Database Management Systems

BITS Pilani
Hyderabad Campus

Dr.R.Gururaj
CS&IS Dept.

Lecture Session-2

DBMS Concepts

innovate achieve lead

Content

- *Describing and Storing data in DBMS*
- *Three schema Architecture*
- *Data independence*
- *Queries*
- *Transactions*
- *Structure of a DBMS*
- *People who work with DBMS*

Describing and storing data in DBMS

Data model

Is a collection of high-level data description constructs that hide many low-level details.

DBMS allows a user to define the data to be stored in terms of a data model.

Semantic data models**:** More abstract high-level data models that make it easier for a user to come up with a good initial description of the data in an enterprise. Contain wide variety of constructs that help describe a real-world enterprise data.
Ex. ER model

Representational / Implementation data models:
These are DBMS specific data models and are built around just few basic constructs.
Ex. Relational data model, Object data model

A database design in terms of a semantic model serves as a useful starting point and is subsequently translated into a database design in terms of the data model the DBMS supports.

Relational Model:

The central data description construct in this model is a relation, which can be thought of as a set of records.

Schema: Description of data in terms of a data model is called a schema. A relation schema specifies the name of the relation, field, type etc.

Ex. *Student (sid: string; name: string; age: integer)*

every row follows the schema of the relation.

Instance of a relation:

Student

sid	name	age
A120	Raju	21
A134	Kiran	19
C110	John	22

A schema can be regarded as a template for describing a student.

We can specify integrity constraints which are conditions that need to be satisfied by records in the relation. Ex. uniqueness

The following are some important representational data models (DBMS Specific)

1. ***Network Model***: Though the basic structure is a record,
   the relationships are captured using links.
   The database can be seen as an arbitrary network of records connected by links.
   Ex.: GE's Integrated Data store (IDS), in Early 1960s
2. ***Hierarchical Model***: The records containing data are organized
   as a collection of trees. Ex.: IBMs IMS (Information Management System),
   in late 1960s
3. ***Relational Model***: (early 1970s)Data & relationships are captured as tables & keys.
   Ex.: Oracle, IBMs DB2, MySQL, Informix, Sybase, MS Access, Ingress, MySQL etc.
   The basic storage structure is a record.
4. ***Object Data Model***: Objects created through object–oriented programs
   can be stored in database.
   Ex.: Object Store
5. ***Object Relational Model***: Objects can be stores in tables.
   Ex.: Oracle, Informix

Database Schema

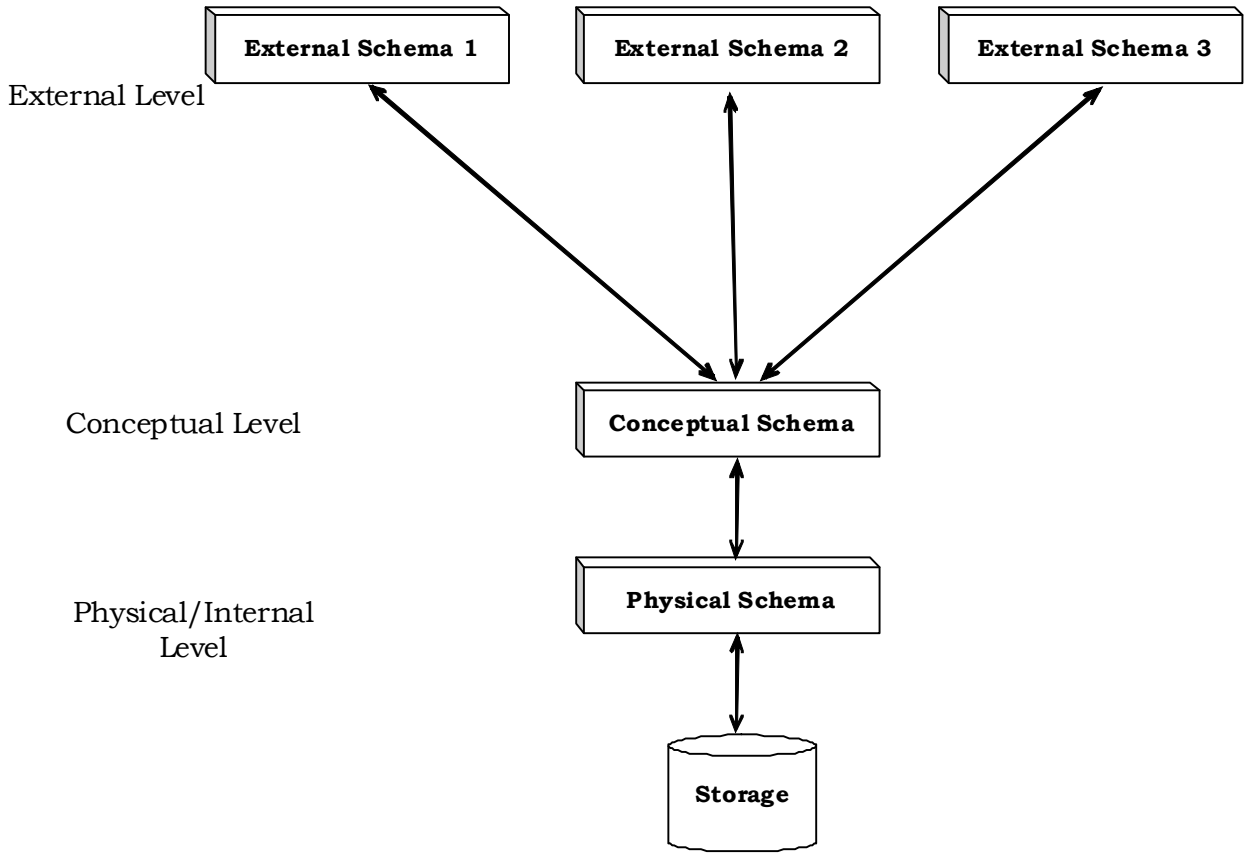
Database Schema: Description of a database is called as *database Schema*

Three-Schema Architecture

A database can be described using three different levels of abstractions. Description at each level can be defined by a schema. For each abstraction we focus on one of the specific issues such as user views, concepts, storage etc.

