ISYS2120 – Data & Information Management

Week 3B: Relational Design – From ER Diagram to Schema
(Kifer/Bernstein/Lewis - Chapter 3; Ramakrishnan/Gehrke - Chapter 3)

Based on slides from Kifer/Bernstein/Lewis (2006) “Database Systems”
and from Ramakrishnan/Gehrke (2003) “Database Management Systems”,
and including material from Fekete and Röhm.

Prof Alan Fekete

THE UNIVERSITY OF SYDNEY

Mapping E-R Diagrams into Relations

- Mapping rules for
  - Strong Entities
  - Weak Entities
  - Relationships
    - One-to-many, Many-to-many, One-to-one
    - Unary Relationships
    - Ternary Relationships

- We will concentrate in the lecture on typical examples…

Producing a relational schema from conceptual design

- A conceptual design (eg an E-R diagram) shows what facts about the real world should be kept, to support the organizational needs (including operational changes and analysis for reporting)
- To actually keep that information, the organisation will have a relational dbms which stores data structured in a relational schema
- It is essential, that *all* the necessary facts about *any* valid state of the world can be represented as an instance of the schema
- It is very desirable, that the schema has constraints that *enforce* properties that always hold in the domain
  - That is, one cannot represent in database, facts that could not possibly hold in the domain

Value of constraints

- Example: suppose we know that each supplier has a different SuppID
  - We want the dbms to reject any change that would include two suppliers with the same SuppID
- Example: suppose we know that each product comes from at most one supplier
  - We want dbms to reject any change that would give a second supplier for a given product
- Constraints in the schema prevent (some cases of) loss of data quality
  - They help make the database contents matching the real world

Overview of the process

- Look at entity sets and their attributes in ER diagram, and introduce tables that correspond
- Look at relationship sets in ER diagrams, and sometimes introduce extra tables to capture this information; in other cases, add extra columns to tables that were already introduced
- Make sure each entity set key information is reflected in primary key constraints
- Make sure each cardinality constraint from relationship set, is reflected in schema (sometimes by foreign key constraint, sometimes just in table structure with primary key information)
- Make sure each participation constraint from relationship set, is reflected in schema, usually by NOT NULL constraint

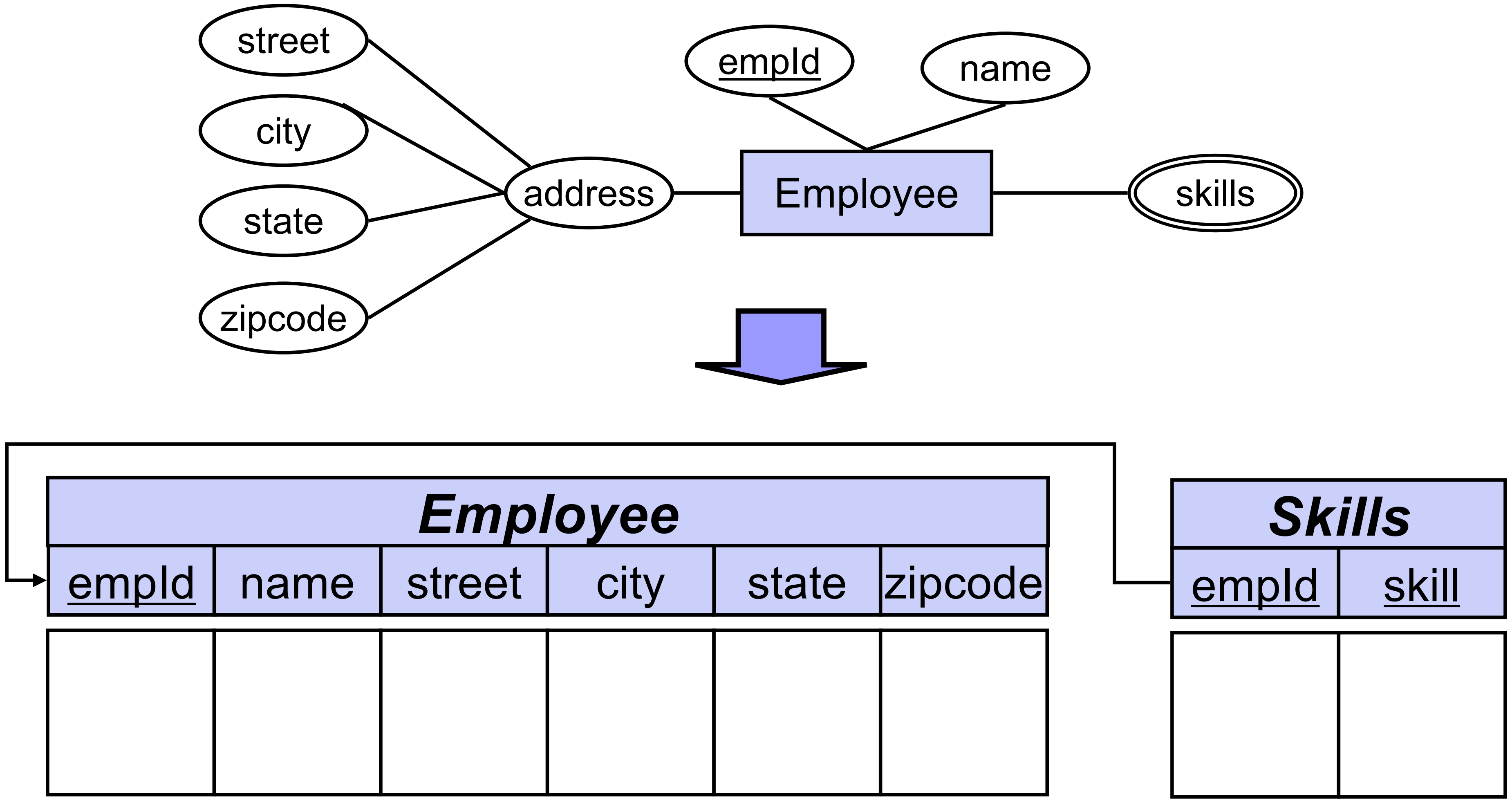
Mapping usual (strong) Entity Set to Tables

