EXERCISE

277

Exercise

- The Prescriptions-R-X chain of pharmacies was recently “ghosted” by their “loyal” DBA. Their recent “hire” needs help to verify existing conceptual diagram. He wants varied perspectives so that he could debug the design and has gathered information as follows:
  - Patients are identified by an SSN, and their names, addresses, and ages must be recorded.
  - Doctors are identified by an SSN. For each doctor, the name, specialty, and years of experience must be recorded.
  - Each pharmaceutical company is identified by name and has a phone number.
  - For each drug, the trade name and formula must be recorded. Each drug is sold by a given pharmaceutical company, and the trade name identifies a drug uniquely from among the products of that company. If a pharmaceutical company is deleted, you need not keep track of its products any longer.

278

Furthermore…

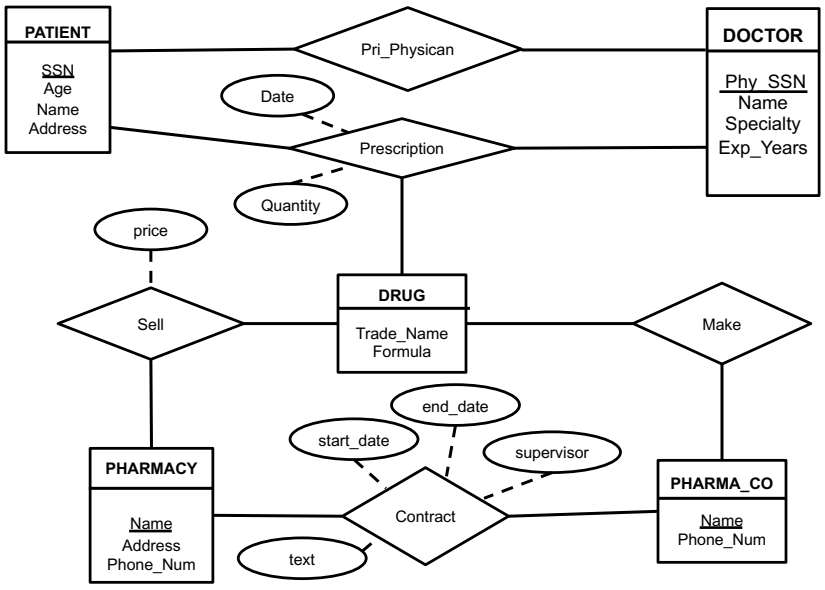
- Each pharmacy has a name, address, and phone number.
- Every patient has a primary physician.
- Every doctor has at least one patient.
- Each pharmacy sells several drugs and has a price for each. A drug could be sold at several pharmacies, and the price could vary from one pharmacy to another.
- Doctors prescribe drugs for patients. A doctor could prescribe one or more drugs for several patients, and a patient could obtain prescriptions from several doctors. Each prescription has a date and a quantity associated with it. You can assume that, if a doctor prescribes the same drug for the same patient more than once, only the last such prescription needs to be stored.

279

Finally…

- Pharmaceutical companies have long-term contracts with pharmacies. A pharmaceutical company can contract with several pharmacies, and a pharmacy can contract with several pharmaceutical companies. For each contract, you have to store a start date, an end date, and the text of the contract.
- Pharmacies appoint a supervisor for each contract. There must always be a supervisor for each contract, but the contract supervisor can change over the lifetime of the contract.

