5. Schema Design and Entity-Relationship Model

CSCI 2541 Database Systems & Team Projects

Wood & Chaufournier

Slides adapted from Prof. Bhagi Narahari; and Silberschatz, Korth, and Sudarshan

Announcements?

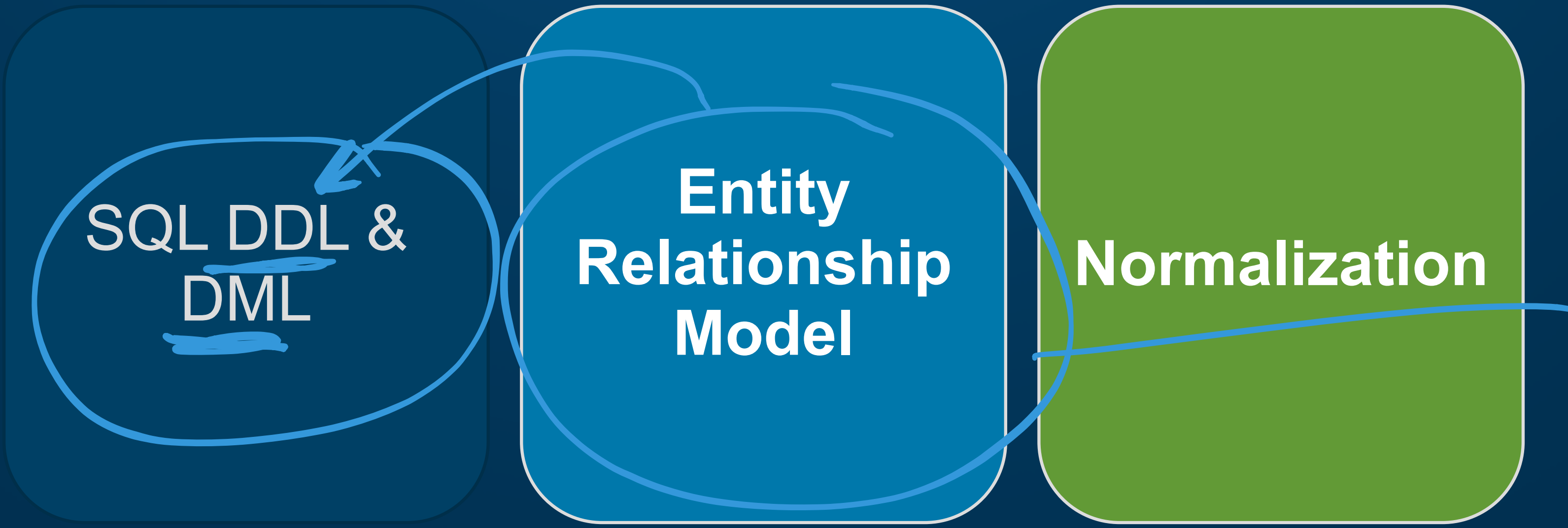
Due Dates — Check Web

SQL Lab - Select → Tomorrow

Relational Alg → 2/15

#Engage

Last time…



SQL DDL & DML

Entity Relationship Model

Normalization

this time…

Design Phases

Initial phase: fully characterize the data needs of the prospective database users

Second phase: choose a data model

- A data model provides a standard way to think about information and how it is related
- Must translate these requirements into a **conceptual schema** of the database
- A fully developed conceptual schema indicates the functional requirements of the enterprise
  - Describes the key pieces of information that must be tracked
  - Describes the kinds of operations (or transactions) that will be performed on the data

Design Phases

Which is harder to fix later?

Final Phase: Moving from an abstract data model to the implementation of the database

- **1. Logical Design** – Deciding on a "good" collection of relation schemas
  - **Business decision** – What attributes should we record in the database?
  - **Computer Science decision** – What relation schemas should we have and how should the attributes be distributed among the various relation schemas?
- **2. Physical Design** – Deciding on the physical layout of the database
  - The DBMS will do some of this for us
  - But we can control things like how indexes are generated to optimize frequent data lookups (later)

App
↓
logic
↓
Phys

Design Alternatives

In designing a database schema, we must ensure that we avoid two major pitfalls:

- **Redundancy**: a bad design may result in repeated information
  - Redundant representation of information may lead to data inconsistency among the various copies of information
- **Incompleteness:** a bad design may make certain aspects of the enterprise difficult or impossible to model

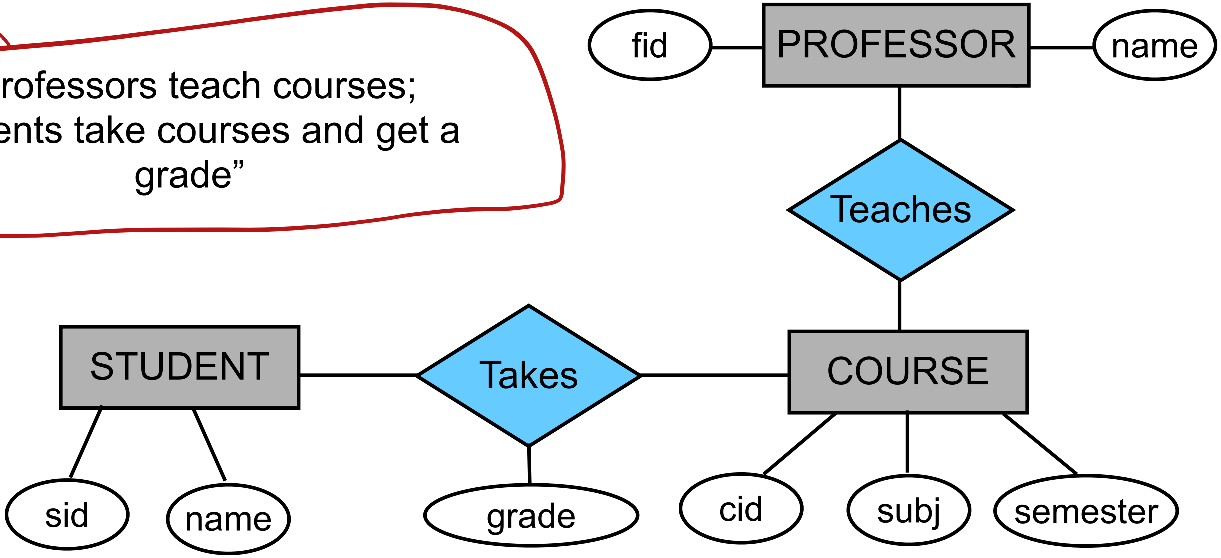
Avoiding bad designs is not enough. There may be a large number of good designs from which we must choose

Entity-Relationship Model

Data model that lets you visualize a conceptual schema based on three simple concepts:

- Entities, Relationships, and Attributes

"Professors teach courses; students take courses and get a grade"



One picture provides info on what your system stores and models

ER Model - Entities

Entity: Real-world object distinguishable from other objects

- An entity is described (in DB) using a set of attributes
- Analogy: Nouns (Student, Course,…)

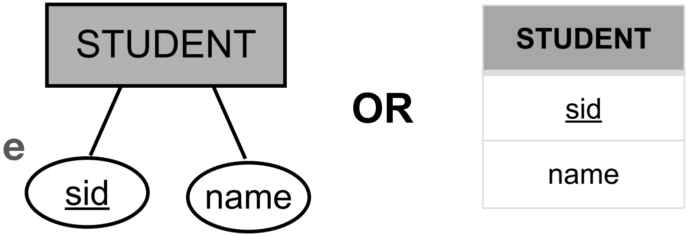
Entity Set: A collection of similar entities. e.g., all employees.

