Relational Database Systems – Part 01

Karine Reis Ferreira
karine@dpi.inpe.br
Database System

**Database:**
- is a collection of related data.
- represents some aspect of the real world.
- is a logically coherent collection of data with some inherent meaning.
- is designed, built, and populated with data for a specific purpose.

Example: Amazon.com (over 2 terabytes - 200 different servers)

Source: (Elmasri and Navathe, 2011)
Database System

**Database Management System (DBMS):**

- is a collection of programs that enables users to create and maintain a database.

- system that facilitates the processes of defining, constructing, manipulating, and sharing databases among various users and applications.

Source: (Elmasri and Navathe, 2011)
Database System

An application program accesses the database by sending queries or requests for data to the DBMS.

Database System: Database + DBMS Software + Application Programs.

Source: (Elmasri and Navathe, 2011)
Advantages of using DBMS

① Controlling redundancy

✓ Problems: duplication of effort, wasted storage space, inconsistent.
✓ Data normalization

② Restricting unauthorized access

✓ Security and authorization subsystem: create accounts and restrictions.

③ Providing persistent storage for program objects

✓ Complex object in C++ stored permanently in an object-oriented DBMS

④ Providing storage structures and search techniques for efficient query processing

✓ Indexes, buffering/caching, query processing and optimization module
Advantages of using DBMS

⑤ Providing backup and recovery

- Recovery subsystem: ensure that the transaction is resumed from the point at which it was interrupted so that its full effect is recorded in the database.

⑥ Providing multiple user interfaces

- Query languages, programming language interfaces, Graphical User Interfaces (GUIs), web GUI interfaces, etc.
DBMS: Multiple User Interfaces

```
int main()
{
    ...
}
```
Advantages of using DBMS

⑦ Representing complex relationships among data
  ✓ Represent a variety of complex relationships among the data, define new relationships, and retrieve and update related data easily and efficiently.

⑧ Enforcing integrity constraints
  ✓ Integrity constraints: referential integrity, key or uniqueness, semantics, ...

⑨ Permitting inference and actions using rules
  ✓ Deductive database systems: define *deduction rules* for *inferencing* new information from the stored database facts.
  ✓ Triggers, stored procedures, ...

Data Model

Data model: a collection of concepts that can be used to describe the structure of a database.

- **High-level or conceptual data models**: provide concepts that are close to the way many users perceive data. Ex.: Entity-Relationship model

- **Representational or implementation data models**: provide concepts that may be easily understood by end users but that are not too far removed from the way data is organized in computer storage. Ex.: Relational data model

- **Low-level or physical data models**: provide concepts that describe the details of how data is stored on the computer storage media.
Database Design

Source: (Elmasri and Navathe, 2011)

Phase 1: Requirements collection and analysis

Phase 2: Conceptual database design

Phase 3: Choice of DBMS

Phase 4: Data model mapping (logical design)

Phase 5: Physical design

Phase 6: System implementation and tuning

Data content, structure, and constraints

Data requirements

Conceptual Schema design (DBMS-independent)

Logical Schema and view design (DBMS-dependent)

Internal Schema design (DBMS-dependent)

DDL statements

SDL statements

Database applications

Processing requirements

Transaction and application design (DBMS-independent)

Frequencies, performance constraints

Transaction and application implementation
Conceptual schema for the database that is independent of a specific DBMS.

High-level data model ER or EER model
Entity-Relationship – ER Model

✓ Proposed in 1976 by Peter Chen
✓ Conceptual data models use concepts such as *entities*, *attributes*, and *relationships*:
  ✓ **Entity**: real-world object or concept from the miniworld that is described in the database.
  ✓ **Attribute**: property of interest that further describes an entity.
  ✓ **Relationship**: association among the entities.
Source: (Elmasri and Navathe, 2011)
Entities:
- EMPLOYEE
- DEPARTMENT
- PROJECT
- DEPENDENT

Relationships:
- WORKS_FOR
- MANAGES
- CONTROLS
- WORKS_ON
- DEPENDENT_OF
- SUPERVISION

Source: (Elmasri and Navathe, 2011)
**ER Model – Types of Attributes**

- **Composite** versus **Simple** (Atomic) Attributes => Composite: can be divided into smaller subparts (ex. Name). Simple: are not divisible.

