FACULTY OF COMPUTER SCIENCE AND ENGINEERING

RELATIONAL DATABASE MODEL

DATABASES - lectures

Outline

- Relational Model Concepts
- Relational Model Constraints and Relational Database Schemas
- (E)ER-to-Relational Mapping Algorithm

Relational Model Concepts

- The relational Model of Data is based on the concept of a **Relation**
- Informally, a **relation** looks like a **table** of values
- A relation typically contains a **set of rows**
- The data elements in each **row** represent certain facts that correspond to a real-world **entity** or **relationship**
- Each **column (attribute)** has a column header that gives an indication of the meaning of the data items in that column

Relational Model Concepts

- Formal schema (description) of a relation:
  - Denoted by $R(A_1, A_2, \ldots, A_n)$
    - R is the name of the relation
    - The attributes of the relation are A1, A2, ..., An
  - A_i has a domain (set of values) D, i.e. $D = dom(A_i)$
  - A tuple (row) is an ordered set of values (enclosed in angled brackets ‘< ... >’)
    - Each value is derived from an appropriate domain.
  - The number of attributes in the relation defines the degree of the relation

Example of a relation - STUDENT

Relation Name

Attributes

STUDENT

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

Tuples

Domain of the Ssn attribute

Figure 5.1
The attributes and tuples of a relation STUDENT.

Terms summary

Informal Terms		**Formal Terms**
Table		Relation
Column Header		Attribute
All possible Column Values		Domain
Row		Tuple
Table Definition		Schema of a Relation
Populated Table		State of the Relation

Characteristics of Relations

- Relations differ from data files with the following:
  - A relation is defined as a set of tuples, hence tuples are not considered to be ordered
  - In general we can consider the attributes (and their values) of a relation to be ordered.
    - The ordering of attributes in a tuple is not important if a tuple can be considered a set of (<attribute_name>,<value>) pairs
  - All values are considered atomic (indivisible).

Relational Integrity Constraints

- Constraints are **conditions** that must hold on **all** valid relation states
- There are three *main types* of constraints in the relational model:
  - **Key** constraints
  - **Entity integrity** constraints
  - **Referential integrity** constraints
- Another implicit constraint is the **domain** constraint
  - Every value in a tuple must be from the *domain of its attribute* (or it could be **null**, if allowed for that attribute)

Key Constraints (1)

- **Superkey** of R:
  - Is a set of attributes SK of R with the following condition:
    - No two tuples in any valid relation state r(R) will have the same value for SK
    - That is, for any distinct tuples t1 and t2 in r(R), t1[SK] $\neq$ t2[SK]
    - This condition must hold in *any valid state* r(R)
- **Key** of R:
  - A "minimal" superkey
  - That is, a key is a superkey K such that removal of any attribute from K results in a set of attributes that is not a superkey (does not possess the superkey uniqueness property)

Key Constraints (2)

- If a relation has several **candidate keys**, one is chosen arbitrarily to be the **primary key**.
  - The primary key attributes are underlined.

A rule of thumb is to choose the shortest possible candidate key as a primary key

Example

The CAR relation with two candidate keys

LicenseNumber is chosen as a primary key

CAR

License_number	Engine_serial_number	Make	Model	Year
Texas ABC-739	A69352	Ford	Mustang	02
Florida TVP-347	B43696	Oldsmobile	Cutlass	05
New York MPO-22	X83554	Oldsmobile	Delta	01
California 432-TFY	C43742	Mercedes	190-D	99
California RSK-629	Y82935	Toyota	Camry	04
Texas RSK-629	U028365	Jaguar	XJS	04

Relational Database Schema

- **Relational Database Schema:**
  - A set S of relation schemas that belong to the same database + a set of relational integrity constraints
  - S is the name of the whole **database schema**
  - S = {R1, R2, ..., Rn}
    - R1, R2, …, Rn are the names of the individual **relation schemas** within the database S
  - The set of relational integrity constraints includes:
    - Entity integrity
    - Referential integrity

Example - COMPANY Relational Database Schema

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

DEPT_LOCATIONS

Dnumber	Dlocation

PROJECT

Pname	Pnumber	Plocation	Dnum

WORKS_ON

Essn	Pno	Hours

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship

Figure 5.5
Schema diagram for the COMPANY relational database schema.

Entity Integrity

- **Entity Integrity:**
  - The *primary key attributes* PK of each relation schema R in S cannot have null values in any tuple of r(R).
    - This is because primary key values are used to *identify* the individual tuples.
    - t[PK] $\neq$ null for any tuple t in r(R)
    - **If PK has several attributes, null is not allowed in any of these attributes**
  - Note: Other attributes of R may be constrained to disallow null values, even though they are not members of the primary key.

