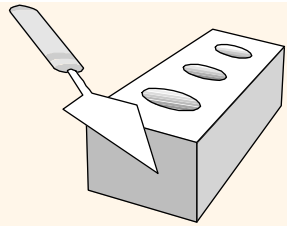
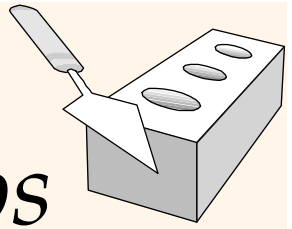


El Modelo Entidad-Relación

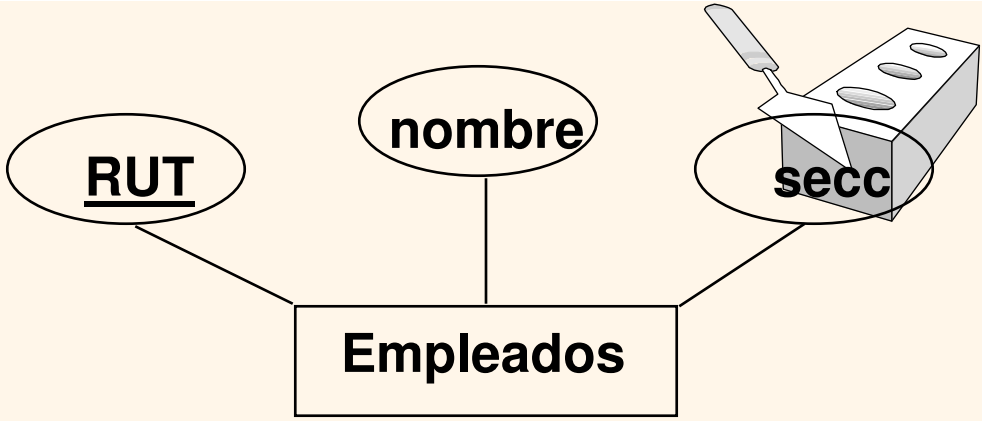
Capítulo 2

Vistazo al Diseño de Base de Datos



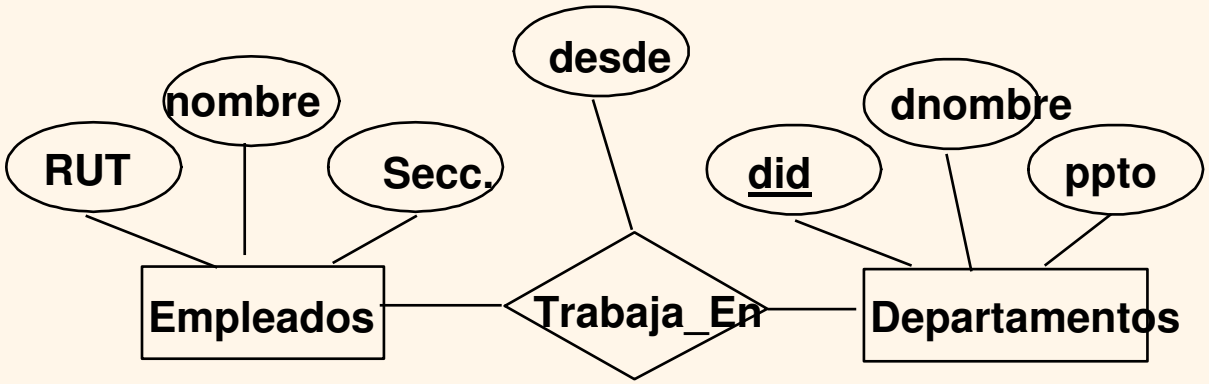
- *Diseño Conceptual*: *(Modelo ER se usa en esta etapa)*
  - ¿Cuáles son las *entitidades* y *relaciones* en la empresa?
  - ¿Que información acerca de esas entidades y relaciones debieramos almacenar en la base de datos?
  - ¿Cuales son las *restricciones de integridad* o *reglas del negocio* que existen?
  - Un esquema de base de datos A en el modelo ER puede ser representada gráficamente (*diagramas ER*).
  - Posteriormente se puede mapear un diagrama ER en un esquema relacional.

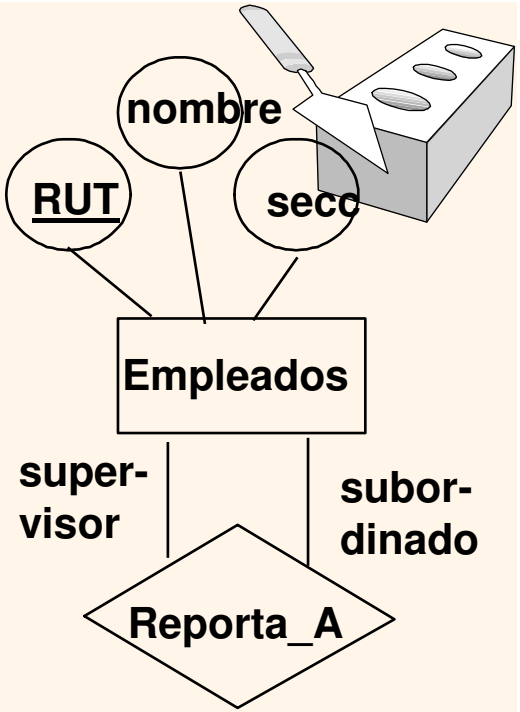
Modelo ER



- *Entidad:* Objeto del mundo real distinguible de otros objetos. Una entidad se describe (en la BD) usando un conjunto de *atributos*.
- *Conjunto de Entidades*: Una colección de entidades similares, por ej. todos los empleados.
  - Todas las entidades de un conjunto tienen los mismos atributos. (al menos hasta que consideremos jerarquías ISA)
  - Cada conjunto de entidades tiene una *llave*.
  - Cada atributo tiene un *dominio*.

