View in Oracle

```
create table location(city varchar2(50) not null, country varchar2(50) not null);

insert into location values('Athens', 'Greece');
insert into location values('Chicago', 'United States');
insert into location values('Lima', 'Peru');
insert into location values('Mumbai', 'India');
insert into location values('Manchester', 'England');
insert into location values('Moscow', 'Russia');
insert into location values('Paris', 'France');
insert into location values('Beijing', 'China');
insert into location values('Rome', 'Italy');
insert into location values('Tokyo', 'Japan');
insert into location values('Madrid', 'Spain');

select * from location;

describe location;

create table weather(city varchar2(50) not null, condition varchar2(50) not null);

insert into weather values('Athens', 'Rain');
insert into weather values('Chicago', 'Cloudy');
insert into weather values('Lima', 'Fog');
insert into weather values('Mumbai', 'Sunny');
insert into weather values('Manchester', 'Cloudy');
insert into weather values('Moscow', 'Rain');
insert into weather values('Paris', 'Fog');
insert into weather values('Beijing', 'Rain');
insert into weather values('Rome', 'Sunny');
insert into weather values('Tokyo', 'Cloudy');
insert into weather values('Madrid', 'Sunny');

select * from weather;

alter table weather add(temperature number(2));

update weather set temperature=30 where condition='Rain';
```

update weather set temperature=40 where condition='Cloudy';

update weather set temperature=35 where condition='Fog';

update weather set temperature=41 where condition='Sunny';

1] Creating a view of a single table:-

Syntax:
Create view view_name as select column1, column2, ... from table_name where condition;

create view locationview as select * from location;

select * from locationview;

2] Updating a View:-

A view can be updated with the create or replace view statement.

Syntax:
create or replace view view_name as select column1, column2, ... from table_name where condition;

create or replace view countryview as select * from location;

select * from countryview;

If a view is based on a single underlying table, you can insert, update, or delete rows in the view. This will actually insert, update, or delete rows in the underlying table.

insert into countryview values('Brasilia', 'Brazil');

select * from countryview;

select * from location;

update countryview set city='Chennai' where country='India';

select * from countryview where country='India';

select * from location where country='India';

delete from countryview where country='Brazil';

select * from countryview where country='Brazil';

select * from location where country='Brazil';

3] Creating a view of two or more tables:-

create or replace view weatherlocationview as select weather.city, condition, location.country from weather, location where weather.city=location.city;

The where clause is used to join two tables based on a common column.
The resulting set of data can be turned into a view (with its own name), which can be treated as if it were a regular table itself.
The power of a view is in its ability to limit or change the way data is seen by a user, even though the underlying tables themselves are not affected.

describe weatherlocation;

select * from weatherlocationview;

NOTE: Views do not contain any data. Tables contain data.

4] Only the column which you have selected while creating a view can be seen in the view:-

Now you want to know the temperature of the city from the weatherlocation view:
select city, condition, country, temperature from weatherlocationview;

```
SQL> select city, condition, country, temperature from weatherlocationview;
select city, condition, country, temperature from weatherlocationview
                               *
ERROR at line 1:
ORA-00904: "TEMPERATURE": invalid identifier
```

Error because temperature was not in the select clause when the view was created.

5] If you alter the underlying table and added a new column then that column will not be viewed in the view. Because it was not selected while creating the view:-
Altering the existing table weather:-

```
alter table weather add(warning varchar2(40));
```

The column warning of the weather table cannot be seen in the weatherlocationview.

```
select * from weatherlocationview;
```

Despite the change to the view's base table, the view is still valid, but the warning column is not visible through it. so after altering the table, replace the view to enable the warning column to be seen.

- The results of querying a view are built instantly from a table (or tables) when you execute the query.
- Until that moment, the view has no data of its own, as a table does. It is merely a description (a SQL statement) of what information to pull out of other tables and how to organize it.
- As a consequence, if a table is dropped, the validity of a view is destroyed.
- Attempting to query a view where the underlying table has been dropped will produce an error message about the view.

```
create or replace view locationview as select * from location;

drop table location;

select * from locationview;

SQL> select * from locationview;
select * from locationview
              *
ERROR at line 1:
ORA-04063: view "SYSTEM.LOCATIONVIEW" has errors
```

6] Drop a view:-

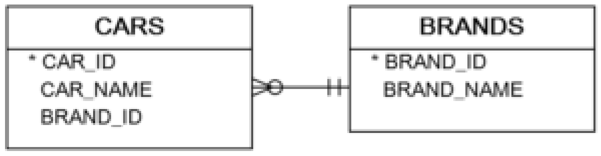
```
 Drop view locationview;
```

Oracle Updatable View

Introduction to Oracle Updatable View

A view behaves like a table because you can query data from it. However, you cannot always manipulate data via views. A view is updatable if the statement against the view can be translated into the corresponding statement against the underlying table.

