

Relational Database Systems

– Part 01 –

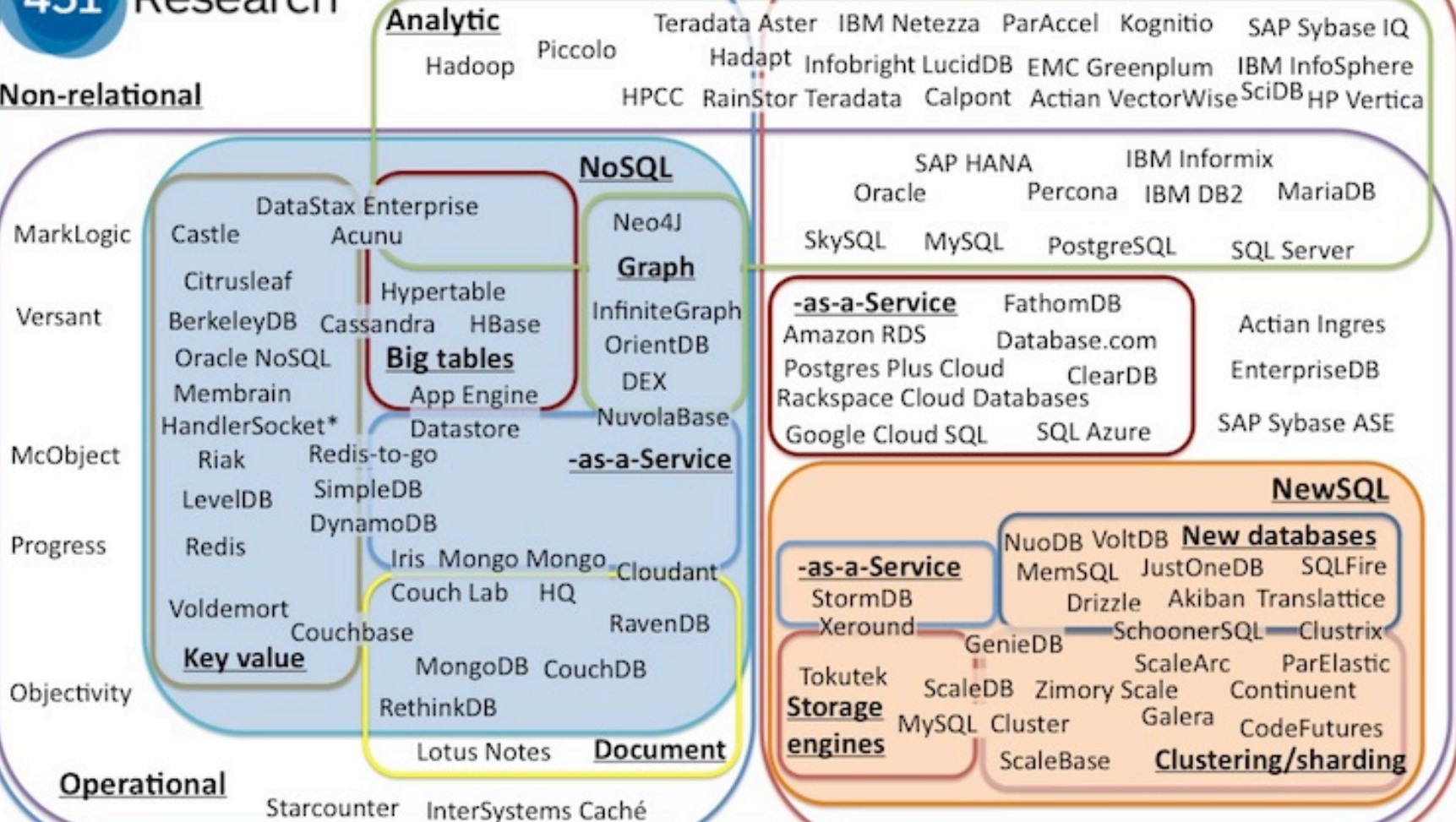
Karine Reis Ferreira

karine.ferreira@inpe.br

The evolving database landscape

451 Research

Non-relational



© 2012 by The 451 Group. All rights reserved

Source: <https://blogs.the451group.com>, 2015

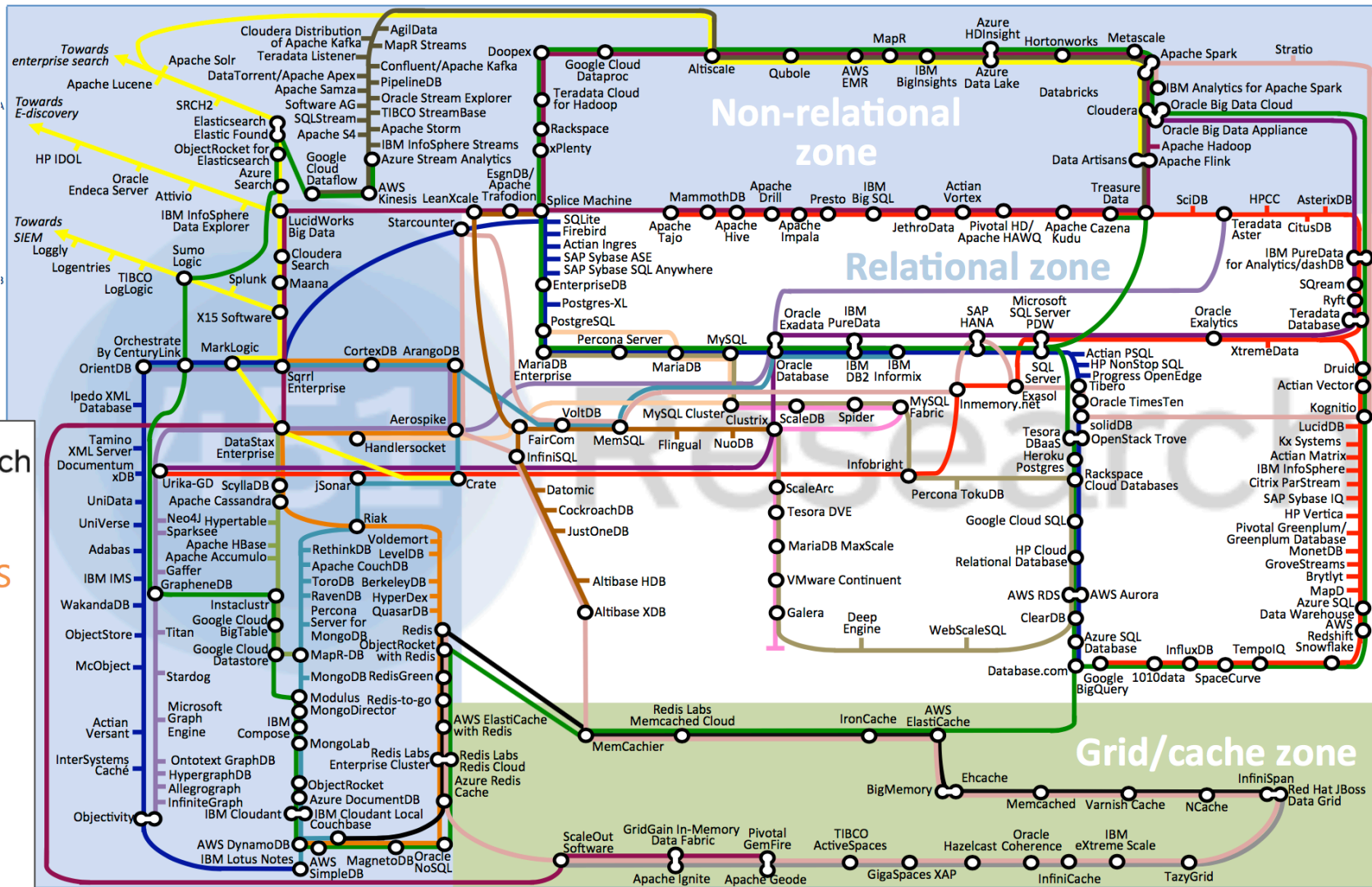
NoSQL e NewSQL

NoSQL:

- ✓ The movement began early 2009!
- ✓ Main motivation: relational database systems are not ready for big data!
- ✓ **Non-relational** database management system (schema-free, not ACID, ...)
- ✓ **Distributed**
- ✓ Open Source
- ✓ **Horizontally scalable**
- ✓ Easy replication support and simple API
- ✓ Examples: Cassandra, MongoDB, CouchBD, Neo4J, ...

NewSQL:

- ✓ New relational database products and services designed to bring the benefits of the **relational model** to **distributed architectures**.
- ✓ SQL as the primary mechanism
- ✓ ACID properties support
- ✓ **Horizontally scalable**
- ✓ Examples of NewSQL database products: VoltDB, RethinkDB, MySQL Cluster with NDB, ...
- ✓ Examples of NewSQL services: Amazon Relational Database Service, Microsoft SQL Azure, Xeround, ...

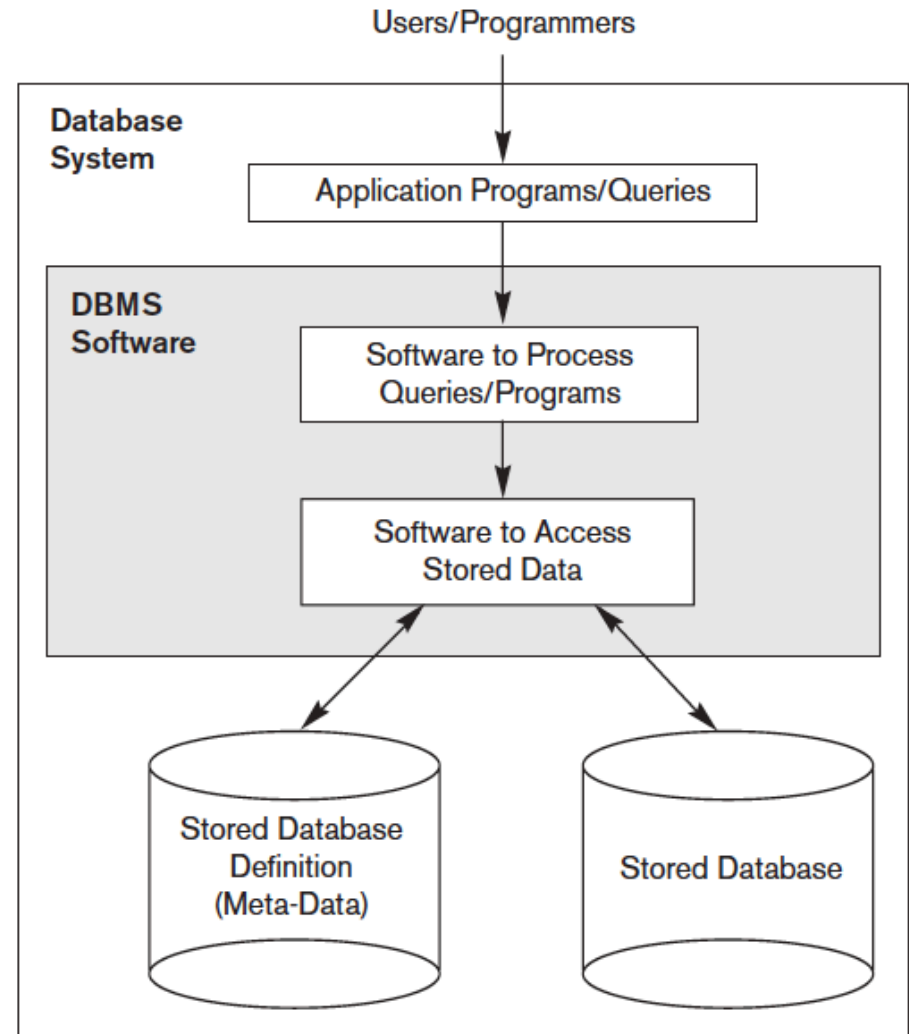


Database System

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)

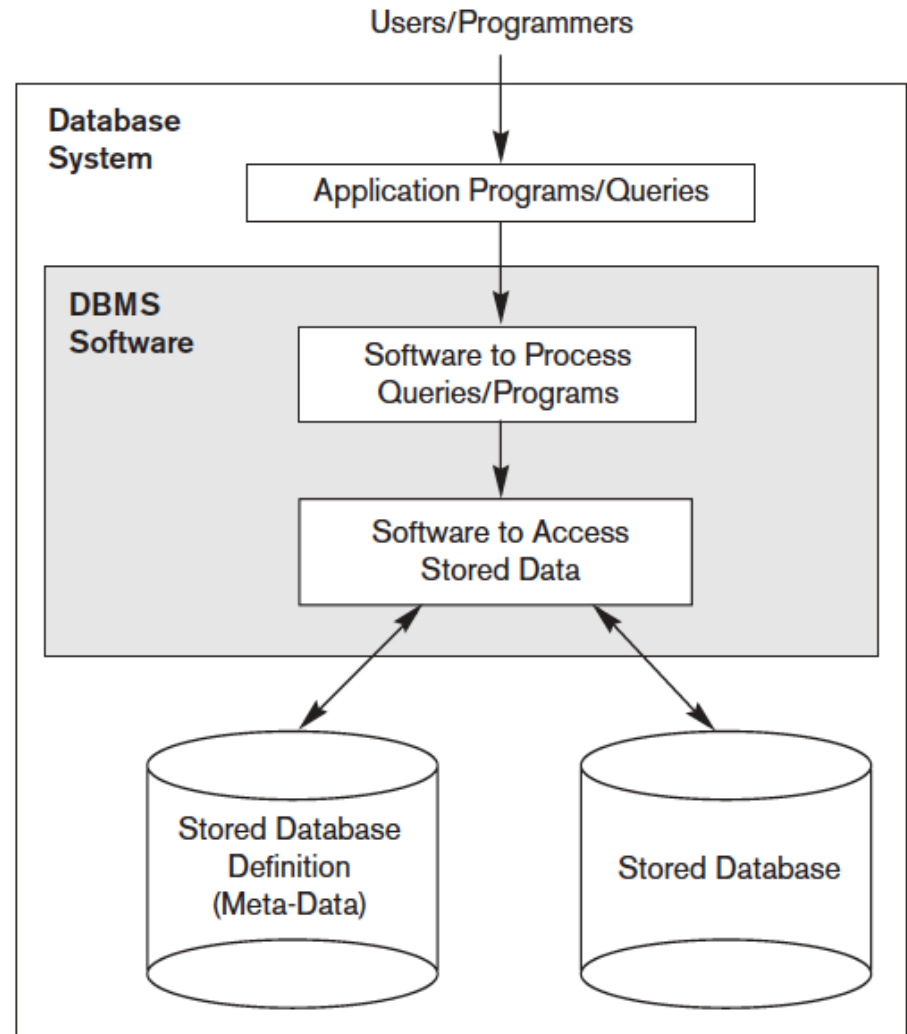


Source: (Elmasri and Navathe, 2011)

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.

