The Entity-Relationship (ER) Model 2

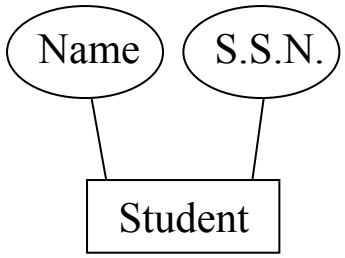
Week 2 - 1

Professor Jessica Lin

Keys

Differences between entities must be expressed in terms of attributes.

• A **superkey** is a set of one or more attributes which, taken collectively, allow us to identify uniquely an entity in the entity set.
• For example, in the entity set *student*, *{name*, *S.S.N.}* is a superkey.
• Note that *name* alone is not, as two students could have the same name.
• A superkey may contain extraneous attributes, and we are often interested in the smallest superkey. A superkey for which no subset is a superkey is called a **candidate key**.



Name	**S.S.N.**
Lisa	111-11-1111
Bart	222-22-2222
Lisa	333-33-3333
Sue	444-44-4444

We can see that *{Name,S.S.N.}* is a **superkey**

In this example, *S.S.N.* is a **candidate** key, as it is minimal, and uniquely identifies a students entity.

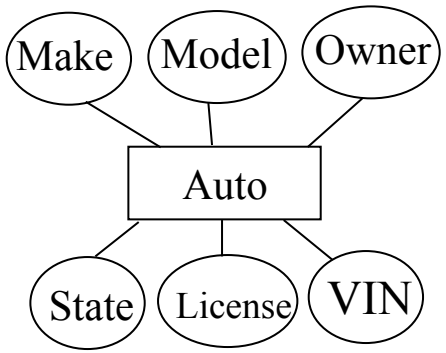
Keys Cont.

•A **primary key** is a candidate key (there may be more than one) chosen by the DB designer to identify entities in an entity set.

In the example below…

{Make,Model,Owner,State,License#,VIN#} is a superkey
{State,License#,VIN#} is a superkey
{Make,Model,Owner} is *not* a superkey

{State,License#} is a candidate key
{VIN#} is a candidate key

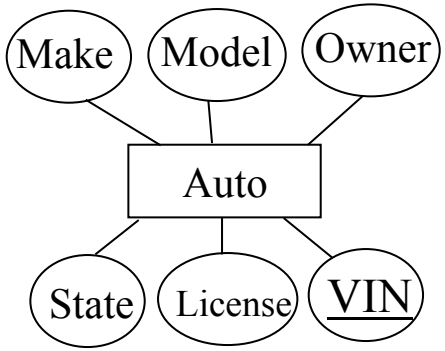


VIN# is the logical choice for **primary key**

Make	**Model**	**Owner**	**State**	**License #**	**VIN #**
Ford	Focus	Mike	CA	SD123	34724
BMW	Z4	Joe	CA	JOE	55725
Ford	Escort	Sue	AZ	TD4352	75822
Honda	Civic	Bert	CA	456GHf	77924

Keys Cont.

- The **primary key** is denoted in an ER diagram by underlining.
- An entity that has a primary key is called a **strong entity**.



Note that a good choice of primary key is very important!

For example, it is usually much faster to search a database by the primary key, than by any other key (we will see why later).

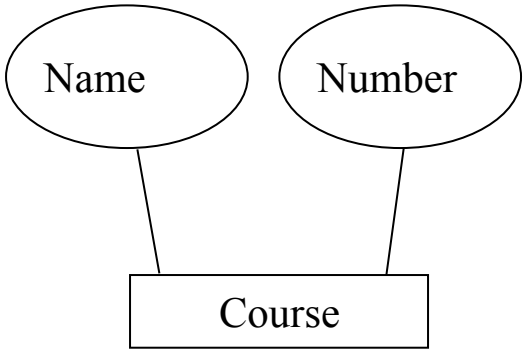
Keys Cont.

An entity set that does not possess sufficient attributes to form a primary key is called a **weak entity set**.

In the example below, there are two different sections of Java being offered (let's say, for example, one by Dr. Smith, one by Dr. Lee).

{Name,Number} is not a superkey, and therefore *course* is a **weak entity**.

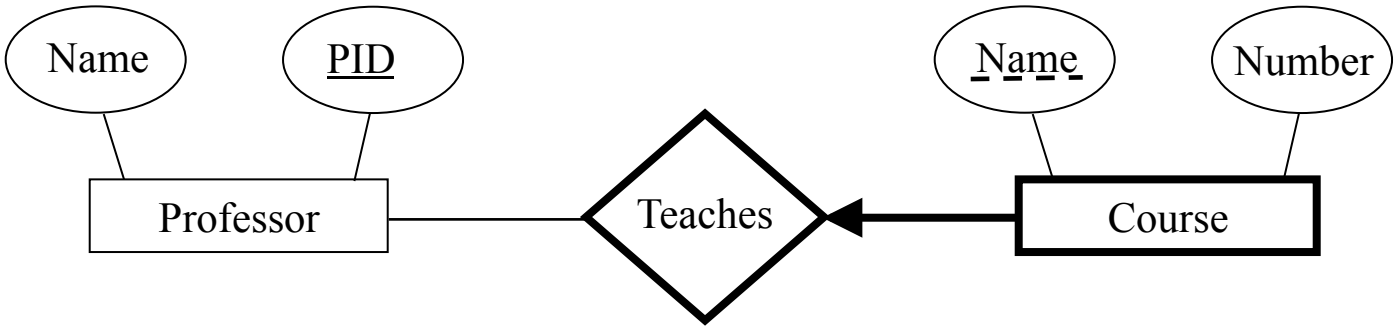
This is clearly a problem, we need some way to distinguish between different courses….



Name	Number
Java	CS211
AI	CS480
Java	CS211
DB	CS450

Keys Cont.

In order to be able to uniquely refer to an item in a weak entity set we must consider some (or all) of its attributes in conjunction with some strong entities primary key. The entity whose primary key is being used is called the **identifying owner**.



