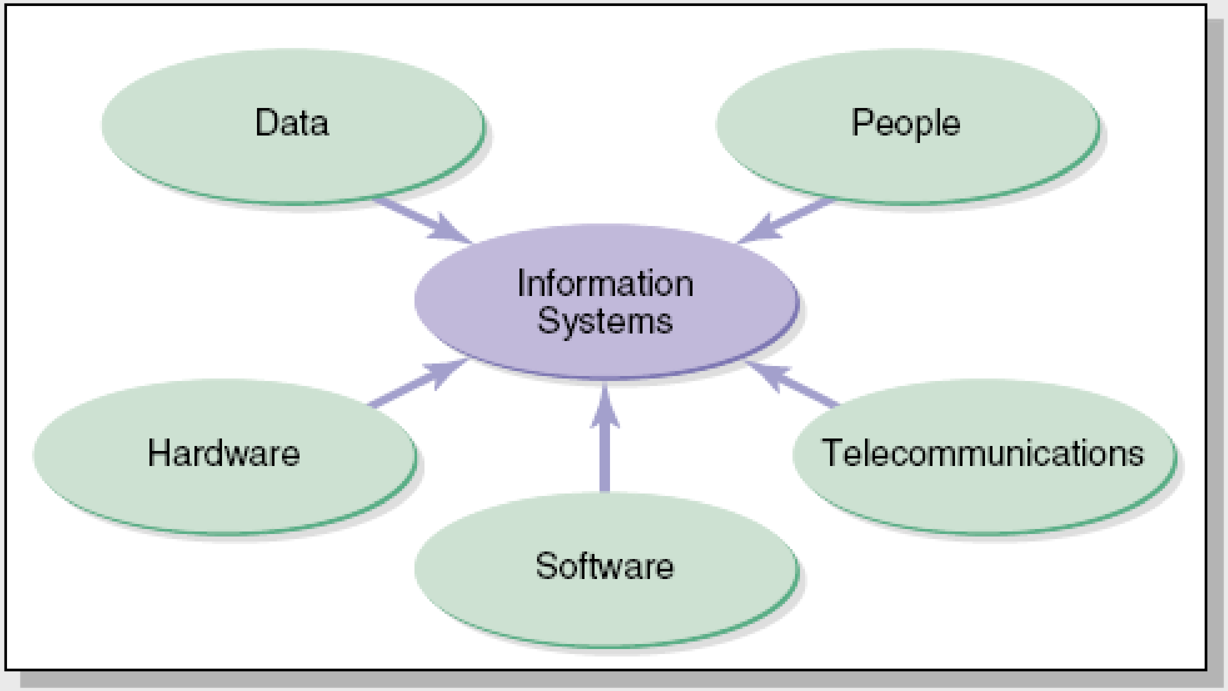
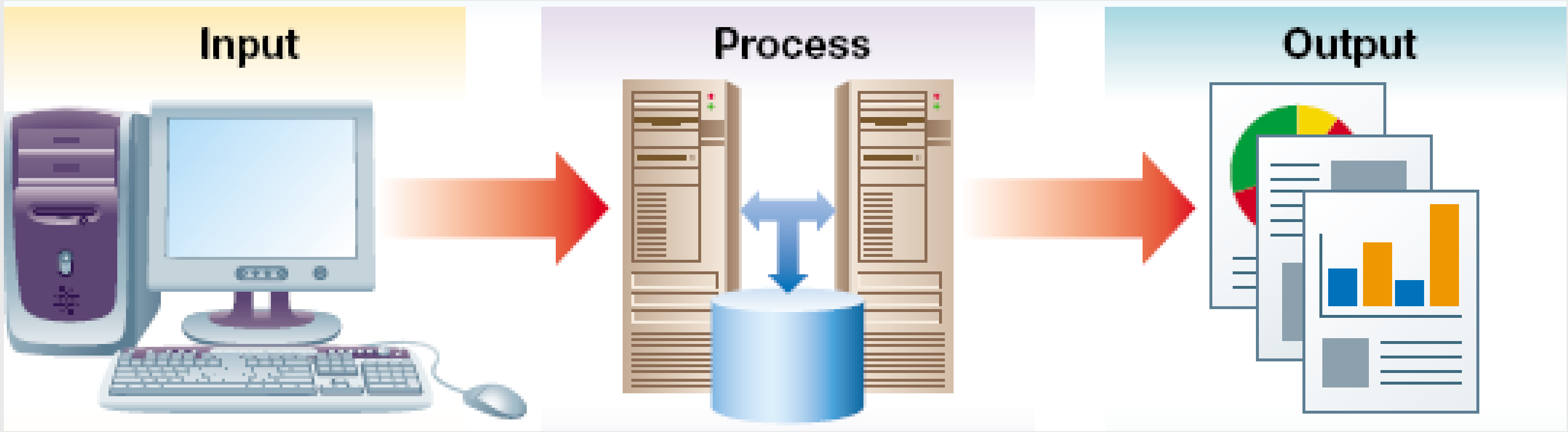
What are Information Systems?



- A combination of technical components
- Built and used by people to collect, create, and distribute useful data
- Used typically in organizational settings but are evolving for personal use

Information Systems: Turn Data into Information



Data

- Raw material
- Unformatted information
- Generally has no context

Information

- Processed material
- Formatted information
- Data given context

Individual time cards for factory workers entered into the payroll system	**Examples**	Department Labor Report, Project Status Report, Employee Payroll Checks

Technical
- Knowledge of hardware, software, networking, and security
- Most IS professionals are not deep technical experts but can direct/manage others with the required technical skills

Business
- Understand the nature of business including process, management, social, and communication domains
- Unique skills over those with only technical skills

Systems
- Knowledge of approaches and methods, also possess critical thinking and problem solving skills necessary to build and integrate large information systems
- Unique skills over those with only technical skills

- Executive Information Systems
- Decision Support Systems (both levels)
- Management Information Systems
- Transaction Processing Systems
- Expert Systems

- Functional Area Information Systems (Across all levels within a function)

Database Technology

- A collection of related data organized in a way that makes it valuable and useful
- Allows organizations to retrieve, store, and analyze information easily
- Is vital to an organization’s success in running operations and making decisions

Types of Databases and Database Applications

- Traditional Applications:
  - Numeric and Textual Databases
- More Recent Applications:
  - Multimedia Databases
  - Geographic Information Systems (GIS)
  - Data Warehouses
  - Real-time and Active Databases
  - Many other applications

Basic Definitions

- **Database:**
  - A collection of related data.
- **Data:**
  - Known facts that can be recorded and have an implicit meaning.
- **Mini-world:**
  - Some part of the real world about which data is stored in a database. For example, student grades and transcripts at a university.
- **Database Management System (DBMS):**
  - A software package/ system to facilitate the creation and maintenance of a computerized database.
- **Database System:**
  - The DBMS software together with the data itself. Sometimes, the applications are also included.

Simplified database system environment

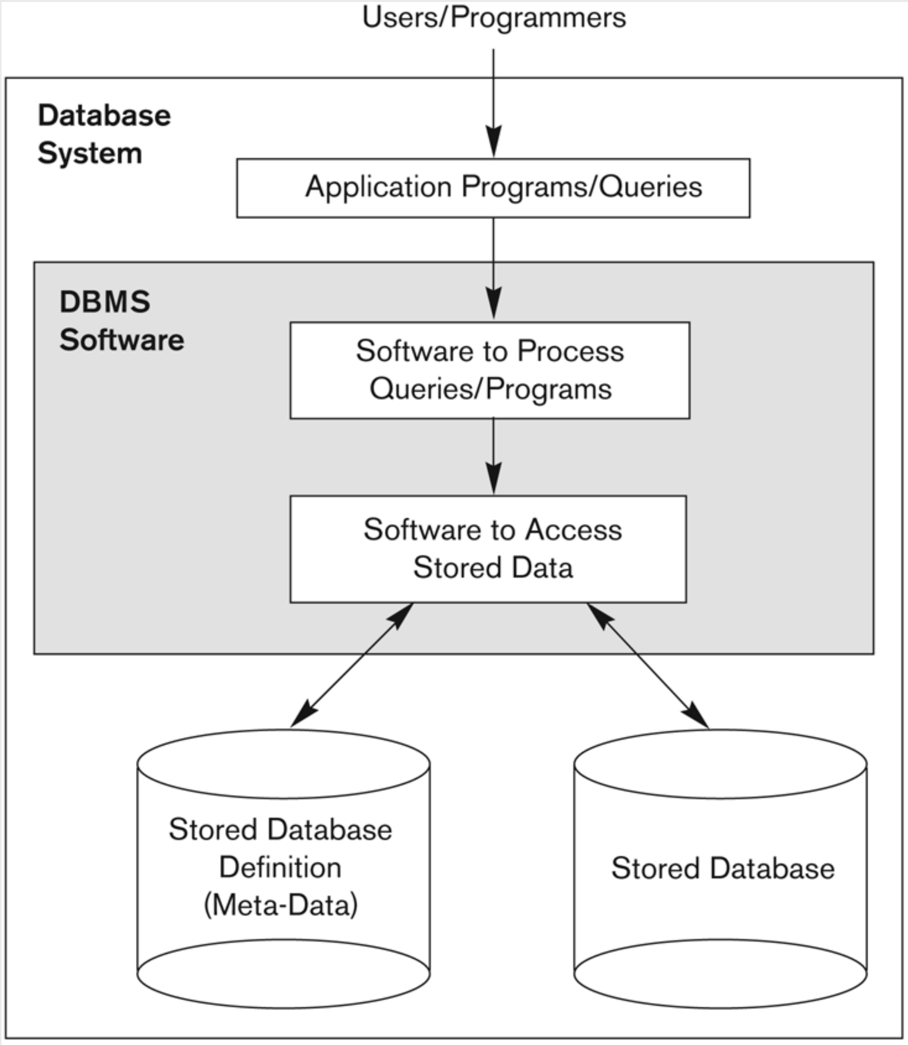


Figure 1.1
A simplified database system environment.

