Introduction to Database Keys

In this lesson, we will highlight some of keys that are present in databases.

WE'LL COVER THE FOLLOWING

Keys are a very important part of the relational database model. They are used to establish and identify relationships between tables and also to uniquely identify any record or row of data inside a table.

A key can be a single attribute or a group of attributes, where the combination may act as a key.

Why do we need a key?

In real-world applications, the number of tables required for storing data is huge, and different tables are related to each other as well.

Also, tables store a lot of data. Tables generally extend to thousands of records, unsorted and unorganized.

Now, to fetch any particular record from such a dataset, you will have to apply some conditions. But what if there is duplicate data present and every time you try to fetch some data by applying certain conditions, you get the wrong data? How many trials before you get the right data?

To avoid all this, keys are defined to easily identify any row of data in a table.

Let's try to understand what keys are all about using the example of a STUDENT table:

Std_Id	Name	Phone
1	John	(201)6723452
2	Adam	(202)1165674
3	Bruce	(480)7867898
4	James	(516)0080080

Super key

A super key is defined as a set of attributes within a table that can uniquely identify each record within a table.

In the table defined above, the super key would include `Std_Id`, (`Std_Id`, `Name`), `Phone` etc.

Confused? The first one is pretty simple as `Std_Id` is unique for every row of data, hence it can be used to identify each row uniquely.

Next comes, (`Std_Id`, `Name`), now `Name` of two students can be the same, but their `Std_Id` can't be same hence this combination can also be a super key.

Similarly, the phone number for every student will be unique (provided it is his/her mobile number), hence, again, `Phone` can also be a key.

So they all are super keys.

Candidate key

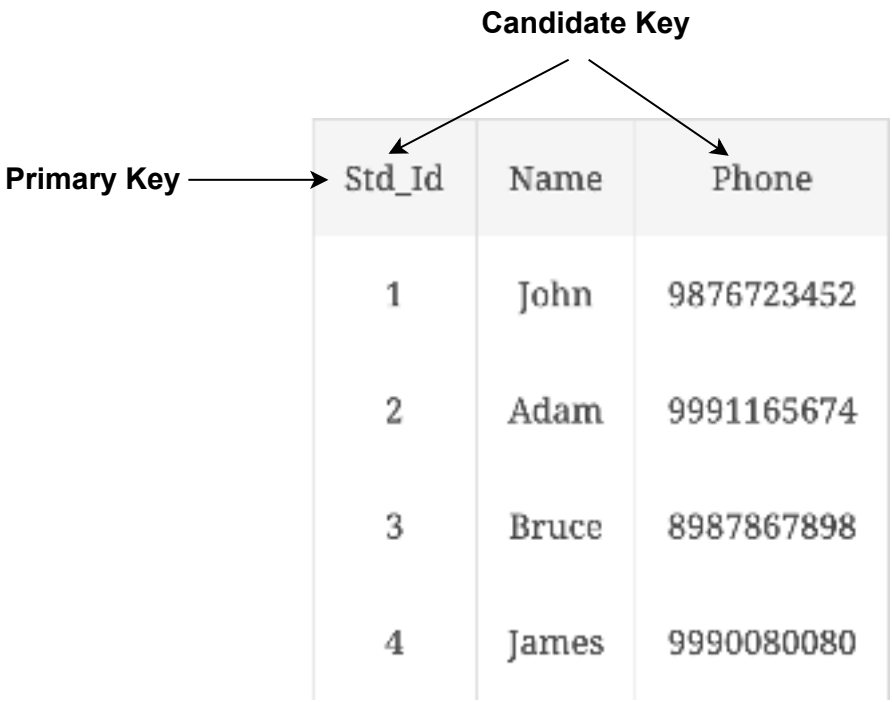
Candidate keys are defined as the minimal set of fields that can uniquely identify each record in a table. There can be more than one candidate key.

In our example, `Std_Id` and `Phone` both are candidate keys for the STUDENT table. The (`Std_Id`, `Name`) super key is not a candidate key as we can remove the `Name` field and still be able to uniquely identify each record.

Furthermore, a candidate key can never be `NULL` or empty. Its value should be unique. Also, a candidate key can be a combination of more than one column (attributes).

Primary key

There can be more than one candidate key in a relation, out of which one can be chosen as the primary key. For Example, `Std_Id` as well as `Phone` both candidate keys for relation STUDENT but `Std_Id` can be chosen as the primary key (only one out of many candidate keys).



Composite key

A key that consists of two or more attributes that uniquely identify any record in a table is called a composite key. But the attributes which together form the composite key are not a key independently or individually.

Std_Id	Subject_Id	Marks
1	8	70

1	12	90
2	12	65

In the above MARKS table, we have the marks scored by a student in a particular subject. In this table `Std_Id` and `Subject_Id` together will form the primary key, hence it is a composite key.

Alternate key

The candidate key other than the primary key is called an alternate key. For Example, `Std_Id` as well as `Phone`, are candidate keys for relation STUDENT but `Phone` will be the alternate key.

Foreign key

A foreign key is a column or group of columns in a relational database table that provides a link between the data in two tables. It acts as a cross-reference between tables because it references the primary key of another table, thereby establishing a link between them. Let's take a look at a simple example:

Student table:

Std_Id	Name	Phone
1	John	9876723452
2	Adam	9991165674
3	Bruce	8987867898
4	James	9990080080

Course table:

Std_Id	Course_No	Course_Name
1	C1	Software Engineering
2	C1	Software Engineering
1	C2	Networks

The `Std_Id` in the COURSE relation acts as the foreign key to `Std_Id` in the STUDENT relation.

It may be worth noting that unlike the primary key of any given relation, foreign keys can be NULL or may contain duplicate tuples, i.e., it need not follow uniqueness constraint.

For Example, `Std_Id` in the COURSE relation is not unique. It has been repeated for the first and third tuple. However, the `Std_Id` in STUDENT relation is a primary key and it needs to be always unique and it cannot be `NULL`.

In the next lesson, we will dive deep into the rules and constraints placed upon relational databases.