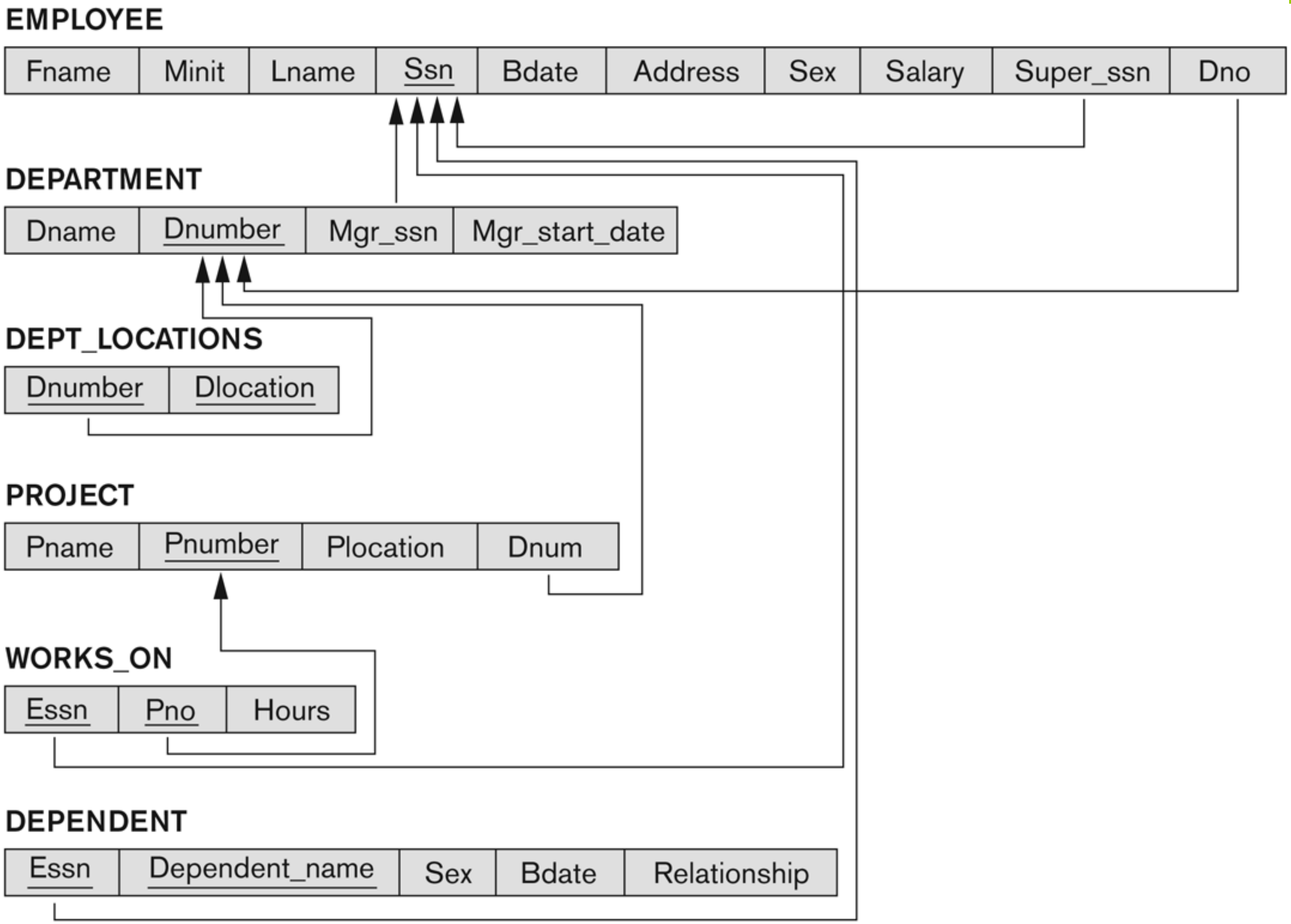
Referential Integrity

- A constraint involving **two** relations
  - The previous constraints involve a single relation.
- Used to specify a **relationship** among tuples in two relations:
  - The **referencing relation** and the **referenced relation**.

Referential Integrity (or foreign key) Constraint

- A set of attributes FK in relation schema *R_1 is a* ***foreign key*** *of R_1 that* ***references*** *relation R_2 if it satisfies the* following rules:
  - 1. The attributes in FK have the same domain(s) as the primary key attributes PK of R_2; the attributes FK are said to **reference** or **refer to** the relation R_2.
  - 2. A value of FK in a tuple t_1 of the current state $r_1(R_1)$ either occurs as a value of PK for some tuple t_2 in the current state $r_2(R_2)$ *or is NULL*. In the former case, we have $t_1[FK]= t_2[PK]$, and we say that the tuple t_1 **references** or **refers to** the tuple t_2.
- In this definition, *R_1 is called the* ***referencing relation*** *and R_2 is the* ***referenced relation.***
- If these two conditions hold, a **referential integrity constraint** from *R1 to R2 is* said to hold.

Example - Referential Integrity Constraints for COMPANY database



Exercise

- Consider the following relations for a database that keeps track of student enrollment in courses and the books adopted for each course :
  - STUDENT(SSN, Name, Major, Bdate)
  - COURSE(Course#, Cname, Dept)
  - ENROLL(SSN, Course#, Quarter, Grade)
  - BOOK_ADOPTION(Course#, Quarter, Book_ISBN)
  - TEXT(Book_ISBN, Book_Title, Publisher, Author)
- **Specify the foreign keys for this schema, stating any assumptions you make.**

Other Types of Constraints

- Semantic Integrity Constraints:
  - based on application semantics and cannot be expressed by the model per se
  - **Examples:**
    - “the salary of an employee cannot be greater than the salary of his manager”
    - “the max. no. of hours per employee for all projects he or she works on is 56 hrs per week”

Populated database state

- Each *relation* will have many tuples in its current relation state
- The *relational database state* is a union of all the individual relation states
- Whenever the database is changed, a new state arises
- Basic operations for changing the database:
  - INSERT a new tuple in a relation
  - DELETE an existing tuple from a relation
  - MODIFY an attribute of an existing tuple

Populated database state for COMPANY

EMPLOYEE

Fname	Minit	Lname	Ssn	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	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

WORKS_ON

Essn	Pno	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

DEPENDENT

Essn	Dependent_name	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

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

PROJECT

Pname	Pnumber	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

ER-to-Relational Mapping Algorithm

- Relational database schema design
  - Based on the conceptual schema
- Algorithm composed of seven steps for transforming the basic ER constructs to constructs of the relational model
- Additional steps for the EER model

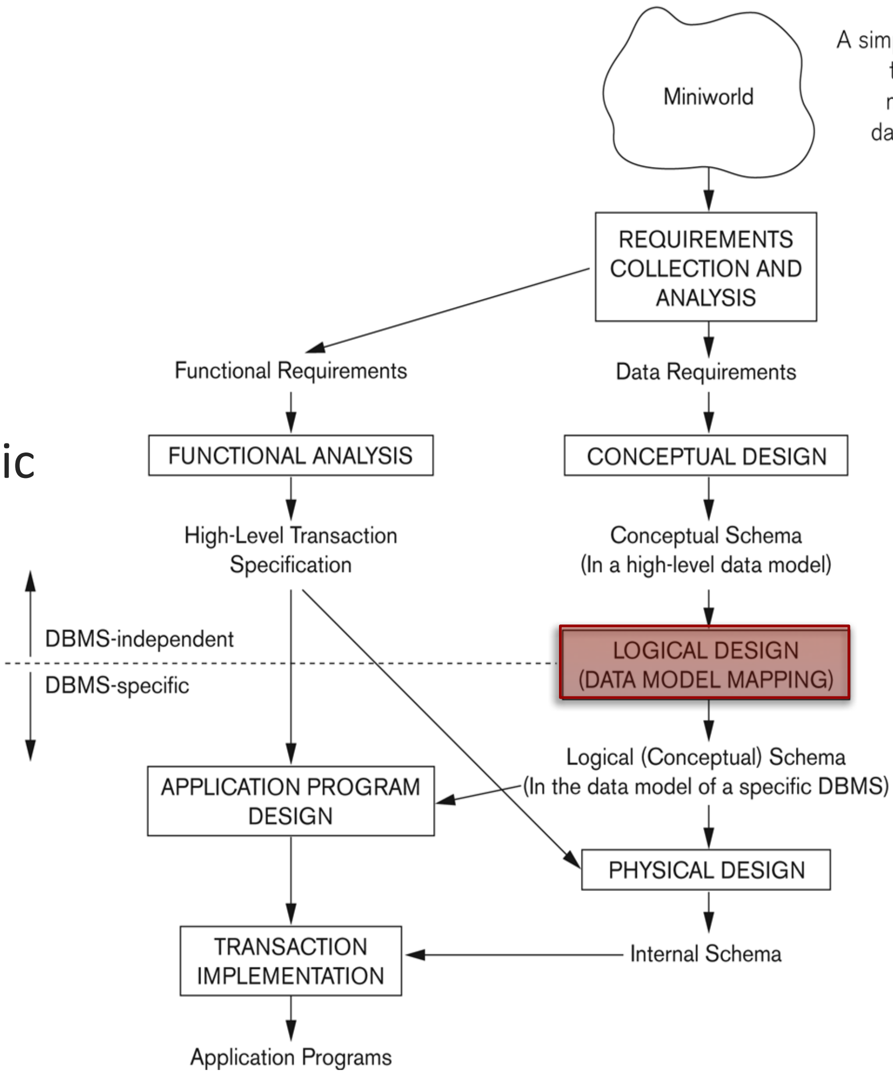


Figure 3.1
A simplified diagram to illustrate the main phases of database design.