- All entities in an entity set have the same set of attributes. (Until we consider ISA hierarchies, anyway!)
- Each entity set has a **key**
- Each attribute has a **domain**

An **entity instance** is a particular example or occurrence of an entity type…eg: Faculty Tim Wood

Representation/Syntax:

- **Entity** set represented by **Rectangle**
- **Attribute** represented by **Oval or a Table**
  - Unique (Key) attributes underlined



ER Model - Relationships

Relationship: Association among two or more entities. E.g., Dan takes Database Course; Maya works in Research department.

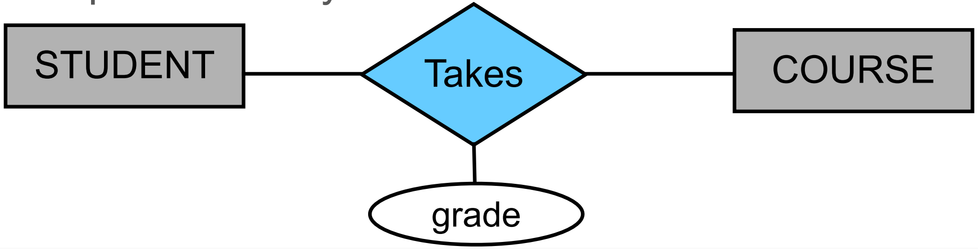
- Relationship can also have attributes (that appear only for this relationship set)
- Analogy: Verb (Takes, Belongs to, Works_On,….)

Relationship Set: Collection of relationships

- An n-ary relationship set R relates n entity sets E1 ... En; each relationship in R involves entities e1 ∊ E1, ..., en ∊ En

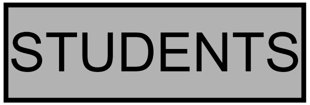
Representation/Syntax: a **Diamond** symbol

- Attributes represented by Oval

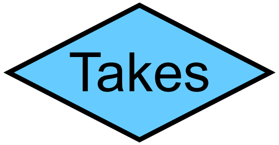


Conceptual Design Process

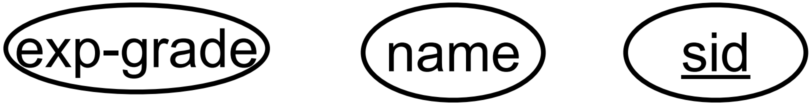
What are the entities being represented?



What are the relationships?



What info (attributes) do we store about each?



What keys & integrity constraints do we have?

Pet Example

A veterinary clinic wants to track information about its customers (human and animal). Pet owners have a name and account ID. Pets have a name, age, and weight. Whenever a pet comes for an appointment we must record a date, symptoms, and diagnosis.

