Database Programming – Lecture 4

ERD & DB Design

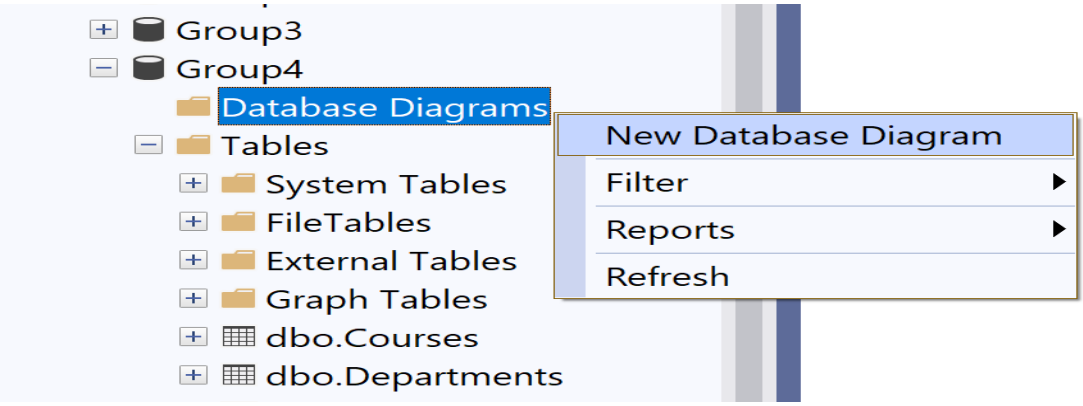
BY

MOHAMMED ELSADEK

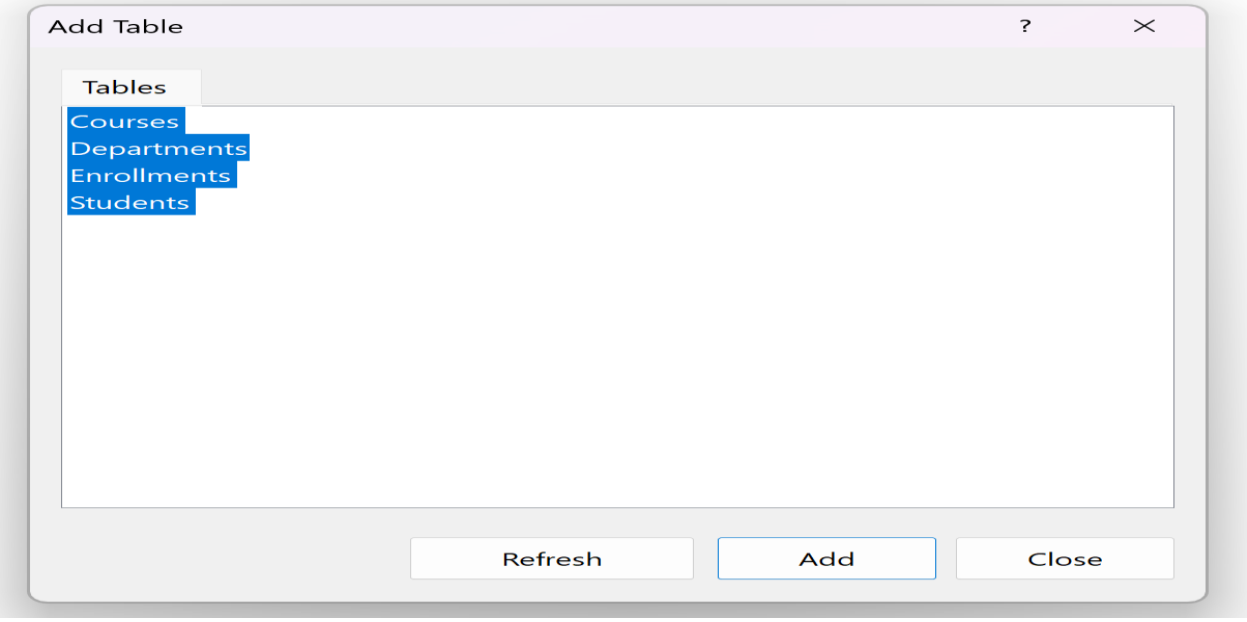
Constructing (Drawing) the Entity Relationship Diagram ERD