Typical DBMS Functionality

- *Define* a particular database in terms of its data types, structures, and constraints
- *Construct* or Load the initial database contents on a secondary storage medium
- *Manipulating* the database:
  - Retrieval: Querying, generating reports
  - Modification: Insertions, deletions and updates to its content
  - Accessing the database through Web applications
- *Processing* and *Sharing* by a set of concurrent users and application programs – yet, keeping all data valid and consistent

Database Terminology

Entities

- Things we store information about. (i.e. **persons, places, objects, events**, etc.)
- Have **relationships** to **other entities** (i.e. the entity Student has a relationship to the entity Grades in a University Student database

Attributes

- These are **pieces of information** about an **entity** (i.e. Student ID, Name, etc. for the entity Student)

Advantages of the Database Approach

Advantages	Description
Program-data independence	Much easier to evolve and alter software to changing business needs when data and programs are independent.
Minimal data redundancy	Single copy of data assures that data storage is minimized.
Improved data consistency	Eliminating redundancy greatly reduces the opportunities for inconsistency.
Improved data sharing	Easier to deploy and control data access using a centralized system.
Increased productivity of application development	Data standards make it easier to build and modify applications.
Enforcement of standards	A centralized system makes it much easier to enforce standards and rules for data creation, modification, naming, and deletion.
Improved data quality	Centralized control, minimized redundancy, and improved data consistency help to enhance the quality of data.
Improved data accessibility	Centralized system makes it easier to provide access for new personnel within or outside organizational boundaries.
Reduced program maintenance	Information changed in the central database is replicated seamlessly throughout all applications.

Table 3.1 Advantages of the database approach.

Example of a simple database

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	04	King
92	CS1310	Fall	04	Anderson
102	CS3320	Spring	05	Knuth
112	MATH2410	Fall	05	Chang
119	CS1310	Fall	05	Anderson
135	CS3380	Fall	05	Stone

GRADE_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Figure 1.2
A database that stores student and course information.

Example of a simplified database catalog

RELATIONS

Relation_name	No_of_columns
STUDENT	4
COURSE	4
SECTION	5
GRADE_REPORT	3
PREREQUISITE	2

COLUMNS

Column_name	Data_type	Belongs_to_relation
Name	Character (30)	STUDENT
Student_number	Character (4)	STUDENT
Class	Integer (1)	STUDENT
Major	Major_type	STUDENT
Course_name	Character (10)	COURSE
Course_number	XXXXNNNN	COURSE
….	….	…..
….	….	…..
….	….	…..
Prerequisite_number	XXXXNNNN	PREREQUISITE

Figure 1.3
An example of a database catalog for the database in Figure 1.2.

Note: Major_type is defined as an enumerared type with all known majors. XXXXNNNN is used to define a type with four alpha characters followed by four digits

Main Characteristics of the Database Approach

- **Data Abstraction:**
  - A **data model** is used to hide storage details and present the users with a conceptual view of the database.
  - Programs refer to the data model constructs rather than data storage details
- **Support of multiple views of the data:**
  - Each user may see a different view of the database, which describes **only** the data of interest to that user.

Main Characteristics of the Database Approach (continued)

- **Sharing of data and multi-user transaction processing:**
  - Allowing a set of **concurrent users** to retrieve from and to update the database.
  - *Concurrency control* within the DBMS guarantees that each **transaction** is correctly executed or aborted
  - *Recovery* subsystem ensures each completed transaction has its effect permanently recorded in the database
  - **OLTP** (Online Transaction Processing) is a major part of database applications. This allows hundreds of concurrent transactions to execute per second.

Data Models

- **Data Model:**
  - A set of concepts to describe the ***structure*** of a database, the ***operations*** for manipulating these structures, and certain ***constraints*** that the database should obey.
- **Data Model Structure and Constraints:**
  - Constructs are used to define the database structure
  - Constructs typically include ***elements*** (and their ***data types***) as well as groups of elements (e.g. ***entity, record, table***), and ***relationships*** among such groups
  - Constraints specify some restrictions on valid data; these constraints must be enforced at all times

Categories of Data Models

- **Conceptual (high-level, semantic) data models:**
  - Provide concepts that are close to the way many users perceive data.
    - (Also called ***entity-based*** or ***object-based*** data models.)
- **Physical (low-level, internal) data models:**
  - Provide concepts that describe details of how data is stored in the computer. These are usually specified in an ad-hoc manner through DBMS design and administration manuals
- **Implementation (representational) data models:**
  - Provide concepts that fall between the above two, used by many commercial DBMS implementations (e.g. relational data models used in many commercial systems).

Schemas versus Instances

- Database Schema:
  - The ***description*** of a database.
  - Includes descriptions of the database structure, data types, and the constraints on the database.
- Schema Diagram:
  - An ***illustrative*** display of (most aspects of) a database schema.
- Schema Construct:
  - A ***component*** of the schema or an object within the schema, e.g., STUDENT, COURSE.

Example of a Database Schema

STUDENT

Name	Student_number	Class	Major

COURSE

Course_name	Course_number	Credit_hours	Department

PREREQUISITE

Course_number	Prerequisite_number

SECTION

Section_identifier	Course_number	Semester	Year	Instructor

GRADE_REPORT

Student_number	Section_identifier	Grade

Figure 2.1
Schema diagram for the database in Figure 1.2.

DBMS Languages

- Data Definition Language (DDL)
- Data Manipulation Language (DML)
  - High-Level or Non-procedural Languages: These include the relational language -- Structured Query Language (SQL)
    - May be used in a standalone way or may be embedded in a programming language
  - Low Level or Procedural Languages:
    - These must be embedded in a programming language

ER Model Concepts

- Entities and Attributes
  - Entities are specific objects or things in the mini-world that are represented in the database.
    - For example the EMPLOYEE John Smith, the Research DEPARTMENT, the ProductX PROJECT
  - Attributes are properties used to describe an entity.
    - For example an EMPLOYEE entity may have the attributes Name, SSN, Address, Sex, BirthDate
  - A specific entity will have a value for each of its attributes.
    - For example a specific employee entity may have Name='John Smith', SSN='123456789', Address ='731, Fondren, Houston, TX', Sex='M', BirthDate='09-JAN-55‘
  - Each attribute has a *value set* (or data type) associated with it – e.g. integer, string, subrange, enumerated type, …

Types of Attributes (1)

- Simple
  - Each entity has a single atomic value for the attribute. For example, SSN or Sex.
- Composite
  - The attribute may be composed of several components. For example:
    - Address(Apt#, House#, Street, City, State, ZipCode, Country), or
    - Name(FirstName, MiddleName, LastName).
    - Composition may form a hierarchy where some components are themselves composite.
- Multi-valued
  - An entity may have multiple values for that attribute. For example, Color of a CAR or PreviousDegrees of a STUDENT.
    - Denoted as {Color} or {PreviousDegrees}.

Entity Types and Key Attributes (1)

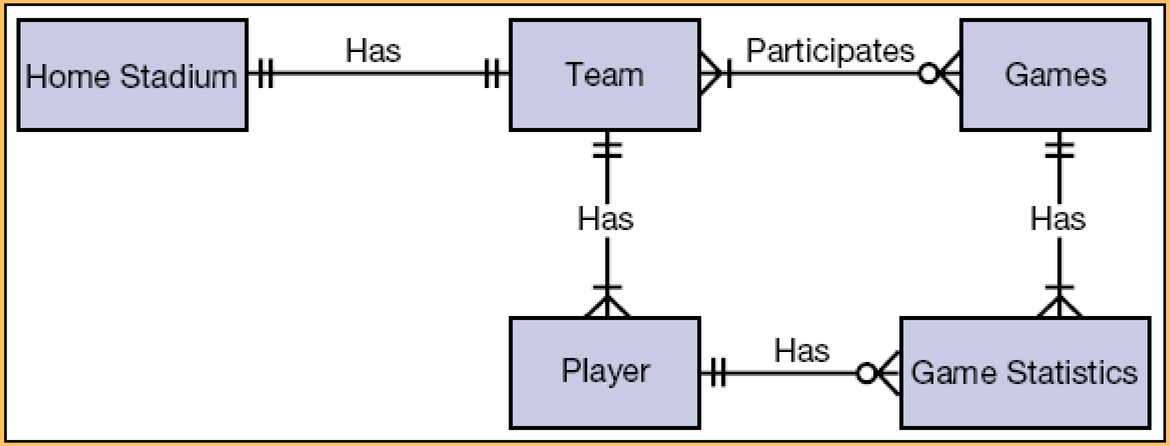
- Entities with the same basic attributes are grouped or typed into an entity type.
  - For example, the entity type EMPLOYEE and PROJECT.
- An attribute of an entity type for which each entity must have a unique value is called a key attribute of the entity type.
  - For example, SSN of EMPLOYEE.

