
MAHARISHI UNIVERSITY of MANAGEMENT
Engaging the Managing Intelligence of Nature
Computer Science Department
CS401 Modern Programming Practices (MPP)

Lecture 6: Relational Model & Normalization

Wholeness of the Lesson

The relational model is an application of elementary relation theory to systems that provide shared access to large banks of formatted data.

Science & Technology of Consciousness: Consciousness-based education is an application of Maharishi's Vedic Science in the field of education.

Lesson Outline

1. Relational Model
2. Introduce to Normalization
3. Functional Dependency
4. The process of normalization

What is DBMS?

- A Database Management System (DBMS) is software that is used to store, manage, and retrieve data efficiently in a structured way. It acts as an interface between the database and the users or applications that access it.
- Key Functions of a DBMS
  - Data Storage & Retrieval – Stores and retrieves data efficiently.
  - Data Security & Integrity – Ensures only authorized users can access data.
  - Concurrency Control – Manages multiple users accessing data simultaneously.
  - Backup & Recovery – Protects data from crashes and failures.
  - Query Processing – Allows users to retrieve data using SQL queries.

Relational Model

- The Relational Model is a conceptual framework for organizing and managing data in a database.
- It was introduced by E.F. Codd in 1970 and is the foundation for modern relational database management systems (RDBMS).

Key Concepts of the Relational Model

- Tables (Relations):
  - Data is organized into tables, also called relations.
  - Each table represents an entity or a relationship between entities.
  - Example: A Students table represents the entity "Student."
- Rows (Tuples):
  - Each row in a table represents a single record or instance of an entity.
  - Example: A row in the Students table represents a specific student.
- Columns (Attributes):
  - Each column represents an attribute or property of the entity.
  - Example: Columns in the Students table might include StudentID, Name, and Age.
- Constraints:
  - Rules enforced on data to maintain integrity.
  - Examples: Primary Key, Foreign Key, Unique, Not Null, Check.

Key Concepts of the Relational Model - Relational Keys

- Candidate Key
- Primary Key
- Foreign Key

- Composite Key
- Natural Key
- Surrogate Key

Relational Keys contd..

- **Candidate Key**
  - a candidate key is a set of one or more attributes (columns) in a table that can uniquely identify each row (tuple) in that table. A candidate key must satisfy two properties:
    - Uniqueness: No two rows in the table can have the same value for the candidate key.
    - Irreducibility: No subset of the candidate key can uniquely identify a row. In other words, the candidate key must be minimal.

Relational Keys contd..

- **Primary Key**
  - Candidate key selected to identify tuples uniquely within relation.
  - Characteristics of a primary key
    - Is unique
    - Cannot be NULL
    - Value never changes
    - Only one PK for one table.

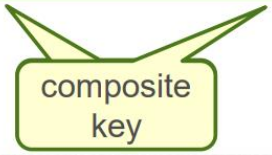
StudID	Roll No	First Name	LastName	Email
1	11	Tom	Price	abc@gmail.com
2	12	Nick	Wright	xyz@gmail.com
3	13	Dana	Natan	mno@yahoo.com

primary key

Relational Keys contd..