- Each **entity type** becomes a table
  - **Simple attributes**
    E-R simple attributes map directly onto columns in the table
    - Primary key of the table, is column(s) that correspond to primary key of the entity set
  - **Composite attributes**
    Composite attributes are flattened out by creating a separate column for each component attribute
    => We use only their simple, component attributes
  - **Multi-valued attribute**

    Do not include these in the table that corresponds to the entity set!
  - Instead, introduce a separate table, with one column giving one value of the attribute, and another column which is a foreign key referencing the entity's primary key
    - The primary key of this table, is the combination of all the columns!

Example: Mapping Entity Types

- Employee entity type with composite/multi-valued attributes



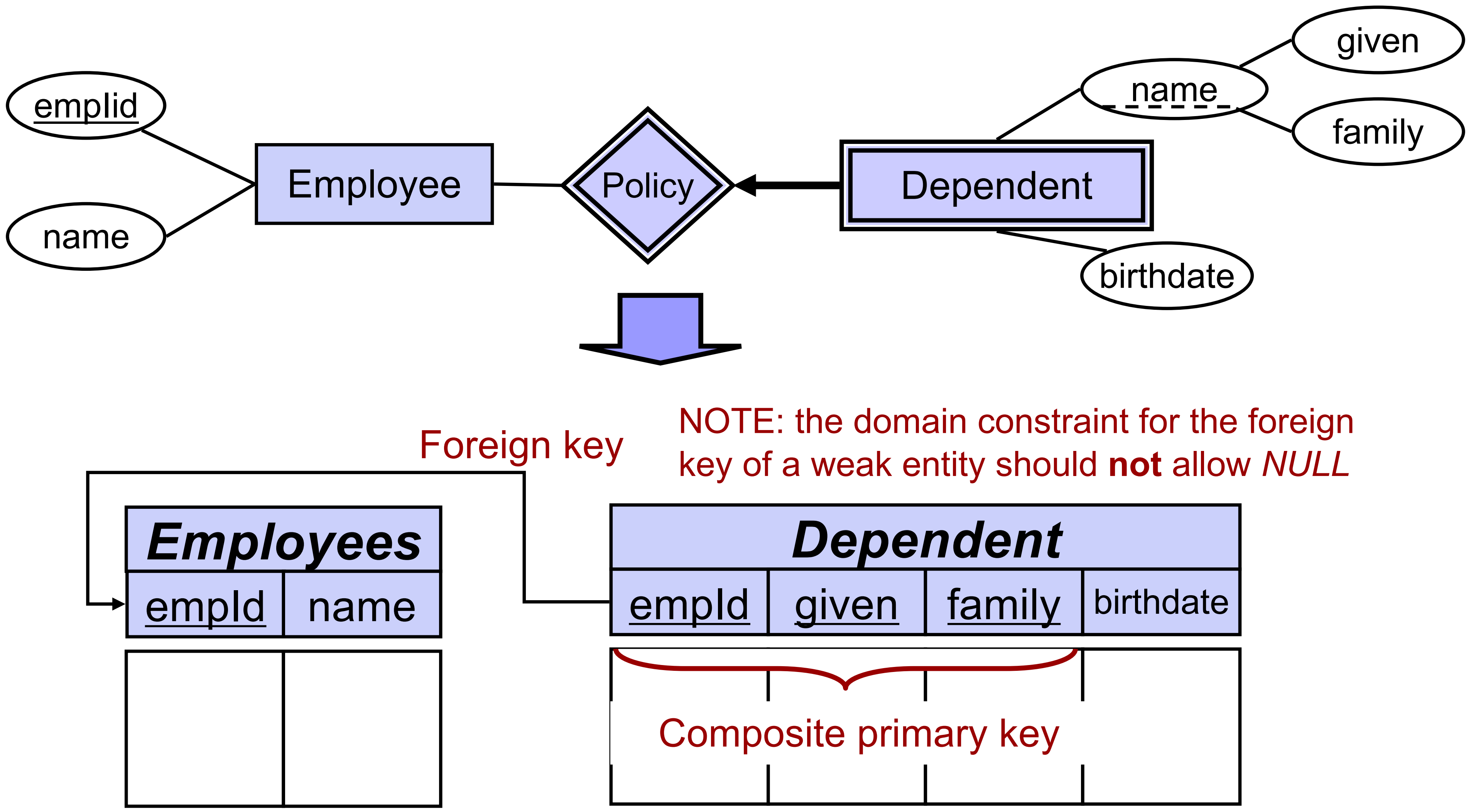
PK-/FK reference between Employee table and table for multi-valued Skills attribute.

Mapping of Weak Entity Types

- Each **weak entity type** becomes a separate table
  - introduce a separate table with columns for the attributes (as for strong entity set) and *also* a column which is a foreign key taken from the superior entity
  - primary key of this table is the combination with both
    - Partial key (discriminator) of weak entity
    - Foreign key column Primary key of identifying relation (strong entity)