Designing Databases – Data Model

ER or EER Data Model (Conceptual Data Model)

- A map or diagram that represents **entities** and their **relationships**
- Used by Database Administrators to design **tables** with their corresponding **associations**

Example: ERD (Entity Relationship Diagram)



Designing Databases – Keys

Database Keys
Mechanisms used to identify, select, and maintain one or more records using an application program, query, or report

Primary Key
A unique attribute type used to identify a single instance of an entity.

Compound Primary Key
A unique combination of attributes types used to identify a single instance of an entity

Secondary Key
An attribute that can be used to identify one or more records within a table with a given value

Entity Types and Key Attributes (2)

- A key attribute may be composite.
  - VehicleTagNumber is a key of the CAR entity type with components (Number, State).
- An entity type may have more than one key.
  - The CAR entity type may have two keys:
    - VehicleIdentificationNumber (popularly called VIN)
    - VehicleTagNumber (Number, State), aka license plate number.
- Each key is <u>underlined</u>

Entity Type CAR with two keys and a corresponding Entity Set

(a)

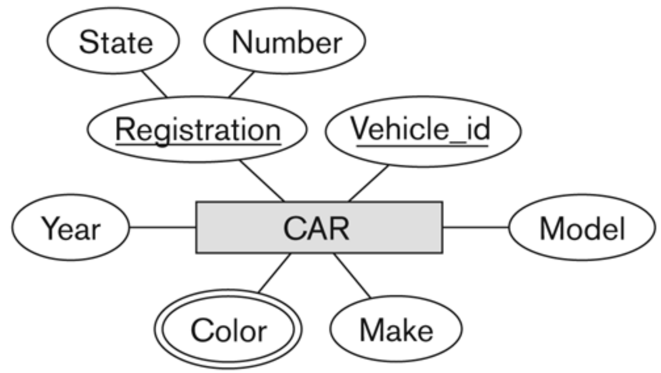


Figure 3.7
The CAR entity type with two key attributes, Registration and Vehicle_id. (a) ER diagram notation. (b) Entity set with three entities.

(b)

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

CAR_1
((ABC 123, TEXAS), TK629, Ford Mustang, convertible, 2004 {red, black})

CAR_2
((ABC 123, NEW YORK), WP9872, Nissan Maxima, 4-door, 2005, {blue})

CAR_3
((VSY 720, TEXAS), TD729, Chrysler LeBaron, 4-door, 2002, {white, blue})

⋮

Initial Design of Entity Types:
EMPLOYEE, DEPARTMENT, PROJECT, DEPENDENT

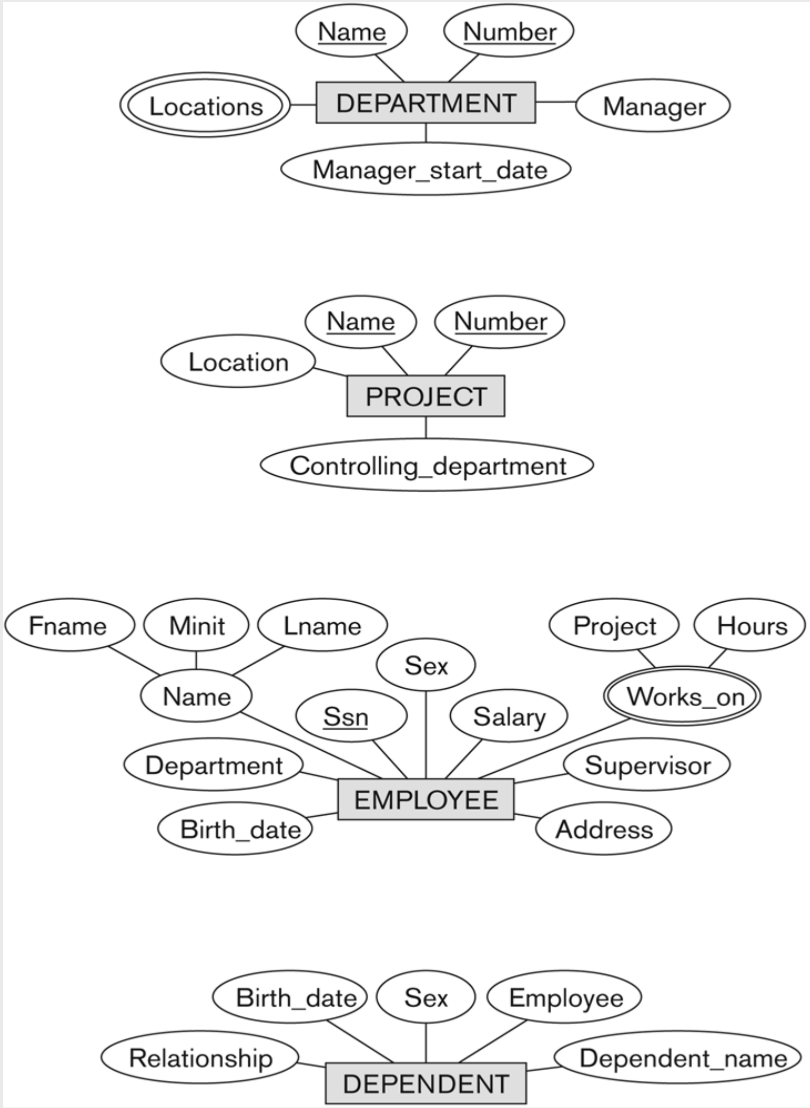


Figure 3.8
Preliminary design of entity types for the COMPANY database. Some of the shown attributes will be refined into relationships.

Refining the initial design by introducing **relationships**

- The initial design is typically not complete
- Some aspects in the requirements will be represented as **relationships**
- ER model has three main concepts:
  - Entities (and their entity types and entity sets)
  - Attributes (simple, composite, multivalued)
  - Relationships (and their relationship types and relationship sets)

ER DIAGRAM – Relationship Types are:

WORKS_FOR, MANAGES, WORKS_ON, CONTROLS, SUPERVISION, DEPENDENTS_OF

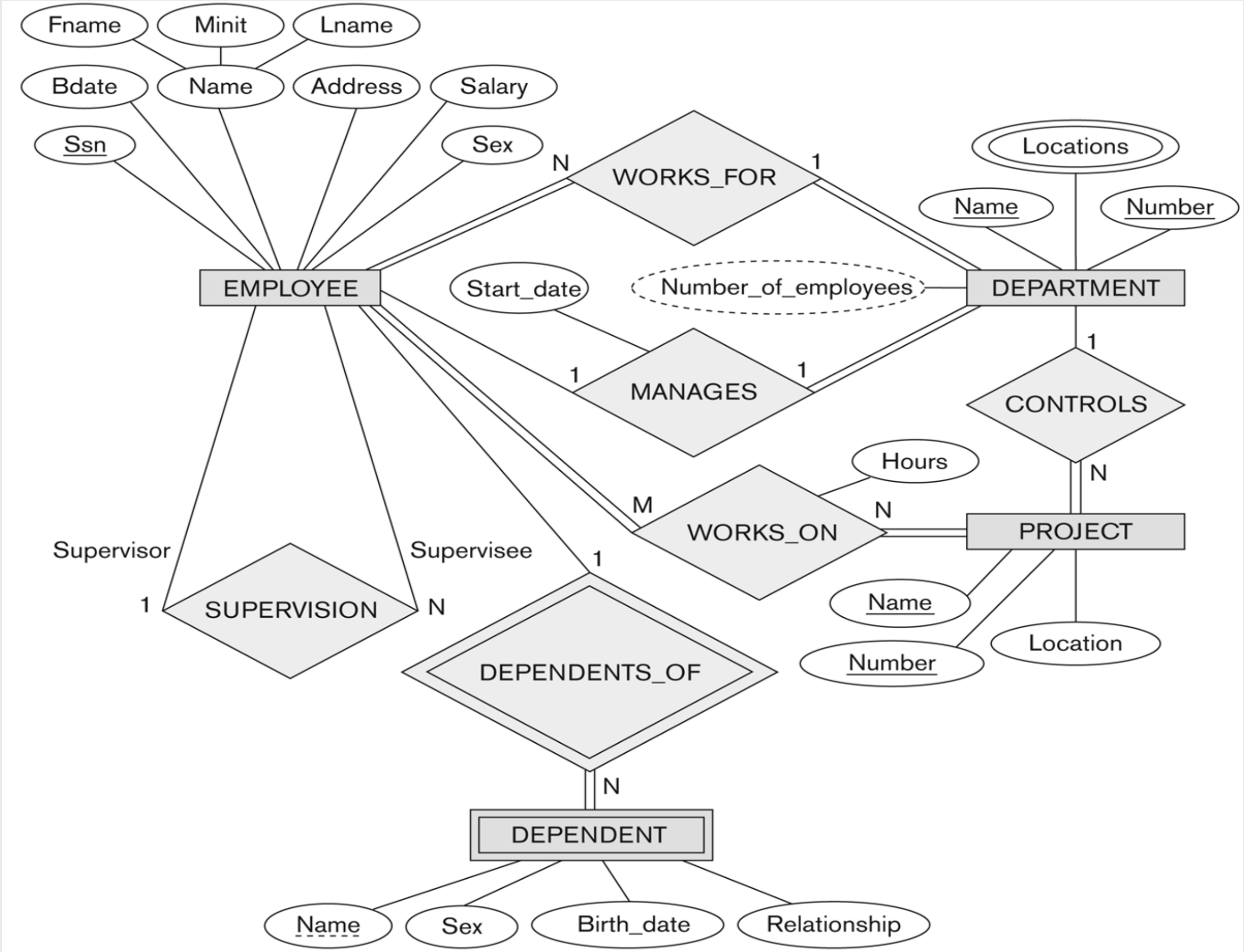


Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

Weak Entity Types

- An entity that does not have a key attribute
- A weak entity must participate in an identifying relationship type with an owner or identifying entity type
- Entities are identified by the combination of:
  - A partial key of the weak entity type
  - The particular entity they are related to in the identifying entity type
- **Example:**
  - A DEPENDENT entity is identified by the dependent's first name, *and* the specific EMPLOYEE with whom the dependent is related
  - Name of DEPENDENT is the *partial key*
  - DEPENDENT is a *weak entity type*
  - EMPLOYEE is its identifying entity type via the identifying relationship type DEPENDENT_OF

