

# Normalization

## Library Book Borrowing System

### Scenario 01:

A library stores book borrowing records, including **student details, book details, and librarian details** in the same table. This creates redundancy when multiple students borrow the same book.

Given Table: Book\_Borrowing

Borrow_ID	Student_ID	Student_Name	Book_ID	Book_Title	Librarian_Name	Librarian_Office
301	101	Alice	B001	SQL Basics	Mr. Parker	Room 5
302	102	Bob	B002	Data Science	Ms. Rose	Room 3
303	103	Charlie	B001	SQL Basics	Mr. Parker	Room 5

### Tasks:

1. Identify and explain anomalies in the table.
2. Determine the highest normal form satisfied.
3. Convert to 3NF and define the schema.

## **Scenario 02:**

A hospital stores **patient appointments, doctor details, and room assignments** in a single table. Each **doctor** sees multiple patients, and each **room is assigned to a doctor**. However, this design leads to **data redundancy and anomalies** when updating doctor details or room changes.

**Given Table: Patient\_Appointments**

	Patient_ID	Patient_Name	Doctor_ID	Doctor_Name	Specialization	Room_No	Date
	101	Alice	201	Dr. Brown	Cardiology	12	D1
	102	Bob	202	Dr. Smith	Orthopedics	15	D2
	103	Charlie	201	Dr. Brown	Cardiology	12	D3
	104	David	203	Dr. Adams	Neurology	20	D4
	105	Eve	201	Dr. Brown	Cardiology	12	D5

---

### **Tasks:**

- 1. Identify and explain the anomalies in the table (Insertion, Update, and Deletion).**
- 2. Determine the highest normal form satisfied.**
- 3. Convert the table into 3rd Normal Form (3NF) and define the schema of the new tables.**

# Online Shopping Order System

## Scenario 03:

An e-commerce website tracks **orders**, **customer details**, and **product details** in a **single table**. Each order contains **customer information**, **product details**, and **seller details**, causing **repetition when multiple orders are placed by the same customer or contain the same product**.

Given Table: `orders`

Order_ID	Customer_ID	Customer_Name	Product_ID	Product_Name	Seller_ID	Seller_Name	Price
701	301	Alice	P101	Laptop	S1	TechStore	900
702	302	Bob	P102	Phone	S2	MobileMart	500
703	301	Alice	P102	Phone	S2	MobileMart	500
704	303	Charlie	P103	Tablet	S3	TabletShop	300
705	302	Bob	P101	Laptop	S1	TechStore	900

---

## Tasks:

1. Identify and explain the anomalies in the table (Insertion, Update, and Deletion).
2. Determine the highest normal form satisfied.
3. Convert the table into 3rd Normal Form (3NF) and define the schema of the new tables.

# Relational Algebra

## Set 01

**Suppliers(sID, sName, address)**

**Parts(pID, pName, colour)**

**Catalog(sID, pID, price)**

**Catalog[sID] ⊆ Suppliers[sID]**

**Catalog[pID] ⊆ Parts[pID]**

**Note:**

- In this schema, everywhere we want values to match across relations, the attributes have matching names. And everywhere the attributes have matching names, we want values to match across relations.
- This means that natural join will do exactly what we want in all cases.
  1. Find the names of all red parts.
  2. Find all prices for parts that are red or green. (A part may have different prices from different manufacturers.)
  3. Find the sIDs of all suppliers who supply a part that is red or green.
  4. Find the sIDs of all suppliers who supply a part that is red and green.
  5. Find the names of all suppliers who supply a part that is red or green.

## Set 02

**Employees(number, name, age, salary)**

**Supervises(boss, employee)**

**Supervises[boss] ⊆ Employees[number]**

**Supervises[employee] ⊆ Employees[number]**

**Note:**

- In this schema, wherever we want values to match across relations, the attributes do not have matching names. This means that natural join will not force things to match up as we'd like.
  - In fact, since there are no attribute names in common across the two relations, natural join is no different from Cartesian product.
  - We are forced to use selection to enforce the necessary matching
6. Find the names and salaries of all bosses who have an employee earning more than 100.

Hint: Below each subexpression, write the names of the attributes in the resulting relation

## Set 03

We will use the following **example relations**:

1. **Employee (EmpID, Name, Age, Salary, DeptID)**
  - o Stores employee details.
2. **Department (DeptID, DeptName, Location)**
  - o Stores department details.
3. **Project (ProjID, ProjName, DeptID)**
  - o Stores project information.
4. **WorksOn (EmpID, ProjID, HoursWorked)**
  - o Stores which employees work on which projects.

