ECE30030/ITP30010 Database Systems

E-R Model

Reading: Chapter 6

Charmgil Hong

charmgil@handong.edu

Spring, 2023

Handong Global University

Announcement

- Midterm exam
  - Regular class hours on Thursday, April 27 (at OH401)
  - Covered topics
    - Relational-DBMS motivation and background
    - Data models
    - Relational algebra
    - SQL DML
    - SQL DDL
    - E-R model
    - Normalization
  - You may bring a cheat-sheet with you:
    - A4-sized, up to 2-page (front and back)
    - Hand-written (machine-printed sheets are NOT allowed)
    - Your name and student number should be written on the top of both sides

Announcement

- Homework assignment #3 is out
  - Due: By the end of Saturday, April 22
  - Submit only the following problems
    - **Problem 1-3**
    - **Problem 5 a-d**
    - *We will keep the remaining problems for the next homework*
      - *Problem 4*
      - *Problem 5 e-l*
  - Please start early

QnA

float_num FLOAT	double_num DOUBLE	int_num INT	bigint_num BIGINT	deci_num DECIMAL
3.14159	3.14159265359	3141592	3141592	3.14

- SELECT *, float_num+deci_num, double_num+deci_num,
  int_num+deci_num, bigint_num+deci_num FROM num_types;

float_num FLOAT	double_num DOUBLE	int_num INT	bigint_num BIGINT	deci_num DECIMAL
3.14159	3.14159265359	3141592	3141592	3.14

float_num+deci_num DOUBLE	double_num+deci_num DOUBLE	int_num+deci_num DECIMAL	bigint_num+deci_num DECIMAL
6.281592741012574	6.28159265359	3141595.14	3141595.14

QnA

float_num FLOAT	double_num DOUBLE	int_num INT	bigint_num BIGINT	deci_num DECIMAL
3.14159	3.14159265359	3141592	3141592	3.14

- SELECT *, float_num+deci_num, double_num+deci_num,
  int_num+bigint_num, float_num+float_num,
  float_num+double_num FROM num_types;

float_num FLOAT	double_num DOUBLE	int_num INT	bigint_num BIGINT	deci_num DECIMAL
3.14159	3.14159265359	3141592	3141592	3.14

float_num+deci_num DOUBLE	double_num+deci_num DOUBLE	int_num+bigint_num BIGINT	float_num+float_num DOUBLE	float_num+double_num DOUBLE
6.281592741012574	6.28159265359	6283184	6.2831854820251465	6.283185394602573

Agenda

- Designing a database
- E-R diagrams

Design Phases

- Initial phase: characterize fully the data needs of the prospective database users
- Second phase: choose a data model
  - Apply the concepts of the chosen data model
  - Translate the requirements into a conceptual schema of the database
  - A fully developed conceptual schema indicates the functional requirements of the enterprise
    - Describe the kinds of operations (or transactions) that will be performed on the data

Design Phases

- Final Phase: Move from an abstract data model to the implementation of the database
  - Logical Design – Deciding on the database schema
    - Database design requires that we find a "good" collection of relation schemas
    - *Business decision – What attributes should we record in the database?*
    - *Computer Science decision – What relation schemas should we have and how should the attributes be distributed among the various relation schemas?*
  - Physical Design – Deciding on the physical layout of the database

Design Phases

- In designing a database schema, we must ensure that we avoid two major pitfalls:
  - Redundancy: a bad design may result in repeated information
    - Redundant representation of information may lead to data inconsistency among the various copies of information
  - Incompleteness: a bad design may make certain aspects of the enterprise difficult or impossible to model

- Avoiding bad designs is not enough. There may be a large number of good designs from which we must choose

Design Approaches

- Entity Relationship Model
  - Models an enterprise as a collection of *entities* and *relationships*
    - Entity: a “thing” or “object” in the enterprise that is distinguishable from other objects
      - Described by a set of *attributes*
    - Relationship: an association among several entities
  - Represented diagrammatically by an *entity-relationship diagram* (E-R diagram)

- Normalization Theory
  - Formalize what designs are bad, and test for them

Agenda

- Designing a database
- **E-R diagrams**
  - Mapping cardinalities
  - Primary keys in E-R models
  - Weak entity sets
  - Reduction to relation schemas

E-R Model for Database Modeling

- The E-R data model was developed to facilitate database design by allowing specification of a database schema
  - Database schema represents the overall logical structure of a database

- The E-R data model employs three basic concepts:
  - Entity sets
  - Relationship sets
  - Attributes

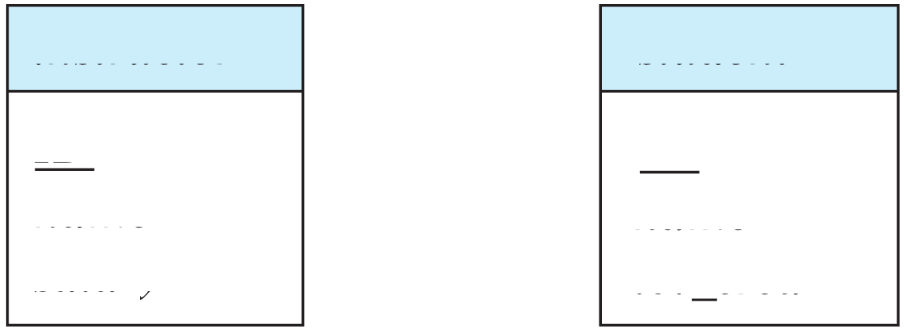
- The E-R model has an associated diagrammatic representation
  - E-R diagram can express the overall logical structure of a database graphically

Entity Sets

- An entity is an object that exists and is distinguishable from other objects
  - *E.g.*, specific person, company, event, plant
- An entity set is a set of entities of the same type that share the same properties
  - *E.g.*, set of all persons, companies, trees, holidays
- An entity is represented by a set of attributes; *i.e.*, descriptive properties possessed by all members of an entity set
  - *E.g.*, *instructor* = (*ID, name, salary*)
    course= (*course_id, title, credits*)
- A subset of the attributes form a primary key of the entity set; *i.e.*, uniquely identifying each member of the set

Representing Entity Sets in E-R Diagrams

- Entity sets can be represented graphically as follows:
  - Rectangles represent entity sets
  - Attributes listed inside entity rectangle
  - Underline indicates primary key attributes



