Relational Database Design

Design of ER Model in ER Diagram

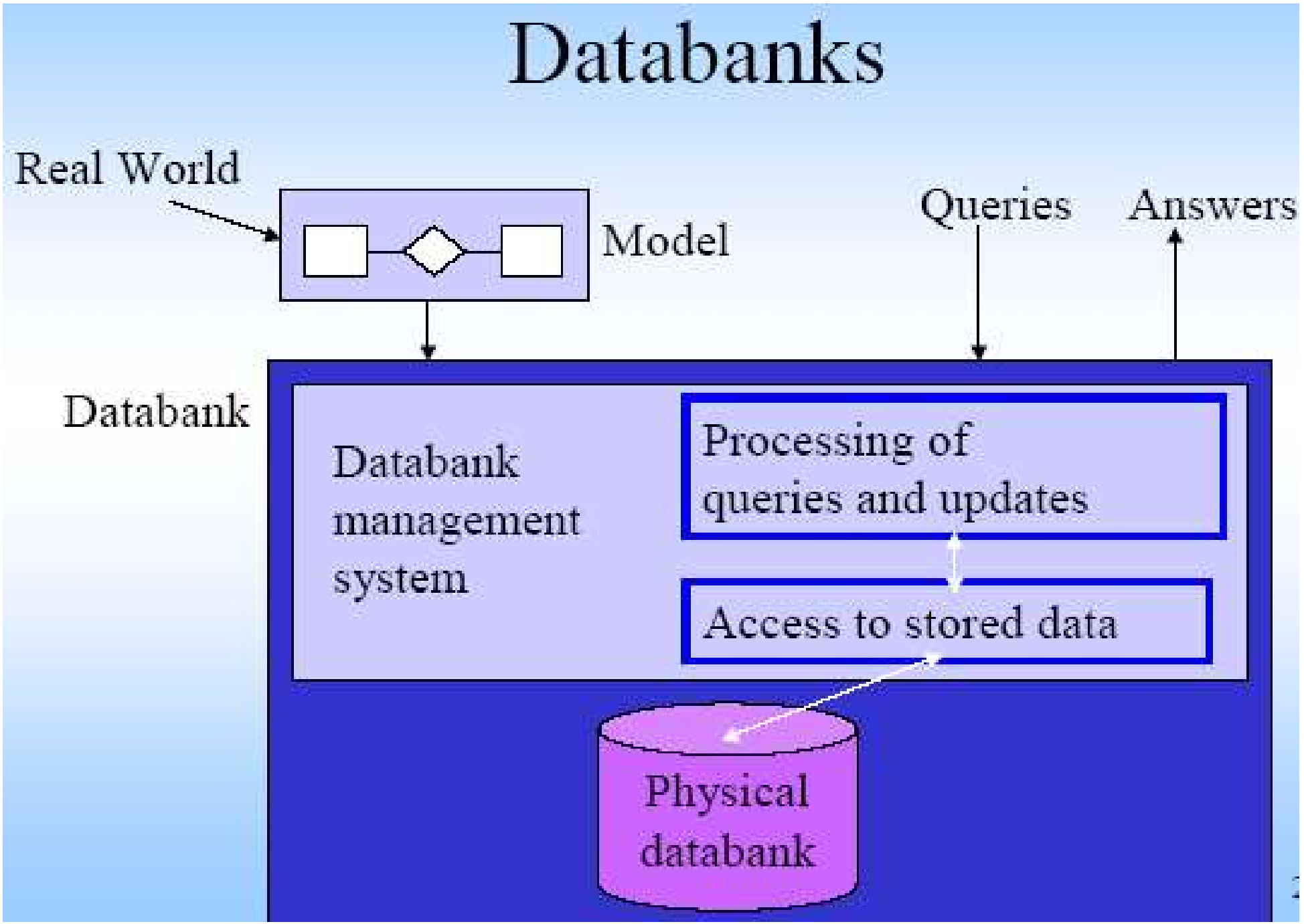
Dr. Sunnie S. Chung

CIS430/530

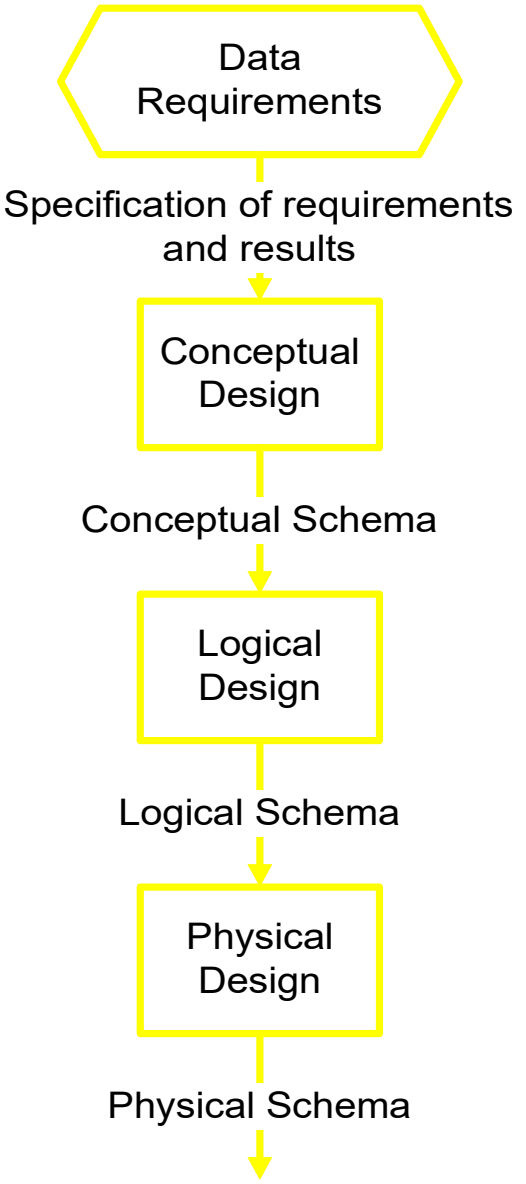
Learning Objectives

- ✓ Design E-R (Entity Relationship) Model
- ✓ How to Identify an Entity
- ✓ How to Identify Attribute Types
- ✓ How to Identity a Relationship between two Entities
- ✓ How to Identify Cardinality and Participation of Each Relationship
- ✓ Discuss the role of designing databases in the analysis and design of an information system
- ✓ Learn how to transform an entity-relationship (ER) Diagram into an Equivalent Set of Well-structured Relations

Databanks


Real World
Model
Queries
Answers
Databank
Databank management system
Processing of queries and updates
Access to stored data
Physical databank

Phases of Database Design



- Conceptual design begins with the collection of requirements and results needed for the database (ER Diagram)
- Logical schema is a description of the structure of the database (Relational Scheme)
- Physical schema is a description of the implementation (dictionaries, system catalogs, table files, indexes, programs

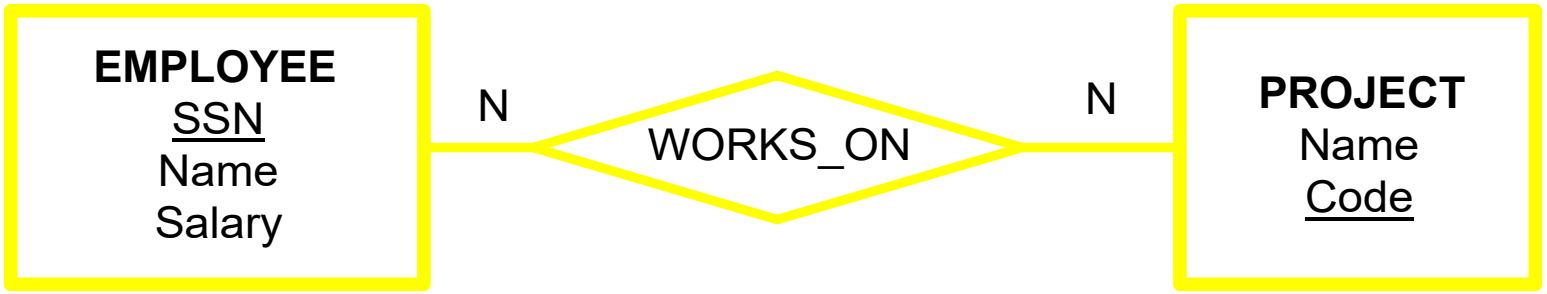
Models

A ***Data Model*** is a collection of objects in efficient Structures that can be used to represent a set of *data* and *operations* to Store and Manage the data for Business Operations

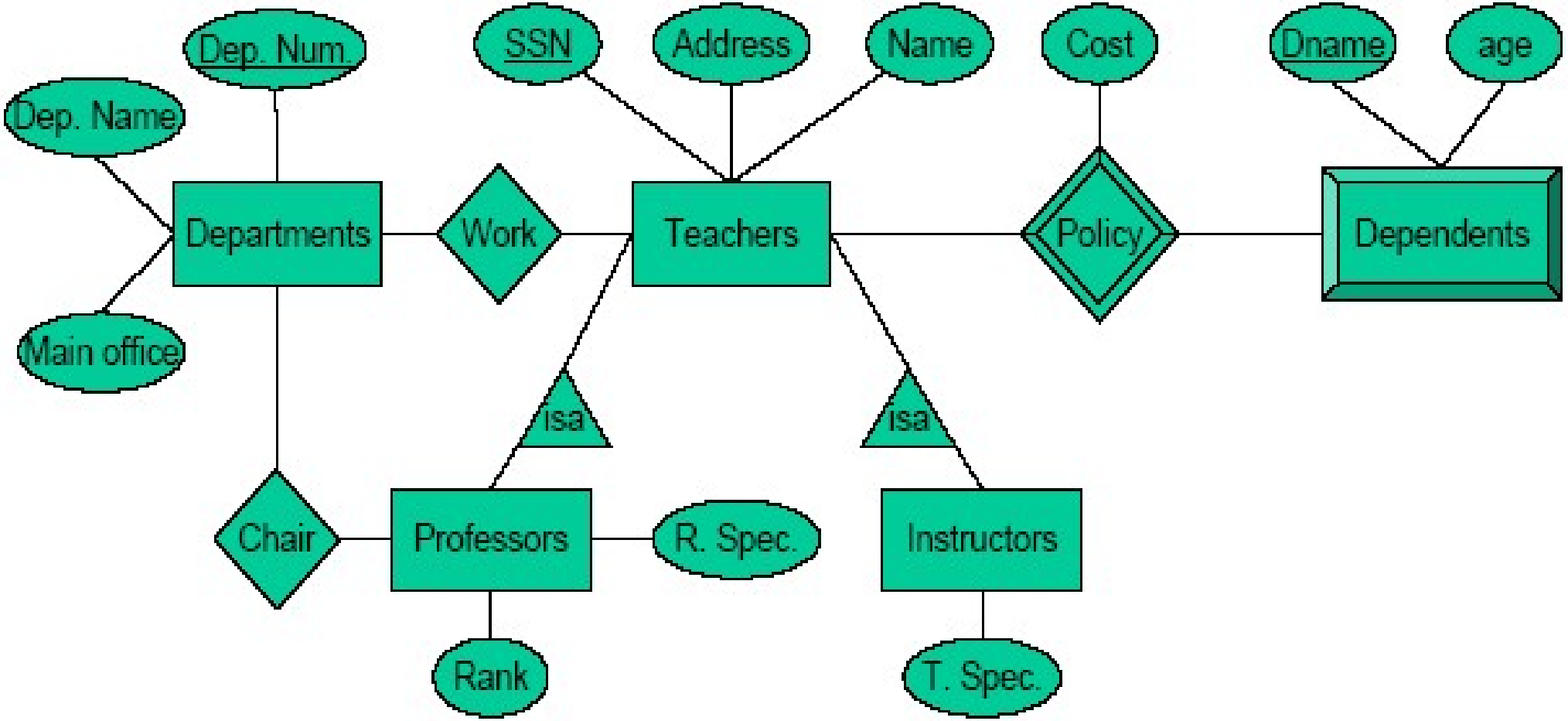
- ***Conceptual Models*** are tools for representing reality at a very high-level of abstraction **(ER Model in E-R Diagram)**
- ***Logical Models*** are data descriptions that can be processed by computers **(Database Scheme by DDL**

Example of E-R Model

- Every employee works in at least one project
- Every project has employees working on it

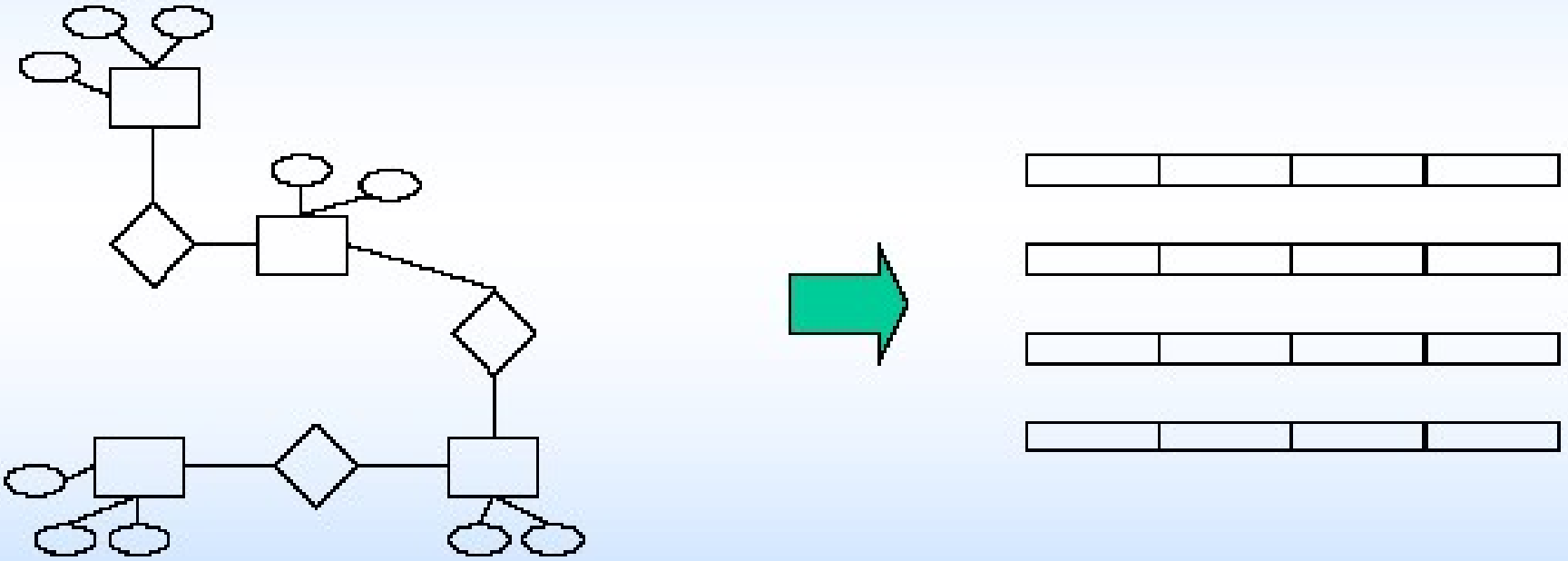


Example


Dep. Num.
SSN
Address
Name
Cost
Dname
age
Dep. Name
Departments
Work
Teachers
Policy
Dependents
Main office
isa
isa
Chair
Professors
R. Spec.
Instructors
Rank
T. Spec.

