Relational Database Development

Relational Database Management Systems
Primary and Foreign Keys
Entity-Relationship Diagrams
Database Conventions
Business Requirements
Cardinality
Bridge Tables
Derived attributes
Strong and Weak Relationships

DAY 1

Contents

File-Based Data Systems

A classic example of a file-based data storage system is an Excel file, often used to store and organize a variety of data.

Consider this Excel sheet:

Banquet Information

banquetNo	supervisor	contact	guestCount	cost
346	Sally Jones	313-525-3322	200	$4,533.00
328	Mary Chen	218-533-5344	100	$3,522.00
455	Amir Beaudrie	613-535-3502	150	$720.00
423	Sally Jones	313-525-3322	50	$5,680.00
326	Peter Davies	218-488-3211	200	$2,300.00

Each row represents a unique banquet, detailing the event's `supervisor`, `contact`, `guest count` and `cost`. While this method of data storage is straightforward and user-friendly, it does have its drawbacks:

File-Based Data System Drawbacks

Data Redundancy:	Storing data in multiple files can lead to duplication and discrepancies, making it challenging to ensure data consistency.
Data Isolation:	When data is spread across different files, it can be difficult to retrieve and assemble this data in a coherent and efficient way.
Concurrent Anomalies:	In situations where multiple users need access to the data simultaneously, file-based systems can struggle to manage and coordinate this concurrent access effectively.
Security Issues:	File-based data systems typically lack robust security measures, making them vulnerable to unauthorized access, data corruption, or loss.
Limited Data Analysis:	File-based systems don't natively support complex querying, making it challenging to perform intricate data analysis.
Lack of Flexibility:	As business requirements evolve, it can be quite challenging to scale or modify the data structure in a file-based system.

However, there are more advanced, structured approaches, such as **Database Management Systems** (DBMS), which can mitigate many of these issues.

Quiz Questions 1 and 2

Relational Database Management Systems (RDBMS)

Relational Database Management Systems (RDBMS) have been developed to ensure efficient, accurate, and secure data storage. Currently, they are the go-to choice for managing transactional data. RDBMSs primarily uphold three principles: data security, data integrity, and referential integrity.

Data Security:	RDBMSs ensure data security through a robust framework of processes, rules, and standards designed to protect data from unauthorized access or corruption.
Data Integrity:	This pertains to the accuracy, completeness, and consistency of data within the database. RDBMSs implement rules and validation procedures to enforce data integrity, ensuring the data remains true to its intended form.
Referential Integrity:	This principle maintains the logical relationships between tables in a relational database. Each table must have a primary key, which may appear as a foreign key in other tables due to inter-table relationships. Referential integrity ensures that these relationships remain consistent and valid.

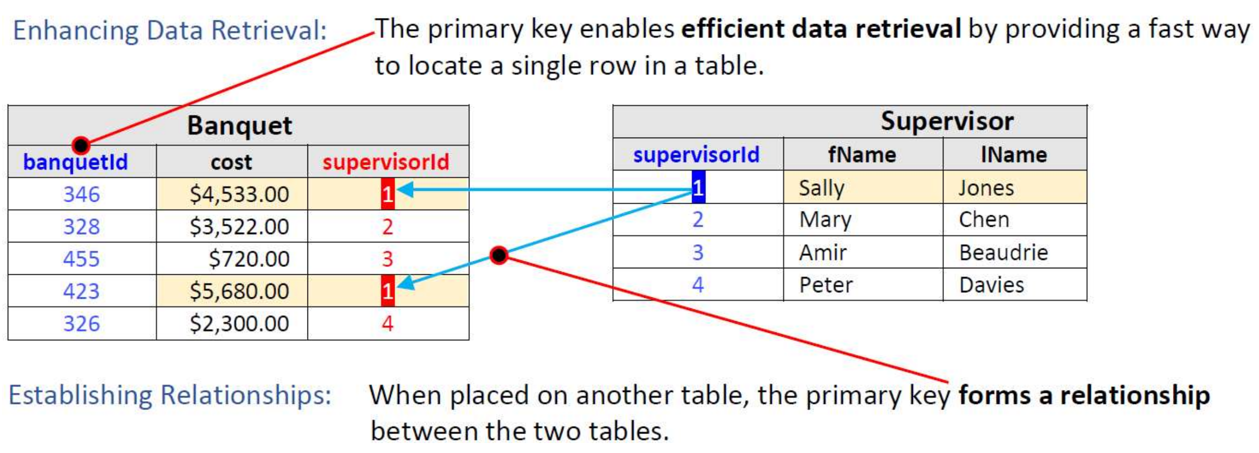
Understanding RDBMS Terminology

When dealing with Relational Database Management Systems (RDBMS), it's important to familiarize yourself with specific terms. Below are some crucial relational database terms and their definitions.