Constraints on Relationships

- Constraints on Relationship Types
  - (Also known as ratio constraints)
  - Cardinality Ratio (specifies *maximum* participation)
    - One-to-one (1:1)
    - One-to-many (1:N) or Many-to-one (N:1)
    - Many-to-many (M:N)
  - Existence Dependency Constraint (specifies *minimum* participation) (also called participation constraint)
    - zero (optional participation, not existence-dependent)
    - one or more (mandatory participation, existence-dependent)

Recursive Relationship Type is: SUPERVISION (participation role names are shown)

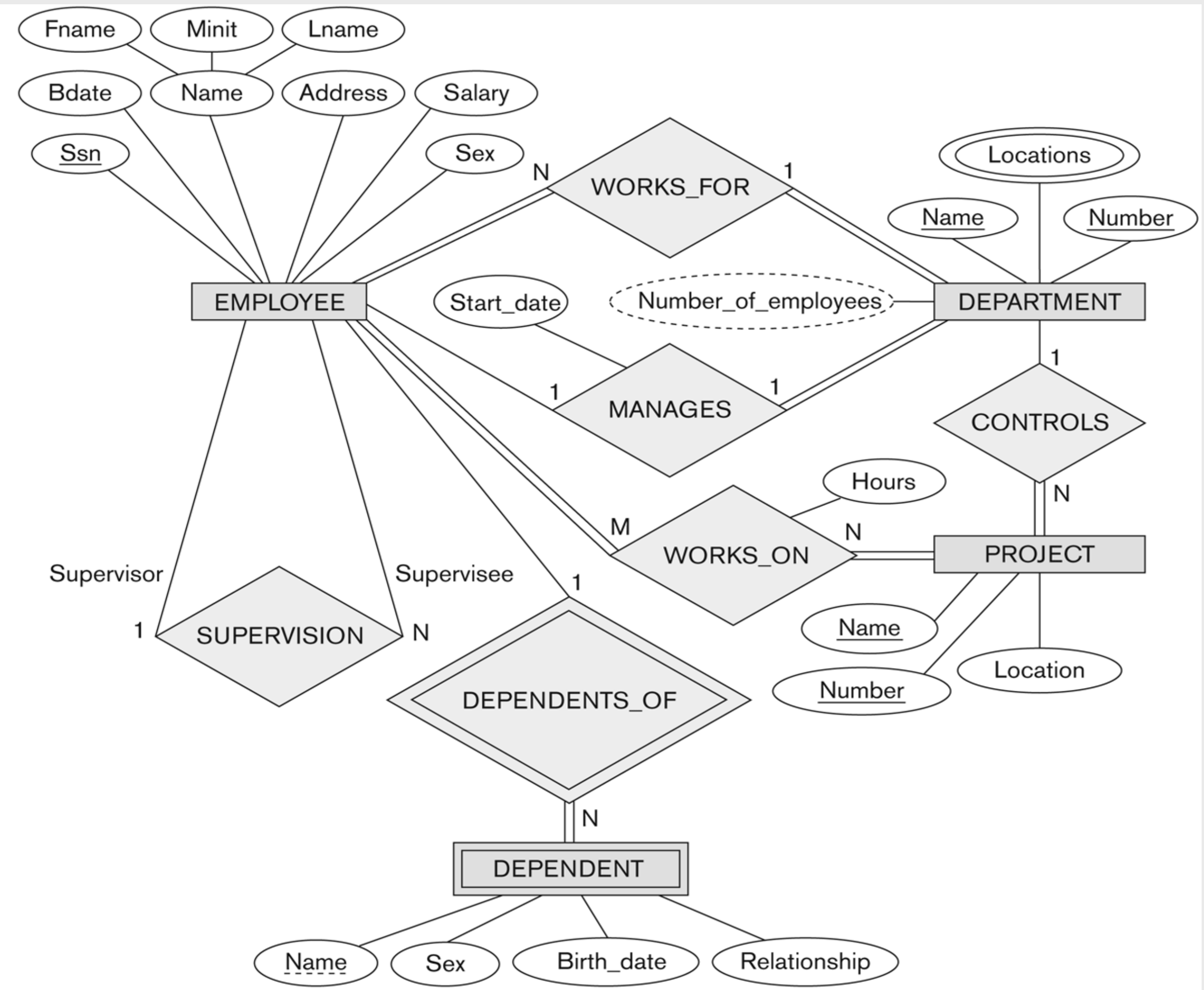


Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

Example Attribute of a Relationship Type: Hours of WORKS_ON

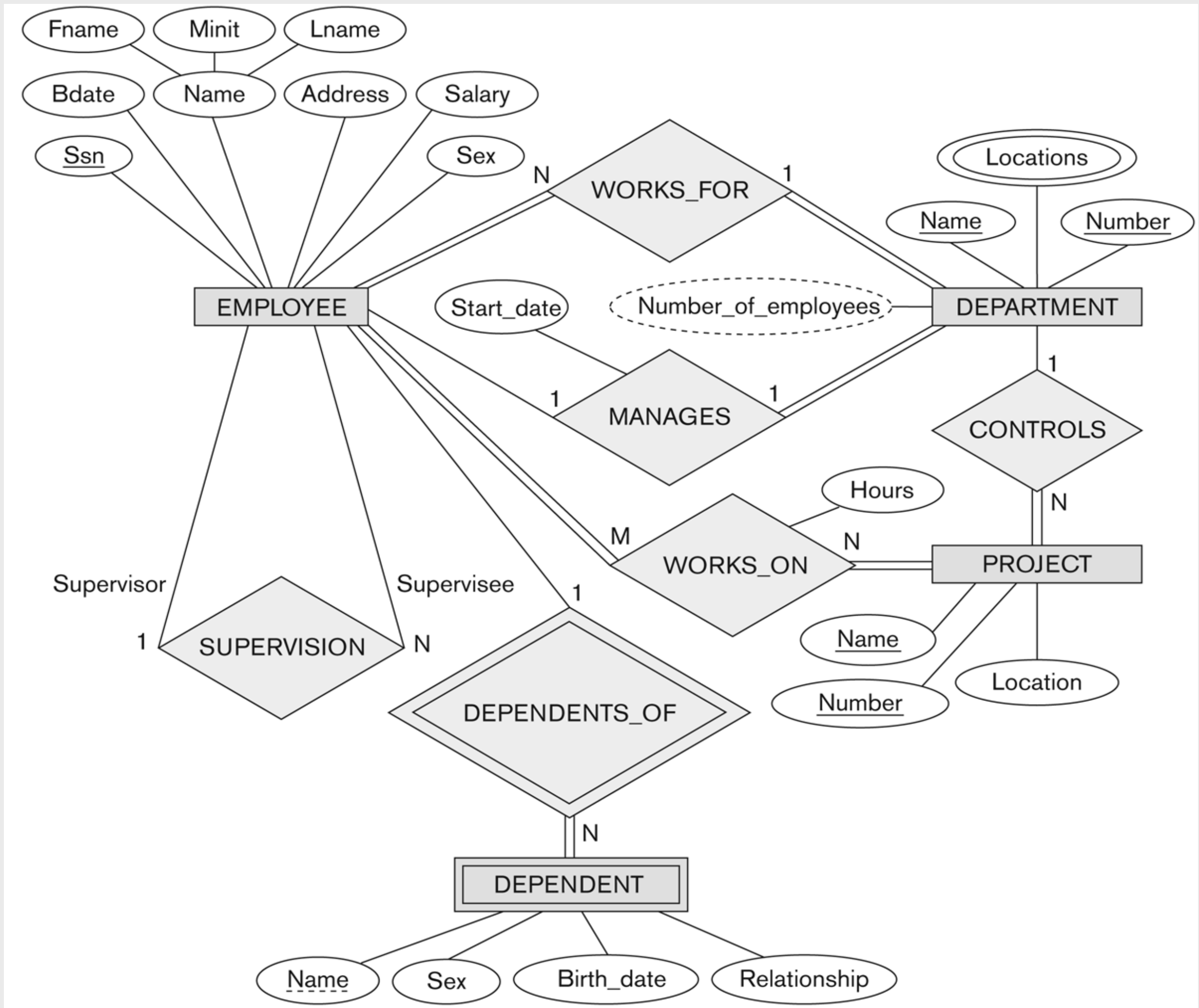


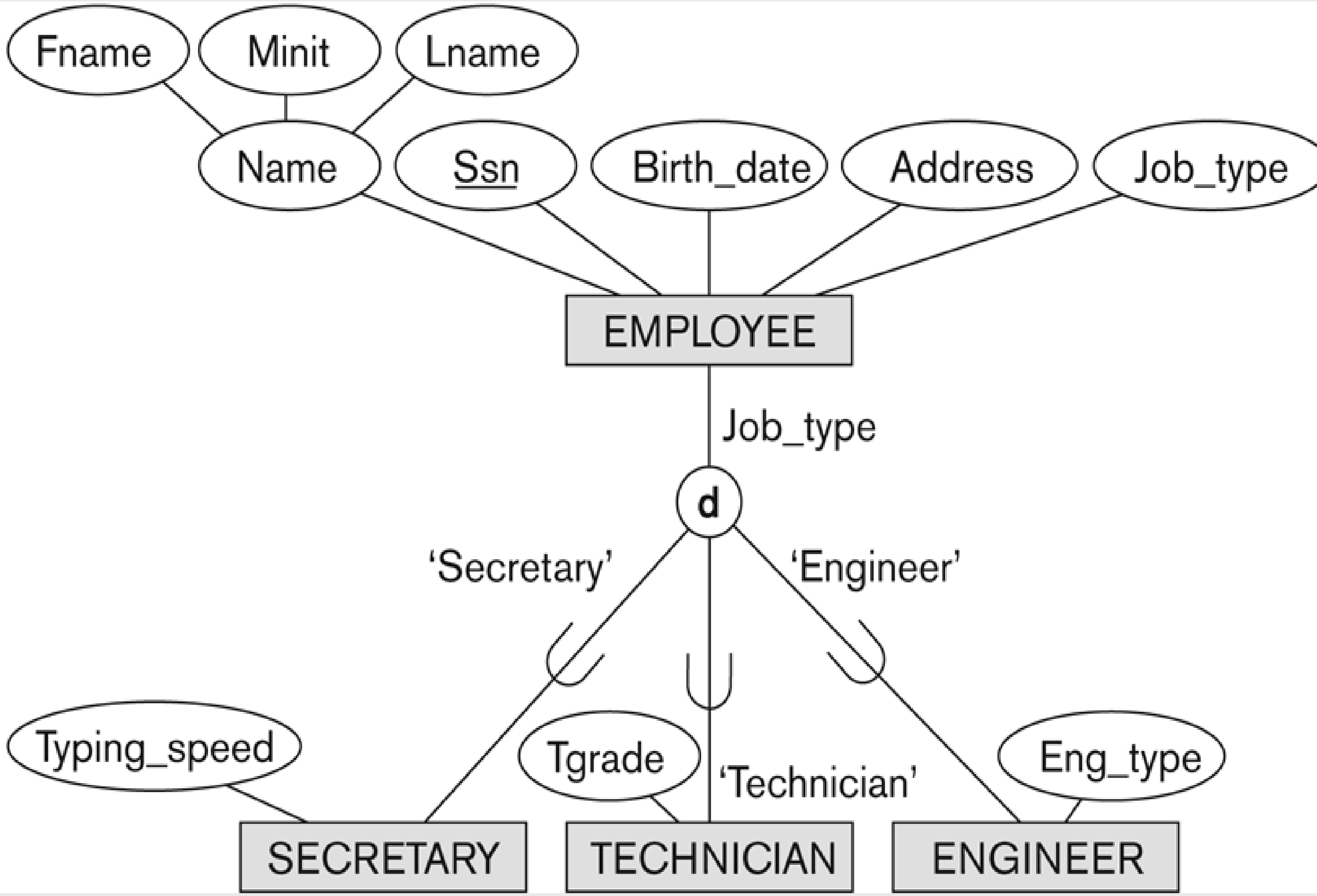
Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

EER Model

- EER stands for Enhanced ER or Extended ER
- EER Model Concepts
  – Includes all modeling concepts of basic ER
  – Additional concepts:
    - subclasses/superclasses
    - specialization/generalization
    - categories (UNION types)
    - attribute and relationship inheritance
  – These are fundamental to conceptual modeling