ER-to-Relational Mapping Algorithm

Example

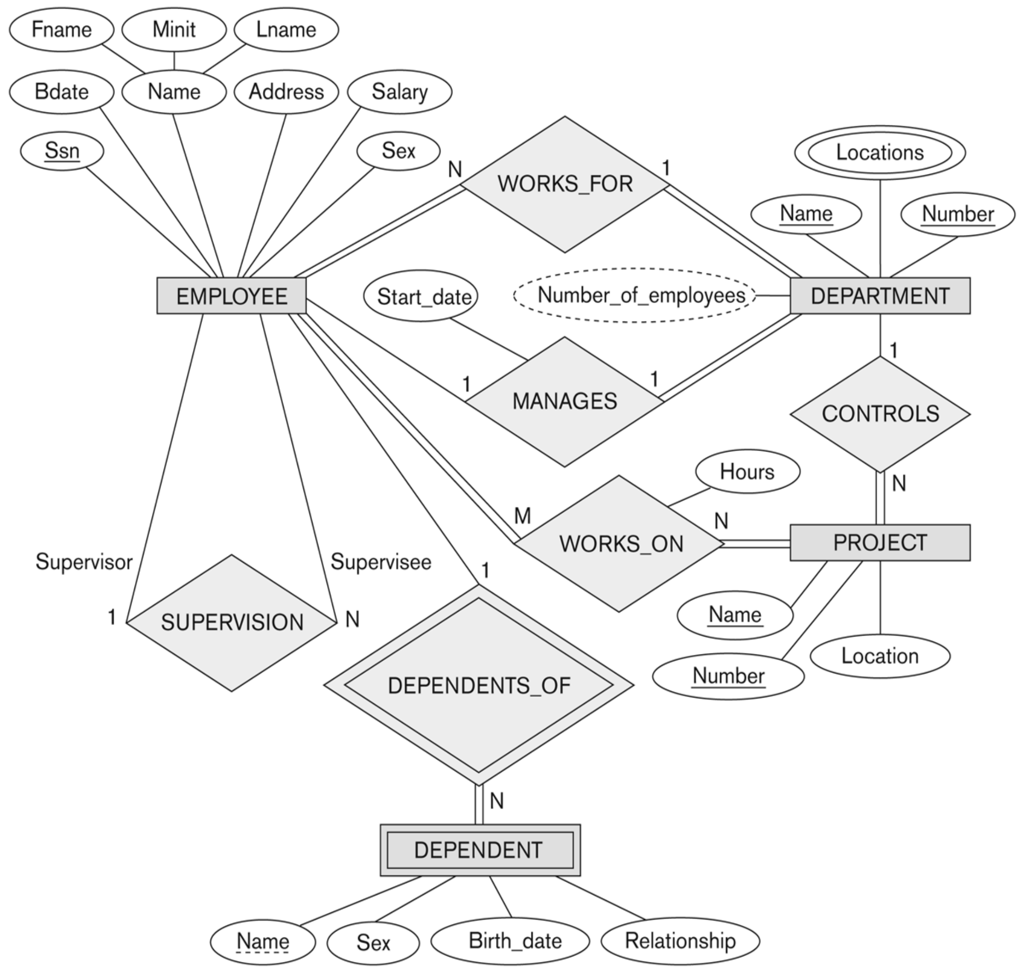


Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

ER-to-Relational Mapping Algorithm

- **ER-to-Relational Mapping Algorithm**
  - Step 1: Mapping of Regular Entity Types
  - Step 2: Mapping of Weak Entity Types
  - Step 3: Mapping of Binary 1:1 Relation Types
  - Step 4: Mapping of Binary 1:N Relationship Types.
  - Step 5: Mapping of Binary M:N Relationship Types.
  - Step 6: Mapping of Multivalued attributes.
  - Step 7: Mapping of N-ary Relationship Types.

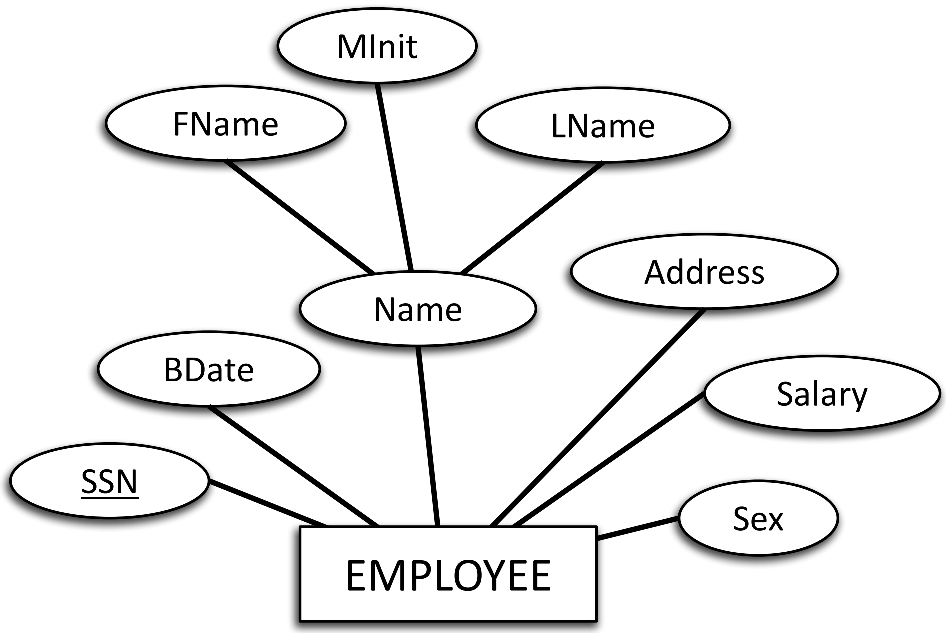
- **Mapping EER Model Constructs to Relations**
  - Step 8: Options for Mapping Specialization or Generalization.

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.
- *Composite* attributes are transformed into *a list of simple* attributes
- 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.