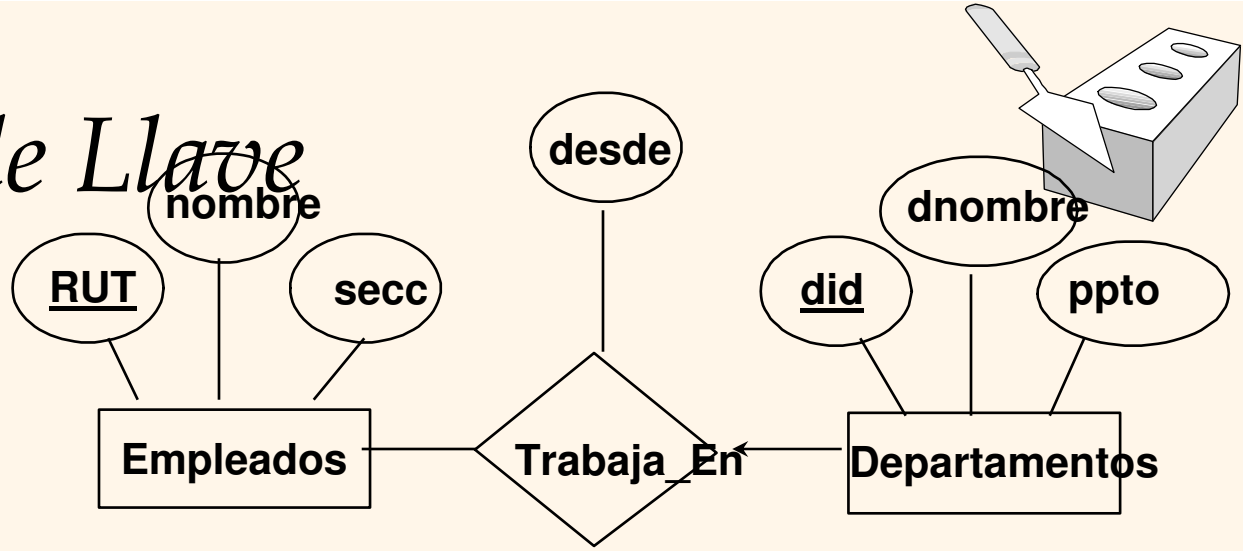
Modelo ER (Cont.)



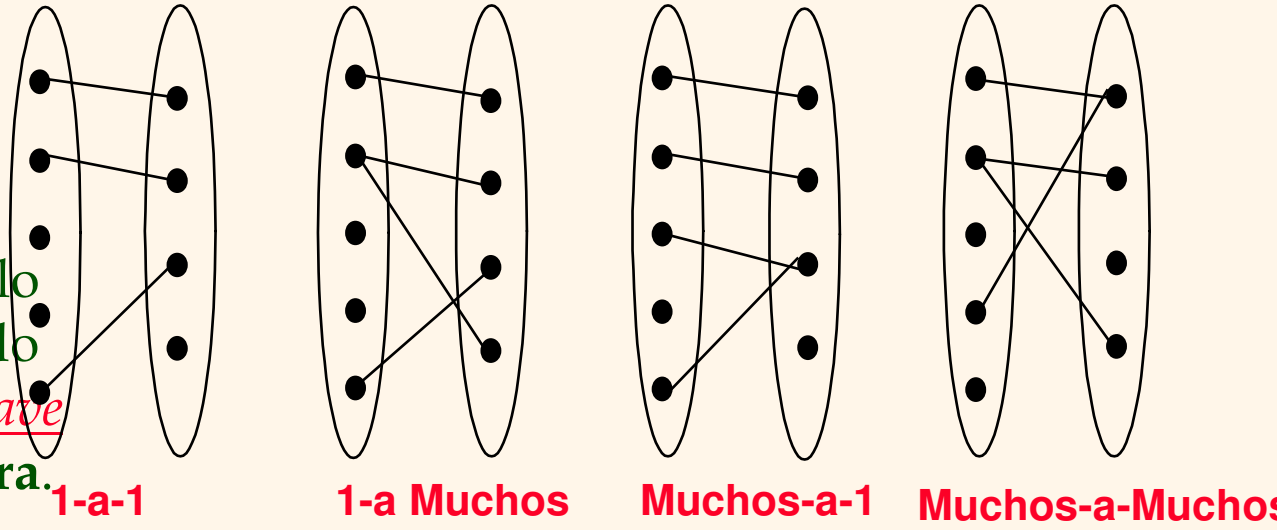


- *Relación*: Asociación entre dos o más entidades. Por ej. Pamela trabaja en el Departamento de Farmacia.
- *Conjunto de Relaciones*: Colección de relaciones similares.
  - Una conjunto de relaciones n-aria R relaciona n conjuntos de entidades E1 ,..., En; cada conjunto de entidades Ej en la relación R involucra alguna entidad de Ej.
  - El mismo conjunto de entidades puede participar en diferentes conjuntos de relaciones, o en diferentes "roles" en el mismo conjunto.