Using SSMS (Old Approach – Still Valid)

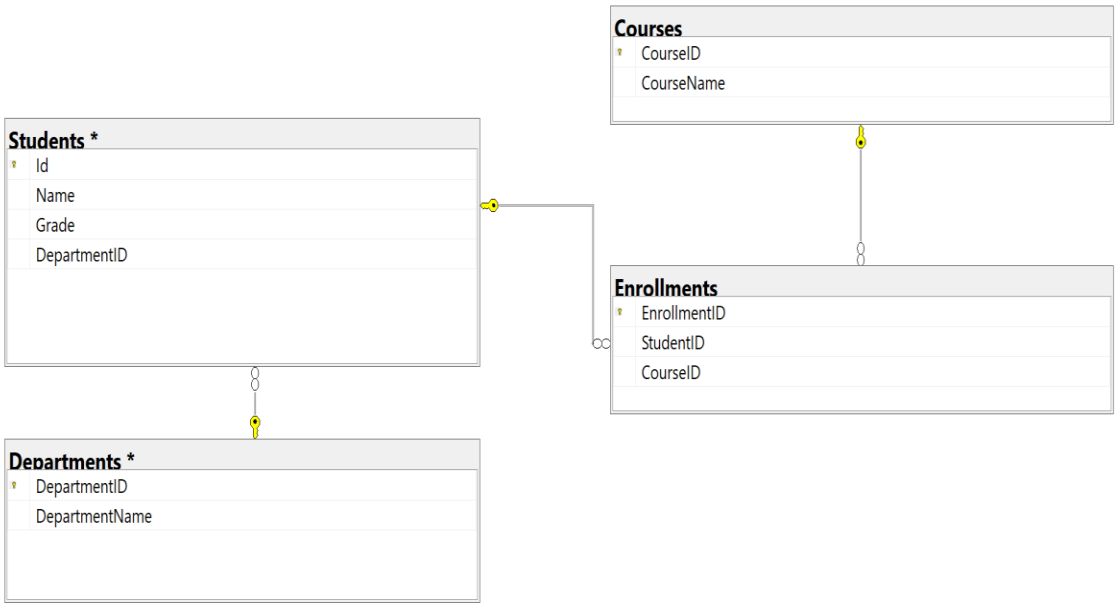
1- Click New DB Diagram



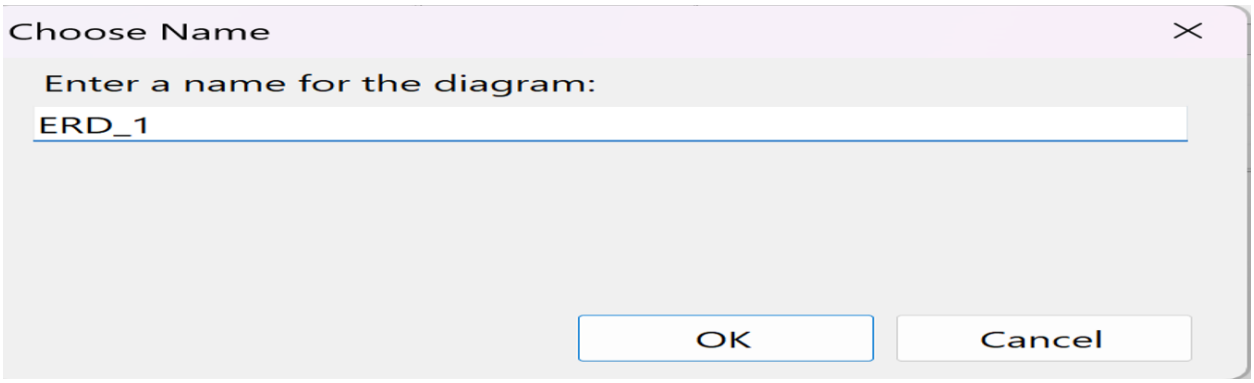
2- Select Tables



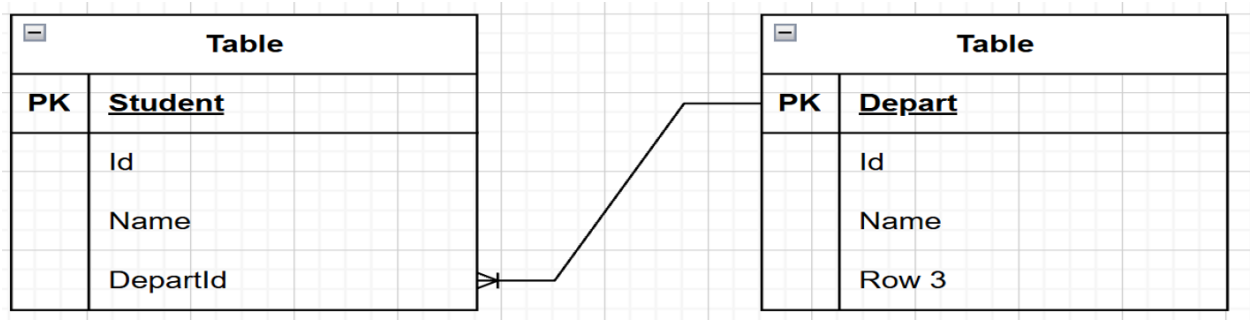
3- ERD


Courses
CourseID
CourseName
Students *
Id
Name
Grade
DepartmentID
Enrollments
EnrollmentID
StudentID
CourseID
Departments *
DepartmentID
DepartmentName

4- Save


Choose Name
Enter a name for the diagram:
ERD_1
OK
Cancel

Using draw.io: https://app.diagrams.net/



Database Design

Database Design is the process of structuring data logically and physically to ensure efficiency, **integrity, scalability, and maintainability**.

Goals of good database design:

- **Eliminate data redundancy**
- **Ensure data integrity**
- **Improve performance**
- **Support scalability**
- **Maintain clarity and simplicity**

Database Design is the foundation of any successful system. A clean ERD and proper **normalization** lead to:

- Better performance
- Easier maintenance
- Reliable data integrity

Database Design Process (Step-by-Step)

Step 1: Requirements Analysis
Understand what the system should store and manage.

Step 2: Identify Entities
Entities represent real-world objects (e.g., Student, Course, Department).

Step 3: Define Attributes
Attributes are properties of entities (e.g., StudentID, Name, Email).

Step 4: Define Primary Keys (PK)
A Primary Key uniquely identifies each record in a table.
Properties of PK:
- Unique
- Not NULL
- Stable

Step 5: Define Relationships
Types:
- One-to-One (1:1)
- One-to-Many (1:M)
- Many-to-Many (M:N)

Step 6: Apply Normalization
Ensure the database follows normalization rules (1NF, 2NF, 3NF).

Step 7: Add Constraints and Indexes
Use:
- NOT NULL
- UNIQUE
- CHECK
- DEFAULT
- FOREIGN KEY
- Indexes for performance

Step 7: Draw ERD and review the full DB design.

Normalization Explained

What DB Normalization?

Normalization is an important process in database design that helps improve the database's efficiency, consistency, and accuracy. It makes it easier to manage and maintain the data and ensures that the database is adaptable to changing business needs.

Why DB Normalization?