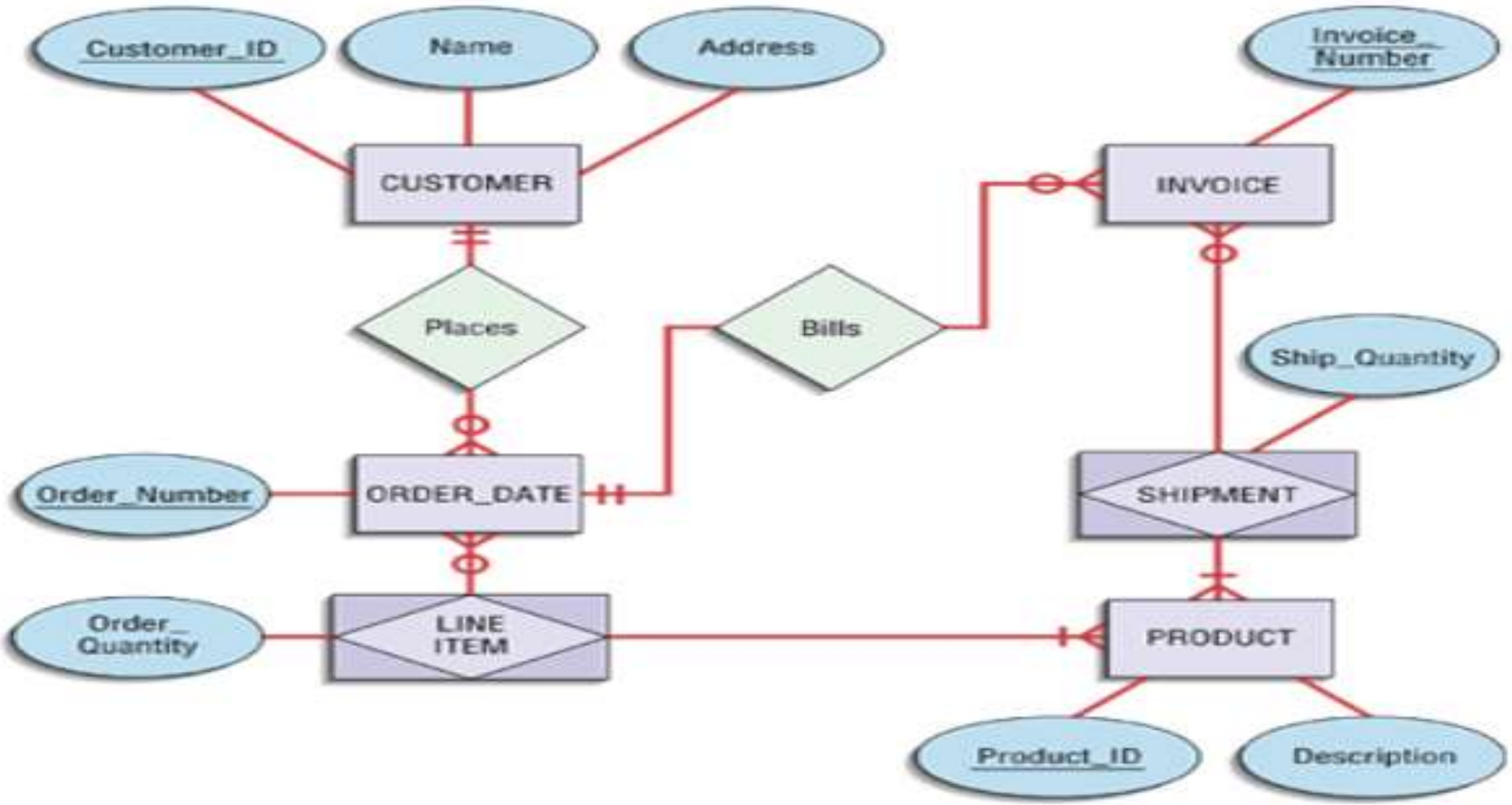
9.7

ER/EER to database schema



Entity Sets to Tables

- Each attribute of the E. S. becomes an attribute of the table



Relations:

CUSTOMER(Customer_ID,Name,Address)
PRODUCT(Product_ID,Description)
ORDER(Order_Number,Customer_ID,Order_Date)
LINE ITEM(Order_Number,Product_ID,Order_Quantity)
INVOICE(Invoice_Number,Order_Number)
SHIPMENT(Invoice_Number,Product_ID,Ship_Quantity)

9.9

Design of ER (Entity-Relationship) Model

- Identify All the Entities
- Identify a Common Set of Attributes for Each Entity
- Identify All the Relationships between Any Two Entities

ER Diagrams, Naming Conventions, and Design Issues

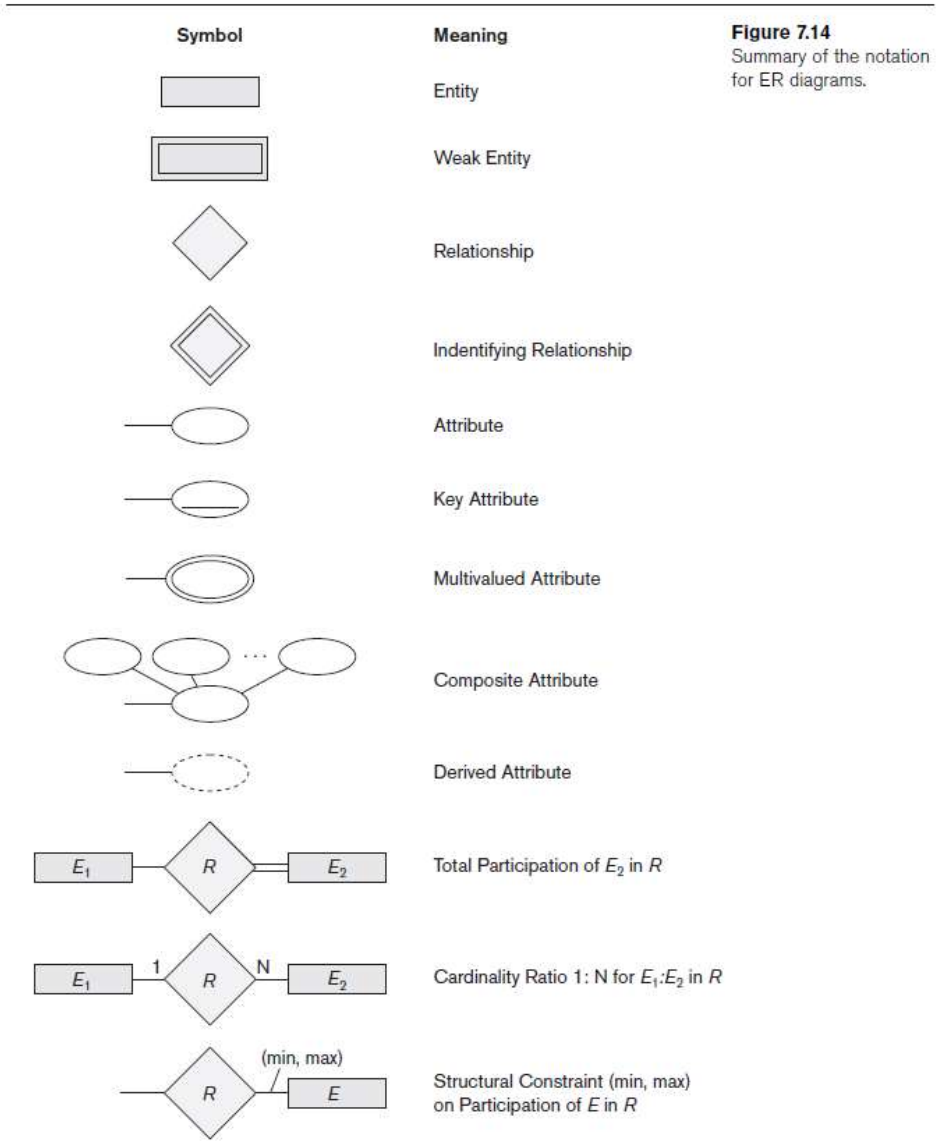
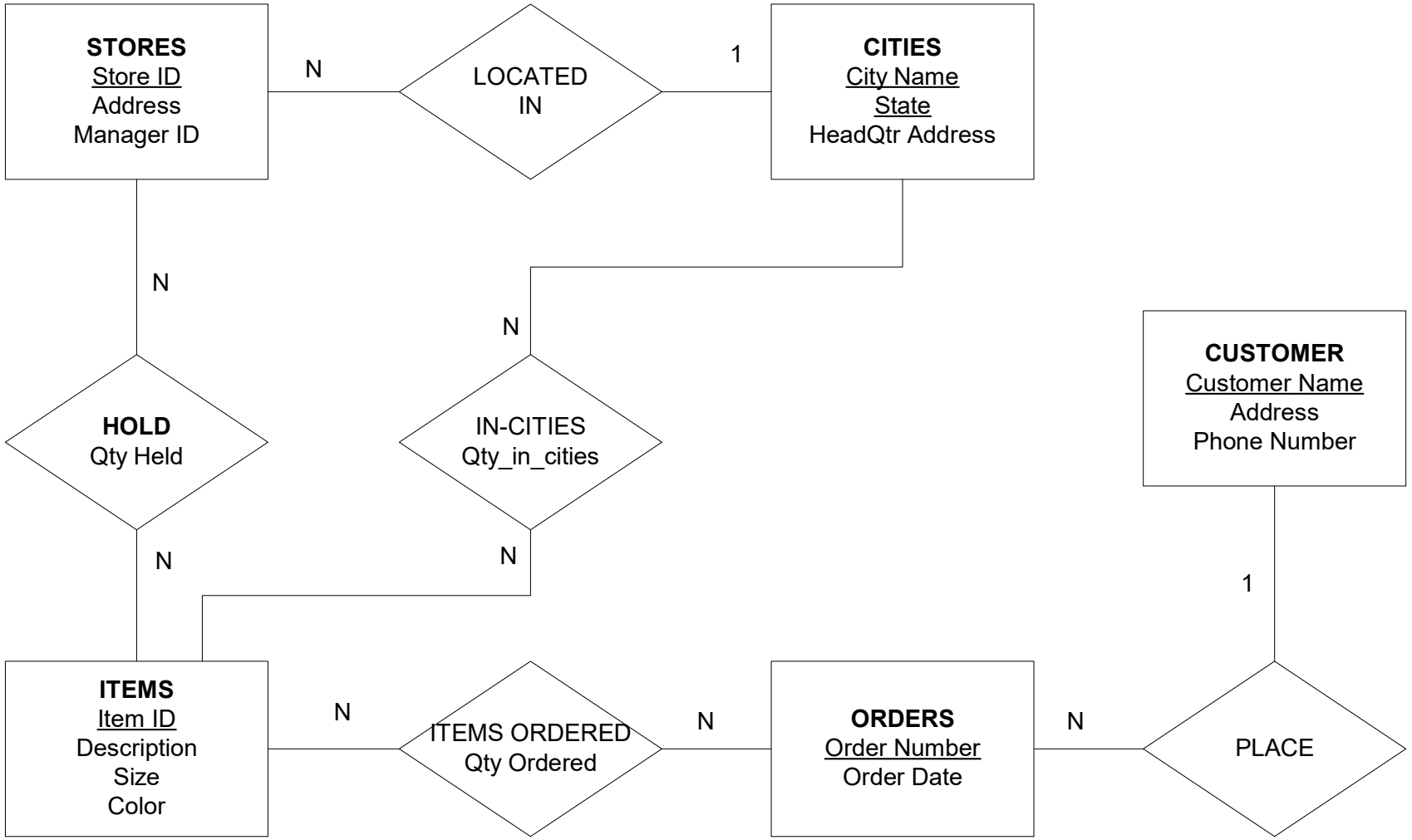


Figure 7.14
Summary of the notation for ER diagrams.