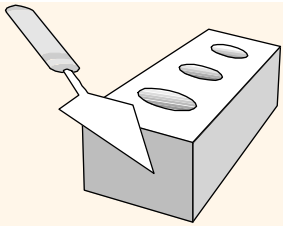
Restricciones de Llave



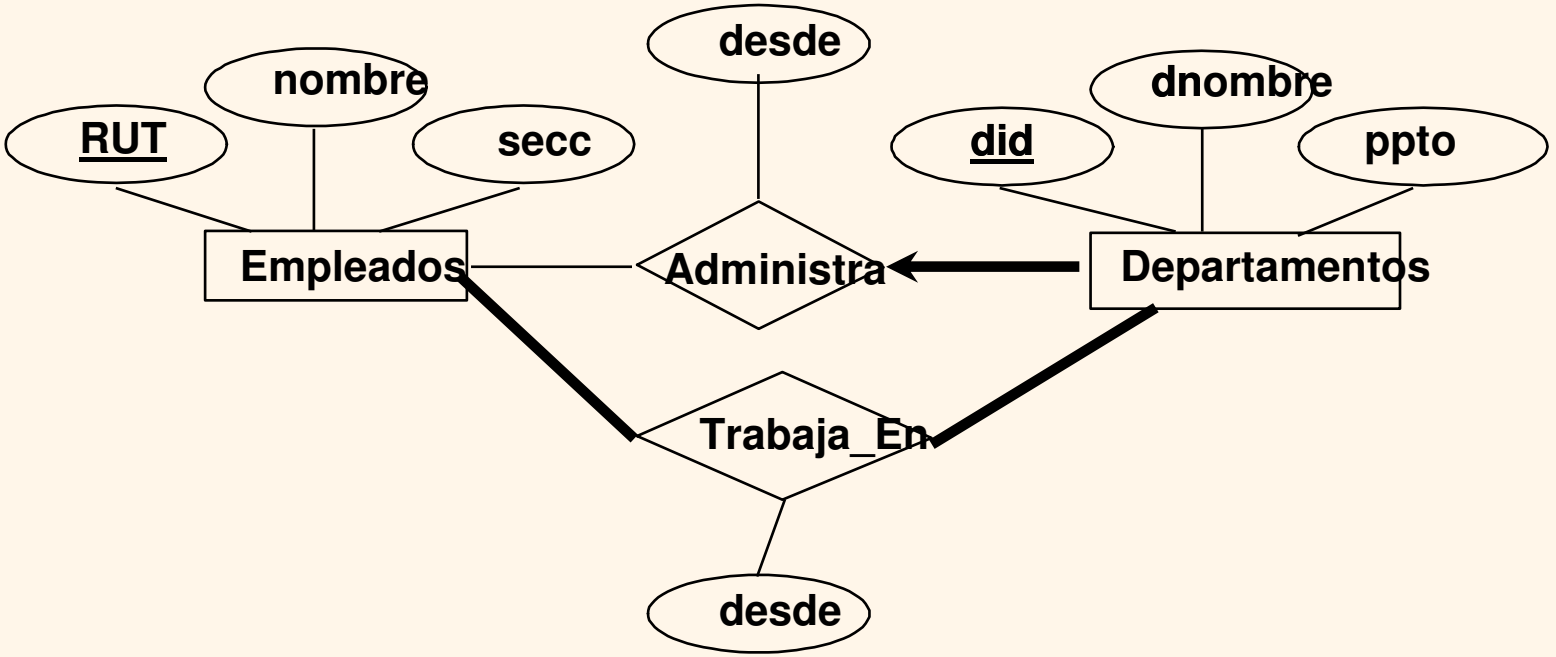
- Relac. **Trabaja_En**:
  - un empleado puede trabajar en muchos deptos;
  - un departamento puede tener muchos empleados.
- En contraste, cada depto. puede tener a lo más un jefe de acuerdo con la *restricción de llave* de la relac. **Administra**.