Before Normalization

Employee_Department

Emp_ID	Emp_Name	Department	Dept_Location	Emp_Skills
101	Nick Wise	HR	London	Recruitment, Payroll
102	John Cader	Finance	Australia	Budgeting
103	Lily Case	HR	London	Recruitment
104	Ford Dawid	IT	Chicago	Programming, Testing

Problems in the Employee_Department Relation

1. **Insertion Anomaly:** If a new department is created but no employee is assigned to it yet. If employee data is mandatory, we cannot store the DB location because we need an employee record to insert.
2. **Update Anomaly:** If the location of the HR department changes, we must update it in multiple rows (for both Nick Wise and Lily Case). If one row is missed, the data becomes inconsistent.
3. **Deletion Anomaly:** If all employees in the IT department leave, we lose the department information, including its location.
4. **Data Redundancy:** The department location is repeated for every employee in the same department.

Normal Forms of DB Normalization

1NF – 2NF – 3NF

First Normal Form (1NF):

- No repeating groups
- Atomic (single) values in each column

Ex. Before 1NF

COURSES Table

ID	Name	Courses
1	A	c1, c2
2	E	c3
3	M	c2, c3

After 1NF

COURSES Table

ID	Name	Courses
1	A	c1
1	A	c2
2	E	c3
3	M	c2
3	M	c3

Second Normal Form (2NF):

- Must be in 1NF
- No partial dependency on **composite primary key**

A composite primary key is the PK that consists of more than one field (column)

Ex. Before 2NF

COURSES Table

STUD_NO	COURSE_NO	COURSE_FEE
1	C1	1000
2	C2	1500
1	C4	2000
4	C3	1000
4	C1	1000
2	C5	2000

After 2NF

STUDENT_COURSES Table

STUD_NO	COURSE_NO
1	C1
2	C2
1	C4
4	C3
4	C1
2	C5

COURSES Table

COURSE_NO	COURSE_FEE
C1	1000
C2	1500
C4	2000
C3	1000
C5	2000

Third Normal Form (3NF):

- Must be in 2NF
- No transitive dependency, a transitive dependency occurs when one non-key attribute depends on another non-key attribute rather than depending directly on the primary key.

A non-key attribute is any attribute, column, that is not part of the primary key.

Ex. Before 3NF

CANDIDATE Table

CAND_NO	CAND_NAME	CAND_STATE	CAND_COUNTRY	CAND_AGE
1	Amayra	West Bengal	India	18
2	Rihaan	Haryana	India	17
3	Manya	Haryana	India	19

After 3NF

CANDIDATE Table

CAND_NO	CAND_NAME	CAND_STATE	CAND_AGE
1	Amayra	West Bengal	18
2	Rihaan	Haryana	17
3	Manya	Haryana	19

STATE_COUNTRY Table

CAND_STATE	CAND_COUNTRY
West Bengal	India
Haryana	India

Normalization helps eliminate redundancy and avoid insert, update, and delete anomalies.
Why is 3NF Important?

1. Eliminates Redundancy: **3NF** helps to remove unnecessary duplication of data by ensuring that non-prime attributes (attributes not part of any candidate key) depend directly on the primary key, not on other non-prime attributes.