Let’s consider the following database tables:



In a database diagram, a car belongs to one brand while a brand has one or many cars. The relationship between brand and car is one-to-many.

The following SQL statements create the cars and brands tables; and also insert sample data into these tables.

create table brands(brand_id number generated by default as identity, brand_name varchar2(50) not null, primary key(brand_id));

create table cars (car_id number generated by default as identity, car_name varchar2(255) not null, brand_id number not null, primary key(car_id), foreign key(brand_id) references brands(brand_id) on delete cascade);

INSERT INTO brands (brand_name) VALUES('Audi');

INSERT INTO brands (brand_name) VALUES('BMW');

INSERT INTO brands (brand_name) VALUES('Ford');

INSERT INTO brands (brand_name) VALUES('Honda');

INSERT INTO brands (brand_name) VALUES('Toyota');

INSERT INTO cars (car_name, brand_id) VALUES('Audi R8 Coupe',1);

INSERT INTO cars (car_name, brand_id) VALUES('Audi Q2', 1);

INSERT INTO cars (car_name, brand_id) VALUES('Audi S1', 1);

INSERT INTO cars (car_name, brand_id) VALUES('BMW 2-serie Cabrio',2);

INSERT INTO cars (car_name, brand_id) VALUES('BMW i8', 2);

INSERT INTO cars (car_name, brand_id) VALUES('Ford Edge',3);

INSERT INTO cars (car_name, brand_id) VALUES('Ford Mustang Fastback', 3);

INSERT INTO cars (car_name, brand_id) VALUES('Honda S2000', 4);

INSERT INTO cars (car_name, brand_id) VALUES('Honda Legend', 4);

INSERT INTO cars (car_name, brand_id) VALUES('Toyota GT86', 5);

INSERT INTO cars (car_name, sbrand_id) VALUES('Toyota C-HR', 5);

Oracle updatable view example:

The following statement creates a new view named cars_master:

CREATE VIEW cars_master AS SELECT car_id, car_name FROM cars;

It's possible to delete a row from the cars table via the cars_master view, for example:

DELETE FROM cars_master WHERE car_id = 1;

You can update any column values to the cars_master view:

UPDATE cars_master SET car_name = 'Audi RS7 Sportback' WHERE car_id = 2;

We could insert and update data from the cars table via cars_master view because Oracle can translate the INSERT and UPDATE statements to the corresponding statements and execute them against the cars table.

However, insert a new row into the cars table via the cars_master view is not possible. Because the cars table has a not null column (brand_id) without a default value.

INSERT INTO cars_master VALUES('Audi S1 Sportback');

Oracle issued an error:

SQL Error: ORA-00947: not enough values

Oracle updatable join view example

Let’s create a join view named all_cars based on the cars and brands tables.

```
CREATE VIEW all_cars AS SELECT car_id, car_name, c.brand_id, brand_name FROM cars c
INNER JOIN brands b ON b.brand_id = c.brand_id;
```

The following statement inserts a new row into the cars table via the call_cars view:

```
INSERT INTO all_cars(car_name, brand_id ) VALUES('Audi A5 Cabriolet', 1);
```

A new row has been inserted into the cars table. This INSERT statement works because Oracle can decompose it to an INSERT statement against the cars table.

The following statement deletes all Honda cars from the cars table via the all_cars view:

```
DELETE FROM all_cars WHERE brand_name = 'Honda';
```

Oracle has some rules and restrictions that apply to updatable join views. One of them is the concept of key-preserved tables.

A key-preserved table is a base table with a one-to-one row relationship with the rows in the view, via either the primary key or a unique key. In the example above, the cars table is a key-preserved table.

Here are some examples of updatable join view restrictions:

- The SQL statement e.g., INSERT, UPDATE, and DELETE, is only allowed to modify data from a single base table.
- For an INSERT statement, all columns listed in the INTO clause must belong to a key-preserved table.
- For an UPDATE statement, all columns in the SET clause must belong to a key-preserved table.
- For a DELETE statement, if the join results in more than one key-preserved table, the Oracle deletes from the first table in the FROM clause.

Besides these restrictions, Oracle also requires that the defining query does not contain any of the following elements:

- Aggregate functions e.g., AVG, COUNT, MAX, MIN, and SUM.
- DISTINCT operator.
- GROUP BY clause.
- HAVING clause.
- Set operators e.g., UNION, UNION ALL, INTERSECT, and MINUS.
- START WITH or CONNECT BY clause
- ROWNUM pseudo-column

Find updatable columns of a join view

To find which column can be updated, inserted, or deleted, you use the user_updatable_columns view. The following example shows which column of the all_cars view is updatable, insertable, and deletable:

```
SELECT * FROM USER_UPDATABLE_COLUMNS WHERE TABLE_NAME = 'ALL_CARS';
```

You have learned about the Oracle updatable view and how to update underlying base tables through it.