Entities

Customer
Animal
Employee
Appointment

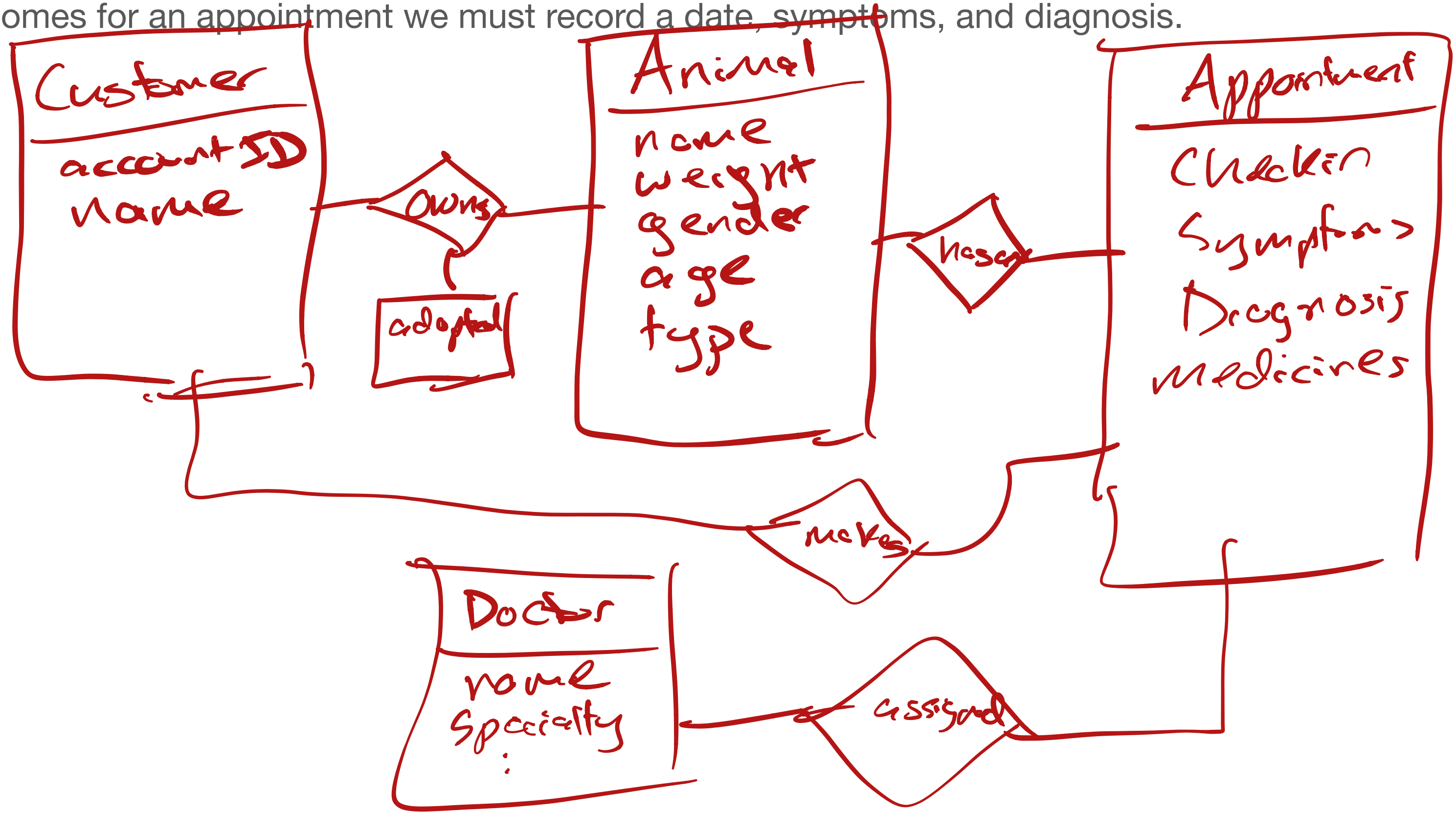
Relationships

has_a
owns
schedules

How would we draw this with ER Diagrams?

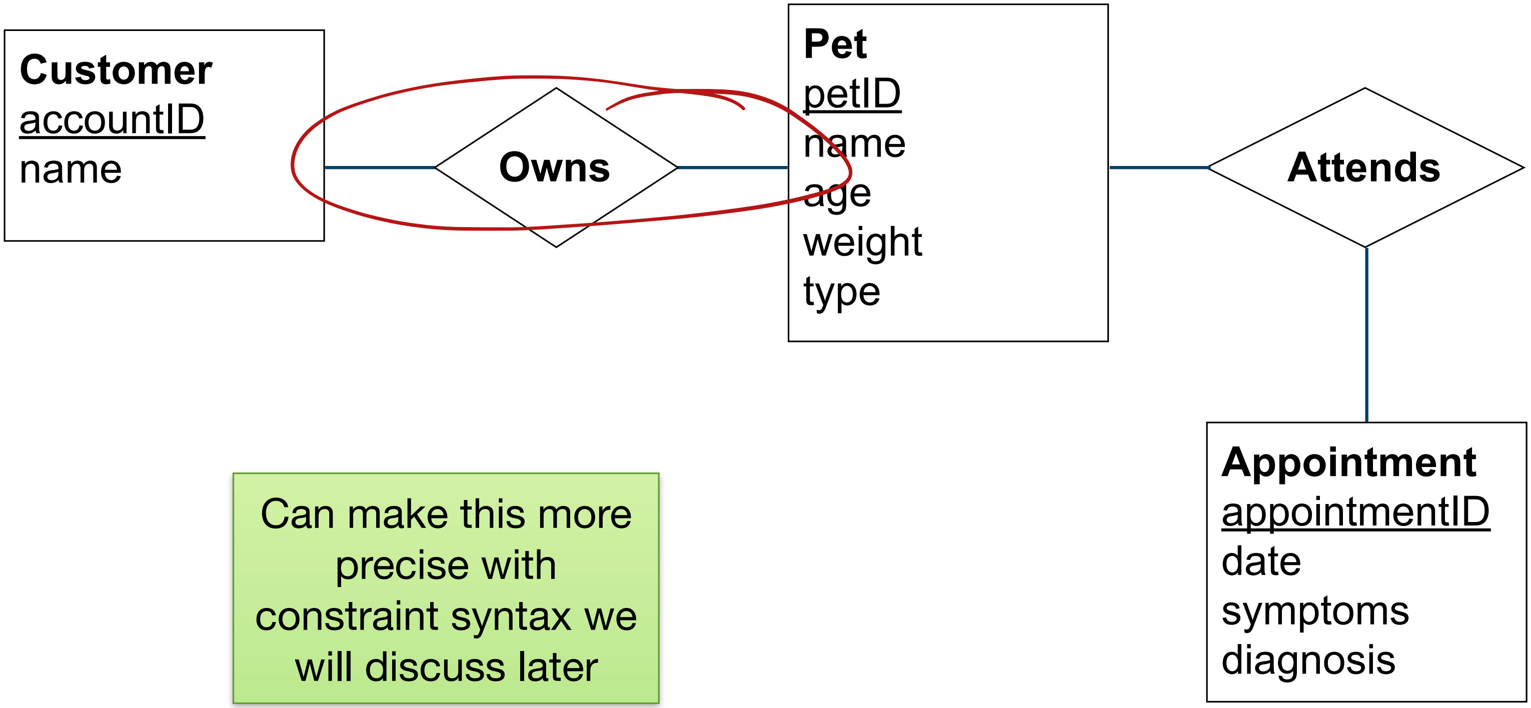
Pet Example

A veterinary clinic wants to track information about its customers (human and animal). Pet owners have a name and account ID. Pets have a name, age, and weight. Whenever a pet comes for an appointment we must record a date, symptoms, and diagnosis.



Pet Solution

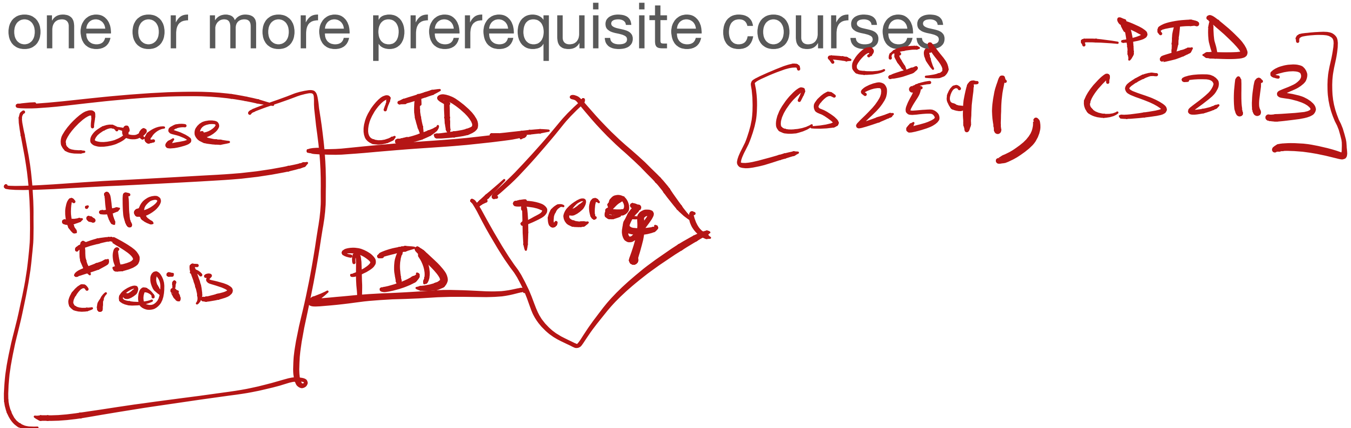
A veterinary clinic wants to track information about its customers (human and animal). Pet owners have a name and account ID. Pets have a name, age, and weight. Whenever a pet comes for an appointment we must record a date, symptoms, and diagnosis.