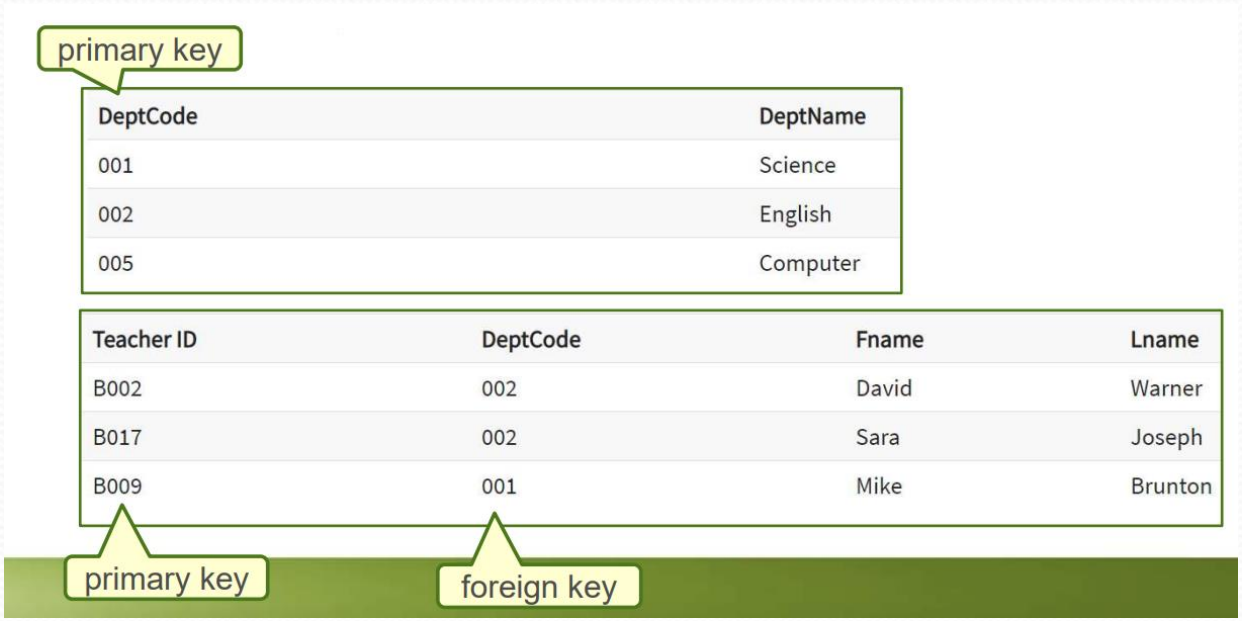
- **Composite Key**
  - When a key consists of more than one attributes (columns) it's called as a **Composite Key**.
  - A composite key is used when no single attribute can uniquely identify a row, but a combination of attributes can.

OrderNo	PorductID	Product Name	Quantity
B005	JAP102459	Mouse	5
B005	DKT321573	USB	10
B005	OMG446789	LCD Monitor	20
B004	DKT321573	USB	15
B002	OMG446789	Laser Printer	3



Relational Keys contd..

- **Foreign Key**
  - A foreign key is a column or a set of columns in a table that establishes a link between data in two tables.
  - A table can have o to many FKs.



Relational Keys contd..

- **Natural Key**
  - A natural key is a type of unique key in a database that is derived from real-world, meaningful data, rather than an artificially generated identifier like a surrogate key (e.g., an auto-incremented ID).
  - Has a meaning in the business domain
    - SSN, ISBN, VIN
- **Surrogate Key**
  - A Surrogate Key is a unique identifier for a record in a database that is artificially created rather than derived from real-world data.
  - Has no meaning in the business domain
    - Auto-incremented Integer (e.g., UserID, OrderID)
    - UUID (Universally Unique Identifier)
    - Database Sequence (e.g., Oracle's SEQUENCE feature)

Always a better choice

Integrity Constraints

- Integrity constraints are rules defined in a database to ensure the accuracy, consistency, and reliability of the data.
- Integrity is concerned with the quality of data itself.
- Integrity is usually expressed in terms of constraints, which are consistency rules that the database is not permitted to violate.
  - For example, we may want to specify a constraint that no member of staff can manage more than 100 properties at any one time.

Types of Integrity Constraints

- **Entity Integrity**
  - Ensures each row in a table has a unique and non-null primary key.
  - Prevents duplicate rows and ensures each record is identifiable.
  - Example:
    - In an `Employees` table, the `EmployeeID` column must be unique and not null.

```
CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,
    Name VARCHAR(100) NOT NULL
);
```

Types of Integrity Constraints contd...

- **Referential Integrity**
  - Maintains consistency between related tables through foreign keys.
  - A foreign key must match an existing primary key value in another table (or be NULL, if allowed).
  - Example:
    - In an `Orders` table, the `CustomerID` column must reference a valid `CustomerID` in the `Customers` table.

```
CREATE TABLE Orders (
    OrderID INT PRIMARY KEY,
    OrderDate DATE,
    CustomerID INT,
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);
```

Types of Integrity Constraints contd...

- **Domain Constraints**
  - Ensure that the value of an attribute (column) falls within a specified domain (set of valid values).
  - Examples:
    - A column Age must be a positive integer.
    - A column Gender must be either 'Male' or 'Female'.

```
CREATE TABLE Students (
    StudentID INT PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    Age INT CHECK (Age > 0)
)
```

- **General Constraints**
  - Additional rules specified by users or database administrators that define or constrain some aspect of the enterprise.
    - E.g. A branch may have only 20 staff members

Main Point

A database can be modeled as a set of relations. It is always advantageous to find a simple basis for a complex field because it provides a way to manage the complexity of the field.

cience & Technology of Consciousness: Vedic Science has discovered that the simplest form of awareness is the basis for all of the creation.