- **Single-Valued** versus **Multivalued** Attributes => Single-Valued: have a single value for a particular entity (ex. BDate, Sex). Multivalued: can have multiple values (ex. Color of a car, Locations).

- **Stored** versus **Derived** Attributes => The Age attribute is a derived attribute and is said to be derivable from the BDate attribute, which is called a stored attribute.

- **NULL** Values => A particular entity may not have an applicable value for an attribute.
ER Model – Key Attributes

Composite Attribute

Multivalued Attribute

Source: (Elmasri and Navathe, 2011)
**ER Model – Domains of Attributes**

- **Value Sets (Domains) of Attributes:** Each simple attribute of an entity is associated with a value set (or domain of values), which specifies the set of values that may be assigned to that attribute for each individual entity.

  - Example: the range of ages allowed for employees is between 16 and 70 => set of integer numbers between 16 and 70.

- integer, string, Boolean, float, enumerated type, etc…
ER Model – Relationships

- **Degree**: number of participating entities. Binary: between two entities, ternary: among three entities, and so on.

- **Recursive** relationships: between two instances of the same entity. Ex. SUPERVISION

- **Cardinality** ratios for binary relationships: 1:1, 1:N, N:1, and M:N.
  - Ex: MANAGES (1:1), WORKS_FOR (1:N), WORKS_ON (M:N)

- **Attributes**: relationship can also have attributes, similar to those of entities.
ER Model – Cardinality

1:1

N:N

Source: (Elmasri and Navathe, 2011)
ER Model – Relationships

- Participation Constraints and Existence Dependencies:
  - Total participation: every entity in the total set of employee entities must be related to a department entity via WORKS_FOR. Total participation is also called existence dependency.
  - Partial participation: some or part of the set of employee entities are related to some department entity via MANAGES, but not necessarily all.

In ER diagrams, total participation (or existence dependency) is displayed as a double line connecting the participating entity type to the relationship, whereas partial participation is represented by a single line.
Total participation

Partial participation

Source: (Elmasri and Navathe, 2011)
**ER Model – Weak Entity**

- **Weak entity**: entity that do not have key attributes of their own. A weak entity type always has a *total participation constraint* (existence dependency) with respect to its *identifying relationship* because a weak entity cannot be identified without an owner entity. A weak entity type normally has a *partial key*, which is the attribute that can uniquely identify weak entities that are related to the same owner entity. Ex. DEPENDENT

- **Strong entity**: regular entity that do have a key attribute.

In ER diagrams, both a weak entity type and its identifying relationship are distinguished by surrounding their boxes and diamonds with double lines. The partial key attribute is underlined with a dashed or dotted line.
ER Model – Notations

Source: (Elmasri and Navathe, 2011)
ER Model – Notations

Source: (Elmasri and Navathe, 2011)
Source: (Elmasri and Navathe, 2011)
ER Model – Ternary Relationship

Source: (Elmasri and Navathe, 2011)
ER Model – Ternary Relationship

✓ Some database design tools are based on variations of the ER model that permit only binary relationships.

