Chapter 5 – Week6

Entity Relationship Model Continued

Announcements

- Lab3
  - On Teams on Monday 4:30pm
  - Due Friday October 22nd
- Assignment 1 – due Oct 23rd

Wednesday – October 20th

Map ER Model to Relational model

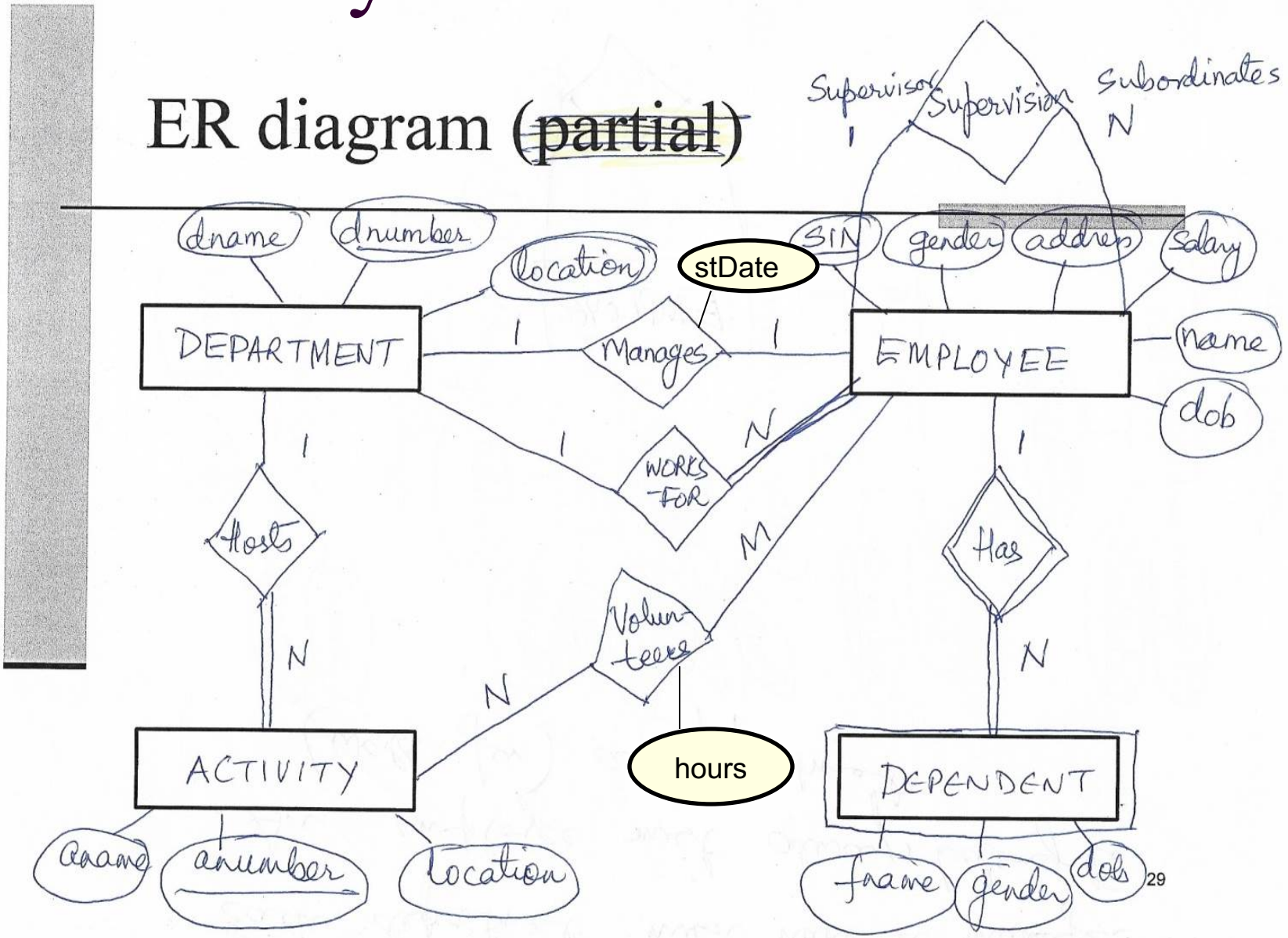
Input: ER Model

Output: Relational Model

General Idea:

- Each entity type (ET) becomes a relation.
- Only the simple components of any composite attribute are taken.
- Each 1:1 and 1:N relationship adds an attribute (as foreign key) to an existing ET.
- Each M:N relationship becomes a new relation
- Each multi-valued attribute becomes a new relation

University Staff DB ER model



ER-to-Relational Mapping :Step 1

For each regular (strong) entity type E in the ER schema, create a relation R that includes all the simple attributes of E. Include only the simple component attributes of a composite attribute. Choose one of the key attributes of E as primary key for R. If the chosen key of E is composite, the set of simple attributes that form it will together form the primary key of R.