Example: Mapping of Weak Entity

- Weak entity set ‘Dependent’ with composite partial key

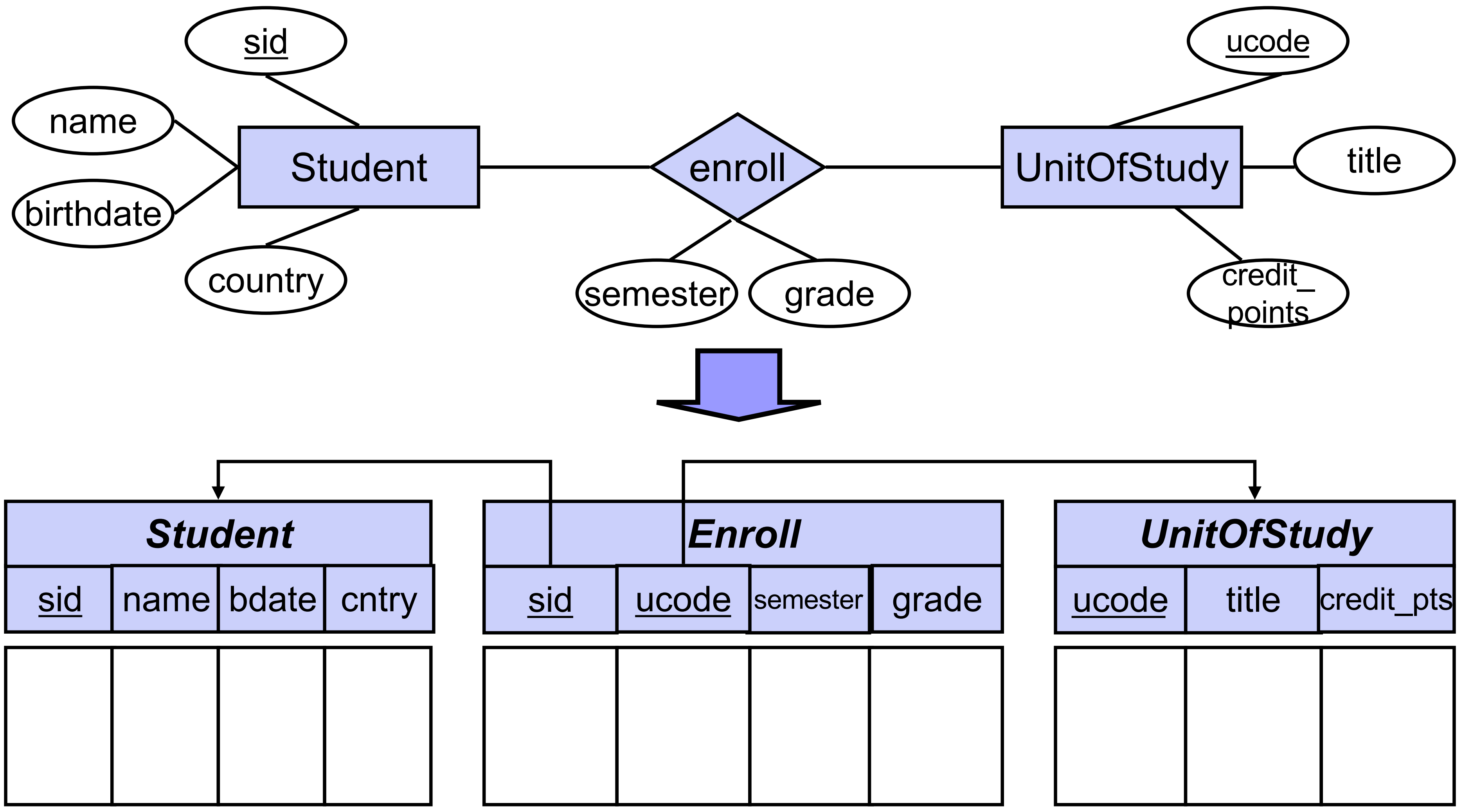


Mapping of Relationship Types

- There is a general technique (can be used for any relationship set)
- Introduce an extra table with columns that are foreign keys (referencing the primary keys of the entity types involved), and also columns for any attributes on the relationship set itself
  - Name of table can be name of relationship type, or sometimes, use the combination of entity set names