Lesson Outline

1. Relational Model
2. Introduce to Normalization
3. Functional Dependency
4. The process of normalization

Normalization

- Normalization is a database design technique to organize large amounts of data efficiently so that a fact is stored at one place only.
- Normalization helps to reduce data redundancy.
- **Benefits of Minimizing Data Redundancy**
  - Updates to the data stored in the database are achieved with a minimal number of operations thus reducing the opportunities for data inconsistencies.
  - Reduction in the file storage space required by the base relations thus minimizing costs.
  - Database will be easier for the user to access and maintain.
  - Avoids problems like update anomalies.

Purpose of Normalization

- Purpose of Normalization is to produce a set of suitable relations that support the data requirements of an enterprise.
- **Characteristics of a suitable set of relations include:**
  - The *minimal* number of attributes necessary to support the data requirements of the enterprise;
  - Attributes with a close logical relationship are found in the same relation;
  - *Minimal* redundancy with each attribute represented only once with the important exception of attributes that form all or part of foreign keys.

Data Redundancy Problems

Staff

staffNo	sName	position	salary	branchNo
SL21	John White	Manager	30000	B005
SG37	Ann Beech	Assistant	12000	B003
SG14	David Ford	Supervisor	18000	B003
SA9	Mary Howe	Assistant	9000	B007
SG5	Susan Brand	Manager	24000	B003
SL41	Julie Lee	Assistant	9000	B005

Branch

branchNo	bAddress
B005	22 Deer Rd, London
B007	16 Argyll St, Aberdeen
B003	163 Main St, Glasgow

Staff Branch

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

- StaffBranch relation has redundant data; the details of a branch are repeated for every member of staff.
- In contrast, the branch information appears only once for each branch in the Branch relation and only the branch number (branchNo) is repeated in the Staff relation, to represent where each member of staff is located.

Data Redundancy

STUDENTS TABLE

rollno	name	branch	hod	office_tel
1	Akon	CSE	Mr. X	53337
2	Bkon	CSE	Mr. X	53337
3	Ckon	CSE	Mr. X	53337
4	Dkon	CSE	Mr. X	53337

Data Redundancy and Update Anomalies

- Relations that contain redundant information may potentially suffer from update anomalies.

- Types of update anomalies include
  - **Insertion**
  - **Deletion**
  - **Modification**

Update Anomalies - Insertion

- To insert details of new staff into the StaffBranch relation:
  - For the staff located at branch number B007, we must enter the correct details of this branch so that these details will be consistent with other tuples in this relation.
- To insert details of a new branch which has not assigned any staff yet:
  - Violates entity integrity as staffNo being a PK cannot be null.
  - So cannot insert a new branch unless there are staff in that branch!

StaffBranch

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

Update Anomalies - Insertion

STUDENTS TABLE

rollno	name	branch	hod	office_tel
1	Akon	CSE	Mr. X	53337
2	Bkon	CSE	Mr. X	53337
3	Ckon	CSE	Mr. X	53337
4	Dkon	CSE	Mr. X	53337
5	Ekon	CSE	Mr. X	53337

We have to enter branch, hod and office_tel for every student we add to the table

Update Anomalies - Deletion

- If we delete a tuple from the StaffBranch relation that represents the last member of staff located at a branch, the details about that branch are also lost from the database.

StaffBranch

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

Update Anomalies - Modification

STUDENTS TABLE

rollno	name	branch	hod	office_tel
1	Akon	CSE	~~Mr. X~~ Mr. Y	53337
2	Bkon	CSE	~~Mr. X~~ Mr. Y	53337
3	Ckon	CSE	~~Mr. X~~ Mr. Y	53337
4	Dkon	CSE	~~Mr. X~~ Mr. Y	53337

When Mr. X leaves the department and is replaced by Mr. Y we have to update all records

Easy to forget a record

What if we miss one? What if someone adds a new value while we are performing this update? What if someone overwrites the value, or does something else to one of the rows, which means our update didn't work?

Update Anomalies - Modification

- Updating an address of a branch in the *StaffBranch* relation may leave the database in an inconsistent state if the update is not done properly to all the tuples with that branch number.