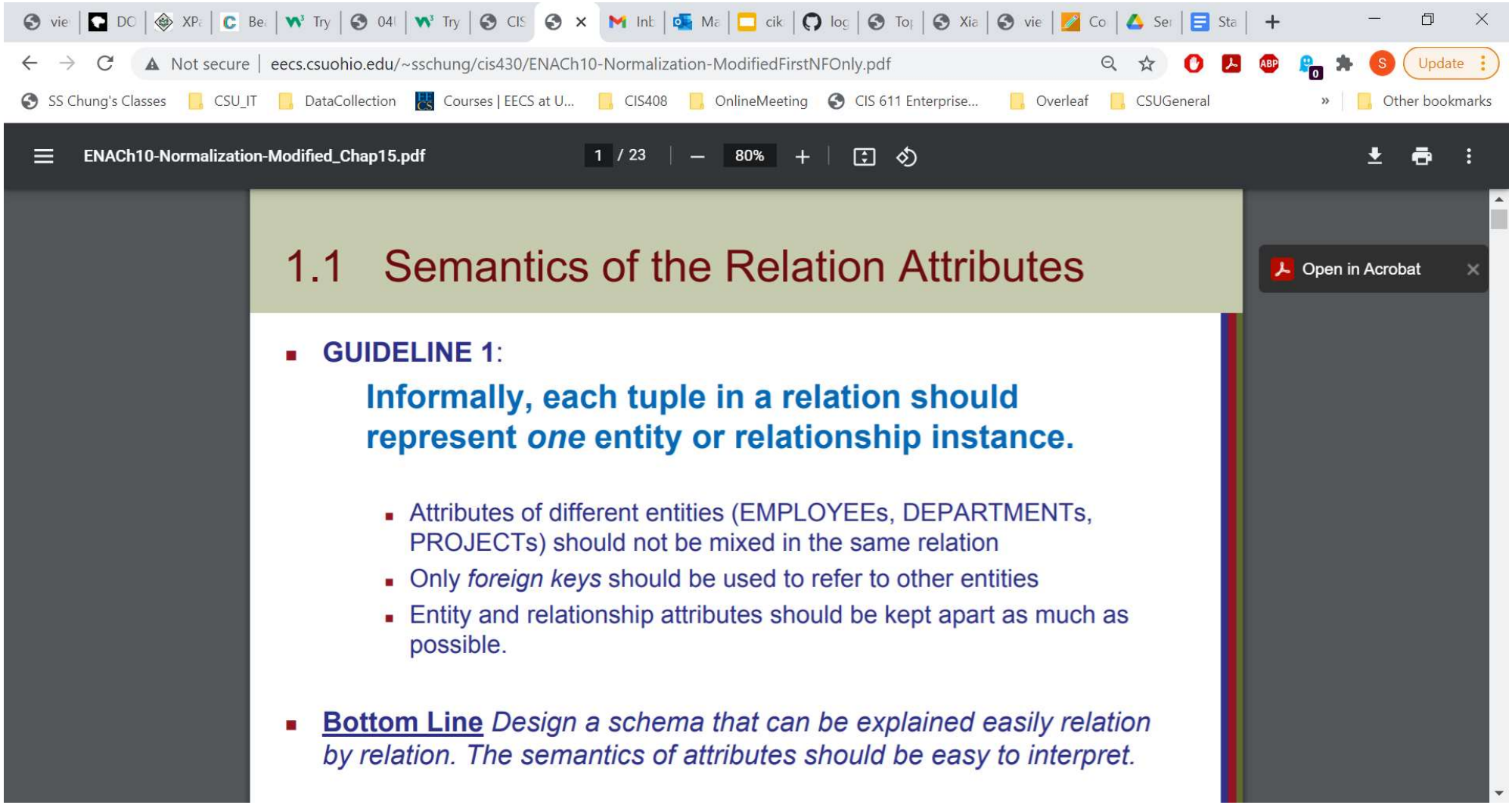
Example: Department Store 1/2

- A department store operates in several cities
- In a city there is one headquarter coordinating the local operations
- A city may have several stores
- Stores hold any amount of items
- Customers place their orders for any number of items to a given store

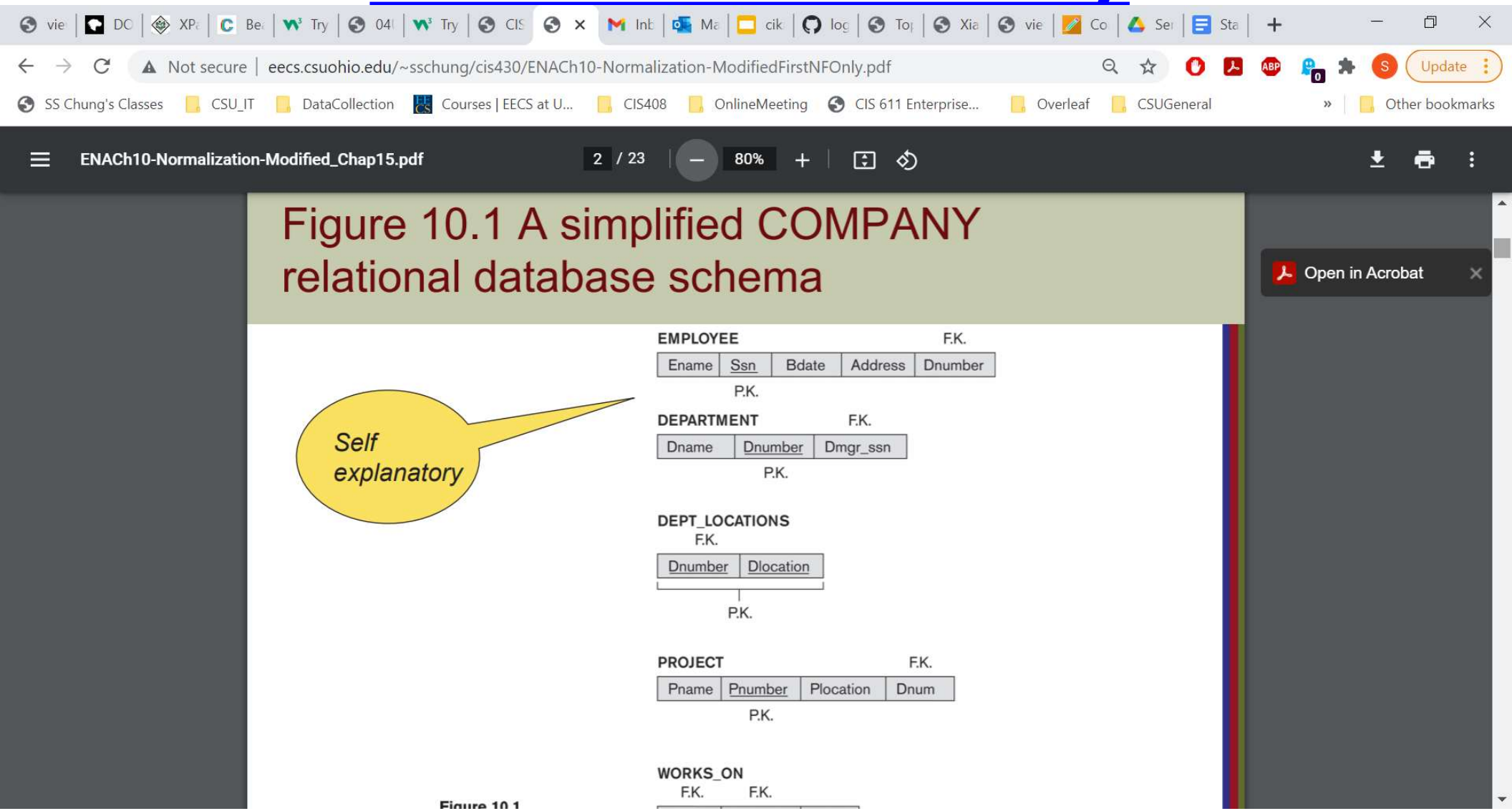
Example: Department Store
2/2



Guidelines for: Entity Identification



Guidelines for Entity



Guidelines for Entity

Not secure | eecs.csuohio.edu/~sschung/cis430/ENACh10-Normalization-ModifiedFirstNFOnly.pdf

SS Chung's Classes | CSU_IT | DataCollection | Courses | EECS at U... | CIS408 | OnlineMeeting | CIS 611 Enterprise... | Overleaf | CSUGeneral | Other bookmarks

ENACh10-Normalization-Modified_Chap15.pdf 3 / 23 75%

1.2 Redundant Information in Tuples and Update Anomalies

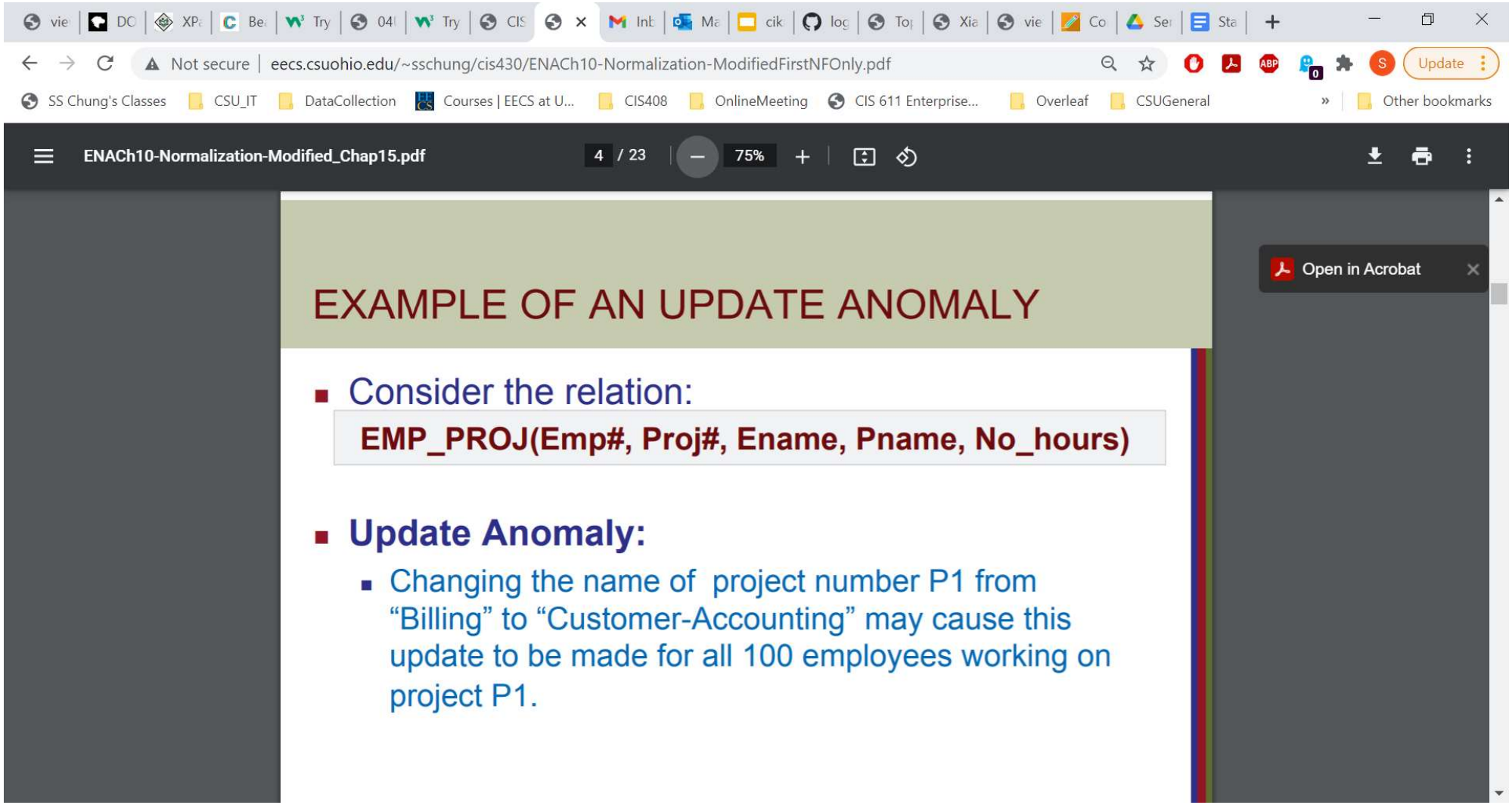
- **Big (and common) DB Problem:**
  In a 'poorly designed' DB information is stored redundantly

Consequences:

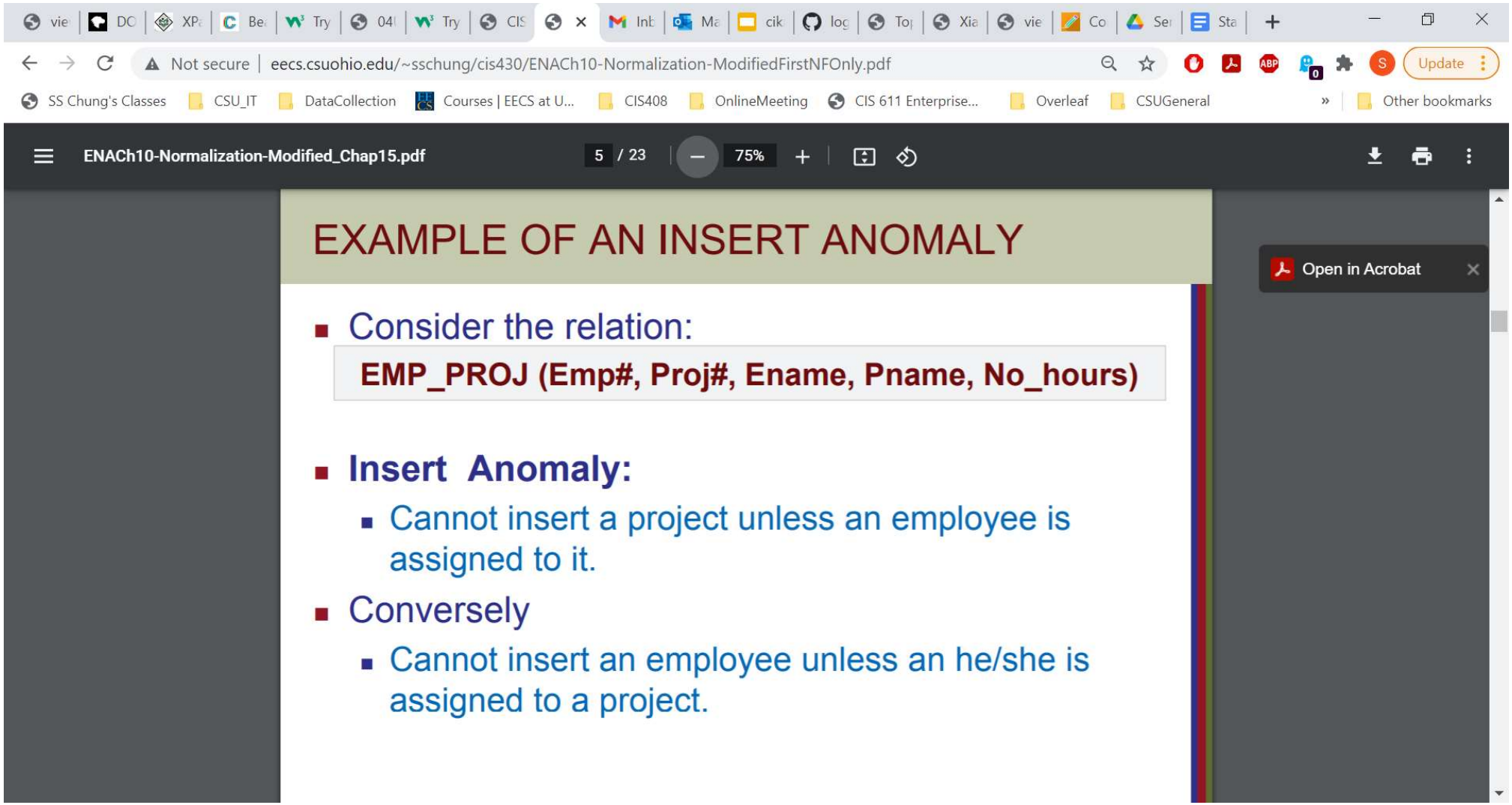
- Wastes storage
- Causes problems with update anomalies
  - Insertion anomalies
  - Deletion anomalies
  - Modification anomalies