  - Constraints in table depend on those in relationship type
  - Eg, if an entity type has cardinality that an entity is involved in at most one relationship, this means the primary key of new table is just the foreign key for that entity type
    - Otherwise, primary key of new table will be the combination of all the columns which are foreign keys referencing involved entity types

Example: Mapping of Many-to-Many Relationship Type

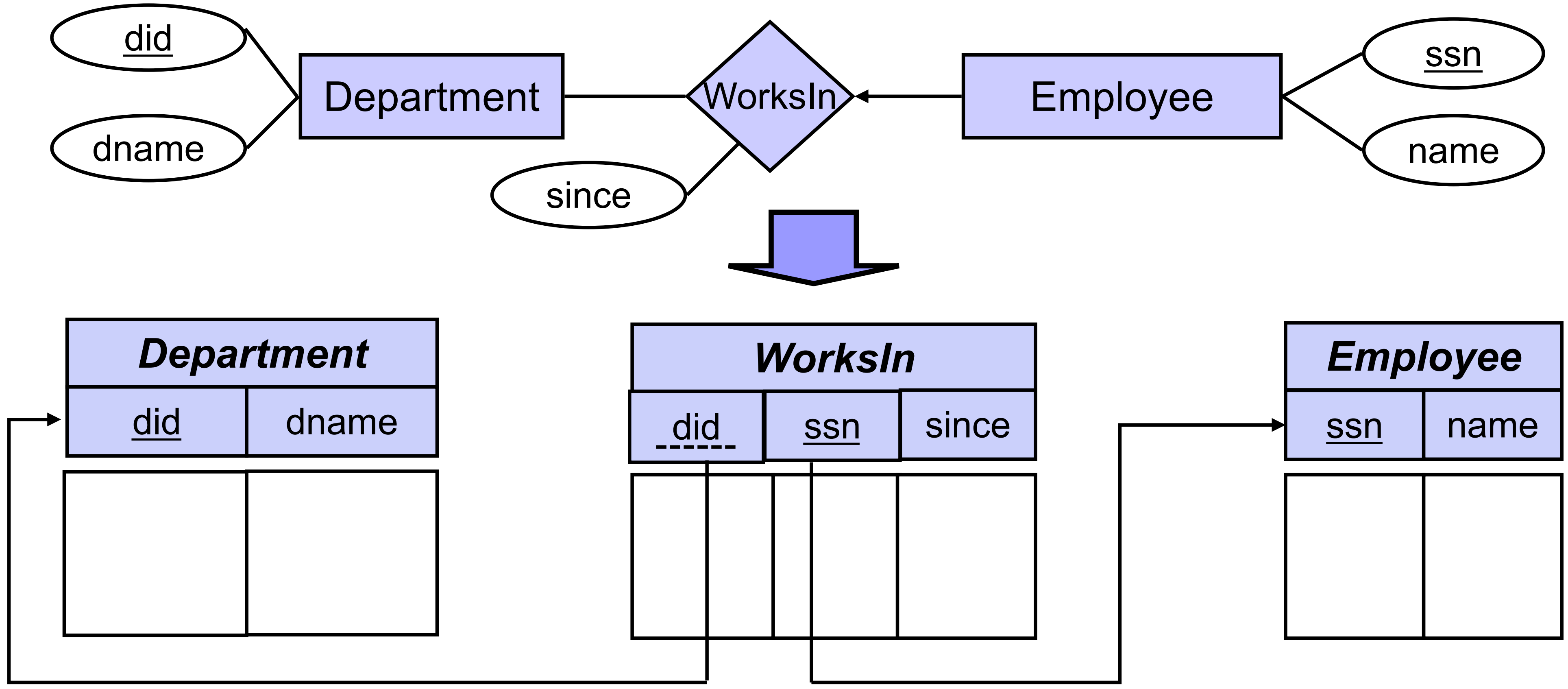
- Many-to-many relationship between Student & UnitOfStudy