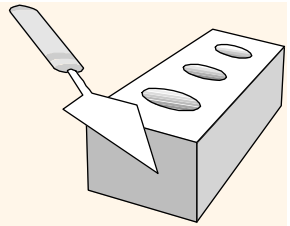


Restricciones de Participación



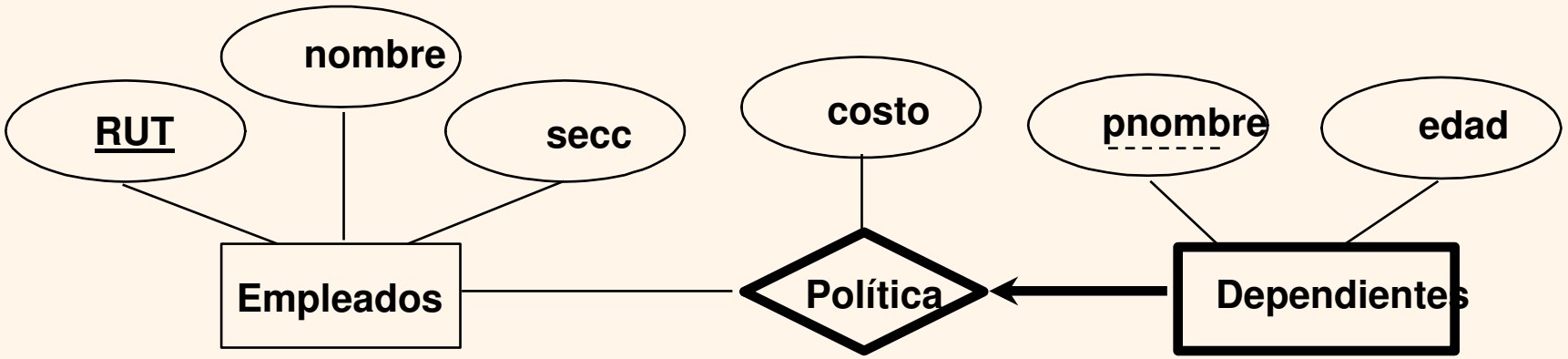
- ¿Tiene cada departamento un administrador?
  - *Restricción de participación*: la participación de Departamentos en Administra se dice *total* (resp. *parcial*) si si (resp. no).
    - Cada entidad de Departamento debe aparecer en una instancia de la relación Administra.



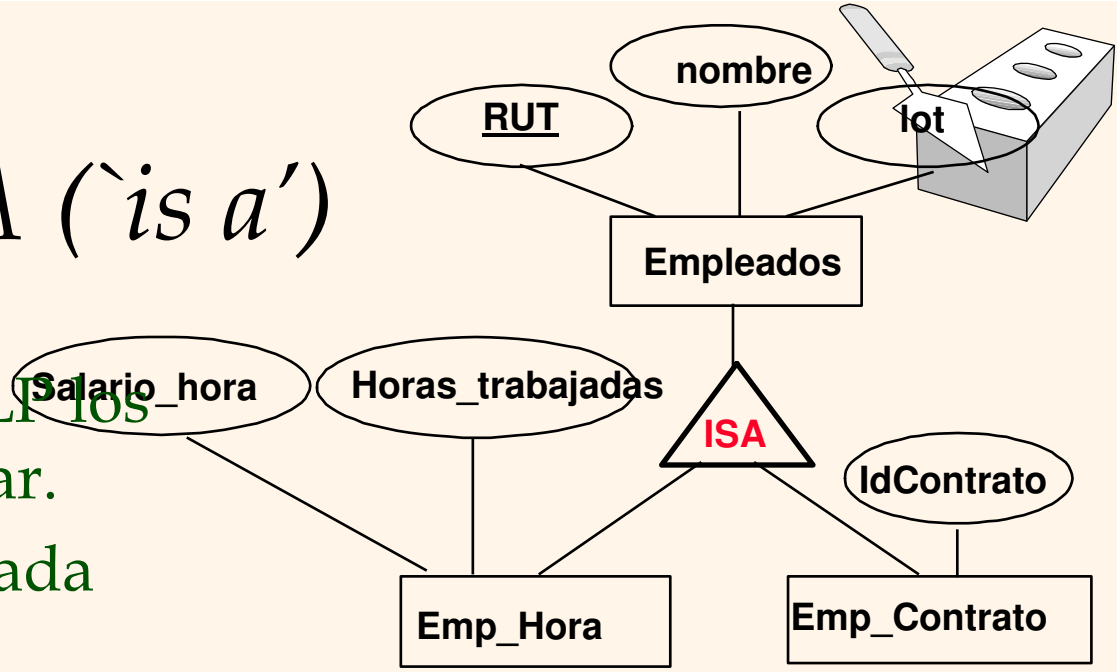


Entidades Débiles

- Una *entidad* puede ser identificada únicamente por medio de su llave más la llave de la entidad padre.
  - Un conjunto de Entidades padre y un conjunto de entidades débiles deben participar en un conjunto de relaciones uno-a-muchos (un padre, muchas entidadades débiles).
  - Un conjunto de entidades débiles debe tener participación total en este conjunto de relaciones *identificatorias.*



