

# Introduction to Relational Databases

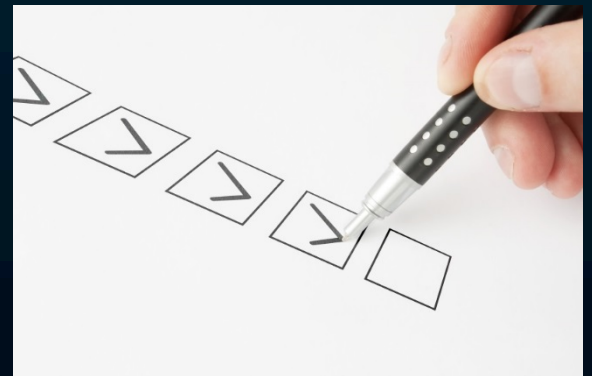
---

## Introduction to Database Design

## 2 Enabling Objectives

After completing this chapter, in the next 30 minutes you will be able to :

- Define with an example at least 1 data model
- List and define at least 1 Database Normalization with an example
- List different Data Types used in a table creation



## Key Topics

- The need of RDBMS
- Types of Data Models
- Different Levels of Database Normalization
- The purpose of SQL

# Database and Data Models

## Need of Database

- To manage large chunks of data
- Accuracy
- Ease of updating data
- Security of data

## What is a Database

- A database is a collection of logically related data
- It is used to search data to answer queries
- A database may be designed for batch processing, real time processing, or online processing

- Is a set of software or programs
  - Enables storing, modifying, and extracting information from a database
- Data can be accessed by using
  - query and reporting tools
  - application programs

# The Relational Database



## The Relational Database

Relational Databases are primarily defined by a collection of tables. The tables (more formally called relations) are made up of:

- A list of attributes (fields, or columns)
- A set of records (tuples, or rows) containing particular data for some or all of the attributes

## Relationships

These tables have relationships among them represented by shared attributes.

Consider the following example of two tables in a database about dogs.

## Relationship Example

Dog: {dog\_id, name, sex, weight, height, color, d\_o\_b, breed\_id}

Breed: {breed\_id, name, temperament, coat}

Using breed\_id, we can find the likely temperament of a particular dog or get a list of all of the dogs we have of a given breed.

# Normalization

- Normalization is a process for determining how to organize our data.
  - Normalizing reduces various troublesome data anomalies.
  - There are six normal forms that have universally accepted definitions. We'll only look at the first four.
  - These focus on the elimination of data redundancy.
  - The other two are quite rare and hard to even think of reasonable examples for.

## Functional Dependencies and Key Fields

In order to go any further we need to understand a couple of central topics.

- Functional Dependencies
- Key Fields

Consider the following fields in an Employee table

{EID, EName, Email, Phone, WorkAddress,  
HomeAddress, TaxJurisdiction, ManagerID,  
DependentNames}

## Functional Dependencies

A functional dependency is a relationship between two attributes where  $y$  is functionally dependant on  $x$  if by knowing  $x$  I can find  $y$ , but not necessarily the other way around. In our table there are the following FDs:

- $EID \rightarrow \{<\text{The entire table}>\}$
- $Email \rightarrow \{<\text{The entire table}>\}$
- $WorkAddress \rightarrow \{TaxJurisdiction\}$

## Keys

- A superkey is a field or set of fields on which all other fields in the table are functionally dependant. (That is, they can uniquely identify rows in the table.)
  - EID and Email are superkeys
  - {EID, EName} is also a superkey, because it is a superset of EID
- A candidate key is a superkey that has no proper subset which is also a superkey
  - Thus EID and Email, but not {EID, EName} are candidate keys.



## Primary Keys

- The primary key is the designated identifying field of a table is chosen by the developer from among the candidate keys.
- Some guidelines:
  - Choose attributes under your organization's control – values such as passport numbers or government ID numbers may change outside of your control
  - Choose meaningless fields over ones with a purpose.
    - EID is a better choice than email address, as, while unique, an email address may change as well.
  - Prefer smaller attribute sets to larger ones

# Normal Forms

## First Normal Form (1NF)

- 1NF states that a relational database table can have no "repeating groups".
- This applies to two things:
  - Collections of items in one field
  - Composite items in a field

Collections are simply any field intended to hold more than one entry.

Composite fields are dependent on the context of the data. Fields should not have data that's intended to be broken down to be used.

## 1NF Example

Consider our Employee table:

{EID, EName, Email, Phone, WorkAddress, HomeAddress, TaxJurisdiction, ManagerID, DependentNames}

- Here, WorkAddress and HomeAddress are likely composite fields and should be broken into multiple fields each.
- Dependents here is intended as a list of the employee's dependents (not a count) and is thus a collection.

## Breaking out a collection

- Collections are resolved by breaking them out into another table (likely named Dependent):
  - {EID, DependentName}
  - Thus, if we want to represent a given employee's 5 dependents, we create five rows in this new table with the same EID and each, different dependent name.

## 1NF is required

For a database to be strictly; a relational database, 1NF is required. Relational databases do not have collections or composite objects in them.

In practice, most commercially used relational databases do support list and object types for fields due to their usefulness.