Alternative name for “join table” could be “Student_Unit”

Example: Mapping of Many-to-One Relationship

- Many Employees to One Department



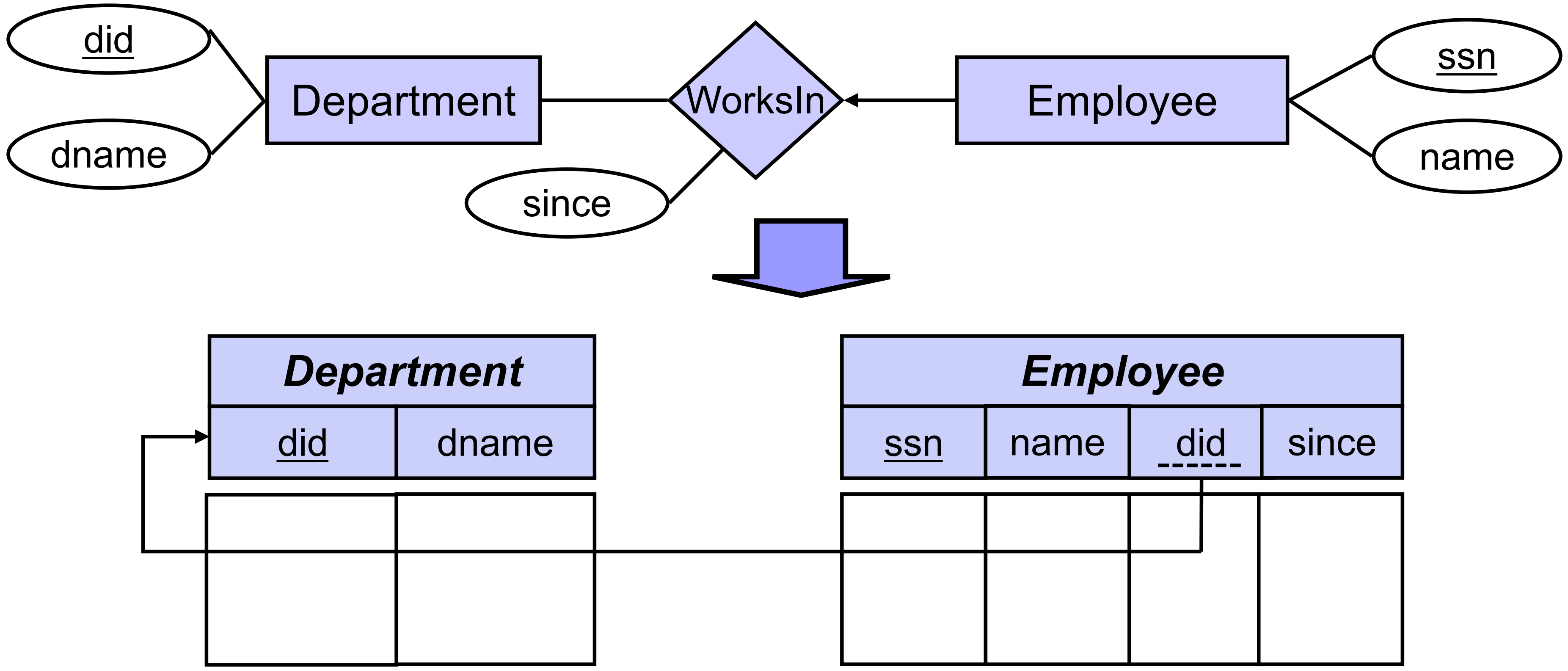
Alternative name for “join table” could be “Employee_Department”