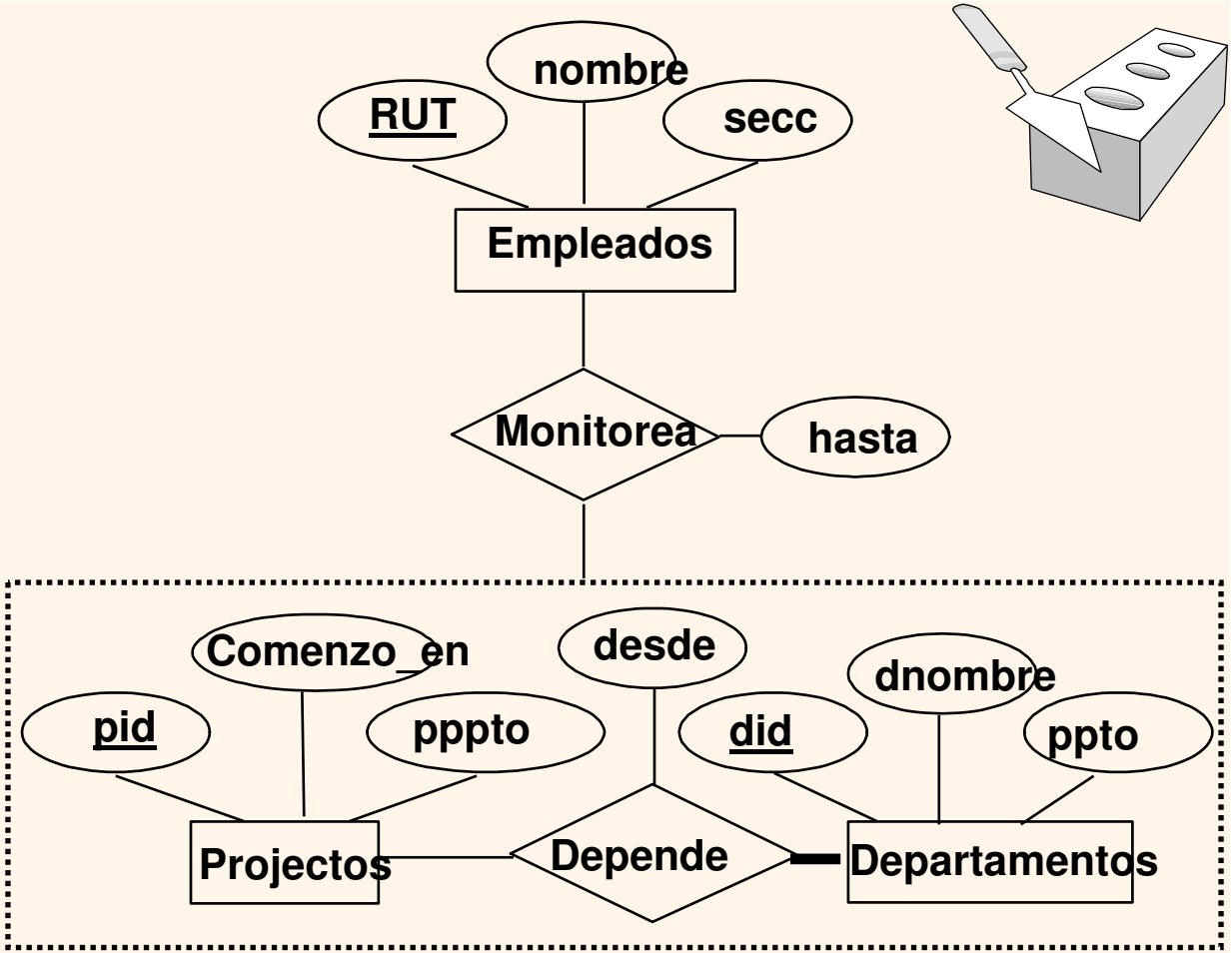
Jerarquías ISA (`is a')



- Como en JAVA u otros LP los atributos se pueden heredar.
- Si declaramos A **ISA** B, cada entidad de A es también considerada un entidad de B.
  - *Restricciones de sobreposición*: ¿Puede Juan ser un empleado por horas y a la vez un empleado por contrato? (*Permitido/No-permitido*)
  - *Restricciones de Cubrimiento*: ¿Tiene que ser cada entidad empleado un Emp_Hora or un Emp_Contrato? *(Si/no)*
  - Razones para usar ISA:
    - Agregrar atributos descriptivos específicos a una subclase.
    - Identificar entidades que participan en una relación.

Agregación

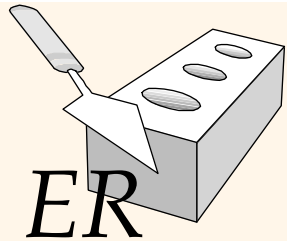
- Usada cuando tenemos un modelo de relación que involucra conj. de entidades y *conj. de relaciones*.
- *Agregación* permite tratar un conjunto de relaciones como un conjunto de entidades para propósitos de participación en otras relaciones.



* *Agregación vs. relaciones ternarias*:

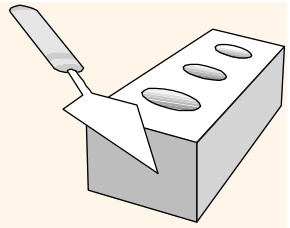
- Monitorea es una relación distinta con un atributo descriptivo.
- También, se puede decir que cada dependencia es monitoreada por a lo más un empleado.