280

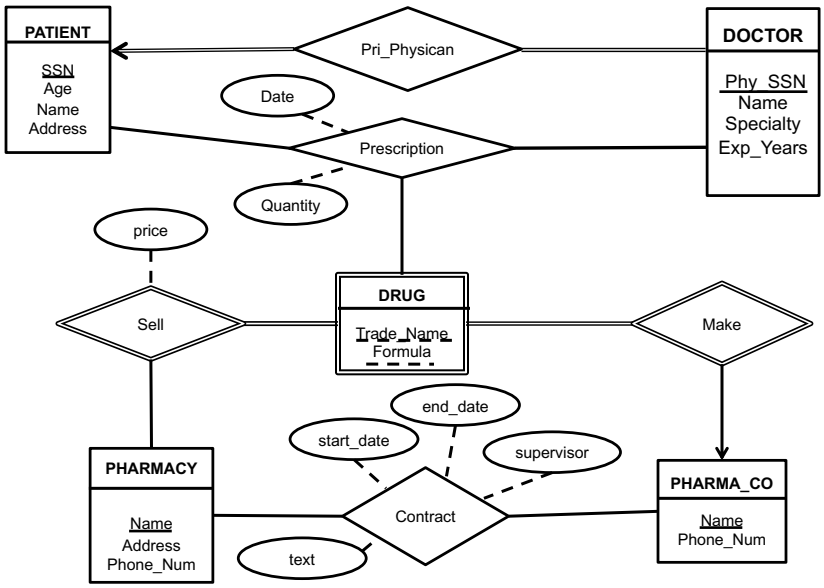


283

Important Relationships

- For each drug, the trade name and formula must be recorded. Each drug is sold by a given pharmaceutical company, and the **trade name identifies a drug uniquely** from among the products of that company. If a pharmaceutical company is deleted, you need not keep track of its products any longer.
- Each pharmacy has a name, address, and phone number.
- Every patient has a primary physician.
- Every doctor has at least one patient.

284



285

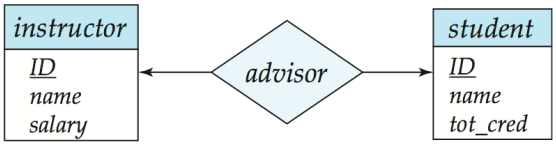
CSC 430/530: DBMS/DT

Lecture 10: Extended ER

288

Cardinality Constraints

- We express cardinality constraints by drawing either a directed line (→), signifying "one," or an undirected line (—), signifying "many," between the relationship set and the entity set.
- **One to One** relationship between an *instructor* and a *student* :
  - A student is associated with at most one *instructor* via the relationship *advisor*
  - A *student* is associated with at most one *department* via *stud_dept*

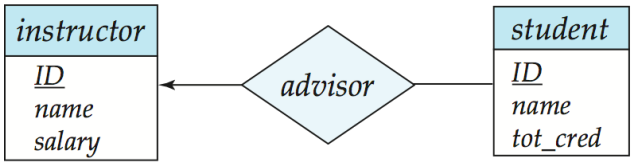


INSTRUCTOR(I_ID, NAME, SALARY)
STUDENT (S_ID, NAME, TOT_CRED, I_ID)

289

One-to-Many Relationship

- one-to-many relationship between an *instructor* and a *student*
  - an instructor advises (including 0) students
  - a student is advised by (at most) one instructor



INSTRUCTOR (I_ID, NAME, SALARY)
STUDENT (S_ID, NAME, TOT_CRED, I_ID)

290

Many-to-One Relationships

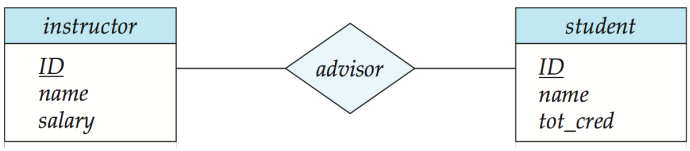
- In a many-to-one relationship between an *instructor* and a *student,*
  - an instructor advises just (at most) one student,
  - and a student is advised by (including 0) instructors

INSTRUCTOR (I_ID, NAME, SALARY, **S_ID**)
STUDENT (S_ID, NAME, TOT_CRED)

291

Many-to-Many Relationship

- An instructor advises several (possibly 0) students via *advisor*
- A student is advised by several (possibly 0) instructors via *advisor*