For this to work, two conditions must be met.

- The weak entity set must have total participation in the relationship
- The identifying owner and the weak entity must participate is a one-to-many relationship.

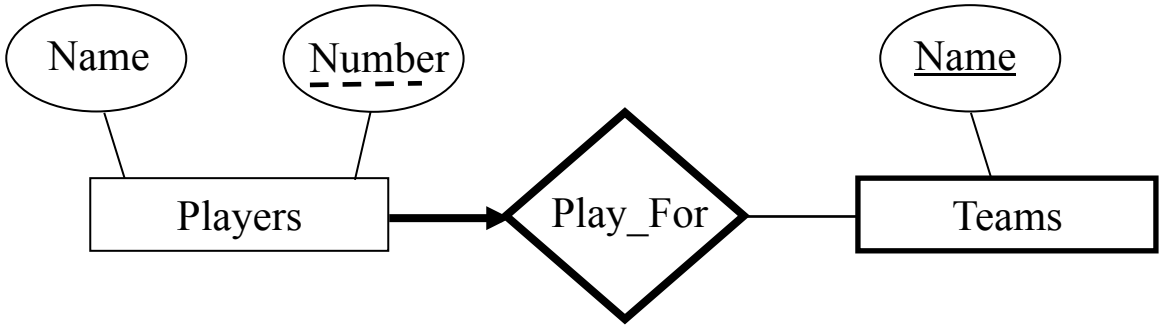
Name	PID
Smith	2345
Lee	7356
Chan	3571

Name	Number
Java	CS112
AI	CS480
Java	CS112
DB	CS450

Weak Entity Sets Example

- Entity sets Teams, and Players.
  - No team has two players with the same number.
  - However, there can be players with the same number on different teams

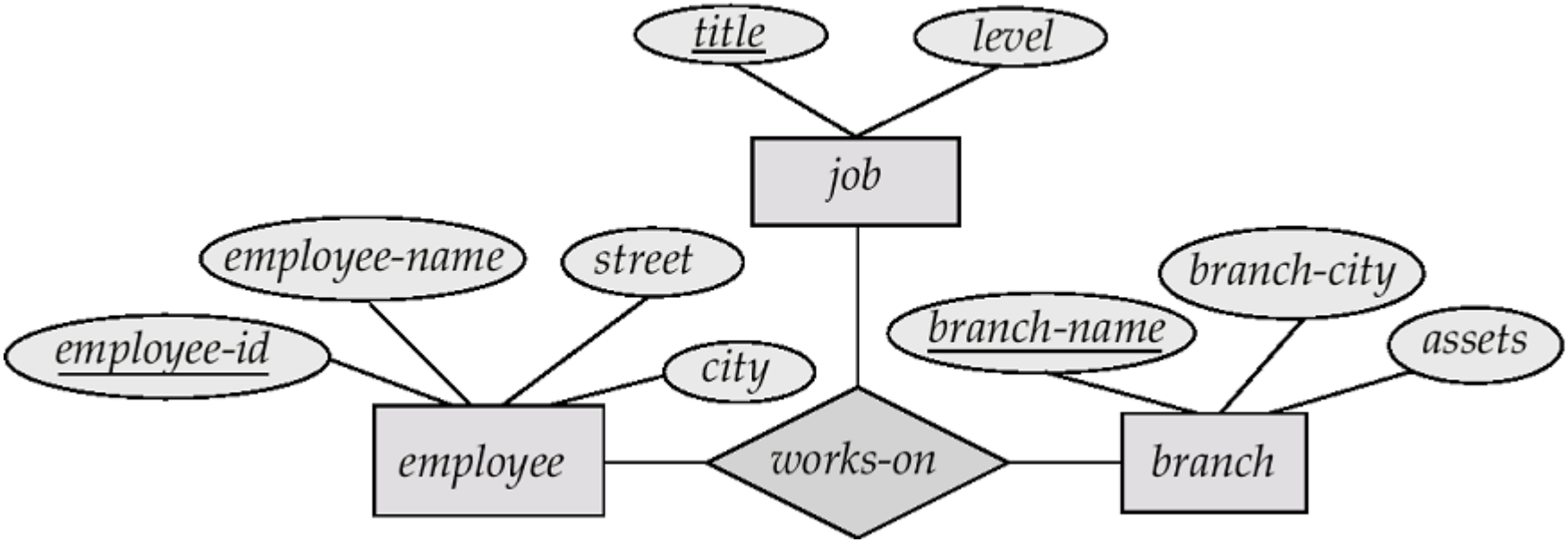
Weak Entity Sets Example



Ternary Relationships

So far, we have only considered binary relationships, however it is possible to have **higher order** relationships, including **ternary** relationships.

Consider the following example that describes the fact that employees at a bank work in one or more bank branches, and have one or more job descriptions.