Diseño Conceptual usando el Modelo ER



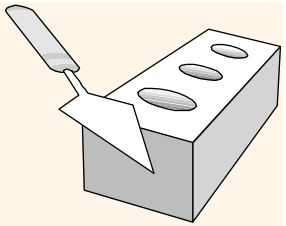
- Decisiones de Diseño:
  - ¿Debe un concepto ser modelado como entidad o como atributo?
  - ¿Debería un concepto ser modelado como entidad o como relación?
  - Identificando relaciones: Binarias or ternarias? Agregación?
- Restricciones en el Modelo ER:
  - Gran parte de la semántica de los datos puede (y debe) ser capturada.
  - Pero algunas restricciones no pueden ser capturadas en diagramas ER.

Entidad vs. Atributo

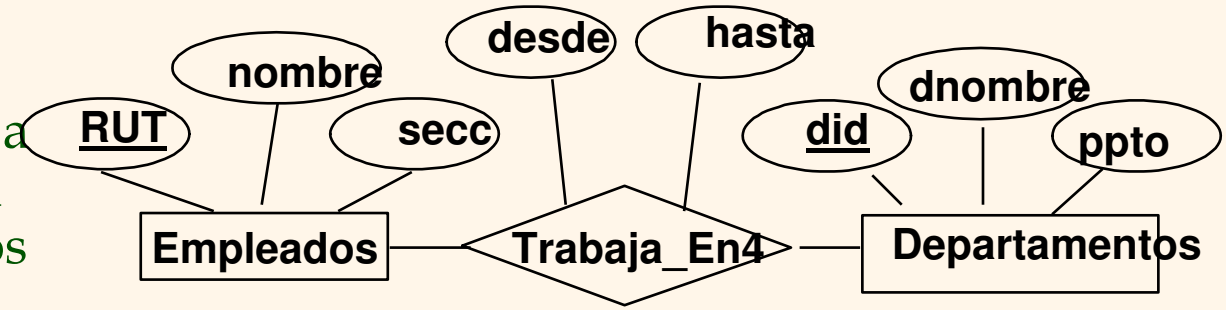


- Debiera ser *dirección* bun atributo de Empleados o de una entidad (conectada a Empleados por una relación)?
- Depende del uso que queramos darle a la información dirección y la semántica de los datos:
  - Si tenemos varias direcciones por empleado, dirección must be an entity (since attributes cannot 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).

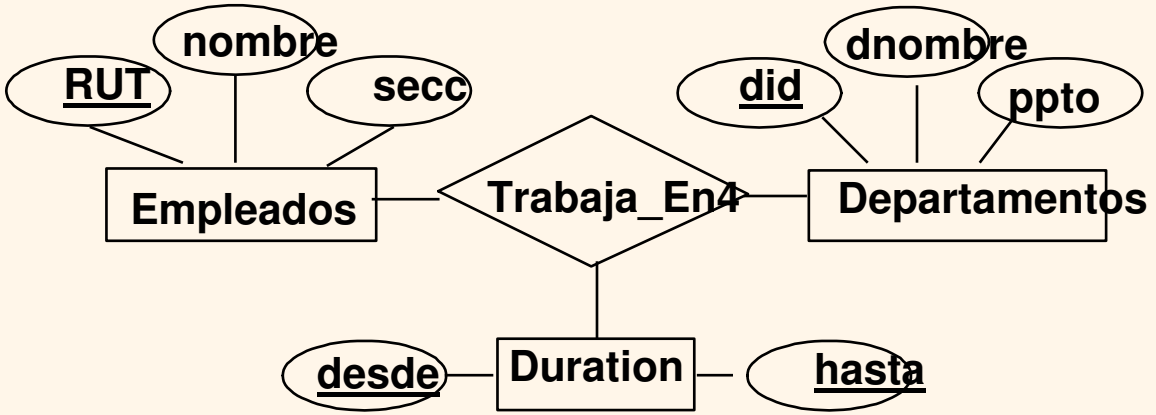
Entidad vs. Atributo (Cont.)

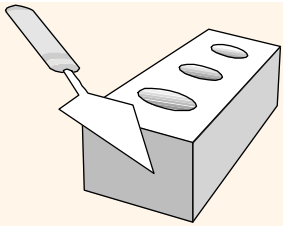


- Trabaja_En4 no permite a un empleado trabajar en un departamento por dos o más períodos.



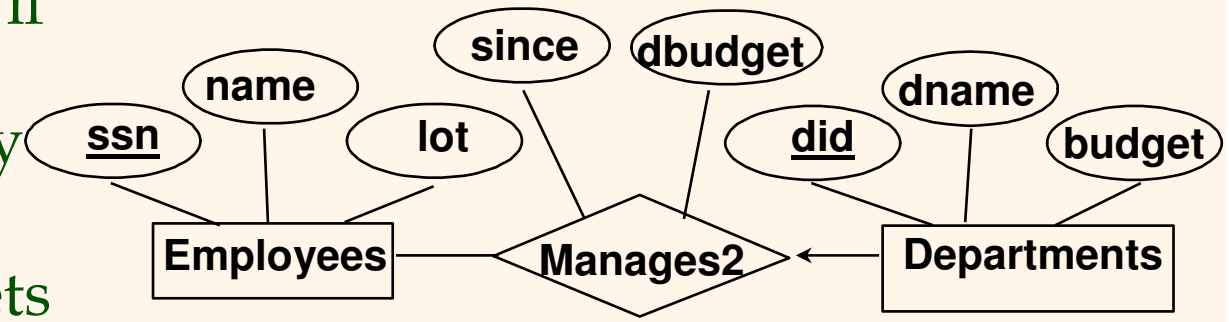
- El problema es similar al de quere almacenar varias direcciones para un empleado: queremos *svarios valores de los atributos descriptivos para cada instancia de esta relación*. Esto se logra introduciendo otro conjunto de entidades Duración.