Relationship Sets

- A relationship is an association among several entities
  - *E.g.*,

    44553 (Peltier) — *advisor* — 22222 (Einstein)

    student entity — relationship set — *instructor* entity

- A relationship set is a mathematical relation among $n \geq 2$ entities, each taken from entity sets

  $$\{(e_1, e_2, \ldots e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}$$

  where $(e_1, e_2, \ldots, e_n)$ is a relationship
  - *E.g.*, (44553,22222) $\in$ *advisor*

Example: Entity and Relationship Sets

- Entity Sets – *instructor* and *student*

76766	Crick
45565	Katz
10101	Srinivasan
98345	Kim
76543	Singh
22222	Einstein

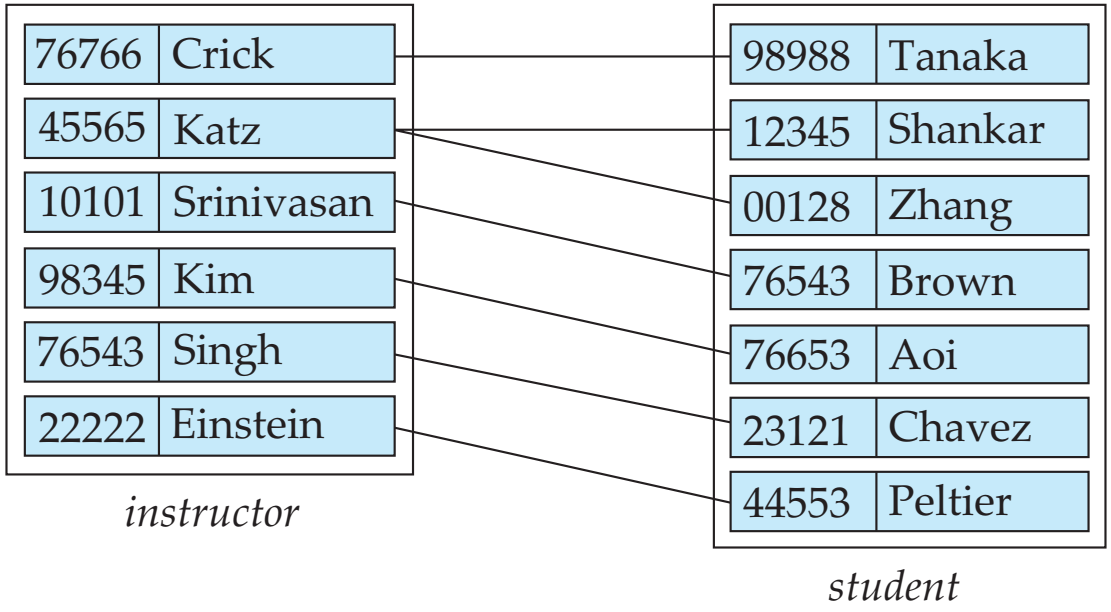
instructor

98988	Tanaka
12345	Shankar
00128	Zhang
76543	Brown
76653	Aoi
23121	Chavez
44553	Peltier

student

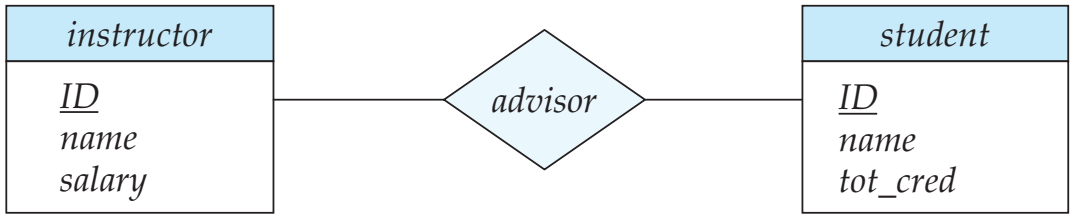
Example: Entity and Relationship Sets

- Relationship Sets – define the relationship set *advisor* to denote the associations between students and the instructors who act as their advisors



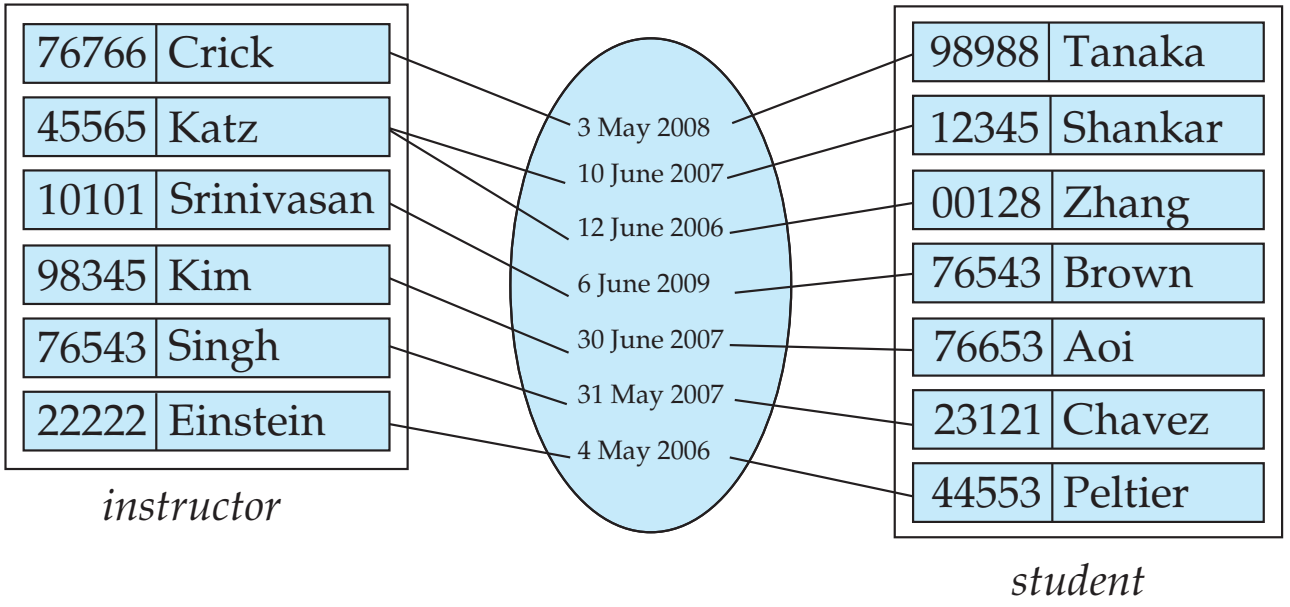
Representing Relationship Sets via E-R Diagrams