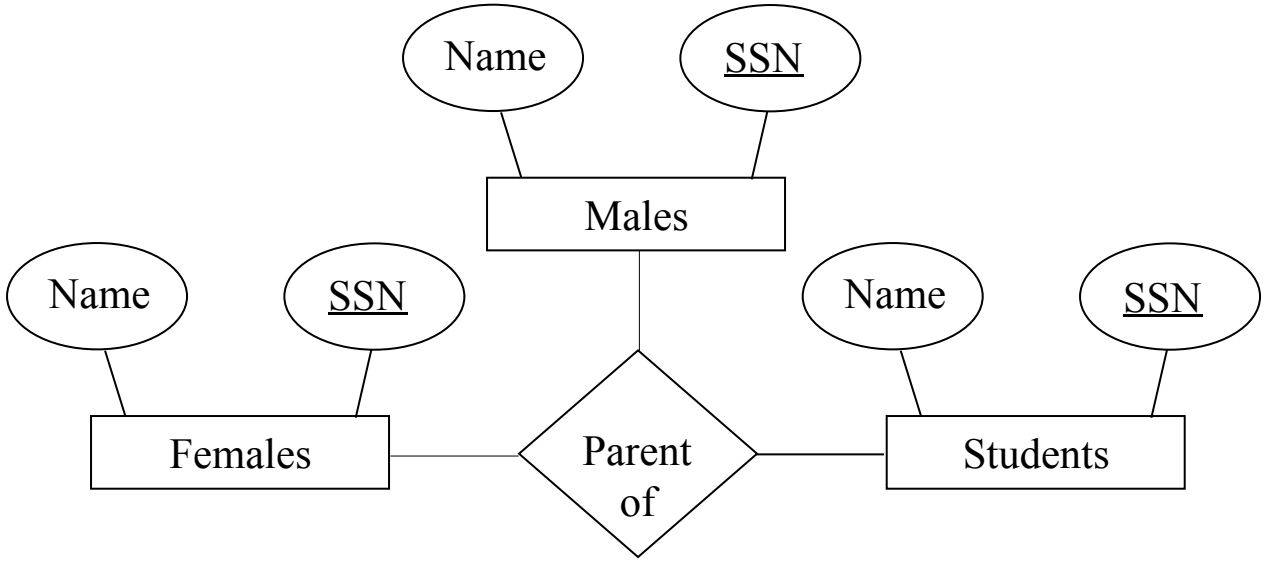
Why not remove the *job* entity by placing the *title* and *level* attributes as part of employee?

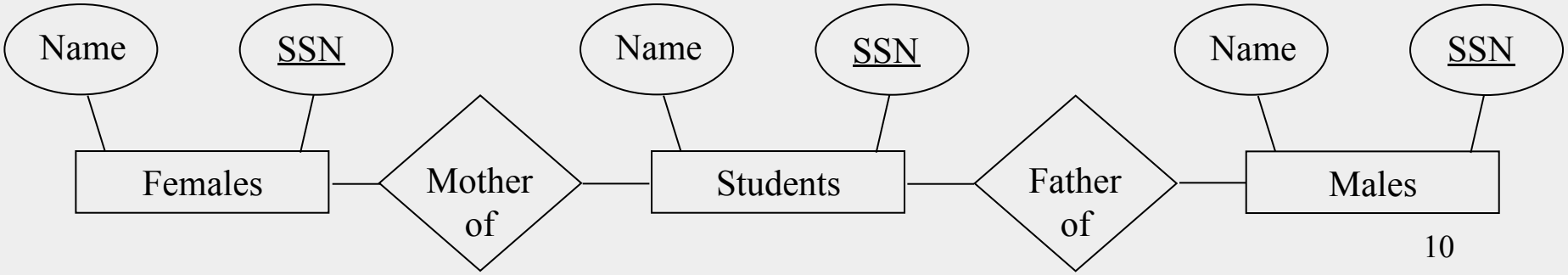
Ternary Relationships

Sometimes you have a choice of a single ternary relationship or two binary relationships…

In general, unless you really need a ternary relationship, use binary relationships.

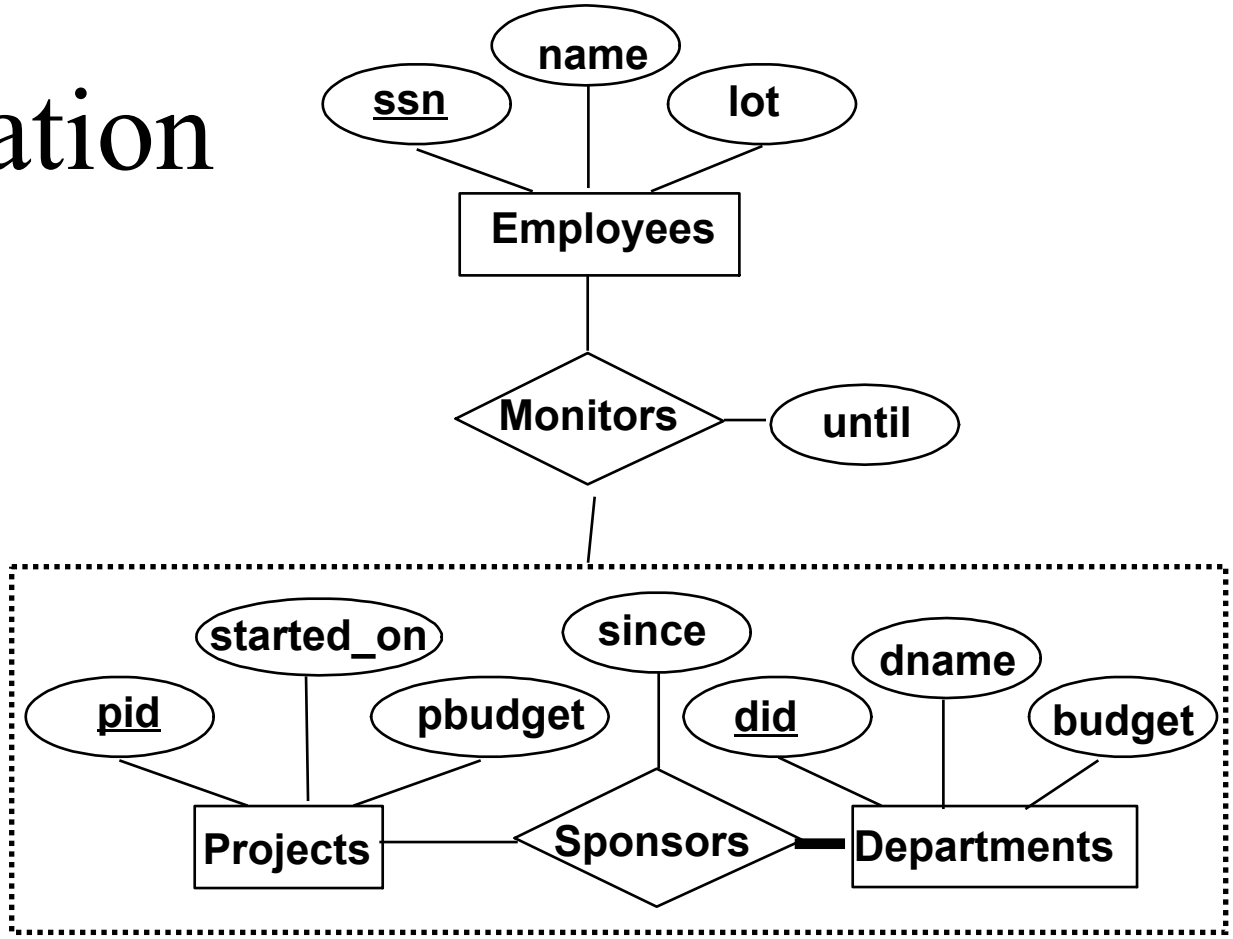
FACT: Every ternary (and higher order) relationship can be converted into a set of binary relationships.





