

Today's Lecture

- Recap - Relational Model Terminology
- Functional dependency
- Normalisation
- Indexing

Recap Quiz - Relational Model Terminology

You are working as a database designer for a retail company. You need to design a database for managing product details. The database table called "**Products**" has the following fields:

ProductID (Unique identifier for each product)

ProductName (Name of the product)

Category (Type of product such as Electronics, Clothing, etc.)

Price (Price of the product)

StockQty (Quantity of the product in stock)

In the relational model, the table "**Products**" is considered as a:

- a) Attribute
- b) Tuple
- c) Relation
- d) Domain

Recap Quiz - Relational Model Terminology

You are working as a database designer for a retail company. You need to design a database for managing product details. The database table called "**Products**" has the following fields:

ProductID (Unique identifier for each product)

ProductName (Name of the product)

Category (Type of product such as Electronics, Clothing, etc.)

Price (Price of the product)

StockQty (Quantity of the product in stock)

The fields **ProductID**, **ProductName**, **Category**, **Price**, and **StockQty** represent the:

- a) Tuples
- b) Domains
- c) Attributes
- d) Degrees

Recap Quiz - Relational Model Terminology

You are working as a database designer for a retail company. You need to design a database for managing product details. The database table called "**Products**" has the following fields:

ProductID (Unique identifier for each product)

ProductName (Name of the product)

Category (Type of product such as Electronics, Clothing, etc.)

Price (Price of the product)

StockQty (Quantity of the product in stock)

A valid range of values for **Price** (e.g., from 1 to 10,000) or for **Category** (e.g., Electronics, Clothing) is referred to as:

- a) Domain
- b) Tuple
- c) Cardinality
- d) Relation

Recap Quiz - Relational Model Terminology

You are working as a database designer for a retail company. You need to design a database for managing product details. The database table called "**Products**" has the following fields:

ProductID (Unique identifier for each product)

ProductName (Name of the product)

Category (Type of product such as Electronics, Clothing, etc.)

Price (Price of the product)

StockQty (Quantity of the product in stock)

Each row in the "**Products**" table (like ProductID = 101, ProductName = "Laptop", etc.) is called a:

- a) Relation
- b) Tuple
- c) Domain
- d) Degree

Recap Quiz - Relational Model Terminology

You are working as a database designer for a retail company. You need to design a database for managing product details. The database table called "**Products**" has the following fields:

ProductID (Unique identifier for each product)

ProductName (Name of the product)

Category (Type of product such as Electronics, Clothing, etc.)

Price (Price of the product)

StockQty (Quantity of the product in stock)

The total number of columns in the "**Products**" table (e.g., 5 columns) is known as:

- a) Attribute
- b) Domain
- c) Degree
- d) Cardinality

Recap Quiz - Relational Model Terminology

You are working as a database designer for a retail company. You need to design a database for managing product details. The database table called "**Products**" has the following fields:

ProductID (Unique identifier for each product)

ProductName (Name of the product)

Category (Type of product such as Electronics, Clothing, etc.)

Price (Price of the product)

StockQty (Quantity of the product in stock)

The total number of rows in the "**Products**" table (e.g., 50 rows) is known as:

- a) Degree
- b) Attribute
- c) Domain
- d) Cardinality

Recap Quiz - Relational Model Terminology

You are working as a database designer for a retail company. You need to design a database for managing product details. The database table called "**Products**" has the following fields:

ProductID (Unique identifier for each product)

ProductName (Name of the product)

Category (Type of product such as Electronics, Clothing, etc.)

Price (Price of the product)

StockQty (Quantity of the product in stock)

A collection of related tables (e.g., a "Products" table, "Customers" table, and "Orders" table) is called:

- a) Attribute
- b) Relation
- c) Relational Database
- d) Cardinality

Recap - Relational Model Terminology

- **Relation:** a table with columns and rows
 - This concept only applies to the logical structure of the database: at the external and conceptual levels
- **Attribute:** a named column of a relation
- **Domain:** the set of allowed values for one or more attributes.
- **Tuple:** a row of a relation.

Relational Model Terminology

- **Degree:** the number of attributes in a relation.
- **Cardinality:** the number of tuples in a relation.
- **Relational Database:** collection of *normalised* relations with distinct relation names.

Normalisation

What is Normalisation?

- **Normalisation** is a technique for producing a set of *suitable relations that support the data requirements* of an enterprise.

