A Relational Model

A relational model stores data in the form of tables; these tables are called relations. A relational model is based on the fact that tables are related to each other.

**Every table in a relational model has the following characteristics:**

1. In a relational model database, the name of every table must be unique.
2. The name of every attribute of a table must be unique to that table.
3. Every table must have a key attribute. The value of this key attribute cannot be the same for any two rows of that table.
4. The data type for each attribute must be defined.
5. The order of the rows does not matter in relation.
6. The order of the attributes does not matter in relation.

Every row or a tuple in a table represents one data record - information about one particular type of entity. Consider Product as a table.

1. Product name and Product Brand as columns.
2. One row in this table will represent one product each.
3. Columns contain the value of each product's properties.

A relational model is a more detailed implementation of the E-R model. E-R models identify real-world objects, finds properties relevant to business requirements and identify a relation between these real-world objects.

The E-R model is more of a design that is built to understand the business requirements and data the business wants to store. A relational model is more detailed in how actually the schema will be implemented in a system. An entity becomes a table, an attribute becomes a column and the relations are established using foreign keys.

Now in the next segment, you will learn about Database Keys.

Additional Reading:

[A discussion on Relational Models](https://opentextbc.ca/dbdesign01/chapter/chapter-7-the-relational-data-model/)

(<https://opentextbc.ca/dbdesign01/chapter/chapter-7-the-relational-data-model/>)

**Database Keys**

## Super key

Super keys are combinations of all possible attributes that can uniquely identify each row of a table.  There can be many such combinations of columns available. All of these combinations are super keys of a table.

The super keys are as follows:

1. <Student ID>
2. <Student ID, Name>

## Candidate key

Candidate keys are the super keys that contain only the necessary attributes to identify each row of a table.

the candidate keys include only <Student ID>.

## Primary key

If a table has many candidate keys, then one of them is chosen by the database designer to uniquely identify each row.

**Please note** in an ERD for a particular entity, the primary key is highlighted by an asterisk (\*) sign in front of it.

## Composite key

This chosen key is the primary key of that table. If this primary key contains more than one attribute, then it is a composite key.

**Please note** in an ERD for a particular entity,  if more than one column names contain an asterisk (\*) sign in front of it, that means the entity has a composite key. This composite key will be the combination of columns mentioned with the asterisk (\*) sign in front of them.

Foreign keys are used to implement relations between tables. One of the tables thus related contains the foreign key that refers to the primary key of the other table. The values present in the foreign key column of one table must be present in the primary key column of the other table. Consider a table A with attributes <A1, A2, A3, A4, A5> and a table B with attributes <B1, B2, B3, B4, B5>. To relate these two tables, we need to have a common column in both the tables. Suppose we keep the A1 column from table A in table B. A1 is the foreign key here.

* Table A: <A1, A2, A3, A4, A5>
* Table B: <B1, B2, B3, B4, B5, A5>

The value in A5 in table B must be present in A1 in table A.

## Additional Reading:

<https://web.csulb.edu/colleges/coe/cecs/dbdesign/dbdesign.php?page=keys.php>

# Subqueries

Subqueries consist of two parts, an inner query and an outer query. These subqueries are further divided into two types:

1. Nested
2. Correlated

## Branch table

|  |  |
| --- | --- |
| branch\_name | varchar(100) |
| **ifsc** | char(12) |
| **branch\_id** | varchar(10) PK |
| **address\_id** | varchar(10) |

## Address table

|  |  |
| --- | --- |
| street | varchar(100) |
| city\_name | varchar(100) |
| state\_name | varchar(100) |
| **address\_id** | varchar(10) PK |
|  |  |

1. Nested

in nested subqueries, first the inner query is executed completely and the result of this inner query is then used to operate on the outer query. You also learnt about the first two steps to create any subquery:

1. Identify the tables that are required to form the required query.
2. Identify the common thing(attribute) between these tables.

       3.   Identify what should be the inner query and what should be the outer query.

Then, using the three steps, you got the solution to the problem discussed earlier which asked for the details of all the branches in Karnataka.

SELECT

branch\_id, branch\_name

FROM

branch

WHERE

address\_id IN (SELECT

address\_id

FROM

address

WHERE

state = ‘Karnataka’)

# 2. Correlated Subqueries

In the previous video, you saw that in correlated subqueries, the outer query is dependent on the inner query. These subqueries are also formed by following the same steps used while creating the nested subqueries. You also saw that the problem discussed in the earlier videos can also be solved using correlated subqueries:

SELECT

branch\_id, branch\_name

FROM

branch

WHERE

EXISTS( SELECT

address\_id

FROM

address

WHERE

state = 'Karnataka'

AND address\_id = branch.address\_id);