1. *External schema*: Used to describe the database at external level. Also described in terms of the data model of that DBMS. This allows data access to be customized at the level of individual users/groups/applications. Any external schema has one or more views and relations from the conceptual schema. This schema design is guided by end user requirements.
2. *Conceptual schema* (logical schema) Describes the stored data in terms of the data model specific to that DBMS. In RDBMS conceptual schema describes all relations that are stored in the database. Arriving at good choice of relations, fields and constraints is known as conceptual database design.
3. *Physical schema*: Describes the physical storage strategy for the database.

Three Schema Architecture



Three schema architecture of DBMS

Data Independence

innovate achieve lead

Data Independence:

The three-level architecture which is the result of the three-level abstraction on database, leads to data independence.

1. *Logical data independence:* changes in conceptual level schema should not affect the application level or external level schemas.

2. *Physical data independence:* The changes in physical features of storage, i.e., changes to the physical storage format should not affect schema at conceptual level.

The above data independence is one of the important advantages of DBMS.

The DBMS stores the description of schemas as System catalog.

Queries

innovate achieve lead

Queries in RDBMS

The ease with which information can be obtained from a database often determines its value to the user.

RDBMS allows users to pose a rich class of questions in the form of queries.

Relational data model has powerful query languages:

Formal query languages (based on strong mathematical logic)
- *Relational algebra*
- *Relational Calculus*

Commercial query language
- *SQL*

Transactions

Transaction Management

A transaction is a collection of operations that perform a single logical operation or function in a database application.

Each transaction is a unit of *atomicity*.

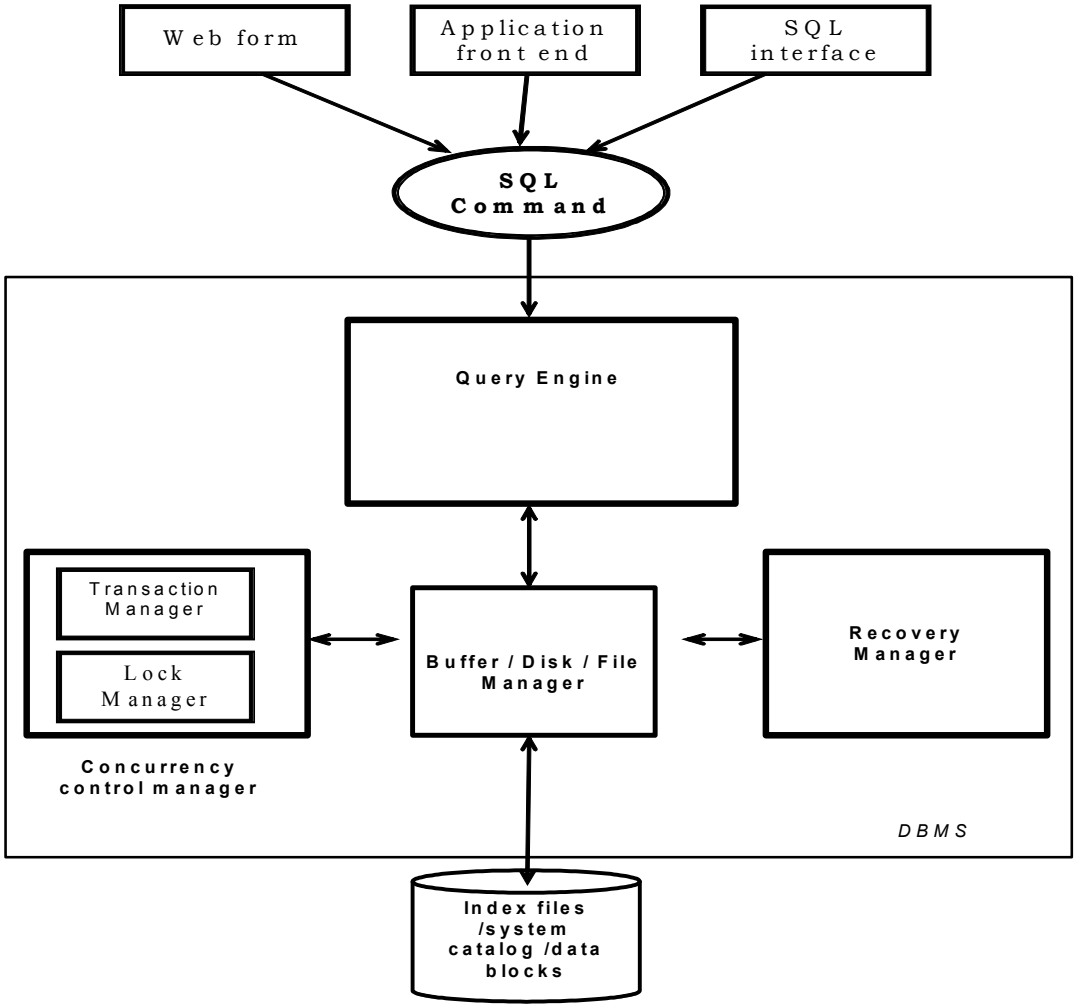
A *transaction* is an atomic unit of work that is either completed in its entirety or not done at all.

Concurrent Transactions

Incomplete Transactions and system crash

For recovery purposes, the system needs to keep track of when the transaction starts, terminates, and commits or aborts.

DBMS Structure



People who work with DBMS

- ***Database Implementers***
- ***End users***
- ***Application Programmers***
- ***Database administrator (DBA)***

DBA's role:

1. ***Design of physical & Conceptual schemas***
2. ***Security and authorization***
3. ***Data availability , recovery and backup***
4. ***Database tuning- modifying the schemas to meet the requirements***

Summary

- ✓ *How data is described in a DBMS*
- ✓ *What is a data model*
- ✓ *What is a schema*
- ✓ *What is three schema architecture of a DBMS*
- ✓ *What is data independence*
- ✓ *Queries and Query languages*
- ✓ *Transaction management*
- ✓ *Components of a DBMS*
- ✓ *People working with DBMS*

