Data Management Using Microsoft SQL Server

Session: 2

E-R Model and Normalization

Objectives

- Define and describe data modeling

- Identify and describe the components of the E-R model

- Identify the relationships that can be formed between entities

- Explain E-R diagrams and their uses

- Describe an E-R diagram, the symbols used for drawing, and show the various relationships

- Describe the various Normal Forms

- Outline the uses of different Relational Operators

➢ A data model is a group of conceptual tools that describes data, its relationships, and semantics.

➢ It also consists of the consistency constraints that the data adheres to.

➢ The Entity-Relationship, Relational, Network, and Hierarchical models are examples of data models.

➢ The development of every database begins with the basic step of analyzing its data in order to determine the data model that would best represent it.

➢ Once this step is completed, the data model is applied to the data.

Data Modeling 1-2

The process of applying an appropriate data model to the data, in order to organize and structure it, is called data modeling.

Data modeling can be broken down into three broad steps:

Conceptual Data Modeling

- The data modeler identifies the highest level of relationships in the data.

Logical Data Modeling

- The data modeler describes the data and its relationships in detail.
- The data modeler creates a logical model of the database.

Physical Data Modeling

- The data modeler specifies how the logical model is to be realized physically.

Data Modeling 2-2

➢ Following figure exhibits the various steps involved in data modeling.

Operational Database

| Physical Data Modeling |
| Logical Data Modeling |
| Conceptual Data Modeling |

Data Modeling

Business Information Requirements
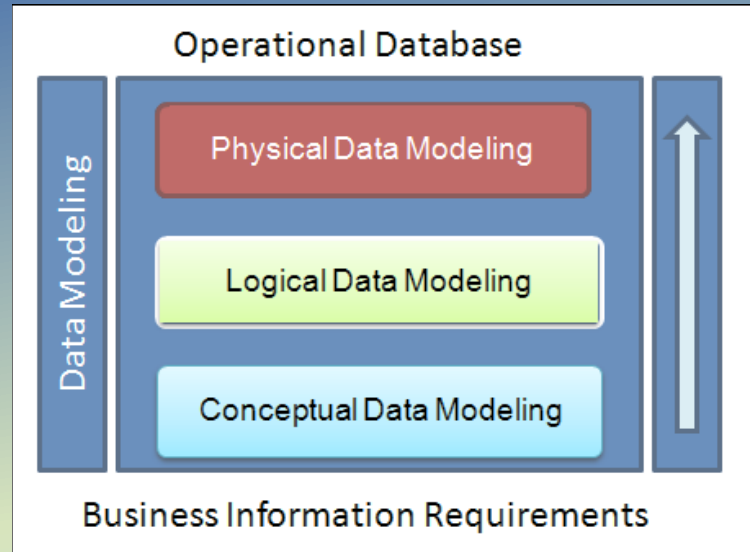
➢ Data models can be classified into three different groups:

Object-based logical models

Record-based logical models

Physical models
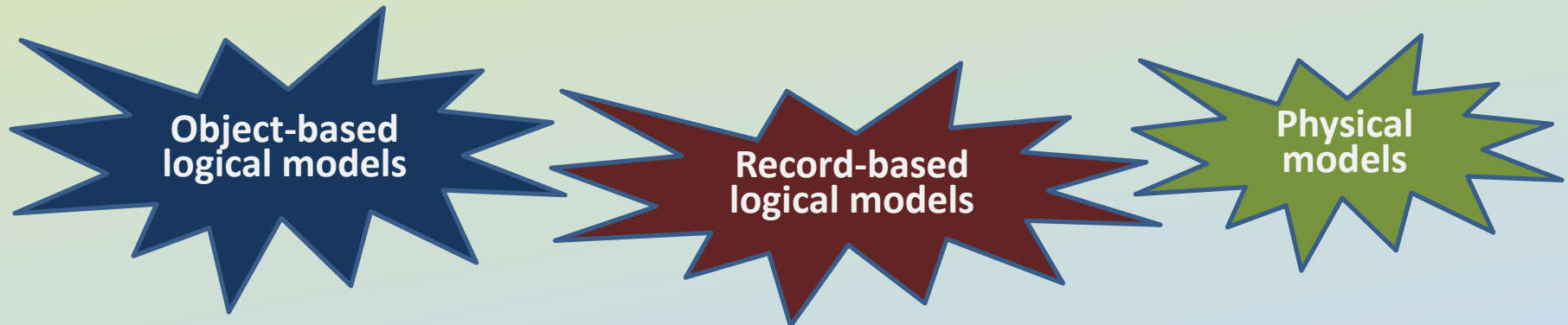
The Entity-Relationship (E-R) Model 1-11
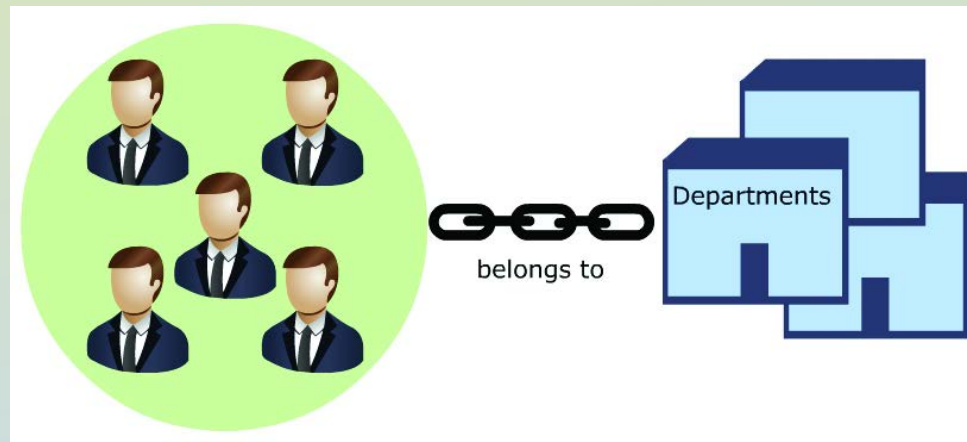
The Entity-Relationship (E-R) model belongs to the first classification.

Data can be perceived as real world objects called entities and the relationships that exist between them.

For example, in an organization, both employee and department are real world objects. An employee belongs to a department.