Aggregation

- Used when we have to model a relationship involving (entitity sets and) a *relationship set*.
  - *Aggregation* allows us to treat a relationship set as an entity set for purposes of participation in (other) relationships.



☛ *Aggregation vs. ternary relationship:*

❖ Monitors is a distinct relationship, with a descriptive attribute.

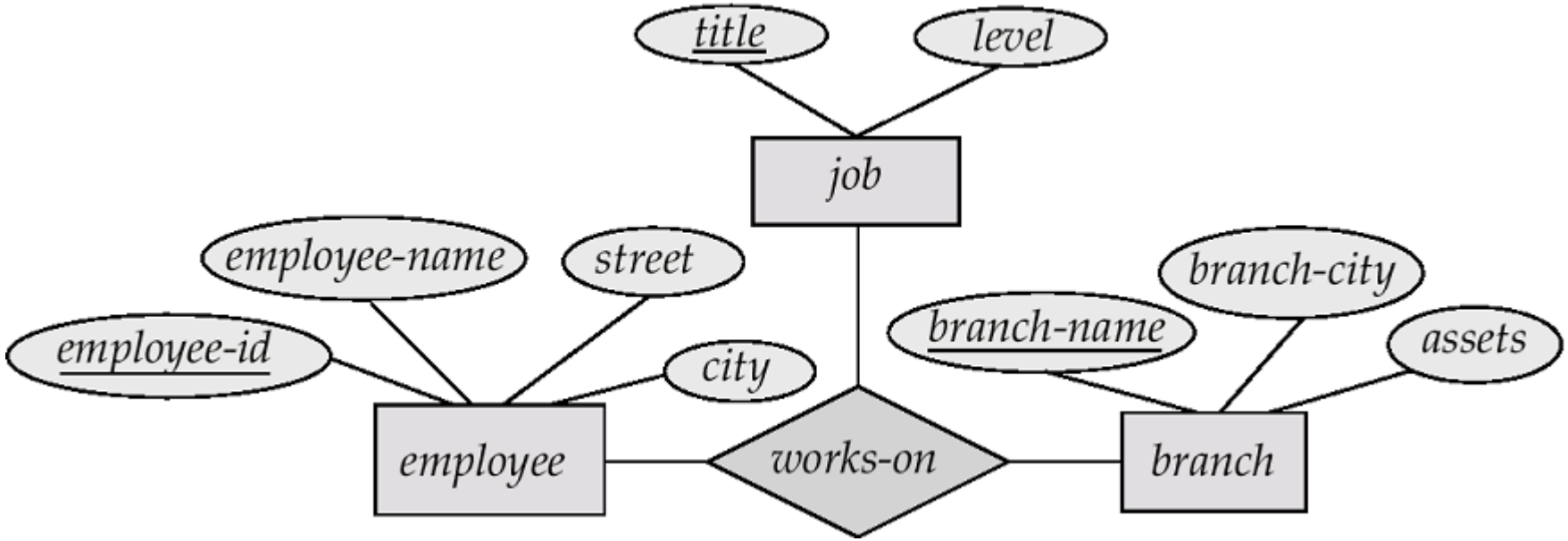
❖ Also, can say that each sponsorship is monitored by at most one employee.

Aggregation, Cont.