✓ A ternary relationship such as SUPPLY must be represented as a weak entity type, with no partial key and with three identifying relationships.
ER Model – Ternary Relationship

Source: (Elmasri and Navathe, 2011)
Exercise 01
The conceptual schema from the high-level data model used in Phase 2 is mapped into the data model of the chosen DBMS.
The Relational Data Model

- First introduced by Ted Codd of IBM Research in 1970 in a classic paper.
- Early 1980s: first commercial implementations, such as the Oracle DBMS.
- Current popular relational DBMSs (RDBMSs) include:
  - Commercial: DB2 and Informix Dynamic Server (from IBM), Oracle (from Oracle), Sybase DBMS (from Sybase) and SQLServer and Access (from Microsoft).
  - Open source: MySQL and PostgreSQL.
Relational DBMS - RDBMS

IBMDB2Universal
Database

MySQL

SQL Server 2008

Informix

PostgreSQL
The Relational Data Model

- Represents the database as a collection of relations (table of values)
- Formal terminology: row => tuple, a column header => attribute, table => relation, data type describing the types of values that can appear in each column => domain of possible values.

Source: (Elmasri and Navathe, 2011)
Relation - Definition

Given the domains $D_1, D_2, ..., D_n$ that are not necessarily distinct, a relation is defined as:

$$ R = \{ (d_1, d_2, ..., d_n) \mid d_1 \in D_1, d_2 \in D_2, ..., d_n \in D_n \} $$

✓ A set $(d_1, d_2, ..., d_n)$ of ordered values define a tupla
✓ A relation is a set of ordered $n$-tuplas, where $n$ define the relation degree
## Relation - Schema x Instances

The table below represents an instance of the EMPLOYEE relation, where each row is an instance of the schema.

### EMPLOYEE

<table>
<thead>
<tr>
<th>Fname</th>
<th>Minit</th>
<th>Lname</th>
<th>Ssn</th>
<th>Bdate</th>
<th>Address</th>
<th>Sex</th>
<th>Salary</th>
<th>Super_ssn</th>
<th>Dno</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>B</td>
<td>Smith</td>
<td>123456789</td>
<td>1965-01-09</td>
<td>731 Fondren, Houston, TX</td>
<td>M</td>
<td>30000</td>
<td>333445555</td>
<td>5</td>
</tr>
<tr>
<td>Franklin</td>
<td>T</td>
<td>Wong</td>
<td>333445555</td>
<td>1955-12-08</td>
<td>638 Voss, Houston, TX</td>
<td>M</td>
<td>40000</td>
<td>888665555</td>
<td>5</td>
</tr>
<tr>
<td>Alicia</td>
<td>J</td>
<td>Zelaya</td>
<td>999887777</td>
<td>1968-01-19</td>
<td>3321 Castle, Spring, TX</td>
<td>F</td>
<td>25000</td>
<td>987654321</td>
<td>4</td>
</tr>
<tr>
<td>Jennifer</td>
<td>S</td>
<td>Wallace</td>
<td>987654321</td>
<td>1941-06-20</td>
<td>291 Berry, Bellaire, TX</td>
<td>F</td>
<td>43000</td>
<td>888665555</td>
<td>4</td>
</tr>
<tr>
<td>Ramesh</td>
<td>K</td>
<td>Narayan</td>
<td>666884444</td>
<td>1962-09-15</td>
<td>975 Fire Oak, Humble, TX</td>
<td>M</td>
<td>38000</td>
<td>333445555</td>
<td>5</td>
</tr>
<tr>
<td>Joyce</td>
<td>A</td>
<td>English</td>
<td>453453453</td>
<td>1972-07-31</td>
<td>5631 Rice, Houston, TX</td>
<td>M</td>
<td>25000</td>
<td>333445555</td>
<td>5</td>
</tr>
<tr>
<td>Ahmad</td>
<td>V</td>
<td>Jabbar</td>
<td>987987987</td>
<td>1969-03-29</td>
<td>980 Dallas, Houston, TX</td>
<td>M</td>
<td>25000</td>
<td>987654321</td>
<td>4</td>
</tr>
<tr>
<td>James</td>
<td>E</td>
<td>Borg</td>
<td>888665555</td>
<td>1937-11-10</td>
<td>450 Stone, Houston, TX</td>
<td>M</td>
<td>55000</td>
<td>NULL</td>
<td>1</td>
</tr>
</tbody>
</table>

The source for this representation is (Elmasri and Navathe, 2011).
Relation – Characteristics

- **Ordering of Tuples in a Relation**: tuples in a relation do not have any particular order.