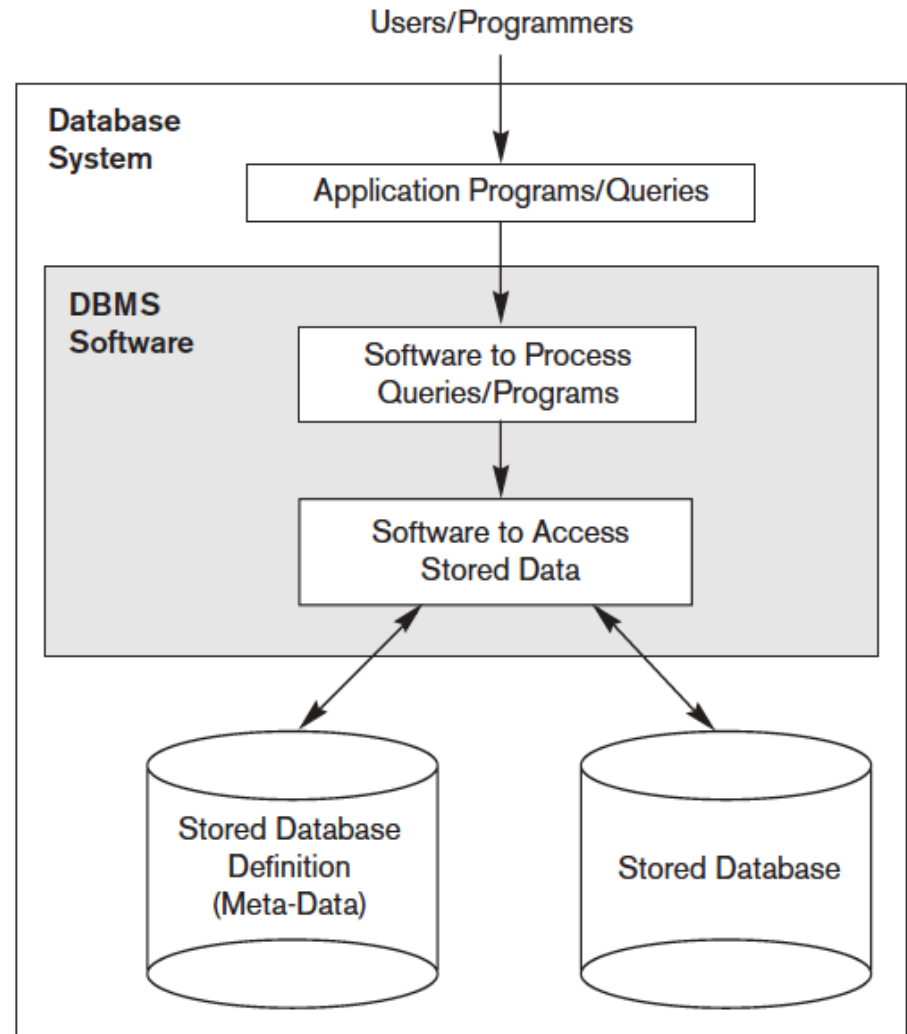


Source: (Elmasri and Navathe, 2011)

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

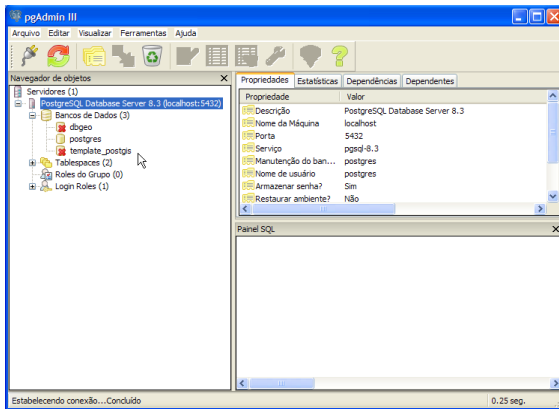
- ✓ Recovery subsystem: ensure that the transaction is resumed from the point at which it was interrupted so that its full effect is recorded in the database.

⑥ Providing multiple user interfaces

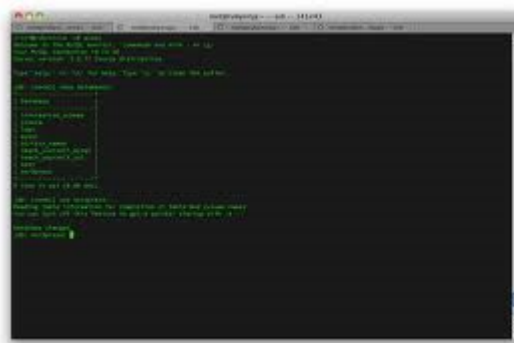
- ✓ Query languages, programming language interfaces, Graphical User Interfaces (GUIs), web GUI interfaces, etc.

DBMS: Multiple User Interfaces

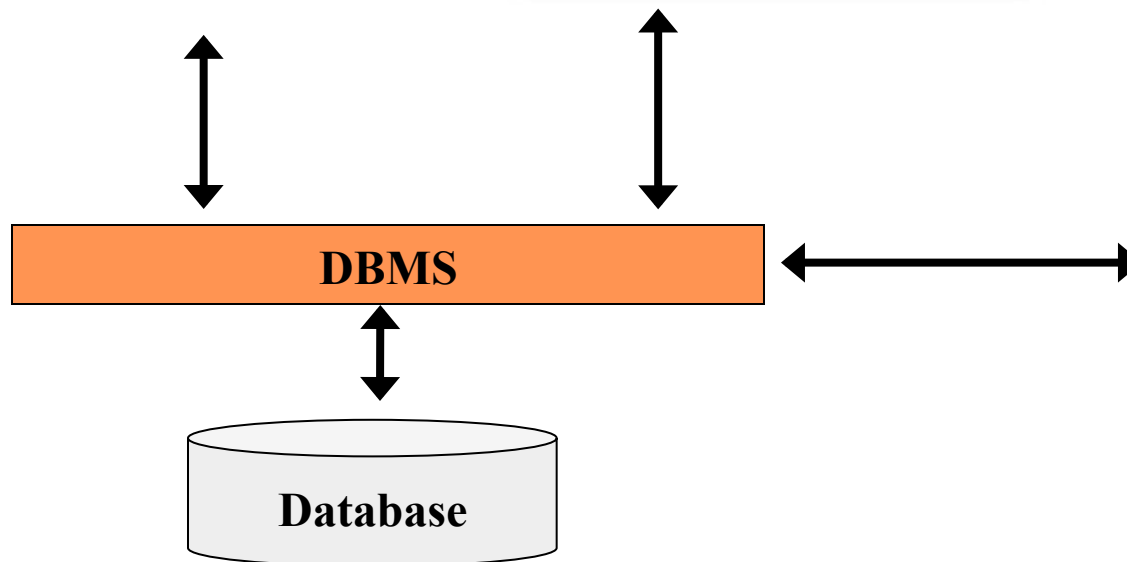
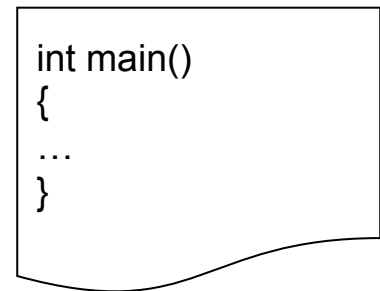
GUI



Prompt



C++ API



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

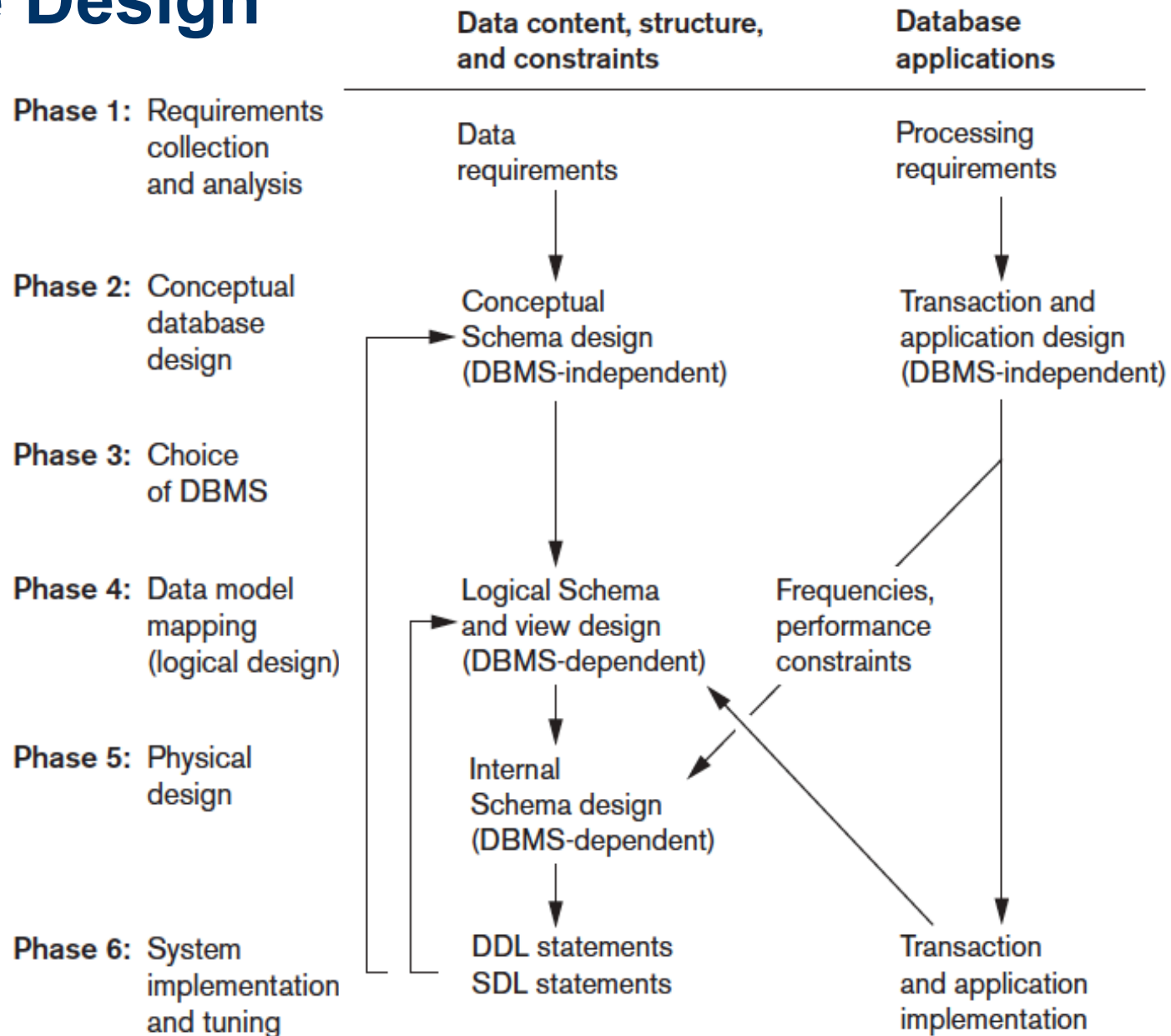
- ✓ Deductive database systems: define *deduction rules* for *inferencing* new information from the stored database facts.
- ✓ Triggers, stored procedures, ...

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



Database Design

Conceptual schema for the database that is independent of a specific DBMS.