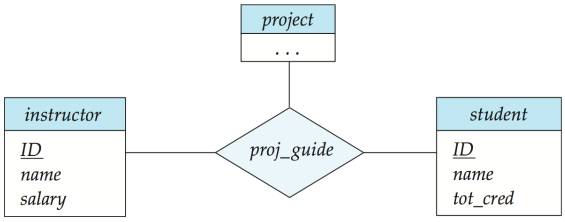


INSTRUCTOR (I_ID, NAME, SALARY)
ADVISOR(**I_ID, S_ID**)
STUDENT (S_ID, NAME, TOT_CRED)

292

Non-binary Relationship Sets

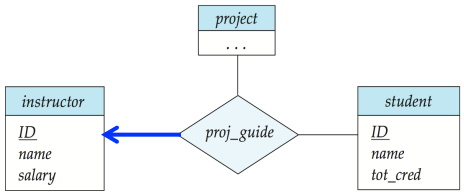
- Most relationship sets are binary
- There are occasions when it is more convenient to represent relationships as non-binary.
- E-R Diagram with a Ternary Relationship



293

Challenge: Cardinality Constraints on Ternary Relationship

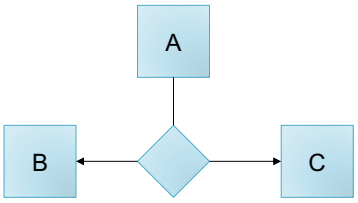
- We allow at most one arrow out of a ternary (or greater degree) relationship to indicate a cardinality constraint
- For example, an arrow from *proj_guide* to *instructor* indicates each student has at most one guide for a project
- If there is more than one arrow, there are two ways of defining the meaning.



294

Challenge: Cardinality Constraints on Ternary Relationship

- For example, a ternary relationship *R* between *A*, *B* and *C* with arrows to *B* and *C* could mean
  1. Each *A* entity is associated with a unique entity from *B* and *C*
     or
  2. Each pair of entities from (*A, B*) is associated with a unique *C* entity, and each pair (*A, C*) is associated with a unique *B*
- Each alternative has been used in different formalisms
- **To avoid confusion, we outlaw more than one arrow**



295

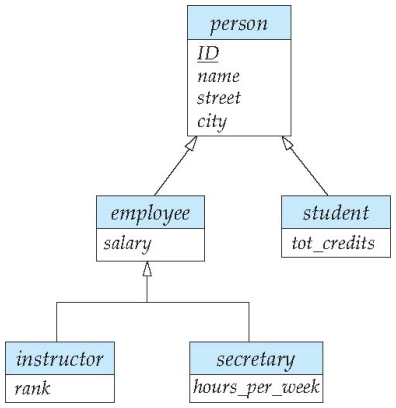
Specialization

- Top-down design process; we designate sub-groupings within an entity set that are distinctive from other entities in the set.
- These sub-groupings become **lower-level entity** sets that have attributes or participate in relationships that do not apply to the higher-level entity set.
- Depicted by a *triangle* component labeled ISA (e.g., *instructor* "is a" *person*).
- **Attribute inheritance** – a lower-level entity set inherits all the attributes and relationship participation of the higher-level entity set to which it is linked.

296

Specialization Example

- **Overlapping** – *employee* and *student*
- **Disjoint** – *instructor* and *secretary*
- Constraint: Total and partial



297

Representing Specialization via Schemas

- Method 1:
  - Form a schema for the higher-level entity
  - Form a schema for each lower-level entity set, include primary key of higher-level entity set and local attributes

schema	attributes
person	ID, name, street, city
student	ID, tot_cred
employee	ID, salary

Drawback: getting information about, an *employee* requires accessing two relations, the one corresponding to the low-level schema and the one corresponding to the high-level schema

298

Representing Specialization as Schemas (Cont.)

