L02: Database Design Using the E-R Model

DSAN 6300/PPOL-6810 : Databases Systems and SQL Programming

Irina Vayndiner

September 5 and 7, 2023

GEORGETOWN UNIVERSITY

Logistics & Info for today's class

- Q01 will be published on **THU 9/7, due on TUE 9/19**
  - Watch Canvas for Announcements and due dates for assignments
- Monday section: class on Thu 9/28 on zoom in lieu of 9/25 class
- Use TA office hours (published on Canvas)
- Lecture
  - Database Architecture and Users
  - Database design using E-R model
- Lab: E-R model exercises

Tentative Class Dates (updated)

	DSAN-6300-01 Mon	DSAN-6300-02& PPOL-6810 Thu
1	8/28	8/31
2	9/5 (Tue!)	9/7
3	9/11	9/14
4	9/18	9/21
5	9/28 **(Thu, 9:30am) on zoom**	9/28
6	10/2	10/5
7	10/16	10/12
8 (midterm)	**10/23**	**10/19**
9	10/30	10/26
10	11/6	11/2
11	11/13	11/9
12	11/20	11/16
13	11/27	11/30
14 (test)	**12/5 (Tue, 10:30am)**	**12/5 (Tue, 10:30am)**

No classes
Mon, 9/4 Labor Day
Mon, 10/9: Mid Semester Holiday
Mon, 9/25: Yom Kippur - class on 9/28 instead!

Class added: Mon 12/5 for in lieu of 8/24 (all 3 sections)

Tools Installation: AWS and MySQL Workbench

To install: No Later Then (NLT) 9/22

1. Install MySQL Workbench
2. Accept Invite to AWS Canvas
3. Attend TA office hours next week if you have questions!

Special TA sessions on the week of 9/18 will be announced soon

!! You will not be able to use SQL in this Course without this setup!!

Outline of Today's Lecture

- Database Architecture and Users
- Database Design Using E-R Model
  - Overview of the Database Design Process
  - The Entity-Relationship Model
  - Mapping Cardinalities
  - Primary Key
  - Entity-Relationship Design Issues
  - Alternative Notations for Modeling Data
  - Other Aspects of Database Design

Elements of DBMS Ecosystem

- Was discussed in Lecture 1
  - Data Models
  - Database Design
  - Data Access (DDL and DML)
  - SQL
  - DBMS Engine
    - Query Processor
    - Transaction manager
    - Storage manager

- **Today:**
  - **Database Architecture**
  - Database Users

Database Architecture Types

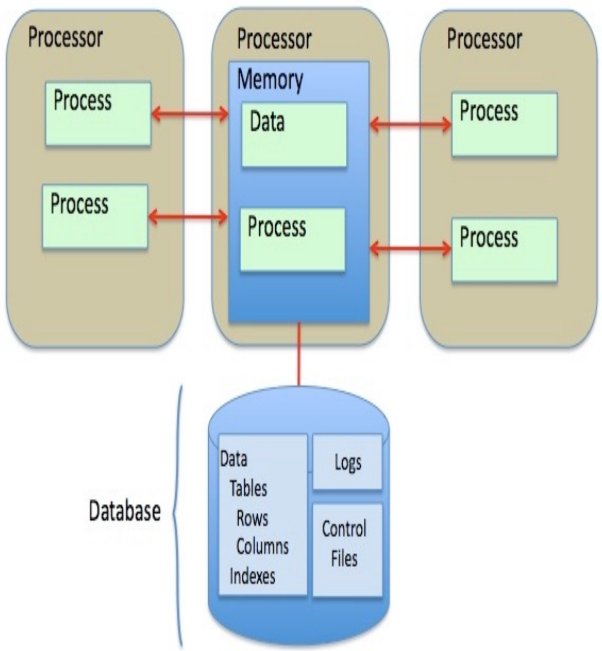
- Centralized (“shared”) databases
  - One to a few cores, shared-memory
- Client-server
  - One server machine executes work on behalf of multiple client machines.
- Multi-node databases can be
  - Shared-memory
  - Shared-disk
  - Shared-everything
  - Shared-nothing
- Distributed databases
  - Geographical distribution
  - Schema/data heterogeneity

“Shared” RDBMS Architectures

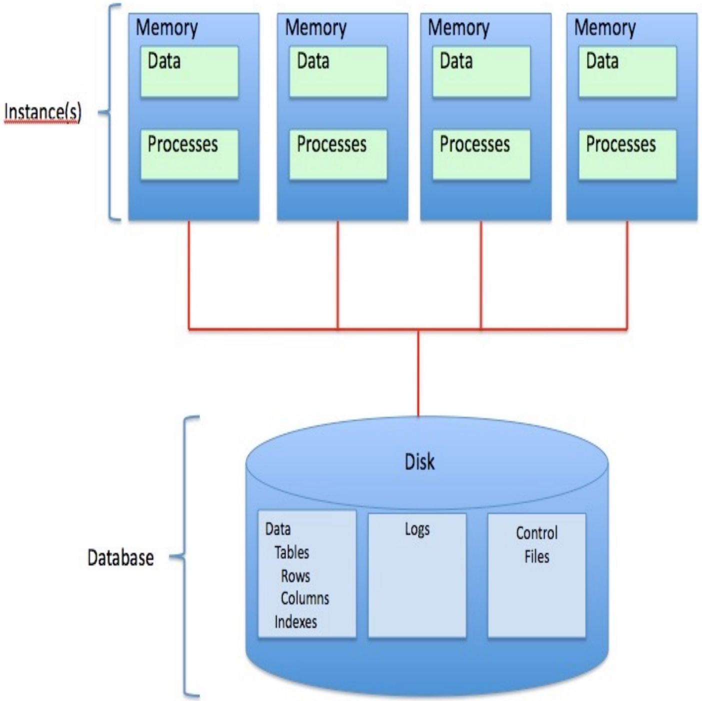
- SMP (symmetric multiprocessing). “Shared-everything" architectures share both disk and memory between nodes in the cluster. E.g.: IBM DB2 for z/OS, Sybase IQ.
  - Strength: Processing power
  - Limitation: Scalability

“Shared-disk” E.g. Oracle RAC

Shared-everything



Shared-disk



Ref: Wikipedia

“Shared-nothing” Architecture Massively Parallel Processing (MPP)

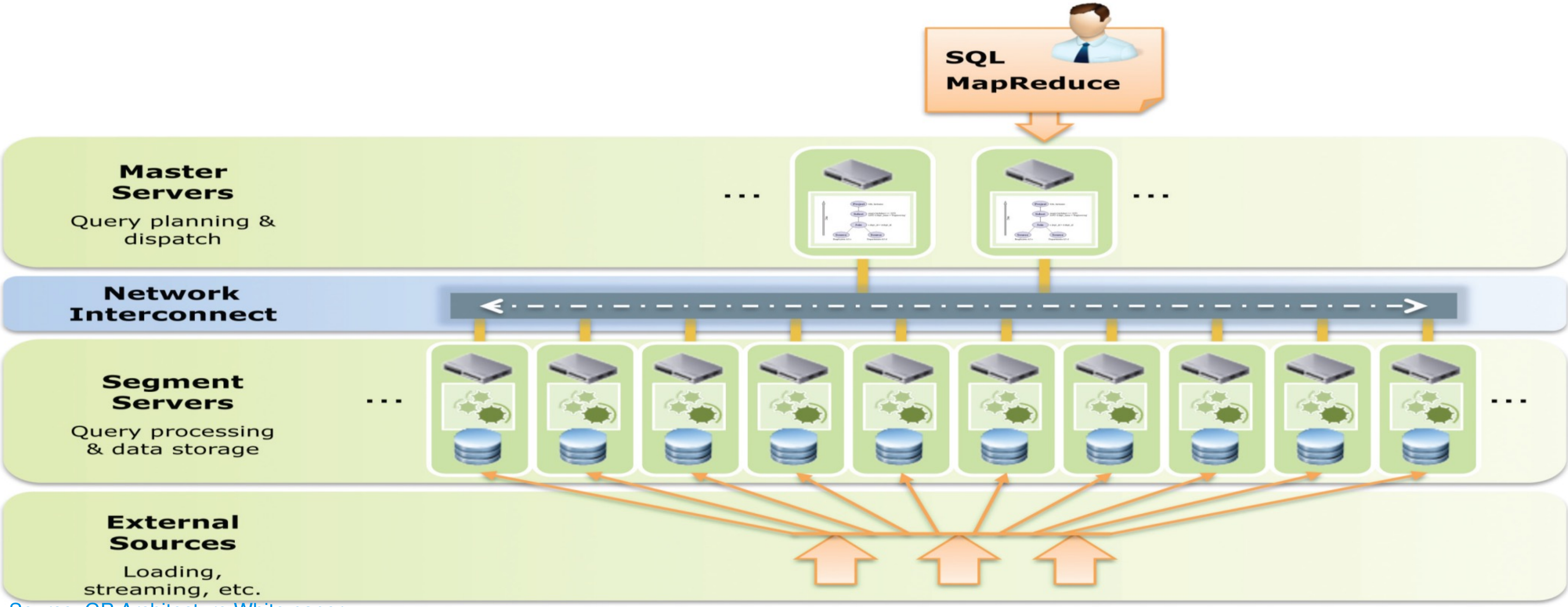
- Divide-and-conquer approach

- Massive refers to the concept of dozens or hundreds of nodes tied together achieving a single process

- Typical MPP architecture implements a shared-nothing (SN) paradigm where each parallel node operates self-sufficiently (“autonomously”) and controls its own memory and disk
  - A collection of nodes, each with local disk and local main memory, connected together by a high-speed network interconnect
  - Processor-disk pairs operating in parallel divide the workload to execute queries over large sets of data.
  - Adding nodes increases performance as well as capacity, scales linearly as new nodes are added into the system, to 100s of TBs and into PBs

Shared Nothing (Massively Parallel) Relational Databases

- Nodes are connected by a high-speed Network Interconnect
- Master node acts as a “brain” that ties all the compute nodes together
  - Determines the optimal query plan
  - Dispatches that query to all the compute nodes
  - Gathers results from compute nodes and sends them to the user. All nodes



Source: GP Architecture White paper

Elements of DBMS Ecosystem

- Data Models
- Database Design
- Data Access (DDL and DML)
  - SQL
- DBMS Engine
  - Query Processor
  - Transaction manager
  - Storage manager
- Database Architecture
- **Database Users**

Database Users

There are four different types of database-system users

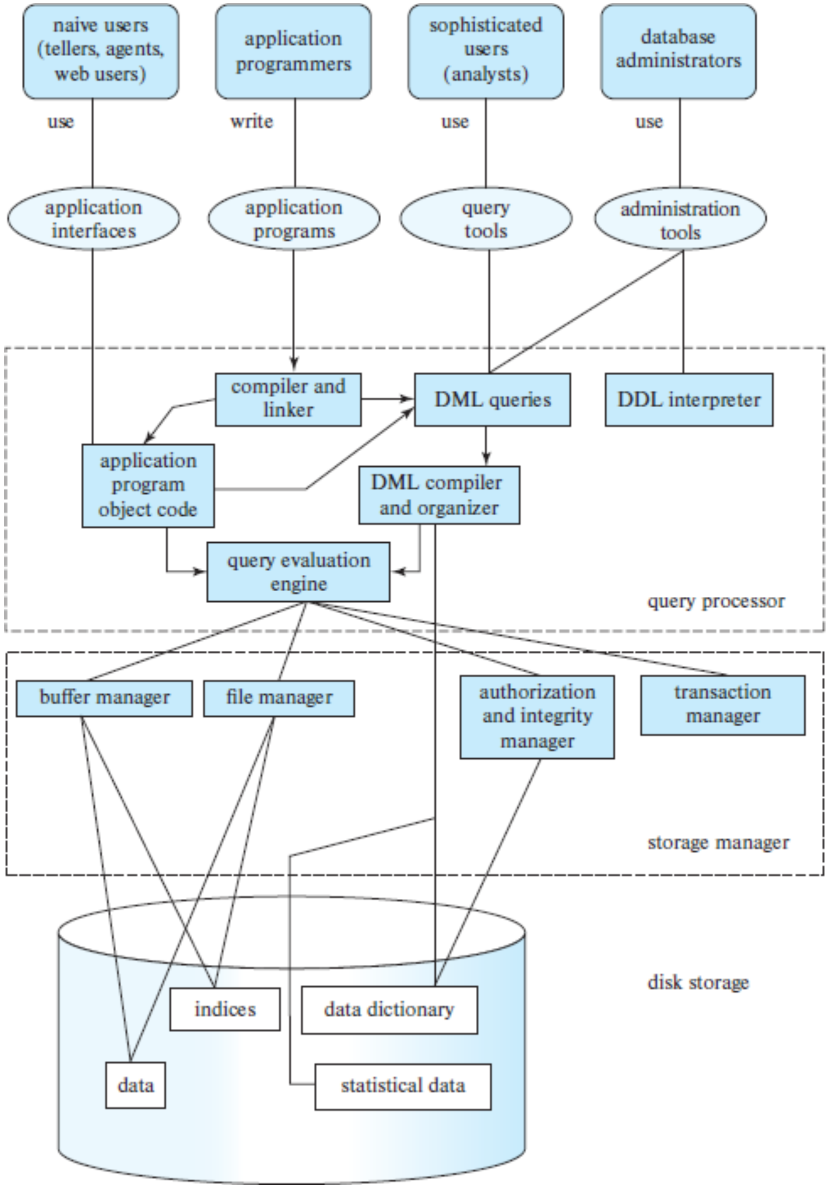
- **End users** -- users who interact with the system by invoking one of the application programs that have been written previously.
- **Application programmers** -- are computer professionals who write application programs.
- **Sophisticated users** -- interact with the system without writing programs
  - Using a database query language or
  - Using tools, such as data analysis software.
- **Specialized users** --write specialized database applications that do not fit into the traditional data-processing framework.
  - For example, CAD, graphic data, audio, video.