Alternative Mapping of Relationship Types

- In cases of many-to-one relationship, there is a different way to map this to the relational schema
- Modify the schema of table that corresponds to the many side, to introduce one or more extra columns
  - foreign key referencing the “one” side, and (if needed) other columns for the attributes of the relationship set
- Primary key of this table stays as before
- If there is also total participation constraint for many side: foreign key of other side is declared NOT NULL

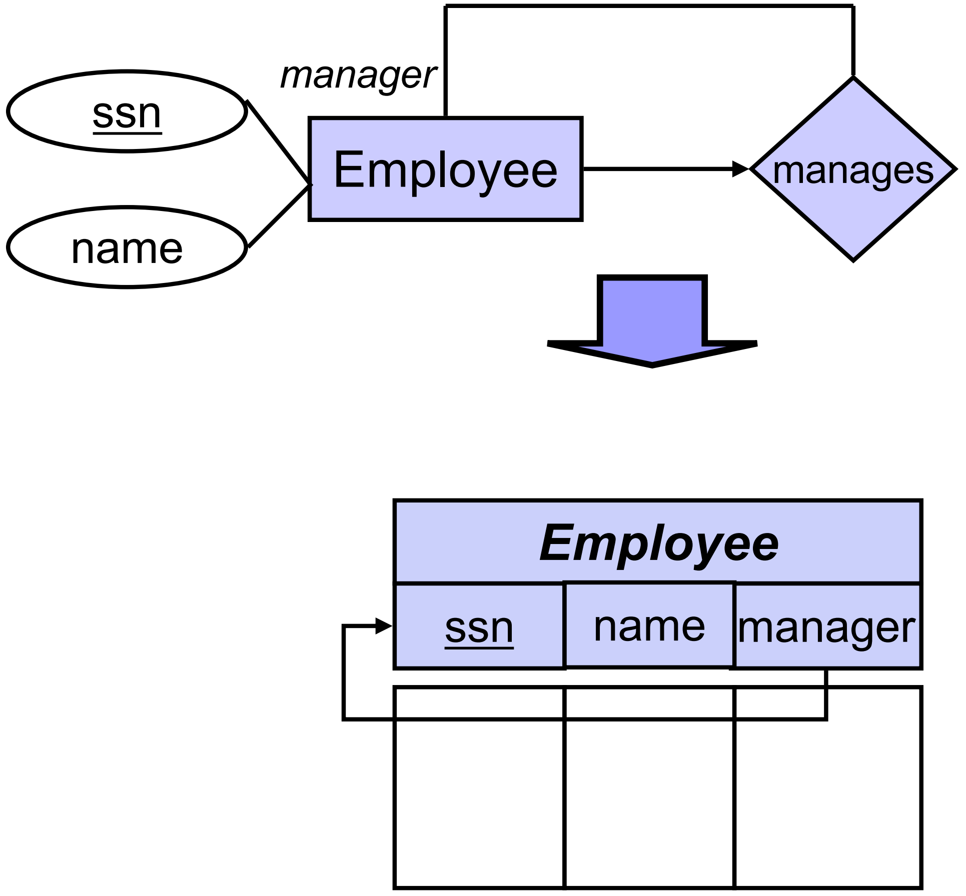
Example: Mapping of One-to-Many Relationships