High-level data model ER or EER model

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

Database applications

Data requirements

Processing requirements

Conceptual Schema design (DBMS-independent)

Transaction and application design (DBMS-independent)

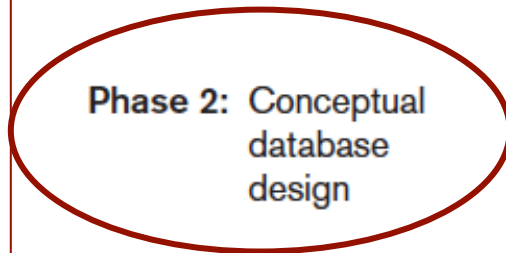
Logical Schema and view design (DBMS-dependent)

Frequencies, performance constraints

Internal Schema design (DBMS-dependent)

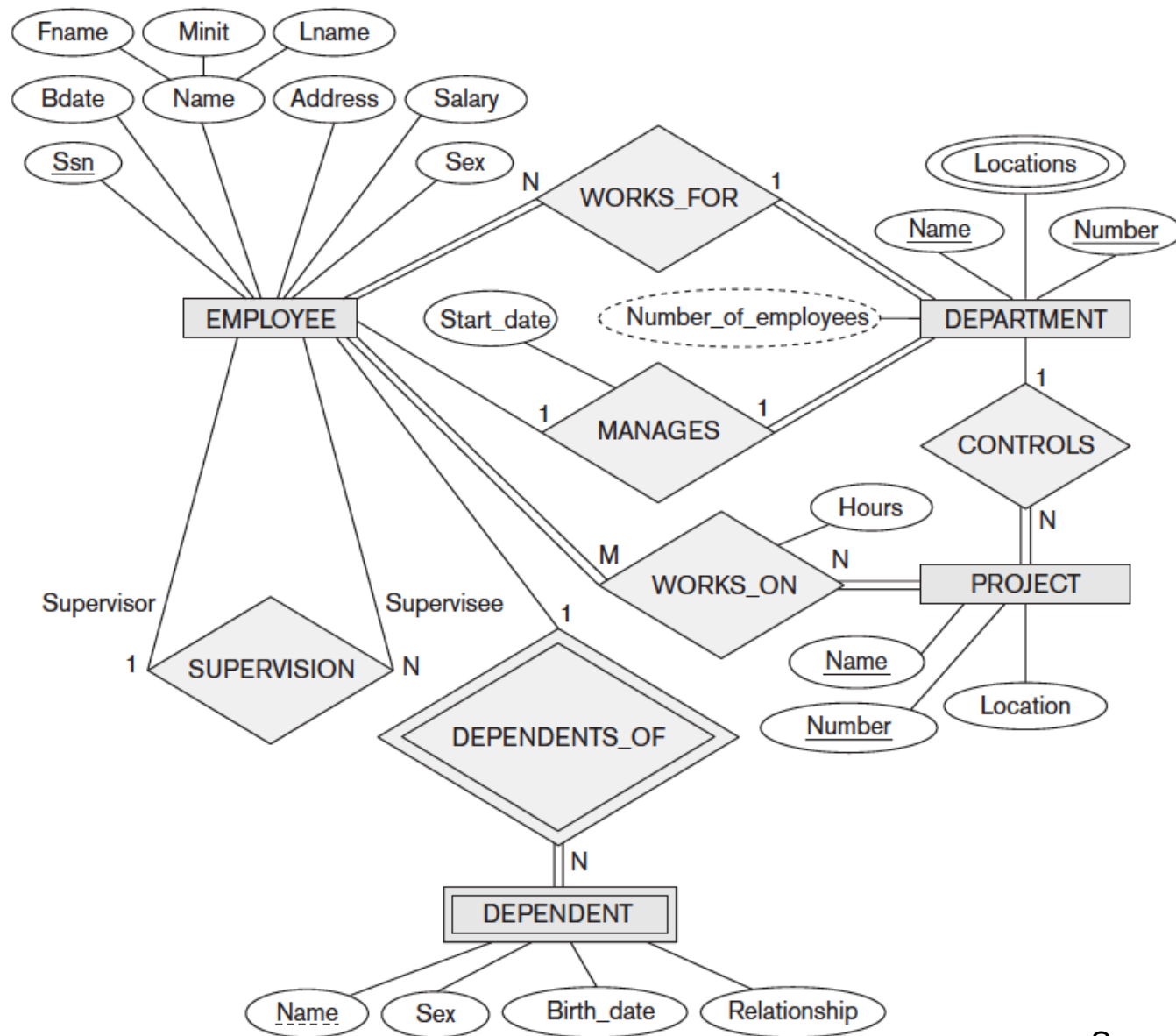
DDL statements
SDL statements

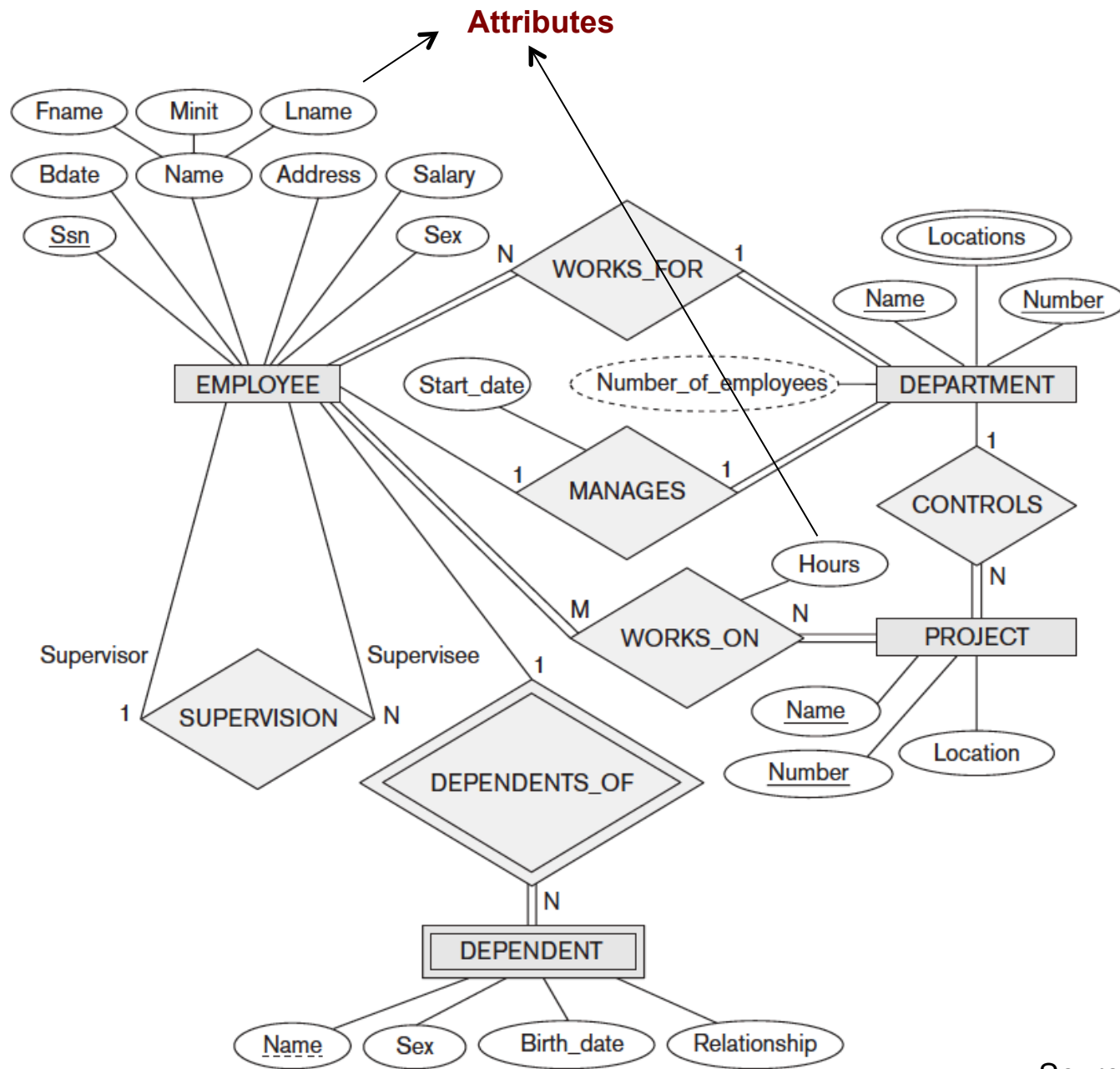
Transaction and application implementation



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.





Entities:

EMPLOYEE,
DEPARTMENT,
PROJECT,
DEPENDENT

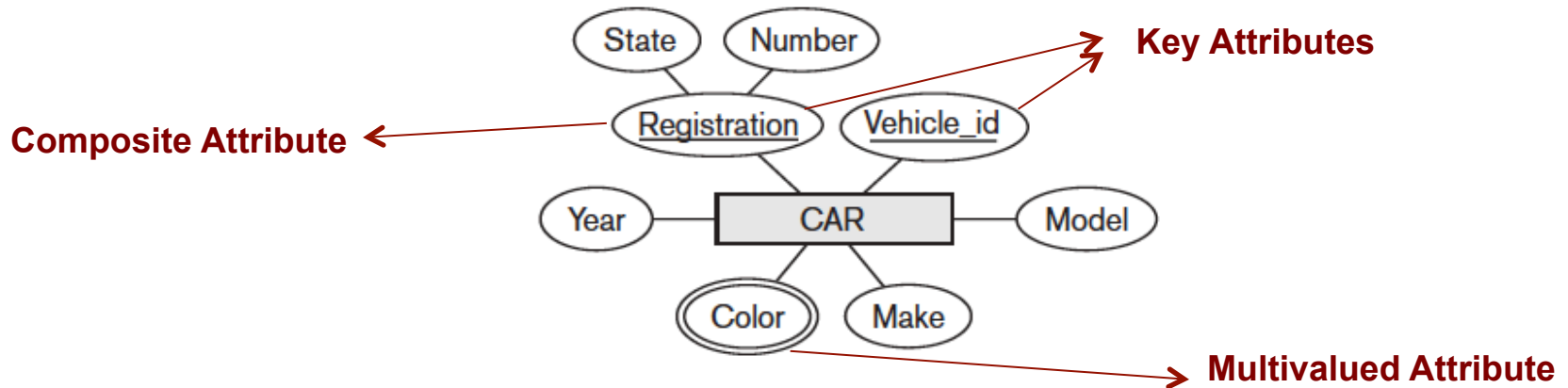
Relationships:

WORKS_FOR,
MANAGES,
CONTROLS,
WORKS_ON,
DEPENDENT_OF,
SUPERVISION

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



CAR
Registration (Number, State), Vehicle_id, Make, Model, Year, {Color}

CAR₁
((ABC 123, TEXAS), TK629, Ford Mustang, convertible, 2004 {red, black})

CAR₂
((ABC 123, NEW YORK), WP9872, Nissan Maxima, 4-door, 2005, {blue})

CAR₃
((VSY 720, TEXAS), TD729, Chrysler LeBaron, 4-door, 2002, {white, blue})

⋮

ER Model – Domains of Attributes

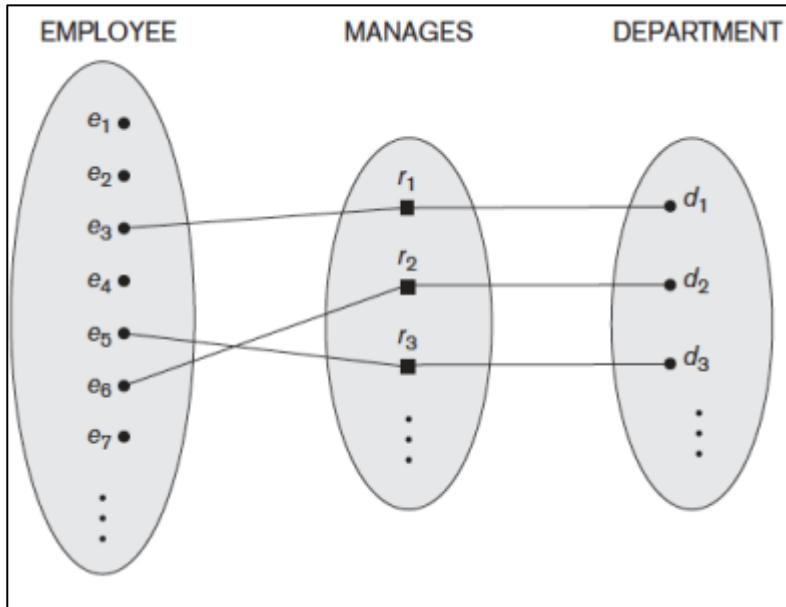
- ✓ **Value Sets (Domains) of Attributes:** Each simple attribute of an entity is associated with a value set (or domain of values), which specifies the set of values that may be assigned to that attribute for each individual entity.
 - ✓ Example: the range of ages allowed for employees is between 16 and 70 => set of **integer** numbers between 16 and 70.
 - ✓ integer, string, Boolean, float, enumerated type, etc...

