Database Keys

Contents

- Introduction
- Primary Key
- Foreign Key
- Secondary Key or Alternative Key
- Simple Key
- Compound Key

1. Introduction

For the purposes of clarity we will refer to keys in terms of RDBMS tables but the same definition, principle and naming applies equally to Entity Modelling and normalization.

Keys are, as their name suggests, a key part of a relational database and a vital part of the structure of a table. They ensure each record within a table can be uniquely identified by one or a combination of fields within the table. They help enforce integrity and help identify the relationship between tables. There are three main types of keys, candidate keys, primary keys and foreign keys. There is also an alternative key or secondary key that can be used, as the name suggests, as a secondary or alternative key to the primary key

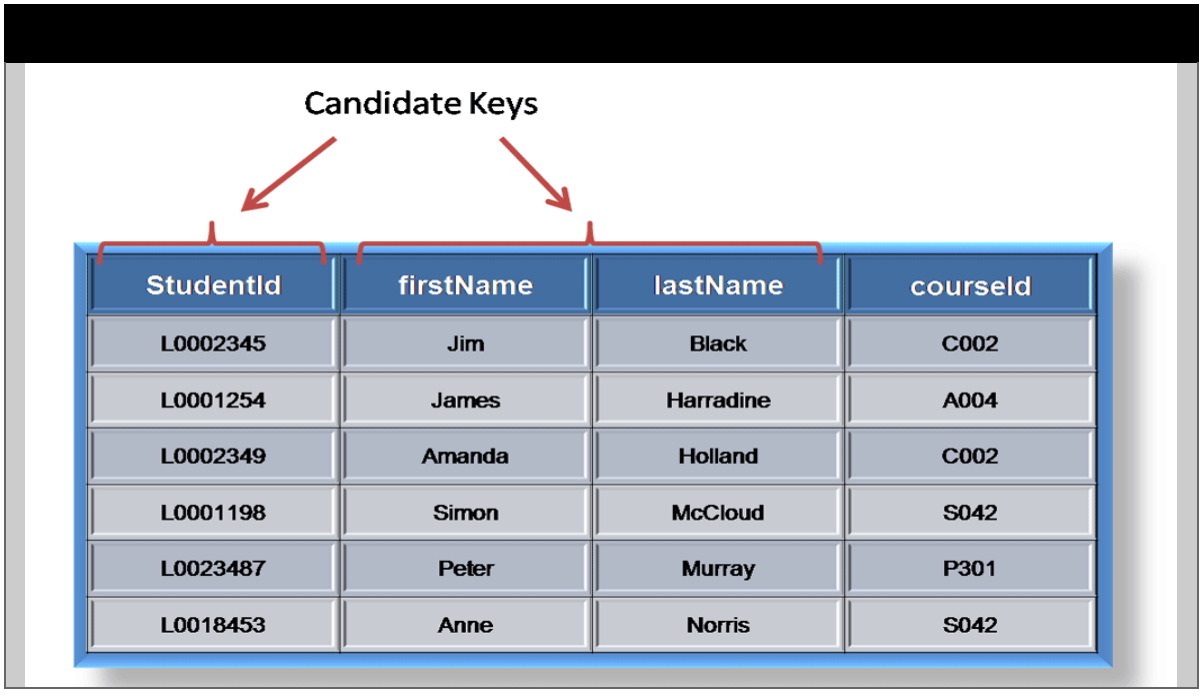
- ✓ ***Super Key***

  A Super key is any combination of fields within a table that uniquely identifies each record within that table.

- ✓ ***Candidate Key***

  A candidate is a subset of a super key. A candidate key is a single field or the least combination of fields that uniquely identifies each record in the table. The least combination of fields distinguishes a candidate key from a super key. Every table must have at least one candidate key but at the same time can have several.

.



StudentId	firstName	lastName	courseId
L0002345	Jim	Black	C002
L0001254	James	Harradine	A004
L0002349	Amanda	Holland	C002
L0001198	Simon	McCloud	S042
L0023487	Peter	Murray	P301
L0018453	Anne	Norris	S042

As an example, we might have a ***student***_id that uniquely identifies the students in a student table. This would be a candidate key. But in the same table we might have the student's first name and last name that also, when combined, uniquely identify the student in a student table. These would both be candidate keys.

In order to be eligible for a candidate key it must pass certain criteria.