Thus, the relation 'belongs to' links an employee to a particular department. The employee-department relation can be modeled as shown in the following figure:

The Entity-Relationship (E-R) Model 2-11

➢ An E-R model consists of five basic components as follows:

Entity	• An entity is a real world object that exists physically and is distinguishable from other objects. • For example, employee, department, vehicle, and account.
Relationship	• A relationship is an association or bond that exists between one or more entities. • For example, belongs to, owns, works for, saves in, and so on.
Attributes	• Attributes are features that an entity has. Attributes help distinguish every entity from another. • For example, the attributes of a student would be roll_number, name, and so on.
Entity Set	• An entity set is the collection of similar entities. • For example, the employees of an organization collectively form an entity set called employee entity set.
Relationship Set	• A collection of similar relationships between two or more entity sets is called a relationship set. • For example, the set of all 'work in' relations that exists between the employees and the department is called the 'work in' relationship set.

➢ The various E-R model components can be seen in the following figure:

Employee Entity Set   **Department Entity Set**

Human Resource   Production

Accounts   Logistics

Relationship Set works in

Employee Entities

Department Entities
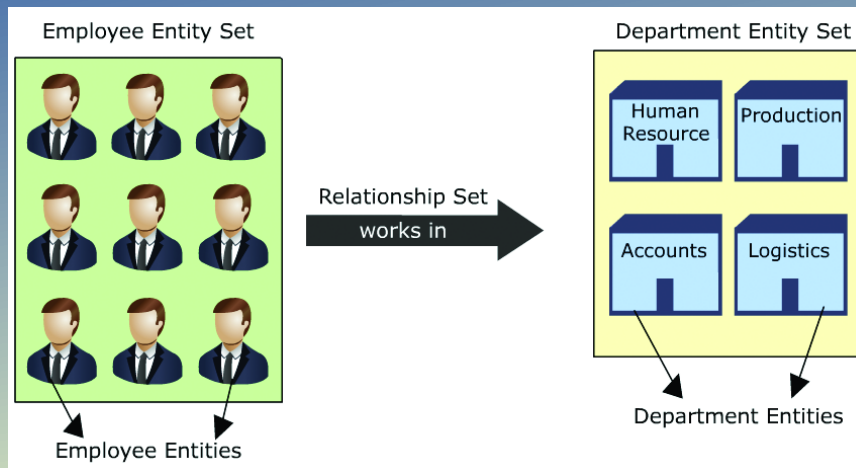
➢ Relationships associate one or more entities and can be of three types as follows:

Self-relationships

➢ Relationships between entities of the same entity set are called self-relationships.

➢ For example, a team member works for the manager.

➢ The relation, 'works for', exists between two different employee entities of the same employee entity set.

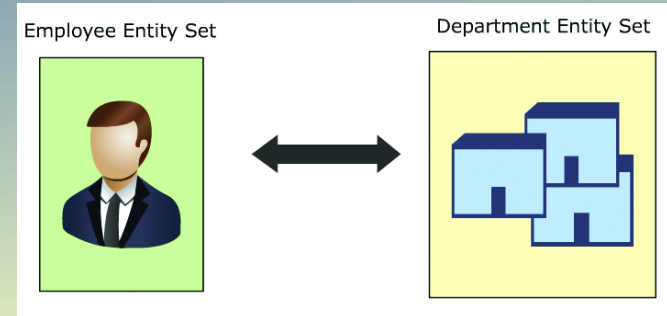➢ The relationship can be seen in following figure:
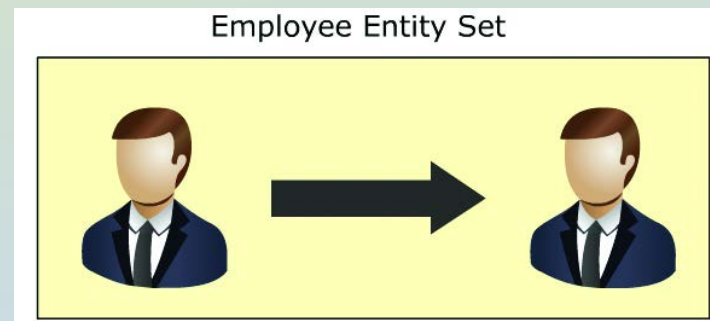
Employee Entity Set

Binary Relationships

➢ Relationships that exist between entities of two different entity sets are called binary relationships.

➢ For example, an employee belongs to a department.

➢ The employee entity belongs to an employee entity set. The department entity belongs to a department entity set.

Ternary Relationships

➢ Relationships that exist between three entities of different entity sets are called ternary relationships.

➢ For example, an employee works in the accounts department at the regional branch.

➢ The relation, 'works' exists between all three, the employee, the department, and the location.

➢ The relationship can be seen in following figure:

➢ The relationship can be seen in following figure:

SQL Server 2012
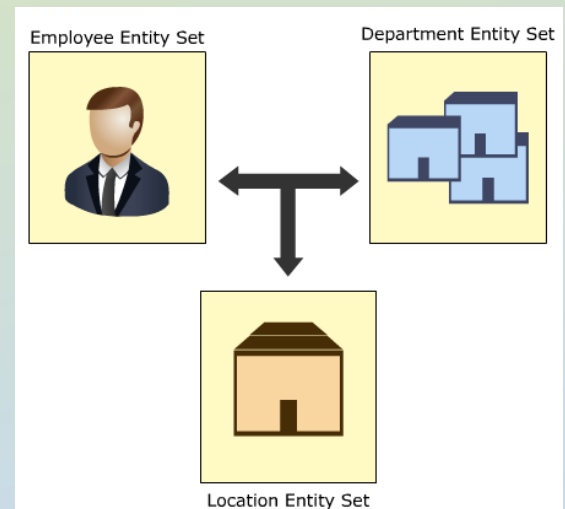
➤ Relationships can also be classified as per mapping cardinalities as follows:

One-to-One

➤ This kind of mapping exists when an entity of one entity set can be associated with only one entity of another set.
➤ For example, every vehicle has a unique registration.
➤ No two vehicles can have the same registration details.
➤ The relation is one-to-one, that is, one vehicle-one registration.
➤ The mapping cardinality can be seen in the following figure:
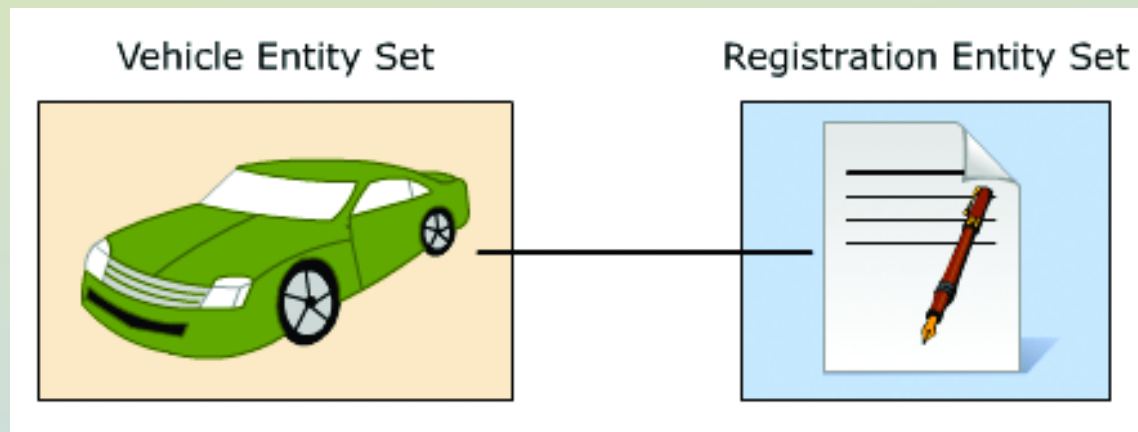
One-to-Many

➢ This kind of mapping exists when an entity of one set can be associated with more than one entity of another entity set.