Table/Entity:	This refers to a logical grouping of data organized into rows and columns, much like a spreadsheet.
Column/Attribute:	A column in a table is known as an attribute, which holds specific data for an entity, such as a name, phone number, or serial number.
Row:	A row represents a single record in a table.
Relation/Association:	This refers to a relationship or association between different entities within the database.
NULL Values:	The term "NULL" signifies the absence of a value or data.
Primary Key:	Each table in a relational database requires a primary key, a unique and non-null identifier for each row.
Composite Primary Key:	When a single column isn't sufficient to guarantee a key’s uniqueness, two or more columns may be combined to create a composite primary key.
Foreign Key:	A column in one table that references the primary key of another, establishing the relationship. A table can have zero or more foreign keys. FK columns may be NULL otherwise each value must match a row in the referenced table.
Schema:	This is a visual representation of the entities and their relationships within the database, often represented as an Entity Relationship Diagram (ERD). A schema, which can be designed independently from the code, is often used to construct databases.

More on Primary Keys

In a RDBM System, keys play a pivotal role in maintaining data integrity and managing relationships between tables. Below are two significant roles the Primary Key serves:



Enhancing Data Retrieval: The primary key enables **efficient data retrieval** by providing a fast way to locate a single row in a table.

Banquet		
banquetId	cost	supervisorId
346	$4,533.00	1
328	$3,522.00	2
455	$720.00	3
423	$5,680.00	1
326	$2,300.00	4

Supervisor		
supervisorId	fName	lName
1	Sally	Jones
2	Mary	Chen
3	Amir	Beaudrie
4	Peter	Davies

Establishing Relationships: When placed on another table, the primary key **forms a relationship** between the two tables.

Primary Key Rules:

Uniqueness: Each table can have only one Primary or Composite Key. The values in this key must be unique, distinguishing each record in the table.

Non-nullability: The Primary or Composite Key cannot contain null values. Every record must have a valid, unique key value.

Minimality: Every column used to define the key must be necessary to ensure uniqueness, also known as **Minimal Form**.

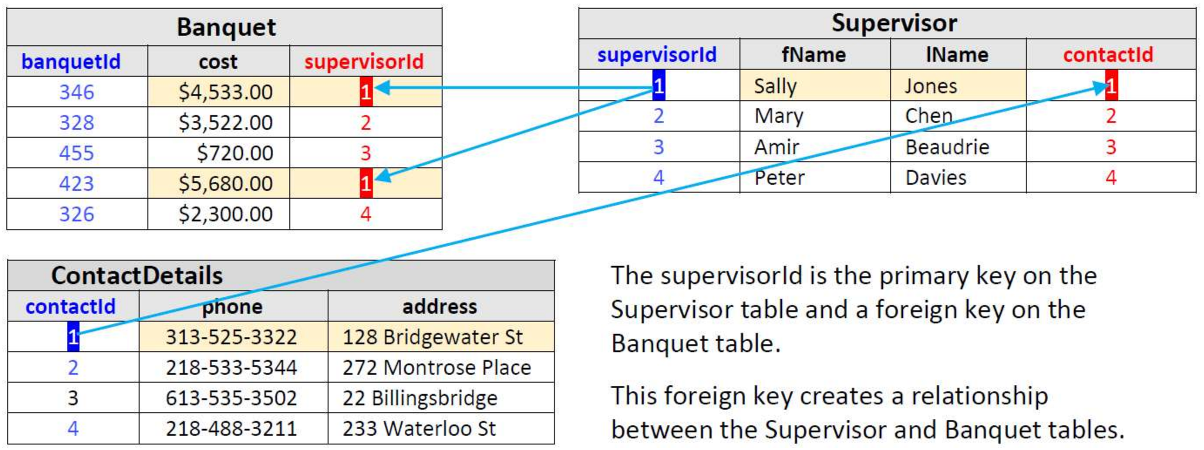
For example, every book in the library is uniquely identified by a composite key that consists of **AuthorName** + **Title**. This is sufficient to ensure uniqueness because no two books will have the same author and title.

