Views | Normalization

Satyam Govila

Views

In SQL, a view is a virtual table based on the result-set of an SQL statement.

A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

You can add SQL statements and functions to a view and present the data as if the data were coming from one single table.

A view is created with the `CREATE VIEW` statement.

Significance of Views

- VIEWS increase re-usability.
- VIEWS help in data security. You can use views to show only authorized information to users and hide sensitive data like credit card numbers.
- If you have any frequently used complex query, you can create a view based on it so that you can reference to the view by using a simple SELECT statement instead of typing the query all over again.

Note : *View is unique in a database Because views and tables in the same database share the same namespace, the name a view cannot the same as the name of an existing table.*

View : Syntax and Example

```
CREATE VIEW view_name AS
SELECT column1, column2.....
FROM table_name
WHERE [condition];
```

Updating a View

A view can be updated under certain conditions which are given below –

- The SELECT clause may not contain the keyword DISTINCT.
- The SELECT clause may not contain summary functions.
- The SELECT clause may not contain set functions.
- The SELECT clause may not contain set operators.
- The SELECT clause may not contain an ORDER BY clause.
- The FROM clause may not contain multiple tables.
- The WHERE clause may not contain subqueries.
- The query may not contain GROUP BY or HAVING.
- Calculated columns may not be updated.
- All NOT NULL columns from the base table must be included in the view in order for the INSERT query to function.

So, if a view satisfies all the above-mentioned rules then you can update that view.

View : Deleting a view

```
DROP VIEW view_name;
```

SQL creating VIEW with JOIN

Sample table: orders Alias ‘a’

ORD_NUM	ORD_AMOUNT	ADVANCE_AMOUNT	ORD_DATE	CUST_CODE	AGENT_CODE
200114	3500	2000	15-AUG-08	C00002	A008
200122	2500	400	16-SEP-08	C00003	A004
200118	500	100	20-JUL-08	C00023	A006
200119	4000	700	16-SEP-08	C00007	A010
200121	1500	600	23-SEP-08	C00008	A004
200130	2500	400	30-JUL-08	C00025	A011
200134	4200	1800	25-SEP-08	C00004	A005
200108	4000	600	15-FEB-08	C00008	A004
200103	1500	700	15-MAY-08	C00021	A005

Sample table: agents Alias ‘c’

AGENT_CODE	AGENT_NAME	WORKING_AREA	COMMISSION	PHONE_NO
A007	Ramasundar	Bangalore	0.15	077-25814763
A003	Alex	London	0.13	075-12458969
A008	Alford	New York	0.12	044-25874365
A011	Ravi Kumar	Bangalore	0.15	077-45625874
A010	Santakumar	Chennai	0.14	007-22388644
A012	Lucida	San Jose	0.12	044-52981425
A005	Anderson	Brisban	0.13	045-21447739
A001	Subbarao	Bangalore	0.14	077-12346674

Sample table: customer Alias ‘b’

CUST_CODE	CUST_NAME	CUST_CITY	WORKING_AREA	CUST_COUNTRY	GRADE	OPENING_AMT	RECEIVE_AMT	PAYMENT_AMT	OUTSTANDING_AMT	PHONE_NO	AGENT_CODE
C00013	Holmes	London	London	UK	2	6000.00	5000.00	7000.00	4000.00	BBBBBBB	A003
C00001	Micheal	New York	New York	USA	2	3000.00	5000.00	2000.00	6000.00	CCCCCCC	A008
C00020	Albert	New York	New York	USA	3	5000.00	7000.00	6000.00	6000.00	BBBBSBB	A008
C00025	Ravindran	Bangalore	Bangalore	India	2	5000.00	7000.00	4000.00	8000.00	AVAVAVA	A011
C00024	Cook	London	London	UK	2	4000.00	9000.00	7000.00	6000.00	FSDDSDF	A006
C00015	Stuart	London	London	UK	1	6000.00	8000.00	3000.00	11000.00	GFSGERS	A003
C00002	Bolt	New York	New York	USA	3	5000.00	7000.00	9000.00	3000.00	DDNRDRH	A008
C00018	Fleming	Brisban	Brisban	Australia	2	7000.00	7000.00	9000.00	5000.00	NHBGVFC	A005