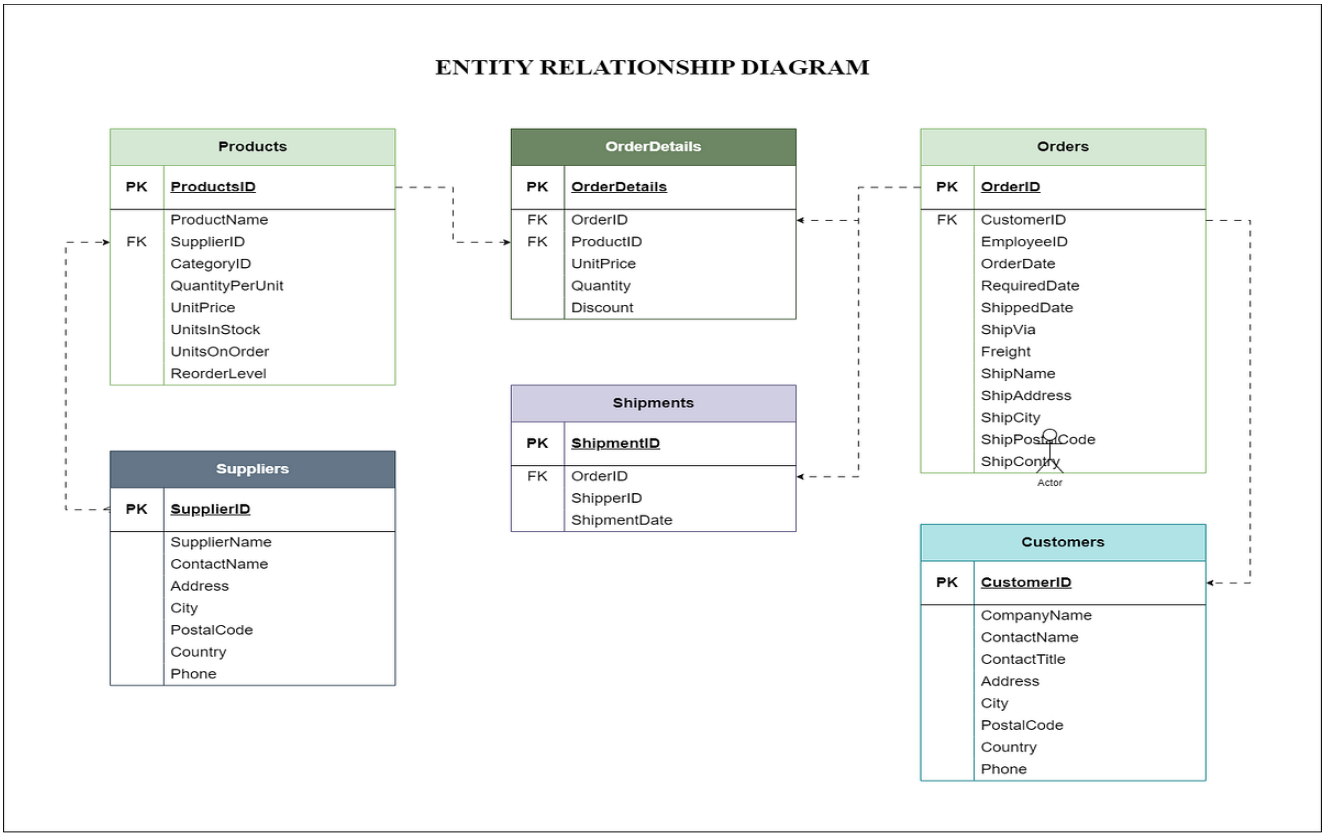
2. Prevents Anomalies: A table in **3NF** is free from common anomalies such as:

- **Insertion Anomaly**: The inability to insert data without having to insert unwanted or redundant data.
- **Update Anomaly**: The need to update multiple rows of data when a change occurs in one place.
- **Deletion Anomaly**: The unintended loss of data when a record is deleted.

3. Preserves Functional Dependencies: **3NF** ensures that all functional dependencies are preserved, meaning that the relationships between attributes are maintained.

4. Lossless Decomposition: When decomposing a relation to achieve **3NF**, the decomposition should be **lossless**, meaning no information is lost in the process of normalization.

Example: Simple Purchasing Orders Database Design



[Logical vs Physical Design]

Logical Design:
- ERD (Entity Relationship Diagram)
- Entities and relationships

- Business rules

Physical Design:
- SQL tables
- Data types
- Constraints

Common Design Mistakes

- Storing multiple values in one column
- Using names instead of IDs for relationships
- No normalization
- Mixing unrelated data in one table