Mapping of Regular Entity Types

Example



EMPLOYEE

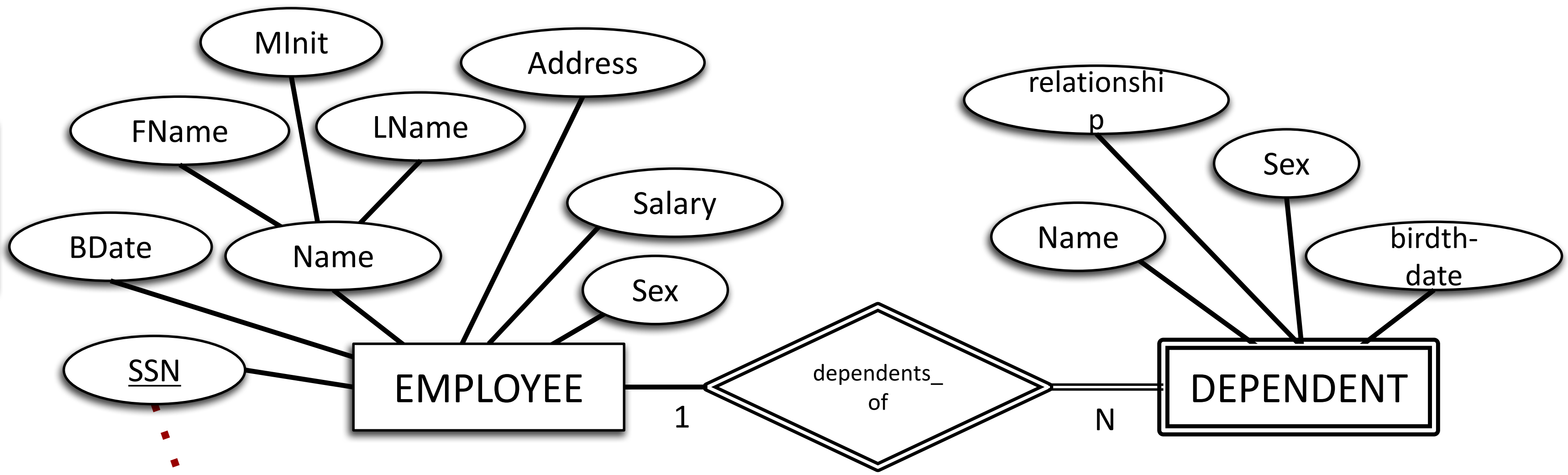
<u>SSN</u>	Fname	MInit	LName	BDate	Address	Salary	Sex

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.

Mapping of Weak Entity Types

Example



DEPENDENT

<u>Essn</u>	<u>DependentName</u>	BDate	Sex	relationship

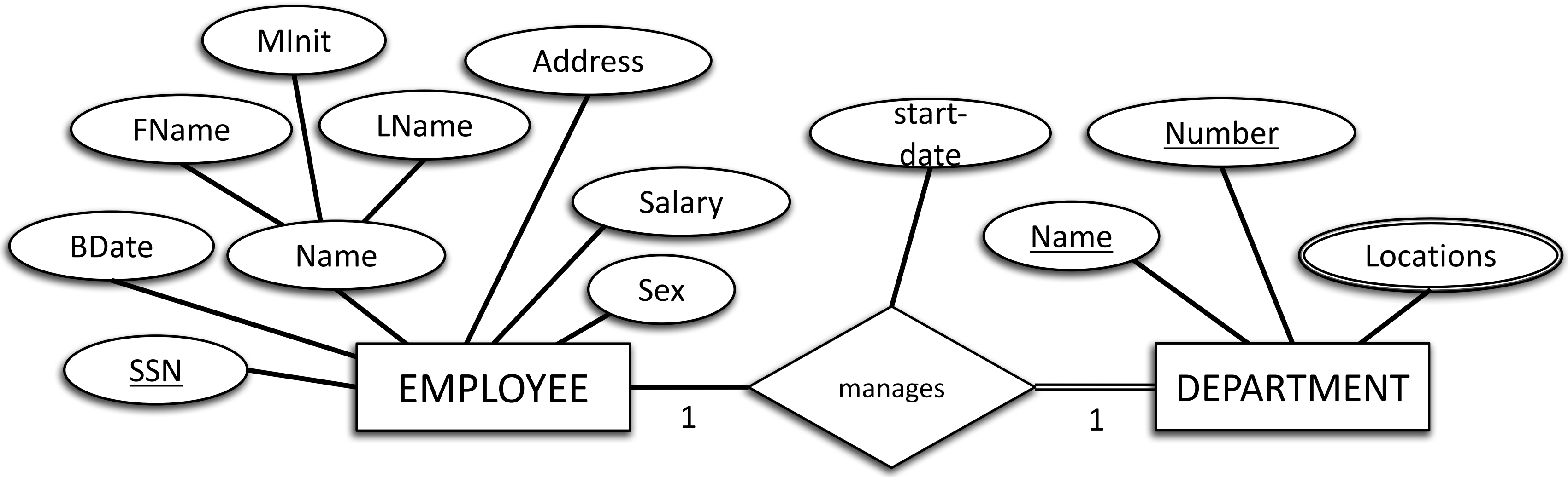
Mapping of Binary 1:1 Relation Types

- 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 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.

Mapping of Binary 1:1 Relation Types

Example



DEPARTMENT

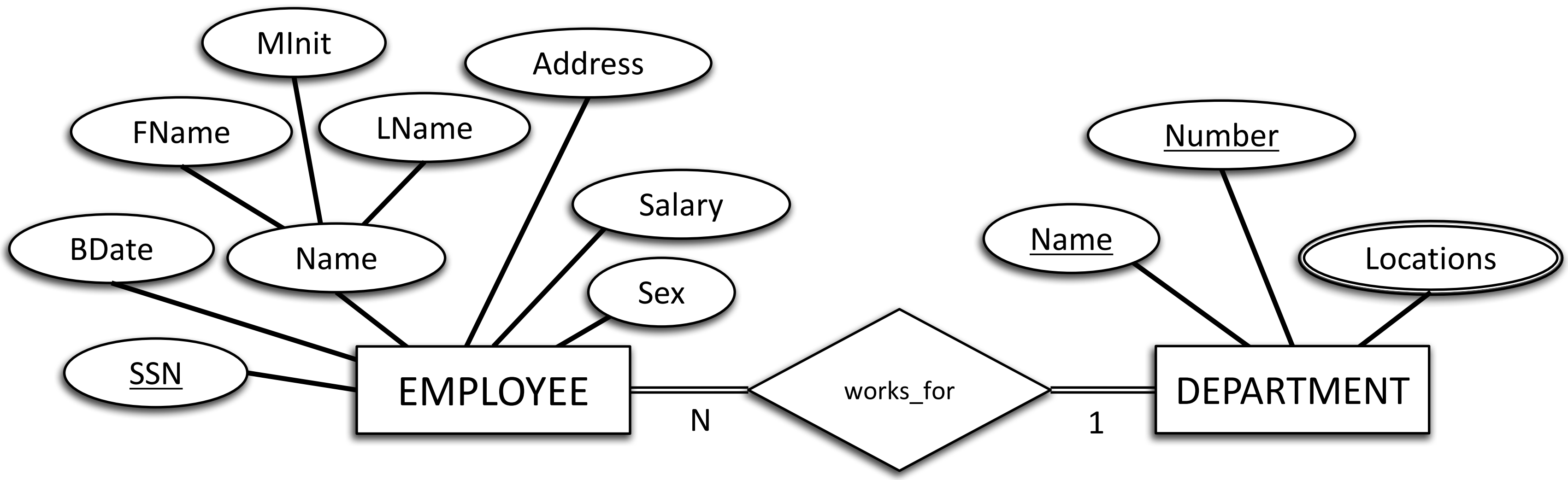
DNumber	DName	Mgr_SSN	Mgr_start_date

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.
  - This will never occur since we use the rule that no attributes should be assigned to a 1:N relationship types