- The additional EER concepts are used to model applications more completely and more accurately
  – EER includes some object-oriented concepts, such as inheritance

Representing Specialization in EER Diagrams

Figure 4.4
EER diagram notation for an attribute-defined specialization on Job_type.



Attribute Inheritance in Superclass / Subclass Relationships

- An entity that is member of a subclass *inherits*
  - All attributes of the entity as a member of the superclass
  - All relationships of the entity as a member of the superclass
- Example:
  - In the previous slide, SECRETARY (as well as TECHNICIAN and ENGINEER) inherit the attributes Name, SSN, …, from EMPLOYEE
  - Every SECRETARY entity will have values for the inherited attributes

Specialization (1)

- Specialization is the process of defining a set of subclasses of a superclass
- The set of subclasses is based upon some distinguishing characteristics of the entities in the superclass
  - Example: {SECRETARY, ENGINEER, TECHNICIAN} is a specialization of EMPLOYEE based upon *job type.*
    - May have several specializations of the same superclass

Specialization (3)

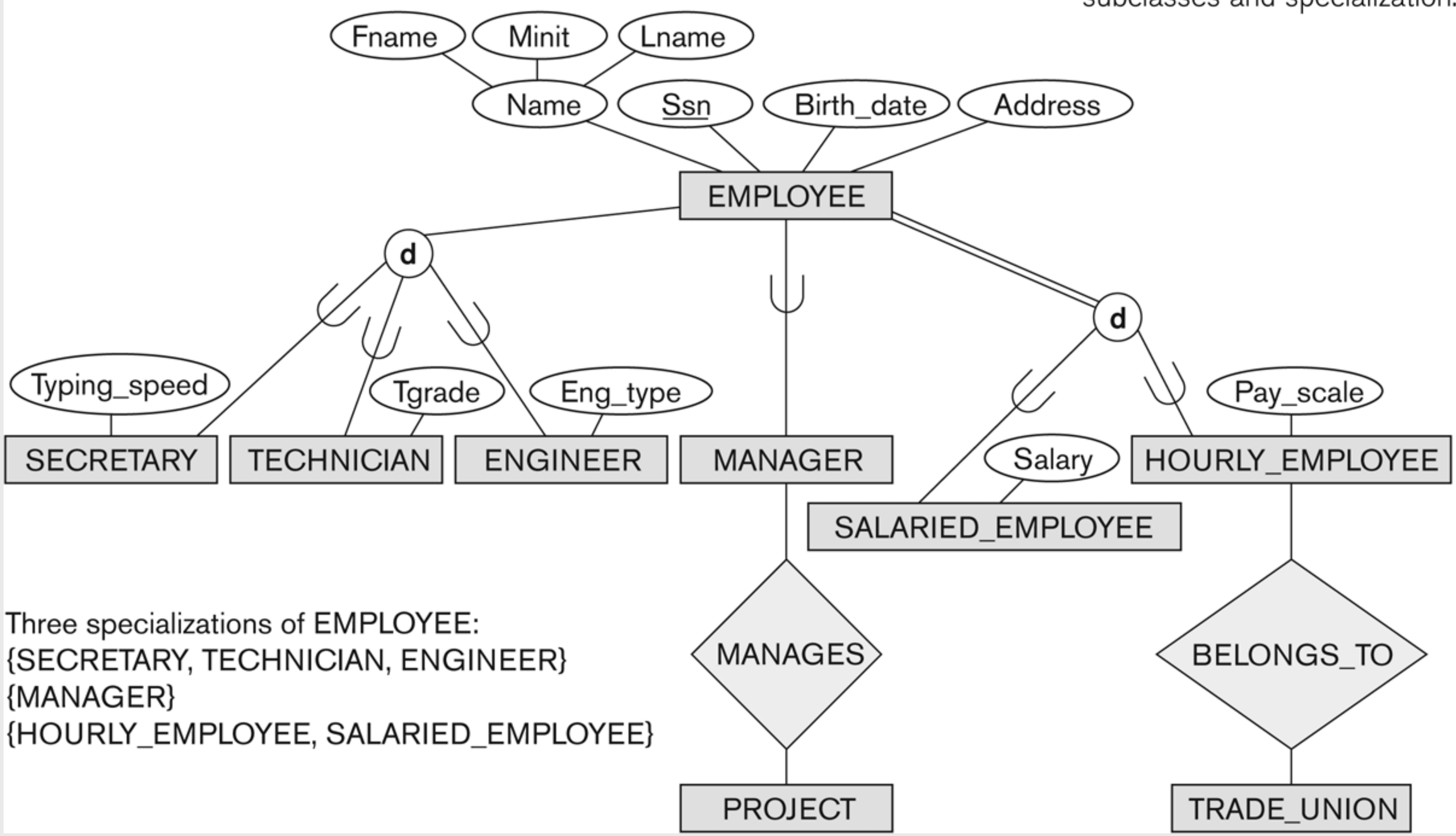


Figure 4.1
EER diagram notation to represent subclasses and specialization.