Open in Acrobat

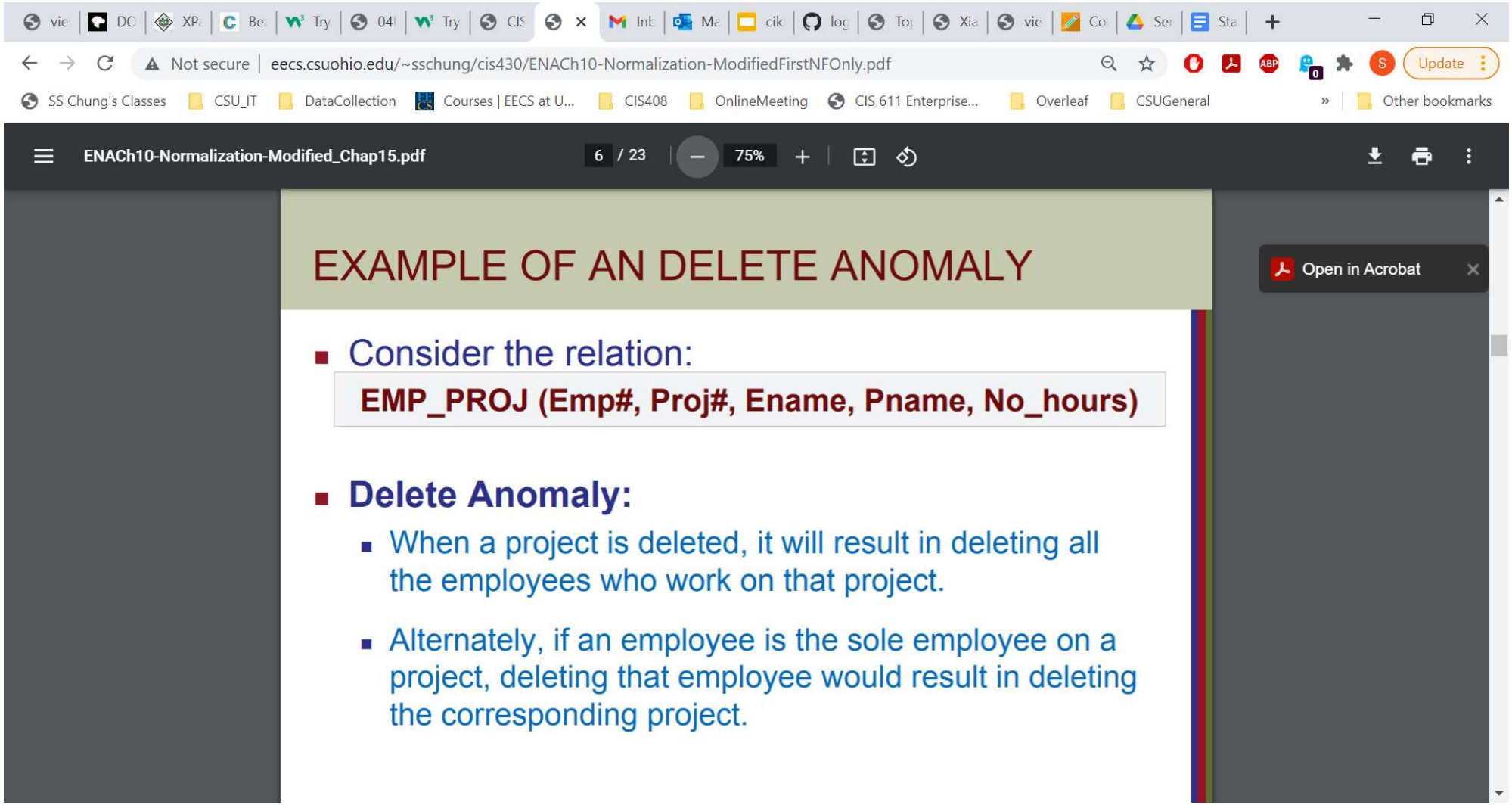
Example of Bad Entity



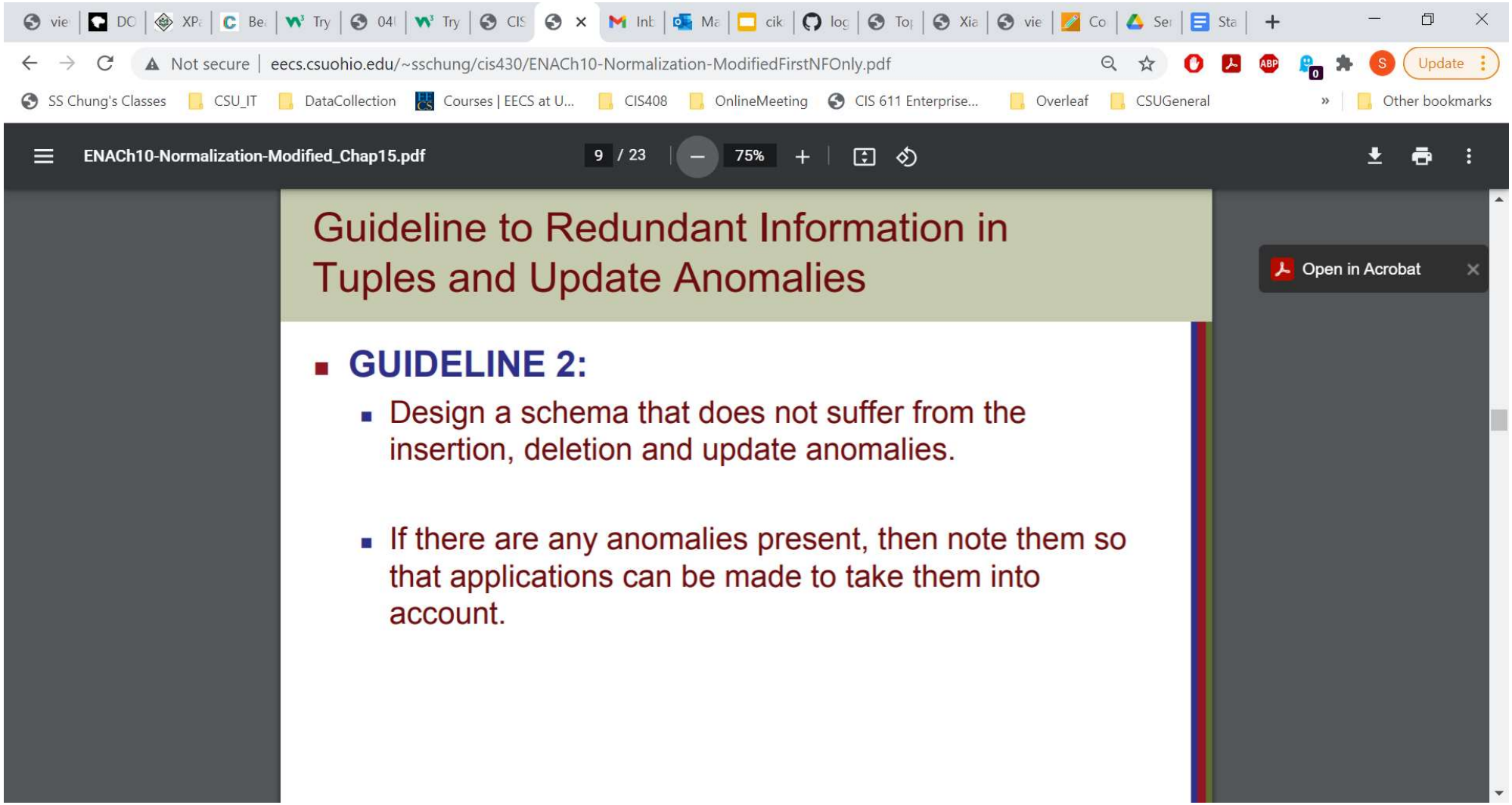
Example of Bad Entity



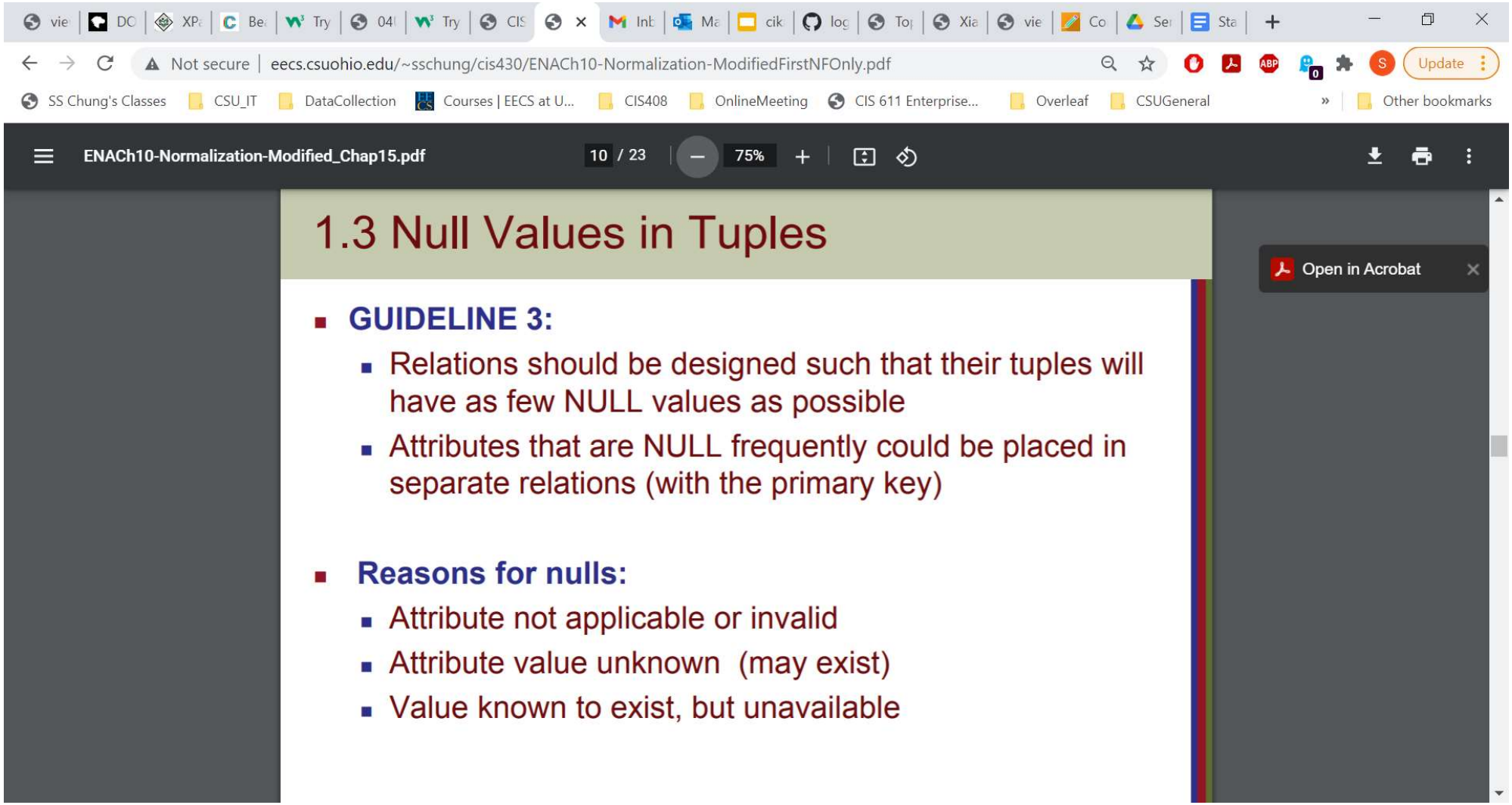
Example of Bad Entity



Guidelines for Entity



Guidelines for Entity



Attributes

- An entity is represented by a set of attributes, that is descriptive properties possessed by all members of an entity set.