- Diamonds represent relationship sets



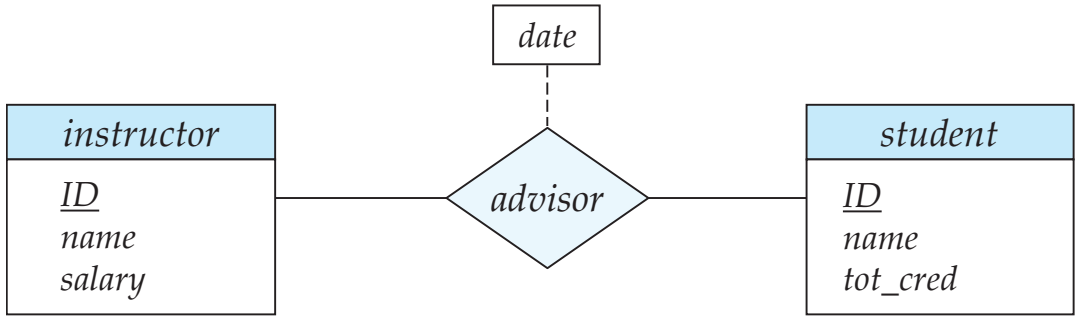
Example: Entity and Relationship Sets

- An attribute can also be associated with a relationship set
  - *E.g.*, the *advisor* relationship set between entity sets *instructor* and *student* may have the attribute *date* which tracks when the student started being associated with the advisor



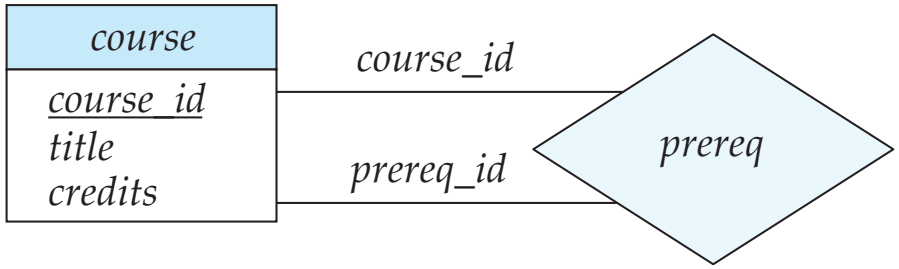
Relationship Sets with Attributes

- An attribute can also be associated with a relationship set



Roles

- Entity sets of a relationship need not be distinct
  - Each occurrence of an entity set plays a “role” in the relationship
  - *E.g.*, The labels “*course_id*” and “*prereq_id*” are called roles



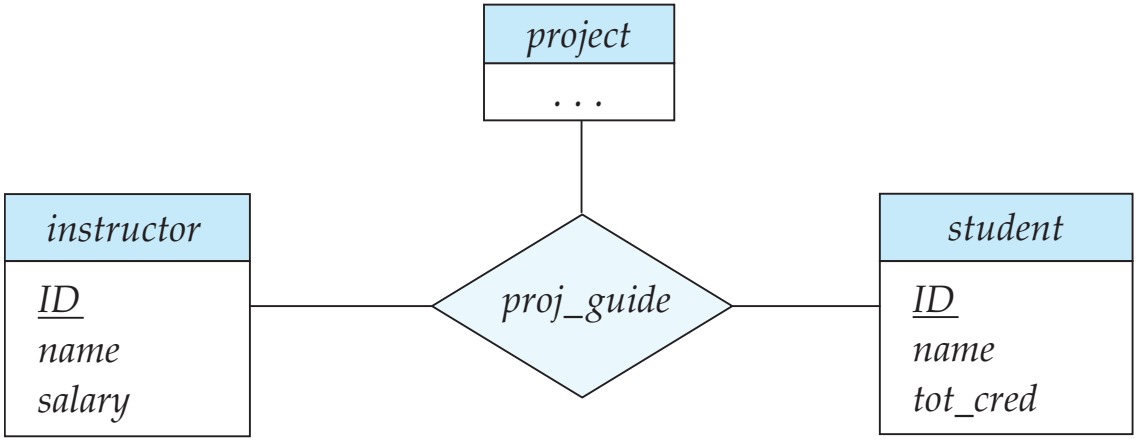
Degree of a Relationship Set

- Binary relationship
  - Involves two entity sets (or degree two)
  - Most relationship sets in a database system are binary

- Relationships between more than two entity sets are rare but possible
  - *E.g.*, *students* work on research *projects* under the guidance of an *instructor*
  - Relationship *proj_guide* is a ternary relationship between *instructor, student,* and *project*

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:

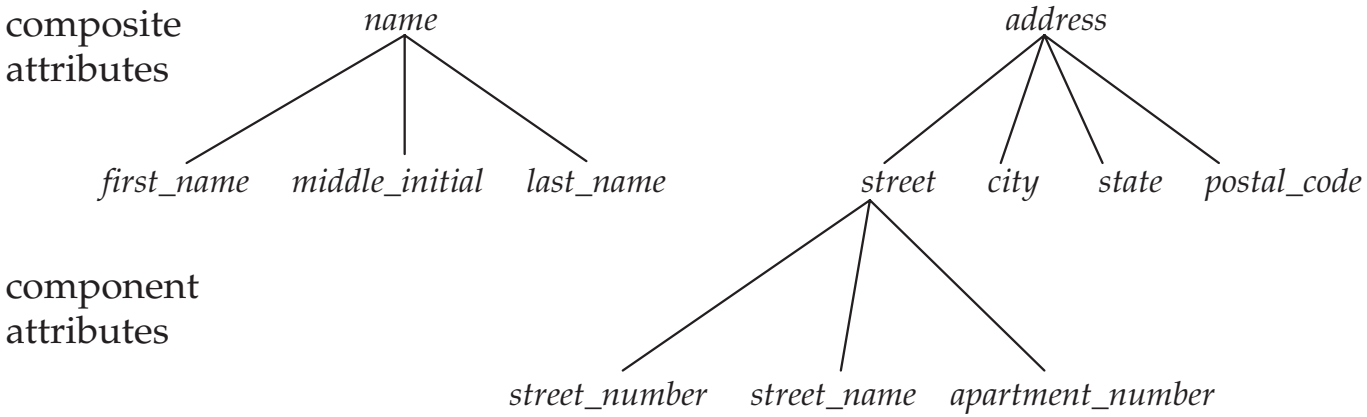


Complex Attributes

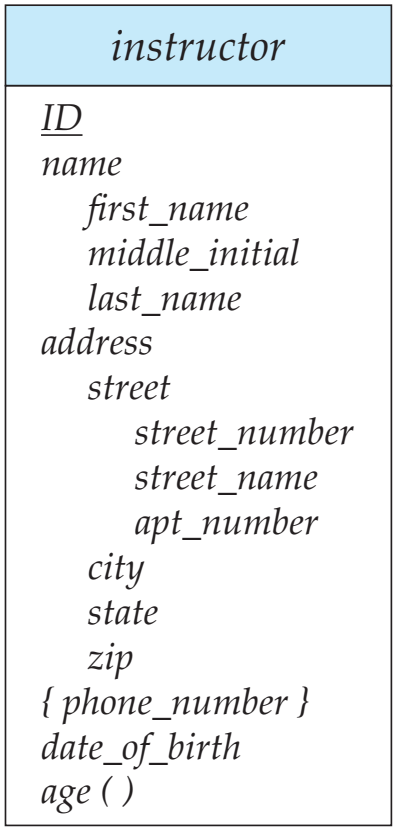
- Attribute types:
  - Simple and composite attributes
  - Single-valued and multivalued attributes
    - *E.g.*, multivalued attribute: *phone_numbers* – a person can have more than one phone numbers
  - Derived attributes: attributes that can be computed from other attributes
    - *E.g.*, age, given date_of_birth
- Domain: the set of permitted values for each attribute

Composite Attributes

- Composite attributes allow us to divided attributes into subparts (other attributes)



Representing Complex Attributes in E-R Diagrams



Agenda

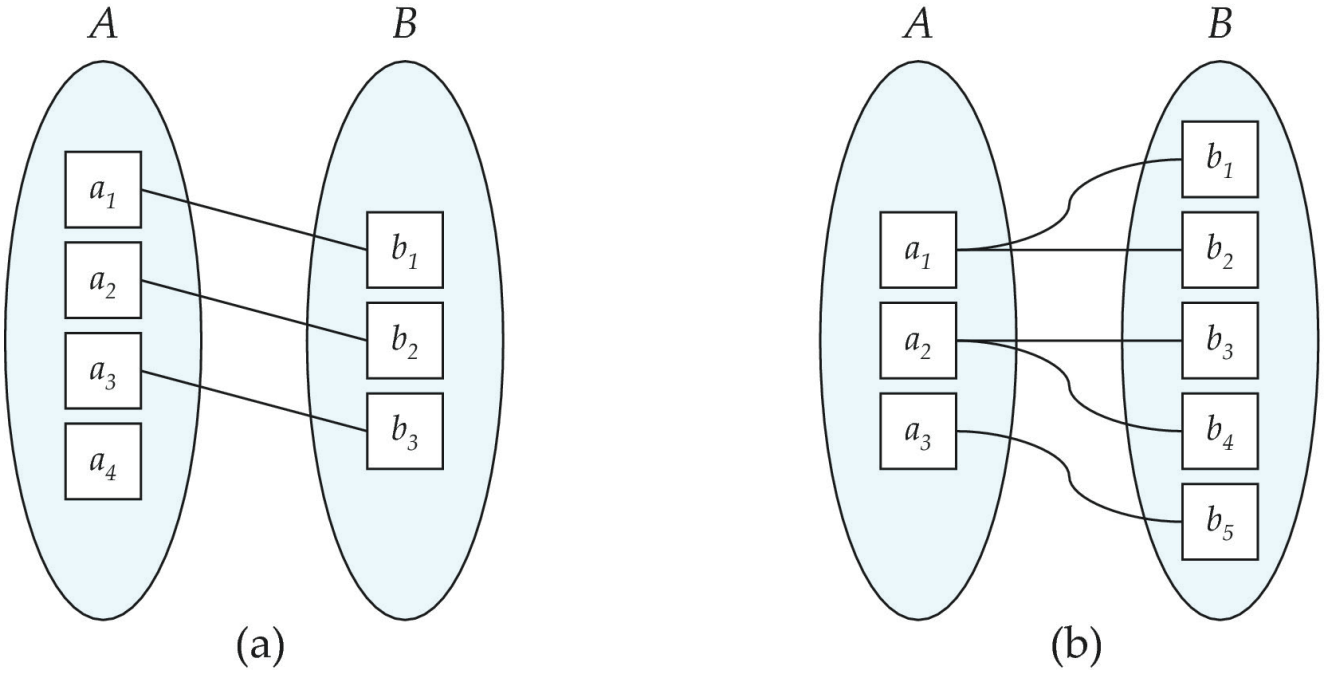
- SQL data definition language (DDL)
- Designing a database
- E-R diagrams
  - **Mapping cardinalities**
  - Primary keys in E-R models
  - Weak entity sets
  - Reduction to relation schemas

Mapping Cardinalities

- Express the number of entities to which another entity can be associated via a relationship set
  - Most useful in describing binary relationship sets

- For a binary relationship set the mapping cardinality must be one of the following types:
  - *One to one*
  - *One to many*
  - *Many to one*
  - *Many to many*