- Many Employees to One Department

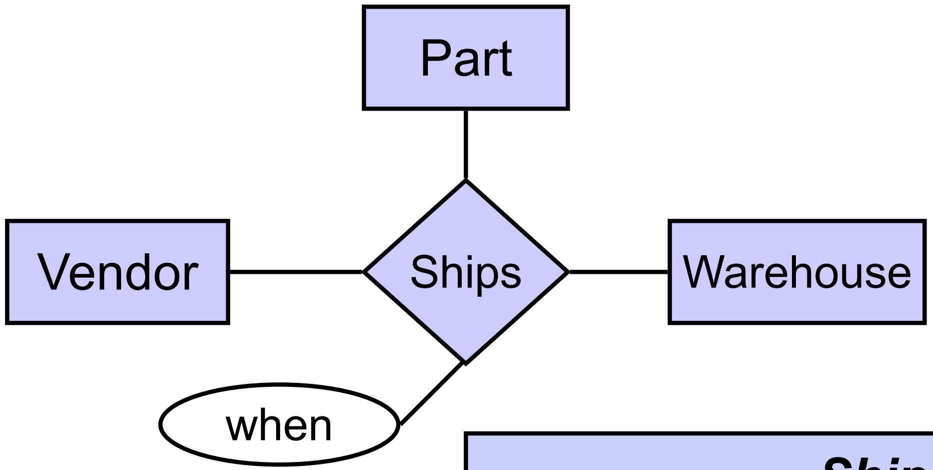


Examples: mapping of recursive relationship

- Many Employees to One Department



Example: mapping of ternary relationship



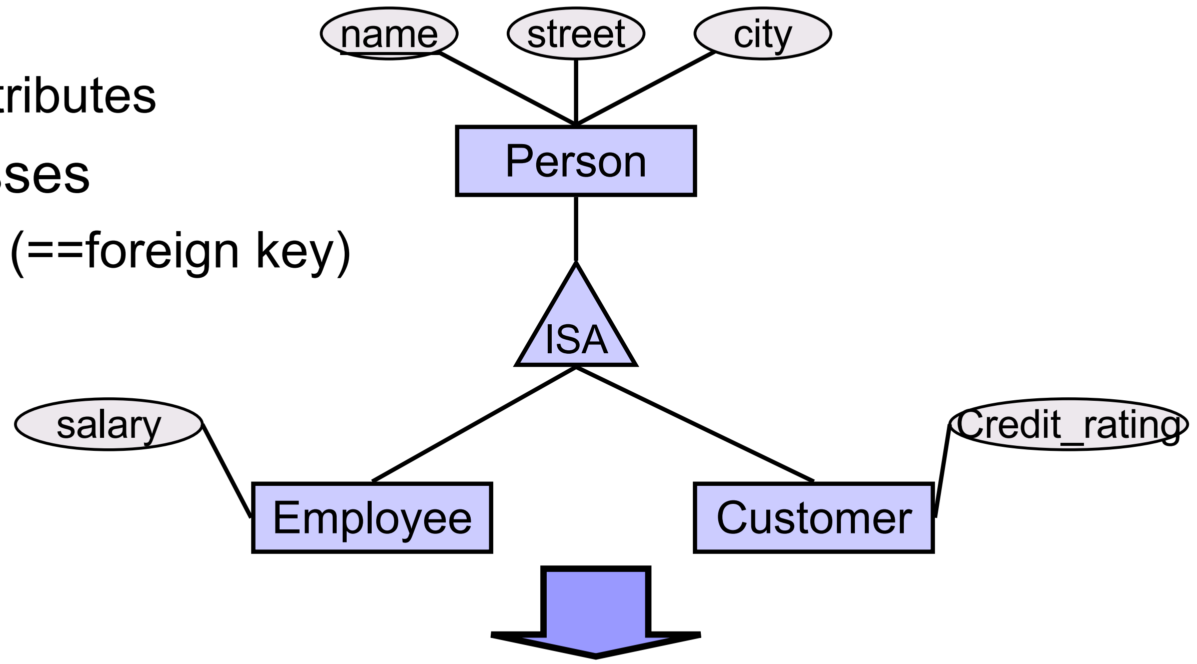
Ships			
PartID	VendorID	WarehouseID	when