Employee

Man#	Name	Birthdate	JobHistory				Children	
123	Michael Swart	Nov. 22	JobDate	Title	SalaryHistory		Childname	Birthyear
			2000	Lackey	SalaryDate	Salary	Mini-me 1	2000
					2000	\$1,000,000	Mini-me 2	2002
					2001	\$2,000,000		
			2002	Senior Lackey	SalaryDate	Salary		
					2002	\$3,000,000		
					2003	\$4,000,000		

What is Normalisation?

- **Normalisation** is a technique for producing a set of *suitable relations that support the data requirements* of an enterprise.

Employee'

Man#	Name	Birthdate
123	Michael Swart	Nov. 22

JobHistory'

<u>Man#</u>	<u>JobDate</u>	Title
123	2000	Lackey
123	2002	Senior Lackey

Children'

Man#	ChildName	BirthYear
123	Mini-me 1	2000
123	Mini-me 2	2002

SalaryHistory'

<u>Man#</u>	<u>JobDate</u>	<u>SalaryDate</u>	Salary
123	2000	2000	\$1,000,000
123	2000	2001	\$2,000,000
123	2002	2002	\$3,000,000
123	2003	2003	\$4,000,000

The Purpose of Normalisation

- The main aim of a relational database design is to group attributes into relations to ***minimise data redundancy***.
 - Normalisation, helps to achieve this.
- Benefits of using a database that has a *suitable* set of relations is that the database will be:
 - easier for the user to access and maintain the data.
 - take up minimal storage space on the computer.

Problems of not properly normalised tables

- There are two main problems associated with a relation that is not appropriately normalised:
 - Data Redundancy
 - Update Anomalies for:
 - Insertions (adding new),
 - Deletions, and
 - Updates (modifications).

Data Redundancy and Update Anomalies

- What will be the primary key for this table?
- Can you spot potential problems with this table design?

CampusStaff Table

staffNo	staffName	position	salary	campusID	campusAddress
SW101	John Smith	Lecturer	30000	C002	Swansea IQ, SA1 8PH
SW102	Diane Lee	PVC	70000	C001	Carmarthen Campus, Carmarthen, SA31 3EP
SW103	Ann Beech	Lecturer	18000	C002	Swansea IQ, SA1 8PH
SW105	David Ford	Snr Lecturer	41000	C001	Carmarthen Campus, Carmarthen, SA31 3EP
SW106	Mary Howe	Dean, Humanities	55000	C003	Lampeter Campus Ceredigion SA48 7ED
SW107	Susan Brand	Receptionist	18000	C001	Swansea IQ, SA1 8PH

Introduction to Functional Dependencies

- What are Functional Dependencies (FD)?

Example Dataset 1: Student Information

Student_ID	Name	Department	HOD
101	Alice	Computer Science	Dr. John
102	Bob	Mechanical Eng.	Dr. Smith
103	Charlie	Computer Science	Dr. John

- Identify the attribute(s) that uniquely determine student's name and department?
- Identify the attribute(s) that uniquely determine the head of department?

Introduction to Functional Dependencies

- What are Functional Dependencies (FD)?

Example Dataset 1: Student Information

Student_ID	Name	Department	HOD
101	Alice	Computer Science	Dr. John
102	Bob	Mechanical Eng.	Dr. Smith
103	Charlie	Computer Science	Dr. John

- Student ID uniquely determines the student's name and department
- Department determines the Head of Department

Introduction to Functional Dependencies

- What are Functional Dependencies (FD)?

Example Dataset 1: Student Information

Student_ID	Name	Department	HOD
101	Alice	Computer Science	Dr. John
102	Bob	Mechanical Eng.	Dr. Smith
103	Charlie	Computer Science	Dr. John

- Student ID uniquely determines the student's name and department

Student_ID → Name, Department

Determinant

Dependents

- Department determines the Head of Department

Department → HOD

Determinant

Dependent

The arrowline is
read as
'determines'

Introduction to Functional Dependencies

- What are Functional Dependencies (FD)?

A functional dependency (FD) means that one attribute uniquely determines another.

Denoted as $\mathbf{X} \rightarrow \mathbf{Y}$, where $\mathbf{X}$ is a determinant, and $\mathbf{Y}$ is the dependent attribute.

Activity: Identifying Functional Dependencies

With the following dataset identify the functional dependencies.

Employee_ID	Employee_Name	Department	Manager
201	John	HR	Mr. Parker
202	Sarah	IT	Mr. Stevens
203	Alice	HR	Mr. Parker

Activity: Identifying Functional Dependencies

With the following dataset identify the functional dependencies.

Employee_ID	Employee_Name	Department	Manager
201	John	HR	Mr. Parker
202	Sarah	IT	Mr. Stevens
203	Alice	HR	Mr. Parker

Employee_ID → Employee_Name, Department

Department → Manager

Normalization

- **Normalization** is the process of organizing data to minimize redundancy and improve data integrity.