Database Administrator (DBA)

A person who has central control over the DBMS is called a database administrator (DBA), whose functions are:

- Schema definition
- Storage structure and access-method definition
- Schema and physical-organization modification
- Granting of authorization for data access
- Routine maintenance
- Periodically backing up the database
- Ensuring that enough free disk space is available for normal operations, and upgrading disk space as required
- Monitoring jobs running on the database and ensuring that performance is not degraded by very expensive tasks submitted by some users

Putting It All Together



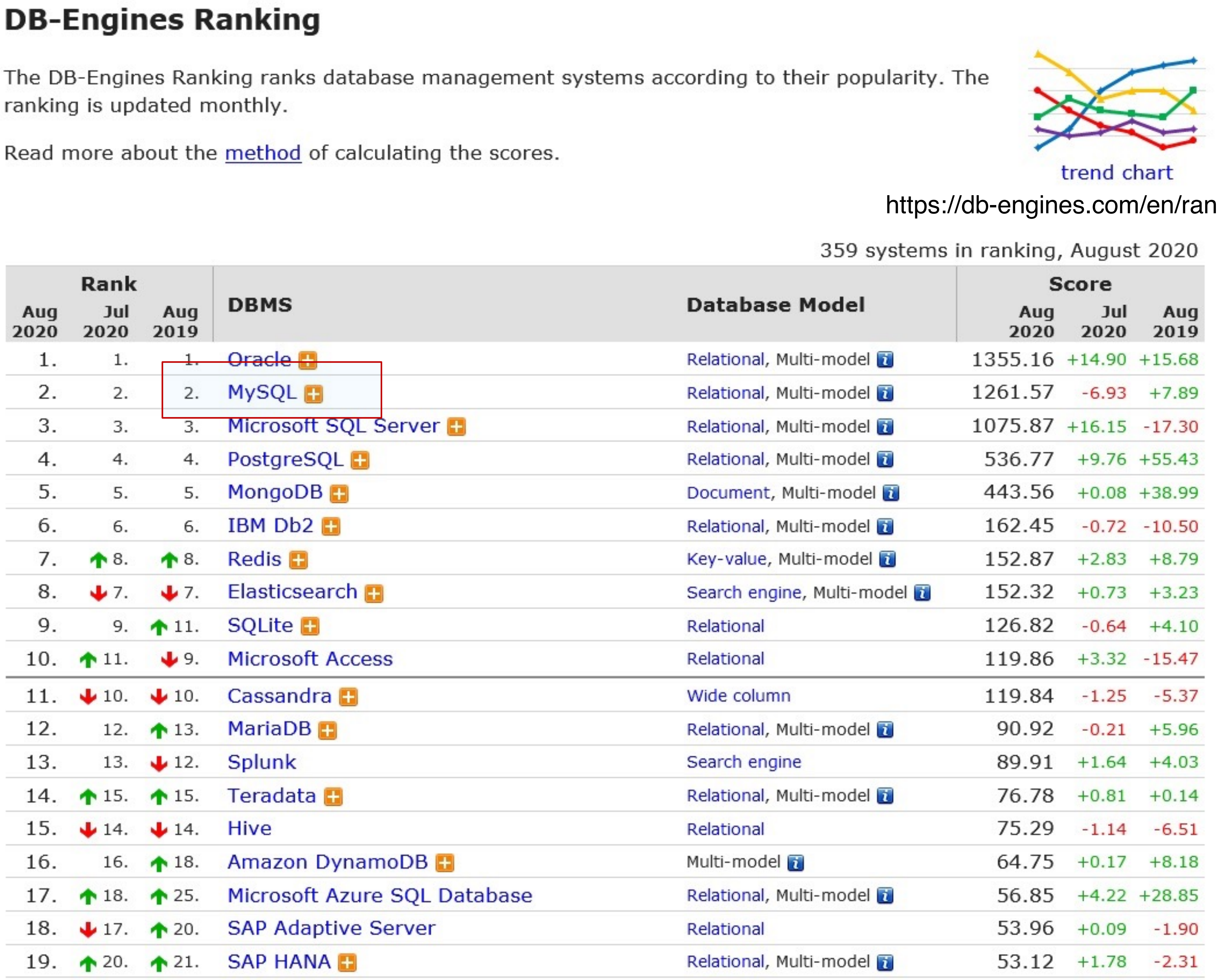
Hands-on Exercises: MySQL and AWS RDS

- MySQL is the world's most popular open source relational database and Amazon RDS makes it easy to set up, operate, and scale MySQL deployments in the cloud.

- Amazon RDS for MySQL allows you up to focus on application development by managing time-consuming database administration tasks including backups, software patching, monitoring, scaling and replication.

- Amazon RDS supports MySQL Community Edition
  - We will use **8.0.34** in Class

MySQL Ranking



DB-Engines Ranking

The DB-Engines Ranking ranks database management systems according to their popularity. The ranking is updated monthly.

Read more about the method of calculating the scores.

https://db-engines.com/en/ranking

359 systems in ranking, August 2020

Rank			DBMS	Database Model	Score		
Aug 2020	Jul 2020	Aug 2019			Aug 2020	Jul 2020	Aug 2019
1.	1.	1.	Oracle	Relational, Multi-model	1355.16	+14.90	+15.68
2.	2.	2.	MySQL	Relational, Multi-model	1261.57	-6.93	+7.89
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	1075.87	+16.15	-17.30
4.	4.	4.	PostgreSQL	Relational, Multi-model	536.77	+9.76	+55.43
5.	5.	5.	MongoDB	Document, Multi-model	443.56	+0.08	+38.99
6.	6.	6.	IBM Db2	Relational, Multi-model	162.45	-0.72	-10.50
7.	↑ 8.	↑ 8.	Redis	Key-value, Multi-model	152.87	+2.83	+8.79
8.	↓ 7.	↓ 7.	Elasticsearch	Search engine, Multi-model	152.32	+0.73	+3.23
9.	9.	↑ 11.	SQLite	Relational	126.82	-0.64	+4.10
10.	↑ 11.	↓ 9.	Microsoft Access	Relational	119.86	+3.32	-15.47
11.	↓ 10.	↓ 10.	Cassandra	Wide column	119.84	-1.25	-5.37
12.	12.	↑ 13.	MariaDB	Relational, Multi-model	90.92	-0.21	+5.96
13.	13.	↓ 12.	Splunk	Search engine	89.91	+1.64	+4.03
14.	↑ 15.	↑ 15.	Teradata	Relational, Multi-model	76.78	+0.81	+0.14
15.	↓ 14.	↓ 14.	Hive	Relational	75.29	-1.14	-6.51
16.	16.	↑ 18.	Amazon DynamoDB	Multi-model	64.75	+0.17	+8.18
17.	↑ 18.	↑ 25.	Microsoft Azure SQL Database	Relational, Multi-model	56.85	+4.22	+28.85
18.	↓ 17.	↑ 20.	SAP Adaptive Server	Relational	53.96	+0.09	-1.90
19.	↑ 20.	↑ 21.	SAP HANA	Relational, Multi-model	53.12	+1.78	-2.31

Some of MySQL Customers

YouTube
PayPal
Google
facebook
ebay
CISCO
LinkedIn
UBER
TESLA
NETFLIX
GitHub
Walmart
Booking.com
Zappos.com
H&M
Flipkart
Paytm
Alibaba.com
WeChat
Tencent 腾讯
淘宝网 Taobao.com
airbnb
Alcatel·Lucent

Amazon Relational Database Service (RDS)

https://www.youtube.com/watch?v=eMzCI7S1P9M

Summary for Using Databases

- A major purpose of a Database system is to provide users with a way to manage and use that data.
- Underlying the structure of a database is the data model. The relational data model is the most widely deployed model for storing data in databases.
- A data-manipulation language (DML) is a language that enables users to access or manipulate data. SQL is the most widely used DML language.
- A database system has several subsystems: storage manager, query processor, transaction manager
- The architecture of a database system can be centralized or parallel

Takeaways

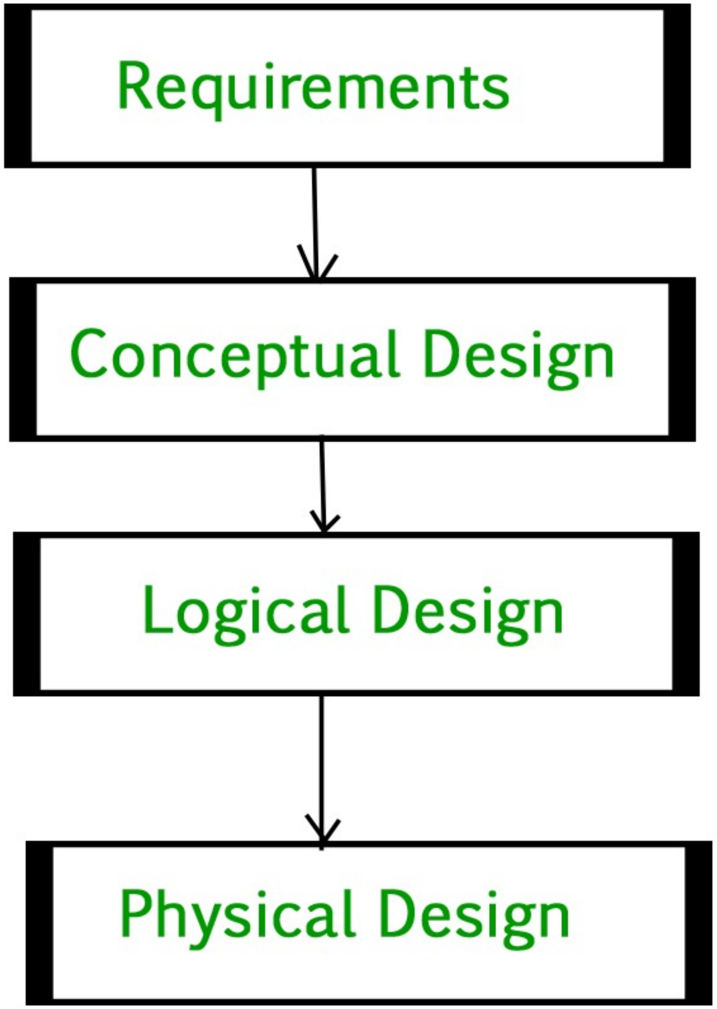
- Databases and SQL touch almost every part of modern technology
- This course will better equip you to succeed in your study, your research, and your job

Today’s main subject

Database Design

Database Design Phases

- Requirements Analysis-- characterize fully the data needs of the prospective database users.
- **Conceptual Design** -- choosing a **data model** (today's lecture!)
  - **The Entity-Relationship (E-R) model** is typically the result of conceptual design.
  - Conceptual schema specifies the **entities** that are represented in the database, the **attributes** of the entities, the **relationships** among the entities, and **constraints** on the entities and relationships
- Logical Design – Deciding on the database schema.
- Physical Design – Deciding on the physical layout of the database



E-R model for Database Modeling

- The E-R data model was developed to facilitate database design by allowing specification of an **enterprise schema**
- The E-R data model employs three basic concepts
  - entity sets
  - relationship sets
  - attributes
- The E-R model also has an associated *graphical representation*, the **E-R diagram**, which can express the overall structure of a database graphically.