Mapping of Binary 1:N Relationship Types

Example



EMPLOYEE

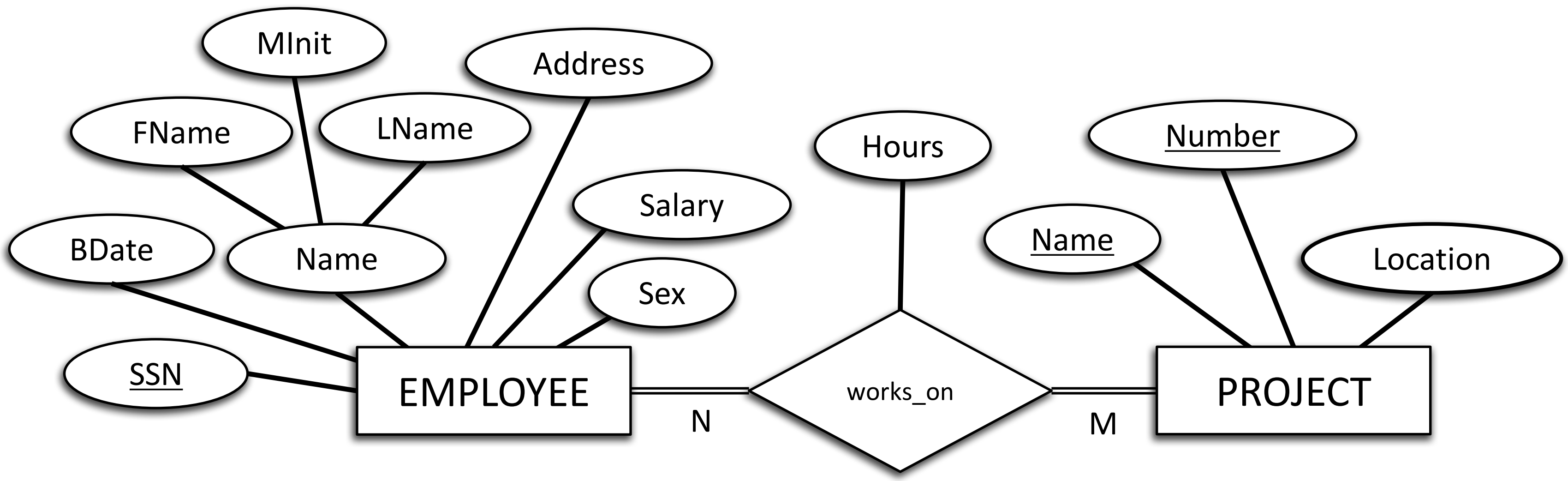
<u>SSN</u>	Fname	MInit	LName	BDate	Address	Salary	Sex	DNo

Mapping of Binary M:N Relationship Types

- For each regular binary M:N relationship type R, create a new relation Q to represent R.
- Include as foreign key attributes in Q the primary keys of the relations that represent the participating entity types; their combination will form the primary key of Q.
- Also include any simple attributes of the M:N relationship type (or simple components of composite attributes) as attributes of Q.

Mapping of Binary M:N Relationship Types

Example



EMPLOYEE

SSN	Fname	MInit	LName	BDate	Address	Salary	Sex	DNo

PROJECT

PNumber	PName	Location

WORKS_FOR

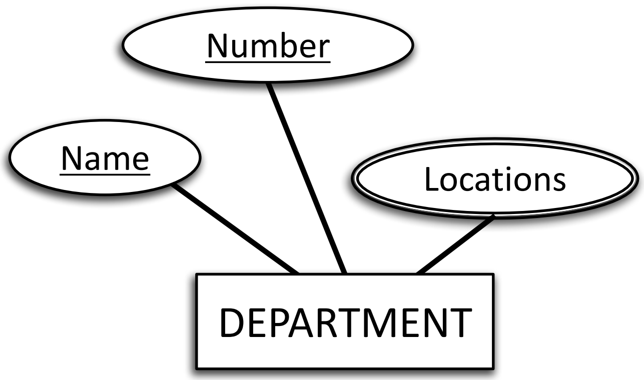
SSN	PNumber	Hours

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.

Mapping of Multivalued attributes

Example



DEPARTMENT

DNumber	DName	Mgr_SSN	Mgr_start_date

DEPT_LOCATIONS

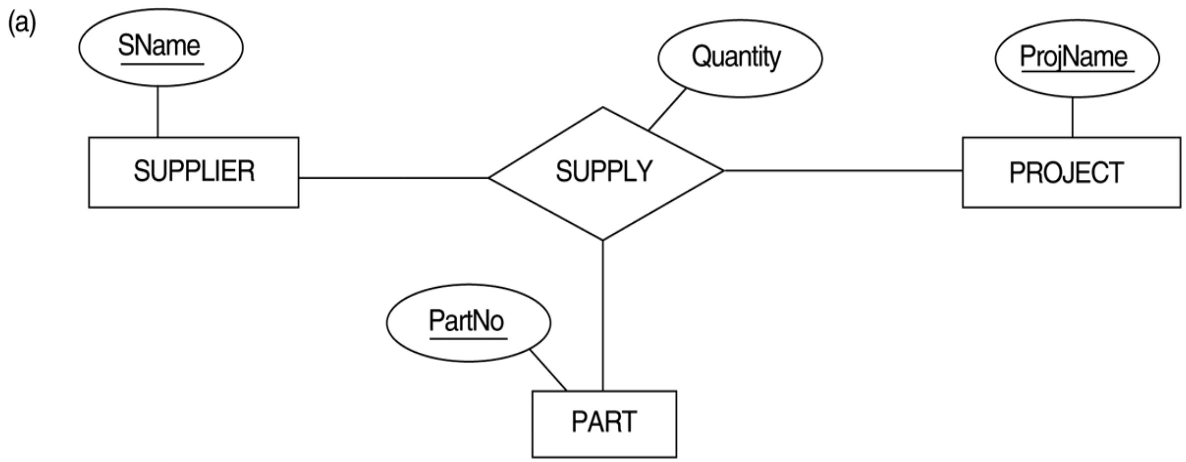
DNumber	Location

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 (or simple components of composite attributes) as attributes of S.