Normalization - First Normal Form (1NF)

Example Dataset 2: Non-1NF

Student_ID	Name	Courses
101	Alice	CSE101, MTH102
102	Bob	PHY101
103	Charlie	CSE101, ENG101, PHY101

Normalization - First Normal Form (1NF)

Example Dataset 2: Non-1NF

Student_ID	Name	Courses
101	Alice	CSE101, MTH102
102	Bob	PHY101
103	Charlie	CSE101, ENG101, PHY101

1NF requires that all entries in a column are atomic (indivisible), and each row must be unique.

Normalization - First Normal Form (1NF)

Example Dataset 2: Non-1NF

Student_ID	Name	Courses
101	Alice	CSE101, MTH102
102	Bob	PHY101
103	Charlie	CSE101, ENG101, PHY101

contains
atomic values

1NF requires that all entries in a column are atomic (indivisible), and each row must be unique.

Convert to 1NF:

Student_ID	Name	Course
101	Alice	CSE101
101	Alice	MTH102
102	Bob	PHY101
103	Charlie	CSE101
103	Charlie	ENG101
103	Charlie	PHY101

Activity - First Normal Form (1NF)

Normalize the table below to 1NF

Order_ID	Customer	Products
001	Jane	TV, Laptop
002	John	Smartphone
003	Alice	TV, Camera, Headphones

Activity - First Normal Form (1NF)

1NF requires that all entries in a column are atomic (indivisible), and each row must be unique.

Order_ID	Customer	Products
001	Jane	TV, Laptop
002	John	Smartphone
003	Alice	TV, Camera, Headphones

Expected Result (1NF):

Order_ID	Customer	Product
001	Jane	TV
001	Jane	Laptop
002	John	Smartphone
003	Alice	TV
003	Alice	Camera
003	Alice	Headphones

Normalization - Second Normal Form (2NF)

- The table must be in 1NF.
- There should be no **partial dependency**

StudentID	StudentName	CourseID	CourseName	InstructorName	Grade
101	Alice Johnson	C101	Database Systems	Dr. Smith	A
101	Alice Johnson	C102	Data Structures	Dr. Johnson	B
102	Bob Brown	C101	Database Systems	Dr. Smith	B

Identify the composite key from the above table?

Normalization - Second Normal Form (2NF)

- The table must be in 1NF.
- There should be no **partial dependency**

StudentID	StudentName	CourseID	CourseName	InstructorName	Grade
101	Alice Johnson	C101	Database Systems	Dr. Smith	A
101	Alice Johnson	C102	Data Structures	Dr. Johnson	B
102	Bob Brown	C101	Database Systems	Dr. Smith	B

Identify the composite key from the above table?

composite key is **{StudentID, CourseID}**

Which FD is enough to uniquely determine studentname?

a) {StudentID, CourseID} -> {StudentName}

b) {StudentID} -> {StudentName}

Normalization - Second Normal Form (2NF)

- The table must be in 1NF.
- There should be no **partial dependency**

StudentID	StudentName	CourseID	CourseName	InstructorName	Grade
101	Alice Johnson	C101	Database Systems	Dr. Smith	A
101	Alice Johnson	C102	Data Structures	Dr. Johnson	B
102	Bob Brown	C101	Database Systems	Dr. Smith	B

Identify the composite key from the above table?

composite key is {**StudentID, CourseID**}

Which FD is enough to uniquely determine studentname?

a) {StudentID, CourseID} -> {StudentName}

b) {StudentID} -> {StudentName}

Normalization - Second Normal Form (2NF)

- The table must be in 1NF.
- There should be no **partial dependency**

StudentID	StudentName	CourseID	CourseName	InstructorName	Grade
101	Alice Johnson	C101	Database Systems	Dr. Smith	A
101	Alice Johnson	C102	Data Structures	Dr. Johnson	B
102	Bob Brown	C101	Database Systems	Dr. Smith	B

Identify the composite key from the above table?

composite key is {**StudentID, CourseID**}

Which FD is enough to uniquely determine CourseName?

a) {StudentID, CourseID} -> {CourseName}

b) {CourseID} -> {CourseName}

c) {CourseID} -> {InstructorName}

Normalization - Second Normal Form (2NF)

- The table must be in 1NF.
- There should be no **partial dependency**

StudentID	StudentName	CourseID	CourseName	InstructorName	Grade
101	Alice Johnson	C101	Database Systems	Dr. Smith	A
101	Alice Johnson	C102	Data Structures	Dr. Johnson	B
102	Bob Brown	C101	Database Systems	Dr. Smith	B

Can you create two tables to eliminate partial dependencies?