Example:

Customer = (customer-id, customer-name, customer-street, customer-city)

Loan = (loan-number, amount)

- *Domain* – the set of permitted values for each attribute

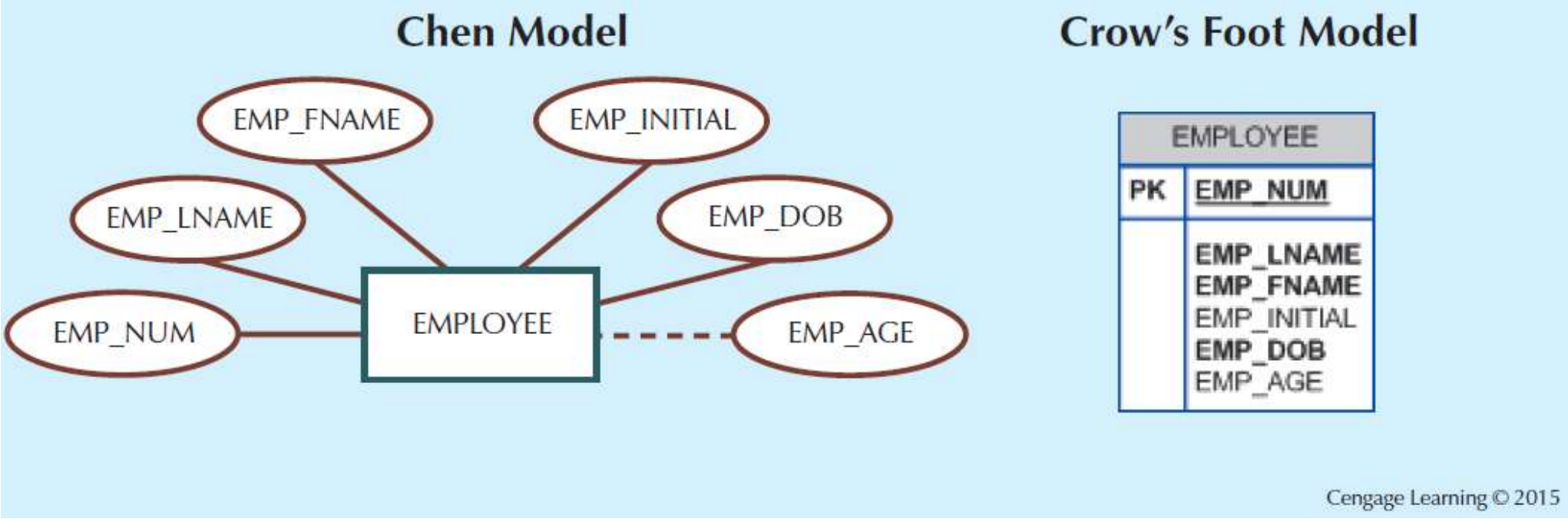
Attribute Types

- **Composite Key (identifier)**: Primary key composed of more than one attribute
- **Composite attribute**: Attribute that can be subdivided to yield additional attributes
- **Simple attribute**: Attribute that cannot be subdivided
- **Single-valued attribute**: Attribute that has only a single value (Simple attribute)
- **Multivalued attributes**: Attributes that have many values for a given object record (a tuple)

Attribute Types

- **Multivalued (Set Valued) attributes**: Attributes that have many values and require creating:
  - Several new attributes, one for each component of the original multivalued attribute
  - A new entity composed of the original multivalued attribute's components
- **Derived attribute**: Attribute whose value is calculated from other attributes
  - Derived using an algorithm

Two E-R Diagram Models with a Derived Attribute



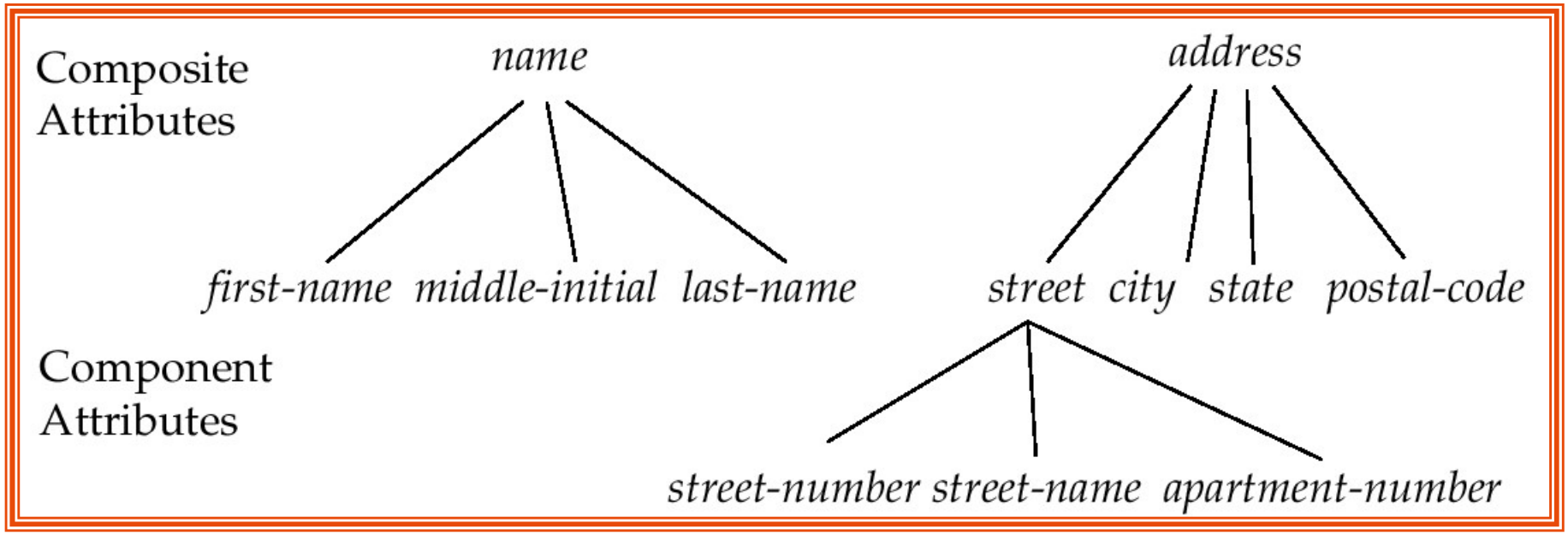
Attribute types:

- *Simple* and *Composite* attributes.

- *Single-valued* and *Multi-valued* attributes
  - E.g. multivalued attribute: *phone-numbers*

- *Derived* attributes
  - Can be computed from other attributes
  - E.g. *age*, given date of birth

Composite Attributes

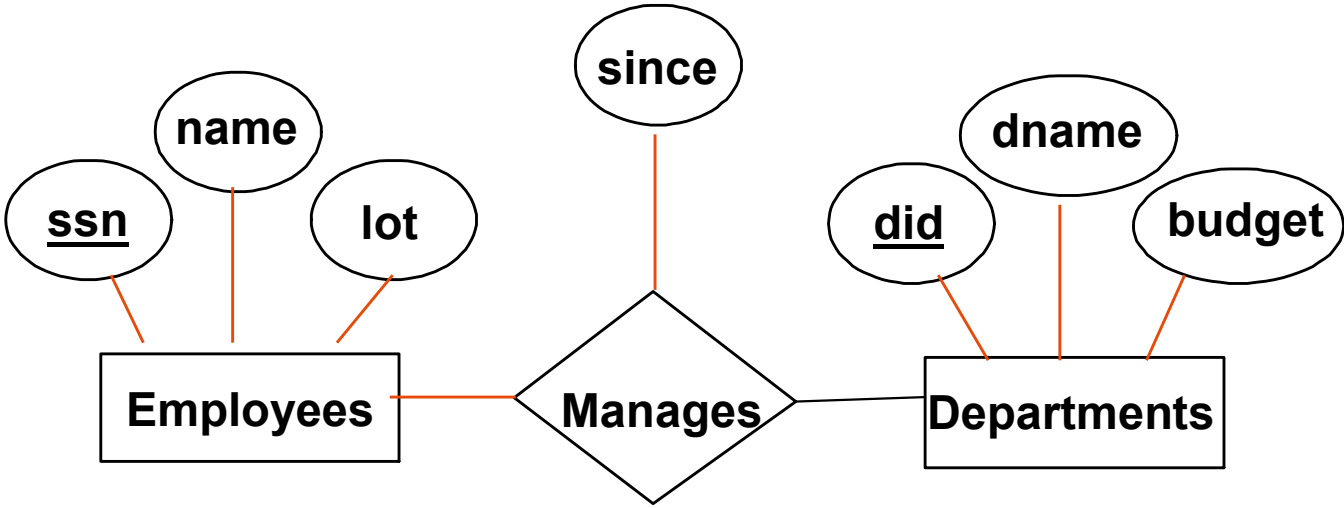


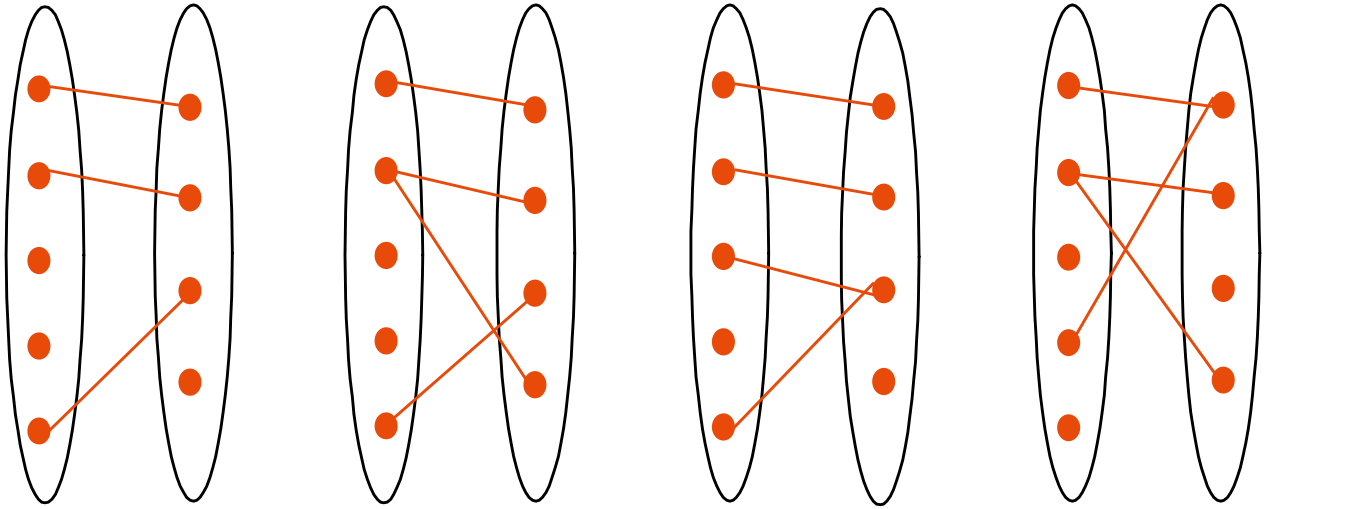
Relationship Types in ER Diagrams

- Specify structural constraints on Relationships
  - Replaces Cardinality ratio (1:1, 1:N, M:N) and single/double line notation for Participation constraints
  - Associate a pair of integer numbers (min, max) with each participation of an entity type *E* in a relationship type *R*, where $0 \leq \text{min} \leq \text{max}$ and $\text{max} \geq 1$

Cardinality Ratio (1:1, 1:N, M:N)

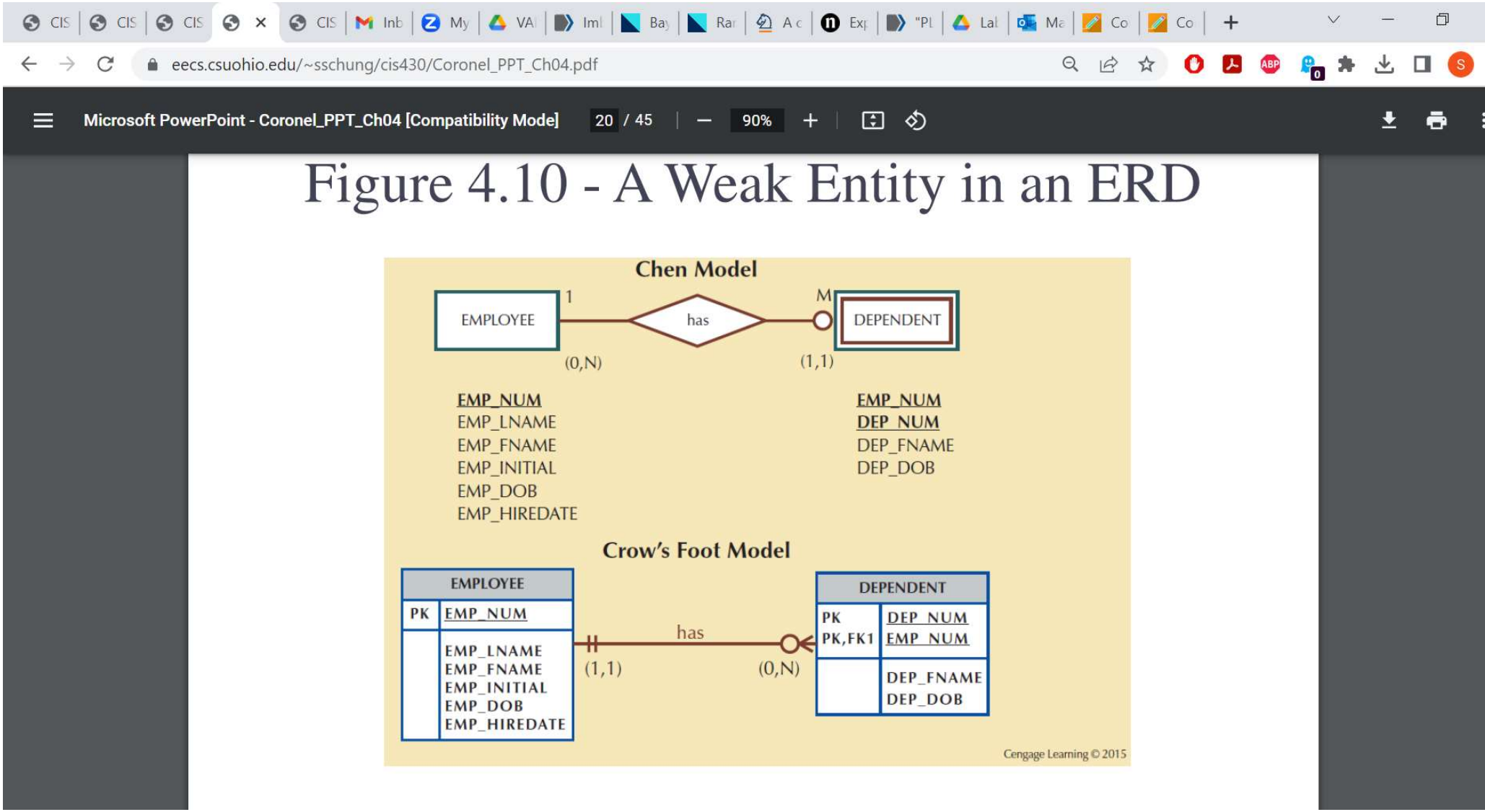
- 1:N :Each dept has at most one manager on Manages.