Mapping of N-ary Relationship Types

Example



SUPPLIER

<u>SName</u>	…

PROJECT

<u>ProjName</u>	…

PART

<u>PartNo</u>	…

SUPPLY

<u>SName</u>	<u>ProjName</u>	<u>PartNo</u>	Quantity

Summary of Mapping constructs and constraints

Table 7.1 Correspondence between ER and Relational Models

ER Model	**Relational Model**
Entity type	"Entity" relation
1:1 or 1:N relationship type	Foreign key (or "relationship" relation)
M:N relationship type	"Relationship" relation and two foreign keys
n-ary relationship type	"Relationship" relation and n foreign keys
Simple attribute	Attribute
Composite attribute	Set of simple component attributes
Multivalued attribute	Relation and foreign key
Value set	Domain
Key attribute	Primary (or secondary) key

Mapping EER Model Constructs to Relations

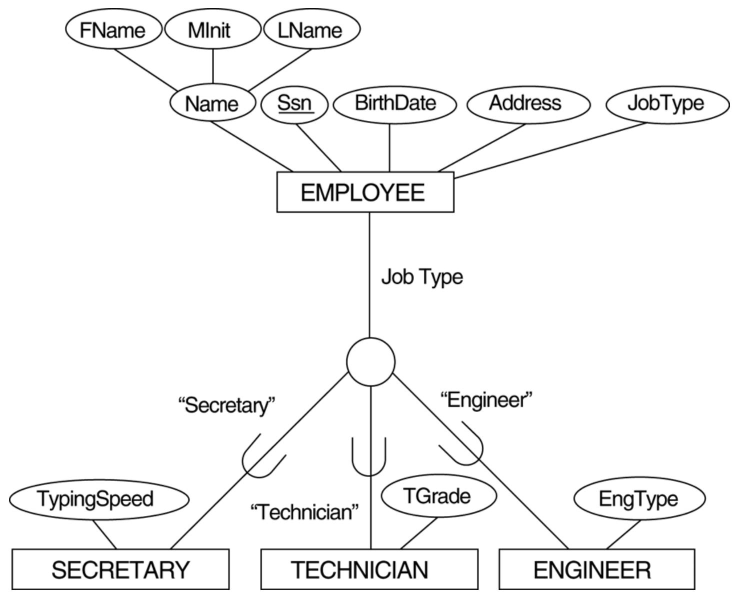
- **Step8: Options for Mapping Specialization or Generalization.**
  - Convert each specialization with m subclasses {S1, S2,….,Sm} and generalized superclass C, where the attributes of C are {k,a1,…an} and k is the (primary) key, into relational schemas using one of the four following options:
    - 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

Mapping EER Model Constructs to Relations

- **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 attributes Attrs(Li) = {k} U {attributes of Si} and PK(Li)=k. This option works for any specialization (total or partial, disjoint or 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).

Mapping EER Model Constructs to Relations

Example



(a) EMPLOYEE

<u>SSN</u>	FName	MInit	LName	BirthDate	Address	JobType

SECRETARY

<u>SSN</u>	TypingSpeed

TECHNICIAN

<u>SSN</u>	TGrade

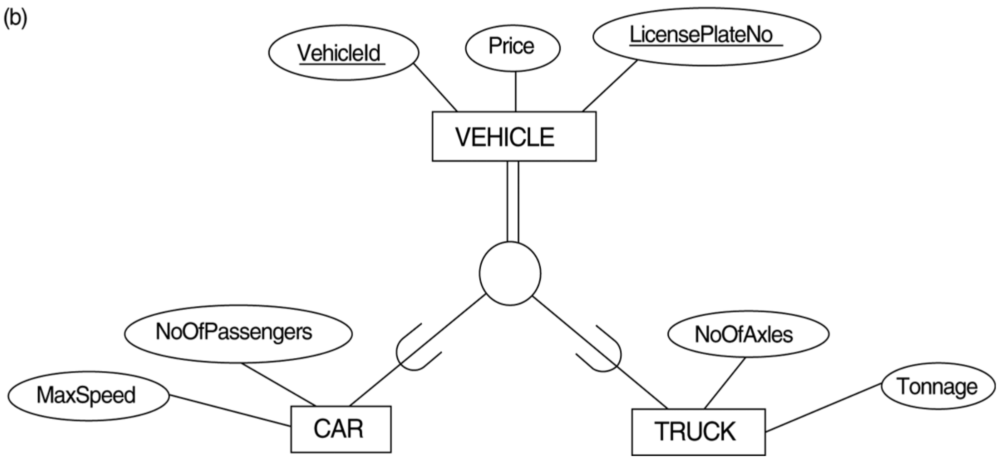
ENGINEER

<u>SSN</u>	EngType

Mapping EER Model Constructs to Relations

Example

(b)



(b) CAR

<u>VehicleId</u>	LicensePlateNo	Price	MaxSpeed	NoOfPassengers

TRUCK

<u>VehicleId</u>	LicensePlateNo	Price	NoOfAxles	Tonnage

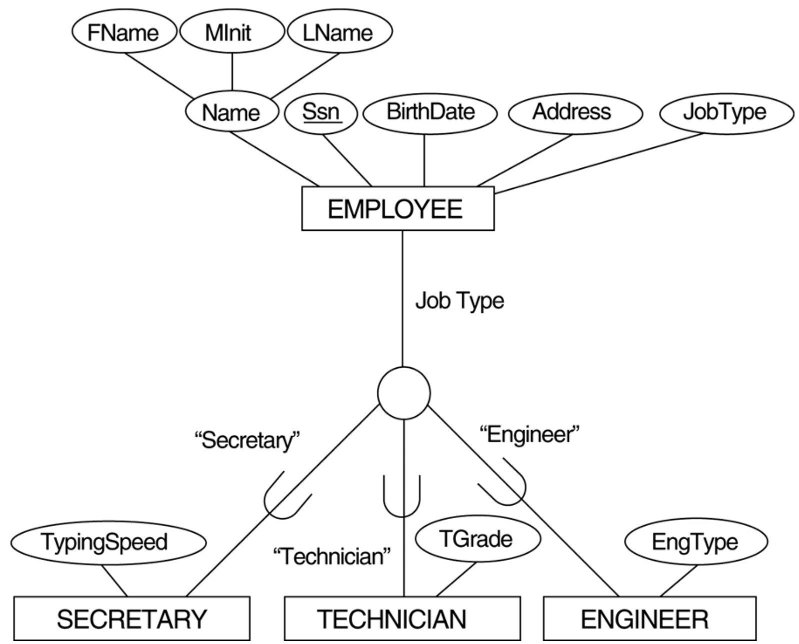
Mapping EER Model Constructs to Relations

- **Option 8C: Single relation with one type attribute**
  - Create a single relation L with attributes Attrs(L) = $\{k,a_1,\ldots a_n\}$ U {attributes of S_1} U…U {attributes of S_m} U {t} and PK(L) = k. The attribute t is called a type (or **discriminating**) attribute that indicates the subclass to which each tuple belongs

- **Option 8D: Single relation with multiple type attributes**
  - Create a single relation schema L with attributes Attrs(L) = $\{k,a_1,\ldots a_n\}$ U {attributes of S_1} U…U {attributes of S_m} U $\{t_1, t_2,\ldots,t_m\}$ and PK(L) = k. Each t_i, 1 < I < m, is a Boolean type attribute indicating whether a tuple belongs to the subclass S_i.

