Difference between SQL Keys (Primary Key, Super Key, Candidate Key, Foreign Key)

BEGINNER SQL STRUCTURED DATA TECHNIQUE

Introduction

> SQL Keys is the Key to your success in Analytics!

Data is growing at an exponential rate and so is the demand for professionals who are well versed with the databases.

Organizations all over the world are looking for data scientists and analysts who can draw meaningful insights from the vast amounts of data. And one of the most important languages for handling databases is SQL. That is why those professionals with a background in SQL have an edge over their peers when it comes to working with databases.

An important aspect of working with databases is actually creating one. Creating a database is an altogether different ball game when compared to retrieving data from databases. Why so? Well, creating a database requires a thorough knowledge of how the tables relate to each other within a database and how to handle records so that there is no duplicate data.

A very important aspect of creating databases is understanding the concept of Keys in SQL. These are nothing but a group of columns that can help you identify rows in a table uniquely. But like any other entity, there are many kinds of Keys in SQL.

In this article, I will discuss some of the most common SQL keys that any data scientist or analyst should know before they even start working with databases!

I suggest checking out the SQL course for Data Science if you're new to SQL.

Table of Contents

What are keys in DBMS?

Databases are used to store massive amounts of information which is stored across multiple tables. Each table might be running into thousands of rows. Needless to say, there will be many duplicate rows with redundant information. How do we deal with that? How do we manage records so that we are storing only unique data? And, how do we relate the multiple tables that are present in the database?

SQL keys are the answer to all these queries.

> An SQL key is either a single column (or attribute) or a group of columns that can uniquely identify rows (or tuples) in a table.

SQL keys ensure that there are no rows with duplicate information. Not only that, but they also help in establishing a relationship between multiple tables in the database. Therefore, it becomes imperative to learn about the different keys in SQL.

What is a Super key in SQL?

> Super key is a single key or a group of multiple keys that can uniquely identify tuples in a table.

Super Key can contain multiple attributes that might not be able to independently identify tuples in a table, but when grouped with certain keys, they can identify tuples uniquely.

Let me take an example to clarify the above statement. Have a look at the following table.

Id	Name	Gender	City	Email	Dep_Id
1	Ajay	M	Delhi	ajay@gmail.com	1
2	Vijay	M	Mumbai	vijay@gmail.com	2
3	Radhika	F	Bhopal	radhika@gmail.com	1
4	Shikha	F	Jaipur	shikha@gmail.com	2
5	Hritik	M	Jaipur	hritik@gmail.com	2

5 rows in set (0.00 sec)

Consider that *Id* attribute is unique to every employee. In that case, we can say that the *Id* attribute can uniquely identify the tuples of this table. So, *Id* is a Super key of this table. Note that we can have other Super Keys too in this table.

For instance – *(Id, Name), (Id, Email), (Id, Name, Email)*, etc. can all be Super keys as they can all uniquely identify the tuples of the table. This is so because of the presence of the *Id* attribute which is able to uniquely identify the tuples. The other attributes in the keys are unnecessary. Nevertheless, they can still identify tuples.

What is a Candidate key?

> Candidate key is a single key or a group of multiple keys that uniquely identify rows in a table.

A Candidate key is a subset of Super keys and is devoid of any unnecessary attributes that are not important for uniquely identifying tuples.

The value for the Candidate key is unique and non-null for all tuples. And every table has to have at least one Candidate key. But there can be more than one Candidate Key too.

For example, in the example that we took earlier, both *Id* and *Email* can act as a Candidate for the table as they contain unique and non-null values.

Id	Name	Gender	City	Email	Dep_Id
1	Ajay	M	Delhi	ajay@gmail.com	1
2	Vijay	M	Mumbai	vijay@gmail.com	2
3	Radhika	F	Bhopal	radhika@gmail.com	1
4	Shikha	F	Jaipur	shikha@gmail.com	2
5	Hritik	M	Jaipur	hritik@gmail.com	2

5 rows in set (0.00 sec)

On the other hand, we cannot use the attributes like *City* or *Gender* to retrieve tuples from the table as they have no unique values.

```
mysql> Select * from Employee where Gender='M';
```

Id	Name	Gender	City	Email	Dep_Id
1	Ajay	M	Delhi	ajay@gmail.com	1
2	Vijay	M	Mumbai	vijay@gmail.com	2
5	Hritik	M	Jaipur	hritik@gmail.com	2