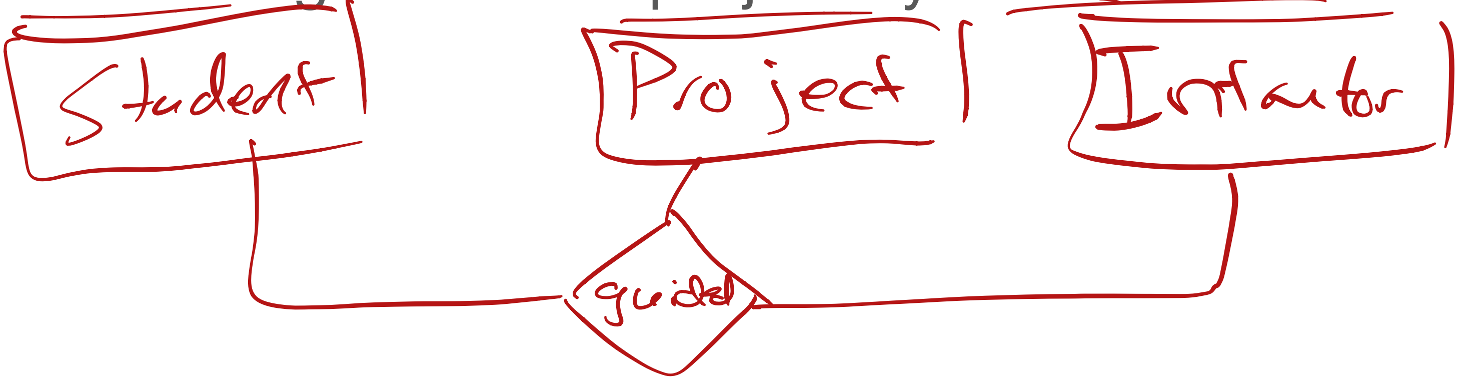
Can make this more precise with constraint syntax we will discuss later

What about these?

1 A course has a title, ID, number of credits, and may have one or more prerequisite courses

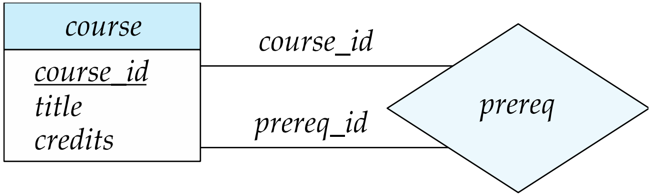


2 A student is guided on a project by an instructor



What about these?

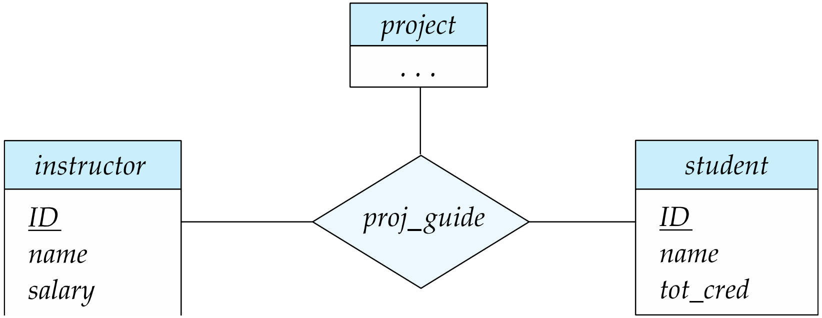
A course has a title, ID, number of credits, and may have one or more prerequisite courses



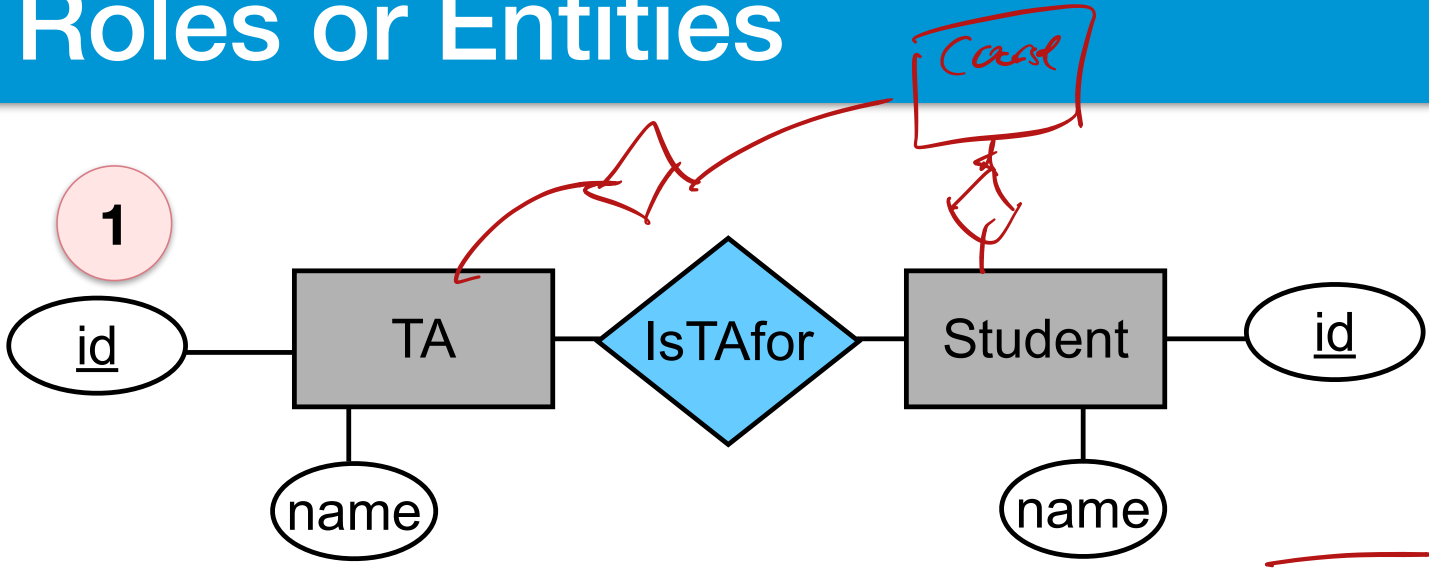
Roles can annotate a connection when a relationship links multiple of the same type of entity