Mapping EER Model Constructs to Relations

Example

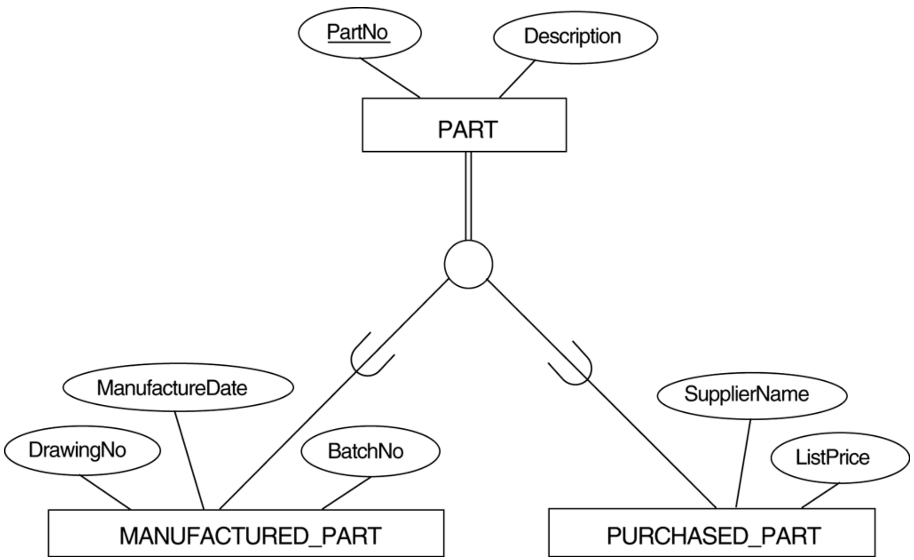


(c) EMPLOYEE

SSN	FName	MInit	LName	BirthDate	Address	JobType	TypingSpeed	TGrade	

Mapping EER Model Constructs to Relations

Example



(d) PART

PartNo	Description	MFlag	DrawingNo	ManufactureDate	BatchNo	PFlag	SupplierName	ListPrice

Mapping EER Model Constructs to Relations

- Mapping of Shared Subclasses (Multiple Inheritance)
  - A shared subclass, such as STUDENT_ASSISTANT, is a subclass of several classes, indicating multiple inheritance. These classes must all have the same key attribute; otherwise, the shared subclass would be modeled as a category.
  - We can apply any of the options discussed in Step 8 to a shared subclass, subject to the restriction discussed in Step 8 of the mapping algorithm. Below both 8C and 8D are used for the shared class STUDENT_ASSISTANT.

