**Relational Database Model**

RDBMS stands for Relational Database Management System. RDBMS data is structured in database tables, fields and records. Each RDBMS table consists of database table rows. Each database table row consists of one or more database table fields. RDBMS store the data into collection of tables, which might be related by common fields (database table columns). RDBMS also provide relational operators to manipulate the data stored into the database tables. Most RDBMS use SQL as database query language.

The most popular RDBMS are MS SQL Server, DB2, Oracle and MySQL.

The relational model is an example of record-based model. Record based models are so named because the database is structured in fixed format records of several types. Each table contains records of a particular type. Each record type defines a fixed number of fields, or attributes. The columns of the table correspond to the attributes of the record types. The relational data model is the most widely used data model, and a vast majority of current database systems are based on the relational model.

**Structure of RDBMS**

What is table

The RDBMS database uses tables to store data. A table is a collection of related data entries and contains rows and columns to store data.

A table is the simplest example of data storage in RDBMS. Let's see the example of student table.

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **AGE** | **COURSE** |
| 1 | Ajeet | 24 | B.Tech |
| 2 | aryan | 20 | C.A |
| 3 | Mahesh | 21 | BCA |
| 4 | Ratan | 22 | MCA |
| 5 | Vimal | 26 | BSC |

What is field

Field is a smaller entity of the table which contains specific information about every record in the table. In the above example, the field in the student table consist of id, name, age, course.

What is row or record

A row of a table is also called record. It contains the specific information of each individual entry in the table. It is a horizontal entity in the table. For example: The above table contains 5 records.

Let's see one record/row in the table.

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | Ajeet | 24 | B.Tech |

What is column

A column is a vertical entity in the table which contains all information associated with a specific field in a table. For example: "name" is a column in the above table which contains all information about student's name.

|  |
| --- |
| Ajeet |
| Aryan |
| Mahesh |
| Ratan |
| Vimal |

NULL Values

The NULL value of the table specifies that the field has been left blank during record creation. It is totally different from the value filled with zero or a field that contains space.

**Database Schema**

A database schema is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and how the relations among them are associated. It formulates all the constraints that are to be applied on the data.

A database schema defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams. It’s the database designers who design the schema to help programmers understand the database and make it useful.



A database schema can be divided broadly into two categories −

* **Physical Database Schema** − This schema pertains to the actual storage of data and its form of storage like files, indices, etc. It defines how the data will be stored in a secondary storage.
* **Logical Database Schema** − This schema defines all the logical constraints that need to be applied on the data stored. It defines tables, views, and integrity constraints.

**Database Instance**

Database schema is the skeleton of database. It is designed when the database doesn't exist at all. Once the database is operational, it is very difficult to make any changes to it. A database schema does not contain any data or information. A database instance is a state of operational database with data at any given time. It contains a snapshot of the database. Database instances tend to change with time. A DBMS ensures that its every instance (state) is in a valid state, by diligently following all the validations, constraints, and conditions that the database designers have imposed.

**Schema Diagram**

Employee

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Street | City | Country | Emplyeeid | Birthdate | Gender | did |

Department

|  |  |  |
| --- | --- | --- |
| Name | Departmentid | eeid |

Locations

|  |  |
| --- | --- |
| locid | name |

Project

|  |  |  |
| --- | --- | --- |
| projectid | name | depid |

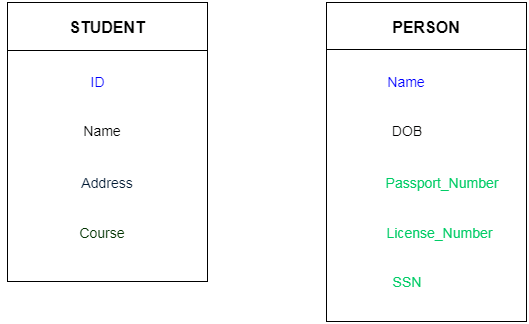
Works on

|  |  |  |
| --- | --- | --- |
| eid | pid | hours |

**Keys**

Keys play an important role in the relational database.It is used to uniquely identify any record or row of data from the table. It is also used to establish and identify relationships between tables.

For example:  In Student table, ID is used as a key because it is unique for each student. In PERSON table, passport\_number, license\_number, SSN are keys since they are unique for each person.