Normalization - Second Normal Form (2NF)

- The table must be in 1NF.
- There should be no **partial dependency**

StudentID	StudentName	CourseID	CourseName	InstructorName	Grade
101	Alice Johnson	C101	Database Systems	Dr. Smith	A
101	Alice Johnson	C102	Data Structures	Dr. Johnson	B
102	Bob Brown	C101	Database Systems	Dr. Smith	B

Students Table (after removing partial dependencies):

StudentID	StudentName
101	Alice Johnson
102	Bob Brown

Courses Table (after removing partial dependencies):

CourseID	CourseName	InstructorName
C101	Database Systems	Dr. Smith
C102	Data Structures	Dr. Johnson

Where to
place
'Grade'?

Normalization - Second Normal Form (2NF)

- The table must be in 1NF.
- There should be no **partial dependency**

Students Table (after removing partial dependencies):

StudentID	StudentName
101	Alice Johnson
102	Bob Brown

Courses Table (after removing partial dependencies):

CourseID	CourseName	InstructorName
C101	Database Systems	Dr. Smith
C102	Data Structures	Dr. Johnson

Enrollment Table (no partial dependencies, stores relationships and grades):

StudentID	CourseID	Grade
101	C101	A
101	C102	B
102	C101	B

Activity - Second Normal Form (2NF)

- The table must be in 1NF.
- There should be no **partial dependency**

EmployeeID	EmployeeName	DeptID	DeptName	ProjectID	ProjectName	HoursWorked
E001	Alice Johnson	D01	HR	P101	Payroll System	30
E001	Alice Johnson	D01	HR	P102	Recruitment App	25
E002	Bob Brown	D02	IT	P101	Payroll System	40
E003	Charlie Davis	D01	HR	P103	Employee Portal	20
E004	Daisy White	D03	Finance	P104	Budgeting App	35

Step 1: Identify the Composite Key

Step 2: Identify Partial Dependencies

Step 3: Break the Table into 2NF Tables

Activity - Second Normal Form (2NF)

- The table must be in 1NF.
- There should be no **partial dependency**

Employee Table (2NF)

EmployeeID	EmployeeName	DeptID
E001	Alice Johnson	D01
E002	Bob Brown	D02
E003	Charlie Davis	D01
E004	Daisy White	D03

Department Table (2NF)

DeptID	DeptName
D01	HR
D02	IT
D03	Finance

Activity - Second Normal Form (2NF)

- The table must be in 1NF.
- There should be no **partial dependency**

Project Table (2NF)

ProjectID	ProjectName
P101	Payroll System
P102	Recruitment App
P103	Employee Portal
P104	Budgeting App

Assignment Table (2NF)

This table contains the relationships between employees and the projects they are working on, including the HoursWorked.

EmployeeID	ProjectID	HoursWorked
E001	P101	30
E001	P102	25
E002	P101	40
E003	P103	20
E004	P104	35

Normalization - Third Normal Form (3NF)

- The table must be in 2NF.
- There should be no **transitive dependency**

StudentID	StudentName	CourseID	CourseName	InstructorName	InstructorPhone
101	Alice Johnson	C101	Math	Dr. Smith	123-456-7890
102	Bob Brown	C102	History	Dr. Adams	987-654-3210
101	Alice Johnson	C102	History	Dr. Adams	987-654-3210

StudentID → **StudentName**: The student's name depends on the student's ID.

CourseID → **CourseName**: The course name depends on the course's ID.

CourseID → **InstructorName**: The instructor's name depends on the course.

InstructorName → **InstructorPhone**: The instructor's phone number depends on the instructor.

Normalization - Third Normal Form (3NF)

- The table must be in 2NF.
- There should be no **transitive dependency**

StudentID → **StudentName**
CourseID → **CourseName**
CourseID → **InstructorName**
InstructorName → **InstructorPhone**

New Tables (in 3NF):

1. Student Table (No changes):

StudentID	StudentName
101	Alice Johnson
102	Bob Brown

2. Course Table (No transitive dependency):

CourseID	CourseName	InstructorName
C101	Math	Dr. Smith
C102	History	Dr. Adams

3. Instructor Table (Instructor phone moved to a separate table):

InstructorName	InstructorPhone
Dr. Smith	123-456-7890
Dr. Adams	987-654-3210

INDEXING IN DATABASES

Purpose of Indexing

Indexing is a **database optimization technique** used to speed up data retrieval. Think of an index as a "lookup guide" – like an index in a book.