➢ For example, a customer can have more than one vehicle.

➢ Therefore, the mapping is a one to many mapping, that is, one customer - one or more vehicles.

➢ The mapping cardinality can be seen in the following figure:

Vehicle Entity Set

Customer Entity Set
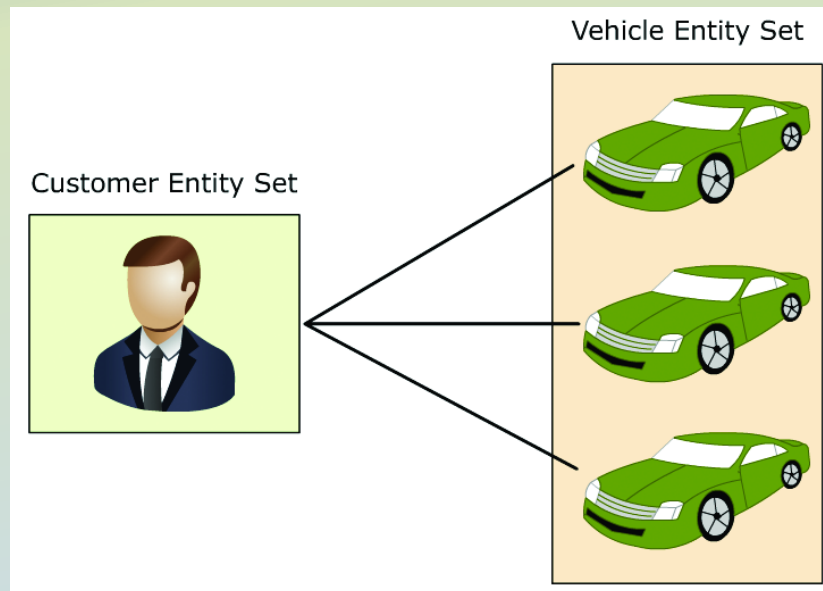
Many-to-One

➤ This kind of mapping exists when many entities of one set is associated with an entity of another set.

➤ This association is done irrespective of whether the latter entity is already associated to other or more entities of the former entity set.

➤ For example, every vehicle has only one manufacturing company but the same company or coalition can manufacture more than one kind of vehicle.

➤ The mapping can be seen in the following figure:
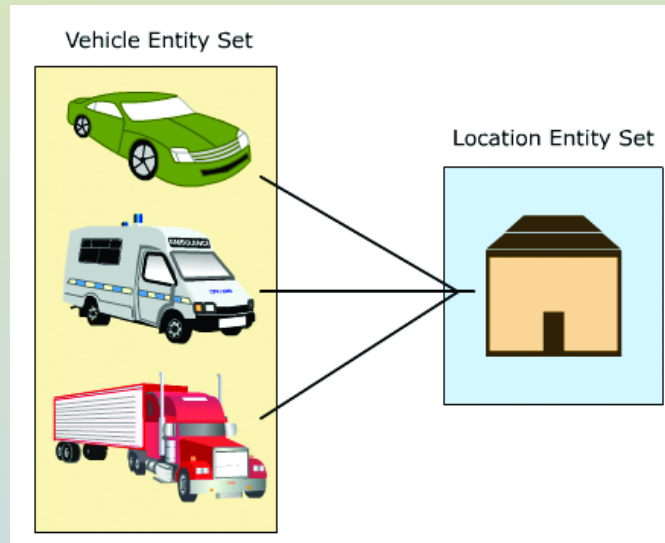
Many-to-Many

- ➤ This kind of mapping exists when any number of entities of one set can be associated with any number of entities of the other entity set.
- ➤ For example, customer can have more than one account and an account can have more than one customer associated with it in case it is a joint account or similar.
- ➤ Therefore, the mapping is many-to-many, that is, one or more customers associated with one or more accounts.
- ➤ The mapping cardinality can be seen in the following figure:
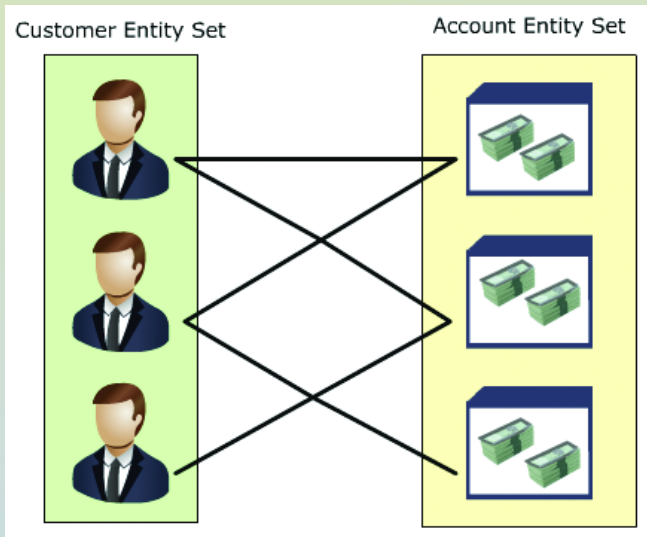
➢ Some additional concepts in the E-R model are as follows:

Primary Keys

➢ A primary key is an attribute that can uniquely define an entity in an entity set.
➢ The following table contains the details of students in a school.

Enrollment_number	Name	Grade	Division
786	Ashley	Seven	B
957	Joseph	Five	A
1011	Kelly	One	A

Student Details

➢ Every student has a unique enrollment number (such as `enrollment_number`), which is unique to the student.

➢ Any student can be identified based on the enrollment number.

➢ Thus, the attribute `enrollment_number` plays the role of the primary key in the `Student Details` table.

SQL
Server
2012

Weak Entity Sets

➢ Entity sets that do not have enough attributes to establish a primary key are called weak entity sets.