Contents

ER-to- Relational Mapping

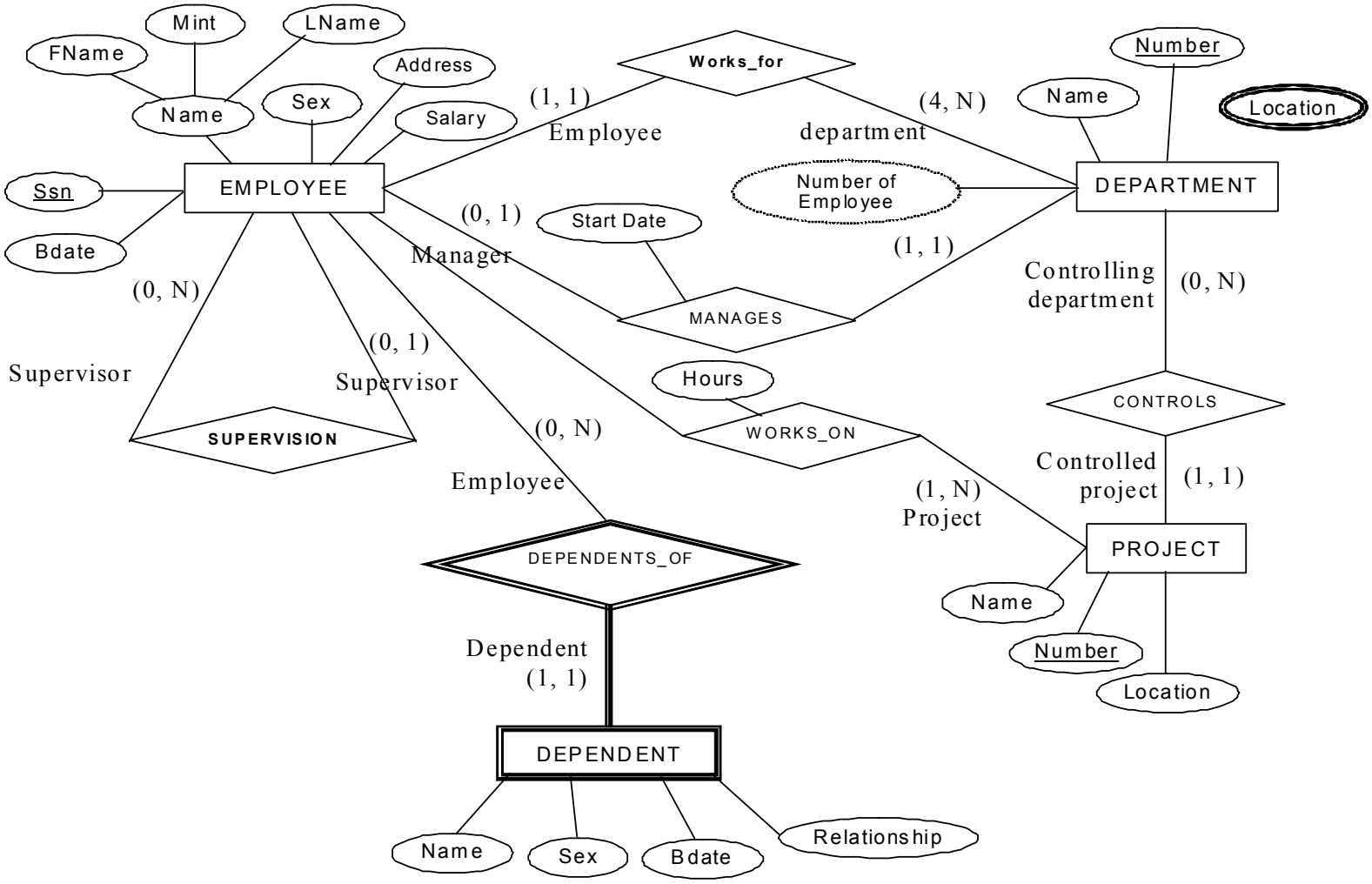
innovate achieve lead

Step 1: Mapping of Regular Entity Types.

- For each regular (strong) entity type E in the ER schema, create a relation R that includes all the simple attributes of E.
- Choose one of the key attributes of E as the primary key for R.
- If the chosen key of E is composite, the set of simple attributes that form it will together form the primary key of R.

Example: We create the relations EMPLOYEE, DEPARTMENT, and PROJECT in the relational schema corresponding to the regular entities in the ER diagram.

- SSN, DNUMBER, and PNUMBER are the primary keys for the relations EMPLOYEE, DEPARTMENT, and PROJECT as shown.


Mint
LName
FName
Address
Name
Sex
Salary
Works_for
(1, 1)
Employee
(4, N)
department
Number
Name
Location
EMPLOYEE
Ssn
Bdate
Number of Employee
DEPARTMENT
(0, 1)
Start Date
Manager
(1, 1)
Controlling department
(0, N)
(0, N)
MANAGES
(0, 1)
Supervisor
Supervisor
Hours
CONTROLS
SUPERVISION
WORKS_ON
(0, N)
Controlled project
(1, 1)
Employee
(1, N)
Project
DEPENDENTS_OF
PROJECT
Name
Number
Location
Dependent
(1, 1)
DEPENDENT
Name
Sex
Bdate
Relationship