Design Approach

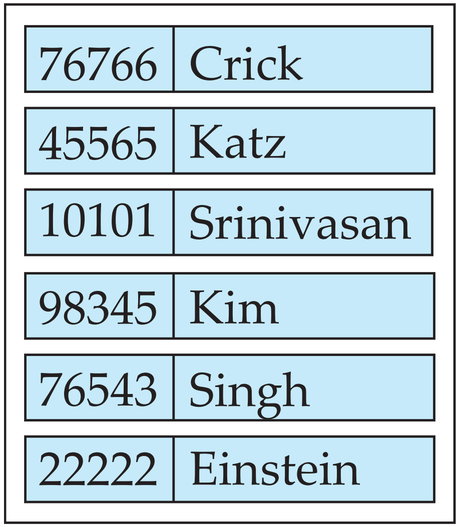
- Entity-Relationship Model
  - Models an enterprise as a collection of *entities* and *relationships*
    - Entity: a “thing” or “object” in the enterprise that is distinguishable from other objects
      - Described by a set of *attributes*
    - Relationship: an association among several entities

Definitions: Entity Sets

- An **entity** is an object that exists and is distinguishable from other objects.
  - Example: specific person, company, event, plant
- An **entity set** is a set of entities of the same type that share the same properties.
  - Example: set of all persons, companies, trees, holidays
- An entity is represented by a set of **attributes**; i.e., descriptive properties that are possessed by all members of an entity set.
  - Example:

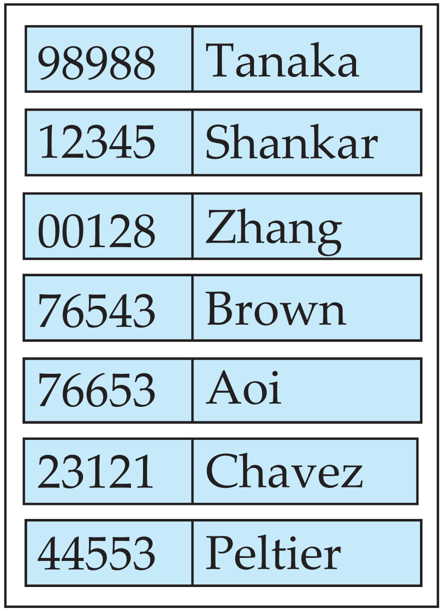
    instructor = (*ID, name, salary*)
    course= (*course_id, title, credits*)
- A subset of the attributes form a **primary key** of the entity set; i.e., **uniquely** identifying each member of the set.
  - We will discuss PK in detail later

Entity Sets -- *instructor* and *student*



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

instructor

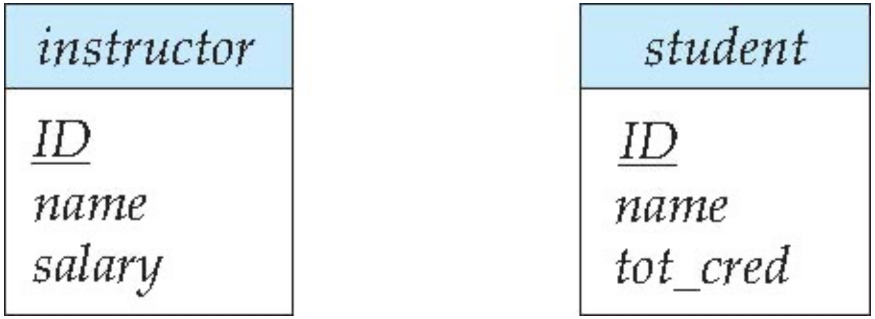


98988	Tanaka
12345	Shankar
00128	Zhang
76543	Brown
76653	Aoi
23121	Chavez
44553	Peltier

student

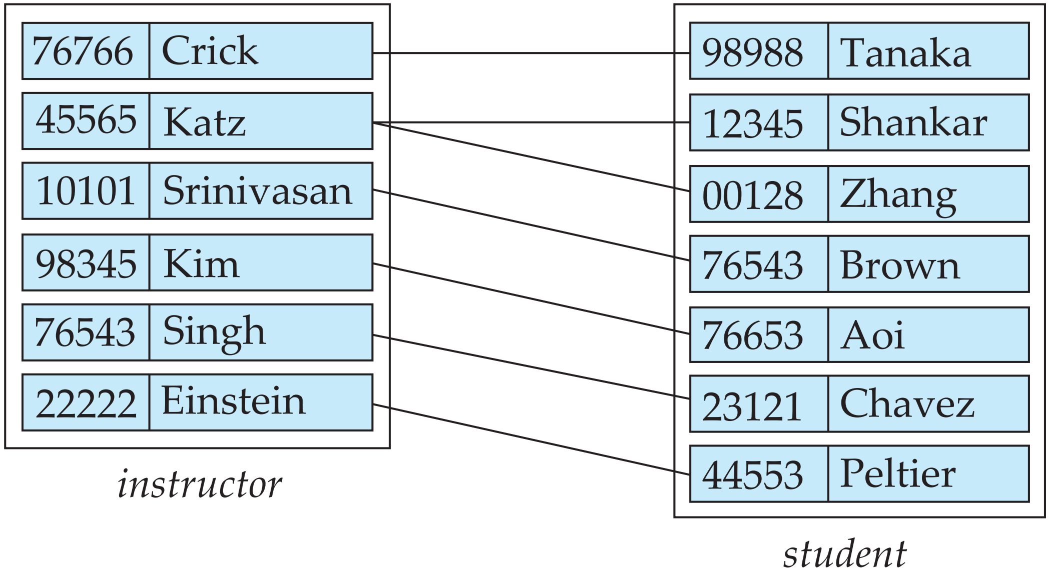
Representing Entity Sets in E-R Diagram

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



Definitions: Relationship Sets

- A **relationship** is an association among several entities
  - Example: we define the relationship ***advisor*** to denote the associations between students and the instructors who act as their advisors.
- A **relationship set** is a collection of similar relationships – similar because it relates entities from the same entity sets
  - Pictorially, we draw a line between related entities.
  - Each line is a relationship, <u>all lines </u>are a <u>relationship set</u>



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

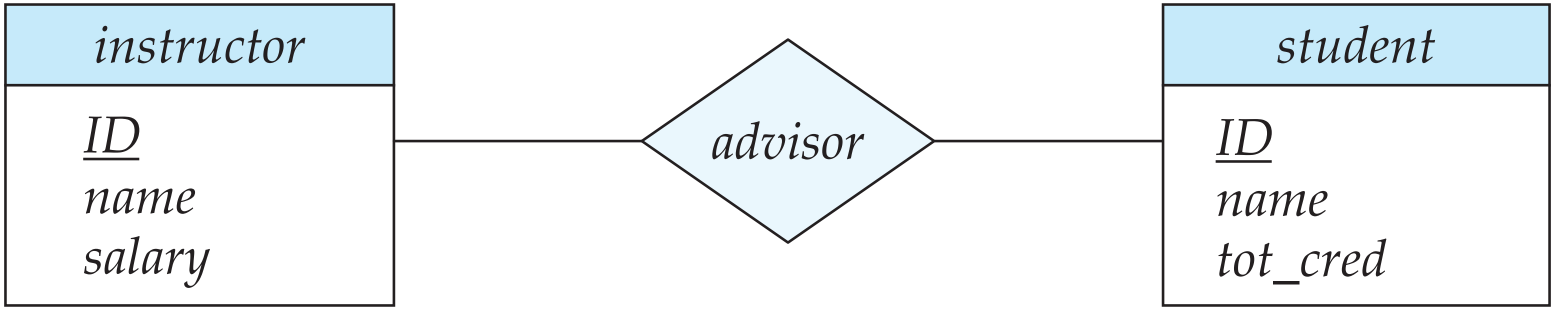
instructor

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

student

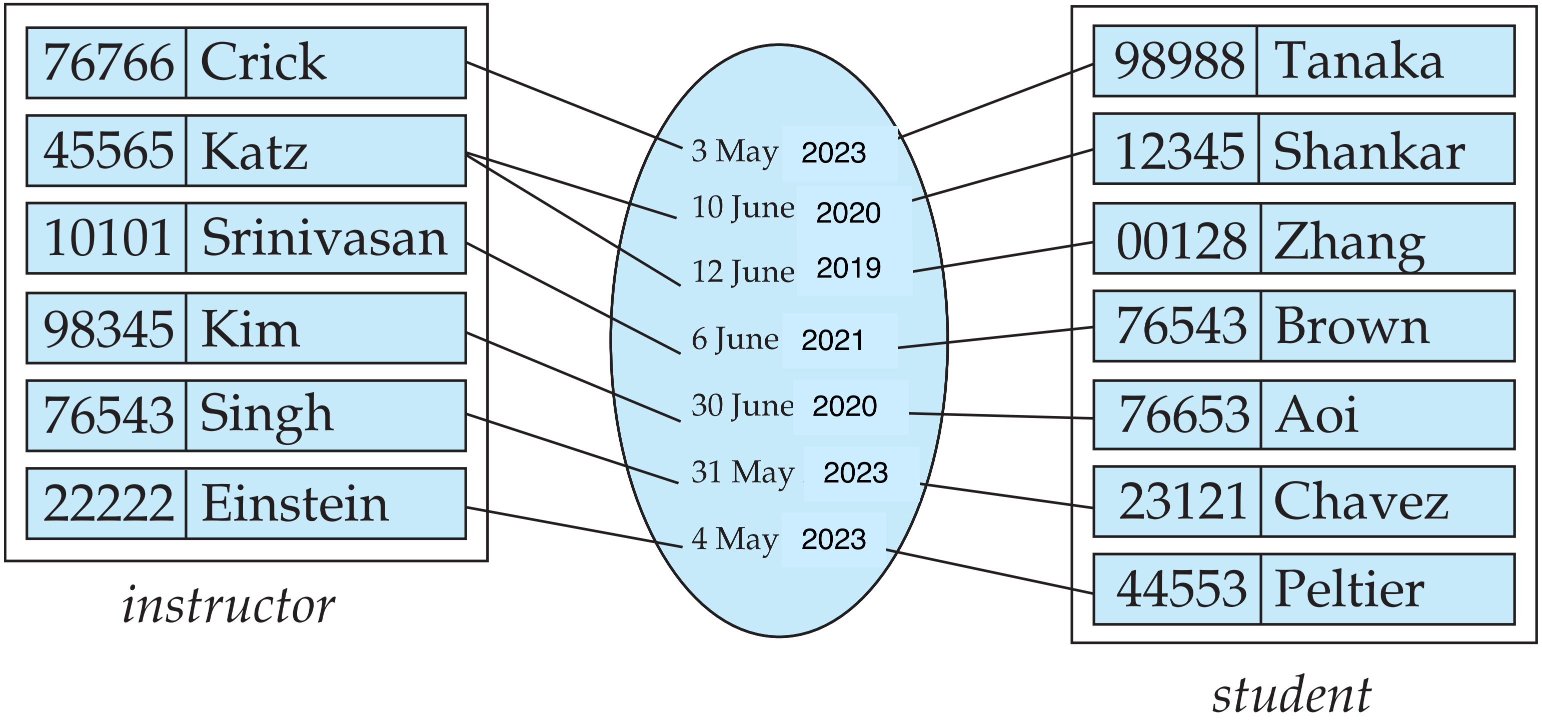
Representing Relationship Sets via E-R Diagrams