Now, suppose we were to include **NumberOfPages** into the key, making it **AuthorName** + **Title** + **NumberOfPages**. This would be an unnecessary addition, and it wouldn't contribute to the uniqueness of the identification. This is because the number of pages in a book does not help differentiate one book from another with the same author and title.

Quiz Questions 3, 4, 5, 6 and 7

I need to reiterate the primary key concept again because it is fundamental to the structure and functioning of relational databases.

Each table is defined by a unique Primary Key, which serves as the identifier for individual records within the table. This Primary Key may also appear in other tables as a Foreign Key, establishing a relationship between the two tables.



Banquet

banquetId	cost	supervisorId
346	$4,533.00	1
328	$3,522.00	2
455	$720.00	3
423	$5,680.00	1
326	$2,300.00	4

Supervisor

supervisorId	fName	lName	contactId
1	Sally	Jones	1
2	Mary	Chen	2
3	Amir	Beaudrie	3
4	Peter	Davies	4

ContactDetails

contactId	phone	address
1	313-525-3322	128 Bridgewater St
2	218-533-5344	272 Montrose Place
3	613-535-3502	22 Billingsbridge
4	218-488-3211	233 Waterloo St

The supervisorId is the primary key on the Supervisor table and a foreign key on the Banquet table.

This foreign key creates a relationship between the Supervisor and Banquet tables.

Similarly, `addressId` is the primary key on the `ContactDetails` table and a foreign key on the Supervisor table, establishing a link between the `ContactDetails` and Supervisor tables.

Now the information for Sally Jones and her contact details is stored once, eliminating redundancy. An important consideration when dealing with large amounts of data. Even though the data in these tables might seem more than that in the original spreadsheet, the real efficiency of this approach becomes evident when the database scales to hold millions of records.

Quiz Questions 8, 9 and 10

Continuing with the banquet tables above, if you wanted to retrieve all banquet information for Sally Jones, you could do so with two simple requests.

The example below shows you what this might look like in plain English (or pseudo-code):

- Request: `Get me the Supervisor record for Sally Jones.`

  Response:

supervisorId	fName	lName	contactId
1	Sally	Jones	1

- Request: `Using Sally's supervisor ID get all her Banquet records.`

  Response:

banquetNo	cost	supervisorId
346	$4,533.00	1
423	$5,680.00	1

Now You Try:

Write the final request in *pseudo code* to return Sally's contact details from the banquet tables above.

Request:

Response:

supervisorId	fName	lName	contactId
1	Sally	Jones	1

Next place the pseudo code statement in the box below that will produce the following results.

Request:

Response:

contactId	phone	address
1	313-525-3322	128 Bridgewater St

Quiz Question 11

RDBMS Advantages and Disadvantages

RDBMSs offer advanced functionality for managing data, making them an invaluable tool in today's data-driven landscape. However, it's also essential to consider their drawbacks and decide if the benefits outweigh the potential complications for a given application.

Advantages:

- They provide robust security measures.
- They allow management of users and groups with varying data access needs.
- They offer dedicated software to create, view, and manage data.
- They enable efficient storage, backup, and restoration of data.

Disadvantages:

- Increased complexity compared to simpler file-based systems.
- Vendor and software dependence can limit flexibility and portability.
- There may be significant licensing, learning, and maintenance costs associated with implementing and running an RDBMS.