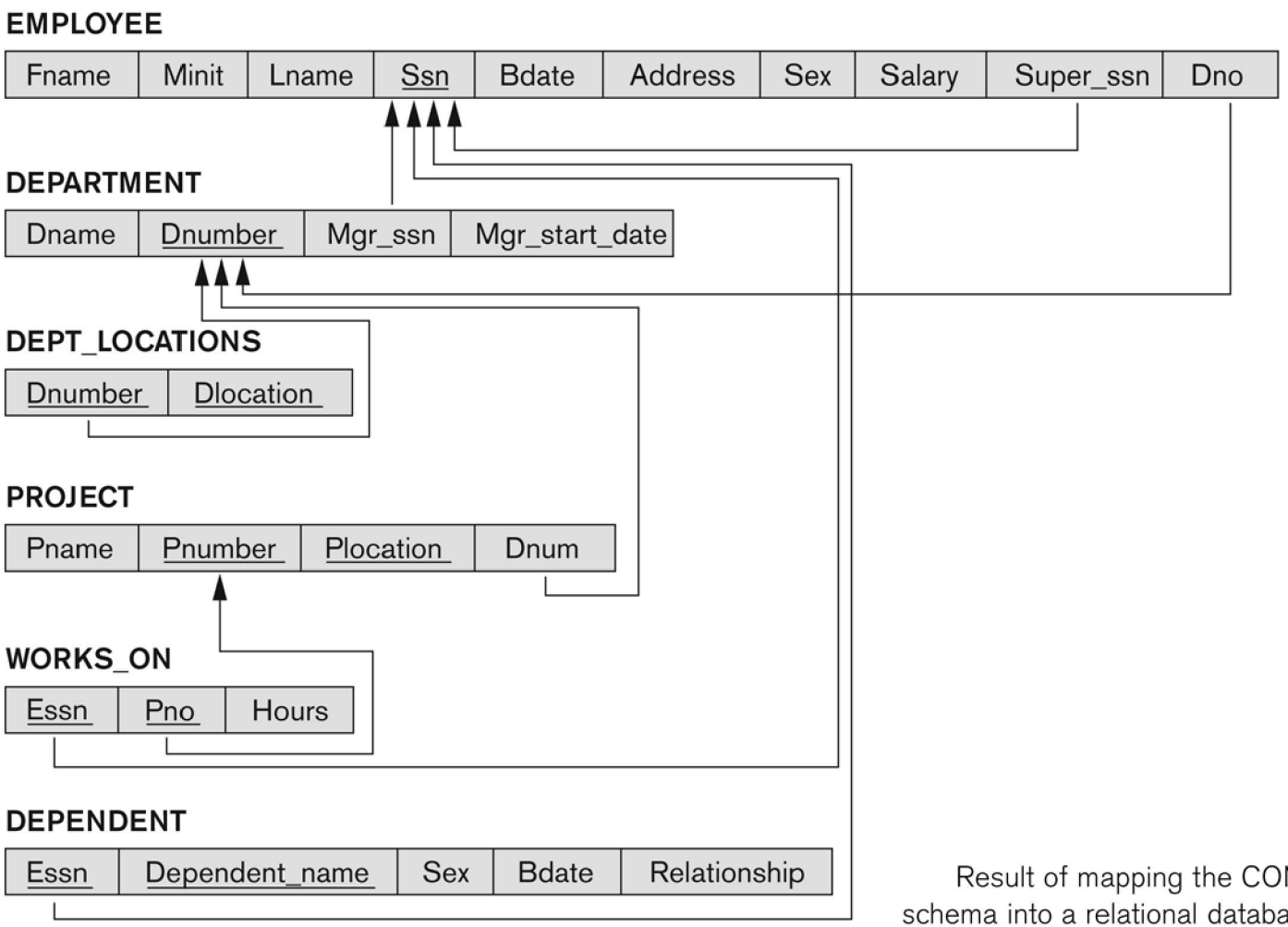


Figure 7.2
Result of mapping the COMPANY ER schema into a relational database schema.

Step 2: Mapping of Weak Entity Types

- For each weak entity type W in the ER schema with owner entity type E, create a relation R & include all simple attributes (or simple components of composite attributes) of W as attributes of R.
- Also, include as foreign key attributes of R the primary key attribute(s) of the relation(s) that correspond to the owner entity type(s).
- The primary key of R is the *combination of* the primary key(s) of the owner(s) and the partial key of the weak entity type W, if any.

Example: Create the relation DEPENDENT in this step to correspond to the weak entity type DEPENDENT.

- Include the primary key SSN of the EMPLOYEE relation as a foreign key attribute of DEPENDENT (renamed to ESSN).
- The primary key of the DEPENDENT relation is the combination {ESSN, DEPENDENT_NAME} because DEPENDENT_NAME is the partial key of DEPENDENT.

Step 3: Mapping of Binary 1:1 Relation Types

For each binary 1:1 relationship type R in the ER schema, identify the relations S and T that correspond to the entity types participating in R.

There are three possible approaches:

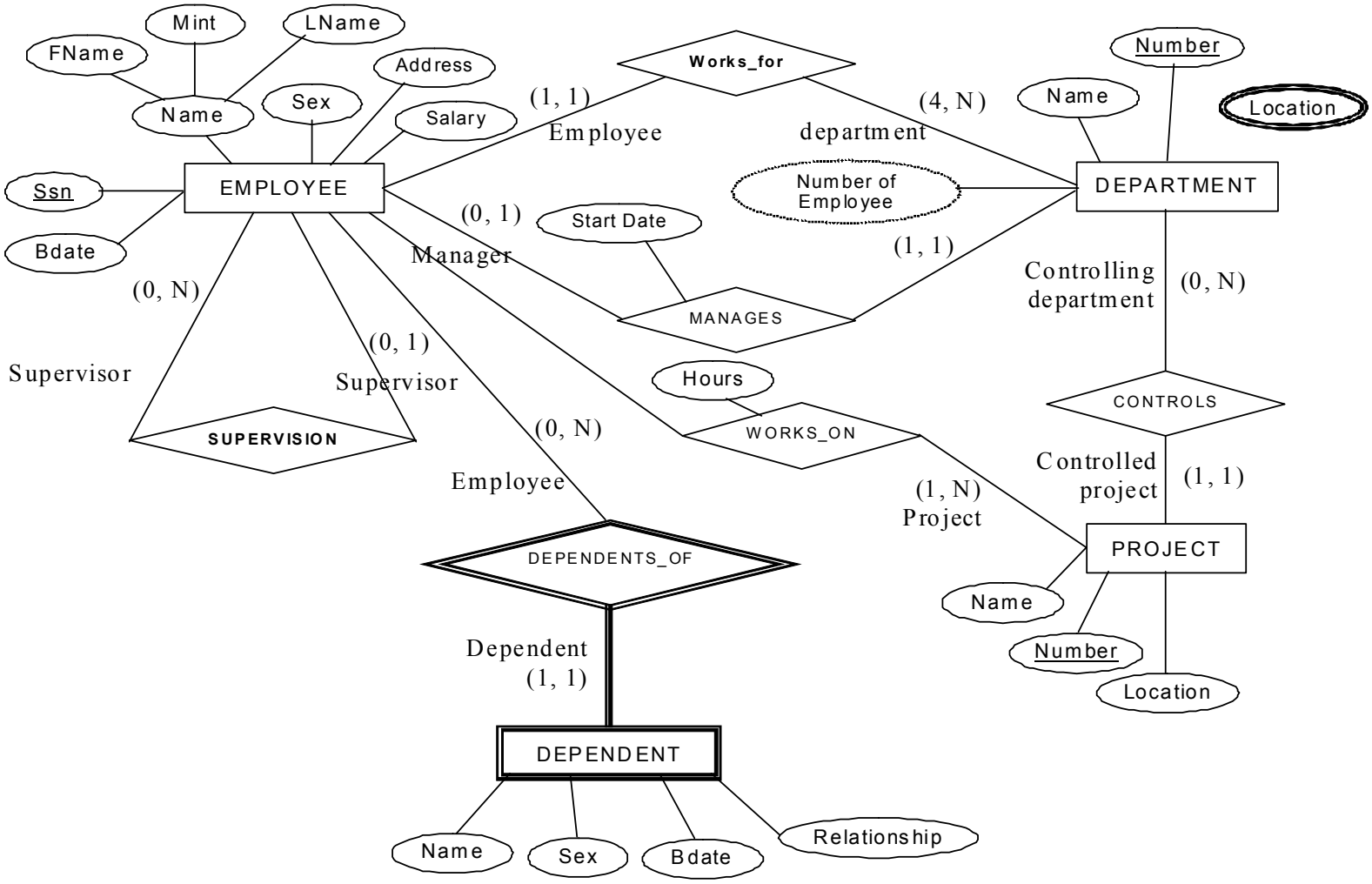
1. **Foreign Key approach:** Choose one of the relations-say S-and include a foreign key in S that refers to the primary key of T. It is better to choose an entity type with total participation in R in the role of S.
   - Example: 1:1 relation MANAGES is mapped by choosing the participating entity type DEPARTMENT to serve in the role of S, because its participation in the MANAGES relationship type is total.
2. **Merged relation option:** An alternate mapping of a 1:1 relationship type is possible by merging the two entity types and the relationship into a single relation. This may be appropriate when both participations are total.
3. **Cross-reference or relationship relation option:** The third alternative is to set up a third relation R for the purpose of cross-referencing the primary keys of the two relations S and T representing the entity types.