- Method 2:
  - Form a schema for each entity set with all local and inherited attributes

schema	attributes
person	ID, name, street, city
student	ID, name, street, city, tot_cred
employee	ID, name, street, city, salary

Drawback: *name, street* and *city* may be stored redundantly for people who are both students and employees

299

Generalization

- **A bottom-up design process** – combine a few entity sets that share the same features into a higher-level entity set.
- Specialization and generalization are simple inversions of each other; they are represented in an E-R diagram in the same way.
- The terms specialization and generalization are used interchangeably.

300

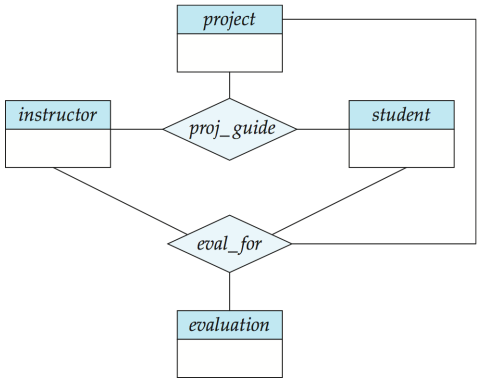
Design Constraints on a Specialization/Generalization

- **Completeness constraint** -- specifies whether an entity in the higher-level entity set must belong to at least one of the lower-level entity sets within a generalization.
  - **total**: an entity must belong to one of the lower-level entity sets
  - **partial**: an entity need not belong to one of the lower-level entity sets
  - **Partial generalization** is the default.

301

Aggregation

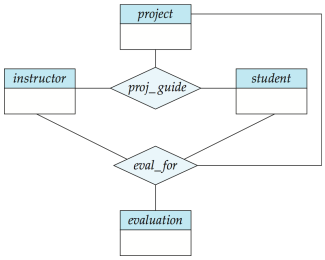
- Consider the ternary relationship *proj_guide*, which we saw earlier
- Suppose we want to record evaluations of a student by a guide on a project



302

Aggregation (Cont.)

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

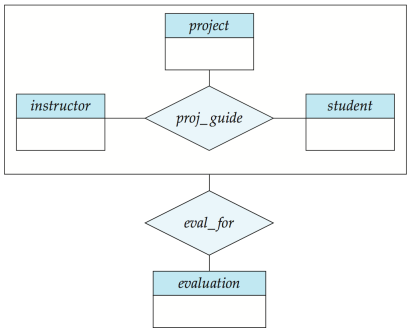


- Eliminate this redundancy via *aggregation*
  - Treat relationship as an abstract entity
  - Allows relationships between relationships
  - Abstraction of relationship into new entity

303

Aggregation (Cont.)

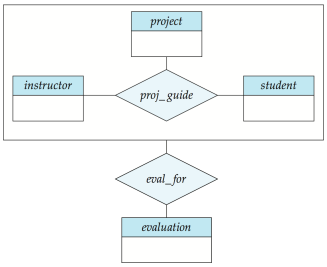
- Eliminate this redundancy via *aggregation* without introducing redundancy, the following diagram represents:
  - A student is guided by a particular instructor on a particular project
  - A student, instructor, project combination may have an associated evaluation



304

Representing Aggregation via Schemas

- To represent aggregation, create a schema containing
  - Primary key of the aggregated relationship,
  - The primary key of the associated entity set
  - Any descriptive attributes



- In our example:
  - The schema *eval_for* is:

    eval_for (*s_ID, project_id, i_ID, evaluation_id*)
  - The schema *proj_guide* is redundant.