**Types of Keys**

1. **Super key**
2. **Candidate key**
3. **Primary key**
4. **Foreign key**
5. **Composite key**
6. **Surrogate key**
7. **Super key**

Super key is a set of an attribute which can uniquely identify a tuple. Super key is a superset of a candidate key.

Employee

|  |
| --- |
| Employee\_id |
| Employ\_name |
| Employee\_address |
| passport\_no. |
| Liscence\_no. |
| Atmcard no. |

**For example:** In the above EMPLOYEE table, for (EMPLOEE\_ID, EMPLOYEE\_NAME) the name of two employees can be the same, but their EMPLYEE\_ID can't be the same. Hence, this combination can also be a key.

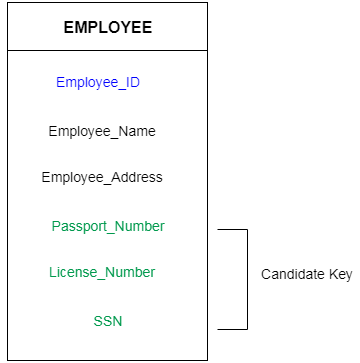
The super key would be EMPLOYEE-ID, (EMPLOYEE\_ID, EMPLOYEE-NAME), etc.

1. **Candidate key**

Each table has only a single primary key. Each relation may have one or more candidate key. One of these candidate key is called Primary Key. Each candidate key qualifies for Primary Key. Therefore candidates for Primary Key is called Candidate Key.

Candidate key can be a single column or combination of more than one column. A minimal super key is called a candidate key.

**For example:** In the EMPLOYEE table, id is best suited for the primary key. Rest of the attributes like SSN, Passport\_Number, and License\_Number, etc. are considered as a candidate key.

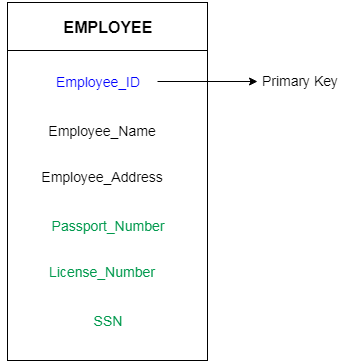


1. **Primary Key**

It is the first key which is used to identify one and only one instance of an entity uniquely. An entity can contain multiple keys as we saw in PERSON table. The key which is most suitable from those lists become a primary key.

In the EMPLOYEE table, ID can be primary key since it is unique for each employee. In the EMPLOYEE table, we can even select License\_Number and Passport\_Number as primary key since they are also unique.

For each entity, selection of the primary key is based on requirement and developers.



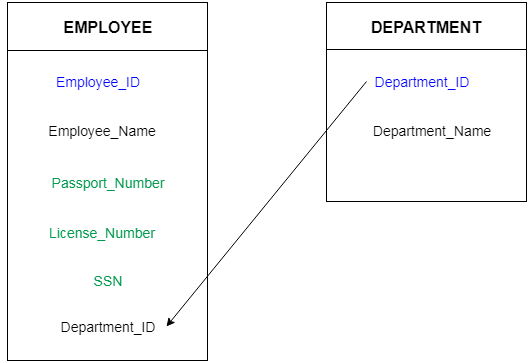
1. **Foreign key**

Foreign keys are the column of the table which is used to point to the primary key of another table.

In a company, every employee works in a specific department, and employee and department are two different entities. So we can't store the information of the department in the employee table. That's why we link these two tables through the primary key of one table.

We add the primary key of the DEPARTMENT table, Department\_Id as a new attribute in the EMPLOYEE table.

Now in the EMPLOYEE table, Department\_Id is the foreign key, and both the tables are related.



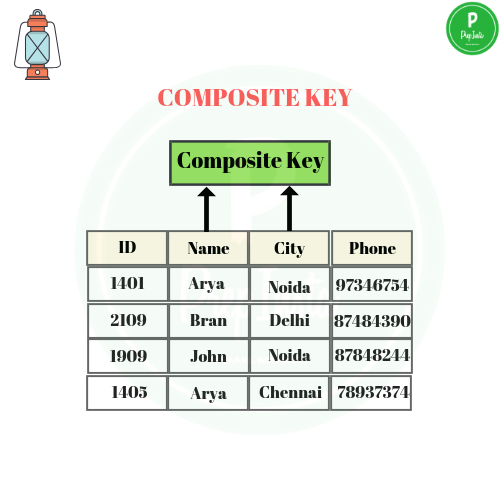
1. **Composite key**

A composite key is a combination of two or more columns in a table that can be used to uniquely identify each row in the table when the columns are combined uniqueness is guaranteed, but when it taken individually it does not guarantee uniqueness.