It must contain unique values

It must not contain null values

It contains the minimum number of fields to ensure uniqueness

It must uniquely identify each record in the table

Once your candidate keys have been identified, you can now select one to be your primary key

- ✓ ***Primary Key***

  A primary key is a candidate key that is most appropriate to be the main reference key for the table. As its name suggests, it is the primary key of reference for the table and is used throughout the database to help establish relationships with other tables. As with any candidate key the primary key must contain unique values, must never be null and uniquely identify each record in the table.

  As an example, a student id might be a primary key in a student table, a department code in a table of all departments in an organization. This module has the code DH3D 35 that is no doubt used in a database somewhere to identify RDBMS as a unit in a table of modules. In the table below we have selected the candidate key ***student***_id to be our most appropriate primary key

Primary Keys

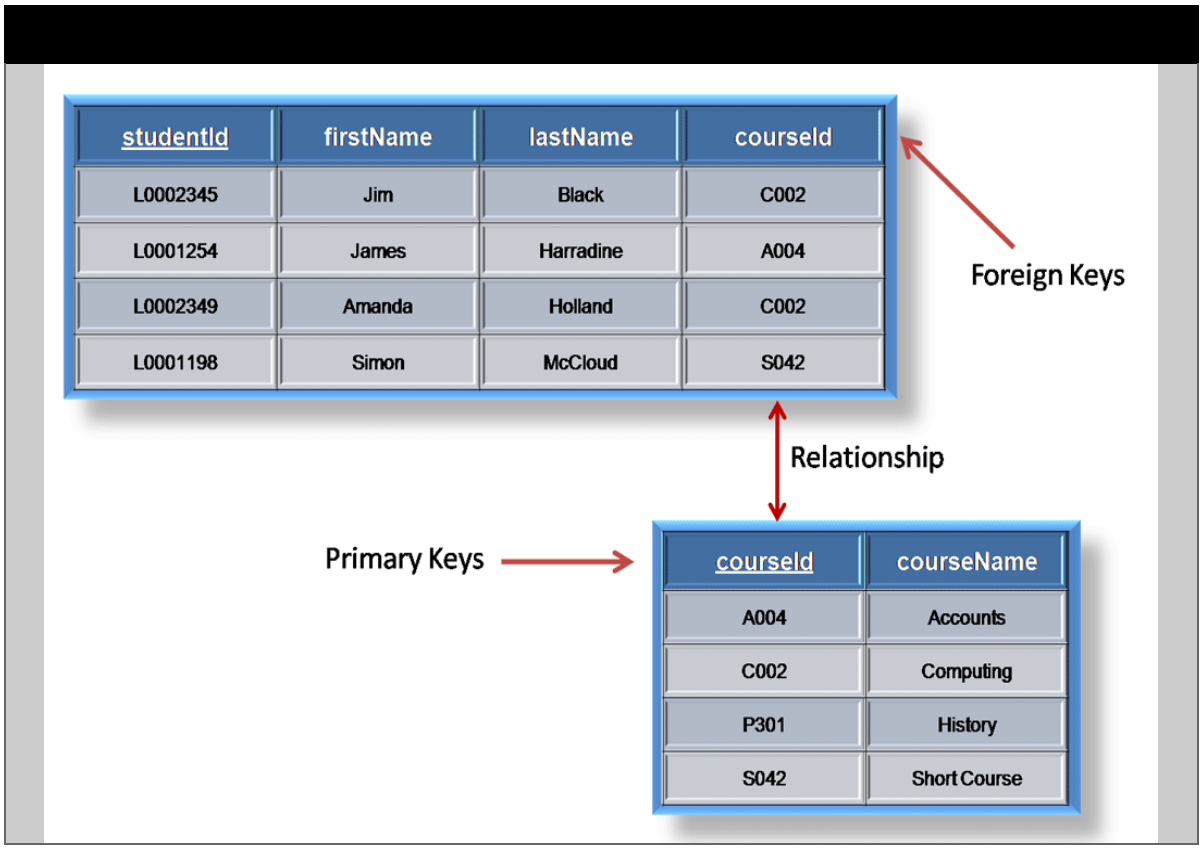
StudentId	firstName	lastName	courseId
L0002345	Jim	Black	C002
L0001254	James	Harradine	A004
L0002349	Amanda	Holland	C002
L0001198	Simon	McCloud	S042
L0023487	Peter	Murray	P301
L0018453	Anne	Norris	S042

Primary keys are mandatory for every table each record must have a value for its primary key. When choosing a primary key from the pool of candidate keys always choose a single simple key over a composite key.