A student is guided on a project by an instructor

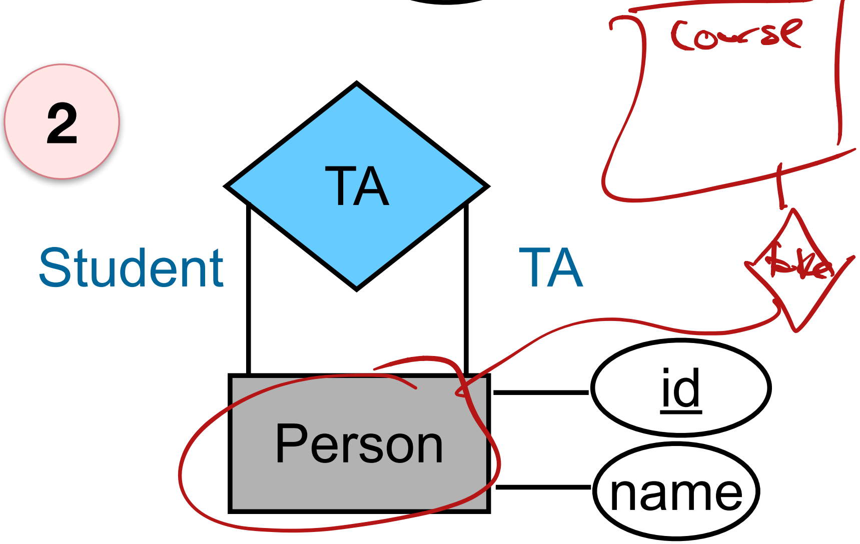
Relationships do not need to be "binary" - can link > 2 entities



Roles or Entities



Are these the same? Which is better?



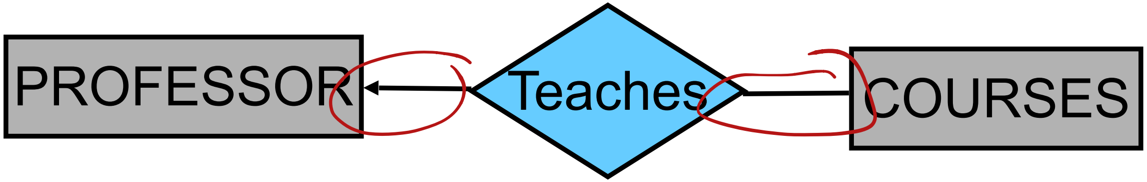
Connectivity in the E-R Diagram

Attributes can **only** be connected to entities or relationships

Entities can **only** be connected to other entities via relationships

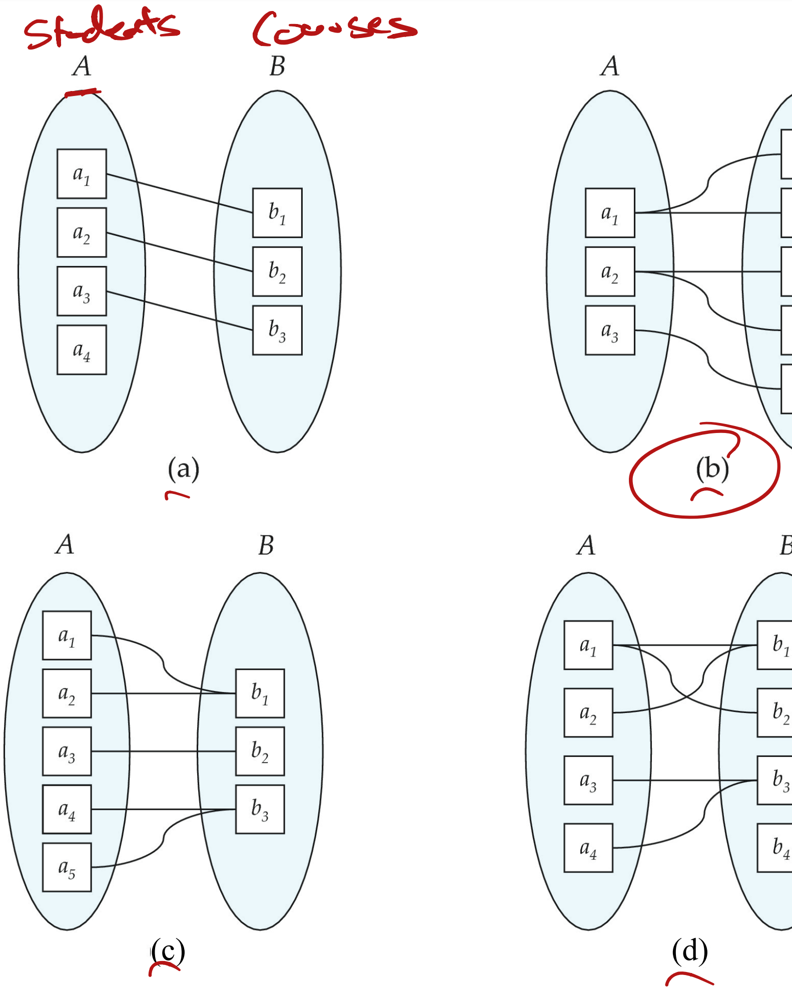
Edges represents **kinds of relationships** and **integrity constraints**

- Use arrows and cardinality annotations



(warning: different ER implementations have slightly different notations!)

Relationship Cardinality



(a)

(b)

(c)

(d)

A → B

Which represents…

B One to Many

A One to One

C Many to One

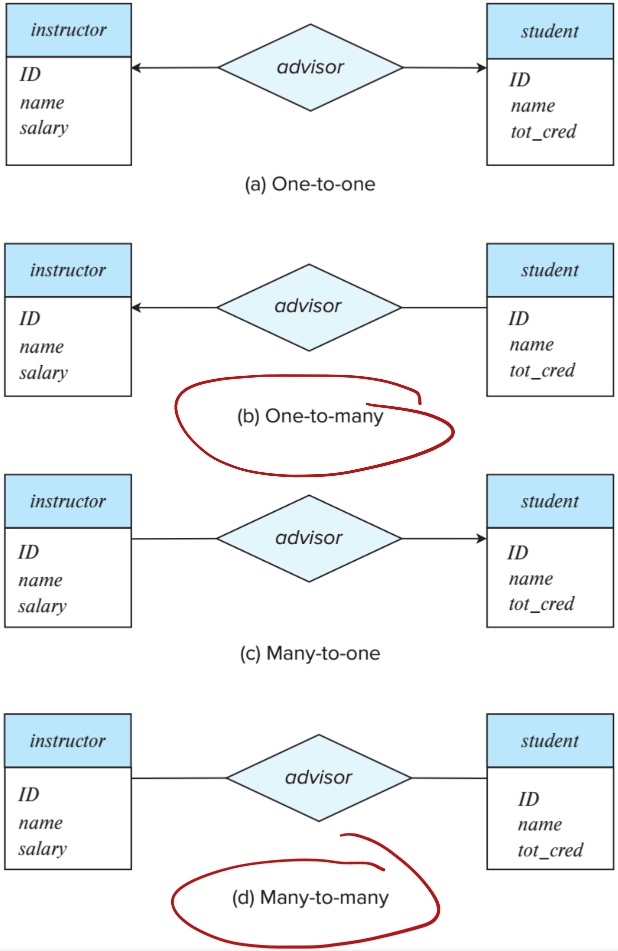
D Many to Many

Relationship Cardinality

We use **arrows** in ER diagrams to indicate cardinality

- An arrow pointing to an Entity means “one” for that entity
- No arrow means “many” of that entity

Which relationship best represents undergrad advising at GW? Why?