- So to avoid all these anomalies, it's required to decompose the StaffBranch table into 2 separate tables *Staff* and *Branch* with the process of Normalization. We'll go through the technique in upcoming slides.

Main Point

Normalization is a bottom-up technique for producing a set of relations which will be free from all the update anomalies.

Science & Technology of Consciousness: Practice of the Transcendental Meditation technique normalizes the functioning of the mind and body, so that they become free from stress and operate more and more in accord with all the laws of nature.

Lesson Outline

1. Relational Model
2. Introduce to Normalization
3. Functional Dependency
4. The process of normalization

Functional Dependencies

- Functional dependency is a concept in database normalization that describes the relationship between attributes (columns) in a table.
- It is a constraint that specifies how the values of one set of attributes determine the values of another set of attributes.

- A functional dependency (FD) is denoted as $X \rightarrow Y$, where:
  - X is a set of attributes (determinant).
  - Y is a set of attributes (dependent).
- This means that the value of X uniquely determines the value of Y. In other words, for any two rows in a table, if the values of X are the same, then the values of Y must also be the same.

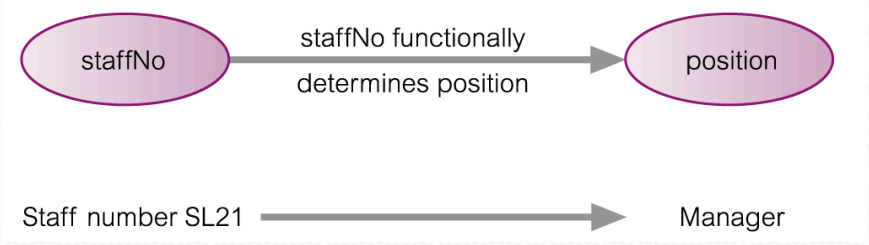
Characteristics of Functional Dependencies

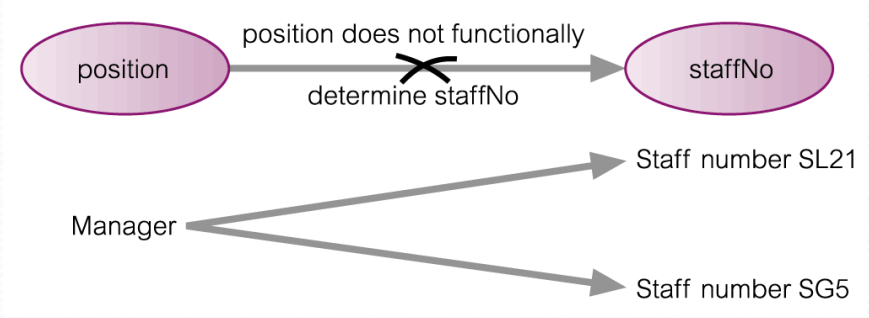
- There is a *one-to-one relationship* between the attribute(s) on the left-hand side (determinant) and those on the right-hand side (dependent) of a functional dependency.
- Holds for *all* time.
- The determinant has the *minimal* number of attributes necessary to maintain the dependency with the attribute(s) on the right hand-side.
  - **This requirement is called *full functional dependency*.**

An Example of Functional Dependency

Staff Branch

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London





Example Functional Dependency that holds true for all Time

- Based on sample data, the following functional dependencies appear to hold.

  staffNo → sName

  sName → staffNo

- However, the only functional dependency that remains true *for all possible values* for the staffNo and sName attributes of the Staff relation is:

  staffNo → sName

StaffBranch

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

Identifying Functional Dependencies

- Identifying functional dependencies is easier with clear attribute meanings and their relationships.
- This information should be provided by the enterprise in the form of discussions with users and/or documentation such as the users' requirements specification.
- However, if these aren't available, designers may need to rely on their experience and common sense.

Example - Identify FDs for the StaffBranch Relation

- Assume that position and branch determine a member of staff's salary.
- There is only one branch at a particular address.

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

Example - Identify FDs for the StaffBranch Relation contd..

- The FDs for the StaffBranch relation are:

staffNo → sName, position, salary, branchNo, bAddress

branchNo → bAddress

bAddress → branchNo

branchNo, position → salary

bAddress, position → salary

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

Identifying Primary Key for a Relation using FDs

- Main purpose of identifying a set of functional dependencies for a relation is to specify the set of integrity constraints that must hold on a relation.
- An important integrity constraint to consider first is the identification of candidate keys, one of which is selected to be the primary key for the relation.