Mapping Cardinalities

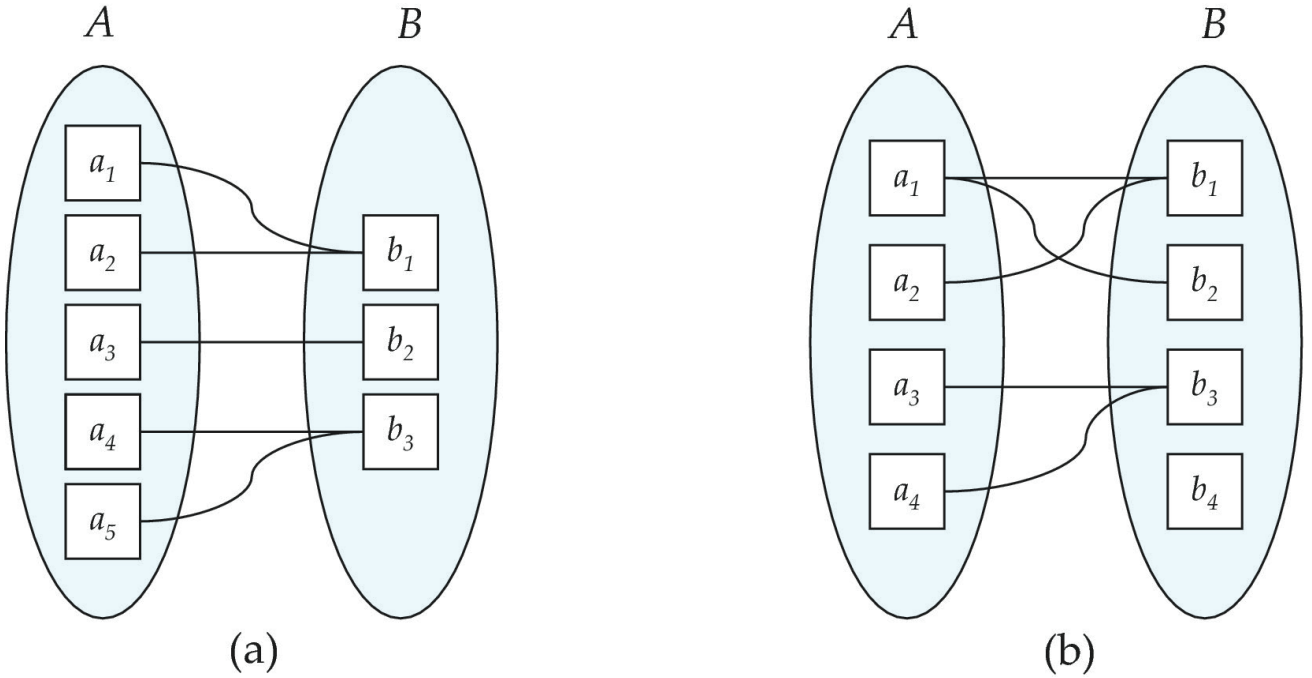


One to one

One to many

- Note: Some elements in *A* and *B* may not be mapped to any elements in the other set

Mapping Cardinalities

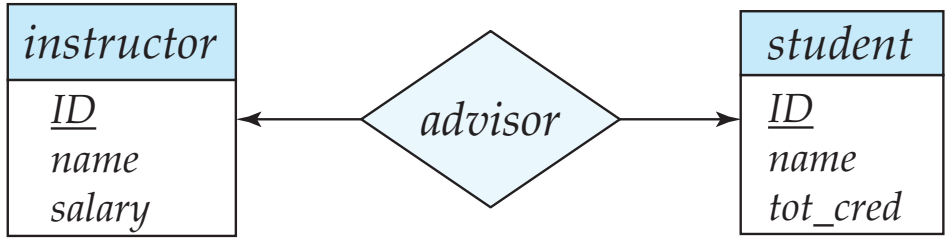


- Note: Some elements in *A* and *B* may not be mapped to any elements in the other set

Representing Cardinalities in E-R Diagrams

- 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*, and *vice versa*



One-to-Many Relationship

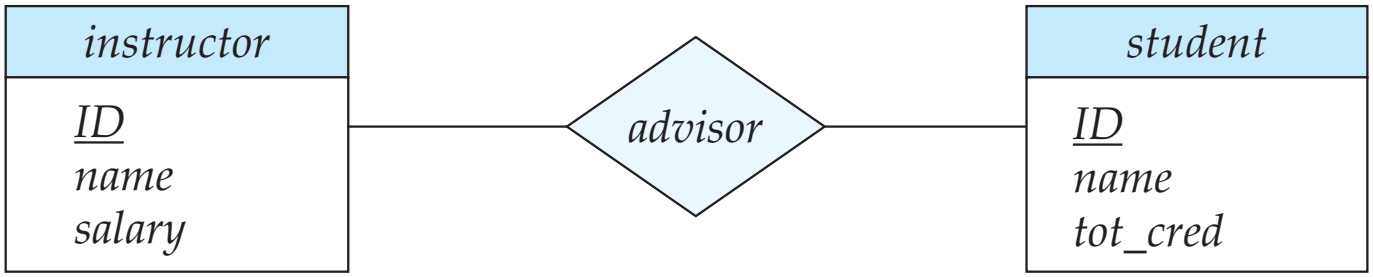
- One-to-many relationship between an *instructor* and a *student*
  - An *instructor* is associated with several (including 0) *students* via *advisor*
  - A *student* is associated with at most one *instructor* via *advisor*

Many-to-One Relationship

- Many-to-one relationship between an *instructor* and a *student*
  - An *instructor* is associated with at most one *student* via *advisor*
  - A *student* is associated with several (including 0) *instructors* via *advisor*

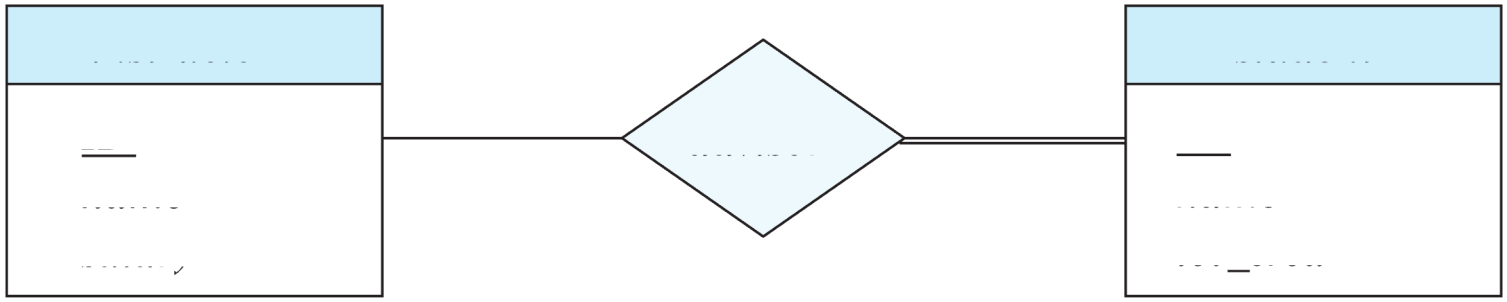
Many-to-Many Relationship

- Many-to-many relationship between an *instructor* and a *student*
  - An *instructor* is associated with several (possibly 0) *students* via *advisor*
  - A *student* is associated with several (possibly 0) *instructors* via *advisor*