Sometimes more than one attributes are needed to uniquely identify an entity. A primary key that is made by the combination of more than one attribute is known as a composite key.

In other words we can say that:

Composite key is a key which is the combination of more than one field or column of a given table. It may be a candidate key or primary key.



1. **Surrogate Key**

A surrogate key also called a synthetic primary key, is generated when a new record is inserted into a table automatically by a database that can be declared as the primary key of that table. It is the sequential number outside of the database that is made available to the user and the application or it acts as an object that is present in the database but is not visible to the user or application.

We can say that, in case we do not have a natural primary key in a table, then we need to artificially create one in order to uniquely identify a row in the table, this key is called the surrogate key or synthetic primary key of the table. However, the surrogate key is not always the primary key. Suppose we have multiple objects in a database that are connected to the surrogate key, then we will have a many-to-one association between the primary keys and the surrogate key and the surrogate key cannot be used as the primary key.

**Example:**  
Suppose we have two tables of two different schools having the same column registration\_no, name, and percentage, each table having its own natural primary key, that is registration\_no.

**Table of school A:**

| **registration\_no** | **name** | **percentage** |
| --- | --- | --- |
| 210101 | Harry | 90 |
| 210102 | Maxwell | 65 |
| 210103 | Lee | 87 |
| 210104 | Chris | 76 |

**Table of school B:**

| **registration\_no** | **name** | **percentage** |
| --- | --- | --- |
| CS107 | Taylor | 49 |
| CS108 | Simon | 86 |
| CS109 | Sam | 96 |
| CS110 | Andy | 58 |

Now, suppose we want to merge the details of both the schools in a single table.  
Resulting table will be:

| **surr\_no** | **registration\_no** | **name** | **percentage** |
| --- | --- | --- | --- |
| 1 | 210101 | Harry | 90 |
| 2 | 210102 | Maxwell | 65 |
| 3 | 210103 | Lee | 87 |
| 4 | 210104 | Chris | 76 |
| 5 | CS107 | Taylor | 49 |
| 6 | CS108 | Simon | 86 |
| 7 | CS109 | Sam | 96 |
| 8 | CS110 | Andy | 58 |

As we can observe the above table and see that registration\_no cannot be the primary key of the table as it does not match with all the records of the table though it is holding all unique values of the table . Now , in this case, we have to artificially create one primary key for this table. We can do this by adding a column surr\_no in the table that contains anonymous integers and has no direct relation with other columns . This additional column of surr\_no is the surrogate key of the table.

**Relational Algebra**

Relational algebra is a procedural query language, which takes instances of relations as input and yields instances of relations as output. It uses operators to perform queries. An operator can be either unary or binary. They accept relations as their input and yield relations as their output. Relational algebra is performed recursively on a relation and intermediate results are also considered relations.

There are two types of relational operators:

1. Basic operators

* Projection
* Select
* Union
* Set Different
* Cross Product
* Rename

1. Derived operators

* Join
* Intersect
* Division

**Projection**

This operation shows the list of those attributes that we wish to appear in the result. Rest of the attributes are eliminated from the table. It is denoted by ∏

Notation: ∏ A1, A2, An (r)

**Where**

**A1, A2, A3**is used as an attribute name of relation **r**.

.

Student

|  |  |  |  |
| --- | --- | --- | --- |
| Roll | Name | Address | Age |
| 1 | Anjit | lamjung | 22 |
| 2 | Ayush | jhapa | 21 |
| 3 | Krishna | kathmandu | 23 |
| 4 | Ayush | pokhara | 20 |

Retreive the roll number from table student.

∏ Roll (Student)

|  |
| --- |
| Roll |
| 1 |
| 2 |
| 3 |
| 4 |

Retreive the name from table student.