Generalization

- Generalization is the reverse of the specialization process
- Several classes with common features are generalized into a superclass;
  - original classes become its subclasses
- Example: CAR, TRUCK generalized into VEHICLE;
  - both CAR, TRUCK become subclasses of the superclass VEHICLE.
  - We can view {CAR, TRUCK} as a specialization of VEHICLE
  - Alternatively, we can view VEHICLE as a generalization of CAR and TRUCK

Generalization (2)

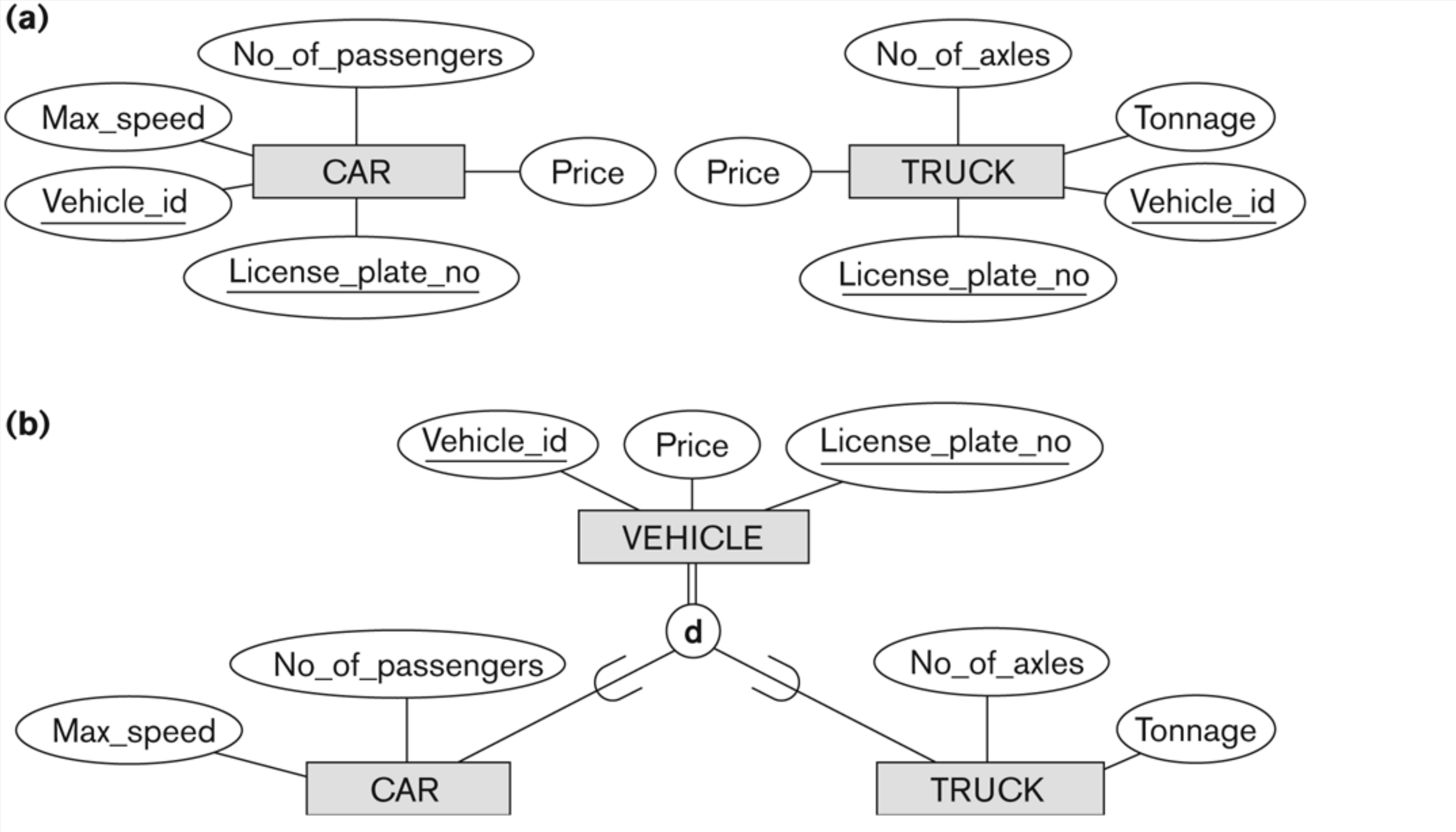


Figure 4.3
Generalization. (a) Two entity types, CAR and TRUCK.
(b) Generalizing CAR and TRUCK into the superclass VEHICLE.

Designing Databases – Keys (Example)

Primary Key
- Student ID

Secondary Key
- Major

ENTITIES

Students
Student ID Name Campus Address Major Phone

Grades
Student ID Course ID Section No. Term Grade

Entities are translated into Tables (Students and Grades)

Students

Student ID	Name	Campus Address	Major	Phone
555-39-3232	Joe Jones	123 Any Avenue	Finance	335-2211
289-42-8776	Sally Carter	1200 Wolf Street #12	Marketing	335-8702

Entities are joined by common attributes

Grades

Student ID	Course ID	Section No.	Term	Grade
555-39-3232	MIS 250	2	F'00	D+
555-39-3232	MIS 250	1	F'01	A–
289-42-8776	MIS 250	3	S'02	B+

Compound Primary Key
- Student ID
- Course ID
- Sec No.
- Term

Designing Databases - Associations

Associations

- Define the relationships one entity has to another
- Determine necessary key structures to access data
- Come in three relationship types:
  - One-to-One
  - One-to-Many
  - Many-to-Many

Foreign Key

- An attribute that appears as a non-primary key in one entity (table) and as a primary key attribute in another entity (table)

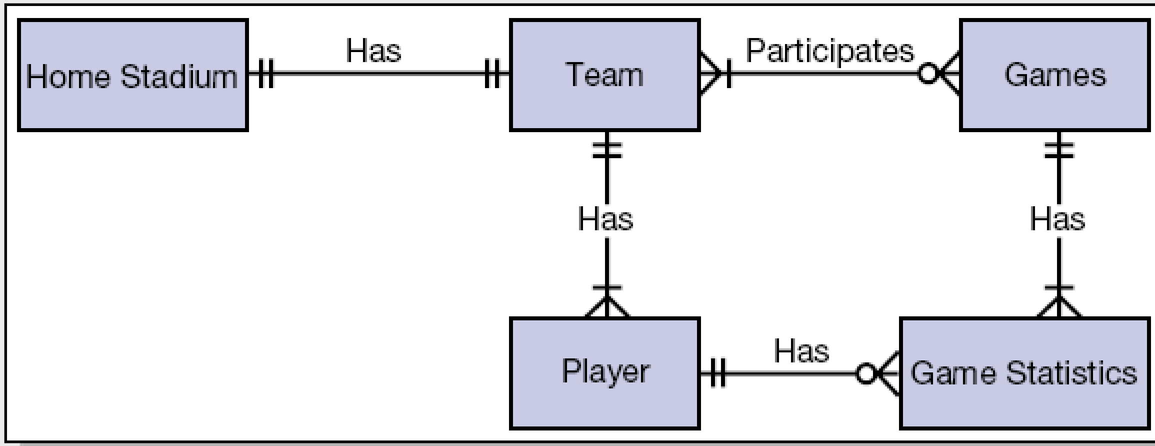
Designing Databases - Associations

Entity Relationship Diagram (ERD)

- Diagramming tool used to express entity relationships
- Very useful in developing complex databases

Example

- Each Home Stadium has a Team (One-to-One)
- Each Team has Players (One-to-Many)
- Each Team Participates in Games
- For each Player and Game there are Game Statistics



Designing Databases - Associations

Relationship	Example	Instructions
One-to-One	Each team has only one home stadium, and each home stadium has only one team.	Place the primary key from each table in the table for the other entity as a foreign key.
One-to-Many	Each player is on only one team, but each team has many players.	Place the primary key from the entity on the one side of the relationship as a foreign key in the table for the entity on the many side of the relationship.
Many-to-Many	Each player participates in games, and each game has many players.	Create a third entity/table and place the primary keys from each of the original entities together in the third table as a combination primary key.

Designing Databases – Associations (Example)

A. One-to-one relationship: Each team has only one home stadium, and each home stadium has only one team.

Team

<u>Team ID</u>	Team Name	*Stadium ID*

B. One-to-many relationship: Each player is on only one team, but each team has many players.

Player

<u>Player ID</u>	Player Name	Position	*Team ID*

C. Many-to-many relationship: Each player participates in many games, and each game has many players.

Player Statistics

<u>Team 1</u>	*<u>Team 2</u>*	*<u>Date</u>*	*<u>Player ID</u>*	Points	Minutes	Fouls

The Relational Model

The Relational Model

- The most **common** type of **database model** used today in organizations
- Is a **three-dimensional model** compared to the traditional two-dimensional database models
  - Rows (first-dimension)
  - Columns (second-dimension)
  - Relationships (third-dimension)
- The **third-dimension** makes this model so powerful because **any row** of data can be **related** to **any other row or rows** of data

Informal Definitions

- 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**
  - In the formal model, rows are called **tuples**
- Each **column** has a column header that gives an indication of the meaning of the data items in that column
  - In the formal model, the column header is called an **attribute name** (or just **attribute**)

The Relational Model - Example

Department Records

Department No	Dept Name	Location	Dean
Dept A			
Dept B			
Dept C			

Instructor Records

Instructor No	Inst Name	Title	Salary	Dept No
Inst 1				
Inst 2				
Inst 3				
Inst 4				

Figure 3.11 ➥ With the relational model, we represent these two entities, department and instructor, as two separate tables and capture the relationship between them with a common column in each table.