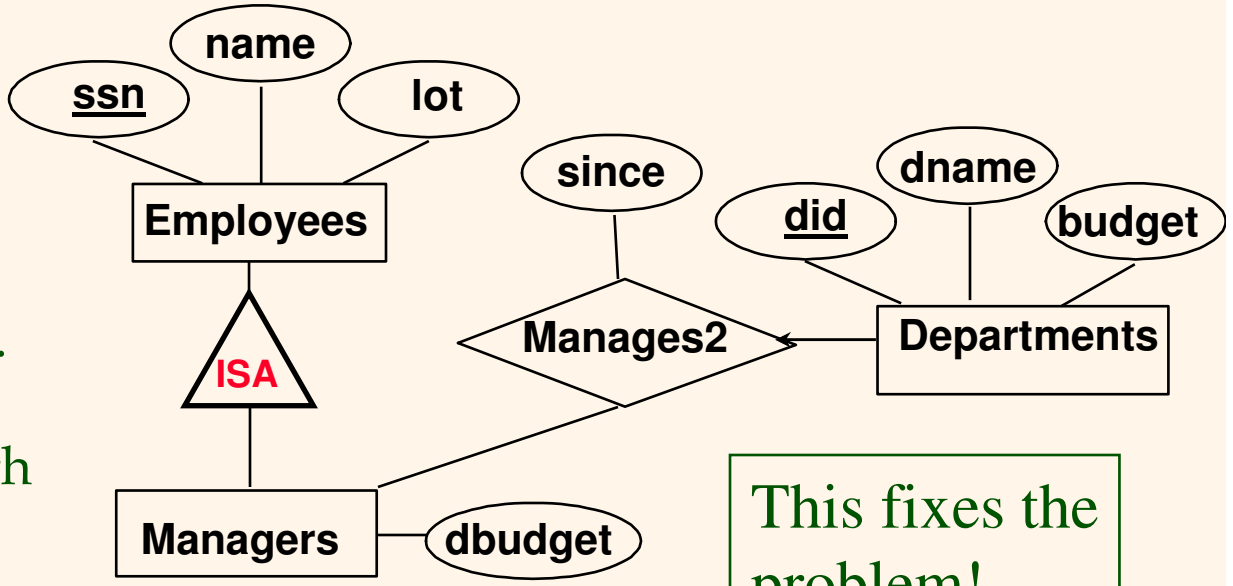




Entity vs. Relationship

- First 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.