ER Model – Relationships

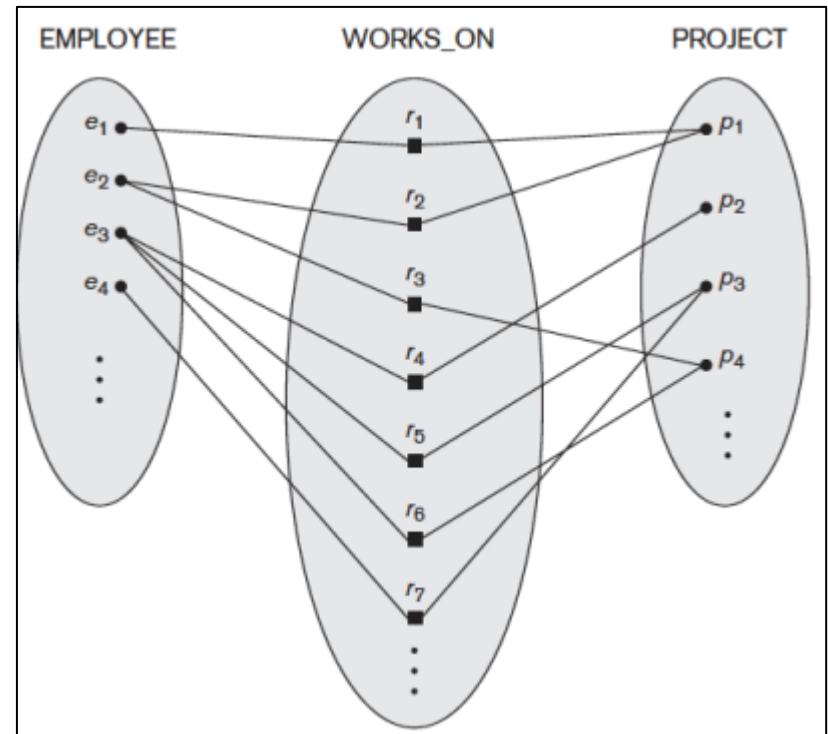
- ✓ **Degree:** number of participating entities. Binary: between two entities, ternary: among three entities, and so on.
- ✓ **Recursive** relationships: between two instances of the same entity. Ex. SUPERVISION
- ✓ **Cardinality** ratios for binary relationships: 1:1, 1:N, N:1, and M:N.
 - ✓ Ex: MANAGES (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

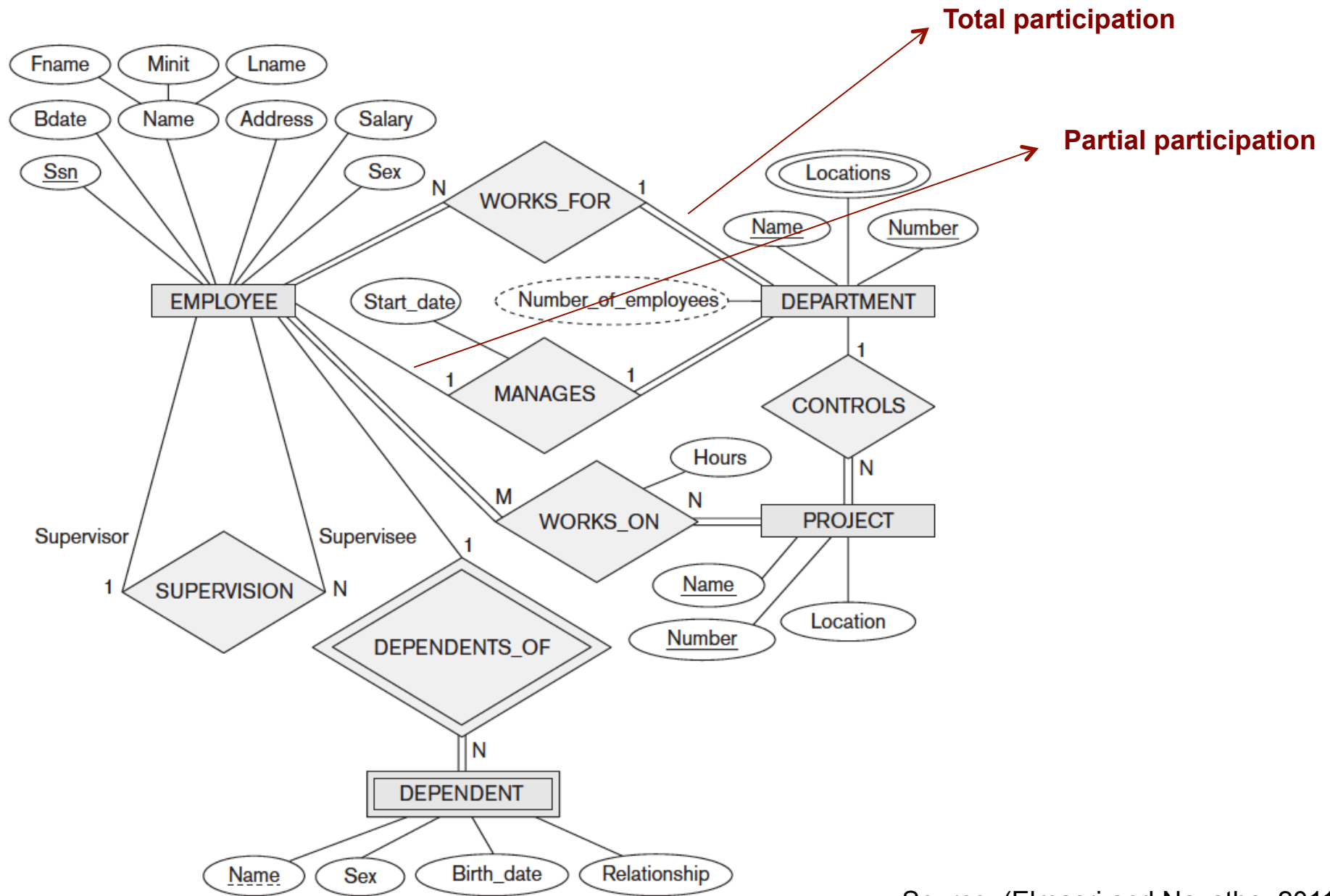


Source: (Elmasri and Navathe, 2011)

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.

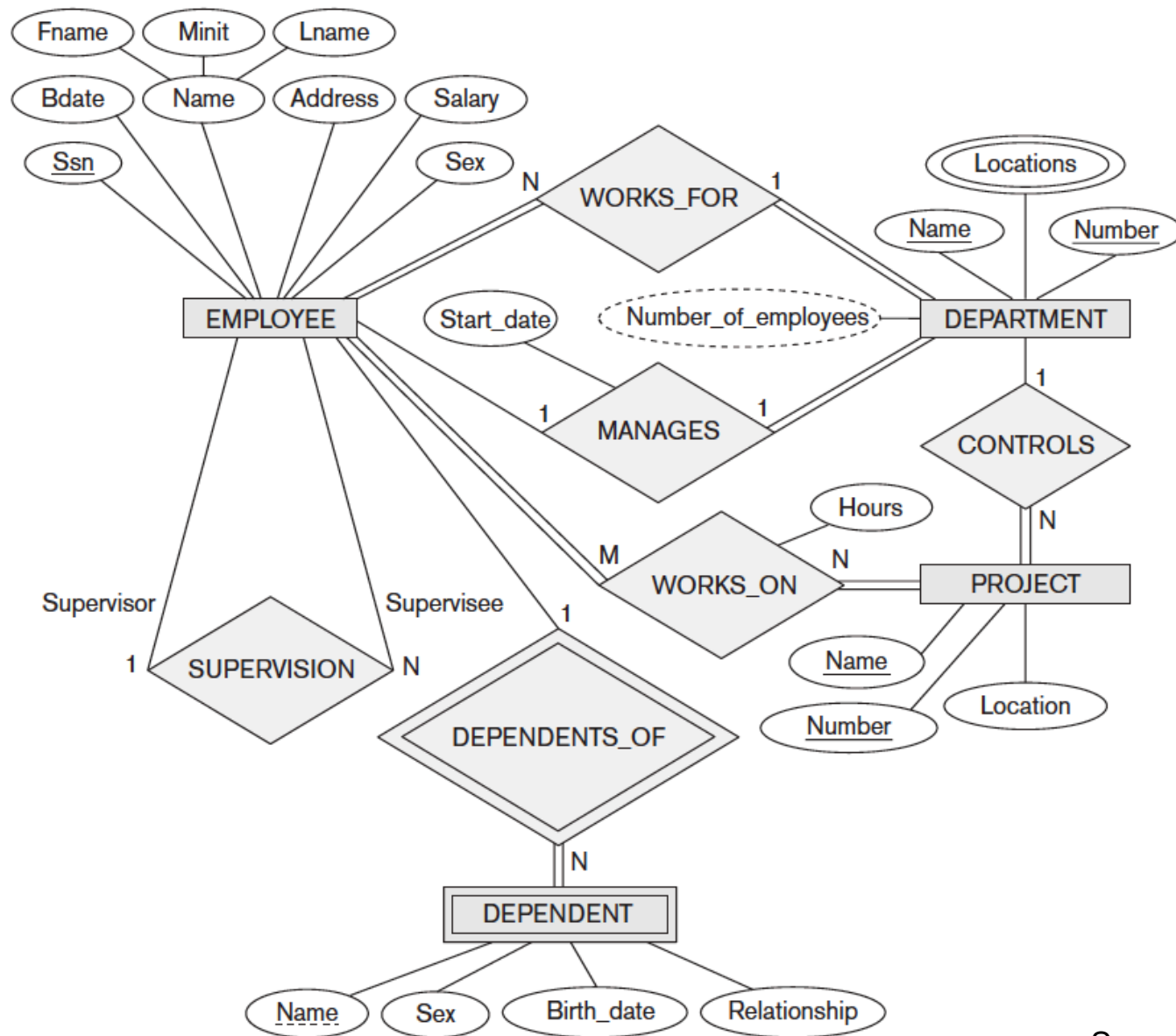
In ER diagrams, total participation (or existence dependency) is displayed as a *double line* connecting the participating entity type to the relationship, whereas partial participation is represented by a single line.



ER Model – Weak Entity


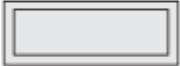






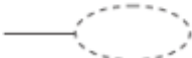
- ✓ **Weak entity:** entity that do not have key attributes of their own. A weak entity type always has a *total participation constraint* (existence dependency) with respect to its **identifying relationship** because a weak entity cannot be identified without an owner entity. A weak entity type normally 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.



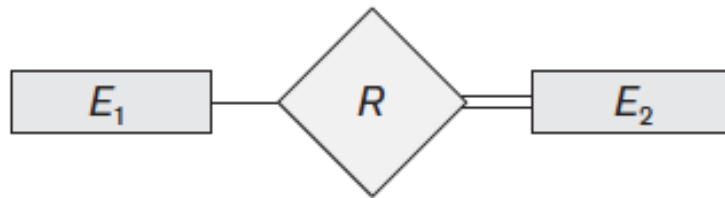
ER Model – Notations

Source: (Elmasri and Navathe, 2011)

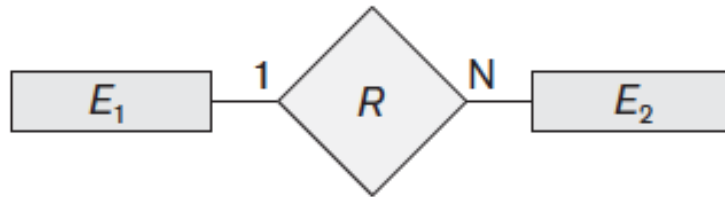
Symbol	Meaning
	Entity
	Weak Entity
	Relationship
	Identifying Relationship
	Attribute
	Key Attribute
	Multivalued Attribute
	Composite Attribute
	Derived Attribute

ER Model – Notations

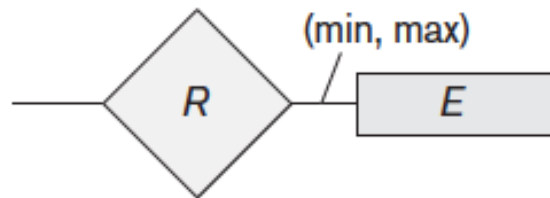
Source: (Elmasri and Navathe, 2011)