Strong Entity Sets

➢ Entity sets that have enough attributes to establish a primary key are called strong entity sets.

➢ Consider the scenario of an educational institution where at the end of each semester, students are required to complete and submit a set of assignments.

➢ The teacher keeps track of the assignments submitted by the students.

➢ An assignment and a student can be considered as two separate entities.

➢ The assignment entity is described by the attributes `assignment_number` and `subject`.

➢ The student entity is described by `roll_number`, `name`, and `semester`.

➢ The assignment entities can be grouped to form an assignment entity set and the student entities can be grouped to form a student entity set.

➢ The entity sets are associated by the relation 'submitted by'.
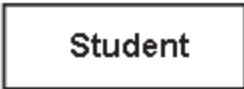
➢ The relationship is depicted in the following figure :

Attributes
- assignment_number
- subject

Assignment Entity Set

Attributes
- roll_number
- name
- semester

Student Entity Set

Submitted by
Relationship Set
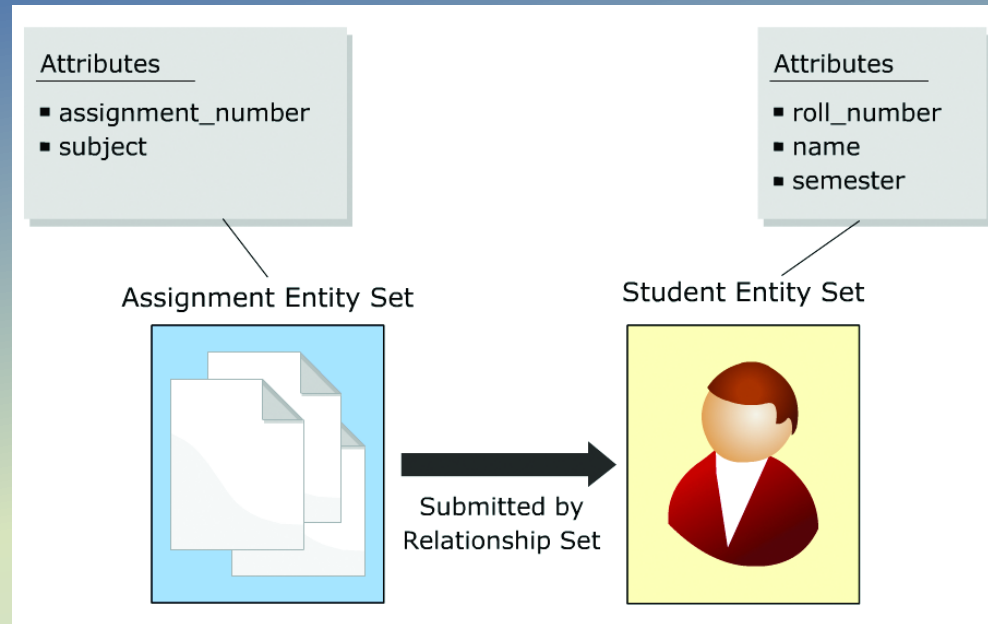
➢ The attributes, **`assignment_number`** and **`subject`**, are not enough to identify an assignment entity uniquely.

➢ The **`roll_number`** attribute alone is enough to uniquely identify any student entity. Therefore, **`roll_number`** is a primary key for the student entity set.

➢ The assignment entity set is a weak entity set since it lacks a primary key.

➢ The student entity set is a strong entity set due to the presence of the **`roll_number`** attribute.

Entity-Relationship Diagrams 1-7

➢ The E-R diagram is a graphical representation of the E-R model.
➢ The E-R diagram, with the help of various symbols, effectively represents various components of the E-R model.
➢ The symbols used for various components can be seen in the following table:

Component	Symbol	Example
Entity	Entity	Student
Weak Entity	Weak Entity	Assignments
Attribute	Attribute	Roll_num
Relationship	Relationship	Saves in
Key Attribute	Attribute	Acct_num

➢ Attributes in the E-R model can be further classified as:

Multi-valued

➢ A multi-valued attribute is illustrated with a double-line ellipse, which has more than one value for at least one instance of its entity.
➢ This attribute may have upper and lower bounds specified for any individual entity value.
➢ The telephone attribute of an individual may have one or more values, that is, an individual can have one or more telephone numbers.
➢ Hence, the telephone attribute is a multi-valued attribute.
➢ The symbol and example of a multi-valued attribute can be seen in the following figure:
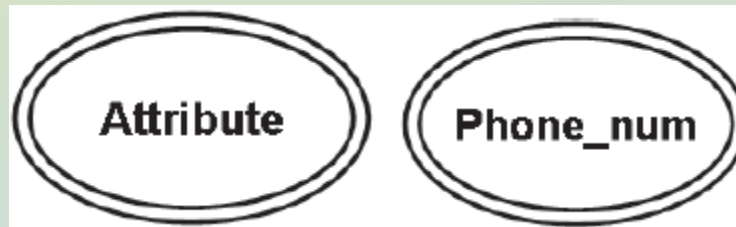
Entity-Relationship Diagrams 3-7

Composite

➢ A composite attribute may itself contain two or more attributes, which represent basic attributes having independent meanings of their own.

➢ The address attribute is usually a composite attribute, composed of attributes such as street, area, and so on.

➢ The symbol and example of a composite attribute can be seen in the following figure:

SQL Server 2012
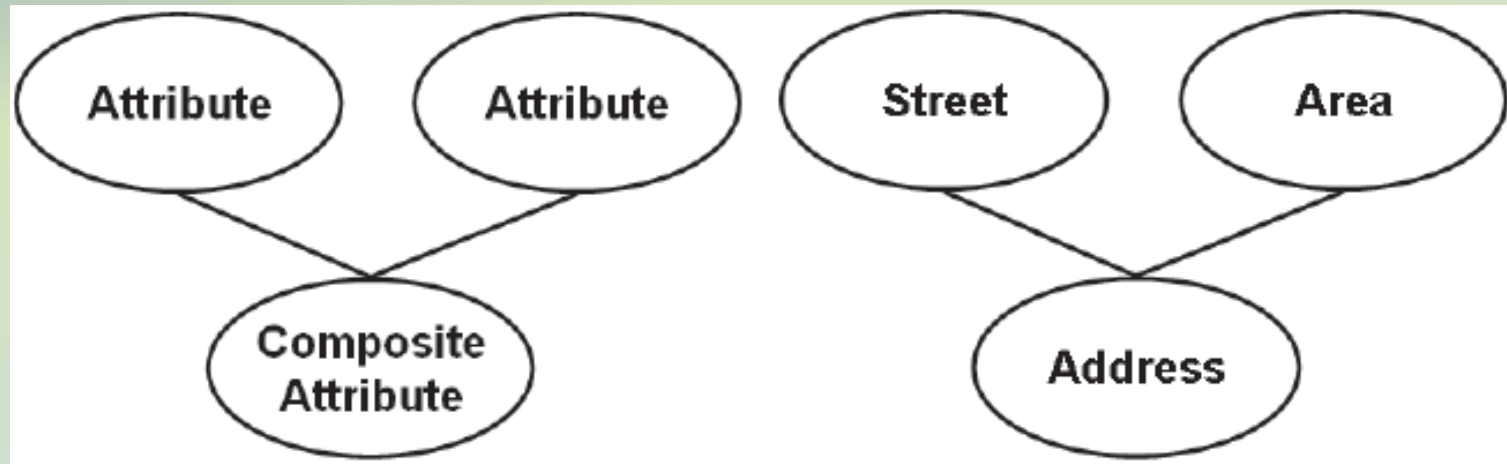
Derived

➤ Derived attributes are attributes whose value is entirely dependent on another attribute and are indicated by dashed ellipses.

➤ The age attribute of a person is the best example for derived attributes.

➤ For a particular person entity, the age of a person can be determined from the current date and the person's birth date.

➤ The symbol and example of a derived attribute can be seen in the following figure:
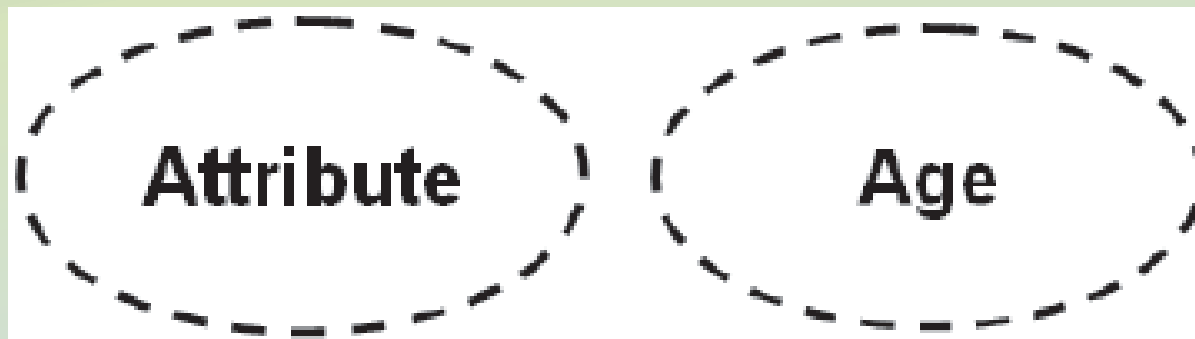
Entity-Relationship Diagrams 5-7

➢ Steps to construct an E-R diagram are as follows:

1 • Gather all the data that needs to be modeled.

2 • Identify data that can be modeled as real world entities.

3 • Identify the attributes for each entity.

4 • Sort entity sets as weak or strong entity sets.

5 • Sort entity attributes as key attributes, multi-valued attributes, composite attributes, derived attributes, and so on.

6 • Identify the relations between the different entities.
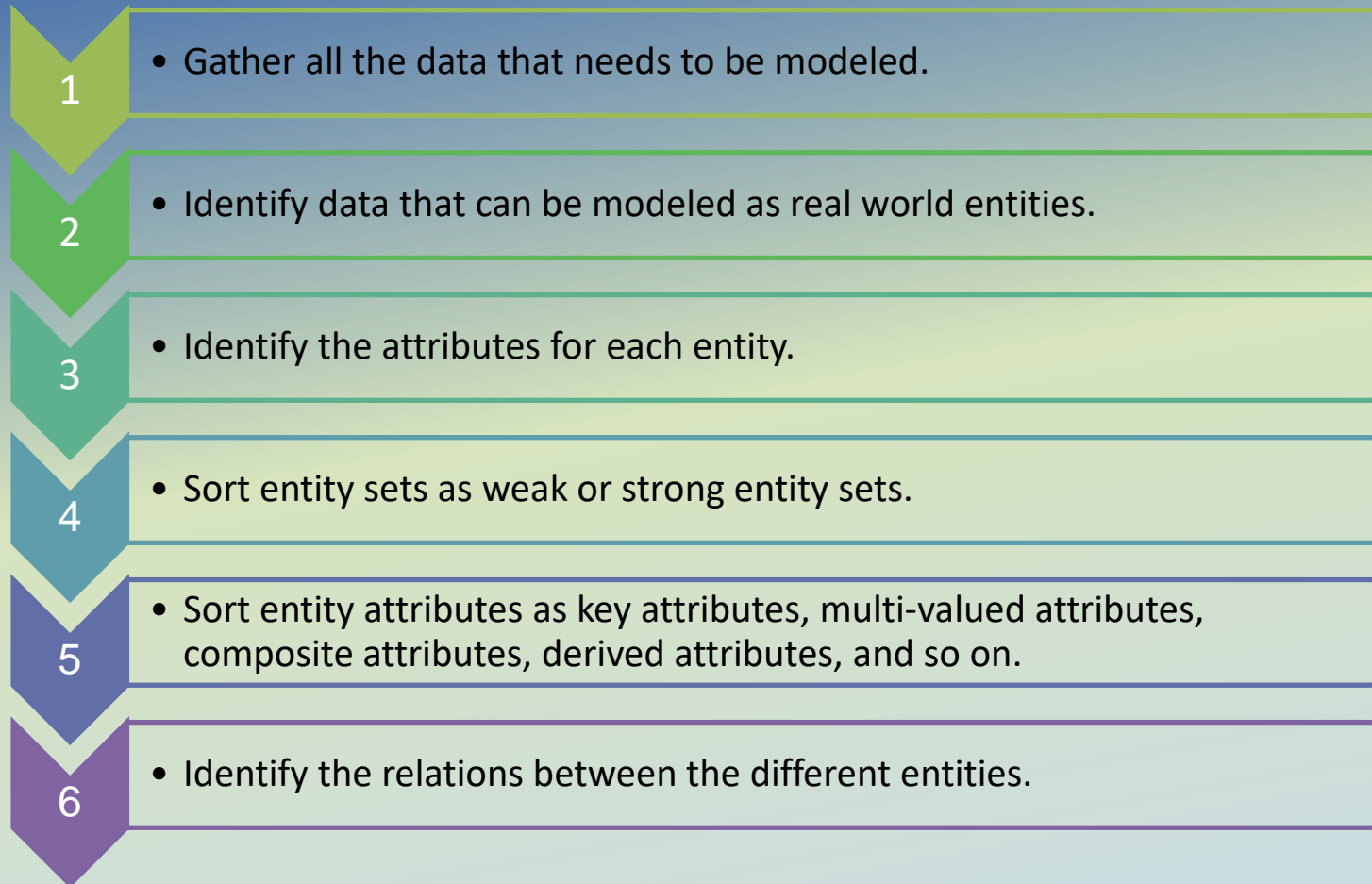
Entity-Relationship Diagrams 6-7