Diamonds represent graphically relationship sets



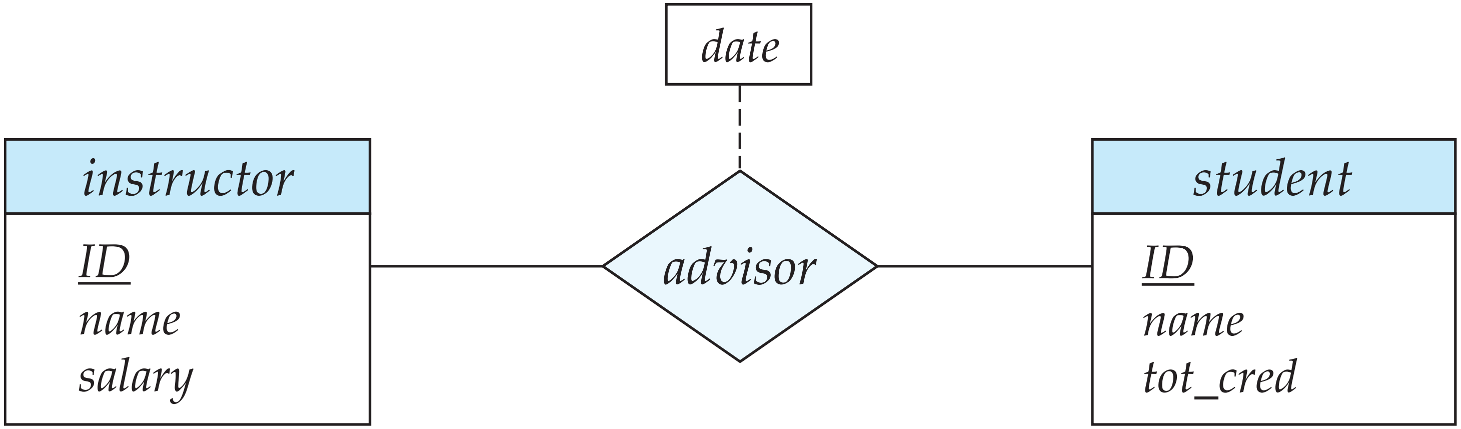
Attributes in Relationship Sets

- An attribute can also be associated with a relationship set.
- For instance, 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



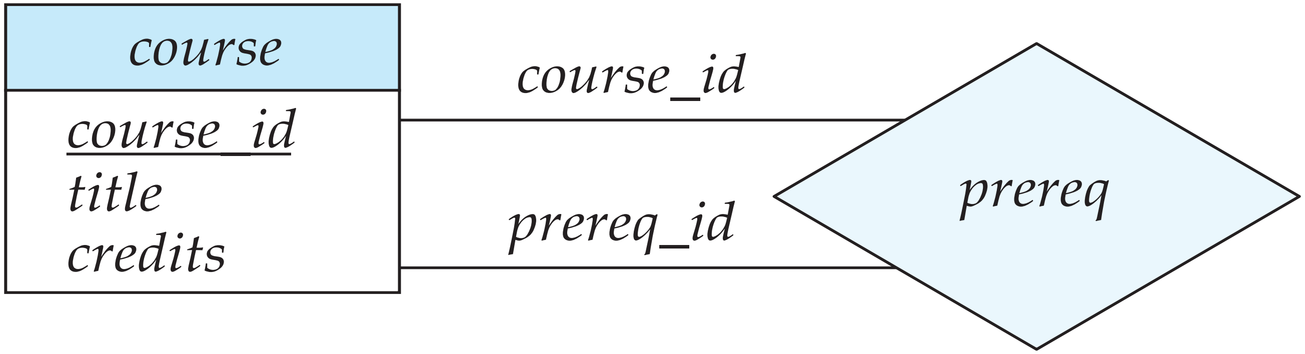
Relationship Sets with Attributes

Dashed line represents the attributes of relationship sets



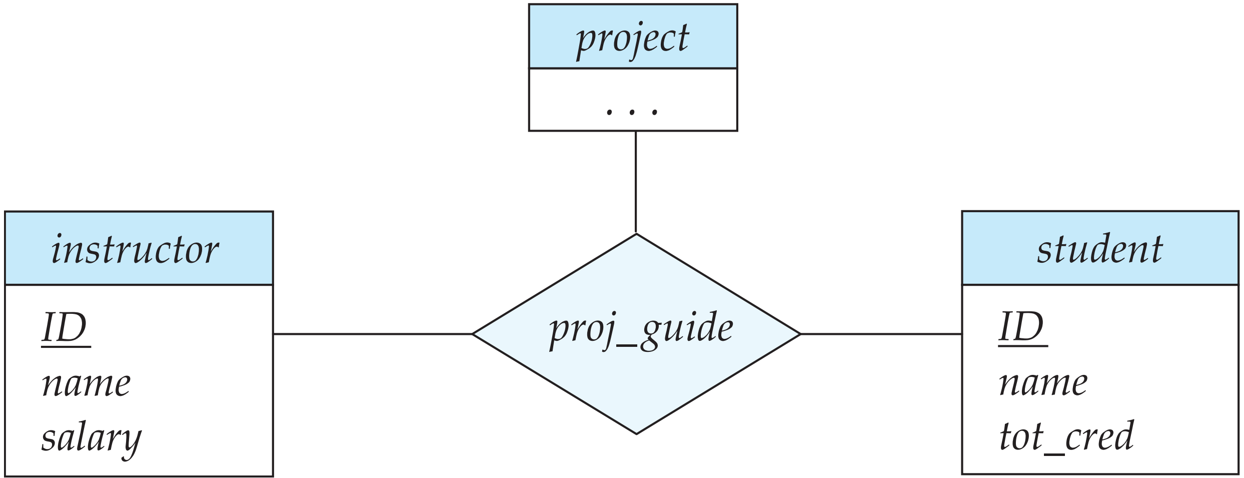
Roles

- Entity sets of a relationship need not be distinct
  - Same entity set can participate in a relationship set more than once, in different roles
  - Each occurrence of an entity set plays a "role" in the relationship
- For example, let *course* be the entity set that records all courses in the University.
  - Say, Course 1 (C1) is a prerequisite for Course 2 (C2) => relationship set *prereq*
  - Ordered pairs of Course entities (C1, C2)
- The labels "*course_id*" and "*prereq_id*" are called **roles**.



Degree of a Relationship Set

- Binary relationship
  - Involves two entity sets (degree = 2).
  - Most relationship sets in a database system are binary.
- Example of Relationships between more than two entity sets
  - Entity set *project* represents all research projects in the university
  - Each project can have multiple associated instructors and students
  - *students* work on research *projects* under the guidance of an *instructor*
  - Q: which instructor is guiding each student?
    - relationship *proj_guide* is a **ternary** (degree =3) relationship between *instructor, student,* and *project*

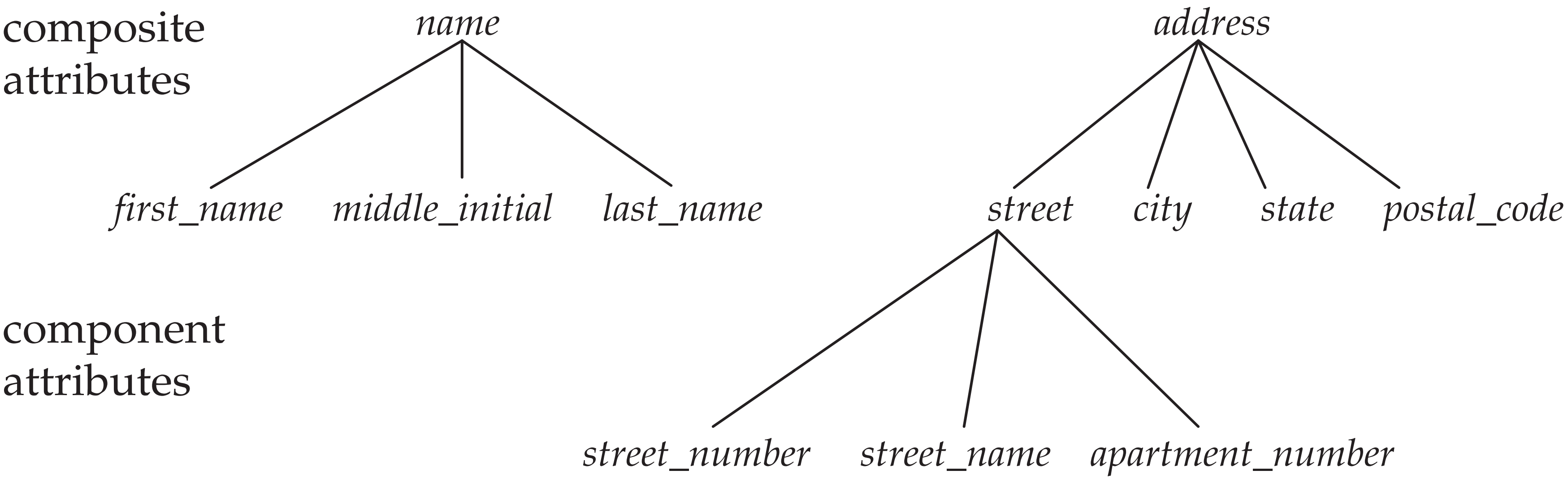


Attribute Types & Domain

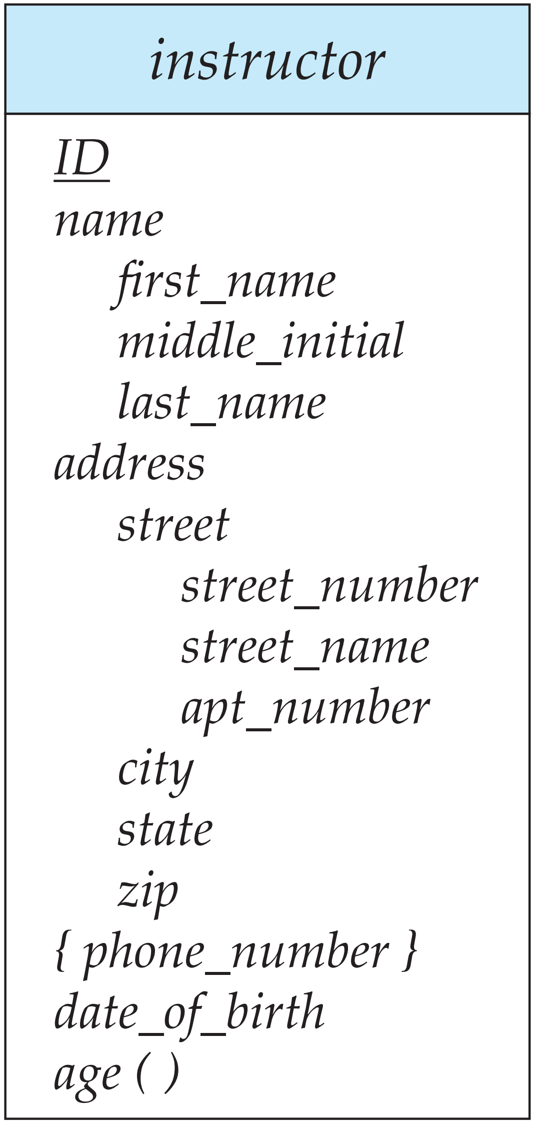
- Attribute types
  - **Simple** and **composite** attributes, for example
    - Single attribute: name or address
    - Composite attribute: first_name, mi, last_name
  - **Single-valued** and **multi-valued** attributes, for example
    - Multivalued attribute: *phone_numbers*
  - **Derived** attributes
    - Can be computed from other attributes
    - Example: adding *age*, given *date_of_birth* already exists
- **Domain (or value set)** – the set of permitted values for each attribute
  - Some examples:
    - *student_name* can be a text string of a certain length
    - *semester* can have values of {Fall, Winter, Spring, Summer}

Composite Attributes

Composite attributes allow us to divide attributes into subparts or sub-attributes (=component attributes).