Consider this ER model, which we have seen before…

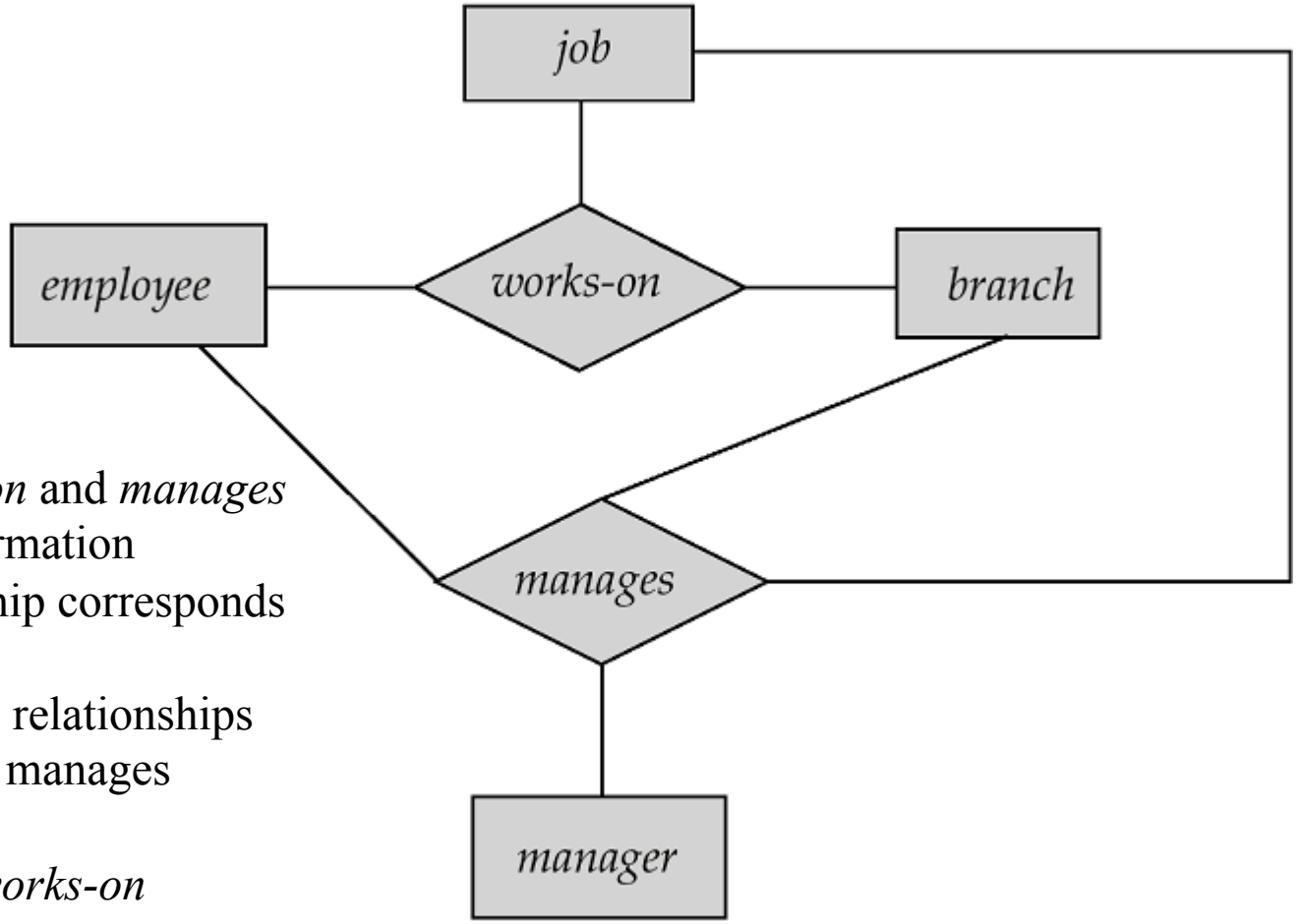
We need to add to it, to reflect that managers manage the various tasks performed by an employee at a branch

Solution?



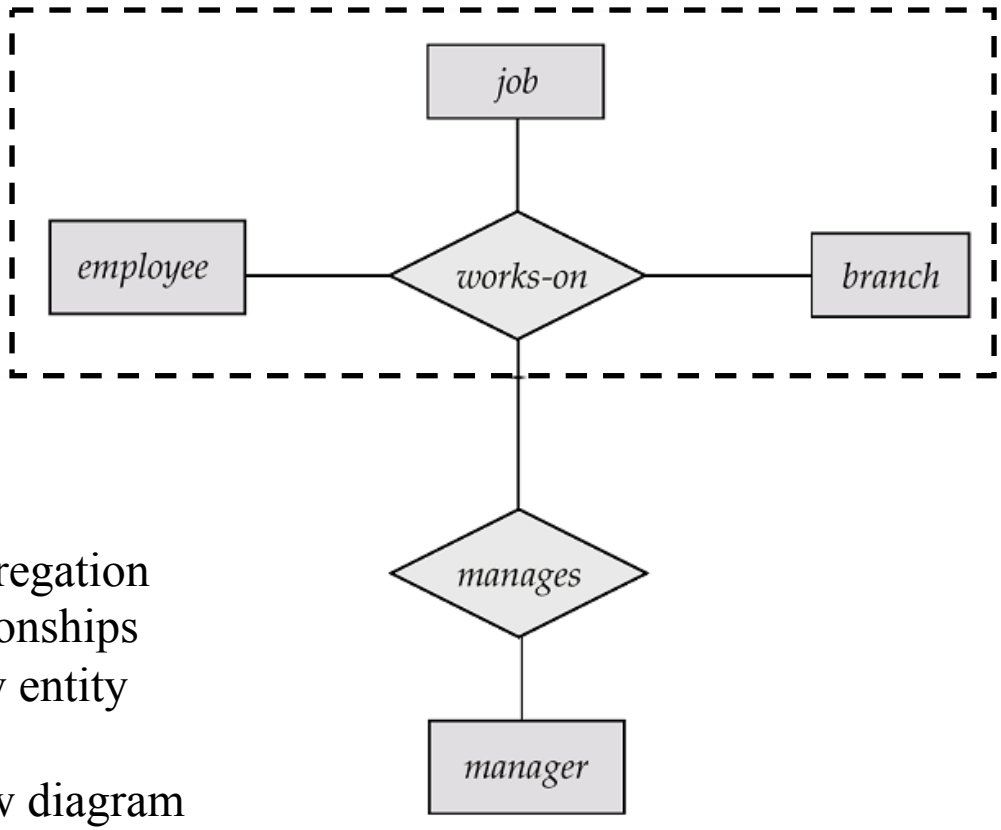
Aggregation Cont.

Note that the attributes are omitted for graphical simplicity.



- Relationship sets *works-on* and *manages* represent overlapping information
- Every *manages* relationship corresponds to a works-on relationship
- However, some *works-on* relationships may not correspond to any manages relationships
- So we can't discard the *works-on* relationship

Aggregation Cont.