3 rows in set (0.01 sec)

Whereas on querying the table on the *Id* attribute will help us to retrieve unique tuples.

```
mysql> Select * from Employee where Id=3;
+----+---------+--------+--------+-------------------+--------+
| Id | Name    | Gender | City   | Email             | Dep_Id |
+----+---------+--------+--------+-------------------+--------+
| 3  | Radhika | F      | Bhopal | radhika@gmail.com |      1 |
+----+---------+--------+--------+-------------------+--------+
1 row in set (0.00 sec)
```

Primary Key in SQL

> Primary key is the Candidate key selected by the database administrator to uniquely identify tuples in a table.

Out of all the Candidate keys that can be possible for a table, there can be only one key that will be used to retrieve unique tuples from the table. This Candidate key is called the Primary Key.

There can be only one Primary key for a table. Depending on how the Candidate Key is constructed the primary key can be a single attribute or a group of attributes. But the important point to remember is that the Primary key should be a unique and non-null attribute(s).

There can be two ways to create a Primary key for the table. The first way is to alter an already created to add the Primary key constraint on an attribute. This is shown below:

```
mysql> Alter table Employee Add Primary Key(Id);
Query OK, 0 rows affected (3.10 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Now if I try to add a new row with duplicate Id value, it will give me an error message.

```
mysql> Insert into Employee values(5,'Abhay','M','Pune','abhay_77@ymail.com');
ERROR 1062 (23000): Duplicate entry '5' for key 'employee.PRIMARY'
```

The second way of adding a Primary key is during the creation of the table itself. All you have to do is add the Primary Key constraint at the end after defining all the attributes in the table.

```
mysql> Create table Student(S_Id int Not Null,
    -> Name varchar(20),
    -> Gender varchar(3),
    -> Class int,
    -> Primary Key(S_Id));
Query OK, 0 rows affected (0.75 sec)
```

To define a Primary Key constraint on multiple attributes, you can list all the attributes in the parenthesis as shown below.

```
mysql> Create table Person(Id int Not Null,
    -> Name varchar(20),
    -> City varchar(20),
    -> Email varchar(20) Not Null,
    -> Phone varchar(10) Not Null,
    -> Primary Key(Id, Email, Phone));
Query OK, 0 rows affected (0.79 sec)
```

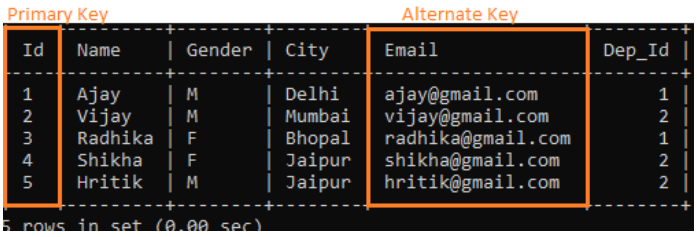
But remember that these attributes should be defined as non-null values otherwise the whole purpose of using the Primary key to identify tuples uniquely gets defeated.

Alternate or Secondary keys in SQL

> Alternate keys are those candidate keys which are not the Primary key.

There can be only one Primary key for a table. Therefore all the remaining Candidate keys are known as Alternate or Secondary keys. They can also uniquely identify tuples in a table, but the database administrator chose a different key as the Primary key.

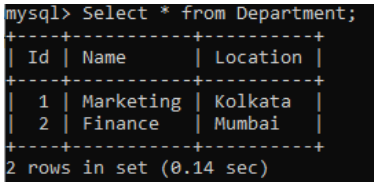
If we look at the Employee table once again, since I have chosen *Id* as the Primary key, the other Candidate Key (*Email*), becomes the Alternate key for the table.



Foreign key in SQL

> Foreign key is an attribute which is a Primary key in its parent table, but is included as an attribute in another host table.

A Foreign key generates a relationship between the parent table and the host table. For example, in addition to the *Employee* table containing the personal details of the employees, we might have another table *Department* containing information related to the department of the employee.



The Primary key in this table is the Department *Id*. We can add this attribute to the Employee by making it the Foreign key in the table. We can either do this when we are creating the table or we can alter the table later to add the Foreign Key constraint. Here I have altered the table, but creating Foreign Key during table creation is similar to that for Primary Key.