Representing Complex Attributes in E-R Diagram



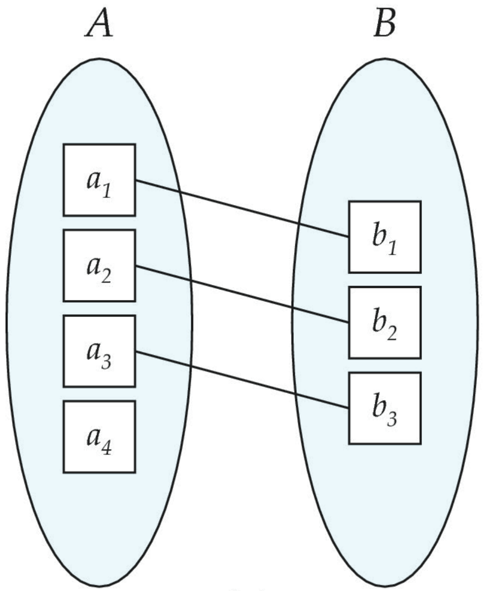
instructor

- *<u>ID</u>*
- *name*
  - *first_name*
  - *middle_initial*
  - *last_name*
- *address*
  - *street*
    - *street_number*
    - *street_name*
    - *apt_number*
  - *city*
  - *state*
  - *zip*
- *{ phone_number }*
- *date_of_birth*
- *age ( )*

Mapping Cardinality Constraints

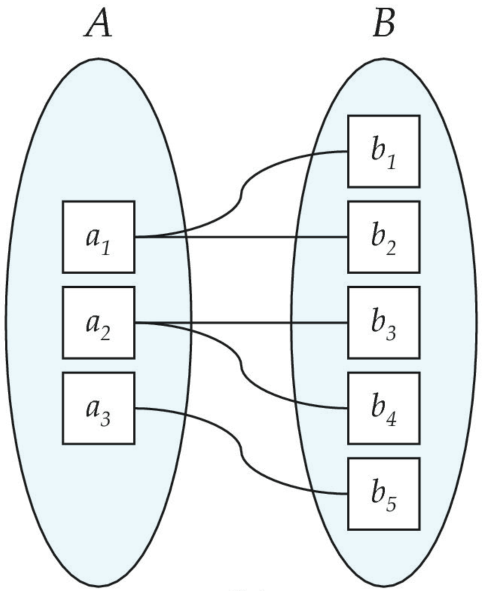
- **Mapping cardinalities** express the *number* of entities to which another entity can be associated via a relationship set. Most useful in describing binary relationship sets.
- For a binary relationship (between A and B) set the mapping cardinality must be one of the following types:
  - **One-to-one**
    - Entity in set A is associated with *at most* one entity in set B
  - **One-to-many**
    - Entity in set A is associated with *any number of* entities in set B
  - **Many-to-one**
    - Entity in set A is associated with *at most* one entity in set B
    - Entity in set B is associated with *any number of* entities in set A
  - **Many-to-many**
    - Entity in set A is associated with *any number of* entities in set B
    - Entity in set B is associated with *any number of* entities in set A

Mapping Cardinalities



(a)

One to one

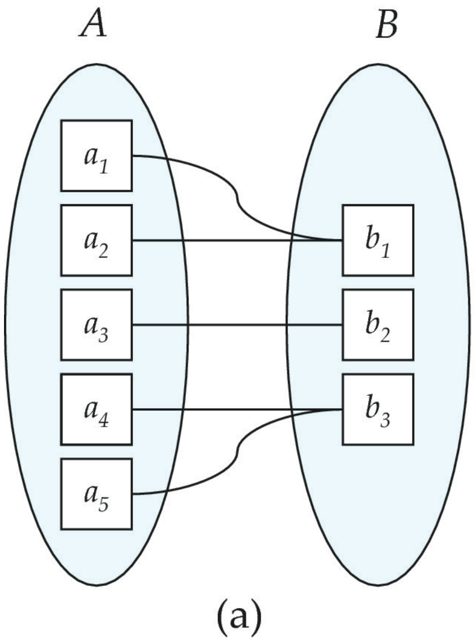


(b)