(a) One-to-one

(b) One-to-many

(c) Many-to-one

(d) Many-to-many

Relationship Cardinality

One-to-One

- An instructor can only advise one student and a student can only have one advisor

One-to-Many

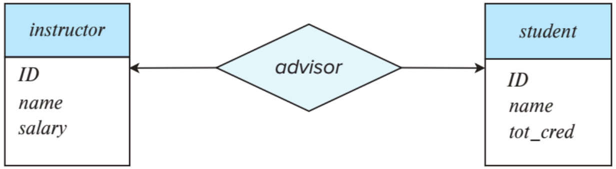
- An instructor can advise many students, but each student only has one advisor

Many-to-One

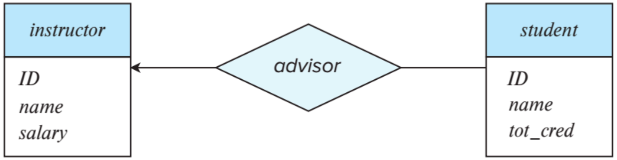
- An instructor can only advise one student, but each student can have many advisors

Many-to-Many

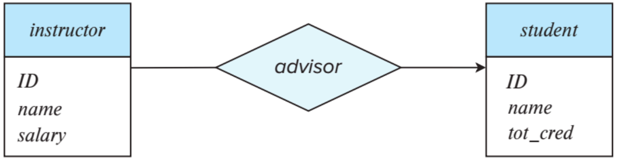
- An instructor can advise many students and each student can have multiple advisors



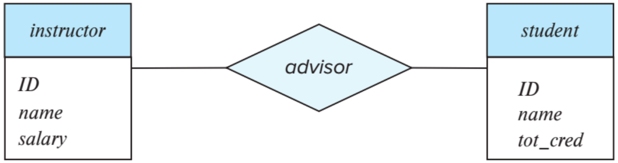
(a) One-to-one



(b) One-to-many



(c) Many-to-one



(d) Many-to-many

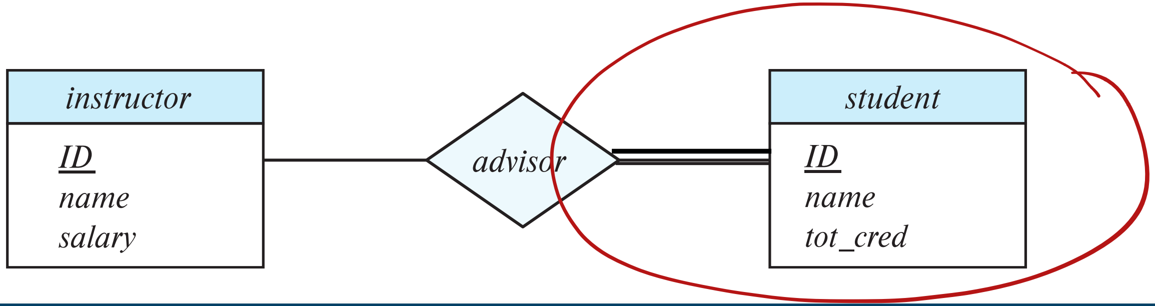
Participation Constraints

Cardinality constraints are upper bound limits

- Limits the maximum number of entities referenced by a relation

Participation Constraints

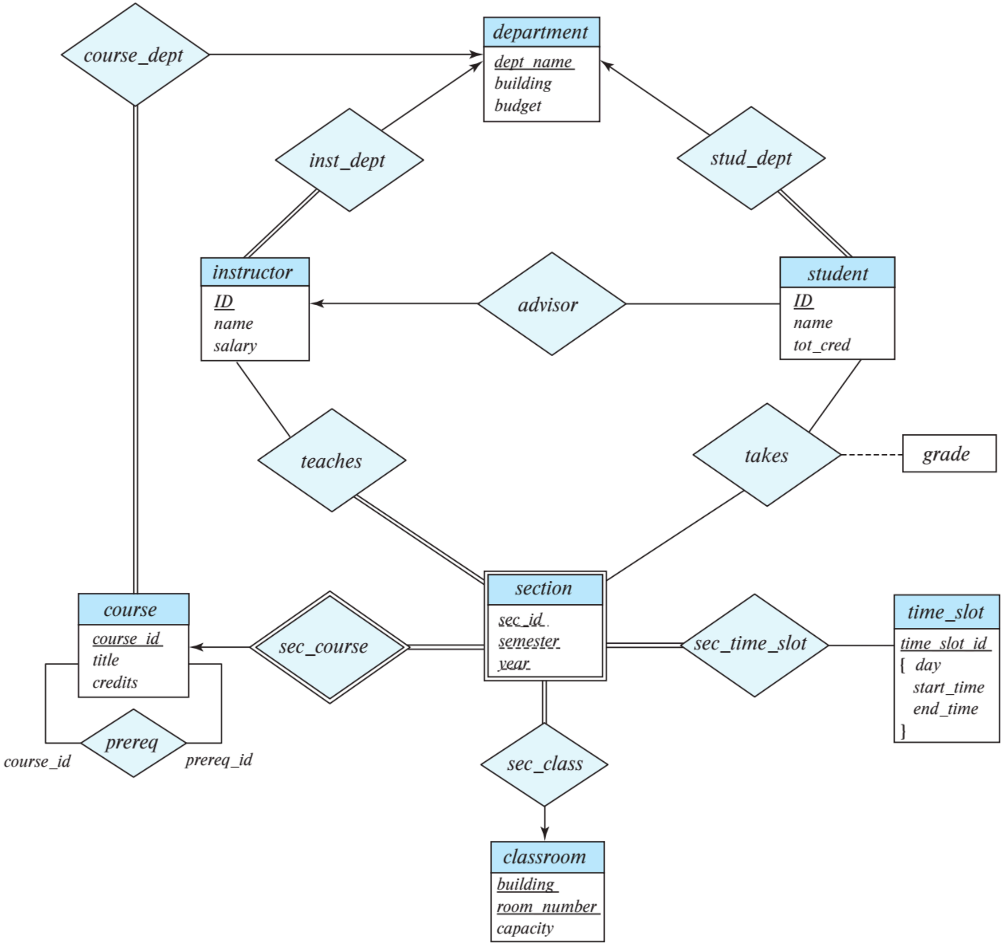
- **Total participation**: all elements from an Entity Set must appear in the Relationship Set (Syntax: double line)
    - Example: “Every student needs an advisor” -> Total participation of Student and Advisor relation
- **Partial participation**: relationship is optional (Syntax: single line)
    - Example: “Not all instructors advise students”