Total Participation of E_2 in R

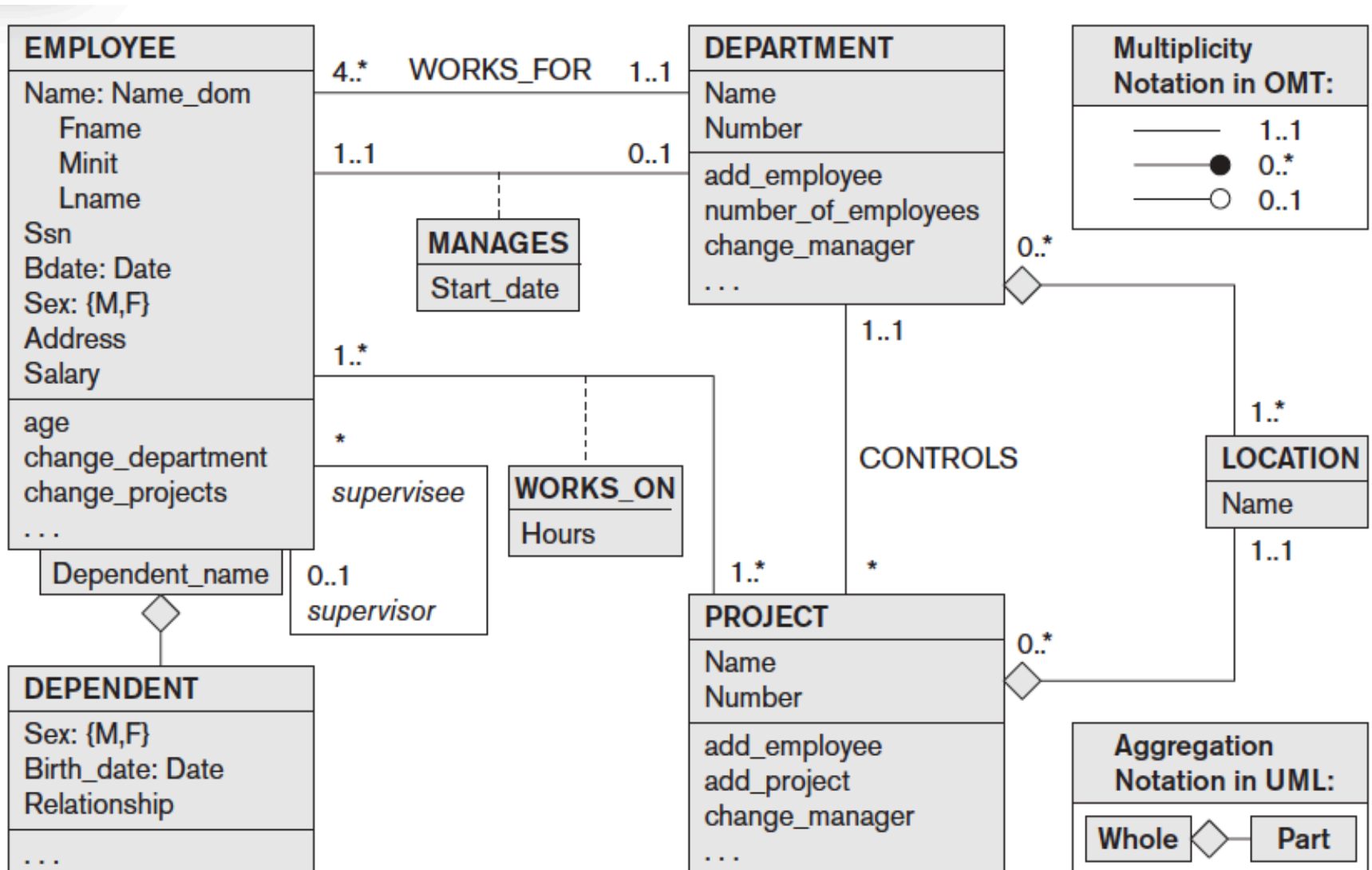


Cardinality Ratio 1: N for $E_1:E_2$ in R



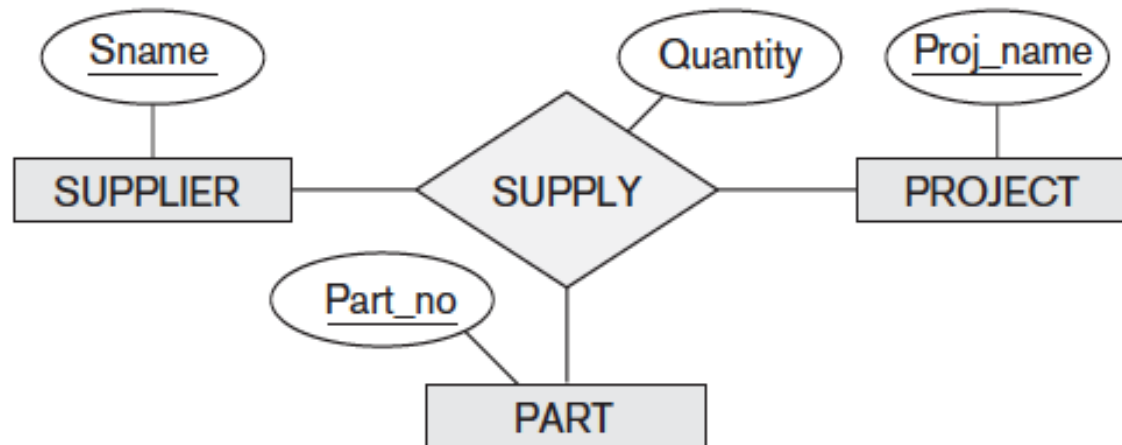
Structural Constraint (min, max)
on Participation of E in R

ER Model – UML Diagram



Source: (Elmasri and Navathe, 2011)

ER Model – Ternary Relationship

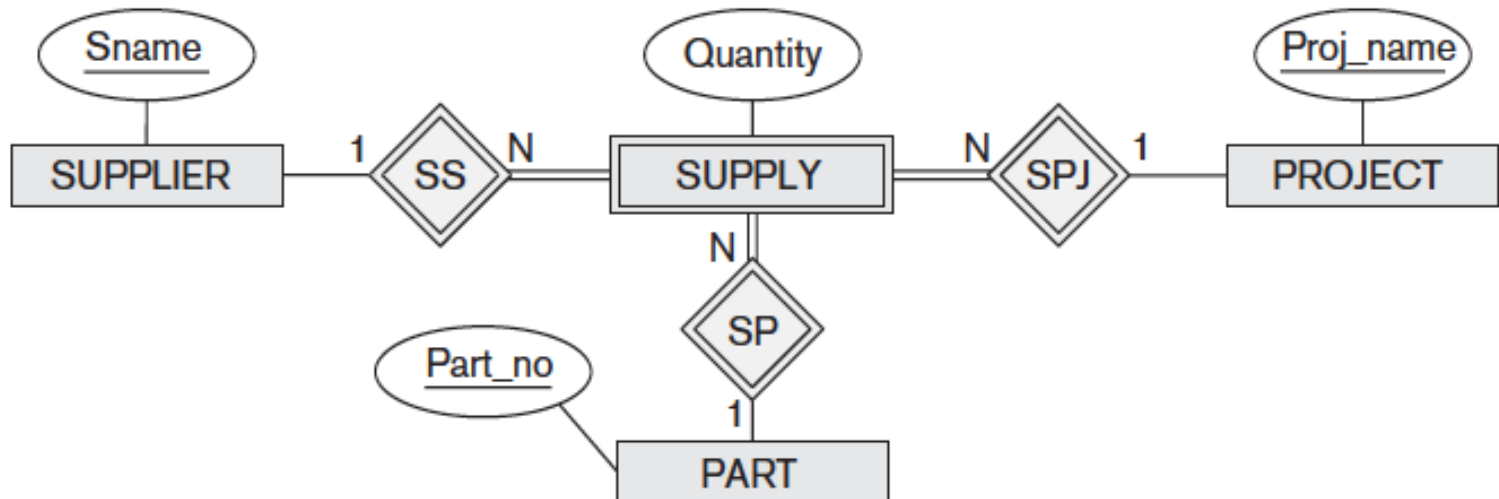
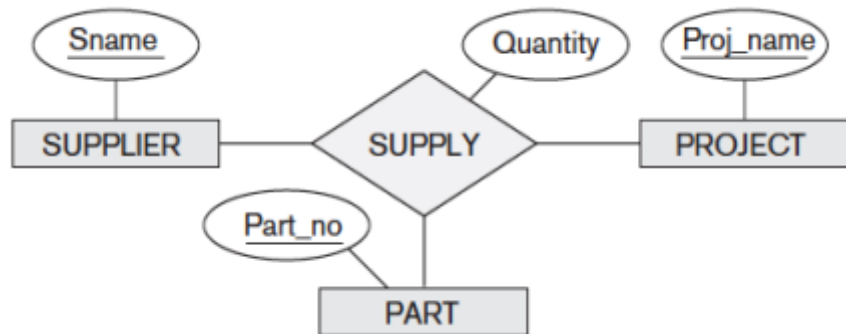


Source: (Elmasri and Navathe, 2011)

ER Model – Ternary Relationship

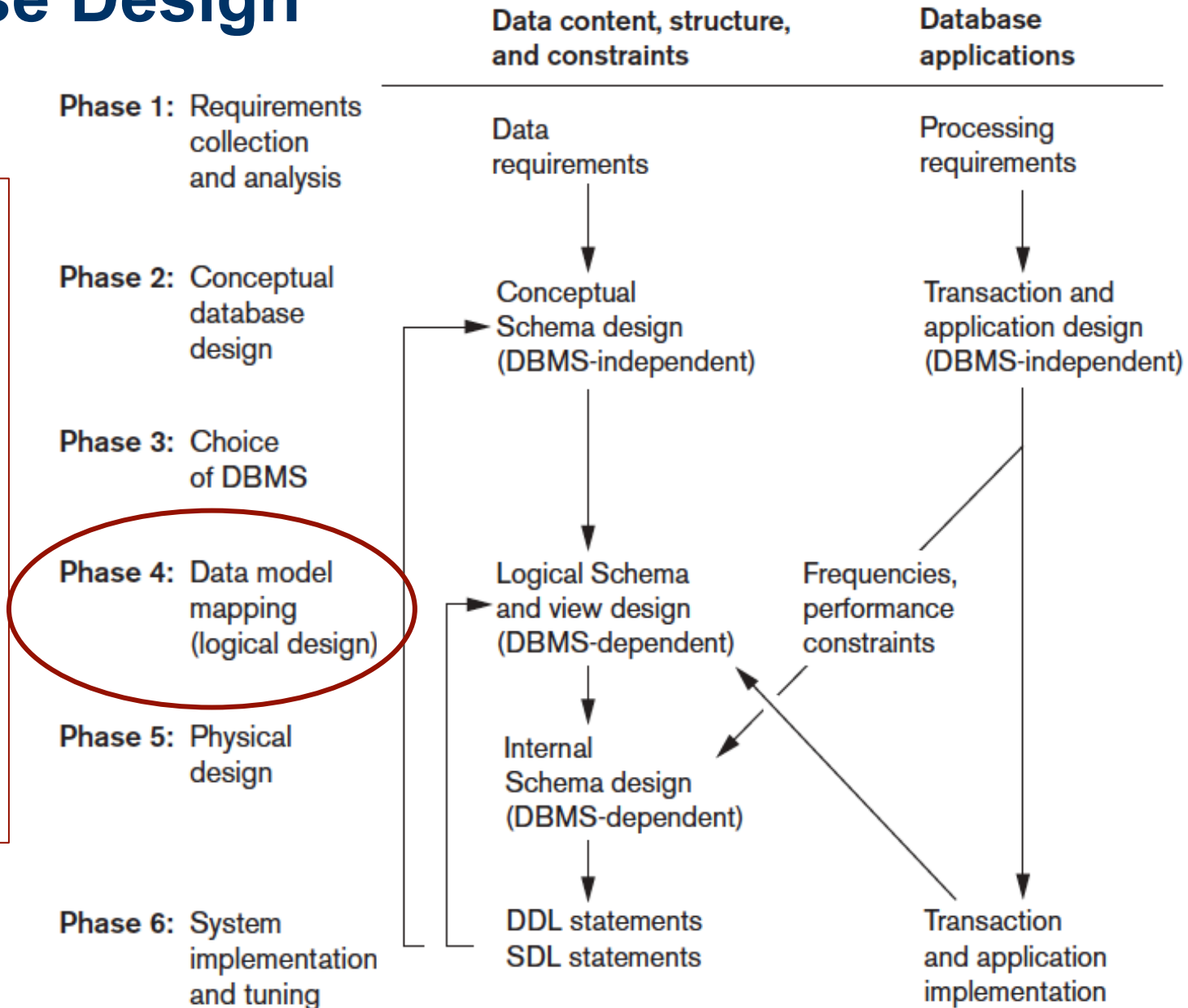
- ✓ Some database design tools are based on variations of the ER model that permit only binary relationships.
- ✓ A ternary relationship such as SUPPLY must be represented as a weak entity type, with no partial key and with three identifying relationships.

ER Model – Ternary Relationship



Database Design

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

IBM

DB2.

Universal
Database

ORACLE®



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.

The diagram illustrates the components of a relation. The 'Relation Name' is 'STUDENT'. The 'Attributes' are 'Name', 'Ssn', 'Home_phone', 'Address', 'Office_phone', 'Age', and 'Gpa'. The 'Tuples' are the rows of data in the table.

Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	(817)373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	(817)375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	(817)749-1253	25	3.53
Rohan Panchal	489-22-1100	(817)376-9821	265 Lark Lane	(817)749-6492	28	3.93
Barbara Benson	533-69-1238	(817)839-8461	7384 Fontana Lane	NULL	19	3.25

Relation - Definition

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

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

- ✓ A set (d_1, d_2, \dots, 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

EMPLOYEE

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

Schema

Instances

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

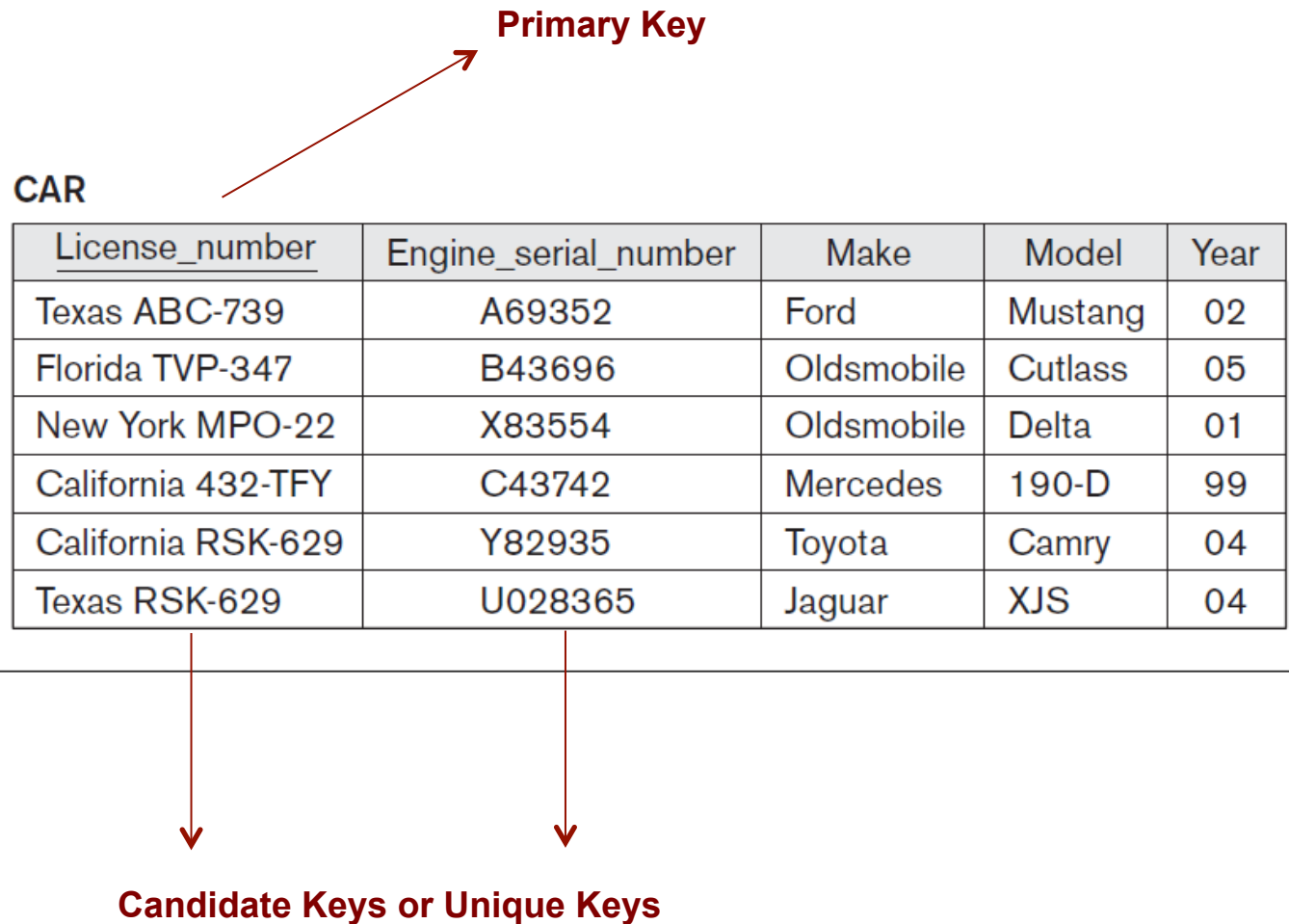


Figure 3.4

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

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.

EMPLOYEE

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

DEPARTMENT

Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

<u>Dnumber</u>	<u>Dlocation</u>
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

<u>Essn</u>	<u>Pno</u>	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

PROJECT

Pname	<u>Pnumber</u>	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

<u>Essn</u>	<u>Dependent_name</u>	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

Primary Keys ?

Unique Keys ?

Foreign Keys ?

Source: (Elmasri and Navathe, 2011)

Foreign Keys

EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno
-------	-------	-------	------------	-------	---------	-----	--------	-----------	-----

DEPARTMENT

Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date
-------	----------------	---------	----------------

DEPT_LOCATIONS

<u>Dnumber</u>	<u>Dlocation</u>
----------------	------------------

PROJECT

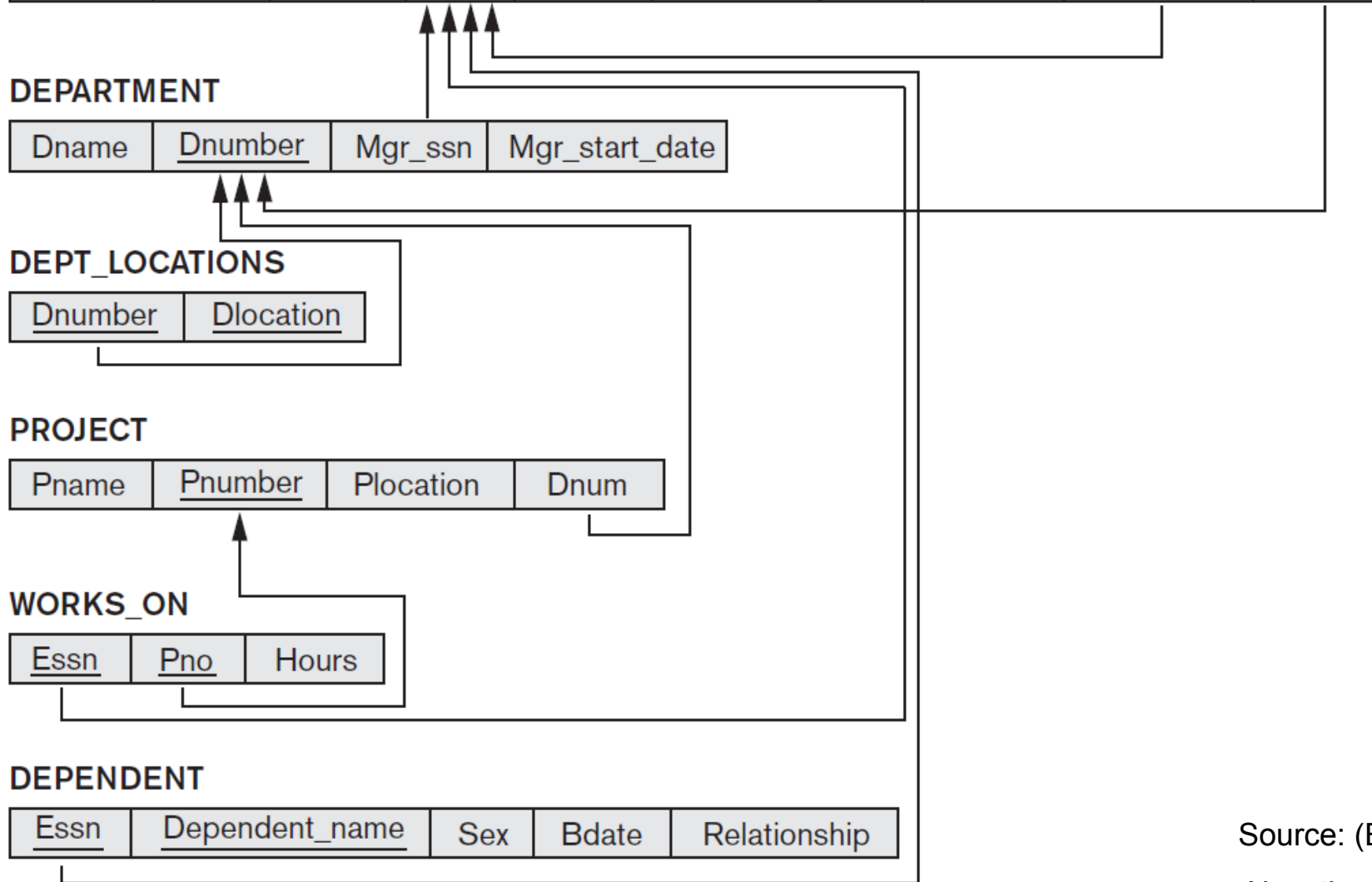
Pname	<u>Pnumber</u>	Plocation	Dnum
-------	----------------	-----------	------

WORKS_ON

<u>Essn</u>	<u>Pno</u>	Hours
-------------	------------	-------

DEPENDENT

<u>Essn</u>	<u>Dependent_name</u>	Sex	Bdate	Relationship
-------------	-----------------------	-----	-------	--------------



Source: (Elmasri and Navathe, 2011)

Foreign Keys

EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno
-------	-------	-------	------------	-------	---------	-----	--------	-----------	-----

DEPARTMENT

Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date
-------	----------------	---------	----------------

DEPT_LOCATIONS

<u>Dnumber</u>	<u>Dlocation</u>
----------------	------------------

PROJECT

Pname	<u>Pnumber</u>	Plocation	Dnum
-------	----------------	-----------	------

WORKS_ON

<u>Essn</u>	<u>Pno</u>	Hours
-------------	------------	-------

DEPENDENT

<u>Essn</u>	<u>Dependent_name</u>	Sex	Bdate	Relationship
-------------	-----------------------	-----	-------	--------------

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 .
 - *Example:* Insert <'Cecilia', 'F', 'Kolonsky', NULL, '1960-04-05', '6357 Windy Lane, Katy,TX', F, 28000, NULL, 4> into EMPLOYEE.
 - *Result:* This insertion violates the entity integrity constraint (NULL for the primary key Ssn), so it is rejected.

Relational Model – Operations

- **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

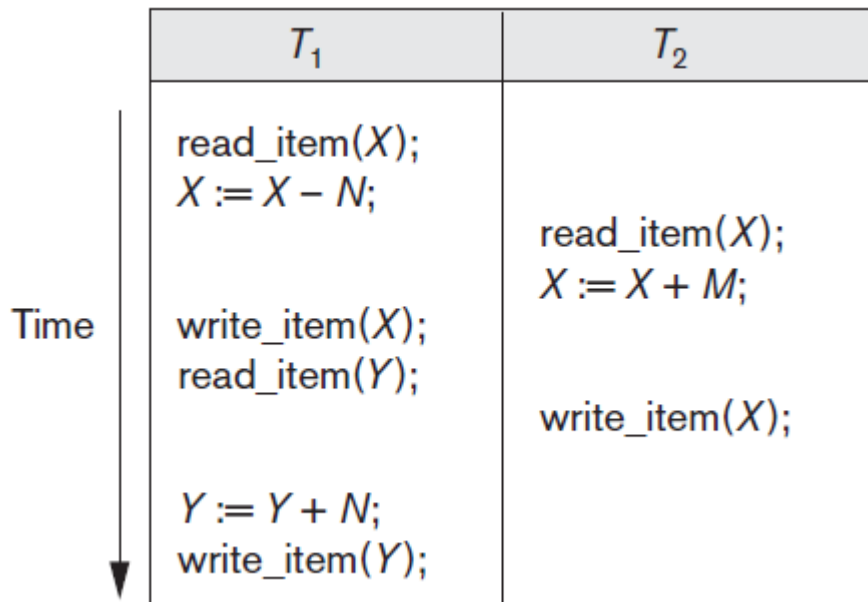
- **Update (or Modify):** change the values of one or more attributes in a tuple (or tuples) of some relation R.
 - ✓ *Example:* Update the Dno of the EMPLOYEE tuple with Ssn = '999887777' to 7.
 - ✓ *Result:* Unacceptable, because it violates referential integrity.

Transaction

- **A transaction** is an executing program that forms a logical unit of database processing that must be completed in its entirety to ensure correctness.
- 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

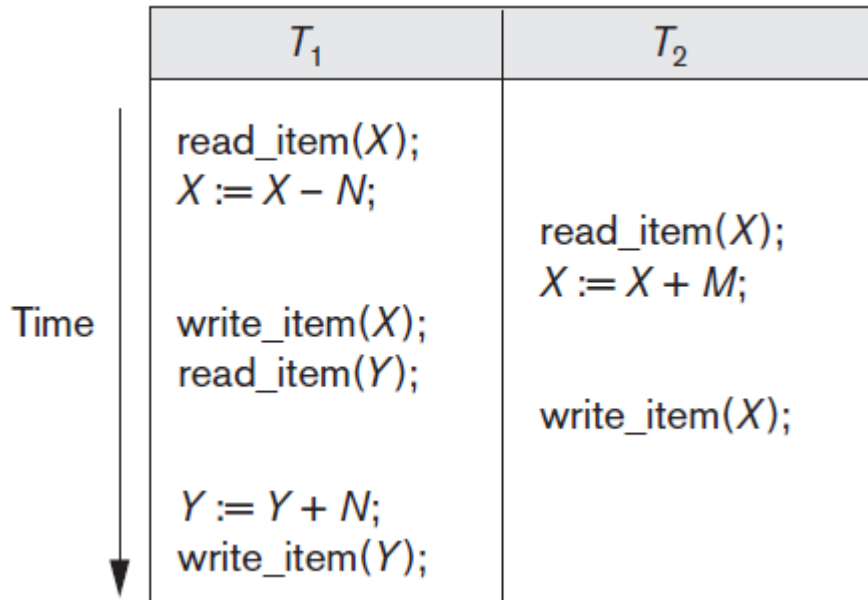
Why concurrency control is need?

Two transactions: T1 and T2



Why concurrency control is need?