- **Ordering of Values within a Tuple**: according to the definition of a relation, an *n-tuple* is an ordered list of *n* values, so the ordering of values in a tuple is important.

- **Values and NULLs in the Tuples**: each value in a tuple is an atomic value (not divisible). NULL values: values of attributes that may be unknown or may not apply to a tuple.
Relation Model Constraints

- Constraints materialize the rules in the miniworld that the database represents.

- Types of constraints:
  - Domain Constraints
  - Key Constraints: primary key and unique key
  - NOT NULL Constraints
  - Referential Integrity Constraints: foreign key
  - Semantic Integrity Constraints
Domain Constraints

- The data types associated with domains typically include:
  - Standard numeric data types for integers (such as short integer, integer, and long integer);
  - Real numbers (float and double precision, float);
  - Characters;
  - Booleans;
  - Fixed-length strings and variable-length strings;
  - Date, time, timestamp, and money
Key Constraints

- All tuples in a relation must also be distinct, that is, no two tuples can have the same combination of values for all their attributes.

- **Candidate keys**: attributes that have uniqueness constraints, that is, no two distinct tuples can have the same value. A relation can have \( n \) candidate keys. Candidate keys are designated as **unique keys**.

- **Primary key**: a candidate key whose values are used to identify tuples in the relation. A relation can have only one primary key.
### Key Constraints

The CAR relation, with two candidate keys: License_number and Engine_serial_number.

#### Source
Elmasri and Navathe, 2011

<table>
<thead>
<tr>
<th>License_number</th>
<th>Engine_serial_number</th>
<th>Make</th>
<th>Model</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Texas ABC-739</td>
<td>A69352</td>
<td>Ford</td>
<td>Mustang</td>
<td>02</td>
</tr>
<tr>
<td>Florida TVP-347</td>
<td>B43696</td>
<td>Oldsmobile</td>
<td>Cutlass</td>
<td>05</td>
</tr>
<tr>
<td>New York MPO-22</td>
<td>X83554</td>
<td>Oldsmobile</td>
<td>Delta</td>
<td>01</td>
</tr>
<tr>
<td>California 432-TFY</td>
<td>C43742</td>
<td>Mercedes</td>
<td>190-D</td>
<td>99</td>
</tr>
<tr>
<td>California RSK-629</td>
<td>Y82935</td>
<td>Toyota</td>
<td>Camry</td>
<td>04</td>
</tr>
<tr>
<td>Texas RSK-629</td>
<td>U028365</td>
<td>Jaguar</td>
<td>XJS</td>
<td>04</td>
</tr>
</tbody>
</table>
Referential Integrity Constraint

- It is specified between two relations and is used to maintain the consistency among tuples in the two relations.

- **Foreign key**: specify a referential integrity constraint between the two relation schemas R1 and R2.
  - The attributes in FK have the same domain(s) as the primary key attributes PK of R2; the attributes FK are said to reference or refer to the relation R2.
  - A value of FK in a tuple t1 of the current state r1(R1) either occurs as a value of PK for some tuple t2 in the current state r2(R2) or is NULL.
### Primary Keys ?

### Unique Keys ?

### Foreign Keys ?

Source: (Elmasri and Navathe, 2011)
Foreign Keys

Source: (Elmasri and Navathe, 2011)
Foreign Keys

Source: (Elmasri and Navathe, 2011)
Semantic Integrity Constraints

- Example: “the salary of an employee should not exceed the salary of the employee’s supervisor and the maximum number of hours an employee can work on all projects per week is 56”

- Such constraints can be specified and enforced within the application programs that update the database, or by using a general-purpose constraint specification language.