Translation to relational model?

Two Different E-R Diagram Models



Crow's Foot Symbols for Cardinality of Relationship

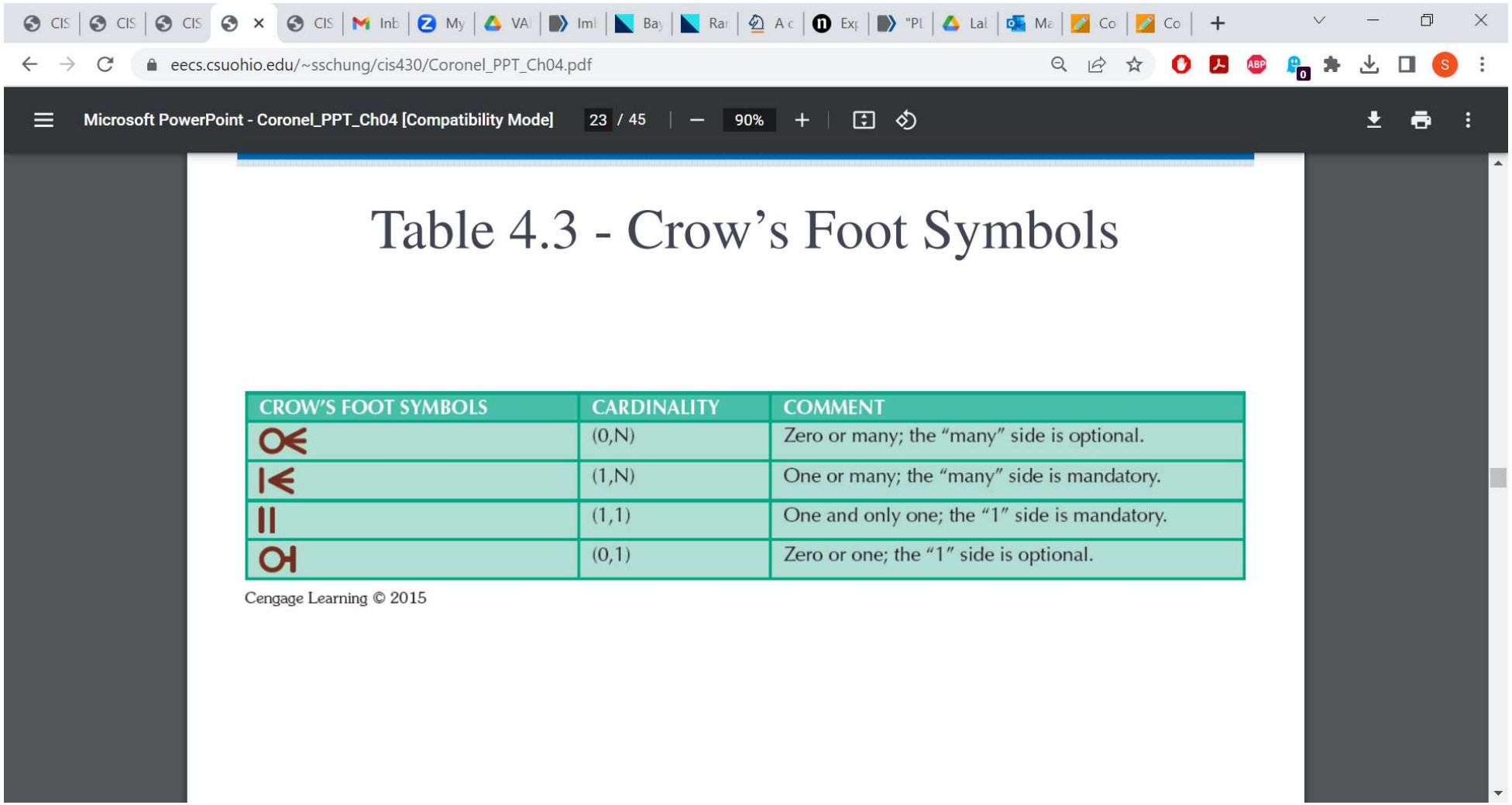


Table 4.3 - Crow's Foot Symbols

CROW'S FOOT SYMBOLS	CARDINALITY	COMMENT
	(0,N)	Zero or many; the "many" side is optional.
	(1,N)	One or many; the "many" side is mandatory.
	(1,1)	One and only one; the "1" side is mandatory.
	(0,1)	Zero or one; the "1" side is optional.

Example of Relationships

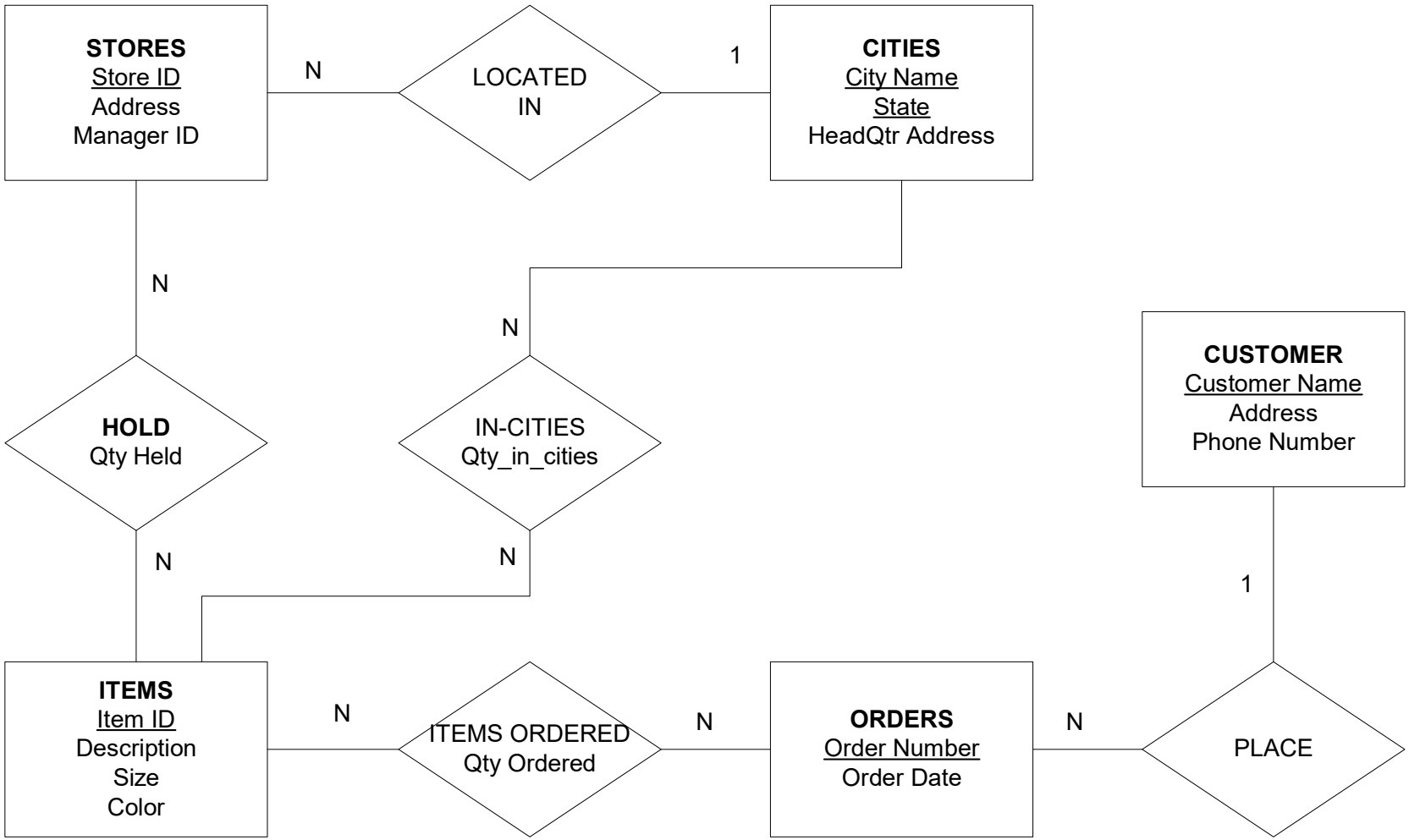
Example: Department Store 1/2

11

- A department store operates in several cities
- In a city there is one headquarter coordinating the local operations
- A city may have several stores
- Stores hold any amount of items
- Customers place their orders for any number of items to a given store

GOAL: Optimize shopping in each city

ER Model for Department Store



Steps to Identify Cardinality of Relationship

How to Identify 1-1, 1-N, or M-N for a Relationship between Two Entities

1. **Pick an object in the left Entity, check whether, for a given object in the left side Entity, it is allowed to have a relationship with one or more objects from the right-side Entity**
2. **Pick an object in the right-side Entity, check whether for a given object in right Entity, it is allowed to have a relationship with one or more objects from the left side of Entity**

If Both Step 1 and Step 2 is NO => 1-1

If Yes in Step 1 but No in Step2 => 1-N

Or

If No in Step 1 but Yes in Step2 => N-1

(Note that N-1 is equivalent to 1-N if you switch the left entity with the right entity)

If Yes in Both Step 1 And Step 2 => M-N

Refining the ER Design for the COMPANY Database

- Change attributes that represent relationships into relationship types
- Determine cardinality ratio and participation constraint of each relationship type

Design Choices for ER Conceptual Design

- Model concept first as an attribute
  - Refined into a relationship if attribute is a reference to another entity type
- Attribute that exists in several entity types may be elevated to an independent entity type
  - Can also be applied in the inverse

Relational Database Model

- Data represented as a set of related tables or relations
- Relation
  - A named, two-dimensional table of data. Each relation consists of a set of named columns and an arbitrary number of unnamed rows
  - Properties
    - Entries in cells are simple
    - Entries in columns are from the same set of values
    - Each row is unique
    - The sequence of columns can be interchanged without changing the meaning or use of the relation
    - The rows may be interchanged or stored in any sequence

Relational Database Model

- Well-Structured Relation
  - A relation that contains a minimum amount of redundancy and allows users to insert, modify and delete the rows without errors or inconsistencies

EMPLOYEE1

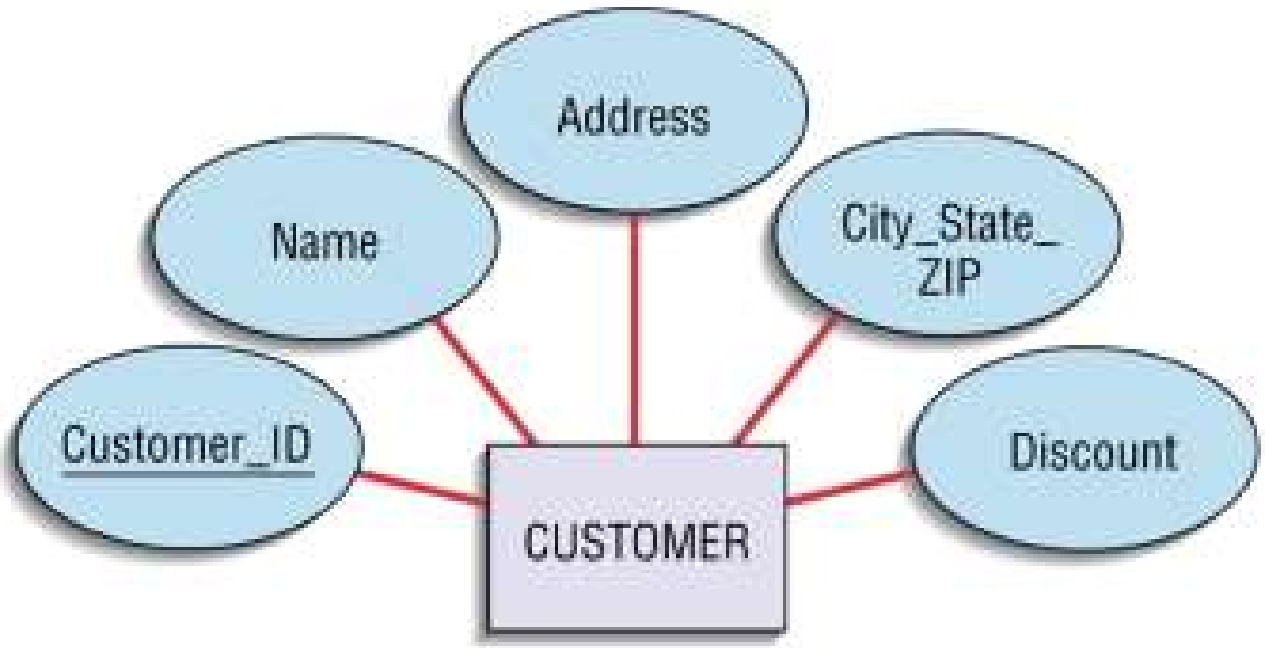
Emp_ID	Name	Dept	Salary
100	Margaret Simpson	Marketing	42,000
140	Allen Beeton	Accounting	39,000
110	Chris Lucero	Info Systems	41,500
190	Lorenzo Davis	Finance	38,000
150	Susan Martin	Marketing	38,500