Characteristics Of Relations

- Values in a tuple:
  - All values are considered atomic (indivisible).
  - Each value in a tuple must be from the domain of the attribute for that column
    - If tuple t = <v1, v2, …, vn> is a tuple (row) in the relation state r of R(A1, A2, …, An)
    - Then each *vi* must be a value from *dom(Ai)*
  - A special **null** value is used to represent values that are unknown or inapplicable to certain tuples.

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 (continued)

- If a relation has several **candidate keys**, one is chosen arbitrarily to be the **primary key**.
  - The primary key attributes are <u>underlined</u>.
- Example: Consider the CAR relation schema:
  - CAR(State, Reg#, <u>SerialNo</u>, Make, Model, Year)
  - We chose SerialNo as the primary key
- The primary key value is used to *uniquely identify* each tuple in a relation
  - Provides the tuple identity
- Also used to *reference* the tuple from another tuple
  - General rule: Choose as primary key the smallest of the candidate keys (in terms of size)
  - Not always applicable – choice is sometimes subjective

Relational Database Schema

- **Relational Database Schema:**
  - A set S of relation schemas that belong to the same database.
  - 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
- Following slide shows a COMPANY database schema with 6 relation schemas

COMPANY 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.

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)

Referential Integrity

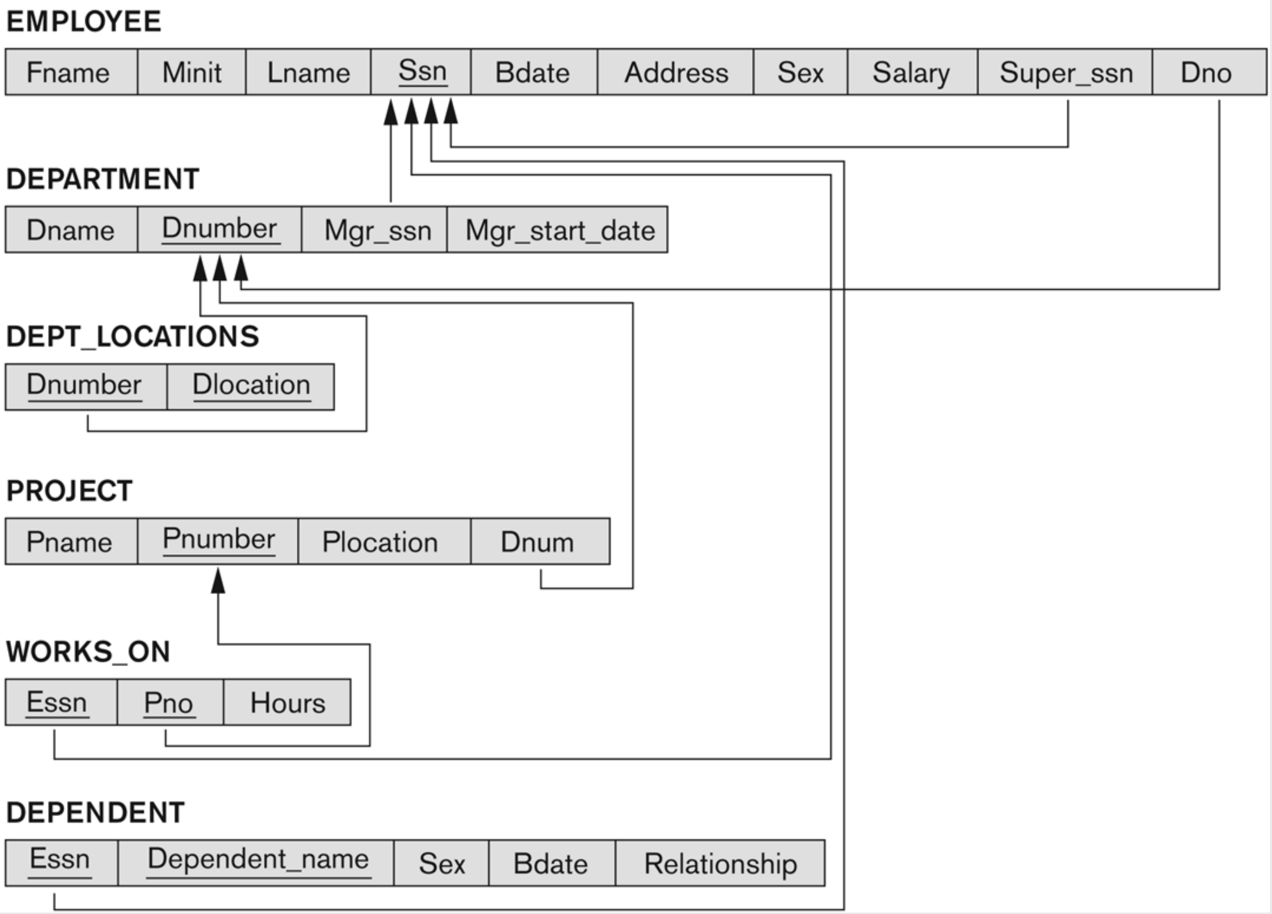
- Tuples in the **referencing relation** R1 have attributes FK (called **foreign key** attributes) that reference the primary key attributes PK of the **referenced relation** R2.
  - A tuple t1 in R1 is said to **reference** a tuple t2 in R2 if t1[FK] = t2[PK].
- A referential integrity constraint can be displayed in a relational database schema as a directed arc from R1.FK to R2.

Referential Integrity (or foreign key) Constraint

- Statement of the constraint
  - The value in the foreign key column (or columns) FK of the the **referencing relation** R1 can be **either**:
    - (1) a value of an existing primary key value of a corresponding primary key PK in the **referenced relation** R2, <u>or</u>
    - (2) a **null**.
- In case (2), the FK in R1 should **not** be a part of its own primary key.

Referential Integrity Constraints for COMPANY database

Figure 5.7
Referential integrity constraints displayed on the COMPANY relational database schema.



ER-to-Relational Mapping

- **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.
  – Step 9: Mapping of Union Types (Categories).

- **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 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.

ER-to-Relational Mapping Algorithm (contd.)

- 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 relationship 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.

ER-to-Relational Mapping Algorithm (contd.)

- **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}

FIGURE 4.11

Ternary relationship types. (a) The SUPPLY relationship.