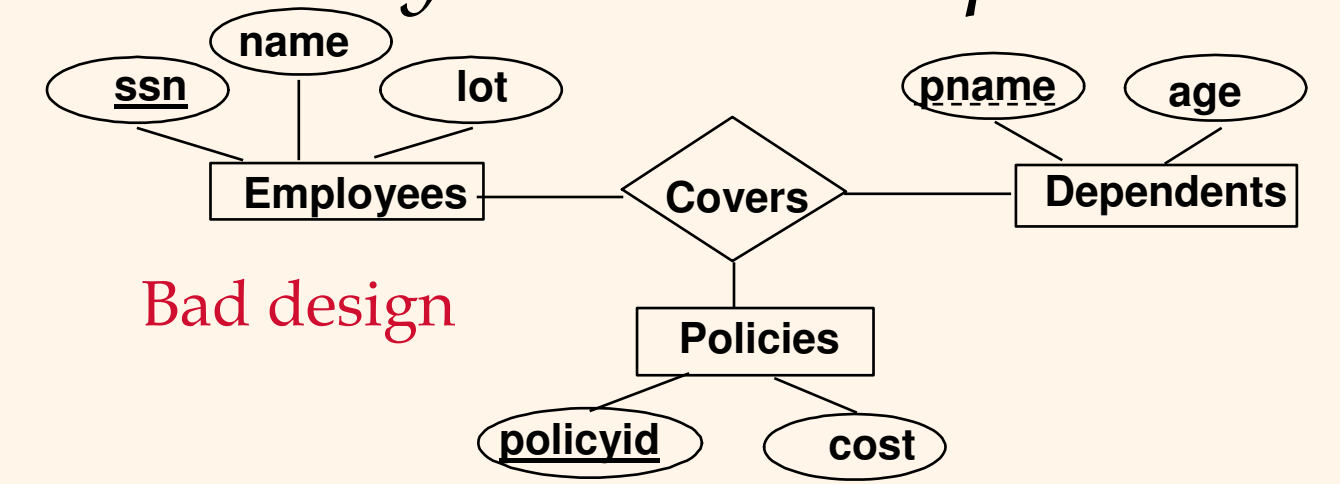




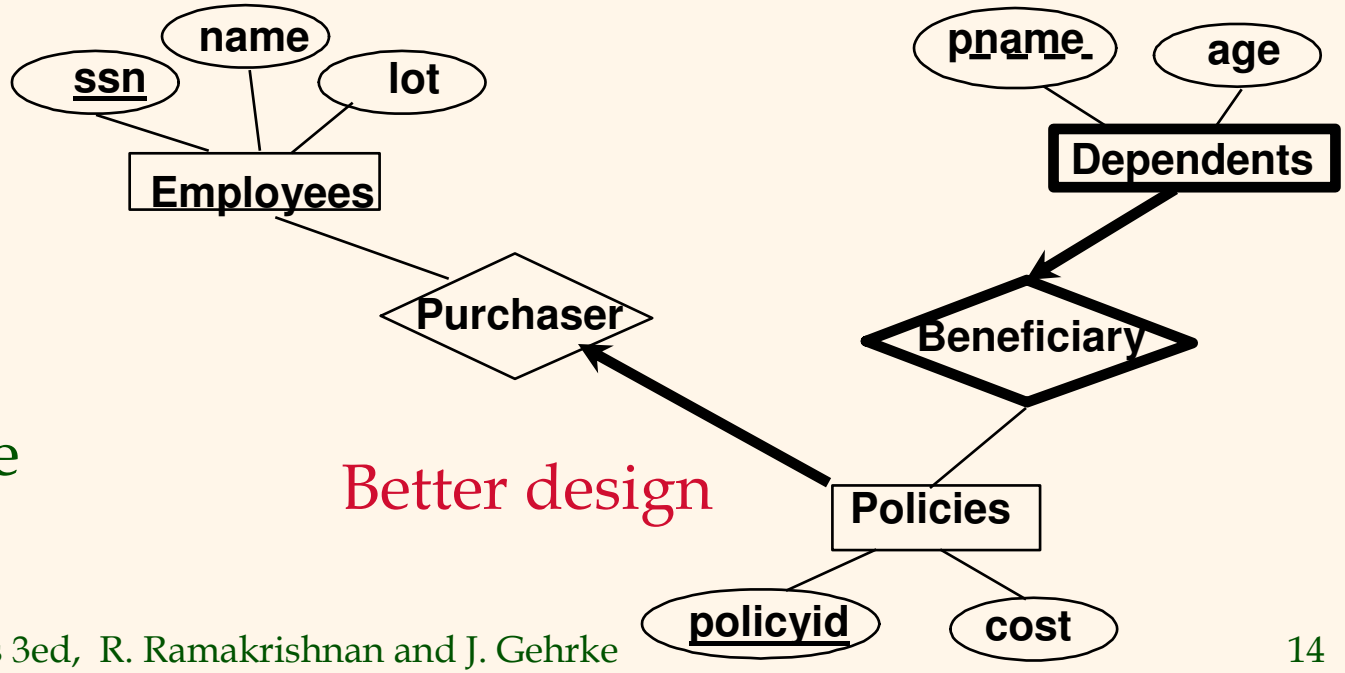
This fixes the problem!

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?

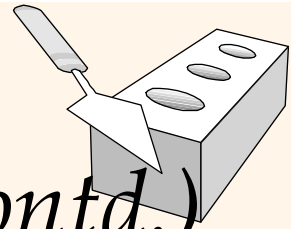


Bad design

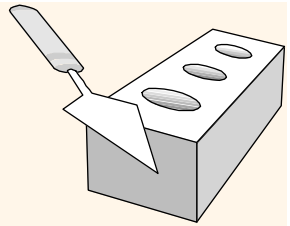


Better design

Binary vs. Ternary Relationships (Contd.)



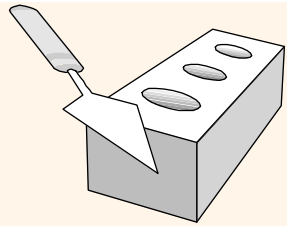
- Previous example illustrated a case when two binary relationships were better than one ternary relationship.
- An example in the other direction: a ternary relation Contracts relates entity sets Parts, Departments and Suppliers, and has descriptive attribute *qty*. No combination of binary relationships 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

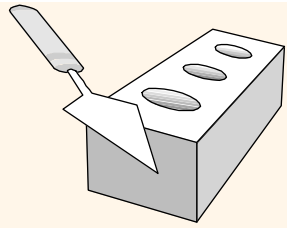
- *Conceptual design* follows *requirements analysis*,
  - Yields a high-level description of data to be stored
- ER model popular for conceptual design
  - Constructs are expressive, close to the way people think about their applications.
- Basic constructs: *entities*, *relationships*, and *attributes*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

Summary of ER (Contd.)



- Several kinds of integrity constraints can be expressed in the ER model: *key constraints*, *participation constraints*, and *overlap/covering constraints* for ISA hierarchies. Some *foreign key constraints* are also implicit in the definition of a relationship set.
  - 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 (Contd.)



- 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.
- Ensuring good database design: resulting relational schema should be analyzed and refined further. FD information and normalization techniques are especially useful.