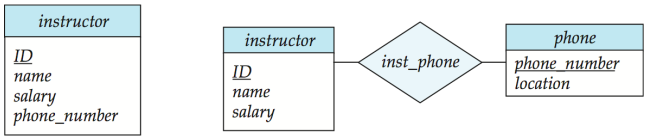
305

DESIGN ISSUES

306

Entities vs. Attributes

- Use of entity sets vs. attributes



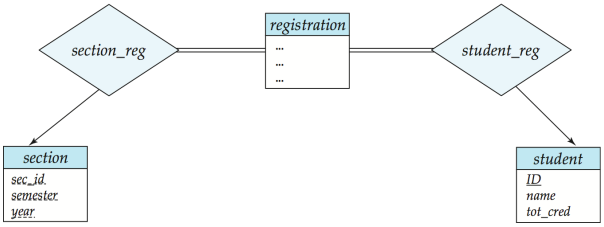
- Use of phone as an entity allows extra information about phone numbers (plus multiple phone numbers)

307

Entities vs. Relationship sets

- **Use of entity sets vs. relationship sets**

  Possible guideline is to designate a relationship set to describe an action that occurs between entities



- **Placement of relationship attributes**

  For example, attribute date as attribute of advisor or as attribute of student

308

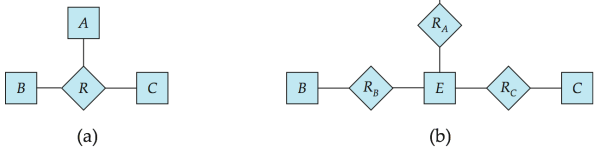
Binary Vs. Non-Binary Relationships

- Although it is possible to replace any non-binary (*n*-ary, for $n > 2$) relationship set by a number of distinct binary relationship sets, a *n*-ary relationship set shows more clearly that several entities participate in a single relationship.
- Some relationships that appear to be non-binary may be better represented using binary relationships
  - For example, a ternary relationship *parents*, relating a child to his/her father and mother, is best replaced by two binary relationships, *father* and *mother*
    - Using two binary relationships allows partial information (e.g., only mother being known)
  - But there are some relationships that are naturally non-binary
    - Example: *proj_guide*

309

Converting Non-Binary Relationships to Binary Form

- In general, any non-binary relationship can be represented using binary relationships by creating an artificial entity set.
  - Replace *R* between entity sets A, B and C by an entity set *E*, and three relationship sets:
    1. R_A, relating *E* and *A*
    2. R_B, relating *E* and *B*
    3. R_C, relating *E* and *C*
  - Create an identifying attribute for *E and* add any attributes of *R* to *E*
  - For each relationship (a_i , b_i , c_i) in *R,* create
    1. a new entity e_i in the entity set *E*
    2. add (e_i , a_i) to R_A
    3. add (e_i , b_i) to R_B
    4. add (e_i , c_i) to R_C



310

Converting Non-Binary Relationships (Cont.)

- Also need to translate constraints
  - Translating all constraints may not be possible
  - There may be instances in the translated schema that cannot correspond to any instance of *R*
  - We can avoid creating an identifying attribute by making E a weak entity set identified by the three relationship sets

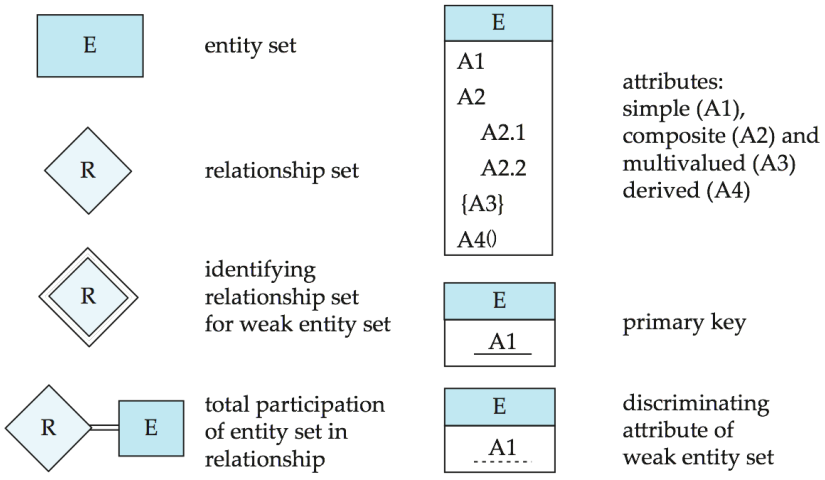
311

E-R 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 specialization/generalization – contributes to modularity in the design.
- The use of aggregation – can treat the aggregate entity set as a single unit without concern for the details of its internal structure.