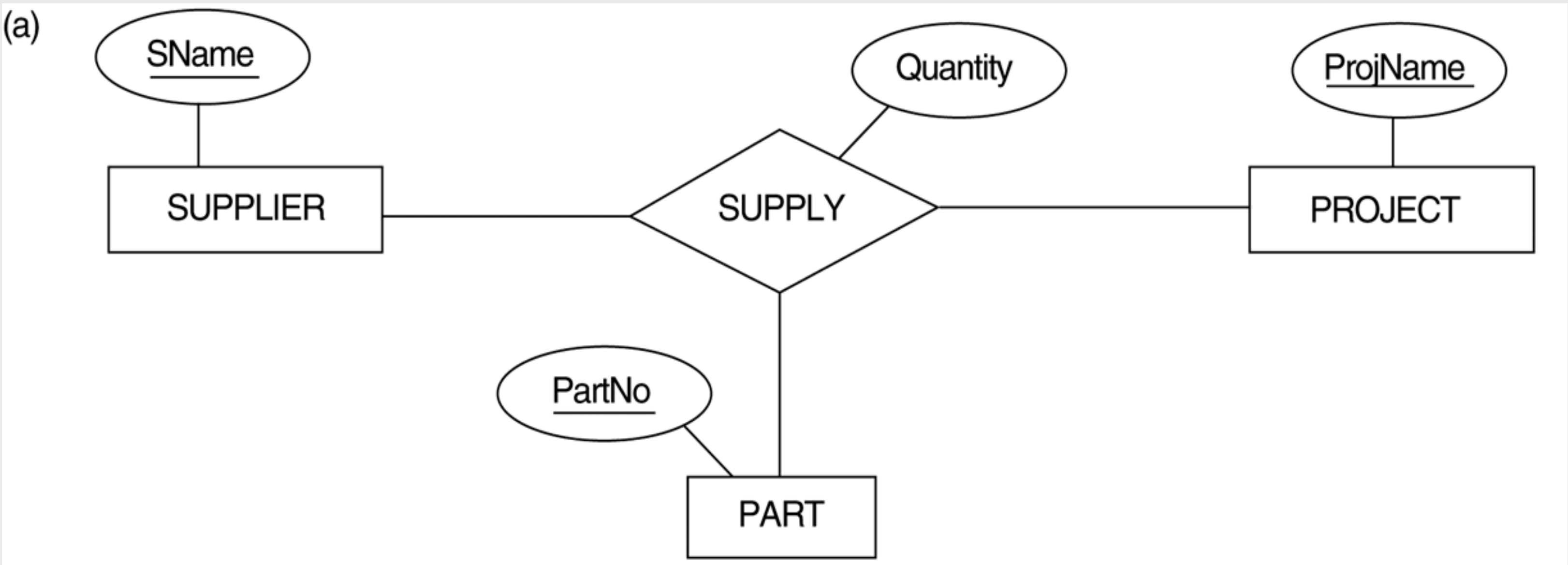
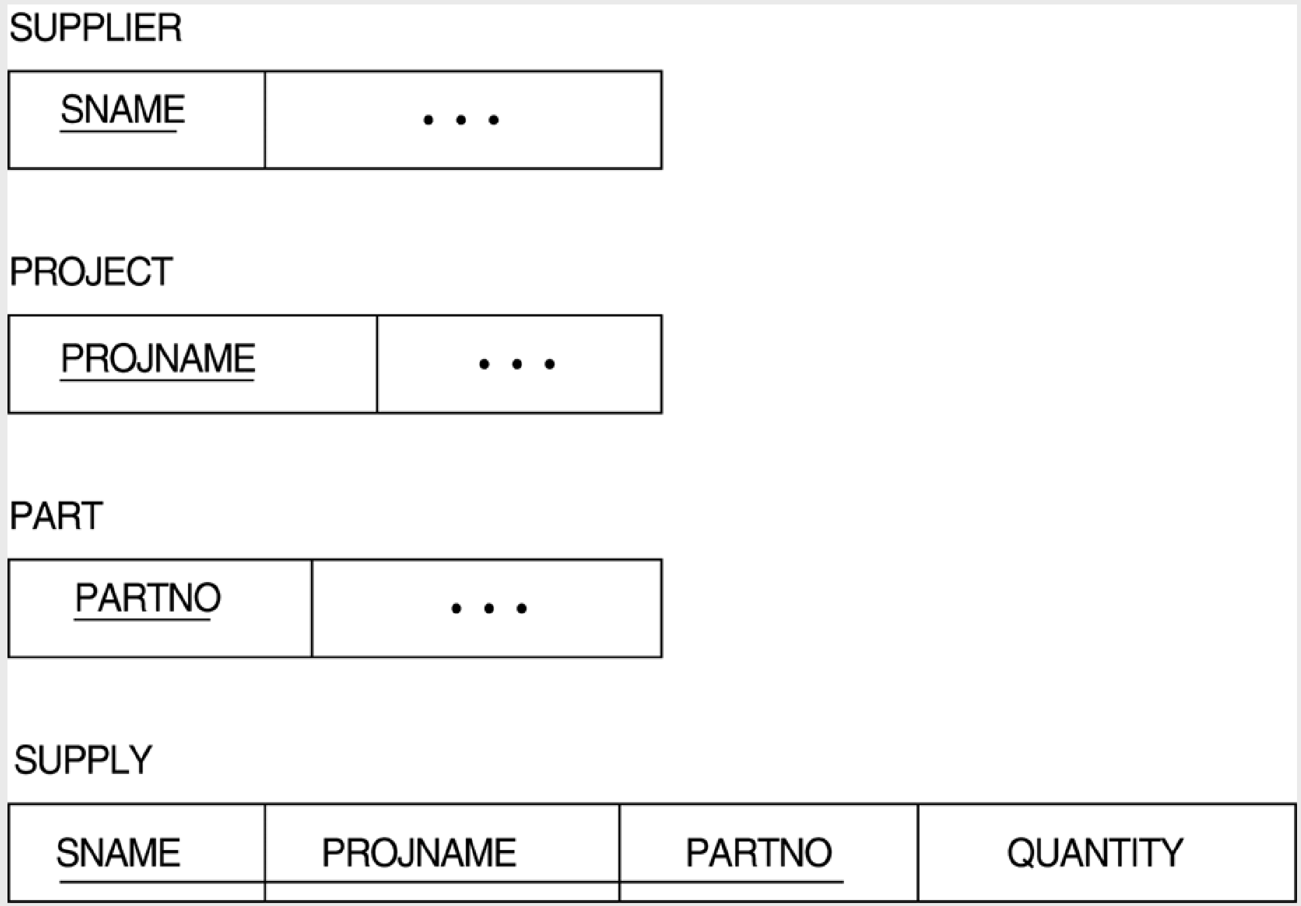


FIGURE 7.3

Mapping the *n*-ary relationship type SUPPLY from Figure 4.11a.



The Relational Model - Normalization

Normalization

- A technique to make complex databases more efficient by eliminating as much redundant data as possible
- Example: Database with redundant data (below)

Microsoft Access - [Student_Course_Grades : Table]

ID	Student ID	Student Name	Campus Address	Major	Phone	Course ID	Course Title	Instructor Nar	Instructor Loc	Instructor Ph	Term	Grade
1	A121	Joy Egbert	100 N. State Street	MIS	555-7771	MIS 350	Intro. MIS	Hess	T240C	555-2222	F'04	A
2	A121	Joy Egbert	100 N. State Street	MIS	555-7771	MIS 372	Database	Sarker	T240F	555-2224	F'04	B
3	A121	Joy Egbert	100 N. State Street	MIS	555-7771	MIS 375	Elec. Comm.	Wells	T240D	555-2228	F'04	B+
4	A121	Joy Egbert	100 N. State Street	MIS	555-7771	MIS 426	Strategic MIS	Datta	T240E	555-2227	F'04	A-
5	A121	Joy Egbert	100 N. State Street	MIS	555-7771	MIS 374	Telecomm	Marrett	T240A	555-2221	F'04	C+
6	A123	Larry Mueller	123 S. State Street	MIS	555-1235	MIS 350	Intro. MIS	Hess	T240C	555-2222	F'04	A
7	A123	Larry Mueller	123 S. State Street	MIS	555-1235	MIS 372	Database	Sarker	T240F	555-2224	F'04	B-
8	A123	Larry Mueller	123 S. State Street	MIS	555-1235	MIS 375	Elec. Comm.	Wells	T240D	555-2228	F'04	A-
9	A123	Larry Mueller	123 S. State Street	MIS	555-1235	MIS 426	Strategic MIS	Datta	T240E	555-2227	F'04	C+
10	A124	Mike Guon	125 S. Elm	MGT	555-2214	MIS 350	Intro. MIS	Hess	T240C	555-2222	F'04	A-
11	A124	Mike Guon	125 S. Elm	MGT	555-2214	MIS 372	Database	Sarker	T240F	555-2224	F'04	A-
12	A124	Mike Guon	125 S. Elm	MGT	555-2214	MIS 375	Elec. Comm.	Wells	T240D	555-2228	F'04	B+
13	A124	Mike Guon	125 S. Elm	MGT	555-2214	MIS 374	Telecomm	Marrett	T240A	555-2221	F'04	B
14	A126	Jackie Judson	224 S. Sixth Street	MKT	555-1245	MIS 350	Intro. MIS	Hess	T240C	555-2222	F'04	A
15	A126	Jackie Judson	224 S. Sixth Street	MKT	555-1245	MIS 372	Database	Sarker	T240F	555-2224	F'04	B+
16	A126	Jackie Judson	224 S. Sixth Street	MKT	555-1245	MIS 375	Elec. Comm.	Wells	T240D	555-2228	F'04	B+
17	A126	Jackie Judson	224 S. Sixth Street	MKT	555-1245	MIS 374	Telecomm	Marrett	T240A	555-2221	F'04	A-
nber)												

Figure 3.12 ➡ Database of students, courses, instructors, and grades with redundant data.

The Relational Model - Normalization

Normalized Database

Microsoft Access

Enrolled Table : Table

Student ID	Course ID	Term	Grade
A121	MIS 350	F'04	A
A121	MIS 372	F'04	B
A121	MIS 374	F'04	C+
A121	MIS 375	F'04	B+
A121	MIS 426	F'04	A-
A123	MIS 350	F'04	A
A123	MIS 372	F'04	B-
A123	MIS 375	F'04	A-
A123	MIS 426	F'04	C+
A124	MIS 350	F'04	A-
A124	MIS 372	F'04	A-
A124	MIS 374	F'04	B
A124	MIS 375	F'04	B+
A126	MIS 350	F'04	A
A126	MIS 372	F'04	B+
A126	MIS 374	F'04	A-
A126	MIS 375	F'04	B+

Record: 1 of 17

Teaching Assignment : Table

Course ID	Instructor Name	Instructor Phone	Term
MIS 350	Hess	555-2222	F'04
MIS 372	Sarker	555-2224	F'04
MIS 374	Marrett	555-2221	F'04
MIS 375	Wells	555-2228	F'04
MIS 426	Datta	555-2227	F'04

Record: 1 of 5

Instructor Table : Table

Instructor Na	Instructor Loc	Instructor Phc
Hess	T240C	555-2222
Sarker	T240F	555-2224
Wells	T240D	555-2228
Datta	T240E	555-2227
Marrett	T240A	555-2221

Record: 1 of 5

Class Table : Table

Course ID	Course Title
MIS 350	Intro. MIS
MIS 372	Database
MIS 375	Elec. Comm.
MIS 426	Strategic MIS
MIS 374	Telecomm

Record: 6 of 6

Student Table : Table

Student ID	Student Name	Campus Address	Major	Phone
A121	Joy Egbert	100 N. State Street	MIS	555-7771
A123	Larry Mueller	123 S. State Street	MIS	555-1235
A124	Mike Guon	125 S. Elm	MGT	555-2214
A126	Jackie Judson	224 S. Sixth Street	MKT	555-1245

Record: 1 of 4

Datasheet View CAPS NUM

Figure 3.13 ➥ Organization of information on students, courses, instructors, and grades after normalization.

The Relational Model – Data Dictionary

Data Dictionary

- Is a document that database designers prepare to help individuals enter data
- Provides several pieces of information about each attribute in the database including:
  - **Name**
  - **Key** (is it a key or part of a key)
  - **Data Type** (date, alpha-numeric, numeric, etc.)
  - **Valid Value** (the format or numbers allowed)
- Can be used to enforce **Business Rules** which are captured by the database designer to prevent illegal or illogical values from entering the database. (e.g. who has authority to enter certain kinds of data)