Two transactions: T1 and T2

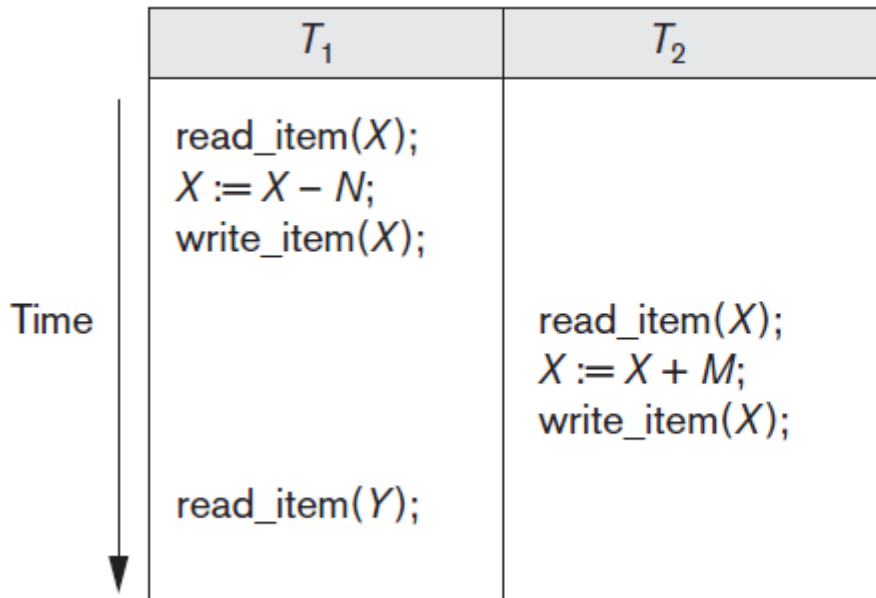


The Lost Update Problem: occurs when two transactions that access the same database items have their operations interleaved in a way that makes the value of some database items incorrect.

← Item X has an incorrect value because its update by T_1 is *lost* (overwritten).

Why concurrency control is need?

Two transactions: T1 and T2



The Temporary Update (or Dirty Read) Problem: occurs when one transaction updates a database item and then the transaction fails for some reason.

Transaction T_1 fails and must change the value of X back to its old value; meanwhile T_2 has read the *temporary* incorrect value of X .

Why recovery from failures is need?

- The DBMS is responsible for making sure that either all the operations in the transaction are completed successfully and their effect is recorded permanently in the database, or that the transaction does not have any effect on the database or any other transactions.
- If a transaction fails after executing some of its operations but before executing all of them, the operations already executed must be **undone** and have no lasting effect.
- Types of failues: computer failure, transaction or system error, disk failure, physical problems, etc.

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)

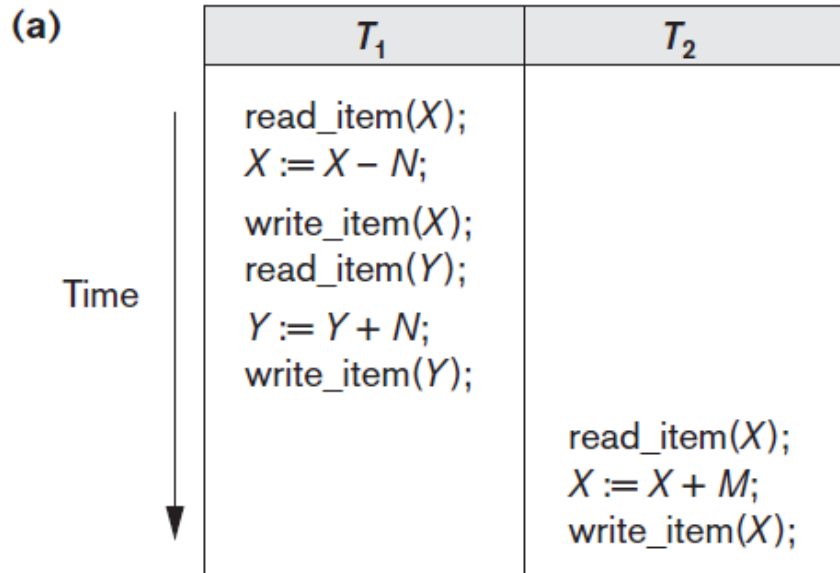
Transaction – ACID properties

- **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 an 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.

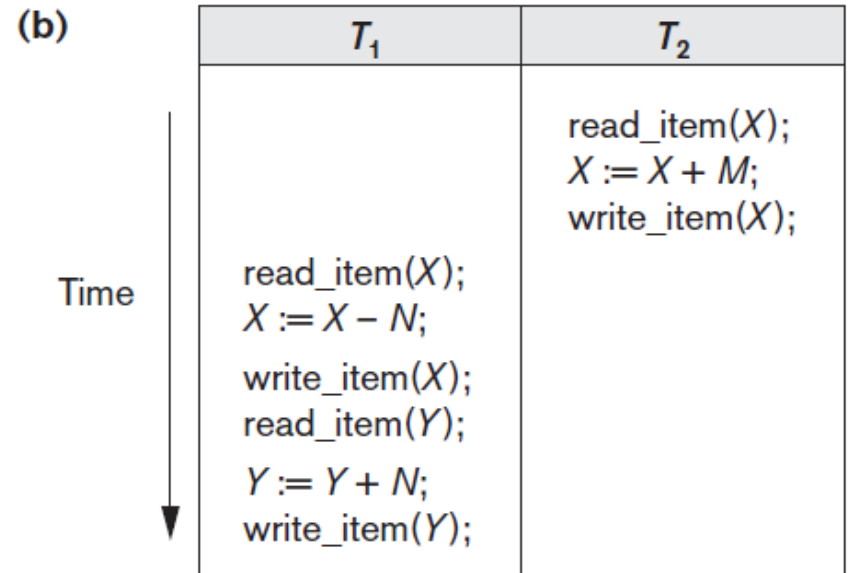
Transaction – Serial Schedule

Serial Schedule: when no interleaving of operations is permitted.

(a) Serial schedule A: T1 followed by T2. (b) Serial schedule B: T2 followed by T1.



Schedule A

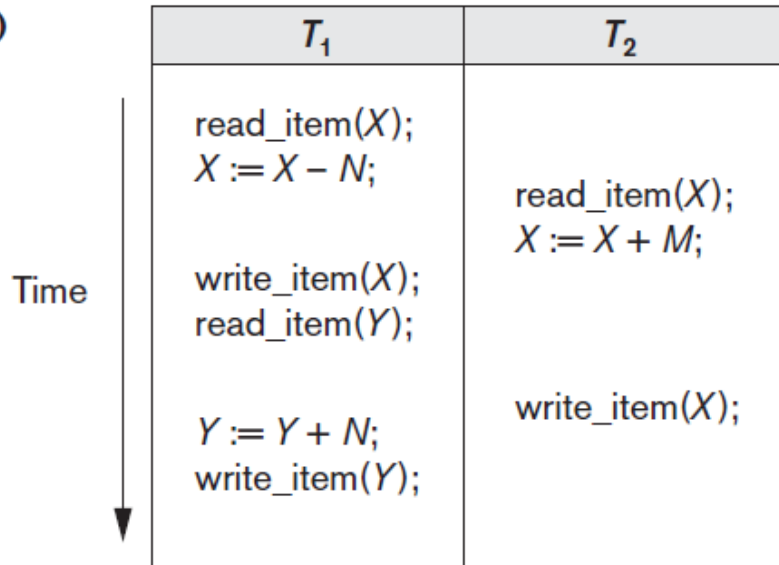


Schedule B

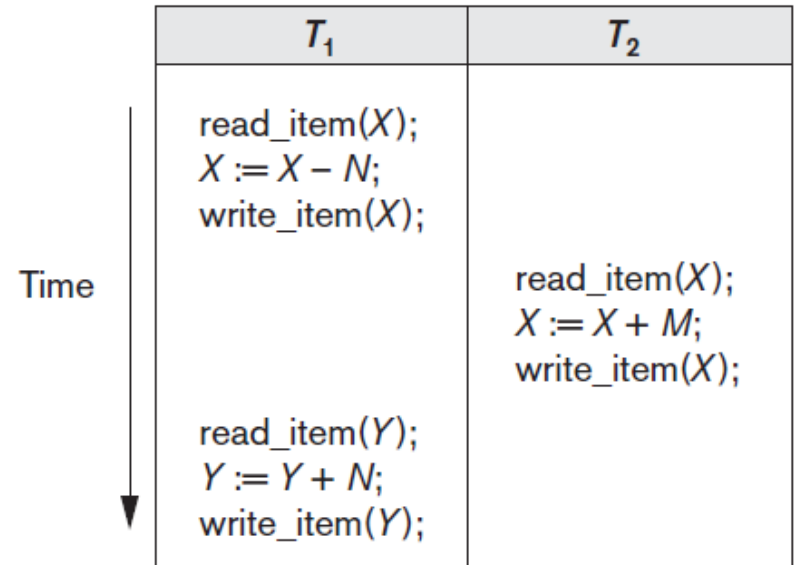
Transaction – Nonserial Schedule

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

(c)



Schedule C

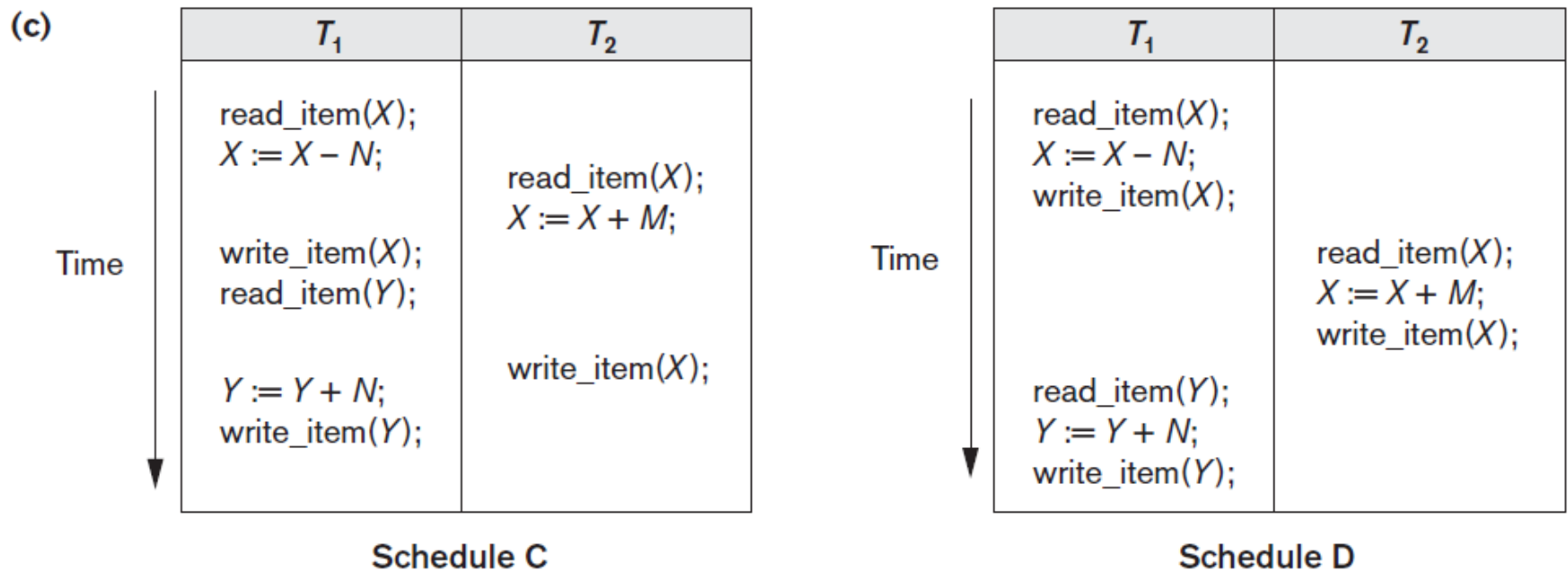


Schedule D

Correct results?

Transaction – Nonserial Schedule

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



C: erroneous result (*lost update problem*) => T_2 reads the value of X before it is changed by T_1 .

D: correct result

Transaction – SQL

```
EXEC SQL WHENEVER SQLERROR GOTO UNDO;
EXEC SQL SET TRANSACTION
    READ WRITE
    DIAGNOSTIC SIZE 5
    ISOLATION LEVEL SERIALIZABLE;
EXEC SQL INSERT INTO EMPLOYEE (Fname, Lname, Ssn, Dno, Salary)
    VALUES ('Robert', 'Smith', '991004321', 2, 35000);
EXEC SQL UPDATE EMPLOYEE
    SET Salary = Salary * 1.1 WHERE Dno = 2;
EXEC SQL COMMIT;
GOTO THE_END;
UNDO: EXEC SQL ROLLBACK;
THE_END: ... ;
```

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.