ER-to-Relational Mapping :Step 2

- For each weak entity type W in the ER schema with owner entity type E, create a relation R, and include all simple attributes (or simple components of composite attributes) of W as attributes of R. In addition, include as foreign key attributes of R the primary key attribute(s) of the relation(s) that correspond to the owner entity type(s); this takes care of the identifying relationship type of W. The primary key of R is the combination of the primary key(s) of the owner(s) and the partial key of the weak entity type W, if any.

ER-to-Relational Mapping :Step 3

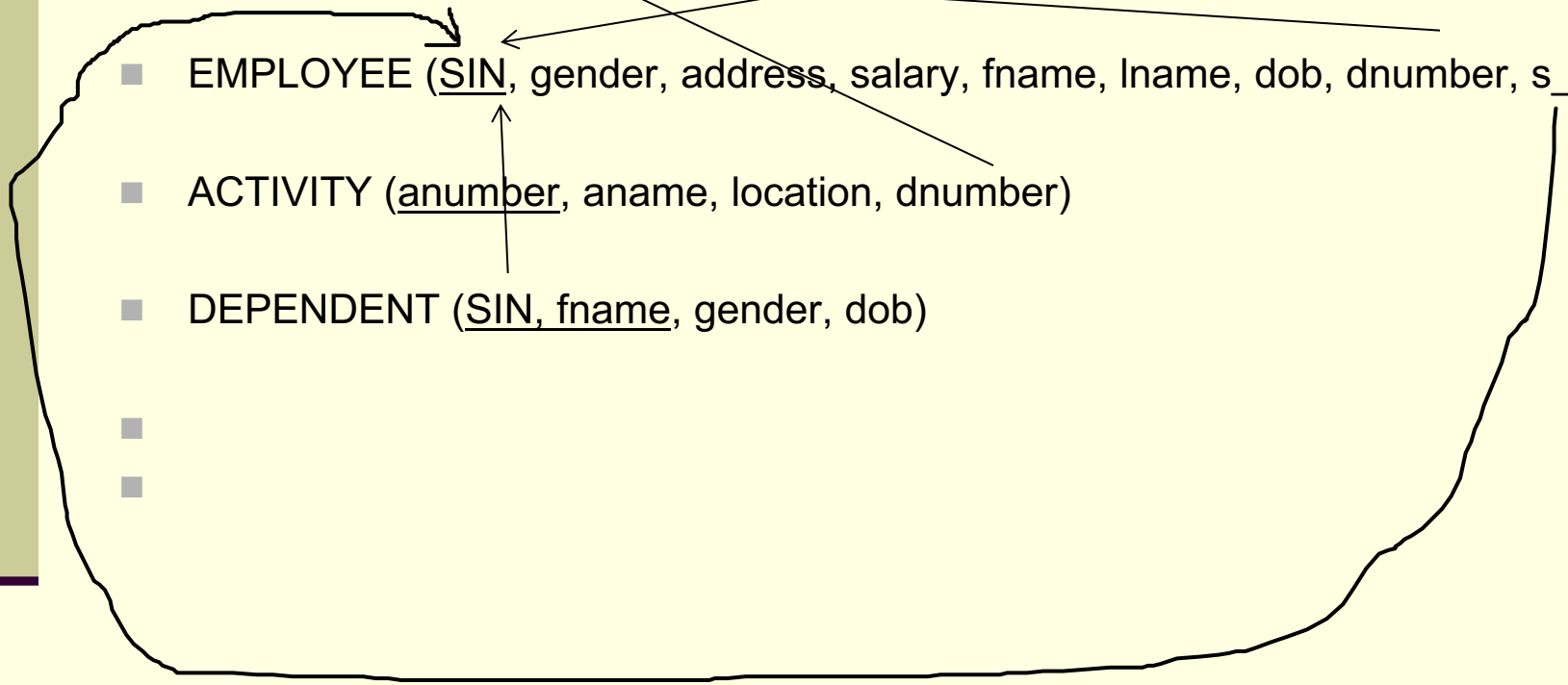
- For each binary 1:1 relationship type R in the ER schema, identify the relations S and T that correspond to the entity types participating in R. Choose one of the relations—S, say—and include as foreign key in S the primary key of T. It is better to choose an entity type with *total participation* in R in the role of S. Include all the simple attributes (or simple components of composite attributes) of the 1:1 relationship type R as attributes of S.