Step 4: Mapping of Binary 1:N Relationship Types.

- For each regular binary 1:N relationship type R, identify the relation S that represent the participating entity type at the N-side of the relationship type.
- Include as foreign key in S the primary key of the relation T that represents the other entity type participating in R.
- Include any simple attributes of the 1:N relation type as attributes of S.

Example: 1:N relationship types WORKS_FOR, CONTROLS, and SUPERVISION in the figure.

- For WORKS_FOR we include the primary key DNUMBER of the DEPARTMENT relation as foreign key in the EMPLOYEE relation and call it DNO.


Mint
LName
FName
Address
Name
Sex
Salary
Works_for
(1, 1)
Employee
(4, N)
department
Number
Name
Location
EMPLOYEE
Ssn
Bdate
Number of Employee
DEPARTMENT
(0, 1)
Start Date
Manager
(1, 1)
Controlling department
(0, N)
(0, N)
MANAGES
(0, 1)
Supervisor
Supervisor
Hours
CONTROLS
SUPERVISION
WORKS_ON
(0, N)
Controlled project
(1, 1)
Employee
(1, N)
Project
DEPENDENTS_OF
PROJECT
Name
Number
Location
Dependent
(1, 1)
DEPENDENT
Name
Sex
Bdate
Relationship

Step 5: Mapping of Binary M:N Relationship Types.

- For each regular 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 type (or simple components of composite attributes) as attributes of S.

Example: The M:N relationship type WORKS_ON from the ER diagram is mapped by creating a relation WORKS_ON in the relational database schema.

- The primary keys of the PROJECT and EMPLOYEE relations are included as foreign keys in WORKS_ON and renamed PNO and ESSN, respectively.
- Attribute HOURS in WORKS_ON represents the HOURS attribute of the relation type. The primary key of the WORKS_ON relation is the combination of the foreign key attributes {ESSN, PNO}.

Step 6: Mapping of Multivalued attributes.

- For each multivalued attribute A, create a new relation R.
- This relation R will 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 of relationship type that has A as an attribute.
- The primary key of R is the combination of A and K. If the multivalued attribute is composite, we include its simple components.

Example: The relation DEPT_LOCATIONS is created.

- The attribute DLOCATION represents the multivalued attribute LOCATIONS of DEPARTMENT, while DNUMBER-as foreign key-represents the primary key of the DEPARTMENT relation.
- The primary key of R is the combination of {DNUMBER, DLOCATION}.

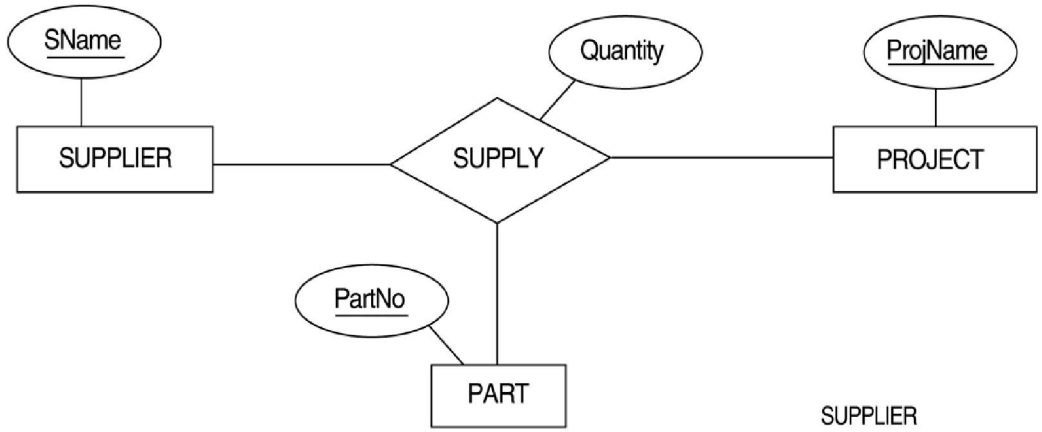
Step 7: Mapping of N-ary Relationship Types.