Total and Partial Participation

- Total participation (*indicated by double line*): every entity in an entity set participates in at least one relationship in the relationship set



participation of *student* in *advisor r*elation is total

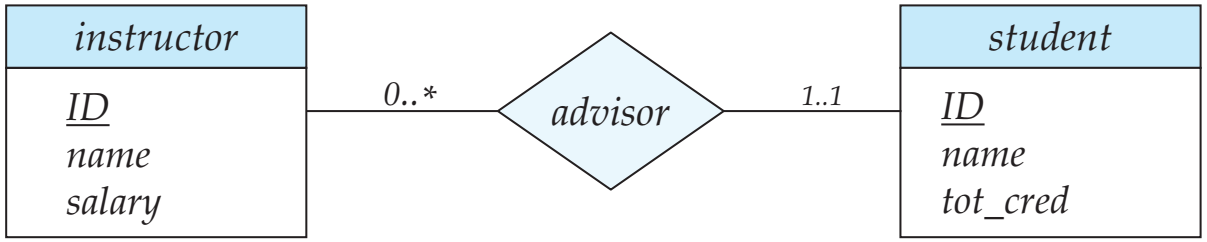
  - *E.g.*, Every *student* must have an associated *instructor*

- Partial participation: some entities may not participate in any relationship in the relationship set
  - *E.g.*, Participation of *instructor* in *advisor* is partial

Notation for Expressing More Complex Constraints

- A line may have an associated minimum and maximum cardinality, shown in the form *l..h*, where *l* is the minimum and *h* the maximum cardinality
  - A minimum value of 1 indicates total participation
  - A maximum value of 1 indicates that the entity participates in at most one relationship
  - A maximum value of * indicates no limit

- Examples
  - Instructor can advise 0 or more students
  - A student must have 1 advisor; cannot have multiple advisors



Agenda

- SQL data definition language (DDL)
- Designing a database
- E-R diagrams
  - Mapping cardinalities
  - **Primary keys in E-R models**
  - Weak entity sets
  - Reduction to relation schemas

Primary Key

- Primary keys provide a way to specify how entities and relationships are distinguished
- We consider:
  - Entity sets
  - Relationship sets
  - Weak entity sets

Primary Key for Entity Sets

- By definition, individual entities are distinct
- From database perspective, the *differences among entities must be expressed in terms of their attributes*
  - The attribute values of an entity must be such that they can uniquely identify the entity
  - No two entities in an entity set are allowed to have exactly the same value for all attributes
- A key for an entity is a set of attributes that suffice to distinguish entities from each other

Primary Key for Relationship Sets

- To distinguish among the various relationships of a relationship set, use the individual primary keys of the entities in the relationship set
  - Let *R* be a relationship set involving entity sets $E_1, E_2, \ldots, E_n$
  - The primary key for *R* is consists of the union of the primary keys of entity sets $E_1, E_2, \ldots, E_n$
  - If the relationship set *R* has attributes $a_1, a_2, \ldots, a_m$ associated with it, then the primary key of *R* also includes the attributes $a_1, a_2, \ldots, a_m$

- Example: relationship set "*advisor*"
  - The primary key consists of *inrsructor.ID* and s*tudent.ID*

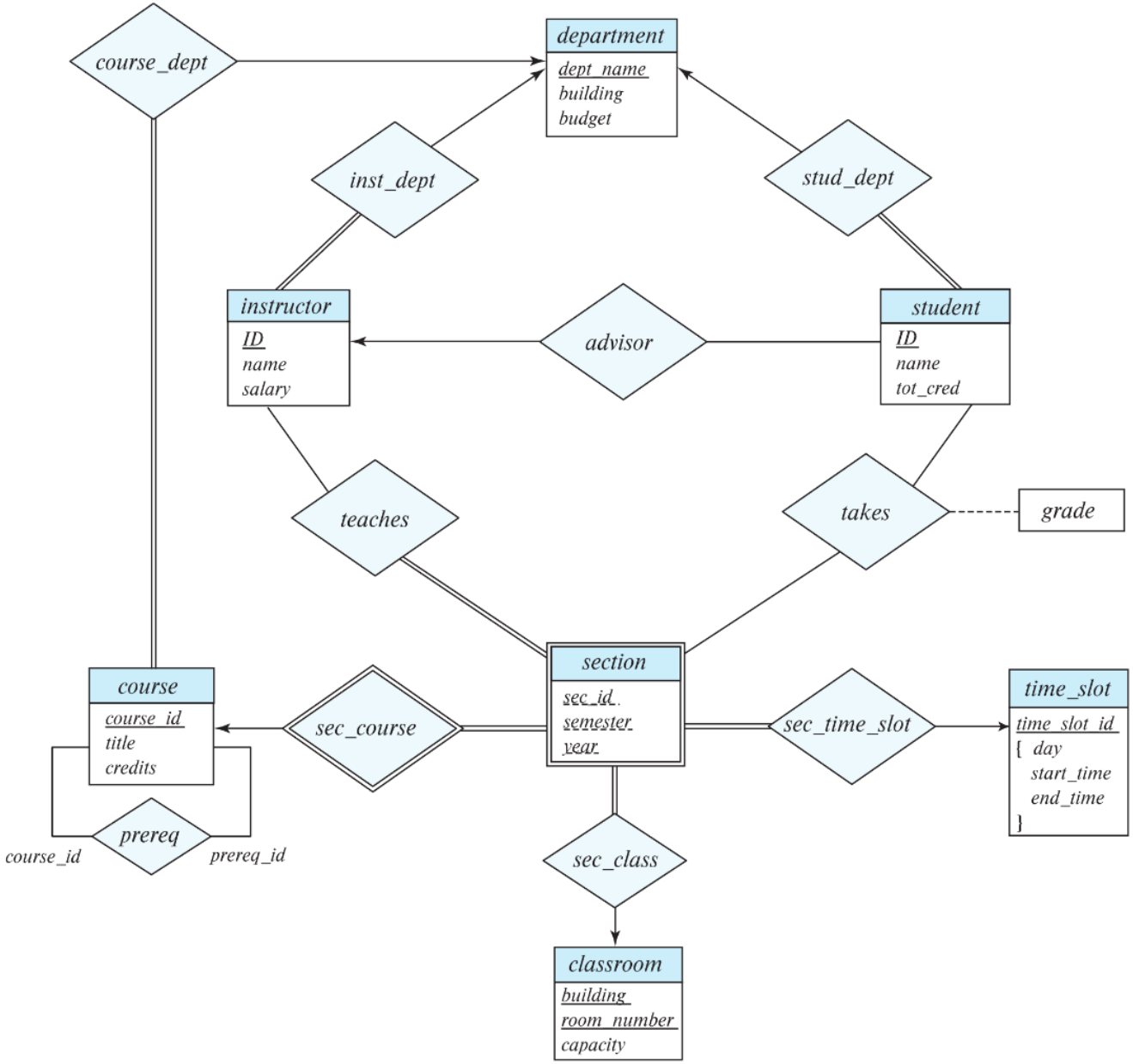
Choice of Primary Key for Binary Relationship