- ✓ ***Foreign Key***

  A foreign key is generally a primary key from one table that appears as a field in another where the first table has a relationship to the second. In other words, if we had a table A with a primary key X that linked to a table B where X was a field in B, then X would be a foreign key in B.

  An example might be a student table that contains the ***course_id*** the student is attending. Another table lists the courses on offer with ***course_id*** being the primary key. The 2 tables are linked through ***course_id*** and as such ***course_id*** would be a foreign key in the student table.



✓ ***Secondary Key or Alternative Key***

A table may have one or more choices for the primary key. Collectively these are known as candidate keys as discuss earlier. One is selected as the primary key. Those not selected are known as secondary keys or alternative keys.

For example in the table showing candidate keys above we identified two candidate keys, ***studentId*** and ***firstName*** + ***lastName***. The ***studentId*** would be the most appropriate for a primary key leaving the other candidate key as secondary or alternative key. It should be noted for the other key to be candidate keys, we are assuming you will never have a person with the same first and last name combination. As this is unlikely we might consider ***fistName+lastName*** to be a suspect candidate key as it would be restrictive of the data you might enter. It would seem a shame to not allow John Smith onto a course just because there was already another John Smith.

✓ ***Simple Key***

Any of the keys described before (ie primary, secondary or foreign) may comprise one or more fields, for example if ***firstName***

and ***lastName*** was our key this would be a key of two fields where as ***studentId*** is only one. A simple key consists of a single field to uniquely identify a record. In addition the field in itself cannot be broken down into other fields, for example, ***studentId***, which uniquely identifies a particular student, is a single field and therefore is a simple key. No two students would have the same student number.

- ✓ ***Compound Key***

  A compound key consists of more than one field to uniquely identify a record. A compound key is distinguished from a composite key because each field, which makes up the primary key, is also a simple key in its own right. An example might be a table that represents the modules a student is attending. This table has a ***studentId*** and a ***moduleCode*** as its primary key. Each of the fields that make up the primary key are simple keys because each represents a unique reference when identifying a student in one instance and a module in the other.

- ✓ ***Composite***

  A composite key consists of more than one field to uniquely identify a record. This differs from a compound key in that one or more of the attributes, which make up the key, are not simple keys in their own right. Taking the example from compound key, imagine we identified a student by their ***firstName*** + ***lastName***. In our table representing students on modules our primary key would now be ***firstName*** + ***lastName*** + ***moduleCode***. Because ***firstName*** + ***lastName*** represent a unique reference to a student, they are not each simple keys, they have to be combined in order to uniquely identify the student. Therefore the key for this table is a composite key.

Example:

Consider a Relation or Table R1. Let **A,B,C,D,E** are the attributes of this relation. **R(A,B,C,D,E)**

A→BCDE This means the attribute 'A' uniquely determines the other attributes B,C,D,E.

BC→ADE This means the attributes 'BC' jointly determines all the other attributes A,D,E in the relation.

Primary Key :A
Candidate Keys :A, BC
Super Keys : A,BC,ABC,AD

ABC,AD are not Candidate Keys since both are not minimal super keys.

- A Practical Example Consider the relation 'Store' with the attributes **'storeNo'**, **'street'**, **'city'** and **'postcode'**. Store (**storeNo**, **street**, **city**, **postcode**) given a value of 'city' we may be able to determine several stores (there could be more than one store in a particular **city**), as such this attribute cannot be used as a candidate key. if we say store number is a unique number that is given to each store then given a value of **'storeNo'** we can only return at most one tuple(because it is a unique identifier) so that is a candidate key. similarly in this example 'postcode' is also a candidate key because it is unique to each store. now consider a relation "Sales" which contains information relating to sales at the stores Sales (**customerNo**, **productNo**, **saleDate**) given a **'customerNo'** there may be several corresponding sales for different products. Similarly given a **'productNo'** there may be several customers who have purchased this item. Therefore customer number by its self or product number by its self cannot be selected as a candidate key. However a combination of both **'customerNo'** and **'productNo'** identifies at one tuple uniquely so this is a candidate key (combining may be refereed to as a composite key but that is just a type of candidate key)

For example, given an employee schema, consisting of the attributes **employeeID**, **name**, **job**, and **departmentID**, we could use the **employeeID** in combination with any or all other attributes of this table to uniquely identify a tuple in the table. Examples of superkeys in this schema would be {**employeeID**, **Name**}, {**employeeID**, **Name**, **job**}, and {**employeeID**, **Name**, **job**, **departmentID**}. The last example is known as trivial superkey, because it uses all attributes of this table to identify the tuple.