## Second Normal Form (2<sup>NF</sup>)

- A table is in 2NF if
  - it is in 1NF and
  - all non-key fields are functionally dependent on the entire Primary Key. (More simply, no partial dependencies.) Consider this table:

Project Code	Project Name	Project Manager	Project Budget	Employee No.	Employee Name	Department No.	Department Name	Hourly Rate
PC010	Reservation System	Mr. Ajay	120500	S100	Mohan	D03	Database	21.00
PC010	Reservation System	Mr. Ajay	120500	S101	Vipul	D02	Testing	16.50
PC010	Reservation System	Mr. Ajay	120500	S102	Riyaz	D01	IT	22.00
PC011	HR System	Mrs. Charu	500500	S103	Pavan	D03	Database	18.50
PC011	HR System	Mrs. Charu	500500	S104	Jitendra	D02	Testing	17.00
PC011	HR System	Mrs. Charu	500500	S315	Pooja	D01	IT	23.50
PC012	Attendance System	Mr. Rajesh	710700	S137	Rahul	D03	Database	21.50
PC012	Attendance System	Mr. Rajesh	710700	S218	Avneesh	D02	Testing	15.50
PC012	Attendance System	Mr. Rajesh	710700	S109	Vikas	D01	IT	20.50

- Let's look at the functional dependencies:
  - Project name, manager, and budget are dependent on ProjectCode
  - EmployeeName, DepartmentNo, and DepartmentName are dependent on EmployeeNo
  - DepartmentName is dependent on DepartmentNo
  - HourlyRate is dependent on EmployeeNo and ProjectCode



## 2NF cont.

- Thus, the only candidate key (and thus the primary key) is {ProjectCode, EmployeeNo}. This is called a composite key
- Only HourlyRate is dependent upon the entire primary key
- To fix this, we put ProjectCode and its dependencies and EmployeeNo and its dependencies into separate tables, leaving only Hourly Rate in the original.

# Second Normal Form (2NF)

Example :

Primary Key

Employee No.	Employee Name	Department No.	Department Name
S100	Mohan	D03	Database
S101	Vipul	D02	Testing
S102	Riyaz	D01	IT
S103	Pavan	D03	Database
S104	Jitendra	D02	Testing
S315	Pooja	D01	IT
S137	Rahul	D03	Database
S218	Avneesh	D02	Testing
S109	Vikas	D01	IT

Composite Key

Project Code	Employee No.	Hourly Rate
PC010	S100	21.00
PC010	S101	16.50
PC010	S102	22.00
PC011	S103	18.50
PC011	S104	17.00
PC011	S315	23.50
PC012	S137	21.50
PC012	S218	15.50
PC012	S109	20.50

Primary Key

Project Code	Project Name	Project Manager	Project Budget
PC010	Reservation System	Mr. Ajay	120500
PC011	HR System	Mrs. Charu	500500
PC012	Attendance System	Mr. Rajesh	710700

## Third Normal Form (3NF)

- A database table is said to be in 3NF if
  - it is in 2NF and
  - No non-key fields are functionally dependent upon any other non-key fields
  - That is, we have no transitive dependencies
- The process of converting a table into 3NF is to remove the transitive dependencies into a new table along with their (now key) field.
- In our example, DepartmentName is functionally dependent upon DepartmentNo, a non-key field.

# Third Normal Form (3NF)

## Example :

Primary Key			
Project Code	Project Name	Project Manager	Project Budget
PC010	Reservation System	Mr. Ajay	120500
PC011	HR System	Mrs. Charu	500500
PC012	Attendance System	Mr. Rajesh	710700

Composite Key		
Project Code	Employee No.	Hourly Rate
PC010	S100	21.00
PC010	S101	16.50
PC010	S102	22.00
PC011	S103	18.50
PC011	S104	17.00
PC011	S315	23.50
PC012	S137	21.50
PC012	S218	15.50
PC012	S109	20.50

Primary Key		
Employee No.	Employee Name	Department No.
S100	Mohan	D03
S101	Vipul	D02
S102	Riyaz	D01
S103	Pavan	D03
S104	Jitendra	D02
S315	Pooja	D01
S137	Rahul	D03
S218	Avneesh	D02
S109	Vikas	D01

Primary Key		FK_Relationship	
Department No.	Department Name	Department No.	Department Name
D01	IT	D01	IT
D02	Testing	D02	Testing
D03	Database	D03	Database

## Boyce Codd Normal Form (BCNF)

- BCNF is one of the more commonly misunderstood forms on the Internet.
- A table is in BCNF iff
  - It is in 3NF and
  - No component of the primary key is dependent on a non-key field
- This can only occur if there are multiple, composite candidate keys.
- Consider this in the abstract:
- We have a table {A, B, C, D}
- Dependencies are
  - $\{A,B\} \rightarrow C, D$
  - $C \rightarrow B$
- $\{A,B\}$  is chosen as primary key.
- We are in 3NF, as both C and D are directly dependent on  $\{A, B\}$
- We are not in BCNF, as B is dependent on C

## Resolving BCNF

- What we missed in our analysis is that if  $C \rightarrow B$ , and  $\{A, B\} \rightarrow D$  then  $\{A, C\} \rightarrow B, D$  as well.
- If we were to have chosen  $\{A, C\}$  as our primary key, we'd now fail to be in 2NF.
- Resolving this to tables  $\{A, C, D\}$  and  $\{C, B\}$  leaves us in BCNF.
- Note that the FDs reflect our data. Our tables conform to our data. And so if we know  $A$  and  $B$ , we'll still have a single  $C$  and  $D$  that correspond to both of them, thus  $\{A, B\} \rightarrow C, D$  will still hold.

**You have successfully completed the  
session on Introduction to database**

