Vertical View

- A vertical view built with predefined columns , however the changes to column data types does not affect the view unless they are type-casted or applied with any system functions.

```
CREATE VIEW Vertical_view AS
SELECT Cust_Id,
       Acct_Num,
       Balance,
      Acct_Type,
      Acct_status,
      Relation
 FROM ACCOUNT
```

Vertical View

- In this example, the columns that are explicitly mentioned while creating the view and those are same as base tables.
- Otherwise it throws an error if any changes applied to columns.

customer_id	account_number	balance_amount	account_type	Account_status	Relation_ship
123002	4000-1956-2001	400000	SAVINGS	ACTIVE	P
123003	4000-1956-2900	750000	SAVINGS	INACTIVE	P
123004	4000-1956-3401	655000	SAVINGS	ACTIVE	P
123001	4000-1956-3456	200000	SAVINGS	ACTIVE	P

Horizontal view Vs Vertical view

Horizontal views	Vertical views
Columns are represented with “*” card	Columns are explicitly defined
Query execution is dynamic and re-calculate the execution plan each time because columns are unknown.	Underlying query execution plan is not repeated because view columns are in sync with base tables.
Maintenance is easy as the view is not worried about changes to columns	View maintenance is required and need to be adjusted according to base table changes.

Vertical View

Horizontal views	Vertical views
Users have access to all view columns like accessing all base table columns.	Many of the columns are hidden.
Users can apply any data type conversions while using the views.	Any data type conversions can be applied unless the other functions inside view definition are not compatible.
Views can be updated easily.	These views can also be updated unless any functions are used on columns in view.

Joined Views

Joined View

- These views represent the data from more than one base tables joined using key columns.

- The underlying query can have JOINS or subqueries on base tables.

Joined View

```
Create View Joined_view
As
Select c.Cust_Id,
       c.Name,
       a.Acct_Num,

       a.Acct_Type,
       a.Balance,
       c.telephone
FROM ACCOUNT  a
JOIN CUSTOMER c
ON a.Cust_Id = c.Cust_Id ;
```

Joined View

customer_id	customer_name	account_number	account_type	balance_amount	telephone
123002	George	4000-1956-2001	SAVINGS	400000	1897617000
123003	Harry	4000-1956-2900	SAVINGS	750000	1897617866
123004	Jack	4000-1956-3401	SAVINGS	655000	1897627999
123001	Oliver	4000-1956-3456	SAVINGS	200000	1897614500
123005	Jacob	4000-1956-5102	SAVINGS	300000	1897637000

Joined View

- In this example, the underlying query is joined with two tables and selected their respective columns.
- From the view point, the representation of columns are viewed as single entity without any complexity.

Drop view

Drop View

- Without affecting the base table data, any type of view can be dropped
  E.g:
  Drop view view_name;

Summary

This slide explains the virtual tables and view. There are multiple types of view and their advantages and disadvantages were also briefly discussed in the above slides but the general idea behind the view is that it provides the logical representation of physical data.

Thank You