- The choice of the primary key for a relationship set depends on the mapping cardinality of the relationship set
  - Many-to-Many relationships: The preceding union of the primary keys is a minimal super key and is chosen as the primary key
  - One-to-Many relationships: The primary key of the "Many" side is a minimal super key and is used as the primary key
    - Many-to-one relationships: The primary key of the "Many" side is a minimal super key and is used as the primary key
  - One-to-one relationships: The primary key of either one of the participating entity sets forms a minimal super key, and either one can be chosen as the primary key

In the video lectures …

- SQL data definition language (DDL)
- Designing a database
- E-R diagrams
  - Mapping cardinalities
  - **Primary keys in E-R models**
  - **Weak entity sets**
  - **Reduction to relation schemas**

E-R Diagram for a *University* Database



Example: a Record Shop

- Entity sets
  - *customer*
  - *product* (CD, vinyl)
  - *purchase*
- Relationship sets
  - *contains* (between *product* and *purchase*)
  - *buyer* (between *customer* and *purchase*)

- An entity is a specific object; *E.g.*, a newest CD of BTS
- A relationship is specific pair of related objects

Example: a Record Shop

- Attributes
  - *customer*
    - *name, address, phone number, email, etc.*
  - *product*
    - *artist, title, price, description*
  - purchase
    - *date, payment_method*
  - Artificial primary keys for all entity sets

Example: a Record Shop

- Types of Relationships
  - 1-to-1
  - 1-to-many
    - *buyer* relation (between *customer* and *purchase*)
  - Many-to-many
    - *contains* relation (between *purchase* and *product*)

E-R Diagram

Example: Flight Database

- Entity sets
  - *airport*
    - *code, name, city*
  - *flight*
    - *flight_num, STD, STA, date_offset*
  - *departure*
    - *dept_date, capacity*
  - *customer*
    - *id, name, address, milege_num*
- Relationship sets
  - *to, from* (*flight, airport*)
  - *flight_dept (flight, departure)*
  - *reserved_on* (*customer, departure*)

E-R Diagram

ECE30030/ITP30010 Database Systems

Normalization

Reading: Chapter 7

Charmgil Hong

charmgil@handong.edu

Spring, 2023

Handong Global University

Agenda

- Motivating example
- Normal forms
- Normalization example

- *Appendix: Normalization theory*

When Data is Jumbled…

- Suppose that we combine *instructor* and *department*
  - (Below represents the join on *instructor* and *department)*

ID	name	salary	dept_name	building	budget
76766	Crick	72000.00	Biology	Watson	90000.00
10101	Srinivasan	65000.00	Comp. Sci.	Taylor	100000.00
45565	Katz	75000.00	Comp. Sci.	Taylor	100000.00
83821	Brandt	92000.00	Comp. Sci.	Taylor	100000.00
98345	Kim	80000.00	Elec. Eng.	Taylor	85000.00
12121	Wu	90000.00	Finance	Painter	120000.00
76543	Singh	80000.00	Finance	Painter	120000.00
32343	El Said	60000.00	History	Painter	50000.00
58583	Califieri	62000.00	History	Painter	50000.00
15151	Mozart	40000.00	Music	Packard	80000.00
22222	Einstein	95000.00	Physics	Watson	70000.00
33456	Gold	87000.00	Physics	Watson	70000.00

When Data is Jumbled…

- Key issues
  - Repetition of data → increases the size of database
  - *Data consistency issues*
    - Insertion anomaly: Inserting redundant data for every new record
    - Deletion anomaly: Loss of related data, when some data is deleted
    - Update anomaly: When updating certain information, every single record must be updated

ID	name	salary	dept_name	building	budget
76766	Crick	72000.00	Biology	Watson	90000.00
10101	Srinivasan	65000.00	Comp. Sci.	Taylor	100000.00
45565	Katz	75000.00	Comp. Sci.	Taylor	100000.00
83821	Brandt	92000.00	Comp. Sci.	Taylor	100000.00
98345	Kim	80000.00	Elec. Eng.	Taylor	85000.00
12121	Wu	90000.00	Finance	Painter	120000.00
76543	Singh	80000.00	Finance	Painter	120000.00
32343	El Said	60000.00	History	Painter	50000.00
58583	Califieri	62000.00	History	Painter	50000.00
15151	Mozart	40000.00	Music	Packard	80000.00

Solution: Decomposition!

- How to avoid the repetition-of-information problem?

ID	name	salary	dept_name	building	budget
76766	Crick	72000.00	Biology	Watson	90000.00
10101	Srinivasan	65000.00	Comp. Sci.	Taylor	100000.00
45565	Katz	75000.00	Comp. Sci.	Taylor	100000.00
83821	Brandt	92000.00	Comp. Sci.	Taylor	100000.00
98345	Kim	80000.00	Elec. Eng.	Taylor	85000.00
12121	Wu	90000.00	Finance	Painter	120000.00
76543	Singh	80000.00	Finance	Painter	120000.00
32343	El Said	60000.00	History	Painter	50000.00
58583	Califieri	62000.00	History	Painter	50000.00
15151	Mozart	40000.00	Music	Packard	80000.00
22222	Einstein	95000.00	Physics	Watson	70000.00
33456	Gold	87000.00	Physics	Watson	70000.00

  - A: Decompose it into two schemas (as they were)
    - Normalization = decomposition of relational schemas
    - Key idea: split relational schemas such that only directly related data composes a relation

Decomposition

- Less redundancy → Uses smaller disk storage; Causes less issues associated with insertion, deletion, and update anomalies

- Not all decompositions are good
  - *E.g.*, Suppose we decompose

    employee(ID, name, street, city, salary)