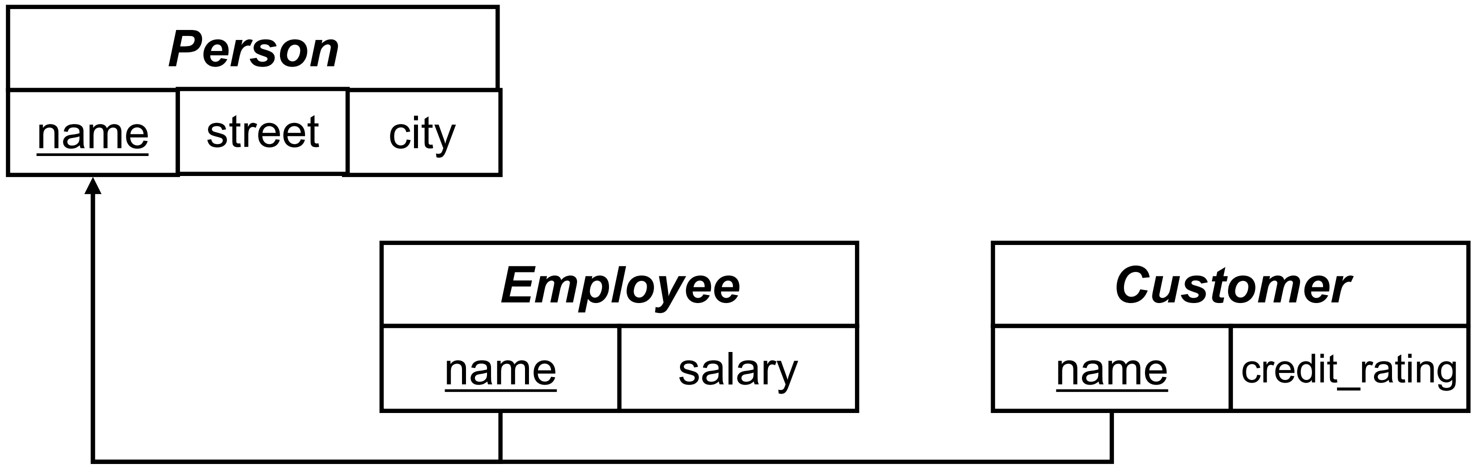
Mapping of ISA-Hierarchies

- Standard way (works always):
  - Distinct relations for the superclass and for each subclass
  - Superclass attributes (including key and possible relationships) go into superclass relation
  - Subclass attributes go into each sub-relation; primary key of superclass relation also included as primary key of subclass relation
  - 1:1 relationship established between superclass and each subclass, with superclass as primary table
    - I.e. primary keys of subclass relations are also a foreign key referencing the superclass relation
- Special case (can be used in presence of a **total** covering constraint):
  - Distinct relations for each subclass with primary key of superclass
  - All superclass attributes are included in each subclass relation
  - No foreign-keys needed

Mapping of ISA-Hierarchy

- Table for superclass
  - Primary key + common attributes
- Separate tables for subclasses
  - Primary key of superclass (==foreign key) + specific attributes





Summary: Schema Correspondence with E-R Model

- Tables correspond to entity types (entity sets) and to many-to-many relationship types/sets and to muti-valued attributes
  - And sometimes to other relationship types/sets,
- A single row correspond with an entity, or a relationship
- Columns correspond with attributes or many-to-one relationships.
- Primary key constraints reflect primary key (identifier) information in entity sets (for a table corresponding to the entity set)
  - In a table corresponding to many-many relationship, the primary key combines the keys of the entity sets involved

Take care

- The word relation (in relational database) is NOT the same as the word relationship (in E-R model)
- In ER diagram, no entity set should ever have attributes that are keys of other entity sets
  - These connections should be shown in separate relationship sets
  - But in relational schema, tables can (and often do) include columns which are foreign keys referencing other tables

References

- Kifer/Bernstein/Lewis (2nd edition)
  - Chapter 3
- Ramakrishnan/Gehrke (3rd edition - the ‘Cow’ book)
  - Chapter 3.1-3.4 and 3.6-3.7, plus Chapter 1.5
- Ullman/Widom (3rd edition)
  - Chapter 2.1 - 2.3, Section 7.1 and Chapter 8.1-8.2
  - *views and foreign keys come later, instead relational algebra is introduced very early on; also briefly compares RDM with XML,*
- Silberschatz/Korth/Sudarshan (5th edition - ‘sailing boat’)
  - Chapter 2.1 - 2.2; Chapter 3.1-3.2 and 3.9
  - *starts with relational algebra early on which we do later*
- Elmasri/Navathe (5th edition)
  - Chapter 2.1-2.3; Chapter 5; Section 8.8
  - *talks first more about system architectures and conceptual modeling*