Complete ER University

Making an ER diagram can…

- Help you understand what constraints are important
- Eliminate redundant data fields across Entities
- Think about important edge cases



From ER to SQL

Once we have an ER model, we can transform it into a SQL (or other) format

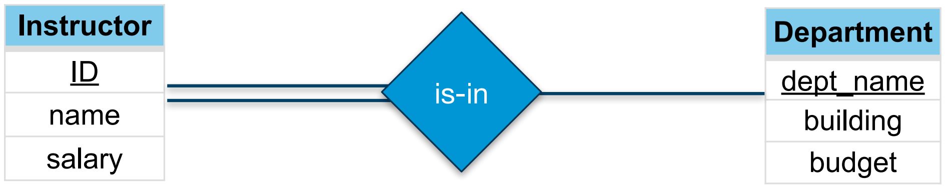
- ER gives us a principled way to define our SQL schema

Relationships map to tables and/or foreign key constraints

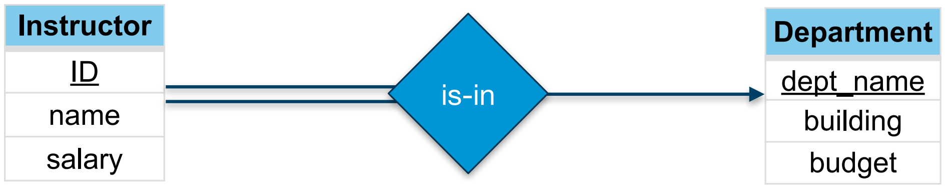
- Simplest approach is every Entity and every Relationship becomes a new Table in SQL
- But *-1 relationships can then be merged with another table, eliminating redundancy

Instructors and Departments

1

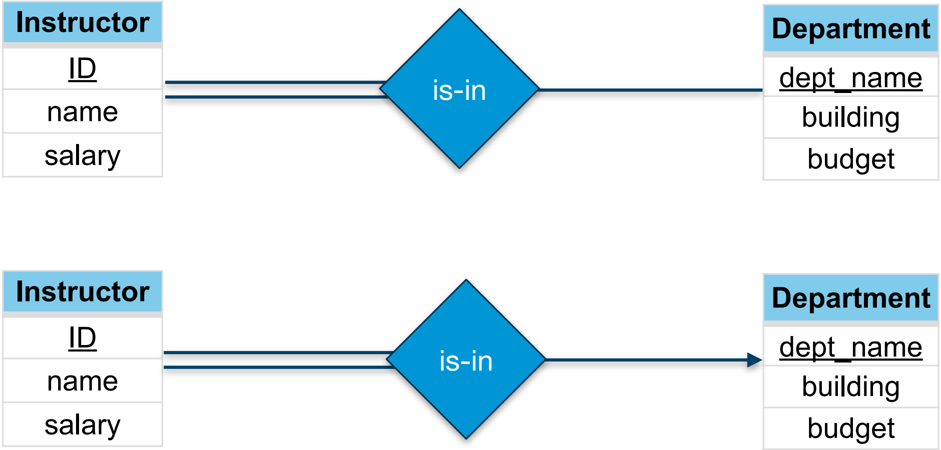


2



What do each of these mean?