- For each n-ary relationship type R, where n>2, create a new relationship 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 (or simple components of composite attributes) as attributes of S.

Example: The relationship type SUPPY in the ER on the next slide.

- This can be mapped to the relation SUPPLY shown in the relational schema, whose primary key is the combination of the three foreign keys {SNAME, PARTNO, PROJNAME}

(a)



SUPPLIER

SNAME	. . .

PROJECT

PROJNAME	. . .

PART

PARTNO	. . .

SUPPLY

SNAME	PROJNAME	PARTNO	QUANTITY

EER-to- Relational Mapping

EER-to- Relational Mapping

Step8: Options for Mapping Specialization or Generalization.

Convert each specialization with m subclasses {S1, S2,….,Sm} and generalized superclass C, where the attributes of C are {k,a1,…an} and k is the (primary) key, into relational schemas using one of the four following options:

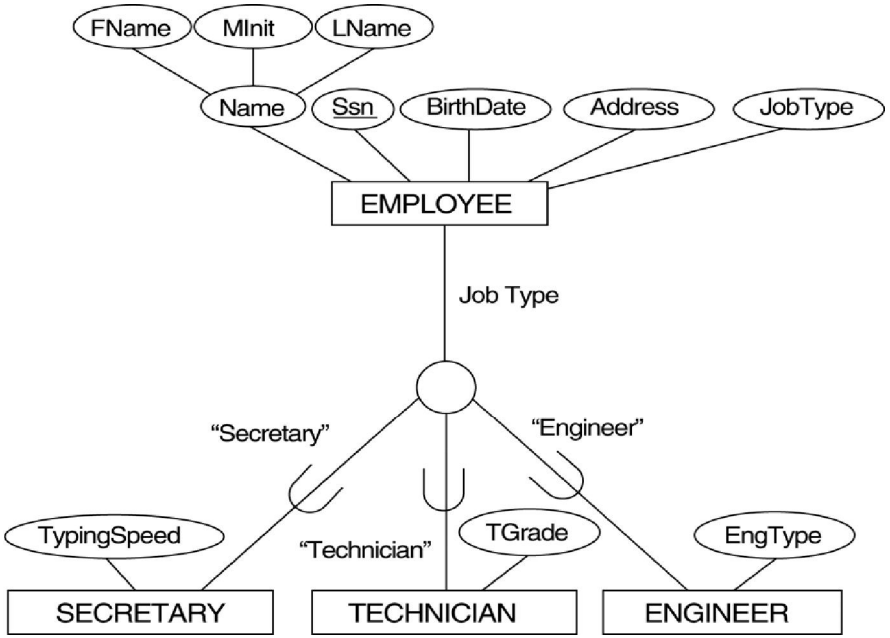
- Option 8A: Multiple relations-Superclass and subclasses
- Option 8B: Multiple relations-Subclass relations only
- Option 8C: Single relation with one type attribute
- Option 8D: Single relation with multiple type attributes

- **Option 8A: Multiple relations-Superclass and subclasses**

  Create a relation L for C with attributes Attrs(L) = {k,a1,…an} and PK(L) = k. Create a relation Li for each subclass Si, 1 < i < m, with the attributesAttrs(Li) = {k} U {attributes of Si} and PK(Li)=k. This option works for any specialization (total or partial, disjoint of over-lapping).

- **Option 8B: Multiple relations-Subclass relations only**

  Create a relation Li for each subclass Si, 1 < i < m, with the attributes Attr(Li) = {attributes of Si} U {k,a1…,an} and PK(Li) = k. This option only works for a specialization whose subclasses are total (every entity in the superclass must belong to (at least) one of the subclasses).



Option 8A: Multiple relations-Superclass and subclasses

(a)