Key benefits:

- Speeds up **SELECT** queries
- Reduces disk I/O
- Enhances performance when filtering or joining large tables

INDEXING IN DATABASES

Concept	Description
Primary Key	Uniquely identifies each row in a table. Automatically creates a unique index.
Foreign Key	Establishes a relationship between two tables.
Index	A structure (usually B-tree) used by the DBMS to quickly locate data.

INDEXING IN DATABASES

Step 1: Open MySQL Console

1. From the Uniform Server control panel, click "**MySQL Console**".

2. When prompted:

- Enter username: root
- Press Enter (leave password blank by default)

INDEXING IN DATABASES

Create a New Database

```
CREATE DATABASE BookStore;  
USE BookStore;
```

INDEXING IN DATABASES

Create Tables with Keys

-- Create Authors table

```
CREATE TABLE Authors (  
    AuthorID INT PRIMARY KEY,  
    AuthorName VARCHAR(100),  
    Country VARCHAR(50)  
);
```

INDEXING IN DATABASES

Create Tables with Keys

-- Create Books table with foreign key

```
CREATE TABLE Books (  
    BookID INT PRIMARY KEY,  
    Title VARCHAR(100),  
    Genre VARCHAR(50),  
    AuthorID INT,  
    FOREIGN KEY (AuthorID) REFERENCES  
    Authors(AuthorID)  
);
```

INDEXING IN DATABASES

Insert Sample Records

INSERT INTO Authors VALUES

(1, 'George Orwell', 'UK'),

(2, 'Haruki Murakami', 'Japan'),

(3, 'Chinua Achebe', 'Nigeria'),

(4, 'J.K. Rowling', 'UK'),

(5, 'Gabriel Garcia Marquez', 'Colombia');

INDEXING IN DATABASES

Insert Sample Records

INSERT INTO Books VALUES

(101, '1984', 'Dystopia', 1),

(102, 'Kafka on the Shore', 'Fiction', 2),

(103, 'Things Fall Apart', 'Historical', 3),

(104, 'Harry Potter', 'Fantasy', 4),

(105, 'One Hundred Years of Solitude', 'Magical Realism', 5);

INDEXING IN DATABASES

Run a Query Without Index

```
SELECT * FROM Books WHERE Genre = 'Fiction';
```

INDEXING IN DATABASES

Check Execution Plan

```
EXPLAIN SELECT * FROM Books WHERE Genre = 'Fiction';
```

INDEXING IN DATABASES

Add an Index on Genre

```
CREATE INDEX idx_genre ON Books(Genre);
```

INDEXING IN DATABASES

Re-run the Query

```
SELECT * FROM Books WHERE Genre = 'Fiction';
```

INDEXING IN DATABASES

Check Execution Plan

```
EXPLAIN SELECT * FROM Books WHERE Genre = 'Fiction';
```

INDEXING IN DATABASES

Optional - Clean Up

DROP DATABASE BookStore;

permanently delete an entire database

DROP TABLE

DROP TABLE Books;
DROP TABLE Authors;

delete individual tables

Default Location of MySQL Database Files in Uniform Server

<UniformServerRoot>\usermysql\data\

C:\UniServerZ\usermysql\data\

C:\UniServerZ\usermysql\data\BookStore\

- **Do not manually edit or delete** files from this folder. Always use SQL commands.
- If you **drop a database** in MySQL (DROP DATABASE BookStore;), this folder is deleted.
- Always make backups using mysqldump rather than copying files manually.

Backup & Restore MySQL Databases in Uniform Server

<UniformServerRoot>\usermysql\data\

C:\UniServerZ\usermysql\data\

C:\UniServerZ\usermysql\data\BookStore\

- **Do not manually edit or delete** files from this folder. Always use SQL commands.
- If you **drop a database** in MySQL (DROP DATABASE BookStore;), this folder is deleted.
- Always make backups using **mysqldump** rather than copying files manually.

Backup & Restore MySQL Databases in Uniform Server

- **Open CMD** or Uniform Server's **MySQL Console**
- **Navigate to the MySQL binary directory** (only needed in CMD):

```
cd C:\UniServerZ\core\mysql\bin
```

Further Reading :

Chapter 11 Database Analysis and the DreamHome Case Study

Chapter 14: Normalisation

Chapter 12: Entity–Relationship Modelling

Chapter 13 Enhanced Entity–Relationship Modelling

Database Systems: A Practical Approach to Design, Implementation,
and Management Sixth edition by • Thomas Connolly • Carolyn Begg