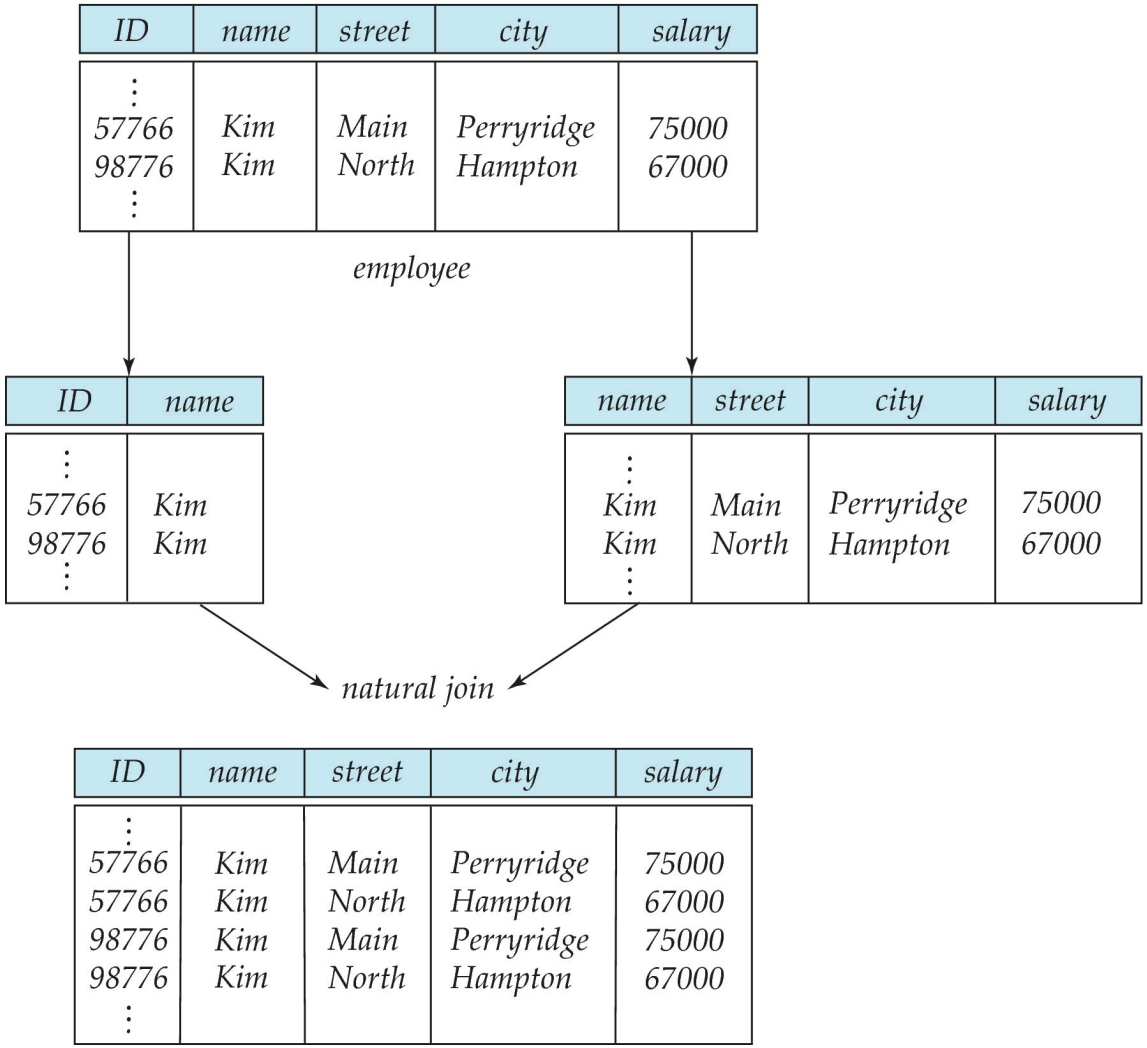
    into

    employee1 (*ID*, *name*)
    employee2 (*name*, *street, city, salary*)

    → Problem: *What if there are two employees with the same name?*

- Lossy decomposition: a decomposition from which the original relation cannot be reconstructed

Lossy Decomposition



Lossless Decomposition

- Let *R* be a relation schema and let R_1 and R_2 form a decomposition of R; that is, R = R_1 U R_2
- A decomposition is a lossless decomposition if there is no loss of information by replacing R with the two relation schemas R_1 U R_2
  - Formally, $\Pi_{R_1}(r) \bowtie \Pi_{R_2}(r) = r$
    - *C.f.*, Conversely, a decomposition is lossy if when the join of the projection results is computed, a proper superset of the original relation is returned

$$r \subset \Pi_{R_1}(r) \bowtie \Pi_{R_2}(r)$$

Example: Lossless Decomposition

- Decomposition of $R = (A, B, C)$ → $R_1 = (A, B)$; $R_2 = (B, C)$

A	B	C
α	1	A
β	2	B

r

A	B
α	1
β	2

$\Pi_{A,B}(r)$

B	C
1	A
2	B

$\Pi_{B,C}(r)$

$\Pi_A(r) \bowtie \Pi_B(r)$

A	B	C
α	1	A
β	2	B

Normalization

- Database normalization: Process of structuring a database to reduce data redundancy and improve data integrity
  - In accordance with a series of normal forms (next topic)
  - Through the process:
    - One can decompose relations to suppress data anomalies
    - One can make sure the decomposition is lossless

Agenda

- Motivating example
- **Normal forms**
- Normalization example

- *Appendix: Normalization theory*

Normal Forms

- Normalization process
  - Normalization is a database design technique, which is used to design a relational database table up to higher normal form
    - Procedurally separates logically independent (but related) data entities into multiple relations
    - Maintains the connections using keys
  - Progressive process
    - A higher level of database normalization cannot be achieved unless the previous levels have been satisfied
      - UNF: Unnormalized form
      - 1NF: First normal form
      - 2NF: Second normal form
      - 3NF: Third normal form
      - BCNF: Boyce-Codd normal form (3.5NF)
      - 4NF: Fourth normal form
      - …

Normal Forms

- Normal forms are backed by a set of normalization theories
  - *Functional dependencies*
    - *Partial dependencies*
    - *Transitive dependencies*
  - *Multi-valued dependencies*

- These theories decide whether a particular relation *R* is in "good form"
  - For a relation *R* is not in "good form", decompose it into a set of relations $\{R_1, R_2, ..., R_n\}$ such that
    - Each relation is in good form
    - The decomposition is a lossless decomposition

In the video lectures …

- Motivating example
- Normal forms
- Normalization example

EOF

- Coming next:
  - Advanced SQL