EMPLOYEE

SSN	FName	MInit	LName	BirthDate	Address	JobType

SECRETARY

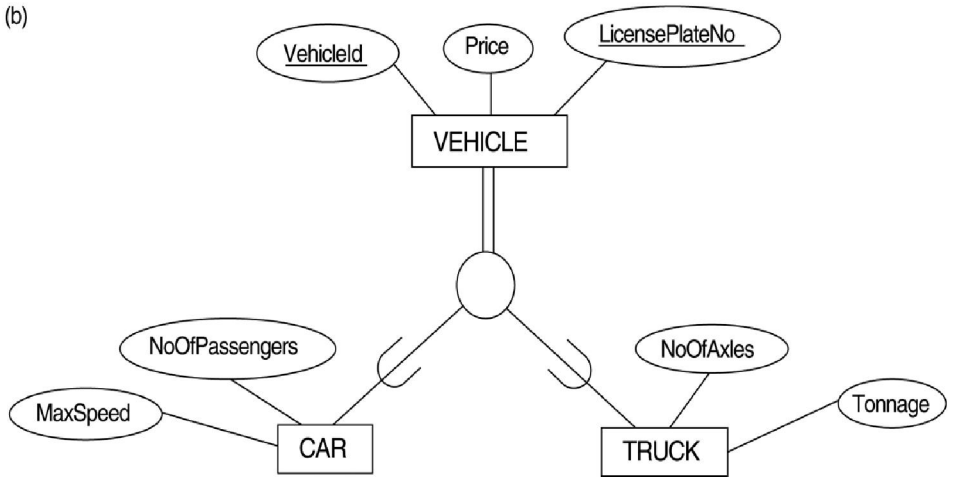
SSN	TypingSpeed

TECHNICIAN

SSN	TGrade

ENGINEER

SSN	EngType



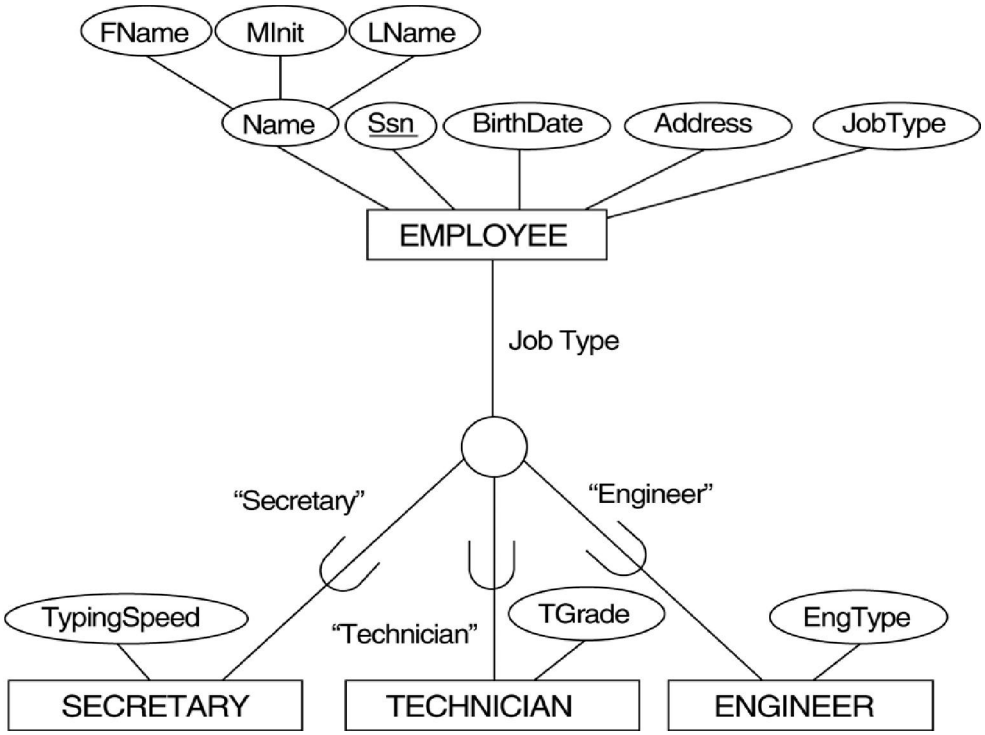
Option 8B: Multiple relations-Subclass relations only

(b) CAR

VehicleId	LicensePlateNo	Price	MaxSpeed	NoOfPassengers

TRUCK

VehicleId	LicensePlateNo	Price	NoOfAxles	



Option 8C: Single relation with one type attribute

EMPLOYEE

Ssn	Fname	Minit	Lname	Birthdatae	Address	jobtype	Typing speed	Tgrade	EngType

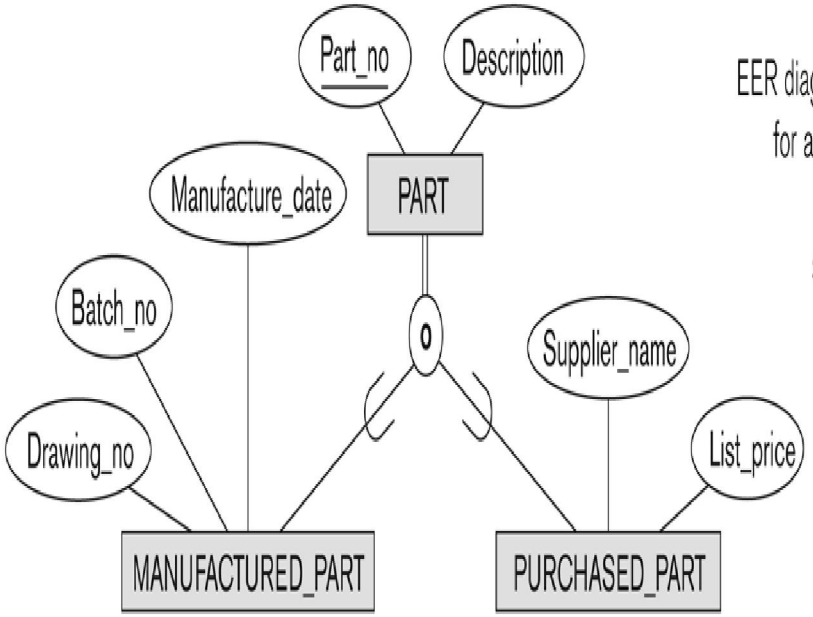


Figure 4.5
EER diagram notation for an overlapping (nondisjoint) specialization.