- Examples: triggers and assertions. In SQL: CREATE ASSERTION and CREATE TRIGGER
Relational Model – Operations

- **Insert**: provides a list of attribute values for a new tuple \( t \) that is to be inserted into a relation \( R \).


  - *Result*: This insertion violates the entity integrity constraint (NULL for the primary key Ssn), so it is rejected.
**Delete:** remove tuples of a relation.

- *Example:* Delete the EMPLOYEE tuple with Ssn = ‘999887777’.

- *Result:* This deletion is not acceptable, because there are tuples in WORKS_ON that refer to this tuple. Hence, if the tuple in EMPLOYEE is deleted, referential integrity violations will result.
Relational Model – Operations

- **Update (or Modify)**: change the values of one or more attributes in a tuple (or tuples) of some relation R.

  ✓ *Example*: Update the Dno of the EMPLOYEE tuple with Ssn = ‘999887777’ to 7.

  ✓ *Result*: Unacceptable, because it violates referential integrity.
**Transaction**

- **A transaction** is an executing program that forms a logical unit of database processing.
- It includes one or more database access operations—these can include insertion, deletion, modification, or retrieval operations.
- End of the transaction: database in a valid or consistent state
- Transactions submitted by various users may execute concurrently and may access and update the same database items.
- Concurrency control and recovery mechanisms are necessary
Problems of concurrent execution

Two transactions: T1 and T2

<table>
<thead>
<tr>
<th>T1</th>
<th>T2</th>
</tr>
</thead>
<tbody>
<tr>
<td>read_item(X); X := X - N;</td>
<td>read_item(X); X := X + M;</td>
</tr>
<tr>
<td>write_item(X);</td>
<td>write_item(X);</td>
</tr>
<tr>
<td>read_item(Y);</td>
<td></td>
</tr>
<tr>
<td>Y := Y + N; write_item(Y);</td>
<td></td>
</tr>
</tbody>
</table>

Time

Item X has an incorrect value because its update by T1 is lost (overwritten).

Source: (Elmasri and Navathe, 2011)
Problems of concurrent execution

Two transactions: T1 and T2

<table>
<thead>
<tr>
<th>Time</th>
<th>T1</th>
<th>T2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>read_item(X);</td>
<td>read_item(X);</td>
</tr>
<tr>
<td></td>
<td>X := X - N;</td>
<td>X := X + M;</td>
</tr>
<tr>
<td></td>
<td>write_item(X);</td>
<td>write_item(X);</td>
</tr>
<tr>
<td></td>
<td>read_item(Y);</td>
<td></td>
</tr>
</tbody>
</table>

Transaction T1 fails and must change the value of X back to its old value; meanwhile T2 has read the temporary incorrect value of X.

Source: (Elmasri and Navathe, 2011)
Transaction

- A transaction is an atomic unit of work that should either be completed in its entirety or not done at all.

- For recovery purposes, the system needs to keep track of when each transaction starts, terminates, and commits or aborts:
  - BEGIN_TRANSACTION
  - END_TRANSACTION
  - COMMIT_TRANSACTION
  - ROLLBACK (or ABORT)
Transaction – ACID properties

- **Atomicity**: A transaction is an atomic unit of processing; it should either be performed in its entirety or not performed at all.

- **Consistency preservation**: if a transaction is completely executed from beginning to end without interference from other transactions, it should take the database from one consistent state to another.

- **Isolation**: The execution of a transaction should not be interfered with by any other transactions executing concurrently.

- **Durability** or permanency. The changes applied to the database by a committed transaction must persist in the database. These changes must not be lost because of any failure.
Transaction – Serial Schedule

Serial schedules involving transactions T1 and T2. (a) Serial schedule A: T1 followed by T2. (b) Serial schedule B: T2 followed by T1.

Source: (Elmasri and Navathe, 2011)
Transaction – Nonserial Schedule

(c) Two nonserial schedules C and D with interleaving of operations.