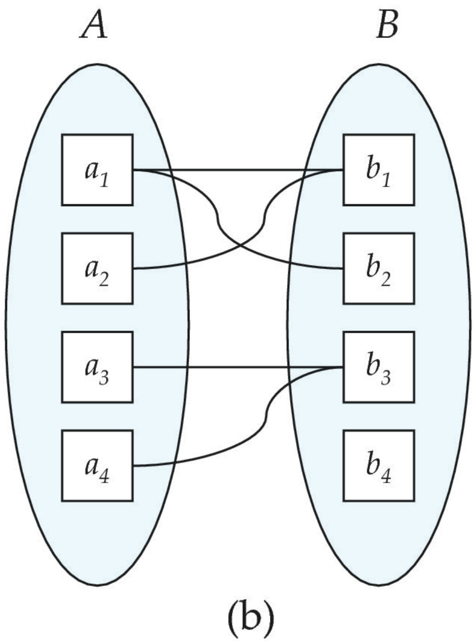
One to many

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

Mapping Cardinalities



Many to one



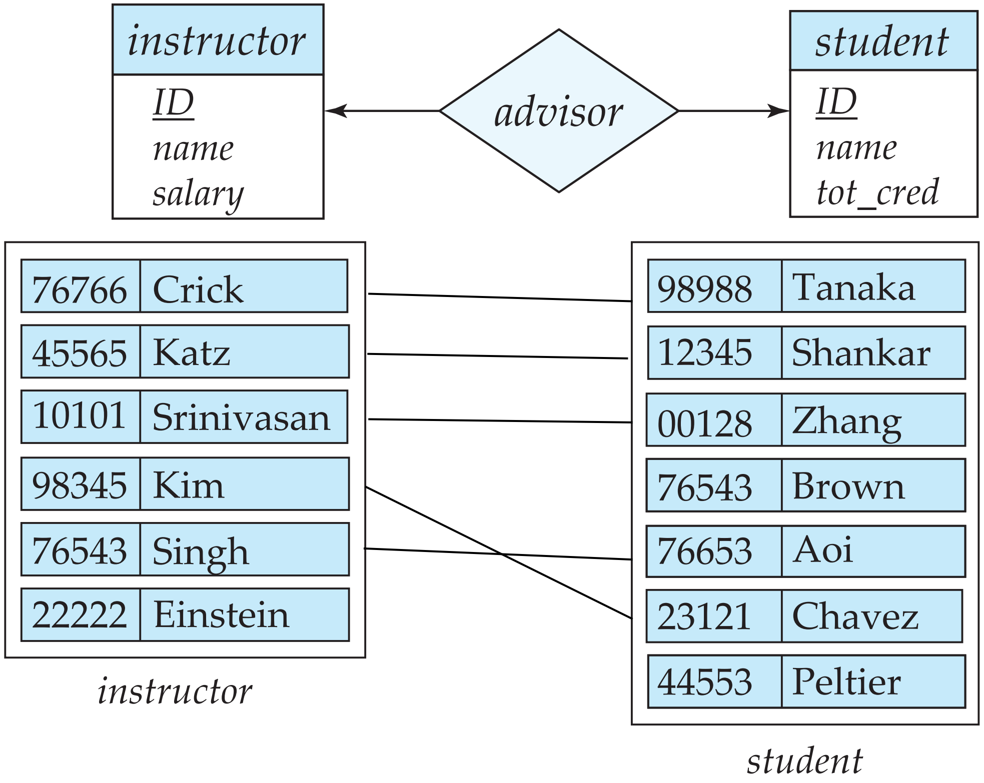
Many to many

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

Representing Cardinality Constraints in E-R Diagram

- 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*



instructor

ID
name
salary

advisor

student

ID
name
tot_cred

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

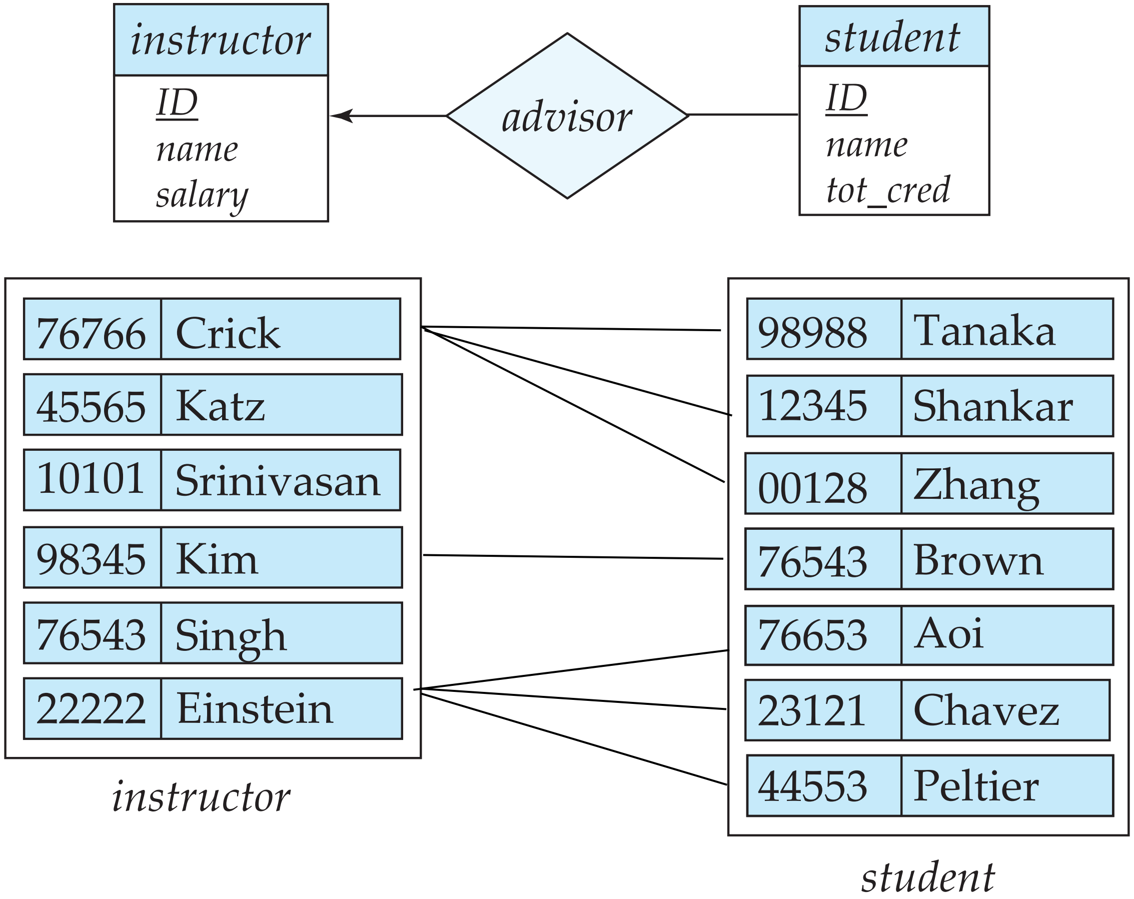
instructor

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

student

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



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

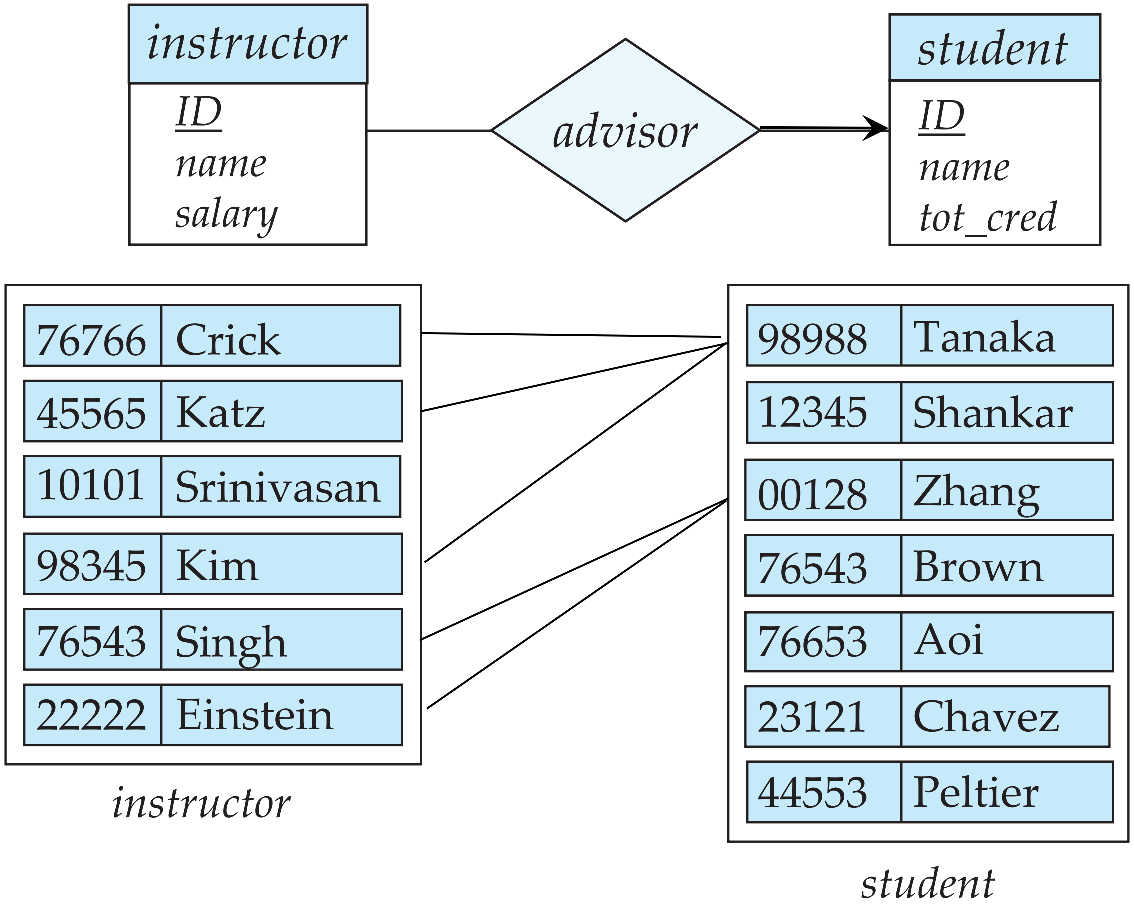
instructor

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

student

Many-to-One Relationships

- In a **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*



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

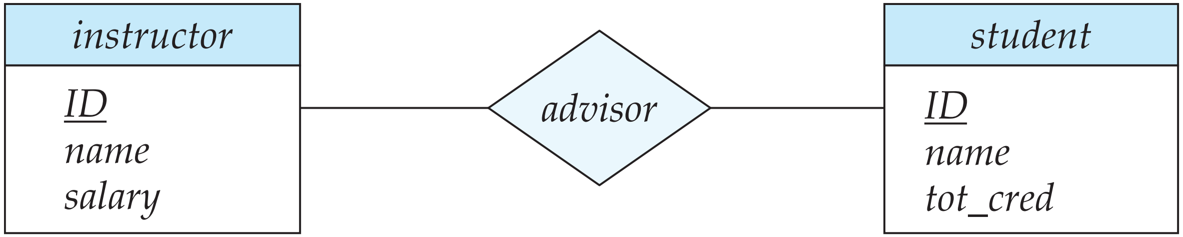
instructor

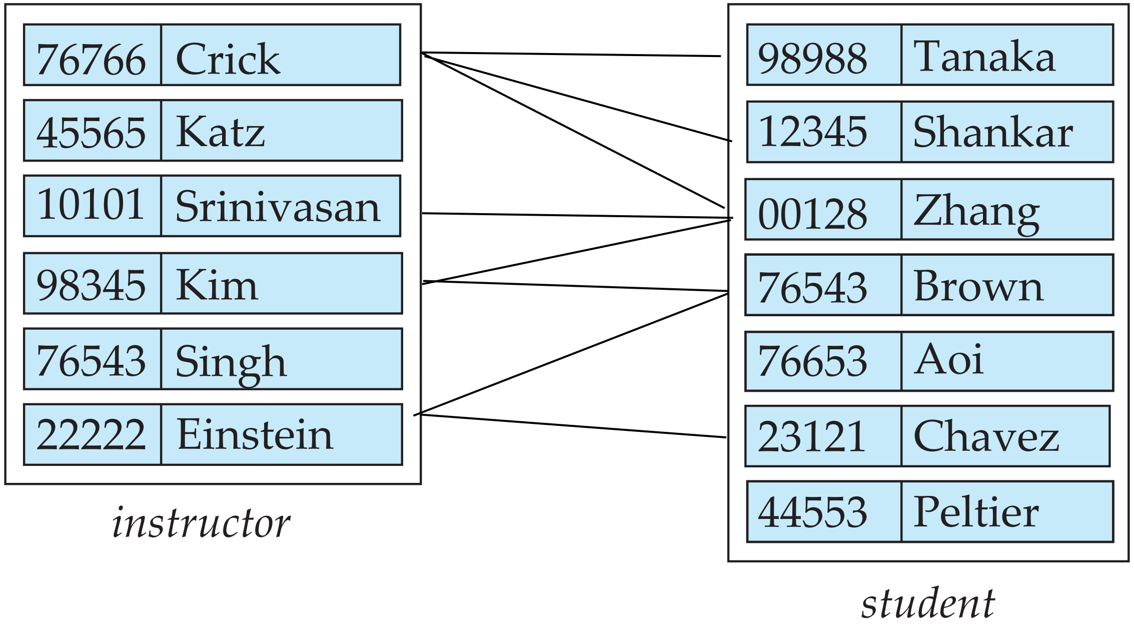
98988	Tanaka
12345	Shankar
00128	Zhang
76543	Brown
76653	Aoi
23121	Chavez
44553	Peltier

student

Many-to-Many Relationship

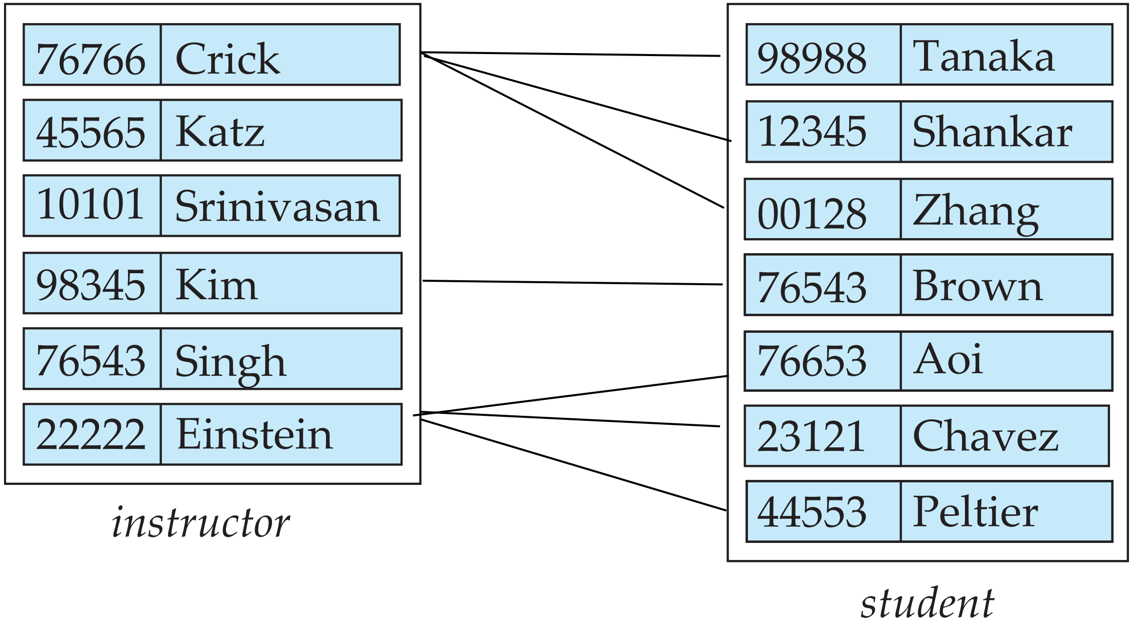
- In a **Many-to-Many** relationship between an *instructor* and a *student*
  - An instructor is associated with several (possibly 0) students via *advisor*
  - A student is associated with several (possibly 0) instructors via *advisor*





Total and Partial Participation

- **Total participation:** every entity in the entity set participates in at least one relationship in the relationship set
  - Example: participation of student in advisor relation is total
    - Every student must have an associated instructor
- **Partial participation**: some entities may not participate in any relationship in the relationship set
  - Example: participation of *instructor* in *advisor* is partial



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

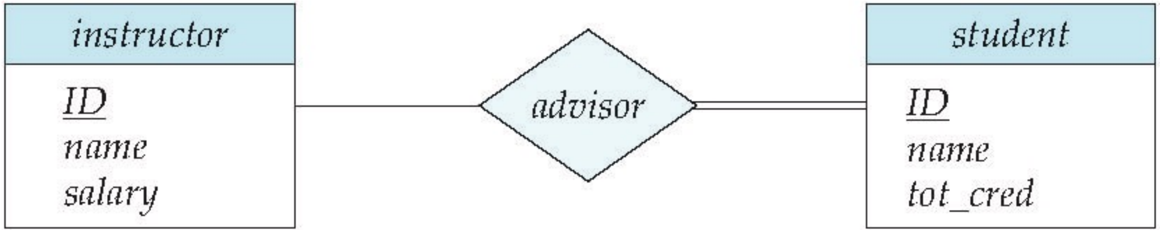
instructor

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

student

Total and Partial Participation (continued)

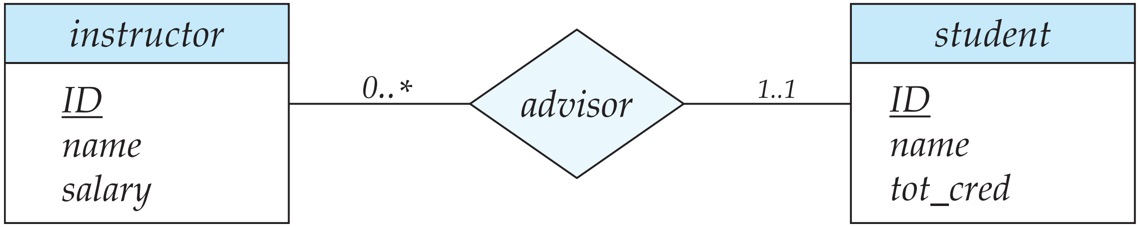
- **Total participation** indicated by *double line*.
  - Every entity in the entity set participates in at least one relationship in the relationship set
- **Partial participation** indicated by regular line.
  - Some entities may not participate in any relationship in the relationship set



Alternative Notation for Expressing Complex Constraints

- A line may have an associated minimum and maximum cardinality, shown in the form *i.h*, where *i* is the minimum and *h* the maximum cardinality
  - A minimum value of 1 indicates *total participation* (0 not allowed)
  - A maximum value of 1 indicates that the entity participates in at most one relationship
  - A maximum value of * indicates no limit

- Example



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

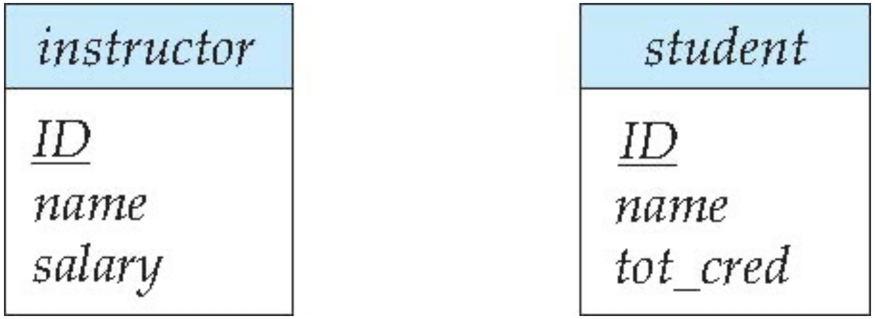
Definitions: Key Constraints

- **Definition 1**: A set of attributes is a **candidate key** or a **key** of an entity set if it is both:
  1. **Unique**: No two entities can have same values for all key attributes
  2. **Minimal:** Cannot delete any of the attributes without impairing #1