Example - Identify Primary Key for StaffBranch Relation

- To identify all candidate key(s), identify the attribute (or group of attributes) that uniquely identifies each tuple in this relation OR that functionally determines all other attributes.
- All attributes that are not part of a candidate key should be functionally dependent on the key.
- The only candidate key and therefore primary key for StaffBranch relation, is **staffNo**, as all other attributes of the relation are functionally dependent on staffNo.

staffNo → sName, position, salary, branchNo, bAddress

branchNo → bAddress

bAddress → branchNo

branchNo, position → salary

bAddress, position → salary

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

Types of Functional Dependencies

- **Full functional dependency**
- **Partial dependency**
- **Transitive dependency**

Full Functional Dependency

- A functional dependency X → Y (where X and Y are sets of attributes) is a **full functional dependency** if:
  - Y is functionally dependent on X (X → Y), AND
  - Y is not functionally dependent on any proper subset of X
- In other words, removing any attribute from X would break the dependency - all attributes in X are necessary to uniquely determine Y.

- **Key Characteristics**
  - **No extraneous attributes**: The determinant (X) doesn't contain any attributes that aren't needed to determine the dependent attributes (Y)
  - **Minimal determinant**: X is the minimal set of attributes that can determine Y
  - **Irreducible**: The dependency cannot be further reduced without losing information

Example of Full Functional Dependency

STOCKS (Symbol, Company, Headquarters, Date, ClosePrice)

Symbol	Company	Headquarters	Date	ClosePrice
MSFT	Microsoft	Redmond, WA	09/07/2023	23.96
MSFT	Microsoft	Redmond, WA	09/08/2023	23.93
MSFT	Microsoft	Redmond, WA	09/09/2023	24.01
ORCL	Oracle	Redwood Shores, CA	09/07/2023	24.27
ORCL	Oracle	Redwood Shores, CA	09/08/2023	24.14
ORCL	Oracle	Redwood Shores, CA	09/09/2023	24.33

- **(Symbol, Date) → ClosePrice** — FFD
- **(Symbol, Date) → Company** X **FFD**

Partial Functional Dependency

- It describes a situation where an attribute (or set of attributes) is functionally dependent on only a part of a candidate key, rather than the entire key.

- A functional dependency $X \rightarrow Y$ is a partial functional dependency if:
  - X is a proper subset of a candidate key (i.e., not the entire key).
  - Y is functionally dependent on X.
- In other words, the dependent attribute(s) Y can be determined by only a part of the candidate key, rather than the entire key.

Example of Partial Dependency

STOCKS (Symbol, Company, Headquarters, Date, ClosePrice)

Symbol	Company	Headquarters	Date	ClosePrice
MSFT	Microsoft	Redmond, WA	09/07/2023	23.96
MSFT	Microsoft	Redmond, WA	09/08/2023	23.93
MSFT	Microsoft	Redmond, WA	09/09/2023	24.01
ORCL	Oracle	Redwood Shores, CA	09/07/2023	24.27
ORCL	Oracle	Redwood Shores, CA	09/08/2023	24.14
ORCL	Oracle	Redwood Shores, CA	09/09/2023	24.33

- **(Symbol, Date) → (Company, Headquarters)**

Because:

- Symbol → (Company, Headquarters)

Partial Dependency

Transitive Dependencies

- A transitive dependency occurs in a relational database when a non-prime (non-key) attribute depends on another non-prime attribute rather than directly on the primary key. This creates an indirect functional dependency, which can lead to data redundancy and update anomalies.
- A transitive dependency exists when:
  - **A** → **B** (Attribute **B** depends on attribute **A**), and
  - **B** → **C** (Attribute **C** depends on attribute **B**),
  - Therefore, **A** → **C** (**C** transitively depends on **A**).

Example of Transitive Dependency

- Consider the following FDs in the StaffBranch relation.
  - **staffNo → sName, position, salary, branchNo, bAddress**
  - **branchNo → bAddress**
- Identifying the Transitive Dependency:
  - Primary Key: **staffNo** (since it determines all other attributes)
  - Non-Key Attributes: **sName**, **position**, **salary**, **branchNo**, **bAddress**
- The transitive dependency exists because:
  - **staffNo → branchNo** (direct FD)
  - **branchNo → bAddress** (direct FD)
  - Therefore, **staffNo → bAddress** (transitive FD, since **bAddress** depends on **staffNo** via **branchNo**)

Main Point

A functional dependency (FD) describes a permanent semantic relationship between sets of attributes in a relation schema that all relation instances of that schema must adhere to.