➢ Consider the scenario of a bank, with customers and accounts. The E-R diagram for the scenario can be constructed as follows:

Step 1: Gather data	• The bank is a collection of accounts used by customers to save money.
Step 2: Identify entities	• Customer • Account
Step 3: Identify the attributes	• Customer: customer_name, customer_address, customer_contact • Account: account_number, account_owner, balance_amount
Step 4: Sort entity sets	• Customer entity set: weak entity set • Account entity set: strong entity set
Step 5: Sort attributes	• Customer entity set: customer_address - composite, customer_contact - multi-valued • Account entity set: account_number → primary key, account_owner – multi-valued
Step 6: Identify relations	• A customer 'saves in' an account. The relation is 'saves in'.

SQL Server 2012

Step 7: Draw diagram using symbols

- The E-R diagram is shown in the following figure:
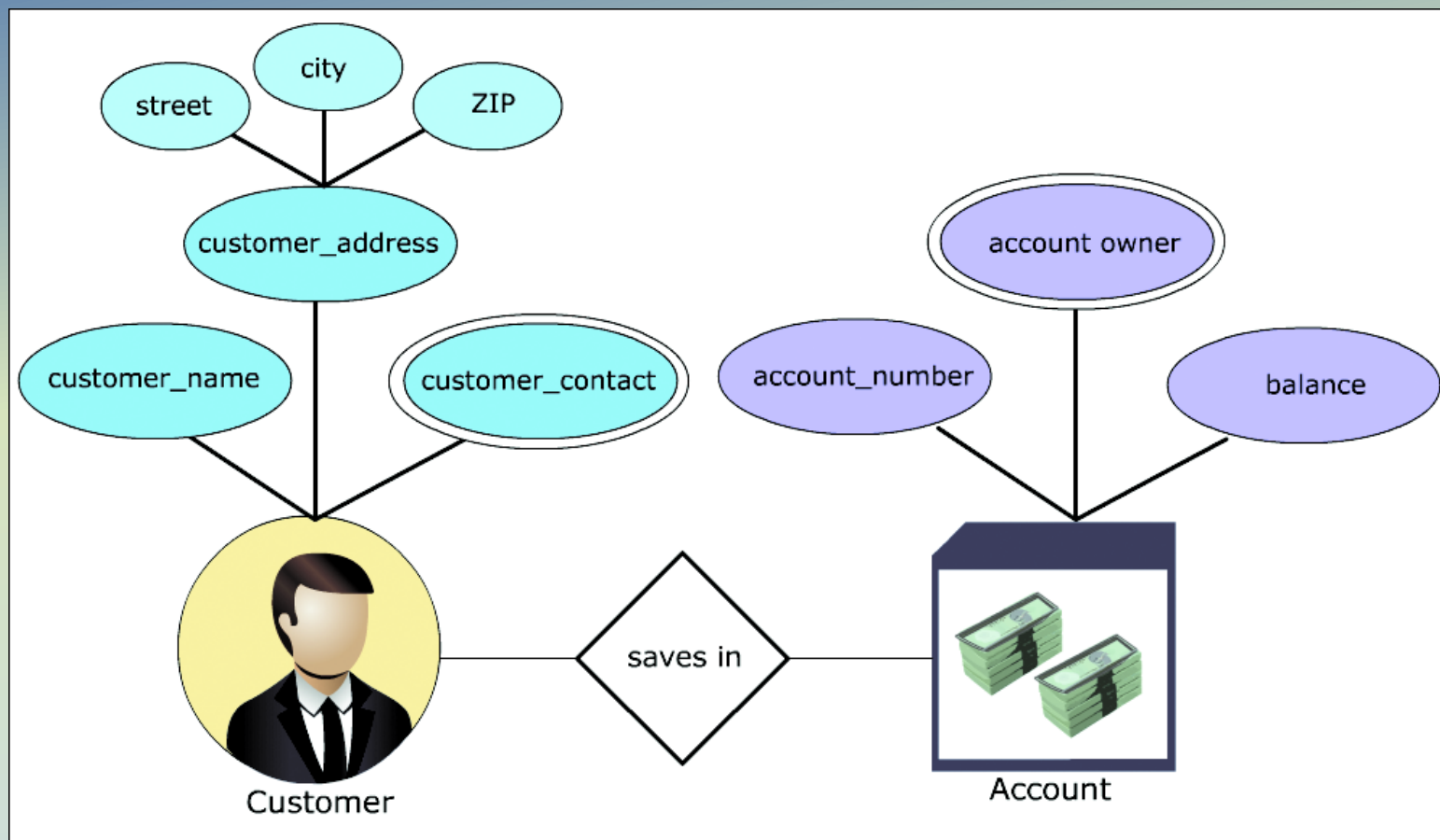
Normalization 1-4

- Initially, all databases are characterized by large number of columns and records.
- This approach has certain drawbacks.
- The following table consist of details of the employees and the project they are working on.

Emp_no	Project_id	Project_name	Emp_name	Grade	Salary
142	113, 124	BLUE STAR, MAGNUM	John	A	20,000
168	113	BLUE STAR	James	B	15,000
263	113	BLUE STAR	Andrew	C	10,000
109	124	MAGNUM	Bob	C	10,000

Repetition Anomaly

- The data such as `Project_id`, `Project_name`, `Grade`, and `Salary` repeat many times.
- This repetition hampers both, performance during retrieval of data and the storage capacity.
- This repetition of data is called the repetition anomaly.

Normalization 2-4

➢ The repetition is shown in the following table with the help of shaded cells:

Emp_no	Project_id	Project_name	Emp_name	Grade	Salary
142	113, 124	BLUE STAR, MAGNUM	John	A	20,000
168	113	BLUE STAR	James	B	15,000
263	113	BLUE STAR	Andrew	C	10,000
109	124	MAGNUM	Bob	C	10,000

Insertion Anomaly

➢ Suppose the department recruits a new employee named **Ann**.
➢ Consider that **Ann** has not been assigned any project. Insertion of her details in the table would leave columns **Project_id** and **Project_name** empty.
➢ Leaving columns blank could lead to problems later.
➢ Anomalies created by such insertions are called insertion anomalies as shown in the following table:

Emp_no	Project_id	Project_name	Emp_name	Grade	Salary
142	113, 124	BLUE STAR, MAGNUM	John	A	20,000
168	113	BLUE STAR	James	B	15,000
263	113	BLUE STAR	Andrew	C	10,000
109	124	MAGNUM	Bob	C	10,000
195	-	-	Ann	C	10,000

Normalization 3-4

Deletion Anomaly

➤ Suppose, Bob is relieved from the project MAGNUM.