∏ Name (Student)

|  |
| --- |
| Name |
| Anjit |
| Ayush |
| Krishna |

Projection only works on columns and by default it retrieves distinct data from the entire database.

**Select**

The select operation selects tuples that satisfy a given predicate. It is denoted by sigma (σ).

Notation:  σ p(r)

Where:

σ is used for selection prediction  
r is used for relation  
p is used as a propositional logic formula which may use connectors like: AND OR and NOT. These relational can use as relational operators like =, ≠, ≥, <, >, ≤.

|  |  |  |  |
| --- | --- | --- | --- |
| Roll | Name | Address | Age |
| 1 | Anjit | lamjung | 22 |
| 2 | Ayush | jhapa | 21 |
| 3 | Krishna | kathmandu | 23 |
| 4 | Ayush | pokhara | 20 |

Retrieve the name of the student whose roll number is 3.

∏ Name (σ Roll = ‘ 3 ‘ (Student))

|  |
| --- |
| Name |
| Krishna |

Select works only on rows.

**Cartesian product**

The Cartesian product is used to combine each row in one table with each row in the other table. It is also known as a cross product.It is denoted by X.

Notation: R1 X R2

R1

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **C** |
| **1** | **2** | **3** |
| **4** | **5** | **6** |
| **7** | **8** | **9** |

R2

|  |  |
| --- | --- |
| **C** | **D** |
| **3** | **4** |
| **5** | **9** |

R1 X R2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **C** | **D** |
| **1** | **2** | **3** | **3** | **4** |
| **1** | **2** | **3** | **5** | **9** |
| **4** | **5** | **6** | **3** | **4** |
| **4** | **5** | **6** | **5** | **9** |
| **7** | **8** | **9** | **3** | **4** |
| **7** | **8** | **9** | **5** | **9** |

**Rename**

The rename operation is used to rename the output relation. It is denoted by rho (ρ).

Example: We can use the rename operator to rename STUDENT relation to STUDENT1.

ρ (STUDENT1, STUDENT)

**Union**

Suppose there are two tuples R and S. The union operation contains all the tuples that are either in R or S or both in R & S.It eliminates the duplicate tuples. It is denoted by ∪.

Notation: R1 ∪ R2

A union operation must hold the following condition:

* R1 and R2 must have the attribute of the same number.
* Duplicate tuples are eliminated automatically.

Student

|  |  |
| --- | --- |
| Std - ID | Name |
| 1 | Anjit |
| 2 | Pabitra |
| 3 | Rabina |
| 4 | Saugat |

Employee

|  |  |
| --- | --- |
| Emp - ID | Name |
| 2 | Pabitra |
| 7 | Ayush |
| 8 | Sonam |

( Student ) U ( Employee )

|  |  |
| --- | --- |
| Std - ID | Name |
| 1 | Anjit |
| 2 | Pabitra |
| 3 | Rabina |
| 4 | Saugat |
| 7 | Ayush |
| 8 | Sonam |

Reterive the name of the student who can be belongs to employee table also.

∏ NAME (Student) ∪ ∏ NAME (Employee)

|  |
| --- |
| Name |
| Anjit |
| Pabitra |
| Rabina |
| Saugat |
| Ayush |
| Sonam |

**Set Difference**

Suppose there are two tuples R and S. The set intersection operation contains all tuples that are in both R & S. It is denoted by intersection ∩.

Notation: R1 - R2

A set difference operation must hold the following condition:

* R1 and R2 must have the attribute of the same number.
* Duplicate tuples are eliminated automatically.

Student

|  |  |
| --- | --- |
| Std - ID | Name |
| 1 | Anjit |
| 2 | Pabitra |
| 3 | Rabina |
| 4 | Saugat |

Employee

|  |  |
| --- | --- |
| Emp - ID | Name |
| 2 | Pabitra |
| 7 | Ayush |
| 8 | Sonam |

( Student ) - ( Employee )

|  |  |
| --- | --- |
| Std - ID | Name |
| 1 | Anjit |
| 3 | Rabina |
| 4 | Saugat |

Retrieve the name of the student who is not employee.

∏ NAME (Student) - ∏ NAME (Employee)

|  |
| --- |
| Name |
| Anjit |
| Rabina |
| Saugat |