Schedule C

<table>
<thead>
<tr>
<th>Time</th>
<th>$T_1$</th>
<th>$T_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>read_item(X); $X := X - N;$</td>
<td>read_item(X); $X := X + M;$</td>
<td></td>
</tr>
<tr>
<td>write_item(X); read_item(Y);</td>
<td>$Y := Y + N;$ write_item(Y);</td>
<td></td>
</tr>
</tbody>
</table>

Schedule D

<table>
<thead>
<tr>
<th>Time</th>
<th>$T_1$</th>
<th>$T_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$read_{item}(X);$ $X := X - N;$</td>
<td>read_item(X); $X := X + M;$</td>
<td></td>
</tr>
<tr>
<td>write_{item}(X);</td>
<td>write_item(Y); $Y := Y + N;$ write_item(Y);</td>
<td></td>
</tr>
</tbody>
</table>

Correct results?

Source: (Elmasri and Navathe, 2011)
Transaction – Nonserial Schedule

(c) Two nonserial schedules C and D with interleaving of operations.

Schedule C:

\[
\begin{align*}
T_1 & : \text{read\_item}(X); \quad X := X - N; \\
     & \text{write\_item}(X); \quad \text{read\_item}(Y); \\
     & \text{Y} := Y + N; \quad \text{write\_item}(Y);
\end{align*}
\]

Schedule D:

\[
\begin{align*}
T_1 & : \text{read\_item}(X); \quad X := X - N; \\
     & \text{write\_item}(X); \quad \text{read\_item}(Y); \\
     & \text{Y} := Y + N; \quad \text{write\_item}(Y);
\end{align*}
\]

\[
\begin{align*}
T_2 & : \text{read\_item}(X); \\
     & \text{write\_item}(X);
\end{align*}
\]

C: erroneous result (lost update problem) \(\Rightarrow\) T2 reads the value of X before it is changed by T1.

D: correct result

Source: (Elmasri and Navathe, 2011)
SQL: Structured Query Language

- A standard (ISO) for relational databases.
- Based on the relational algebra
- Higher-level declarative language interface: user only specifies what the result is to be, leaving the actual optimization and decisions on how to execute the query to the DBMS.

- Statements for:
  - data definitions, queries, and updates: DDL and DML
  - defining views on the database
  - specifying security and authorization
  - defining integrity constraints, and
  - specifying transaction controls.
CREATE DATABASE Test

CREATE TABLE States (NAME VARCHAR(100), UF VARCHAR(2), POP NUMBER(10,10))

INSERT INTO States VALUES ('Minas Gerais', 'MG', 9999)

SELECT * FROM States WHERE UF = 'MG'

SQL-DDL (Data Definition Language) SQL-DML (Data Manipulation Language)
## SQL – Some Commands

<table>
<thead>
<tr>
<th>Commands</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>select</code></td>
<td>Select data</td>
<td>DML</td>
</tr>
<tr>
<td><code>insert, update, delete</code></td>
<td>Add, modify and remove data</td>
<td>DML</td>
</tr>
<tr>
<td><code>commit, rollback</code></td>
<td>Transaction controls</td>
<td>DDL</td>
</tr>
<tr>
<td><code>create, alter, drop</code></td>
<td>create, alter and eliminate schemas and tables</td>
<td>DDL</td>
</tr>
</tbody>
</table>
From ER Model To Relational Model
From ER Model To Relational Model

Figure 3.7
Referential integrity constraint on the COMPANY relational c schema.
ER Model To Relational Model

**Step 1: Mapping of Regular Entity Types.** For each regular entity E, create a relation R that includes all the simple attributes of E.

Example: EMPLOYEE, DEPARTMENT, PROJECT

**Step 2: Mapping of Weak Entity Types.** For each weak entity W with owner entity E, create a relation R and include all simple attributes of W as attributes of R. In addition, include as foreign key attributes of R, the primary key attribute(s) of E.