  - Ex. for instructor(ID, name, dept_name, email, …)
    - {ID}
    - {email}
    - {name, dept_name}
- **Definition 2**: A set of attributes is a **superkey** of an entity set if it is **unique** (no two entities can have same values for all key attributes), but the set is **not necessarily minimal**
  - Ex. instructor(ID, name, dept_name, email, …)
    - superkey: {ID, email}
- THUS: Any key is a superkey, but not the other way around

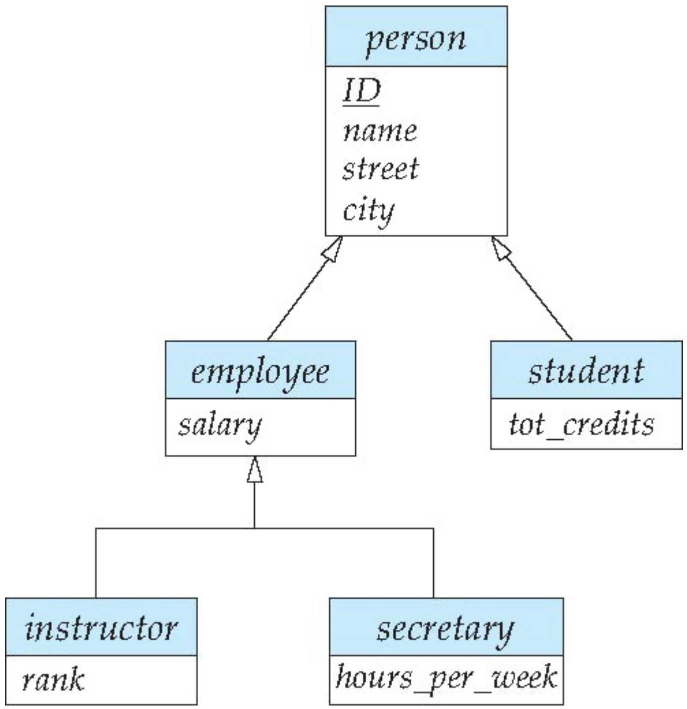
Primary Key for Entity Sets

- **Primary Key (PK)**
  - One candidate key that you (or a DBA) pick to be the **primary key**
  - It is important because -when translated into the relational model- the DBMS can enforce its uniqueness
    - You have to keep it unique over time
    - This takes work!
- The primary key is underlined in an E-R diagram (but not the candidate keys)



Generalization

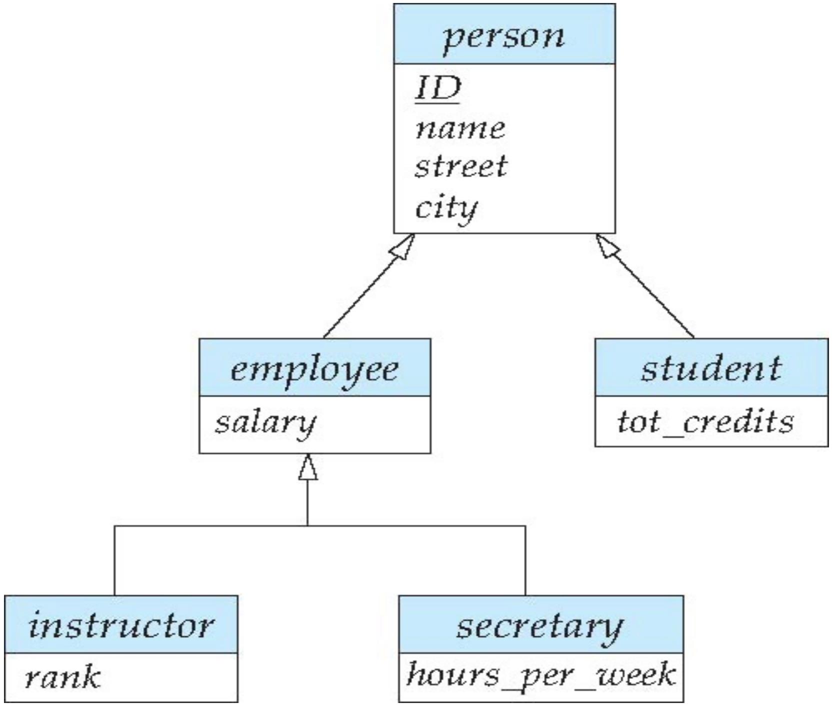
- **A bottom-up design process** – combine a number of entity sets that share the same features into a higher-level entity set.



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.
- Example, the entity set *person* may be further classified as one of the following:
  - *employee*
  - *student*

Specialization Example

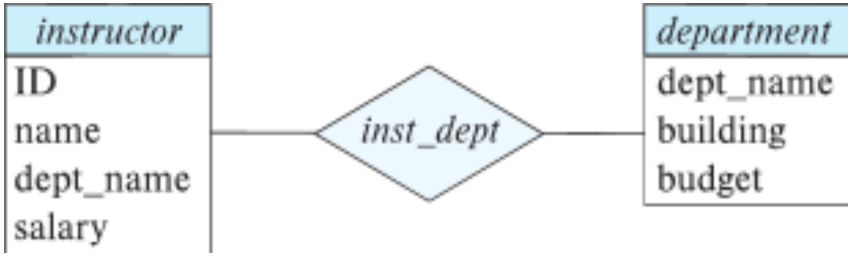


Generalization & Specialization

- Specialization and generalization are simple inversions of each other
  - Represented in an E-R diagram in the same way. New levels of entity representation are distinguished (specialization) or synthesized (generalization)
  - The terms specialization and generalization are used interchangeably.
- Specialization and generalization may be total or partial type
  - **Total** - each high-level entity *must* belong to a low-level entity set
  - **Partial** - Some higher-level entities may not belong to any lower-level entity set.
- **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.

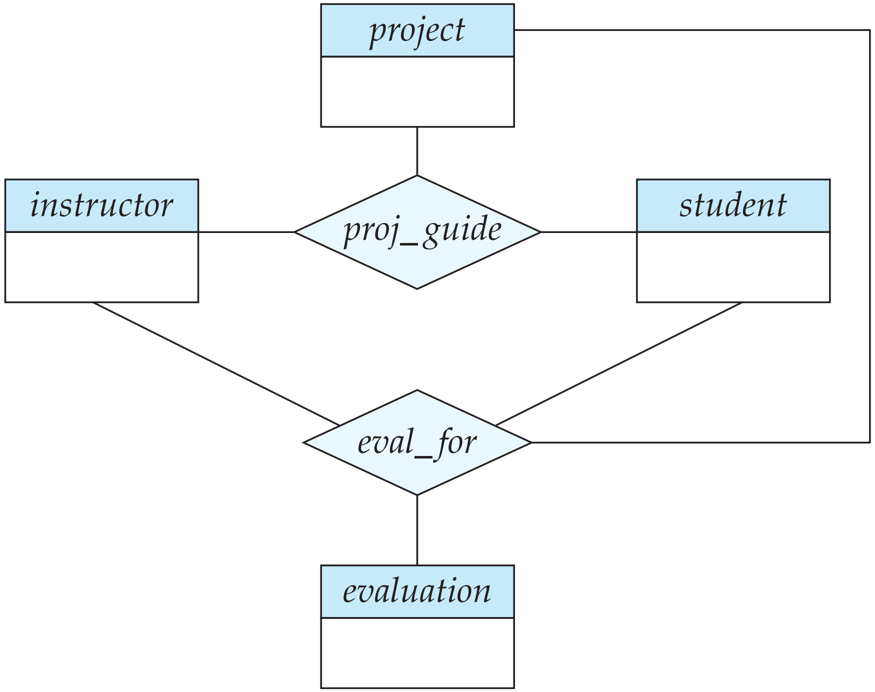
Avoid Redundant Attributes

- Suppose we have entity sets
  - *instructor*
    - with attributes: *ID*, *name*, *dept_name, salary*
  - *department*
    - with attributes: *dept_name, building, budget*
- We model the fact that each instructor has an associated department using a relationship set *inst_dept*
- The attribute *dept_name* in *instructor replicates* information present in the relationship and is therefore *redundant* and needs to be removed.



Aggregation

- One limitation of the E-R model is that it cannot express relationships among relationships.
  - Consider the ternary relationship *proj_guide*, which we saw earlier
  - Suppose we want to record evaluations of a student by a project guide on a project

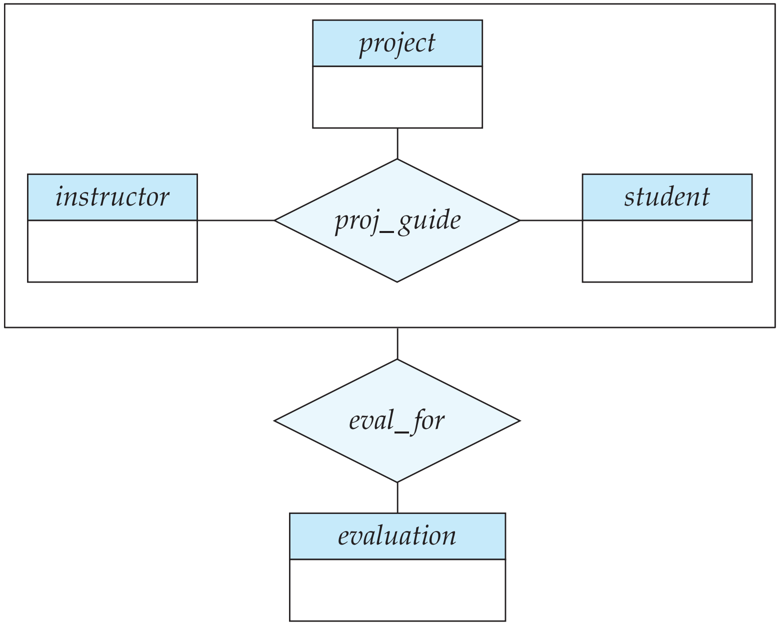


Aggregation (continued)

- Issue: Relationship sets *eval_for* and *proj_guide* represent overlapping (redundant) 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
- Q: What can we do to get rid of that redundancy?
  - A: Eliminate redundancy via *aggregation:*
    - Treat relationship *proj_guide* as an *abstract* entity
    - Allow relationships with that abstract entity
- Will show on the next slide

Aggregation (continued)