➤ Deleting the record deletes Bob's **Emp_no**, **Grade**, and **Salary** details too.

➤ This loss of data is harmful as all of Bob's personal details are also lost.

➤ This kind of loss of data due to deletion is called deletion anomaly as can be seen in the following table:

Emp_no	Project_id	Project_name	Emp_name	Grade	Salary
142	113, 124	BLUE STAR, MAGNUM	John Smith	A	20,000
168	113	BLUE STAR	James Kilber	B	15,000
263	113	BLUE STAR	Andrew Murray	C	10,000

Updating Anomaly

➤ Suppose John was given a hike in **Salary** or John was demoted.

➤ The change in John's **Salary** or **Grade** needs to be reflected in all projects John works for.

➤ This problem in updating all the occurrences is called updating anomaly.

Normalization 4-4

The `Department Employee Details` table is called an unnormalized table. These drawbacks lead to the need for normalization.

Normalization is the process of removing unwanted redundancy and dependencies.

Initially, Codd (1972) presented three normal forms (1NF, 2NF, and 3NF), all based on dependencies among the attributes of a relation.

The fourth and fifth normal forms are based on multi value and join dependencies and were proposed later.

First Normal Form 1-2

➢ In order to achieve the first normal form, following steps need to be performed:

1	• Create separate tables for each group of related data.
2	• The table columns must have atomic values.
3	• All the key attributes must be identified.

➢ Consider the **Employee Project Details** table as follows:

Emp_no	Project_id	Project_name	Emp_name	Grade	Salary
142	113, 124	BLUE STAR, MAGNUM	John	A	20,000
168	113	BLUE STAR	James	B	15,000
263	113	BLUE STAR	Andrew	C	10,000
109	124	MAGNUM	Bob	C	10,000

➢ The table has data related to projects and employees.
➢ The table needs to be split into two tables, that is, a **Project Details** table and an **Employee Details** table.
➢ The table columns, **Project_id** and **Project_names**, have multiple values.

First Normal Form 2-2

➢ The data needs to be split over different rows.
➢ The resultant tables are **Project Details** and **Employee Details** as follows:

Project_id	Project_name
113	BLUE STAR
124	MAGNUM

Project Details

Emp_no	Emp_name	Grade	Salary
142	John	A	20,000
168	James	B	15,000
263	Andrew	C	10,000
109	Bob	C	10,000

Employee Details

➢ The **Project_id** attribute is the primary key for the **Project Details** table.
➢ The **Emp_no** attribute is the primary key for the **Employee Details** table.
➢ Therefore, in first normal form, the initial **Employee Project Details** table has been reduced to the **Project Details** and **Employee Details** tables.

Second Normal Form 1-2

➢ The tables are said to be in second normal form if:

> **They meet the requirements of the first normal form.**

> **There are no partial dependencies in the tables.**

> **The tables are related through foreign keys.**

➢ Partial dependency means a non-key attribute should not be partially dependent on more than one key attribute.

➢ The **Project Details** and **Employee Details** tables do not exhibit any partial dependencies.

➢ The **Project_name** is dependent only on **Project_id** and **Emp_name**, **Grade**, and **Salary** are dependant only on **Emp_no**.

➢ The tables also need to be related through foreign keys.

➢ A third table, named **Employee Project Details**, is created with only two columns, **Project_id** and **Emp_no**.

Second Normal Form 2-2

➢ So, the project and employee details tables on conversion to second normal form generates tables **Project Details**, **Employee Details**, and **Employee Project Details** as follows:

Project_id	Project_name
113	BLUE STAR
124	MAGNUM

Project Details

Emp_no	Emp_name	Grade	Salary
142	John	A	20,000
168	James	B	15,000
263	Andrew	C	10,000
109	Bob	C	10,000

Employee Details

Emp_no	Project_id
142	113
142	124
168	113
263	113
109	124

Employee Project Details

➢ The attributes, **Emp_no** and **Project_id**, of the **Employee Project Details** table combine together to form the primary key.
➢ Such primary keys are called composite primary keys.

Third Normal Form 1-3

➢ To achieve the third normal form:

> **The tables should meet the requirements of the second normal form**

> **The tables should not have transitive dependencies in them**

➢ The `Project Details`, `Employee Details`, and `Employee Project Details` tables are in second normal form.

➢ If an attribute can be determined by another non-key attribute, it is called a transitive dependency.

➢ That is, every non-key attribute should be determined by the key attribute only.

➢ If a non-key attribute can be determined by another non-key attribute, it needs to put into another table.

Third Normal Form 2-3

➢ On observing the different tables, it is seen that the **Project Details** and **Employee Project Details** tables do not exhibit any such transitive dependencies.

➢ The non-key attributes are totally determined by the key attributes.

➢ **Project_name** is only determined by **Project_number**.

➢ On further scrutinizing the **Employee Details** table, a certain inconsistency is seen.

➢ The attribute **Salary** is determined by the attribute **Grade** and not the key attribute **Emp_no**.

➢ Thus, this transitive dependency needs to be removed.

➢ The **Employee Details** table can be split into **Employee Details** and **Grade Salary Details** tables as follows:

Emp_no	Emp_name	Grade
142	John	A
168	James	B
263	Andrew	C
109	Bob	C

Employee Details

Grade	Salary
A	20,000
B	15,000
C	10,000

Grade Salary Details

Third Normal Form 3-3

➢ Thus, at the end of the three normalization stages, the initial **Employee Project Details** table has been reduced to the **Project Details**, **Employee Project Details**, **Employee Details**, and **Grade Salary Details** tables as follows:

Project_id	Project_name
113	BLUE STAR
124	MAGNUM

Project Details

Emp_no	Project_id
142	113
142	124
168	113
263	113
109	124

Employee Project Details

Emp_no	Emp_name	Grade
142	John	A
168	James	B
263	Andrew	C
109	Bob	C

Employee Details

Grade	Salary
A	20,000
B	15,000
C	10,000

Grade Salary Details

Denormalization

By normalizing a database, redundancy is reduced.

This, in turn, reduces the storage requirements for the database and ensures data integrity.

However, it has following drawbacks:

Complex join queries may have to be written often to combine the data in multiple tables.