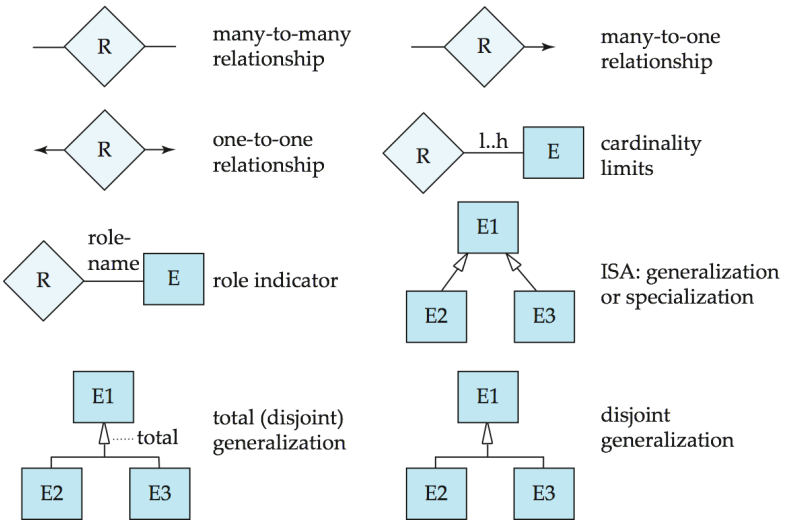
312

Summary of Symbols Used in E-R Notation



313

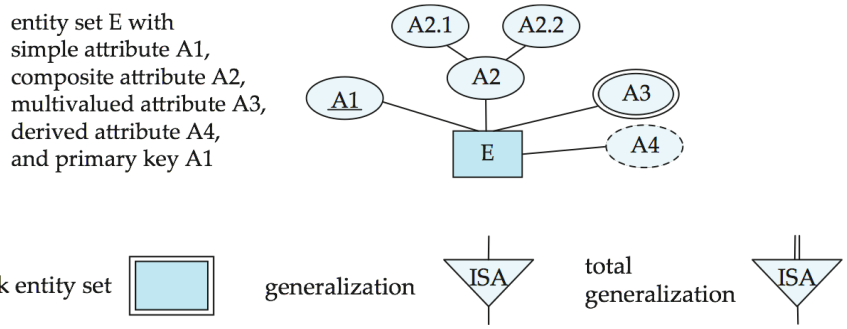
Symbols Used in E-R Notation (Cont.)