Prominent RDBMS Vendors

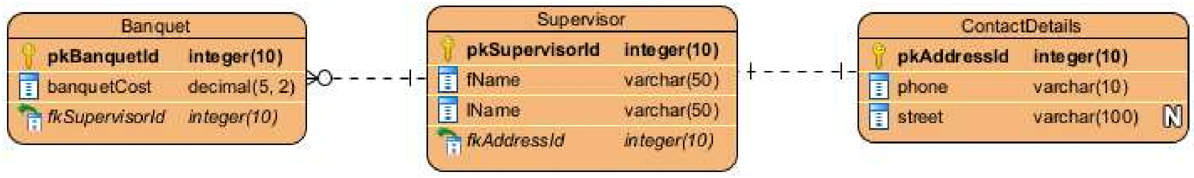
RDBMSs are widely used today, with the most common ones being Oracle, Microsoft SQL Server, MySQL, and PostgreSQL. Another option, SQLite, might not be as common, but is a lightweight and flexible database often used in mobile applications and will be a subject of focus in this course.

Quiz Question 12

Entity-Relationship Diagrams

Entity-Relationship Diagrams (ERD) provide a visual way to show how entities (tables) and their attributes (columns) are interconnected within a relational database system.

The ERD diagram below effectively illustrates entities, attributes, and the relationships required to organize the banquet data.



ERD Notation

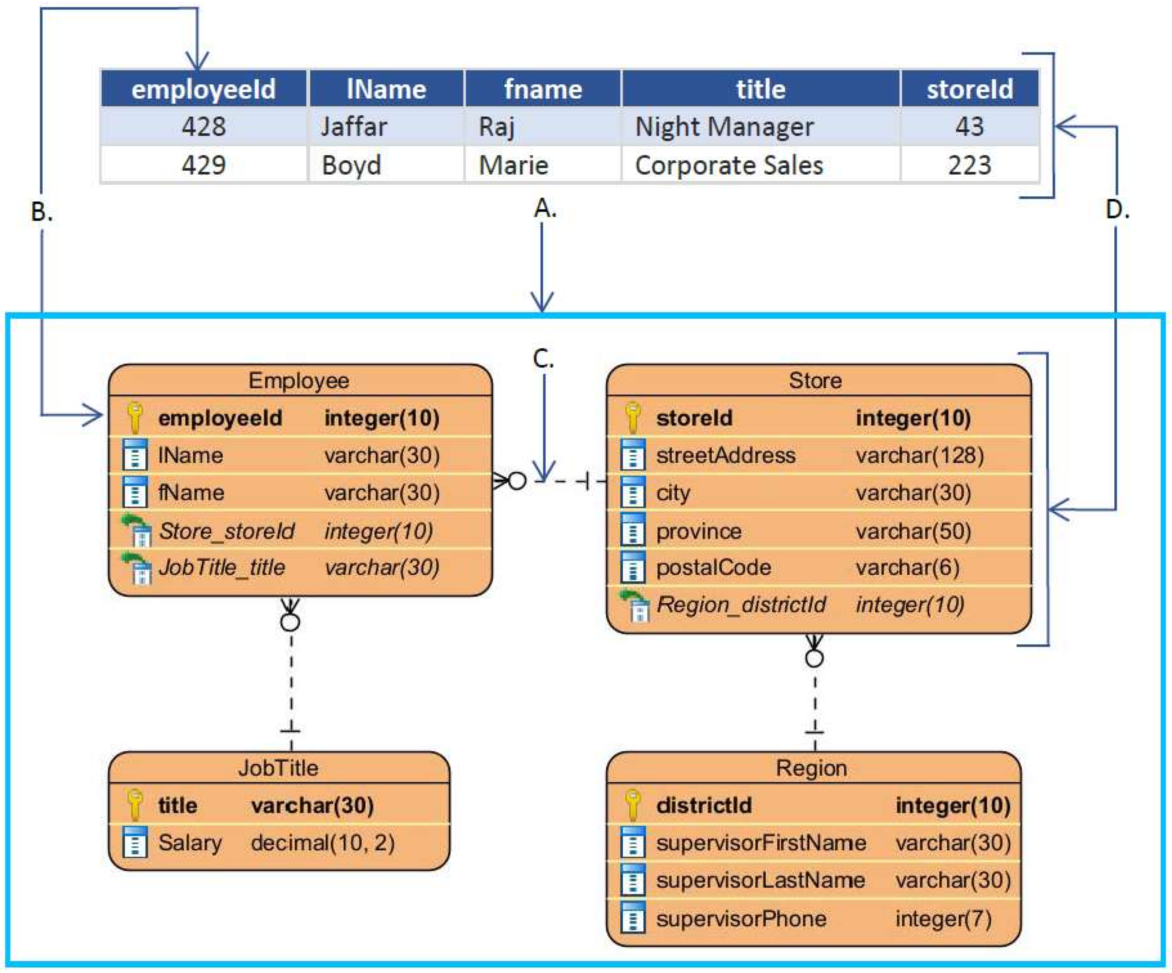
Understanding ERD notation is crucial for deciphering the diagram's meaning. Here are some common components and their naming conventions:

Entity Name:	Should be descriptive and in **PascalCase** (i.e., each word starts with a capital letter, and there are no spaces, like BanquetGuest.
Table Attribute:	Named in camelCase (i.e., the first word is all lowercase, and subsequent words start with a capital letter, with no spaces, like guestCount and should be a descriptive noun.
Primary Key:	Contains unique, non-null values serving as the identifier for records within a table.
Foreign Key:	Establishes a link to the primary key of another table, creating a relationship between tables. Foreign Keys are allowed to be null.
Entity Relationship:	Can range from zero to many, depending on the nature of the relationship between entities. More on this later in the course.
Attribute Data Types:	Defines the type and size of data that an attribute can store, such as varchar(10), integer(5), or decimal(2,0).
N:	A property indicating whether an attribute can contain null values.

No matter what application you choose for developing your database, be it SQL server, Oracle, or SQLite, the Entity Relationship Diagram (ERD) remains the same.

Quiz Questions 13, 14. 15 and 16

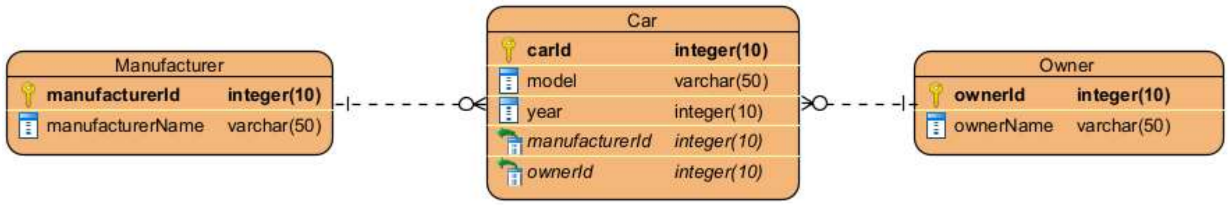
Diagram 1:



employeeId	lName	fname	title	storeId
428	Jaffar	Raj	Night Manager	43
429	Boyd	Marie	Corporate Sales	223

Quiz Question 17

Diagram 2:



Quiz Questions 18 and 19

Database Conventions

Different organizations and developers often adopt distinct convention styles for database programming. While these can vary significantly, it's crucial to select a consistent style and adhere to it throughout your project. In this course we will follow a specific approach regarding naming standards.

Marks will be deducted from exercises and assignments
when the following standards are not adhered to:

Naming

Adopting clear and informative naming standards improves readability and maintenance of your database schema. Here are our guidelines:

Global rules

Be consistent:	across the whole project.
Use meaningful nouns:	avoid cryptic abbreviations.
English only:	letters and digits (no underscores and spaces).
Avoid reserved words:	avoid User, Order. Add a suffix like AccountUser if needed.

Entities (tables)

PascalCase, singular:	CustomerAccount, SalesInvoice. (not Customers)
Naming:	Keep names short but specific (≈ 30 chars max).

Columns (attributes)

camelCase, singular:	salesDate.
Use clear suffixes that signal data type/intent:	
...Id:	for identifiers: customerId, invoiceId.
...Count:	for integers that are tallies: itemCount.
...Amount:	for currency: totalAmount.
is... /has...:	for booleans: isActive, hasWarranty.
Include units:	weightKg, lengthMm.
Store one fact per column:	“minimal form”.

Key columns (pk/fk prefix rule)

Primary keys:	prefix **pk**, then the identifier in camelCase, pkCustomerId, pkInvoiceId.
Foreign keys:	prefix **fk**, followed by the **referenced** Id, fkCustomerId, fkInvoiceId.

Normalization

In relational databases, normalization is the process of organizing data to reduce redundancy and improve data integrity. It divides larger tables into smaller, interconnected tables using primary and foreign keys, ensuring each piece of data is stored in only one place, optimizing storage, and enhancing consistency.