Joins may practically involve more than three tables depending on the need for information.

➤ If such joins are used very often, the performance of the database will become very poor.

➤ In such cases, storing a few fields redundantly can be ignored to increase the performance of the database.

➤ The databases that possess such minor redundancies in order to increase performance are called denormalized databases and the process of doing so is called denormalization.

Relational Operators

➢ The relational model is based on the solid foundation of Relational Algebra.

➢ Relational Algebra consists of a collection of operators that operate on relations.

➢ Each operator takes one or two relations as its input and produces a new relation as its output.

➢ Consider the `Branch Reserve Details` as shown in the following table.

Branch	Branch_id	Reserve (Billion €)
London	BS-01	9.2
London	BS-02	10
Paris	BS-03	15
Los Angeles	BS-04	50
Washington	BS-05	30

Branch Reserve Details

SELECT

- The SELECT operator is used to extract data that satisfies a given condition.
- The lowercase Greek letter sigma, 'σ', is used to denote selection.
- A select operation, on the **Branch Reserve Details** table, to display the details of the branches in London would result in the following table:

Branch	Branch_id	Reserve (Billion €)
London	BS-01	9.2
London	BS-02	10

Details of Branches in London

- A selection on the **Branch Reserve Details** table to display branches with reserve greater than 20 billion Euros would result in the following table:

Branch	Branch_id	Reserve (Billion €)
Los Angeles	BS-04	50
Washington	BS-05	30

Details of Branches with Reserves Greater than 20 Billion Euros

PROJECT

- The PROJECT operator is used to project certain details of a relational table.
- The PROJECT operator only displays the required details leaving out certain columns.
- The PROJECT operator is denoted by the Greek letter pi, 'E'.
- Assume that only the Branch_id and Reserve amounts need to be displayed.
- A project operation to do the same, on the **Branch Reserve Details** table, would result in the following table:

Branch	Branch_id	Reserve (Billion €)
London	BS-01	9.2
London	BS-02	10
Paris	BS-03	15
Los Angeles	BS-04	50
Washington	BS-05	30

Resultant Table With Branch_id And Reserve Amounts

➢ The PRODUCT operator, denoted by 'x' combines each record from the first table with all the records in the second table, thereby, generating all possible combinations between the table records. Consider the following table:

Branch_id	Loan Amount (Billion €)
BS-01	0.56
BS-02	0.84

Branch Loan Details

➢ The product operation on the **Branch Reserve Details** and **Branch Loan Details** tables would result in the following table:

Branch	Branch_id	Reserve (Billion €)	Loan Amount (Billion €)
London	BS-01	9.2	0.56
London	BS-01	9.2	0.84
London	BS-02	10	0.56
London	BS-02	10	0.84
Paris	BS-03	15	0.56
Paris	BS-03	15	0.84
Los Angeles	BS-04	50	0.56
Los Angeles	BS-04	50	0.84
Washington	BS-05	30	0.56
Washington	BS-05	30	0.84

Product of Branch Reserve Details and Branch Loan Details

UNION

- Suppose an official of the bank with the data given in tables `Branch Reserve Details` and `Branch Loan Details` wanted to know which branches had reserves below 20 billion Euros or loans.
- The resultant table would consist of branches with either reserves below 20 billion Euros or loans or both.
- This is similar to the union of two sets of data; first, set of branches with reserve less than 20 billion Euros and second, branches with loans.
- Branches with both, reserves below 20 billion Euros and loans would be displayed only once.
- The `UNION` operator does just that, it collects the data from the different tables and presents a unified version of the complete data.
- The union operation is represented by the symbol, '`U`'.
- The union of the **`Branch Reserve Details`** and **`Branch Loan Details`** tables would generate the following table:

Branch	Branch_id
London	BS-01
London	BS-02
Paris	BS-03

Unified Representation of Branches with Less Reserves or Loans

INTERSECT

➢ Suppose the same official after seeing this data wanted to know which of these branches had both low reserves and loans too.

➢ The answer would be the intersect relational operation.

➢ The `INTERSECT` operator generates data that holds true in all the tables it is applied on.

➢ It is based on the intersection set theory and is represented by the '∩' symbol.

➢ The result of the intersection of the **Branch Reserve Details** and **Branch Loan Details** tables would be a list of branches that have both reserves below 20 billion Euros and loans in their account.

➢ The resultant table generated is as follows:

Branch	Branch_id
London	BS-01
London	BS-02

Branches with Low Reserves and Loans

➢ If the same official now wanted the list of branches that had low reserves but no loans, then the official would have to use the difference operation.

➢ The `DIFFERENCE` operator, symbolized as '–', generates data from different tables too, but it generates data that holds true in one table and not the other.

➢ Thus, the branch would have to have low reserves and no loans to be displayed.

➢ Following table is the result generated:

Branch	Branch_id
Paris	BS-03

Branches with Low Reserves but No Loans

JOIN

➢ The `JOIN` operation is an enhancement to the product operation.

➢ It allows a selection to be performed on the product of tables.

➢ For example, if the reserve values and loan amounts of branches with low reserves and loan values was needed, the product of the **Branch Reserve Details** and **Branch Loan Details** would be required.

➢ Once the product of the two tables would be generated, only those branches would be listed which have both reserves below 20 billion Euros and loans.

➢ Following table is generated as a result of the `JOIN` operation:

Branch	Branch_id	Reserve (Billion €)	Loan Amount (Billion €)
London	BS-01	9.2	0.56
London	BS-02	10	0.84

Detailed List of Branches with Low Reserve and Loans

DIVIDE

➢ Suppose an official wanted to see the branch names and reserves of all the branches that had loans.

➢ This process can be made very easy by using the `DIVIDE` operator.

➢ All that the official needs to do is divide the **Branch Reserve Details** table by the list of branches, that is, the **Branch_Id** column of the **Branch Loan Details** table.

➢ Following table is the result generated.

Branch	Reserve (Billion €)
London	9.2
London	10

Resultant Table of Division Operation

➢ The attributes of the divisor table should always be a subset of the dividend table.
➢ The resultant table would always be void of the attributes of the divisor table, and the records not matching the records in the divisor table.

- Data modeling is the process of applying an appropriate data model to the data at hand.

- E-R model views the real world as a set of basic objects and relationships among them.

- Entity, attributes, entity set, relationships, and relationship sets form the five basic components of E-R model.

- Mapping cardinalities express the number of entities that an entity is associated with.

- The process of removing redundant data from the tables of a relational database is called normalization.

- Relational Algebra consists of a collection of operators that help retrieve data from the relational databases.

- SELECT, PRODUCT, UNION, and DIVIDE are some of the relational algebra operators.