- 1) Find employees earning more than 50,000 and working in the "HR" department.
- 2) Retrieve names of employees and their department names.

\*Try using joins and divisions

- 3) Find employees working on at least one project.
- 4) Find employees who do not work on any project.
- 5) Retrieve all projects that belong to a department with at least one employee.
- 6) Find employees who work on ALL projects of department 'IT'.
- 7) Find employees working on the project "Apollo" and earning more than 60,000.

\*Try using outer

- 8) Retrieve department names that have no employees.
  - 9) Find employees and their project names (even if they don't work on any project).
- 
- 10) Retrieve employees working on the same project as 'Alice'.
  - 11) Find departments where all employees earn more than 40,000.

# Relational Calculus

## TRC1

TRC1:

Relations:

- branch (branch-name, branch-city, assets)
- customer (customer-name, customer-street, customer-city)
- loan (loan-number, branch-name, amount)
- borrower (customer-name, loan-number)
- account (account-number, branch-name, balance)
- depositor (customer-name, account-number)

- 1) Find the branch name, loan number, and amount for loans over \$1500.
- 2) Find the loan number for each loan with an amount greater than \$1500.
- 3) Find the names of all customers who have a loan from the Pune branch.
- 4) Find all customers who have a loan, an account, or both at the bank.

## TRC2:

```
Instructor (ID, Name, Dept_Name, Salary)
Course (Course_ID, Title, Dept_Name, Credits)
Department (Dept_Name, Building, Budget)
Section (Course_ID, Sec_ID, Semester, Year, Building, Room_No, Time_Slot_ID)
Teaches (ID, Course_ID, Sec_ID, Semester, Year)
Student (ID, Name, Dept_Name, Tot_Cred)
Advisor (S_ID, I_ID)
Takes (ID, Course_ID, Sec_ID, Semester, Year, Grade)
Classroom (Building, Room_Number, Capacity)
Time Slot (Time_Slot_ID, Day, Start_Time, End_Time)
```

1. Find the ID, name, dept name, and salary for instructors whose salary is greater than \$80,000.
2. Find the instructor ID for each instructor with a salary greater than \$80,000.
3. Find the names of all instructors whose department is in the Watson building.
4. Find the set of all courses taught in the Fall 2009 semester, the Spring 2010 semester, or both.
5. Find only those course id values for courses that are offered in both the Fall 2009 and Spring 2010 semesters.
6. Find all the courses taught in the Fall 2009 semester but not in the Spring 2010 semester.

## **DRC 1**

The relations (tables):

branch (branch-name, branch-city, assets)  
customer (customer-name, customer-street, customer-city)  
loan (loan-number, branch-name, amount)  
borrower (customer-name, loan-number)  
account (account-number, branch-name, balance)  
depositor (customer-name, account-number)

- 1) Find the loan number, branch name, and amount for loans of over \$1500.**
- 2) Find all loan numbers for loans with an amount greater than \$1500.**
- 3) Find the names of all customers who have a loan from the Pune branch and find the loan amount.**
- 4) Find the names of all customers who have a loan, an account, or both at the Pune branch.**

## **DRC 2**

Relations:

- **Instructor** (*ID, Name, Dept\_Name, Salary*)
- **Course** (*Course\_ID, Title, Dept\_Name, Credits*)
- **Department** (*Dept\_Name, Building, Budget*)
- **Section** (*Course\_ID, Sec\_ID, Semester, Year, Building, Room\_No, Time\_Slot\_ID*)
- **Teaches** (*ID, Course\_ID, Sec\_ID, Semester, Year*)
- **Student** (*ID, Name, Dept\_Name, Tot\_Cred*)
- **Advisor** (*S\_ID, I\_ID*)
- **Takes** (*ID, Course\_ID, Sec\_ID, Semester, Year, Grade*)
- **Classroom** (*Building, Room\_Number, Capacity*)
- **Time Slot** (*Time\_Slot\_ID, Day, Start\_Time, End\_Time*)

- 1) Find ID name Department name salary for instructors whose salary is greater than \$80,000 .**
- 2) Find the instructor ID for each instructor with a salary greater than 80,000**
- 3) find the names of all instructors in the physics department together with the course ID of all courses they teach.**
- 4) Find the set of all courses taught in the Fall 2009 semester, the Spring 2010 semester, or both. (use section relation)**