SQL creating VIEW with JOIN

```
1  CREATE VIEW ordersview
2  AS SELECT ord_num, ord_amount, a.agent_code,
3  agent_name, cust_name
4  FROM orders a, customer b, agents c
5  WHERE a.cust_code=b.cust_code
6  AND a.agent_code=c.agent_code;
```

To create a view 'ordersview' by three tables 'orders', 'customer' and ' agents' with following conditions -

1. 'a' and 'b' and 'c' are the aliases of 'orders' and 'customer' and 'agents' table,

2. 'cust_code' of 'orders' and 'customer' table must be same,

3. 'agent_code' of 'orders' and 'agents' table must be same,

Keys in DBMS

KEYS in DBMS is an attribute or set of attributes which helps you to identify a row(tuple) in a relation(table).

They allow us to find the relation between two tables.

Types of Keys in DBMS

There are mainly Eight different types of Keys in DBMS and each key has it’s different functionality:

1. Super Key
2. Primary Key
3. Candidate Key
4. Alternate Key
5. Foreign Key
6. Compound Key
7. Composite Key
8. Surrogate Key

Types of Keys in DBMS

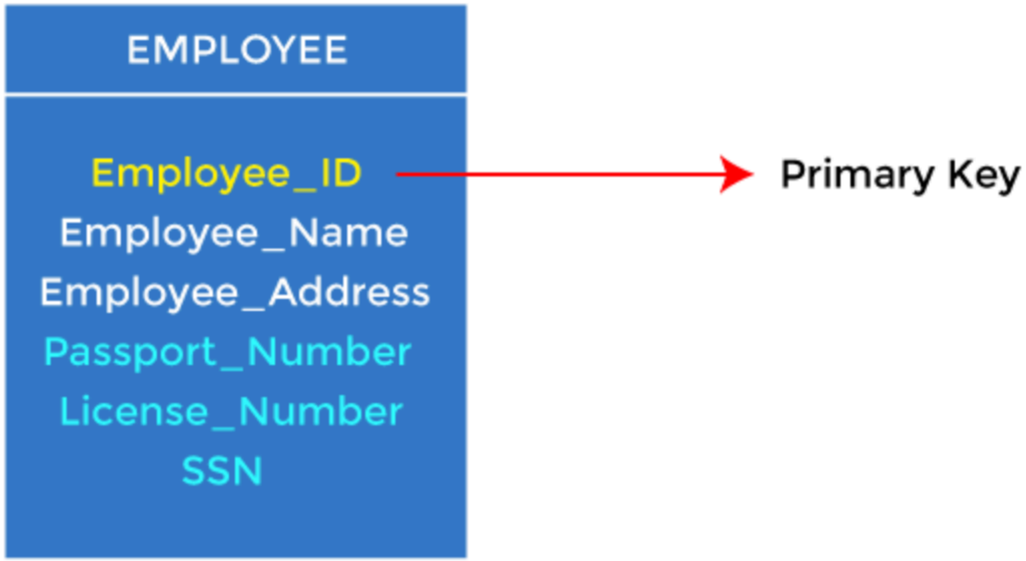
- **Super Key –** A super key is a group of single or multiple keys which identifies rows in a table.
- **Primary Key –** is a column or group of columns in a table that uniquely identify every row in that table.
- **Candidate Key –** is a set of attributes that uniquely identify tuples in a table. Candidate Key is a super key with no repeated attributes.
- **Foreign Key –** is a column that creates a relationship between two tables. The purpose of Foreign keys is to maintain data integrity and allow navigation between two different instances of an entity.