Mapping EER Model Constructs to Relations

Example

PERSON

SSN	Name	BirthDate	Sex	Address

EMPLOYEE

SSN	Salary	EmployeeType	Position	Rank	PercentTime	RAFlag	TAFlag	Project	

ALUMNUS

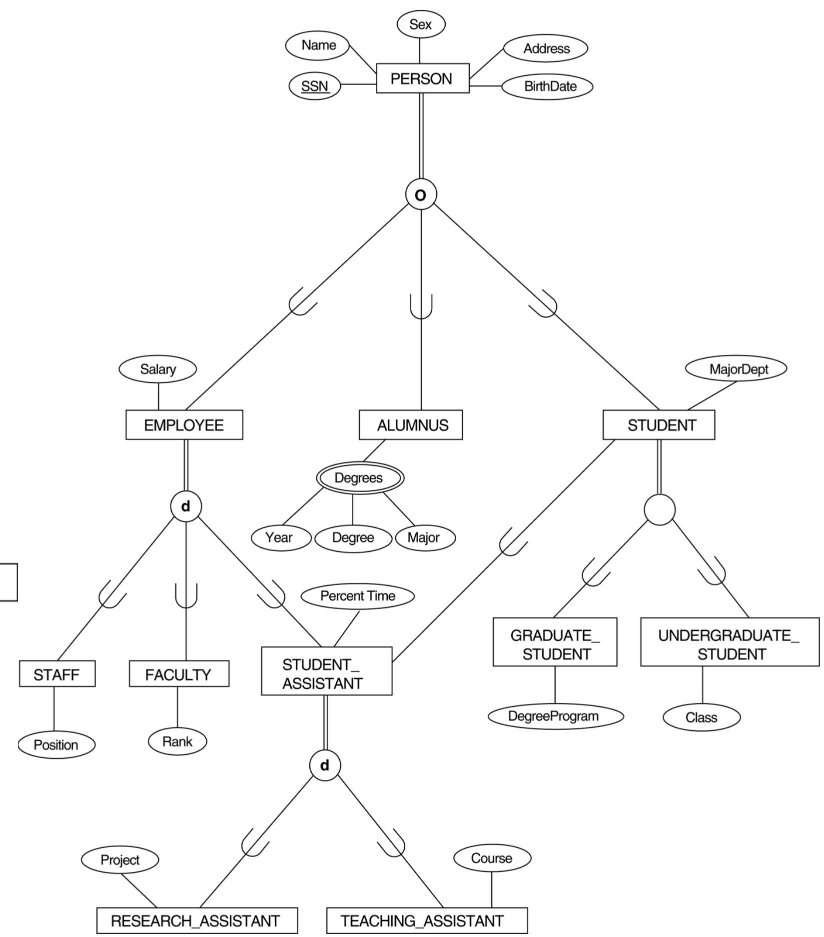
SSN

ALUMNUS_DEGREES

SSN	Year	Degree	

STUDENT

SSN	MajorDept	GradFlag	UndergradFlag	DegreeProgram	Class	StudAssistFlag



Mapping EER Model Constructs to Relations

Example

8A

PERSON

SSN	Name	BirthDate	Sex	Address

8C

EMPLOYEE

SSN	Salary	EmployeeType	Position	Rank	PercentTime	RAFlag	TAFlag	Project	

ALUMNUS

SSN

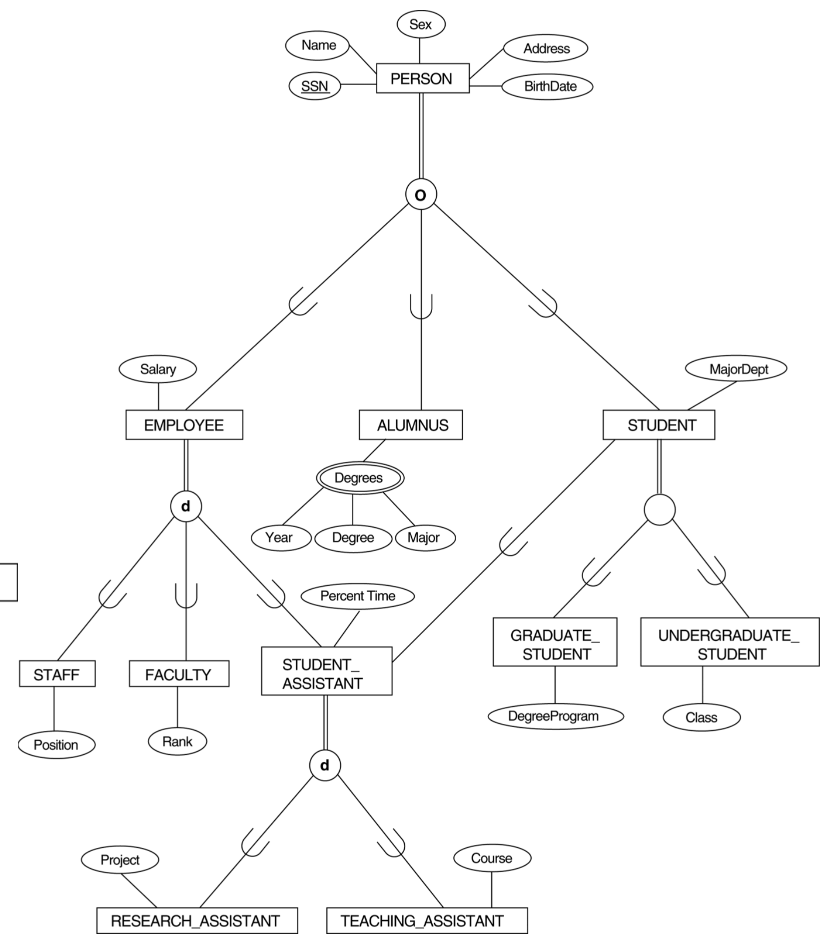
ALUMNUS_DEGREES

SSN	Year	Degree	

8D

STUDENT

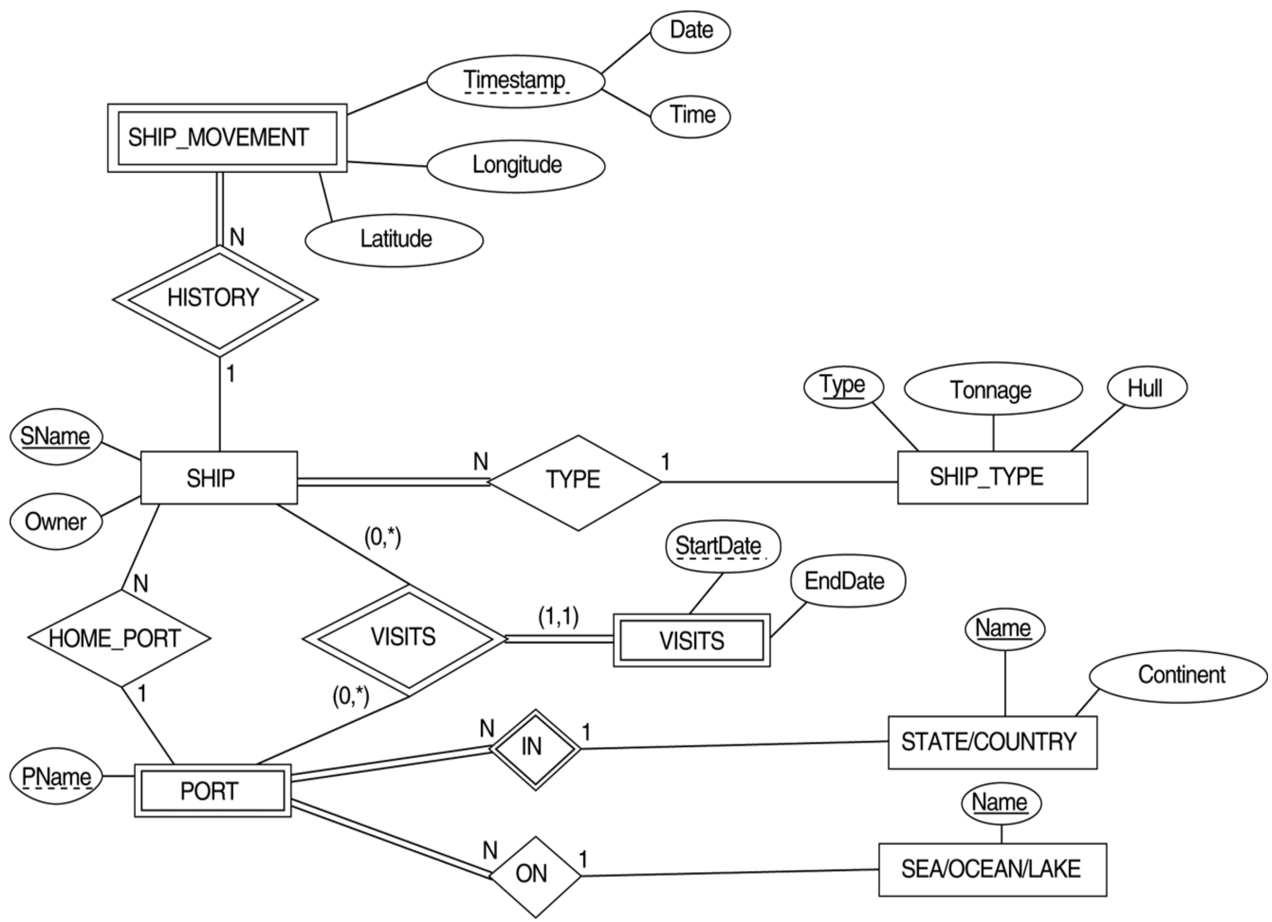
SSN	MajorDept	GradFlag	UndergradFlag	DegreeProgram	Class	StudAssistFlag



Mapping Exercise

Create the relational database schema based on the ER diagram!

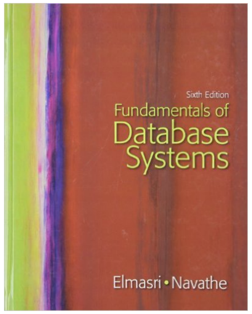
Be careful of the (min, max) notation used on some relationships in the ER!



Summary

- Relational Model Concepts
- Relational Model Constraints and Relational Database Schemas
- **ER-to-Relational Mapping Algorithm**
  - Step 1: Mapping of Regular Entity Types
  - Step 2: Mapping of Weak Entity Types
  - Step 3: Mapping of Binary 1:1 Relation Types
  - Step 4: Mapping of Binary 1:N Relationship Types.
  - Step 5: Mapping of Binary M:N Relationship Types.
  - Step 6: Mapping of Multivalued attributes.
  - Step 7: Mapping of N-ary Relationship Types.
- **Mapping EER Model Constructs to Relations**
  - Step 8: Options for Mapping Specialization or Generalization.

Bibliography



- **Chapter 3**
- **Chapter 9**

- **Chapter 2**