Transforming E-R Diagrams into Relations

- In translating a relationship set to a relation, attributes of the relation must include:
  - The primary key for each participating entity set (as foreign keys).
    - This set of attributes forms a *superkey* for the relation.
  - All descriptive attributes of the relationship set

- The **primary key** must satisfy the following two conditions
  a. The value of the key must **uniquely identify** every row in the relation
  b. The key should be **nonredundant**

9.39



CUSTOMER

Customer_ID	Name	Address	City_State_ZIP	Discount
1273	Contemporary Designs	123 Oak St.	Austin, TX 28384	5%
6390	Casual Corner	18 Hoosier Dr.	Bloomington, IN 45821	3%

9.40

Transforming E-R Diagrams into Relations

Represent Relationships

- Binary 1:N Relationships
  - Add the **Primary key attribute** (or attributes) of the entity on the one side of the relationship as a **Foreign key** in the relation on the other (N) side
  - The one side *migrates* to the many side

Constraints on Binary Relationship Types

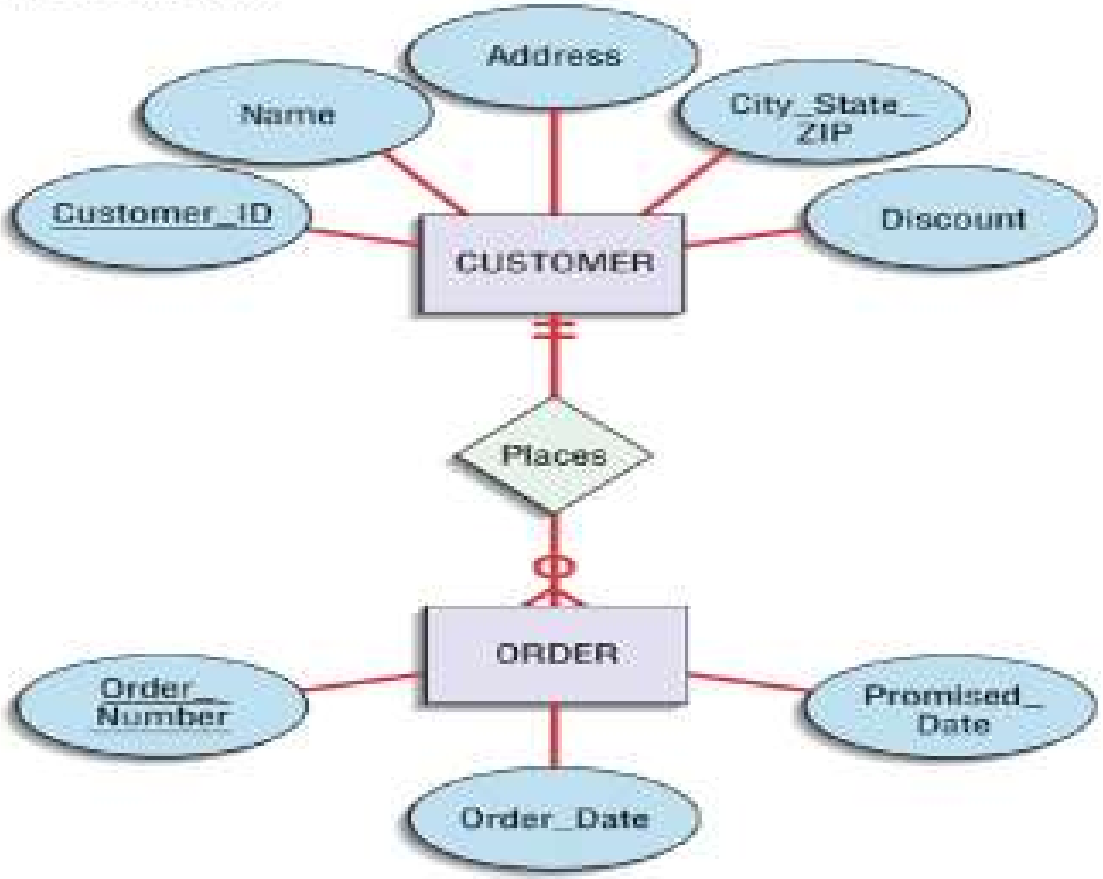
- **Cardinality ratio** for a binary relationship
  - Specifies maximum number of relationship instances that entity can participate in
- **Participation Constraint**
  - Specifies whether existence of entity depends on its being related to another entity
  - Types: **total** and **partial**

Attributes of Relationship Types

- Attributes of 1:1 or 1:N relationship types can be migrated to one entity type
- For a 1:N relationship type
  - Relationship attribute can be migrated only to entity type on N-side of relationship
- For M:N relationship types
  - Some attributes may be determined by combination of participating entities
  - Must be specified as relationship attributes

Weak Entity Types

- Do not have key attributes of their own
  - Identified by being related to specific entities from another entity type
- **Identifying relationship**
  - Relates a weak entity type to its owner
- Always has a total participation constraint



CUSTOMER

Customer_ID	Name	Address	City_State_ZIP	Discount
1273	Contemporary Designs	123 Oak St.	Austin, TX 28384	5%
6390	Casual Corner	18 Hoosier Dr.	Bloomington, IN 45821	3%

ORDER

Order_Number	Order_Date	Promised_Date	Customer_ID
57194	3/15/0X	3/28/0X	6390
63725	3/17/0X	4/01/0X	1273
80149	3/14/0X	3/24/0X	6390

Primary Key Constraints

- A set of fields is a *key* for a relation if :
  1. No two distinct tuples can have same values in all key fields, and
  2. This is not true for any subset of the key. – Key is minimal.
  - However, 2 does not hold (so false) for *superkey – which is not minimal.*
  - If there's more than one keys for a relation, one of the keys is chosen (by DBA) to be the *primary key*.
- E.g., customer_id is a key for Customer. (What about *name*?) The set {customer_id, *name*} could be a superkey.

Primary key can not have null value

Domain Constraint

- The value of each Attribute A with Domain Type $D(A_i)$ must be a atomic value from the domain type $D(A_i)$.

Definitions of Keys and Attributes Participating in Keys

- A **superkey** of a relation schema R = {A1, A2, ..., An} is a set of attributes S, *subset-of* R, with the property that **No two tuples t1 and t2 in any legal relation state r of R will have t1[S] = t2[S].**

  That is, for any given two tuples t1, t2 in data (extensions) of Relation schema R, t1[S] is not identical to t2[S].

- A **key** K is a superkey with the *additional property* that removal of any attribute from K will cause K not to be a superkey any more; **Key is minimal**.

Definitions of Keys and Attributes Participating in Keys

- If a relation schema has more than one key, each is called a **candidate** key.
- One of the candidate keys is *arbitrarily* designated to be the **primary key**, and the others are called **secondary keys**.
- A **Prime attribute** must be a member of any (candidate) key
- A **Nonprime attribute** is not a prime attribute—that is, it is not a member of any (candidate) key.

Foreign Keys, Referential Integrity

- **Foreign key** : Set of fields in one relation that is used to `refer' to a tuple in another relation. (Must correspond to primary key of the second relation.) Like a `logical pointer'.
- E.g. customer_id in Order is a foreign key referring to Customer:

Order (*order_number*, order_date, promised_date, customer_id)

Foreign Keys, Referential Integrity

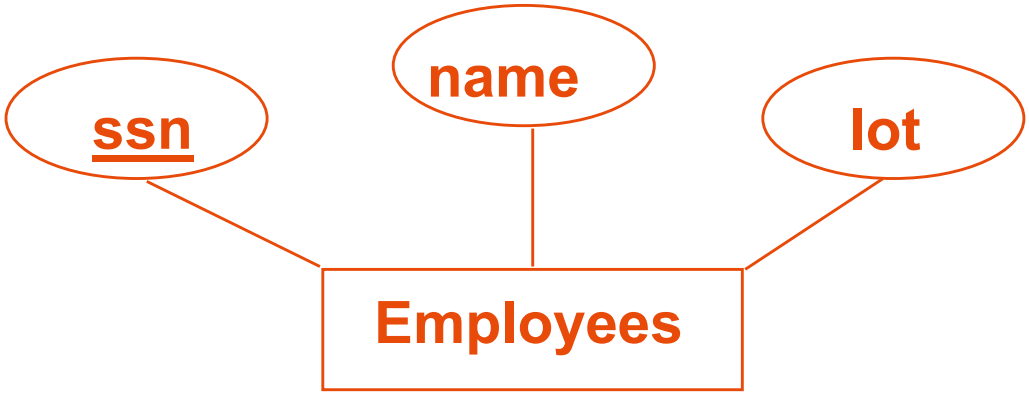
- If all foreign key constraints are enforced, **referential integrity** is achieved; all foreign key values should refer to existing values, i.e., no dangling references.
- Can you name a data model w/o referential integrity?
  - Links in HTML!

Enforcing Referential Integrity

- Consider Students(sid, name, gpa) and Enrolled (rid, semester, sid);
- *sid* in Enrolled is a foreign key that references Students.
- What should be done if an Enrolled tuple with a non-existent student id is **inserted**? **Reject it !**
- What should be done if a Students tuple is **deleted**?
  - Also delete all Enrolled tuples that refer to it.
  - Disallow deletion of a Students tuple that is referred to.
  - Set sid in Enrolled tuples that refer to it to a *default sid*.
  - (In SQL, also: Set sid in Enrolled tuples that refer to it to a special value *null*, denoting `*unknown*' or `*inapplicable*'.)
- Similar if primary key of Students tuple is **updated**.

Logical DB Design: ER to Relational

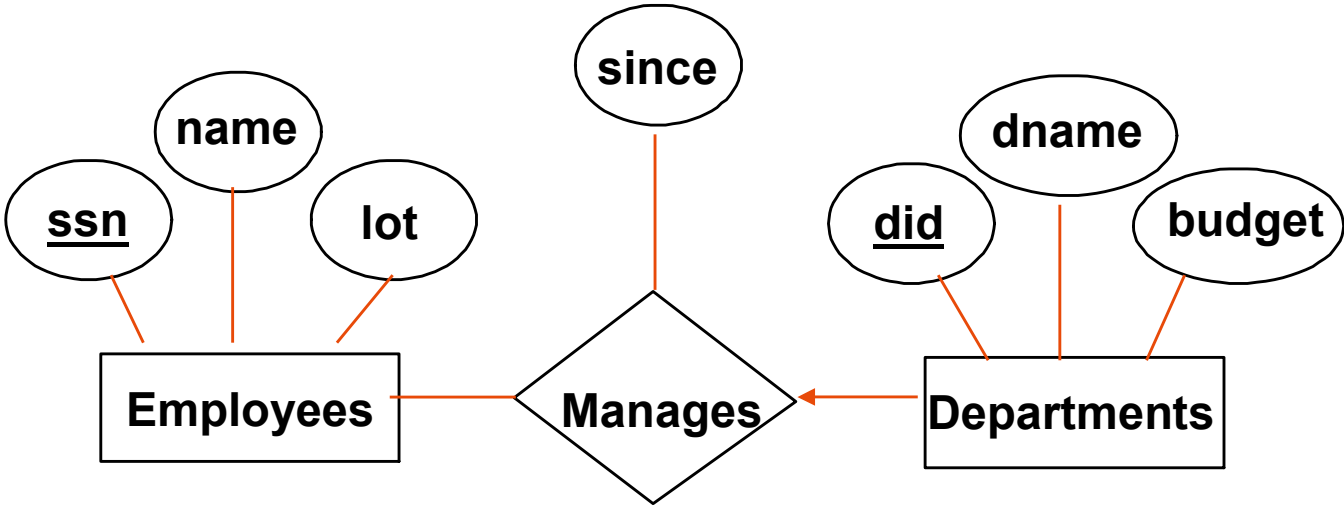
- Entity sets to tables.

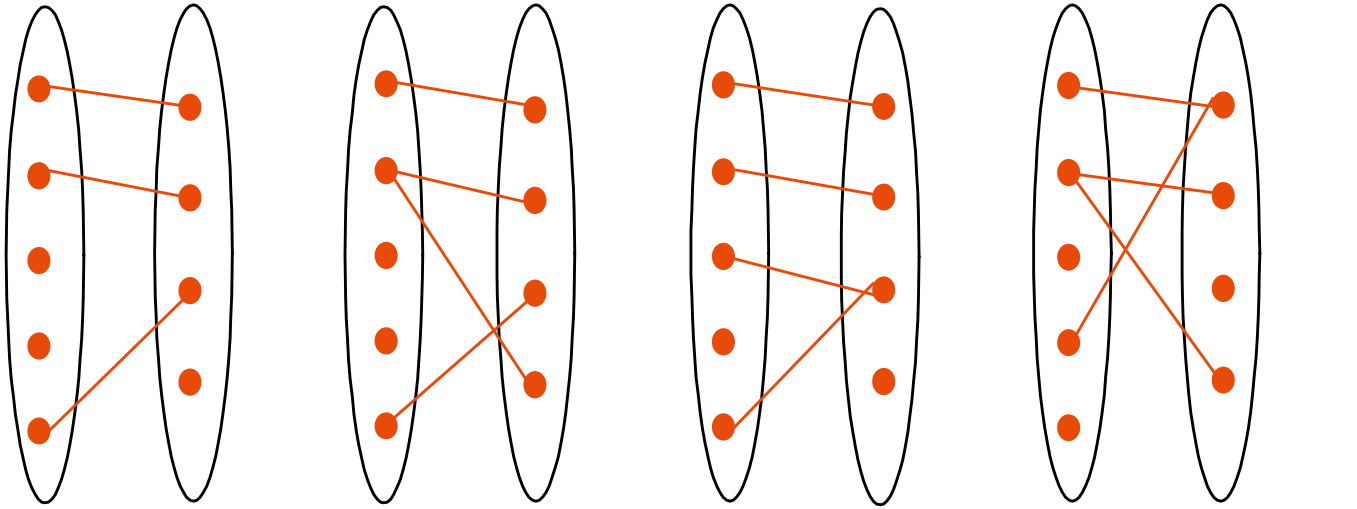



```
CREATE TABLE Employees
    (ssn CHAR(11),
    name CHAR(20),
    lot  INTEGER,
    PRIMARY KEY  (ssn))
```

Review: Key Constraints

- Each dept has at most one manager, according to the ***key constraint*** on Manages.