Minimal Form

The principle of minimal form focuses on maintaining a single value per cell in your table. This approach allows you to query your data easily.

In the `BanquetDetails` table below, addresses are consolidated into a single column. While this might seem efficient, it poses challenges when trying to extract specific information, such as all records from `Middleton`.

BanquetDetails				
pkBanquetNo	supervisor	phone	address	cost
346	Sally Jones	313-525-3322	28 Bridgewater St, Middleton, NS, N2S 4E5	$4,533.00

For ease of querying, using the principle of minimal form is beneficial here. This practice organizes the address into separate columns for street, city, province, and postcode.

The `ContactDetails` table below segregates each address component. This organization allows for more targeted queries, on street, city, province, or postcode.

ContactDetails					
pkAaddressId	phone	street	city	province	postcode
1	313-525-3322	128 Bridgewater St	Middleton	NS	N2S 4E5
2	218-533-5344	272 Montrose Place	Winnipeg	MB	M1N 3J3
3	613-535-3502	22 Billingsbridge	Middleton	NS	N1A 3E2
4	218-488-3211	233 Waterloo St	Winnipeg	MB	M2B 3V4

Now You Try:

The `vicePresidents` table below exhibits several faux pas in terms of adhering to our best practices.

vicePresidents		
VpId	HumanResources	fullNames
1	VP of Sales	Mr. Tom Walsh
2	VP of Marketing	Mrs. Wendy Jones
2	VP of Finance	Dr. William Strange
4	VP of Human Resources	Ms. Sandra Smith

Now You Try:

Consider the poorly created table above representing vice presidents in a company.

Identify and place in the boxes below, convention mistakes in the `vicePresidents` table.

1:	
2:	
3:	
4:	
5:	

Quiz Questions 20, 21 and 22

Business Requirements in DB Design

Database diagrams are visual representations of business rules, providing a non-technical medium to convey database concepts among designers, application developers, and end users.

Gathering Requirements

To accurately represent data, relationships, constraints, and business rules, database designers need to extract requirements from various sources. This often includes end-users, analysts, and developers. They scrutinize various forms and reports such as invoices, purchase orders, monthly summaries, etc. to gather necessary requirements for the data model.

Making Assumptions

Database design is iterative, and during the stages of requirement gathering and design, certain assumptions about the model might be necessary for maintaining data accuracy. To ensure that all project stakeholders understand these assumptions, they should be clearly documented within the data model.

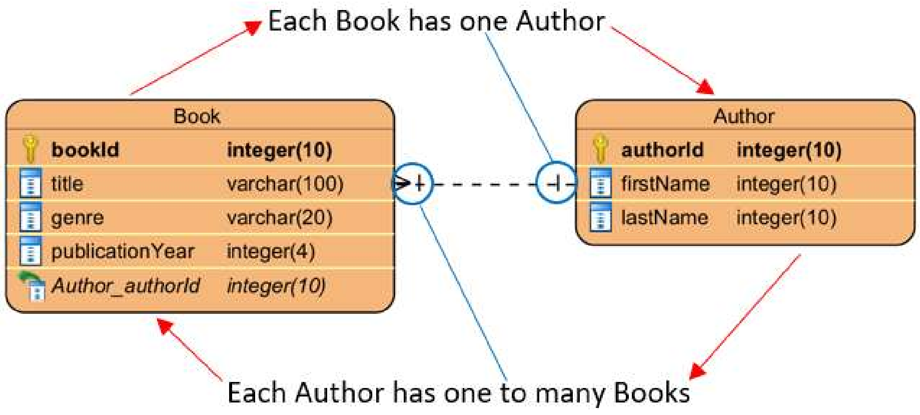
Cardinality

Cardinality describes the numerical relationship between rows in one table and rows in another. Common cardinalities include one-to-one, one-to-many, and many-to-many.

Cardinality Notation

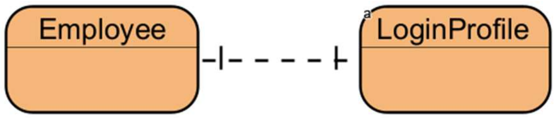
Relationships between tables are represented using Cardinality Notation or Crowfoot Notation. This notation illustrates one entity in relation to another.