314

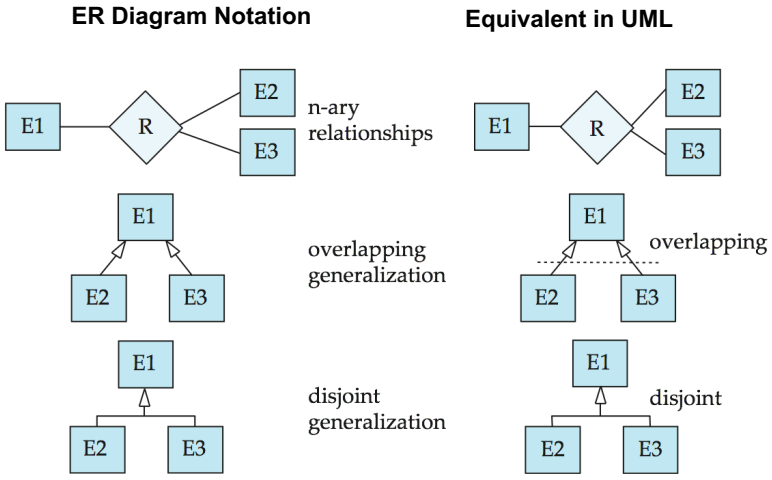
Alternative ER Notations

- Chen, IDE1FX, …



315

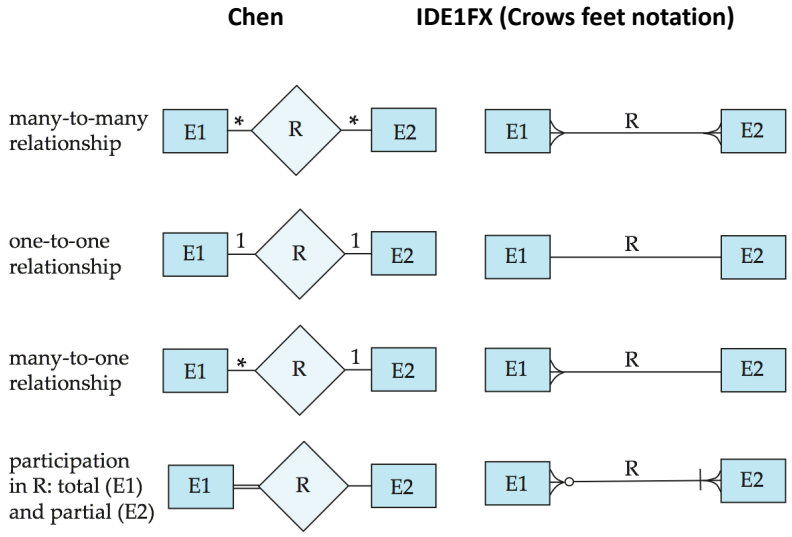
ER vs. UML Class Diagrams



*Generalization can use merged or separate arrows independent of disjoint/overlapping

316

Alternative ER Notations



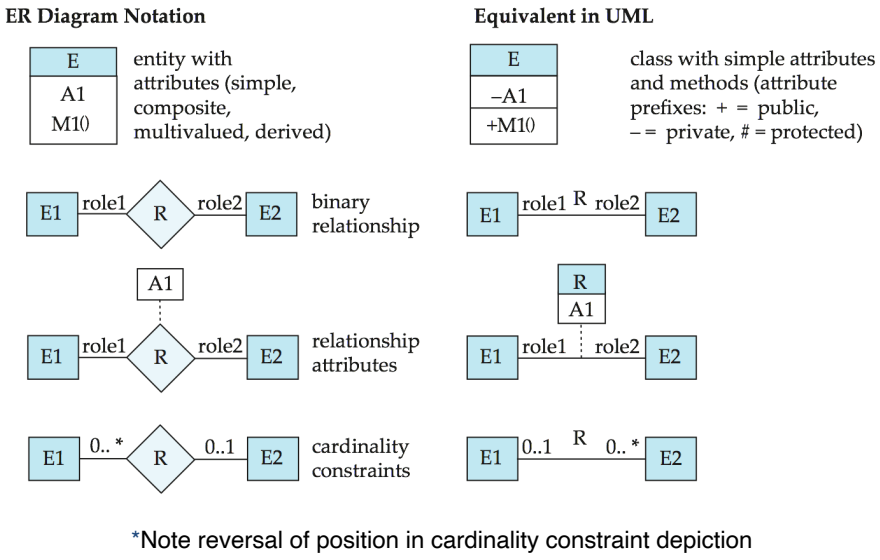
317

UML

- **UML**: Unified Modeling Language
- UML has many components to graphically model different aspects of an entire software system
- UML Class Diagrams correspond to E-R Diagram, but several differences.

318

ER vs. UML Class Diagrams



*Note reversal of position in cardinality constraint depiction

319