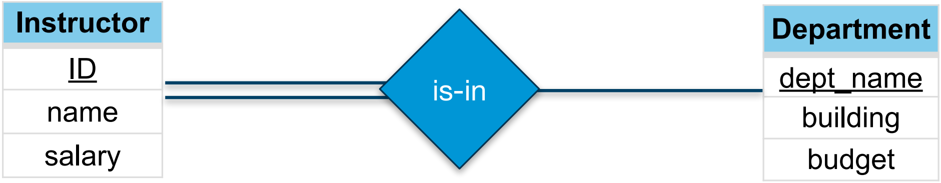
Instructors and Departments

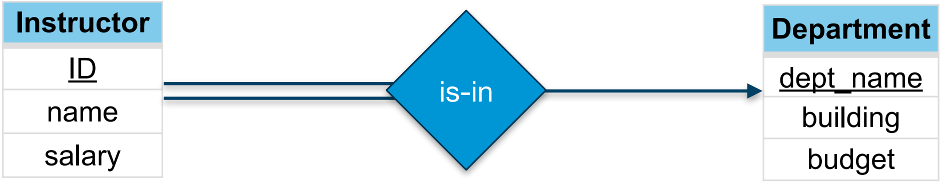


TOP: Every instructor is in at least one department

BOTTOM: Every instructor is in one department

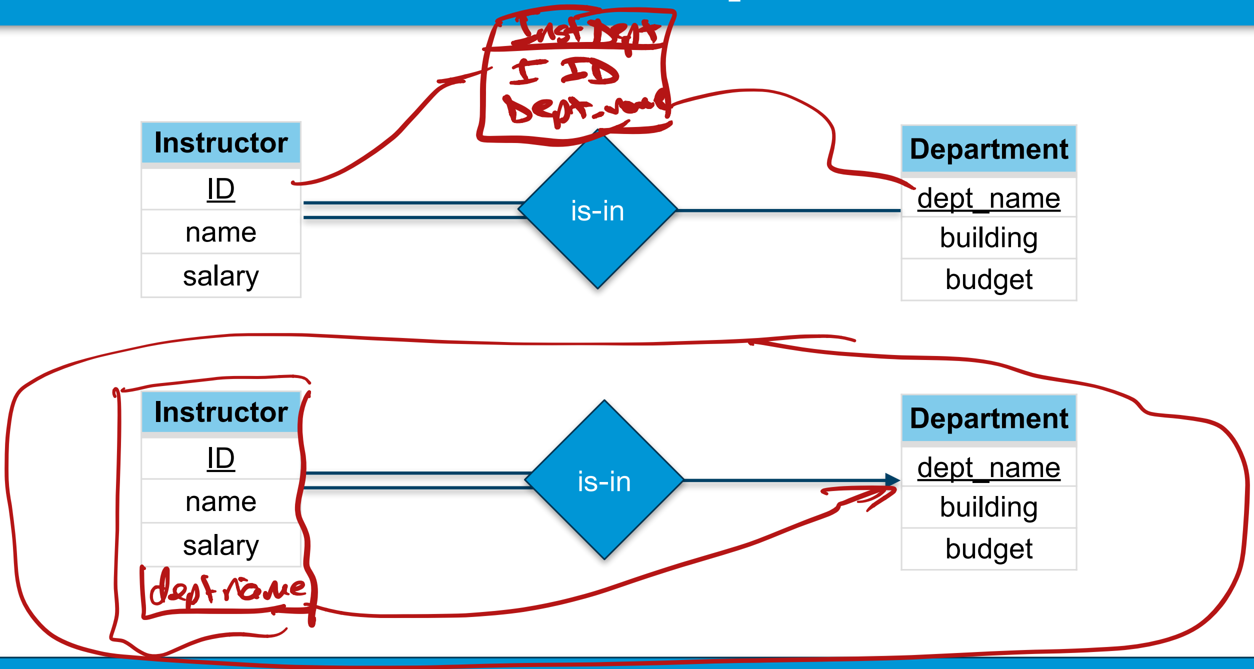
Instructors and Departments





How would you implement each of these in SQL?

Instructors and Departments



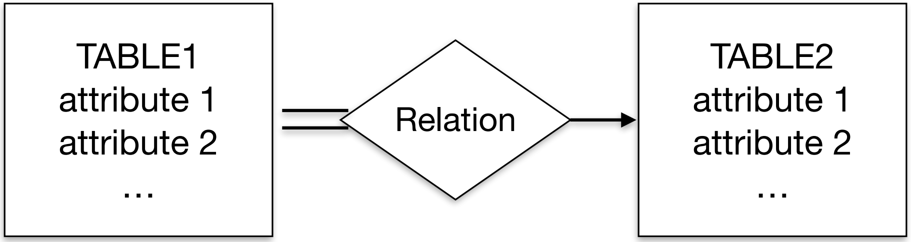
TOP: We would need a third table that would connect each instructor to one or more departments

BOTTOM: We could add dept_name as a foreign key in Instructor

Exercise

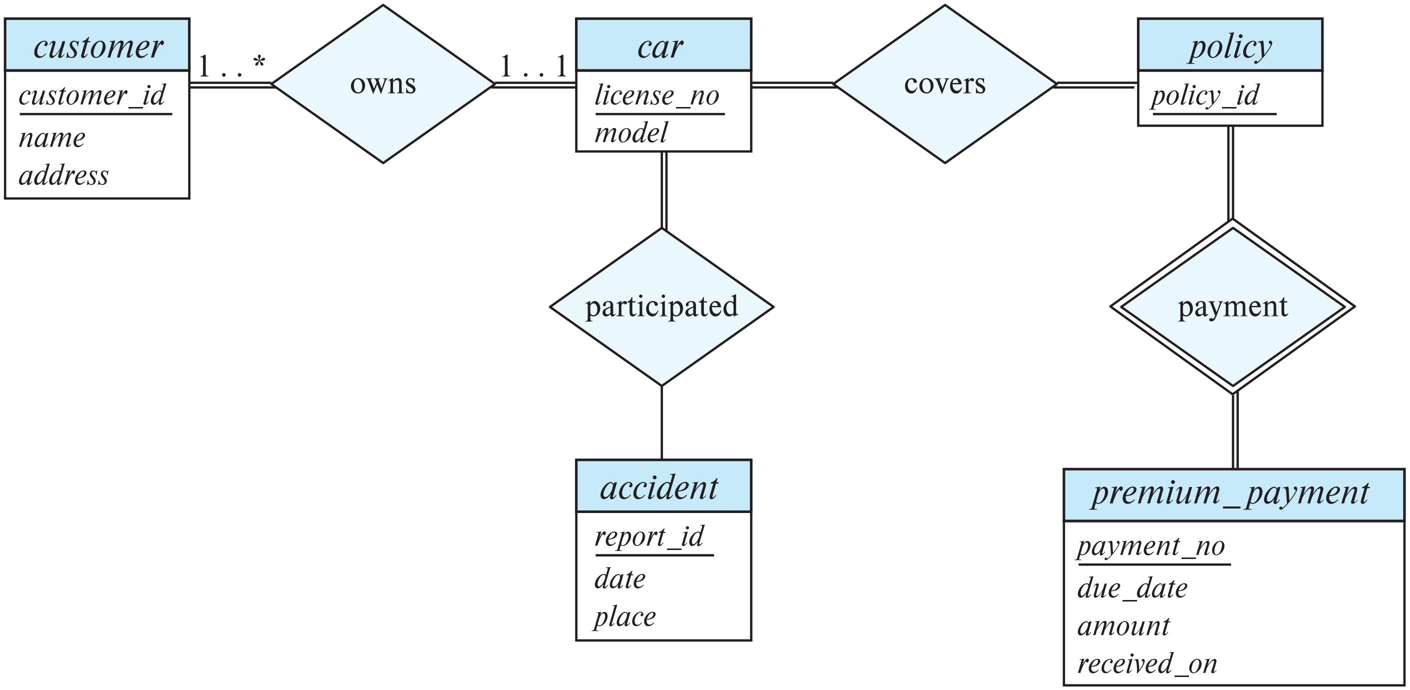
Design an ER diagram for a car insurance company whose customers own one or more cars each. Each car may be associated with a recorded accident. Each insurance policy covers one or more cars and has one or more premium payments associated with it. Each payment is for a particular time period and has an associated due date, and the date when the payment was received.

Sample Syntax



Exercise Sample Answer

Note: this uses some extra syntax / annotations we haven’t discussed



Summary - Conceptual Design

E-R model defines a formal approach for translating business requirements into a data model

Helps identify redundant information and the appropriate ways to link entities

After ER, still need to translate into a DBMS implementation

How can we judge goodness?

Final Phase: Moving from an abstract data model to the implementation of the database

- **1. Logical Design** – Deciding on a "good" collection of schemas
  - **Business decision –** What attributes should we record in the database?
  - **Computer Science decision –** What relation schemas should we have and how should the attributes be distributed among the various relation schemas?
- **2. Physical Design** – Deciding on the physical layout of the database
  - The DBMS will do some of this for us
  - But we can control things like how indexes are generated to optimize frequent data lookups (later)