In database relationships between tables, it's essential to view each connection in both directions. This bidirectional perspective offers a complete view of the data relationship.



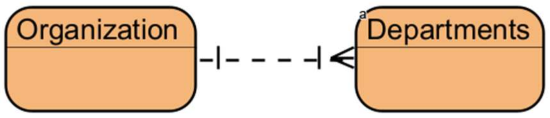
One to One

- Each employee has exactly one login profile.
- Each login profile has exactly one employee.



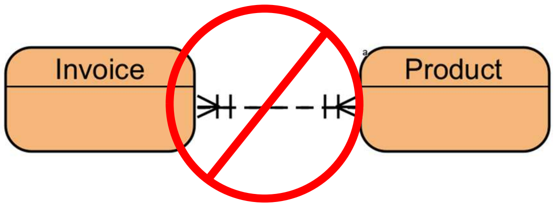
One to One or More

- Each organization has one or more departments.
- Each department has only one organization.



Many to Many

- A product can be on one or many invoices.
- An invoice can have one or many products.

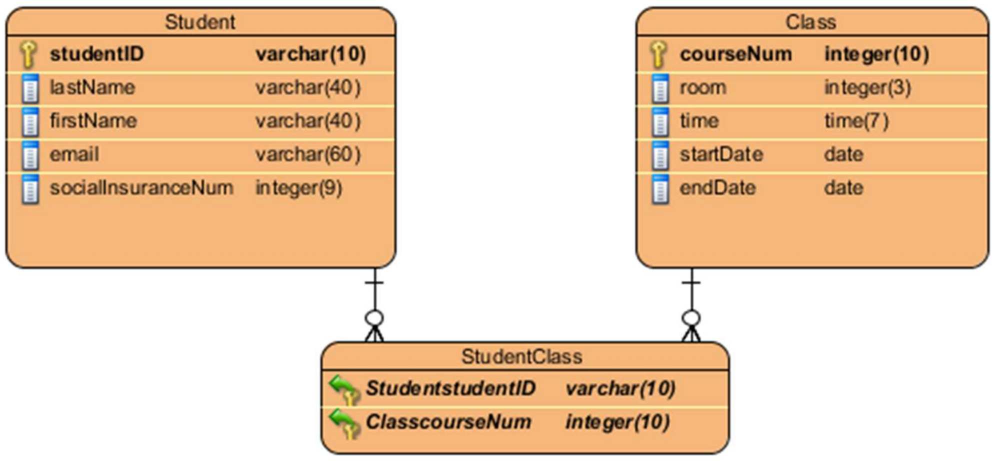


Note: It is crucial to note that many-to-many relationships are not permitted due to the complications that arise when relating rows in such a way. Instead, bridge tables are employed to avoid this problem.

Quiz Questions 23 and 24

Bridge Tables

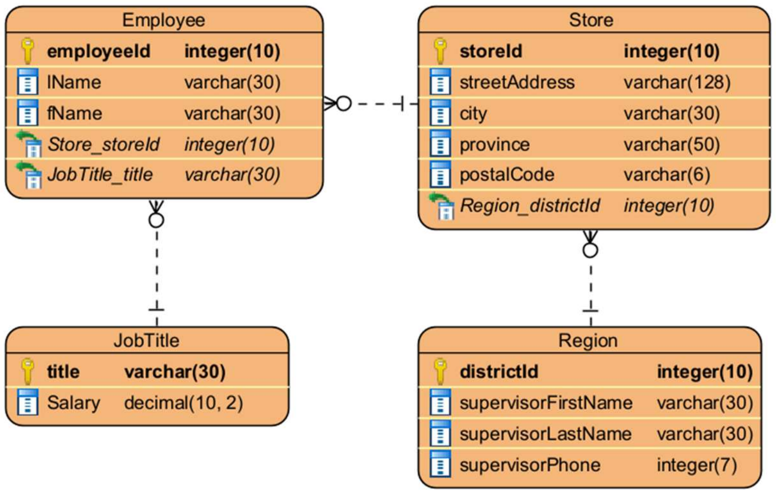
To resolve the issue of many-to-many relationships is to use a Bridge Table. For instance, as shown in the tables below, a student can enroll in multiple classes, and similarly, a class can have multiple students.