ER-to-Relational Mapping :Step 4

- For each regular binary 1:N relationship type R, identify the relation S that represents the participating entity type at the *N-side* of the relationship type. Include as foreign key in S the primary key of the relation T that represents the other entity type participating in R; this is because each entity instance on the N-side is related to at most one entity instance on the 1-side of the relationship type. Include any simple attributes (or simple components of composite attributes) of the 1:N relationship type as attributes of S.

Relational model of University staff database – so far

- DEPARTMENT (dnumber, dname, SIN)
- EMPLOYEE (SIN, gender, address, salary, fname, lname, dob, dnumber, s_sin)
- ACTIVITY (anumber, aname, location, dnumber)
- DEPENDENT (SIN, fname, gender, dob)



ER-to-Relational Mapping :Step 5

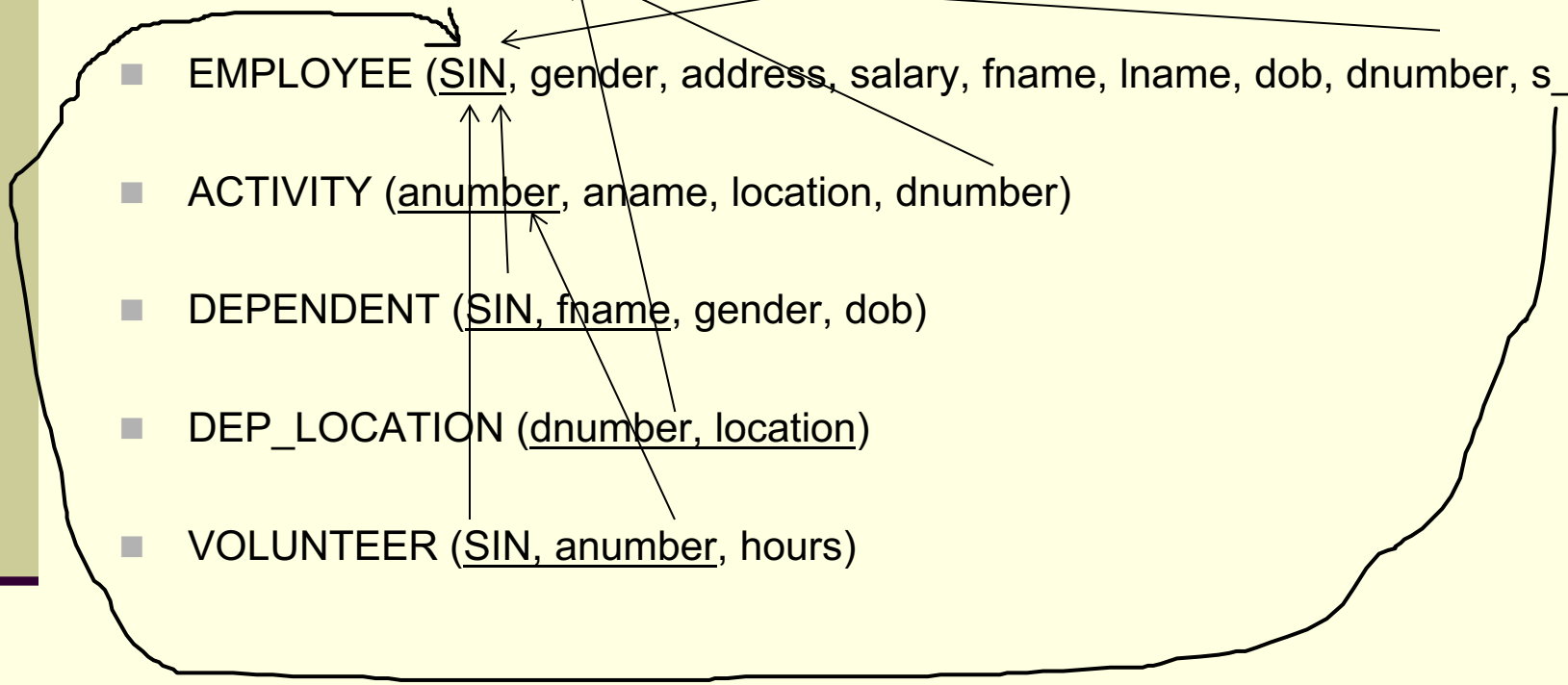
- For each binary M:N relationship type R, create a new relation S to represent R. Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types; their combination will form the primary key of S. Also include any simple attributes of the M:N relationship type (or simple components of composite attributes) as attributes of S. Notice that we cannot represent an M:N relationship type by a single foreign key attribute in one of the participating relations—as we did for 1:1 or 1:N relationship types—because of the M:N cardinality ratio.