SQL Primary Key

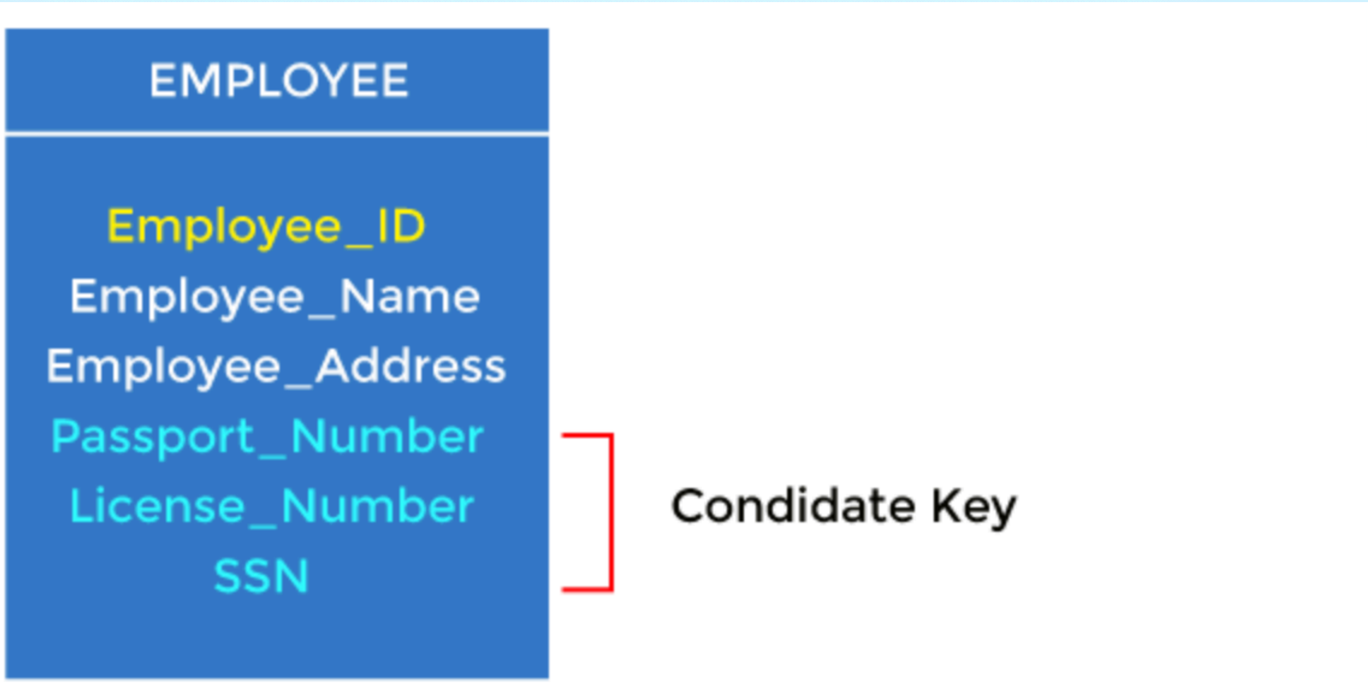
The `PRIMARY KEY` constraint uniquely identifies each record in a table.

Primary keys must contain UNIQUE values, and cannot contain NULL values.

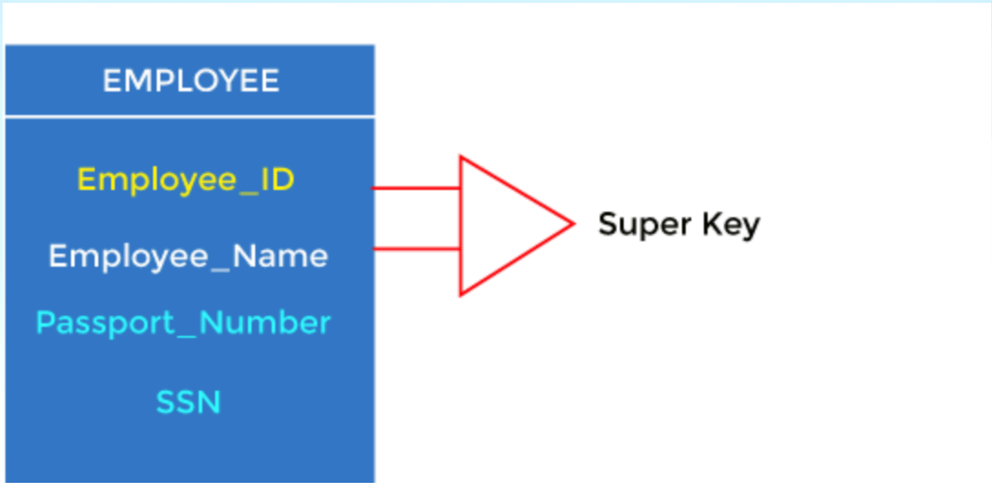
A table can have only ONE primary key; and in the table, this primary key can consist of single or multiple columns (fields).