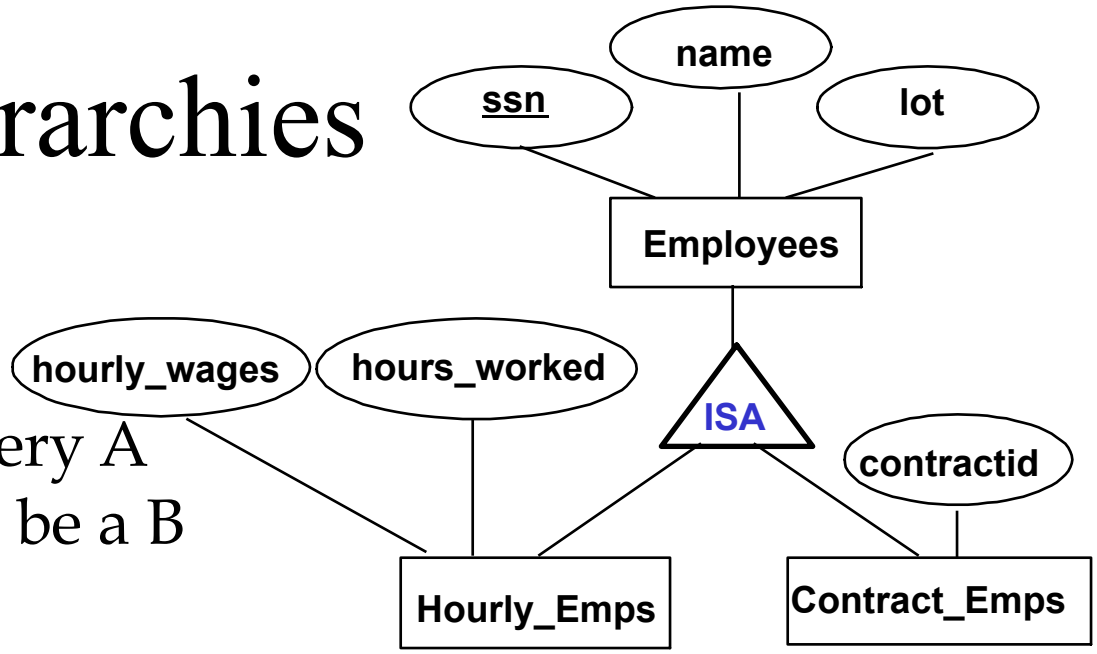
We can eliminate this redundancy via aggregation

- Allows relationships between relationships
- Abstraction of relationship into new entity

• Without introducing redundancy, the new diagram represents:

- An employee works on a particular job at a particular branch
- An employee, branch, job combination may have an associated manager.

ISA (`is a') Hierarchies



- As in C++, or other PLs, attributes are inherited.
- If we declare A **ISA** B, every A entity is also considered to be a B entity.

- *Overlap constraints*: Can Joe be an Hourly_Emps as well as a Contract_Emps entity? (*default: disallowed; A overlaps B*)
- *Covering constraints*: Does every Employees entity also have to be an Hourly_Emps or a Contract_Emps entity? *(default: no; A AND B COVER C)*
- Reasons for using ISA:
  - To add descriptive attributes specific to a subclass.
  - To identify entities that participate in a relationship.

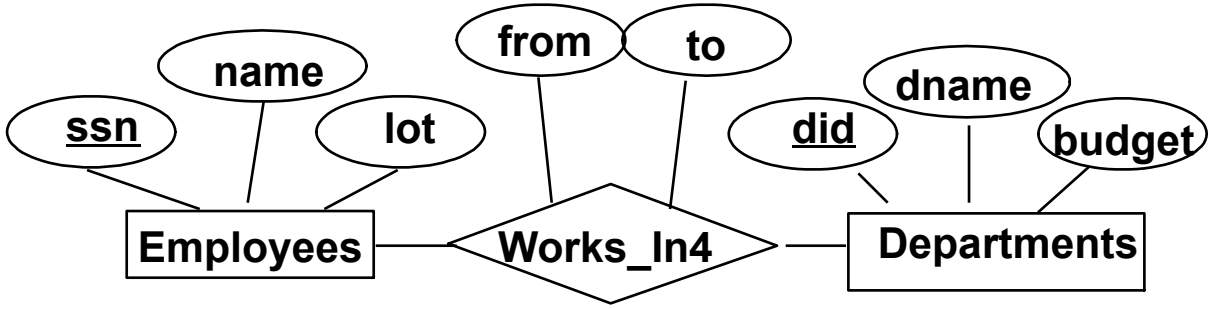
ER Design Decisions

- The use of an attribute or entity set to represent an object.
- Whether a real-world concept is best expressed by an entity set or a relationship set.
- The use of a ternary relationship versus a pair of binary relationships.
- The use of a strong or weak entity set.
- The use of aggregation – can treat the aggregate entity set as a single unit without concern for the details of its internal structure.

Entity vs. Attribute

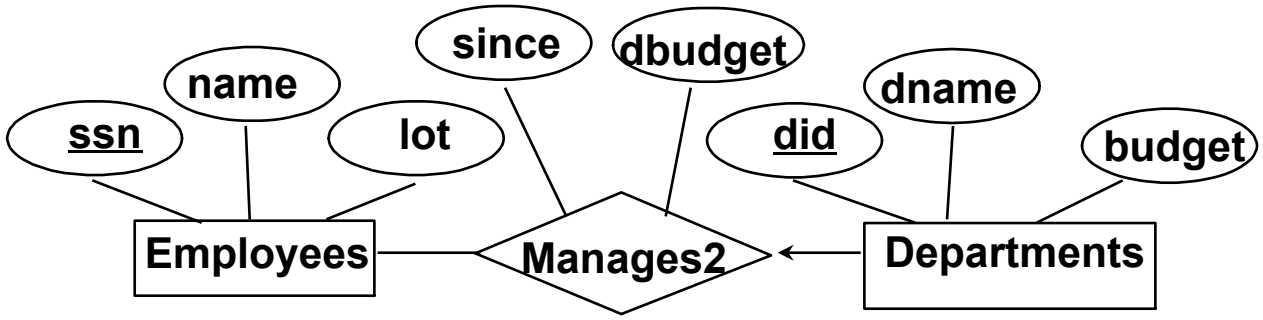
- Should *address* be an attribute of Employees or an entity (connected to Employees by a relationship)?
- Depends upon the use we want to make of address information, and the semantics of the data:
  - If we have several addresses per employee, *address* must be an entity (since attributes cannot be set-valued).
  - If the structure (city, street, etc.) is important, e.g., we want to retrieve employees in a given city, *address* must be modeled as an entity (since attribute values are atomic).

Entity vs. Attribute (Cont.)