Science & Technology of Consciousness: Due to this permanence a functional dependency is functioning as a law of Nature. Vedic Science teaches respect for the laws of Nature and provides a simple technique of TM to bring action into accord with all the laws of Nature.

Lesson Outline

1. Relational Model
2. Introduce to Normalization
3. Functional Dependency
4. The process of normalization

The Process of Normalization

- Formal technique for analyzing a relation based on its primary key and the functional dependencies between the attributes of that relation.

- Often executed as a series of steps. Each step corresponds to a specific normal form, which has known properties.

Example of Unnormalized Form (UNF)

- Let's consider a scenario of a Student Registration System

Student_ID	Student_Name	Course_IDs	Course_Names	Instructor	Instructor_Contact
S101	John Doe	C001, C002	Math, Physics	Dr. Smith, Dr. Brown	12345, 67890
S102	Alice Smith	C001	Math	Dr. Smith	12345

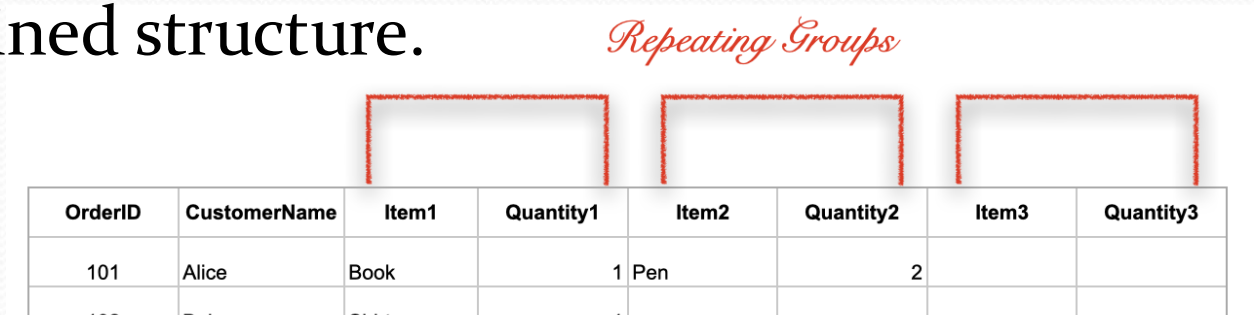
Problem – Unnormalized Table

- ❌ Repeating Groups – Multiple values in Course_IDs, Course_Names, Instructor, and Instructor_Contact fields.
- ❌ Data Redundancy – Instructor details are repeated for each student.
- ❌ Difficult to Query – Searching for students in Physics requires parsing multiple values in Course_Names.

Student_ID	Student_Name	Course_IDs	Course_Names	Instructor	Instructor_Contact
S101	John Doe	C001, C002	Math, Physics	Dr. Smith, Dr. Brown	12345, 67890
S102	Alice Smith	C001	Math	Dr. Smith	12345

First Normal Form (1NF)

- First Normal Form (1NF) ensures that:
  - Each column contains atomic values (no multiple values in a single field).
  - Each column has unique values (no repeating groups or arrays).
  - Each row is uniquely identifiable (using a primary key).
- 1NF is the first step in normalization, eliminating duplicate data by ensuring that each table has a well-defined structure.



OrderID	CustomerName	Item1	Quantity1	Item2	Quantity2	Item3	Quantity3
101	Alice	Book	1	Pen	2		

Step 1: Convert to 1NF

- Remove repeating groups (each field must contain atomic values).

Student_ID	Student_Name	Course_ID	Course_Name	Instructor	Instructor_Contact
S101	John Doe	C001	Math	Dr. Smith	12345
S101	John Doe	C002	Physics	Dr. Brown	67890
S102	Alice Smith	C001	Math	Dr. Smith	12345

Improvements in 1NF

- ✅ No Repeating Groups – Each field has atomic values.
- ✅ Easier Queries – Searching for students enrolled in Physics is now simple.