Option 8D: Single relation with multiple type attributes

EMPLOYEE

Part_no	Descr	Mflag	Drawing_no	Batch_no	Man_date	Pflag	supp_name	list_price

ER-Relational mapping for Company Database

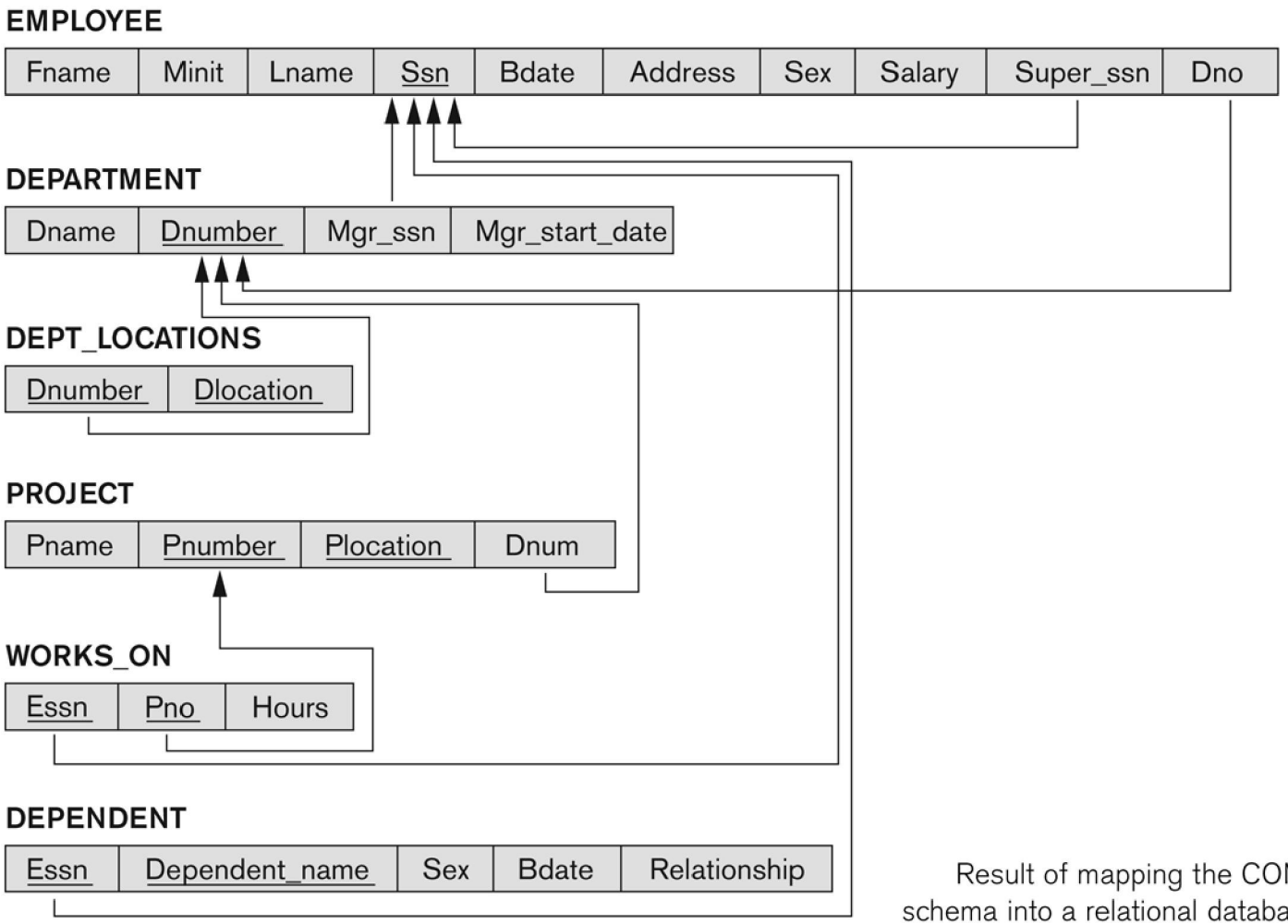
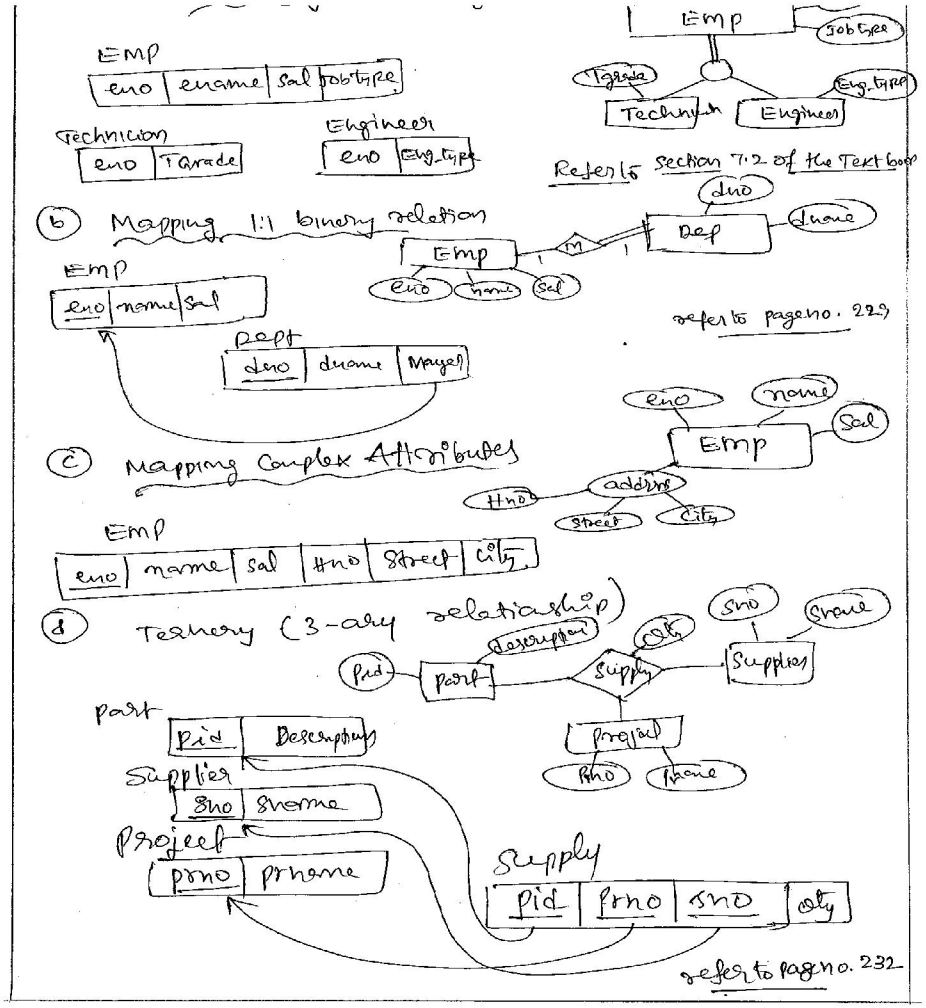


Figure 7.2
Result of mapping the COMPANY ER schema into a relational database schema.

Exercise 1

Explain how you would map the following EER/ER Constructs to Relational model. Give simple examples.

- Mapping specialization.
- Mapping 1:1 binary relationship, where one entity type has *total participation*, and the other entity type has *partial participation*.
- Mapping *Complex attribute* of an entity type.
- Ternary (3-ary) relationship.


Emp
eno ename sal jobtype
Technician
eno TGrade
Engineer
eno Eng_type
Emp
jobtype
TGrade
Eng_type
Technician
Engineer
Refer to Section 7.2 of the Text book
(b) Mapping 1:1 binary relation
Emp
eno name sal
Dept
dno dname Mgrsal
Emp
eno
name
sal
Dept
dno
dname
M
1
refer to page no. 229
(c) Mapping Complex Attributes
Emp
eno name sal Hno Street City
eno
name
sal
Emp
address
Hno
street
City
(d) Ternary (3-ary relationship)
part
Pid Descriptn
Supplier
Sno Sname
Project
Pno Pname
Supply
Pid Pno Sno qty
pid
Descriptn
part
qty
Supply
Sno
Sname
Supplier
Project
Pno
Pname
refer to page no. 232

Summary

We learnt ER/EER to relational mapping.