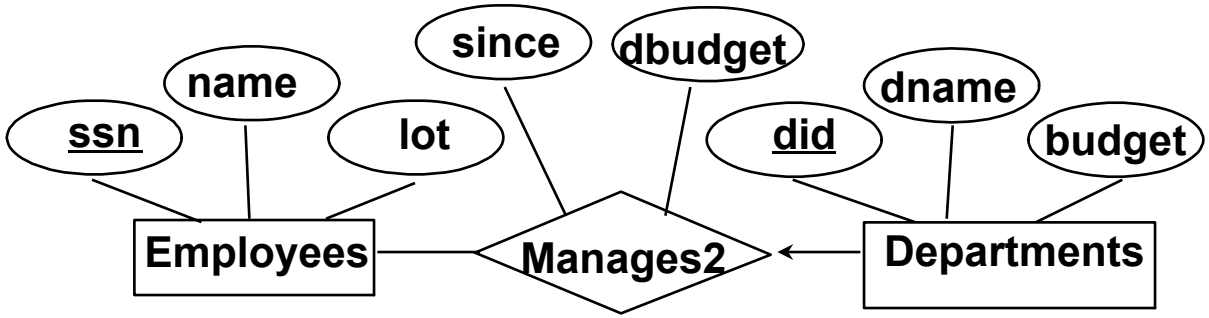
- Works_In4 does not allow an employee to work in a department for two or more periods.
- Similar to the problem of wanting to record several addresses for an employee: We want to record *several values of the descriptive attributes for each instance of this relationship.*
- Solution?

Entity vs. Relationship

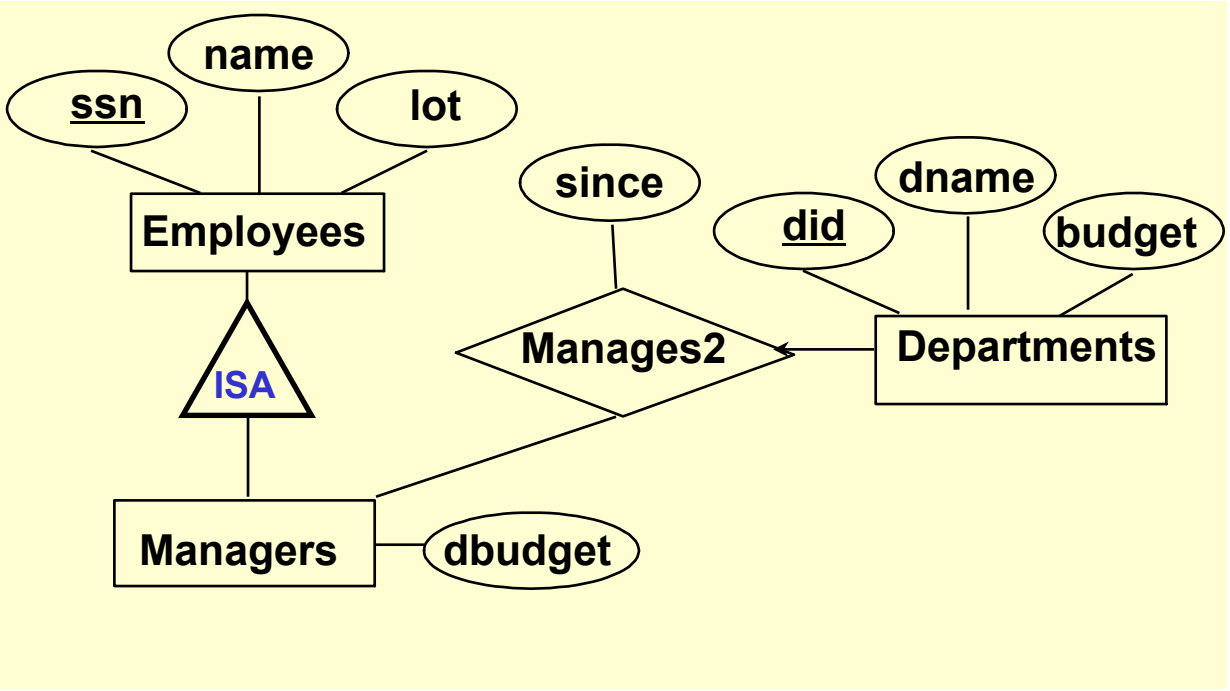


- This ER diagram OK if a manager gets a separate discretionary budget for each dept.
- What if a manager gets a discretionary budget that covers *all* managed depts?
  - Redundancy: *dbudget* stored for each dept managed by manager.
  - Misleading: Suggests *dbudget* associated with department-mgr combination.

Entity vs. Relationship



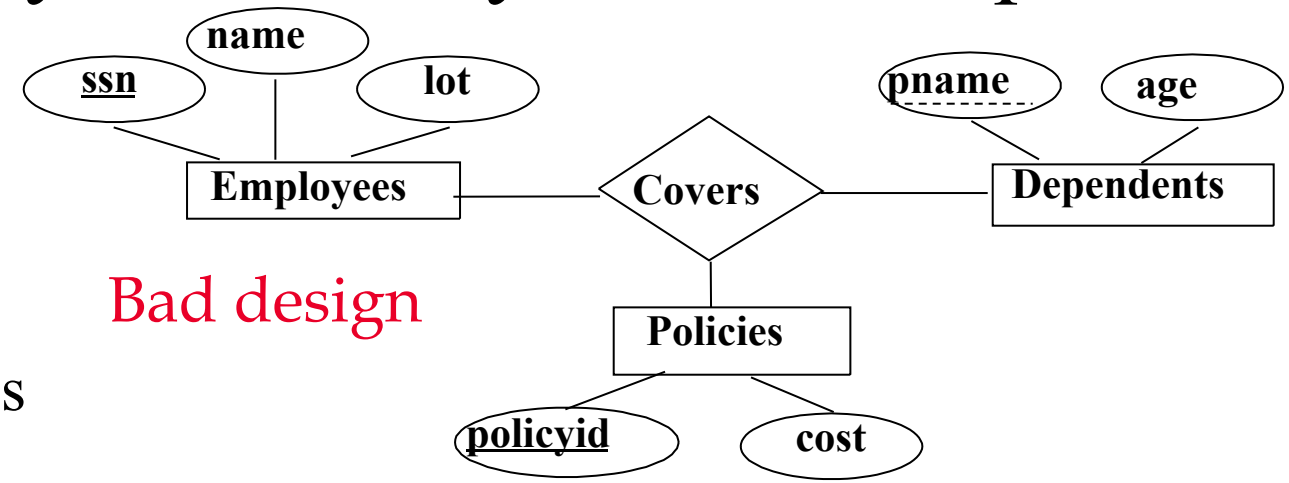
This fixes the Problem!