ER-to-Relational Mapping :Step 6

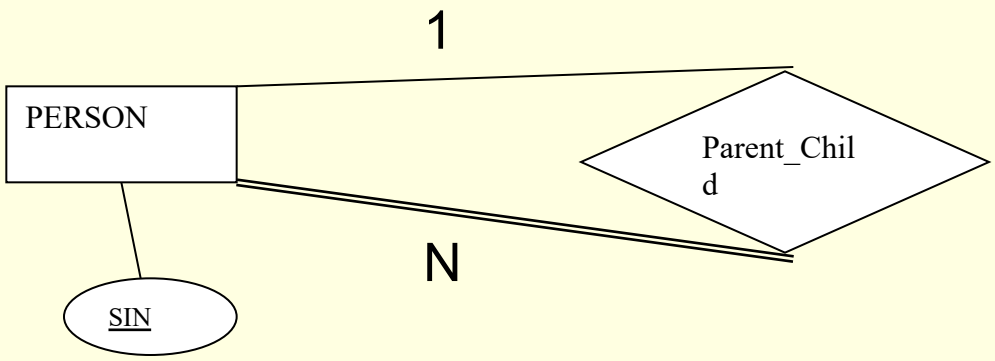
- For each multivalued attribute A, create a new relation R. This relation R will include an attribute corresponding to A, plus the primary key attribute K—as a foreign key in R—of the relation that represents the entity type or relationship type that has A as an attribute. The primary key of R is the combination of A and K. If the multivalued attribute is composite, we include its simple components

Relational model of University staff database

- DEPARTMENT (dnumber, dname, SIN)
- EMPLOYEE (SIN, gender, address, salary, fname, lname, dob, dnumber, s_sin)
- ACTIVITY (anumber, aname, location, dnumber)
- DEPENDENT (SIN, fname, gender, dob)
- DEP_LOCATION (dnumber, location)
- VOLUNTEER (SIN, anumber, hours)



Example : ER – to – Relational



Reverse process: from Relational to ER

S (SNo, SName, Status, City)

P (PNo, PName, Colour, Weight, City)

SP (SNo, PNo, Qty)

Reverse process: from Relational to ER (assume implicit meaning of course and section)

- COURSE(**CRS_CODE**, CRS_DESCRIPTION, CRS_CREDIT)
- SECTION (**CRS_CODE, SECTION#**, YEAR, SEM, CLASS_TIME, ROOM_CODE, PROF_NUM)

- SECTION cannot exist unless it has a FK CRS_CODE that points to an existing COURSE row. But this condition means that SECTION is existence-dependent on COURSE.
- Incidentally, in this example SECTION is also a weak ENTITY to COURSE. That's because a weak entity is defined as one that inherits at least part of its PK from the (parent) COURSE entity *and* it is existence-dependent on the (parent) COURSE entity. (Note that there are *two* requirements that must be met before an entity can be classified as weak).

Reverse process: from Relational to ER

Convert to ER Model:

- COURSE(**CRS_CODE**, CRS_DESCRIPTION, CRS_CREDIT)
- SECTION (**CLASS_CODE**, CRS_CODE, SECTION#, CLASS_TIME, ROOM_CODE, PROF_NUM)

Example (Contd).

- COURSE(**CRS_CODE**, CRS_DESCRIPTION, CRS_CREDIT)
- SECTION (**CLASS_CODE**, CRS_CODE, SECTION#, CLASS_TIME, ROOM_CODE, PROF_NUM)

- CLASS_CODE is a surrogate key
- Let's assume that the CRS_CODE FK in the SECTION is declared to be "not null."

- The SECTION entity is existence-dependent on COURSE. (That's because it is reasonable to assume that COURSE is mandatory to SECTION, since a section does not appear in the class schedule unless there is a course description for it in the course catalog. Note that the "not null" FK requirement means that the CRS_CODE FK in the CLASS entity is mandatory.)

- SECTION is not a weak entity (That's because although the SECTION is existence-dependent on COURSE -- the SECTION entity's PK does not contain the PK of the COURSE entity.)