SQL

SQL-DDL (Data Definition Language)

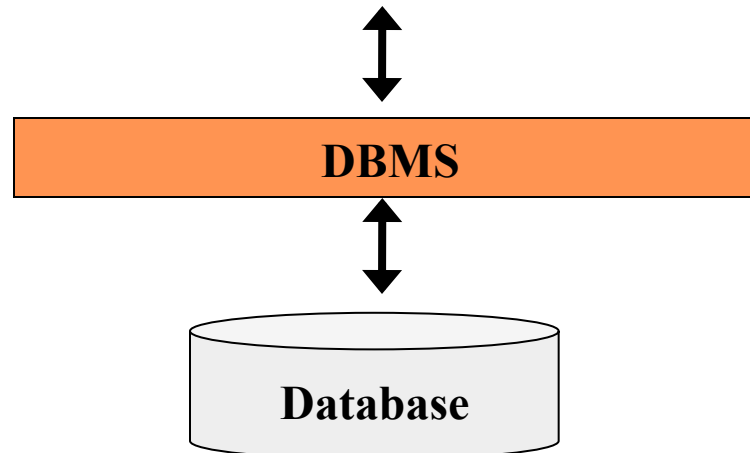
```
CREATE DATABASE Test
```

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

SQL-DML (Data Manipulation Language)

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

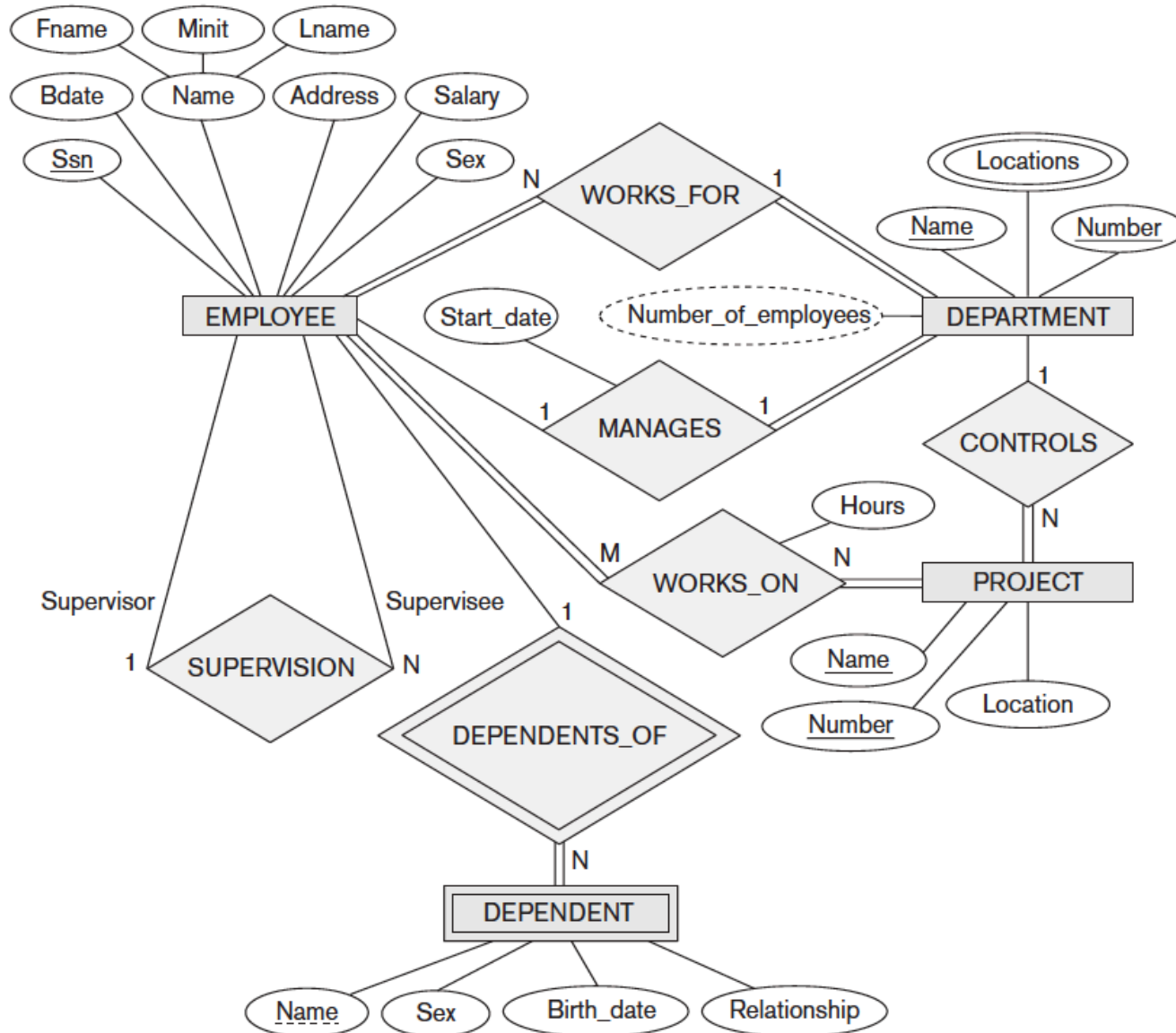
```
SELECT *  
FROM    States  
WHERE   UF = 'MG'
```



SQL – Some Commands

Commands	Description	Type
<i>select</i>	Select data	DML
<i>insert, update, delete</i>	Add, modify and remove data	DML
<i>commit, rollback</i>	Transaction controls	DDL
<i>create, alter, drop</i>	create, alter and eliminate schemas and tables	DDL

From ER Model To Relational Model



From ER Model To Relational Model

Relational
Model

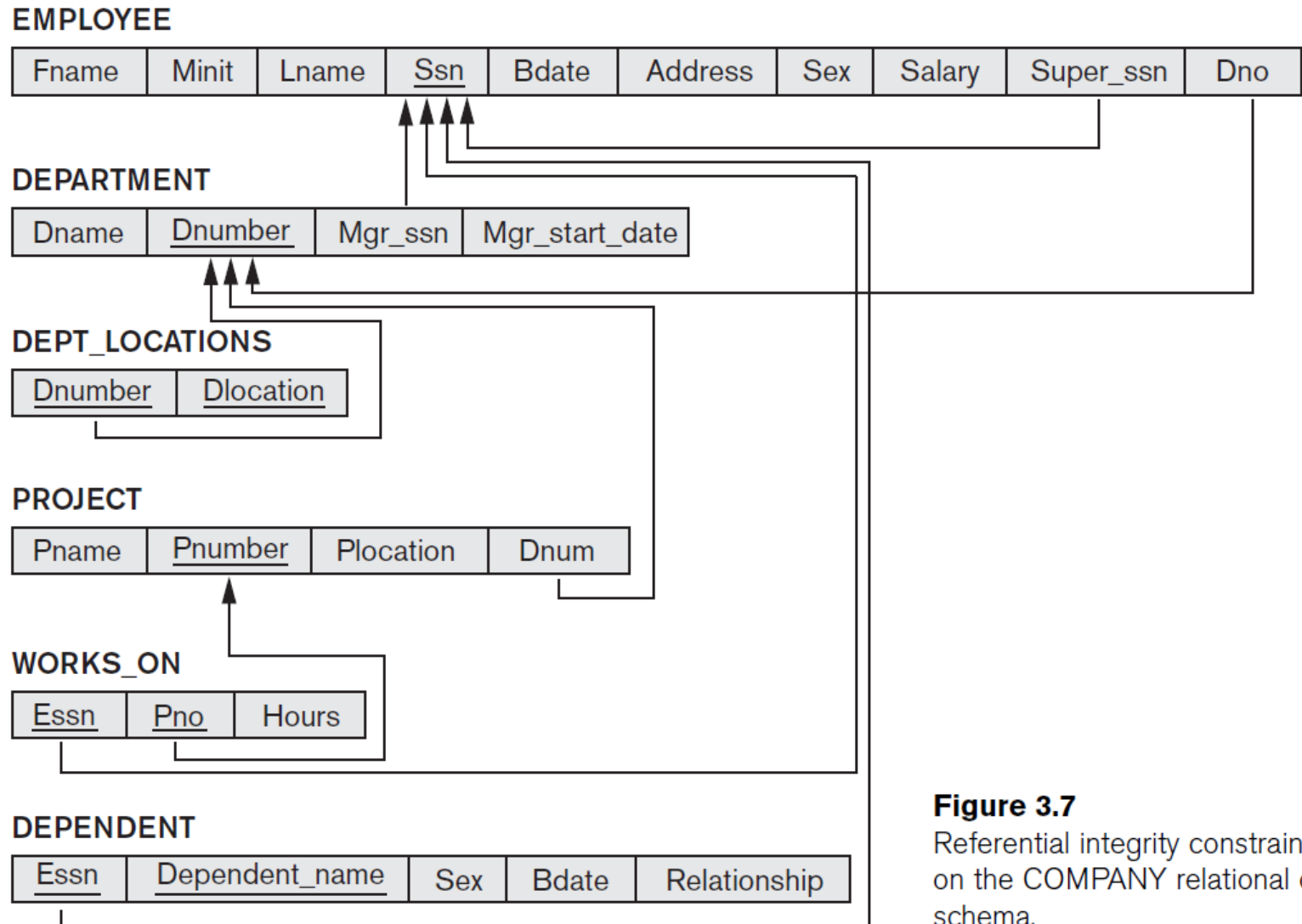
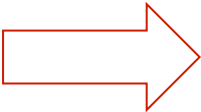


Figure 3.7
Referential integrity constraint
on the COMPANY relational c
schema.

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

ER Model To Relational Model

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.

ER Model To Relational Model

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).

ER Model To Relational Model

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.

ER Model To Relational Model

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

ER MODEL

Entity type

1:1 or 1:N relationship type

M:N relationship type

n -ary relationship type

Simple attribute

Composite attribute

Multivalued attribute

Value set

Key attribute

RELATIONAL MODEL

Entity relation

Foreign key (or *relationship* relation)

Relationship relation and *two* foreign keys

Relationship relation and n foreign keys

Attribute

Set of simple component attributes

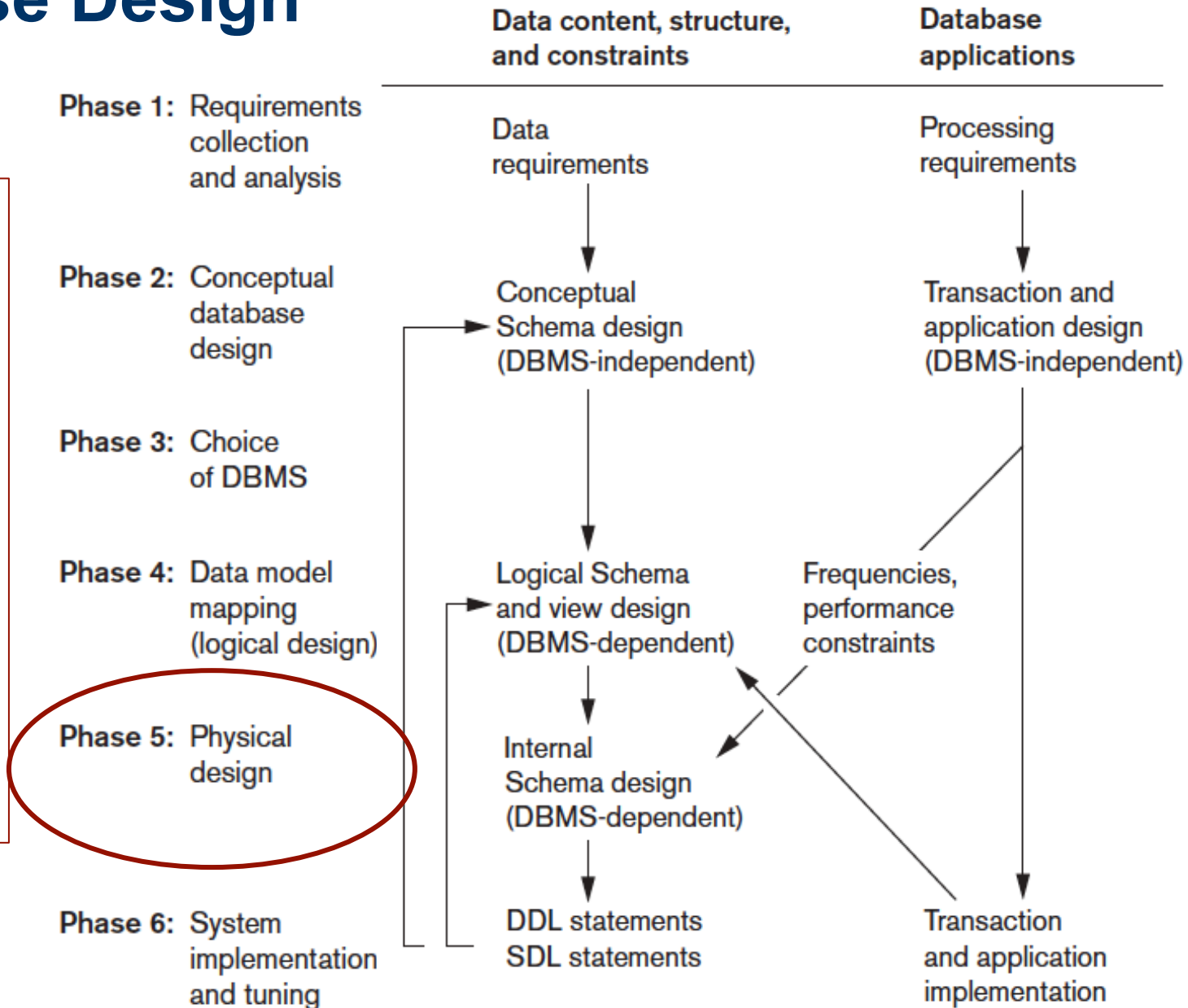
Relation and foreign key

Domain

Primary (or secondary) key

Database Design

Specifications
for the stored
database in terms
of physical file
storage structures,
buffer, and
indexes.



Database Design

Database and application programs are implemented, tested, and eventually deployed for service.

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

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

Database applications

Data requirements

Processing requirements

Conceptual Schema design (DBMS-independent)

Transaction and application design (DBMS-independent)

Logical Schema and view design (DBMS-dependent)

Frequencies, performance constraints

Internal Schema design (DBMS-dependent)

DDL statements
SDL statements

Transaction and application implementation