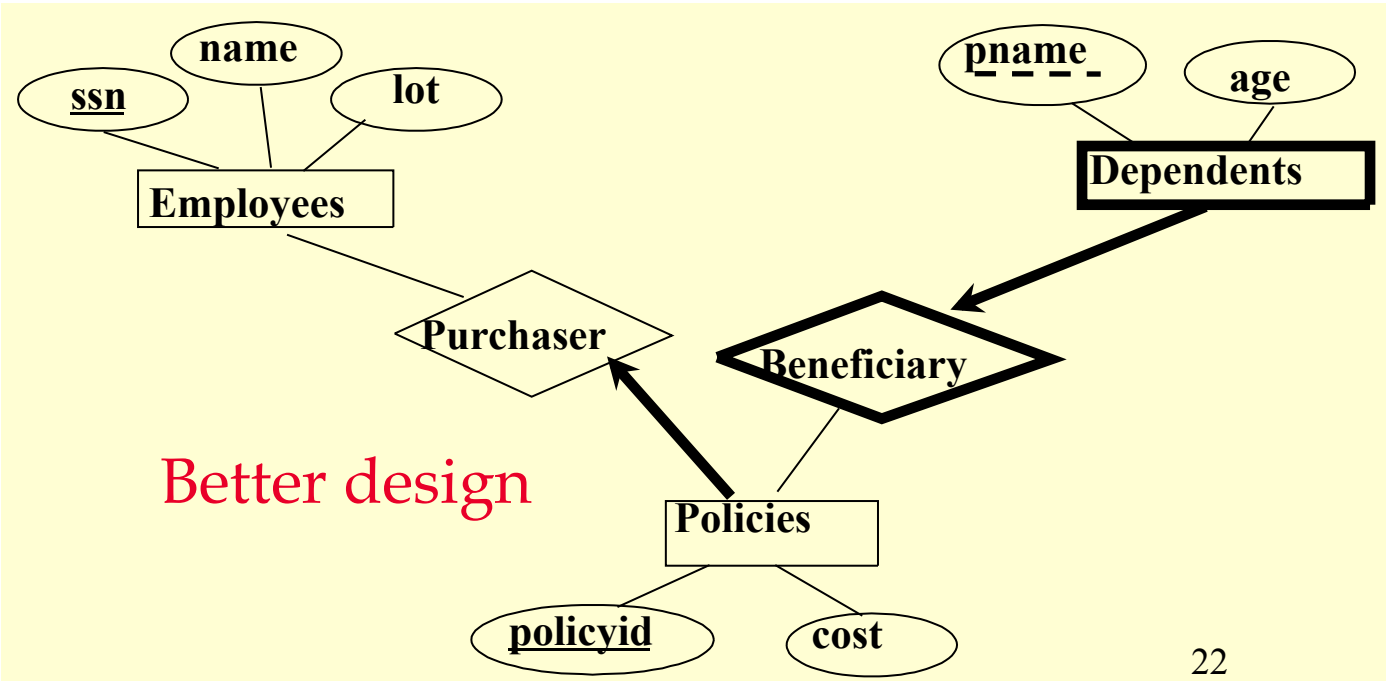


Binary vs. Ternary Relationships

- If each policy is owned by just 1 employee, and each dependent is tied to the covering policy, first diagram is inaccurate.

- What are the additional constraints in the 2nd diagram?





Binary vs. Ternary Relationships (Cont.)

- Previous example illustrated a case when two binary relationships were better than one ternary relationship.
- An example in the other direction: a ternary relation Contracts relates entity sets Parts, Departments and Suppliers, and has descriptive attribute *qty*. No combination of binary relationships is an adequate substitute:
  - S "can-supply" P, D "needs" P, and D "deals-with" S does not imply that D has agreed to buy P from S.
  - How do we record *qty*?

Summary of Conceptual Design

- *Conceptual design* follows *requirements analysis*,
  - Yields a high-level description of data to be stored
- ER model popular for conceptual design
  - Constructs are expressive, close to the way people think about their applications.
- Basic constructs: *entities*, *relationships*, and *attributes* (of entities and relationships).
- Some additional constructs: *weak entities*, *ISA hierarchies*, and *aggregation*.

Summary of ER (Cont.)

- Several kinds of integrity constraints can be expressed in the ER model: *key constraints*, *participation constraints*, and *overlap/covering constraints* for ISA hierarchies.
- Some constraints (notably, *functional dependencies*) cannot be expressed in the ER model.
  - Constraints play an important role in determining the best database design for an enterprise.

Summary of ER (Cont.)

- ER design is *subjective*. There are often many ways to model a given scenario! Analyzing alternatives can be tricky, especially for a large enterprise. Common choices include:
  - Entity vs. attribute, entity vs. relationship, binary or n-ary relationship, whether or not to use ISA hierarchies, and whether or not to use aggregation.
- To ensure good database design, resulting relational schema should be analyzed and refined further. FD information and normalization techniques are especially useful.

Practice Question

- Construct an ER diagram for a car insurance company whose customers own one or more cars each. Each car has associated with it zero to any number of recorded accidents.
  - Entity Sets: ?
  - Relationship: ?