Example: DEPENDENT
Step 3: Mapping of Binary 1:1 Relationship Types: identify the relations S and T that correspond to the relationship and choose one approach:

1. **Foreign key approach**: Choose one of the relations, for example S, and include as a foreign key in S the primary key of T.

2. **Merged relation approach**: Merge the two entities into a single relation.

3. **Cross-reference or relationship relation approach**: set up a third relation R for the purpose of cross-referencing the primary keys of the two relations S and T representing the entities.
Example: (Approach 1): include the primary key of the EMPLOYEE as foreign key in the DEPARTMENT (Mgr_ssn). Also include the simple attribute Start_date of the MANAGES relationship (Mgr_start_date).
Step 4: Mapping of Binary 1:N Relationship Types: identify the relation S that represents the participating entity at the N-side of the relationship. Include as foreign key in S the primary key of the relation T.

Examples:

1. WORKS_FOR => primary key Dnumber of the DEPARTMENT as foreign key in the EMPLOYEE (Dno).

2. SUPERVISION => primary key of the EMPLOYEE as foreign key in the EMPLOYEE (Super_ssn).

3. CONTROLS => foreign key attribute Dnum of PROJECT, which references the primary key Dnumber of the DEPARTMENT.
Step 5: Mapping of Binary M:N Relationship Types: For each binary M:N relationship type R, create a new relation S to represent R. Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types; their combination will form the primary key of S. Also include any simple attributes of the M:N relationship as attributes of S.

Example: relationship WORKS_ON => relation WORKS_ON
Er Model To Relational Model

Step 6: Mapping of Multivalued Attributes: For each multivalued attribute A, create a new relation R that include an attribute corresponding to A, plus the primary key attribute K — as a foreign key in R — of the relation that represents the entity type or relationship type that has A as a multivalued attribute.

Example: create a relation DEPT_LOCATIONS. The attribute Dlocation represents the multivalued attribute LOCATIONS of DEPARTMENT, while Dnumber—as foreign key—represents the primary key of the DEPARTMENT relation.
ER Model To Relational Model

**Step 7: Mapping of N-ary Relationship Types.** For each n-ary relationship type $R$, where $n > 2$, create a new relation $S$ to represent $R$. Include as foreign key attributes in $S$ the primary keys of the relations that represent the participating entity types. Also include any simple attributes of the n-ary relationship type as attributes of $S$. 
## ER Model To Relational - Summary

<table>
<thead>
<tr>
<th>ER MODEL</th>
<th>RELATIONAL MODEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity type</td>
<td><em>Entity</em> relation</td>
</tr>
<tr>
<td>1:1 or 1:N relationship type</td>
<td><em>Foreign key</em> (or <em>relationship</em> relation)</td>
</tr>
<tr>
<td>M:N relationship type</td>
<td><em>Relationship</em> relation and <em>two</em> foreign keys</td>
</tr>
<tr>
<td><em>n</em>-ary relationship type</td>
<td><em>Relationship</em> relation and <em>n</em> foreign keys</td>
</tr>
<tr>
<td>Simple attribute</td>
<td><em>Attribute</em></td>
</tr>
<tr>
<td>Composite attribute</td>
<td>Set of simple component attributes</td>
</tr>
<tr>
<td>Multivalued attribute</td>
<td><em>Relation</em> and foreign key</td>
</tr>
<tr>
<td>Value set</td>
<td><em>Domain</em></td>
</tr>
<tr>
<td>Key attribute</td>
<td>Primary (or secondary) key</td>
</tr>
</tbody>
</table>

Source: (Elmasri and Navathe, 2011)
Exercise 02
Specifications for the stored database in terms of physical file storage structures, buffer, and indexes.

Source: (Elmasri and Navathe, 2011)
Database and application programs are implemented, tested, and eventually deployed for service.

It can result in conceptual, logical or internal (physical) schema changes.

Source: (Elmasri and Navathe, 2011)