The primary keys from the Student and Class tables are used as a composite key in the bridge table.

Now You Try:

Use **Visual Paradigm** to re-recreate the ERD below.



Place a screenshot of your ERD in the box below:

Quiz Questions 25, 26, 27, and 28

Derived attributes

Derived attributes are unique in the sense that they do not physically exist within the database. Rather, their values are calculated based on other existing attributes within the same database. An example of a derived attribute could be the average salary in a company. Storing this attribute directly in the database would not be advisable as its value would fluctuate with every new hire, promotion, or employee exit, leading to constant updates and potential data inconsistency.

Advantages of Derived Attributes:

Always Current:	The value of a derived attribute is always current, as it is calculated from existing attributes each time it is accessed.
Enhanced Accuracy:	Derived attributes tend to have a lower likelihood of being incorrect, as their values are directly computed from other factual data within the database.

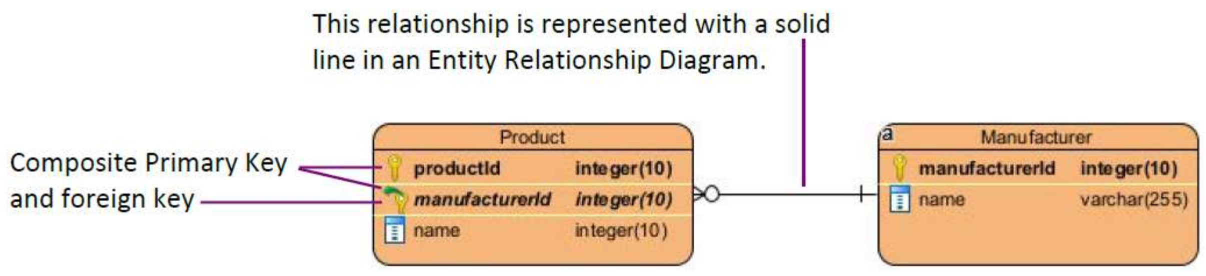
Disadvantages of Derived Attributes:

Resource Consumption:	Calculating the value of a derived attribute can consume significant computational resources, potentially impacting overall system performance.
Query Complexity:	Derived attributes can lead to the necessity for more complex queries, which may require more advanced database handling skills to manage effectively.

Quiz Questions 29 and 30

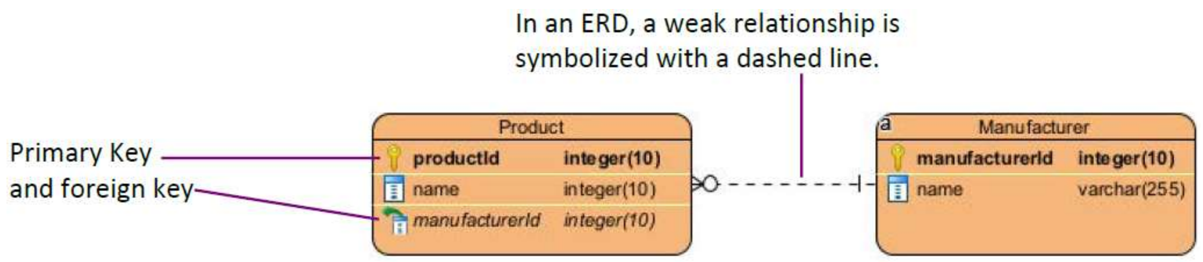
Strong Relationships

A strong relationship in a database is a connection where the primary key of one entity is incorporated within the primary key of the associated entity.


This relationship is represented with a solid line in an Entity Relationship Diagram.
Composite Primary Key and foreign key
Product
productId integer(10)
manufacturerId integer(10)
name integer(10)
Manufacturer
manufacturerId integer(10)
name varchar(255)

Weak Relationships

Contrastingly, a weak relationship is a connection where the primary key of one entity doesn't feature in the primary key of the other related entity. In a weak relationship, both entities can exist independently of each other, indicating less interdependency.


In an ERD, a weak relationship is symbolized with a dashed line.
Primary Key
and foreign key
Product
productId
integer(10)
name
integer(10)
manufacturerId
integer(10)
Manufacturer
manufacturerId
integer(10)
name
varchar(255)