SQL Candidate Key

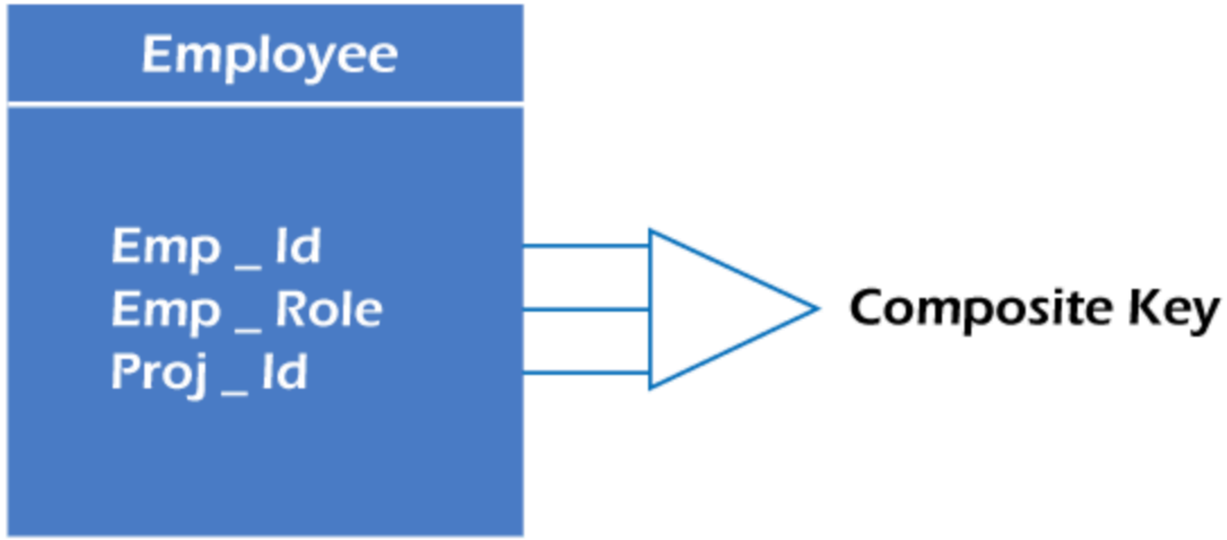


SQL Super Key



SQL Composite Key

Whenever a primary key consists of more than one attribute, it is known as a composite key. This key is also known as Concatenated Key.



SQL Foreign Key

The `FOREIGN KEY` constraint is used to prevent actions that would destroy links between tables.

A `FOREIGN KEY` is a field (or collection of fields) in one table, that refers to the `PRIMARY KEY` in another table.

The table with the foreign key is called the child table, and the table with the primary key is called the referenced or parent table.

SQL Foreign Key

Persons Table

PersonID	LastName	FirstName	Age
1	Hansen	Ola	30
2	Svendson	Tove	23
3	Pettersen	Kari	20

Orders Table

OrderID	OrderNumber	PersonID
1	77895	3
2	44678	3
3	22456	2
4	24562	1

Notice that the "PersonID" column in the "Orders" table points to the "PersonID" column in the "Persons" table.

The "PersonID" column in the "Persons" table is the `PRIMARY KEY` in the "Persons" table.

The "PersonID" column in the "Orders" table is a `FOREIGN KEY` in the "Orders" table.

The `FOREIGN KEY` constraint prevents invalid data from being inserted into the foreign key column, because it has to be one of the values contained in the parent table.

Analysing Database Design

Let’s consider a single large dataset having only single relation.

This large database defined as a single relation may result in data duplication.

Can you think of the disadvantages of having a large database with repetitive data?

Analysing Database Design

This repetition of data may result in:

- Making relations very large.
- Difficult to maintain and update data as it would involve searching many records in relation.
- Wastage and poor utilisation of disk space and resources.
- The likelihood of errors and inconsistencies increases.

How should we handle this problem?

Solution : Normalization

What is Normalization?

- Normalization is a process of decomposing the relations into smaller, simpler, and well-structured relations with fewer attributes.
- It is the process of organising the data in the database.
- It is used to minimise the data redundancy from a relation or set of relations and is also used to eliminate undesirable characteristics like Insertion, Update, and Deletion Anomalies.
- Normalization consists of a series of guidelines that helps to guide you in creating a good database structure.

Any Questions?