```
mysql> Alter table Employee
    -> Add Foreign Key(Dep_Id) References Department(Id);
Query OK, 5 rows affected (3.37 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

Here, *Dep_Id* is now the Foreign Key in table Employee while it is a Primary Key in the Department table.

The Foreign key allows you to create a relationship between two tables in the database. Each of these tables describes data related to a particular field (employee and department here). Using the Foreign key we can easily retrieve data from both the tables.

```
mysql> select employee.name, employee.city, department.name
    -> from employee inner join department
    -> on employee.dep_id = department.id;
+---------+--------+-----------+
| name    | city   | name      |
+---------+--------+-----------+
| Ajay    | Delhi  | Marketing |
| Radhika | Bhopal | Marketing |
| Vijay   | Mumbai | Finance   |
| Shikha  | Jaipur | Finance   |
| Hritik  | Jaipur | Finance   |
+---------+--------+-----------+
5 rows in set (0.00 sec)
```

Note: To operate on Foreign keys, you need to know about Joins which you can find out in detail in [this article](#).

Using Foreign keys makes it easier to update the database when required. This is so because we only have to make the necessary changes in limited rows. For example, if the *Marketing* department shifts from *Kolkata* to *Pune*, instead of updating it for all the relevant rows in the *Employee* table, we can simply update the location in the *Department* table. This ensures that there are only a few places to update and less risk of having different data in different places.

What are Composite keys?

> A Composite key is a Candidate key or Primary key that consists of more than one attribute.

Sometimes it is possible that no single attribute will have the property to uniquely identify tuples in a table. In such cases, we can use a group of attributes to guarantee uniqueness. Combining these attributes will uniquely identify tuples in the table.

Consider the following table:

```
mysql> select * from product;
+----------------+------------+-------------+--------------+----------+
| Transaction_Id | Product_Id | Customer_Id | Product      | Quantity |
+----------------+------------+-------------+--------------+----------+
| A1001          | P1005      | C9001       | Smartphone   |        1 |
| A1001          | P2010      | C9001       | Screen guard |        1 |
| A1002          | P2013      | C9003       | Smartwatch   |        1 |
| A1003          | P2010      | C9010       | Screen guard |        2 |
+----------------+------------+-------------+--------------+----------+
4 rows in set (0.00 sec)
```

Here, neither of the attributes contains unique values to identify the tuples. Therefore, we can combine two or more attributes to create a key that can uniquely identify the tuples. For example, we can group *Transaction_Id* and *Product_Id* to create a key that can uniquely identify the tuples. These are called composite keys.

Key difference between SQL Keys

- **SQL keys** are used to uniquely identify rows in a table.
- **SQL keys can either be a single column or a group of columns.**
- **Super key** is a single key or a group of multiple keys that can uniquely identify tuples in a table.
- **Super keys** can contain redundant attributes that might not be important for identifying tuples.
- **Candidate keys** are a subset of **Super keys**. They contain only those attributes which are required to uniquely identify tuples.
- All **Candidate keys** are **Super keys**. But the vice-versa is not true.
- **Primary key** is a **Candidate key** chosen to uniquely identify tuples in the table.
- **Primary key** values should be unique and non-null.
- There can be multiple **Super keys** and **Candidate keys** in a table, but there can be only one **Primary key** in a table.
- **Alternate keys** are those **Candidate keys** that were not chosen to be the Primary key of the table.
- **Composite key** is a **Candidate key** that consists of more than one attribute.
- **Foreign key** is an attribute which is a **Primary key** in its parent table but is included as an attribute in the host table.
- **Foreign keys** may accept non-unique and null values.

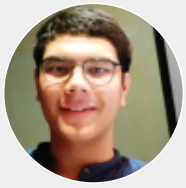
Endnotes

In this article, we covered the most common and widely used keys that any professionals looking to work with databases should know about.

If you are someone who is looking to work with SQL in Python, then I suggest going through this article. Or if you are someone who is looking for some SQL techniques to employ for better data analysis, then you shouldn't miss this great article.

I hope you enjoyed this article and do connect in the comments if you have any doubts about this topic!

Article Url - https://www.analyticsvidhya.com/blog/2020/07/difference-between-sql-keys-primary-key-super-key-candidate-key-foreign-key/



Aniruddha Bhandari

I am on a journey to becoming a data scientist. I love to unravel trends in data, visualize it and predict the future with ML algorithms! But the most satisfying part of this journey is sharing my learnings, from the challenges that I face, with the community to make the world a better place!