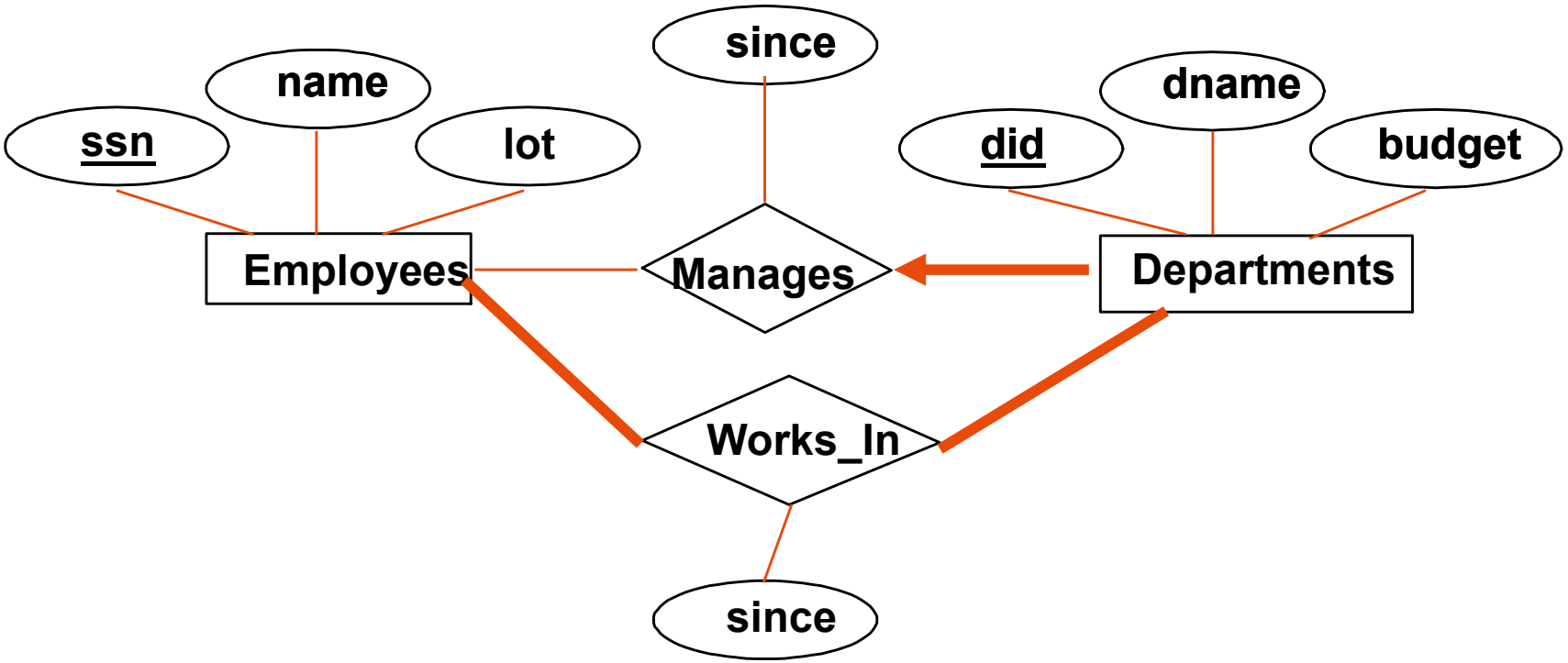


1-to-1 | 1-to Many | Many-to-1 | Many-to-Many

Translation to relational model?

Transforming 1:N, M:N Relationships with Key Constraints

ER Diagram:



Translating ER Diagrams with Key Constraints

- Map relationship to a table:
  - Note that did is the key here!
  - Separate tables for Employees and Departments.
- Since each department has a unique manager,

  we could instead

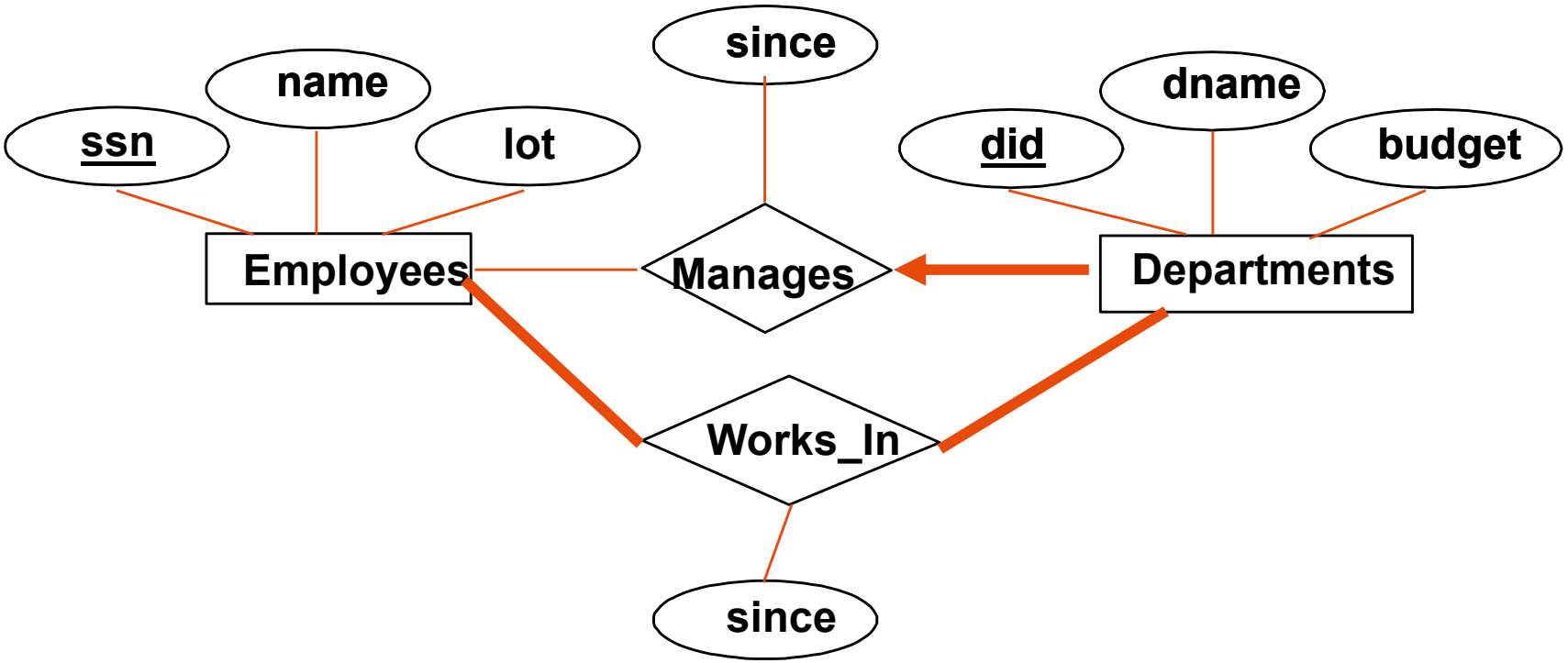
  combine Manages and Departments.

```
CREATE TABLE Manages(
  ssn CHAR(11),
  did INTEGER,
  since DATE,
  PRIMARY KEY (did),
  FOREIGN KEY (ssn) REFERENCES Employees,
  FOREIGN KEY (did) REFERENCES Departments)
```

```
CREATE TABLE Dept_Mgr(
  did INTEGER,
  dname CHAR(20),
  budget REAL,
  ssn CHAR(11),
  since DATE,
  PRIMARY KEY (did),
  FOREIGN KEY (ssn) REFERENCES Employees)
```

Transforming Realtionship to Tables

Example E-R diagram:



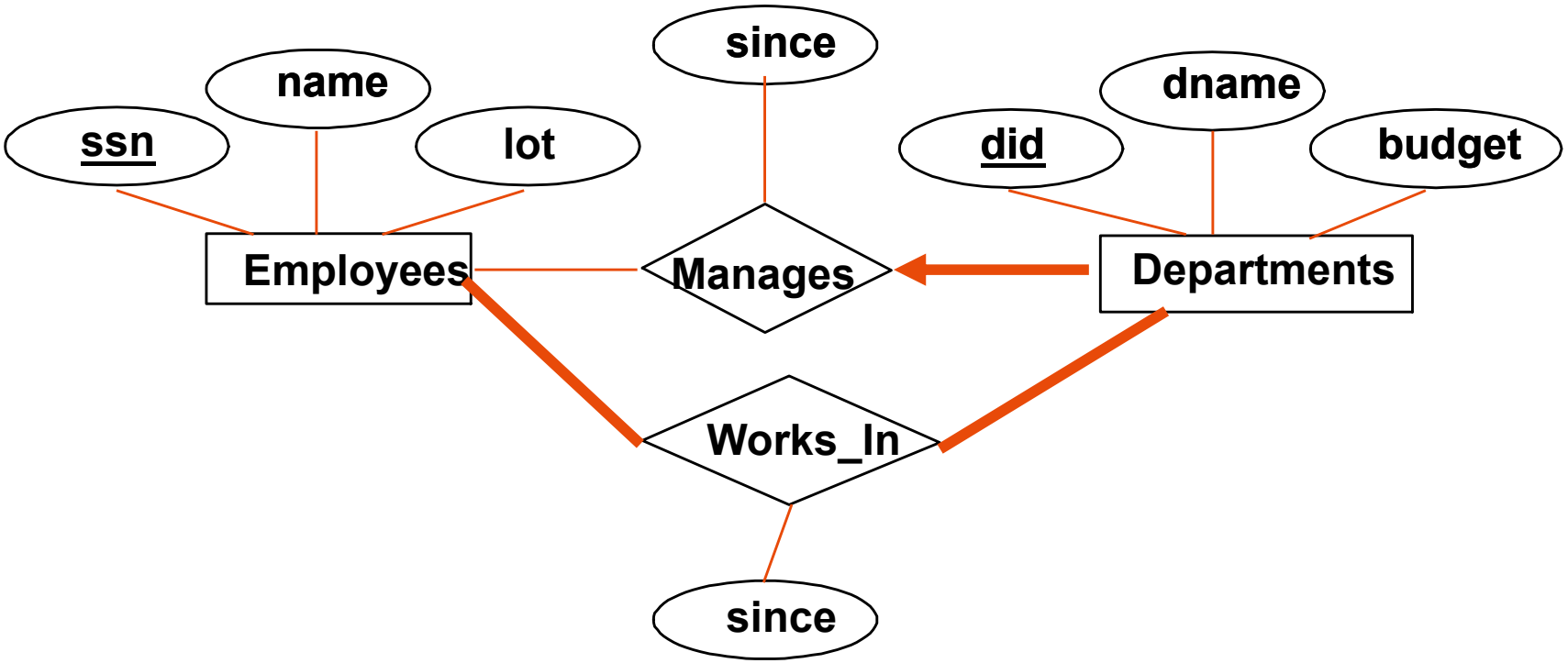
Relationship Sets to Tables

- In translating a relationship Works_In (M-N) to a relation, attributes of the relation must include:
  - Keys for each participating entity set (as foreign keys).
  - This set of attributes forms a ***superkey*** for the relation.
  - All descriptive attributes.

```
CREATE TABLE Works_In(
  ssn  CHAR(1),
  did  INTEGER,
  since  DATE,
  PRIMARY KEY (ssn, did),
  FOREIGN KEY (ssn)
        REFERENCES Employees,
  FOREIGN KEY (did)
        REFERENCES Departments)
```

Review: Participation Constraints

- Does every department have a manager?
  - If so, this is a *participation constraint*: the participation of Departments in Manages is said to be *total* (vs. *partial*).
    - Every *did* value in Departments table must appear in a row of the Manages table (with a non-null *ssn* value!)



Participation Constraints in SQL

- We can capture participation constraints involving one entity set in a binary relationship, but little else (without resorting to CHECK constraints).

```
CREATE TABLE  Dept_Mgr(
   did  INTEGER,
   dname  CHAR(20),
   budget  REAL,
   ssn  CHAR(11) NOT NULL,
   since  DATE,
   PRIMARY KEY  (did),
   FOREIGN KEY  (ssn) REFERENCES Employees,
     ON DELETE NO ACTION)
```

An Example

```
CREATE TABLE Student (
        ID          NUMBER,
        Fname       VARCHAR2(20),
        Lname       VARCHAR2(20),
);
```

Constraints in Create Table

- Adding constraints to a table enables the database system to enforce data integrity.
- Different types of constraints:
  - * Not Null
  - * Unique
  - * Foreign Key
  - * Default Values
  - * Primary Key
  - * Check Condition

Not Null Constraint

```
CREATE TABLE Student (
        ID          NUMBER,
        Fname       VARCHAR2(20) NOT NULL,
        Lname       VARCHAR2(20) NOT NULL,
);
```

Primary Key Constraint

```
CREATE TABLE Student (
        ID          NUMBER    PRIMARY KEY,
        Fname       VARCHAR2(20) NOT NULL,
        Lname       VARCHAR2(20) NOT NULL,
);
```

- Primary Key implies: * NOT NULL * UNIQUE.
- There can only be one primary key.

Primary Key Constraint (Syntax 2)

```
CREATE TABLE Students (
        ID          NUMBER,
        Fname       VARCHAR2(20) NOT NULL,
        Lname       VARCHAR2(20) NOT NULL,
        PRIMARY KEY(ID)
);
```

Needed when the primary key is made up of two or more attributes (fields)

Foreign Key Constraint

```
CREATE TABLE Studies(
        Course          NUMBER,
        Student         NUMBER,
        FOREIGN KEY (Student) REFERENCES
                Students(ID)
);
```

NOTE: ID must be unique (or primary key) in Students table