Problem of 1NF

- Even though it's in 1NF, the table has data redundancy, which can lead to:

1. Repeated Instructor Data
   - For example, Dr. Smith's contact (12345) is repeated for both students taking Math.
   - If Dr. Smith's contact changes, you'd need to update it in multiple places → Update Anomaly.

Student_ID	Student_Name	Course_ID	Course_Name	Instructor	Instructor_Contact
S101	John Doe	C001	Math	Dr. Smith	12345
S101	John Doe	C002	Physics	Dr. Brown	67890
S102	Alice Smith	C001	Math	Dr. Smith	12345

Problem of 1NF

- Even though it's in 1NF, the table has data redundancy, which can lead to:

1. Repeated Student Info
   - "John Doe" is repeated with every course he takes.
   - If his name was misspelled in one row, inconsistency could happen → Insert/Update Anomalies.

Student_ID	Student_Name	Course_ID	Course_Name	Instructor	Instructor_Contact
S101	John Doe	C001	Math	Dr. Smith	12345
S101	John Doe	C002	Physics	Dr. Brown	67890
S102	Alice Smith	C001	Math	Dr. Smith	12345

Convert to 2NF

- Second Normal Form (2NF) ensures that:
  - The table is already in First Normal Form (1NF) (no repeating groups or multi-valued columns).
  - No partial dependency exists, meaning every non-key column must be fully dependent on the whole primary key, not just part of it.

- 2NF primarily applies to tables that have composite primary keys (primary keys made of multiple columns).

Convert to 2NF

- Students Table (Only Student-Specific Data)

Student_ID	Student_Name
S101	John Doe
S102	Alice Smith

- Courses Table (Only Course-Specific Data)

Course_ID	Course_Name	Instructor	Instructor_Contact
C001	Math	Dr. Smith	12345
C002	Physics	Dr. Brown	67890

Convert to 2NF

- Student_Courses Table (Many-to-Many Relationship)

Student_ID	Course_ID
S101	C001
S102	C002

Improvements in 2NF

- ✅ No Partial Dependencies – Student_Name depends only on Student_ID, and Instructor depends only on Course_ID.
- ✅ Less Redundancy – Student and course details are stored separately.

Problem of 2NF

- Problem: Transitive Dependency

- The primary key here is Course_ID.
- Instructor_Contact depends on Instructor, not directly on Course_ID.

- So:
- Course_ID → Instructor
- and
- Instructor → Instructor_Contact

- ⛔ That's a transitive dependency → violates 3NF.

Convert to 3NF

- Third Normal Form (3NF) ensures that:
  - The table is already in Second Normal Form (2NF) (no partial dependencies).
  - No transitive dependency exists, meaning all non-key attributes depend only on the primary key and not on another non-key attribute.

Convert to 3NF

- Updated Courses Table

Course_ID	Course_Name	Instructor_ID
C001	Math	1001
C002	Physics	1002

- New Instructors Table

Instructor_ID	Instructor	Instructor_Contact
1001	Dr. Smith	12345
1002	Dr. Brown	67890

Improvements in 3NF

- No Transitive Dependencies – Instructor_Contact is now stored separately.
- ✅ Better Data Integrity – Instructor contact updates only affect the Instructors Table.
- ✅ Efficient Queries – The database structure is now fully optimized.

Summary of the Normalization Process

Step	Changes Made	Problem Solved
UNF → 1NF	Removed repeating groups	Ensured atomic values (no multiple values in a single column)
1NF → 2NF	Removed partial dependencies	Ensured every non-key attribute depends on the full primary key
2NF → 3NF	Removed transitive dependencies	Ensured non-key attributes only depend on the primary key

Connecting the Parts of Knowledge With the Wholeness of Knowledge

UNITY CHART

CONNECTING THE PARTS OF KNOWLEDGE WITH THE WHOLENESS OF KNOWLEDGE:

Relational Model Basics & Relational Algebra Operations

1. Science repeatedly calls upon mathematics to help it model the real world.
2. The operators of the relational algebra have been found to be ideal for modeling the operations that are needed to query and update a database.

---

3. Transcendental consciousness is the experience of the simplest and most abstract state of awareness which underlies all states of greater excitation.
4. Impulses within the Transcendental Field: Nature accomplishes what it needs by having its impulses in the transcendental field be as efficient as possible.
5. Wholeness moving within itself: In unity consciousness one experiences everything as excitations of pure consciousness that underlies and connects all diversity.