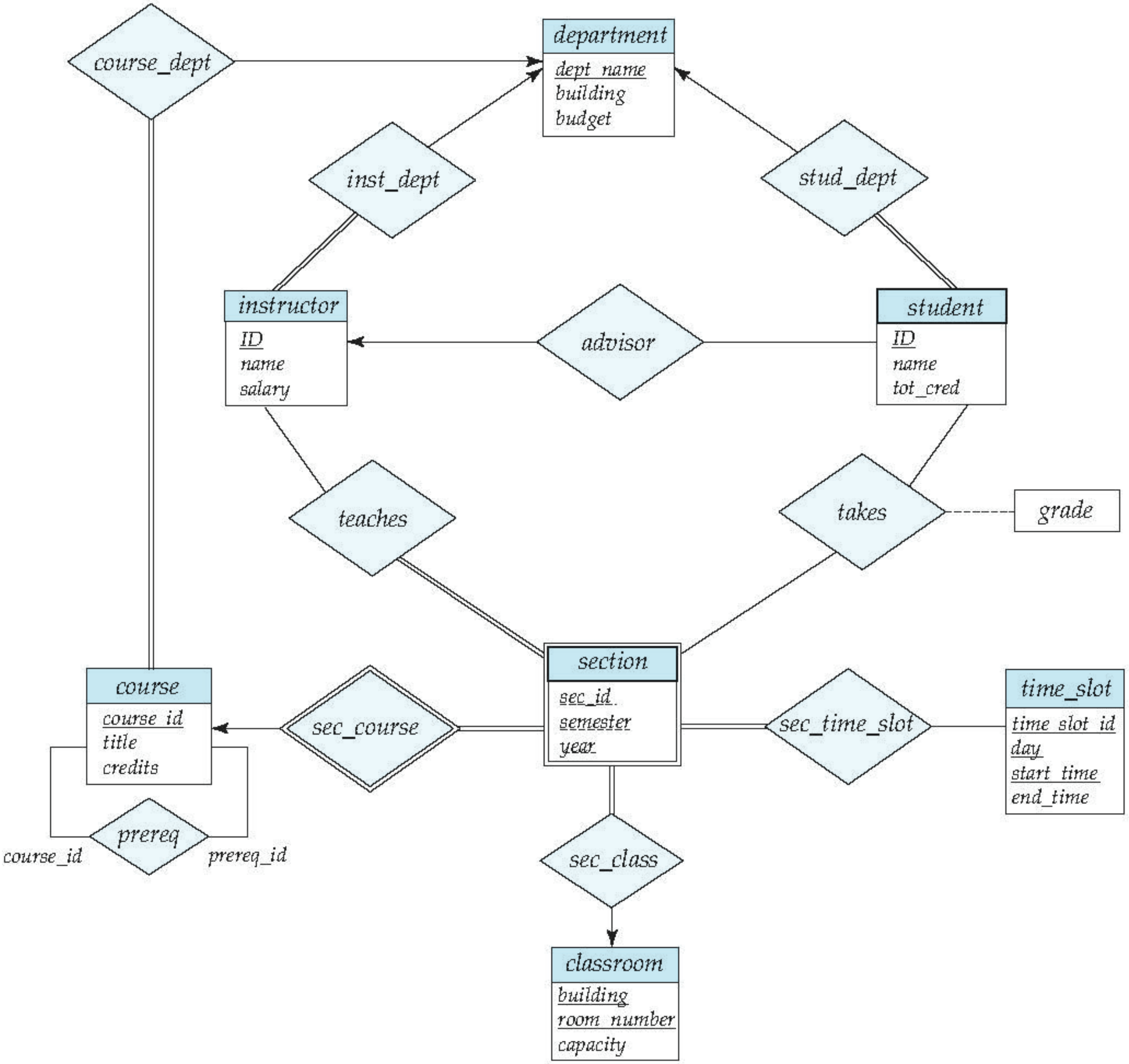
- **Aggregation** as an abstraction through which relationships are treated as entities.
  - Thus, for our example, we regard the **relationship** set *proj_guide* (relating the entity sets *instructor*, *student*, and *project*) as an **entity** set called *proj_guide*.
- The following diagram represents aggregation:
  - A student is guided by a particular instructor on a particular project
  - A student, instructor, project **combination** may have an associated evaluation



University Database

- In this course we will be using a university database to illustrate all the concepts
  - It is published on Canvas
- Data consists of information about:
  - Students
  - Instructors
  - Classes
- Application program examples:
  - Add new students, instructors, and courses
  - Register students for courses, and generate class rosters
  - Assign grades to students, compute Grade Point Averages (GPA) and generate transcripts

E-R Diagram for the University Database



E-R Design Challenges

Design Alternatives

- 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.

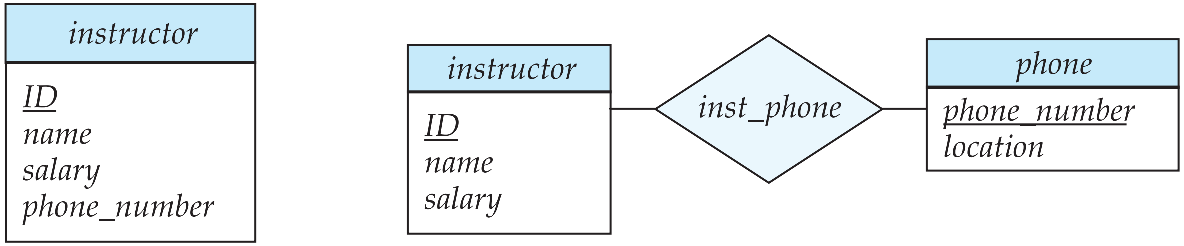
Design Choices

- Should a concept be modeled as an entity or an attribute?
- Should a concept be modeled as an entity or a relationship?
- Identifying relationships: Binary or ternary? Use Aggregation?

- Limits of the E-R Model:
  - A lot of data semantics & requirements can (and should) be captured
  - But some things just cannot be captured in E-R diagrams.

Use Entities vs. Attributes

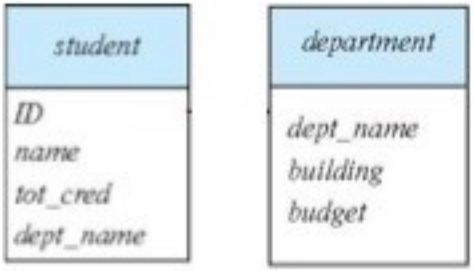
- Use of entity sets vs. attributes



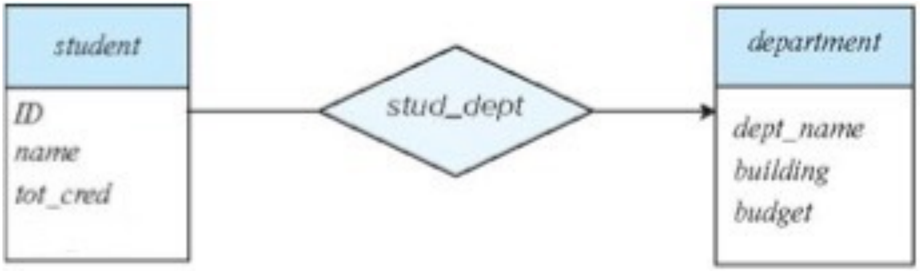
- Should a phone number be modeled as an attribute or an independent (related) entity?
  - Phone number could be a character-string attribute of Instructor entities
  - Will not work in following cases
    - If we want to model that several instructors use the same phone, phone number must be a separate entity
    - If we want to support queries of phone number parts (e.g., area code, phone number can be modeled as a separate entity with sub-attributes)
    - If there can be multiple phone numbers per instructor, phone number must be an entity

Erroneous Use of PK

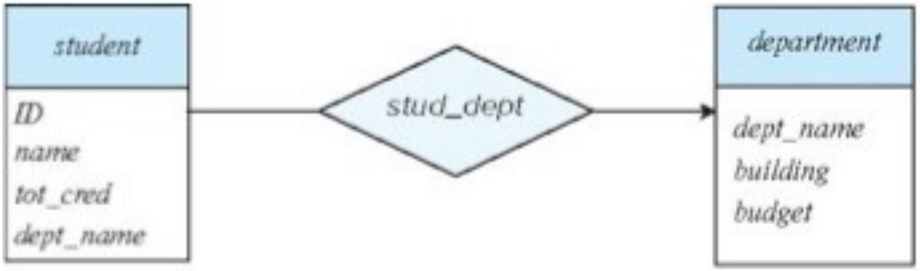
- Common mistake 1: Use of Primary Key (dept_name) of an entity set (department) as an attribute of a different entity set (student)



- Better way: dept_name should not be an attribute of student, *relationship* stud_dept is the correct way. It is always good to have relationship explicit.

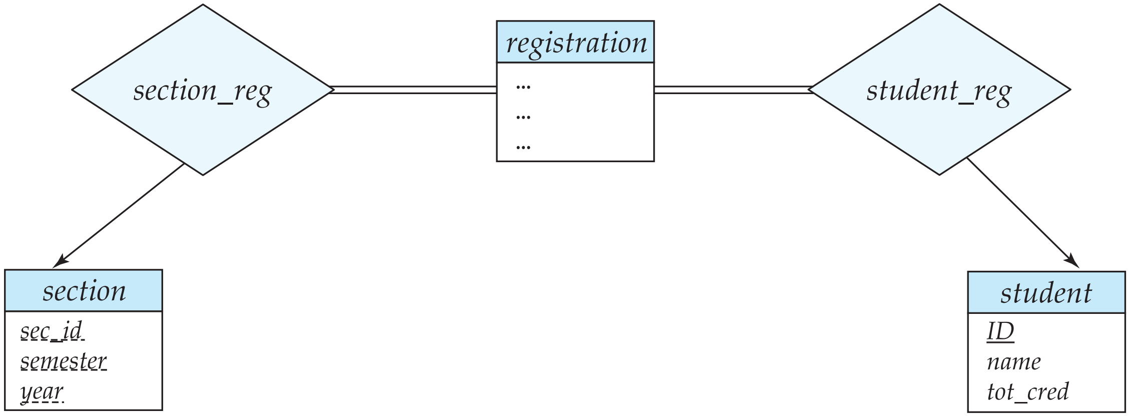


- Common mistake 2: To have both, attribute and relationship is redundant – should also be avoided



Entities vs. Relationship sets

- Compare: *take* relationship in University diagram (shown earlier today) with introducing a new *registration* entity



- Considerations:
  - Pro: *take* relationship
    - More compact
    - Preferable in most cases
  - Pro: *registration* entity
    - Easier to add additional info to course-registration record
- Rule of thumb: designate a relationship set to describe an action (e.g. take) that occurs between entities

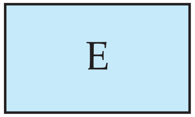
Some Things E-R Just Doesn’t Model Well

- Multi-valued attributes
- Numeric constraints
  - E.g. managers manage between 3 and 7 employees
- Functional dependencies
  - Fields that determine the value of other fields
    - e.g. that a department can’t order two different parts from the same supplier
- Inclusion dependencies
  - Values in one attribute must be a subset of the values in another
- More general constraints
  - Managers must make 10% more than any of their employees
- All of that takes sometimes DBMS tools
  - Triggers/table constraints/application code!

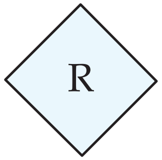
Putting It All Together

- You attempt to capture all the described features in an E-R diagram
  - Determine entities, attributes, relationships
- You (almost certainly will) refine it several times, e.g.
  - moving attributes to the right place, making sure every constraint is correctly marked
  - ask clarifying questions as needed
  - test it out by thinking of example instances
  - explicitly state in English what you couldn’t model in E-R
- The resulting E-R diagram has several important uses
  - you can translate it directly into a set of relations, which can be implemented during database creation stage (via SQL DDL statements)
    - (come back next week!)
  - ER diagrams are great documentation
    - succinct view of an entire information system
    - smart people (like you) will be able to pick one up and understand volumes!

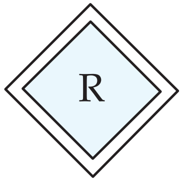
Summary of Symbols Used in E-R Notation



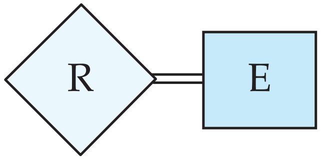
entity set



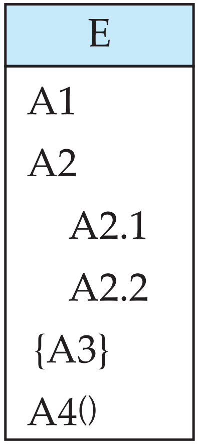
relationship set



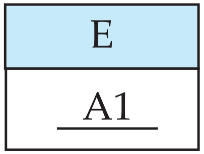
identifying
relationship set
for weak entity set



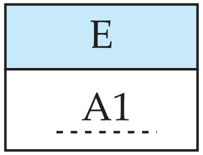
total participation
of entity set in
relationship



attributes:
simple (A1),
composite (A2) and
multivalued (A3)
derived (A4)

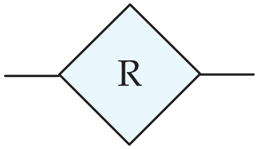


primary key

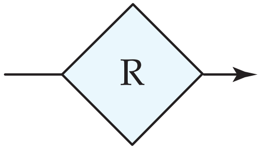


discriminating
attribute of
weak entity set

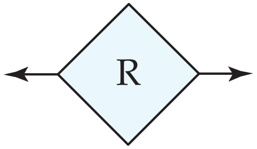
Symbols Used in E-R Notation (continued)



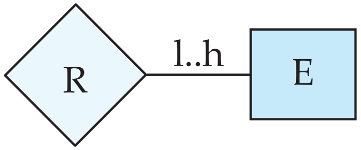
many-to-many relationship



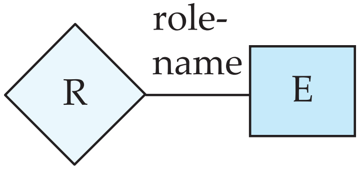
many-to-one relationship



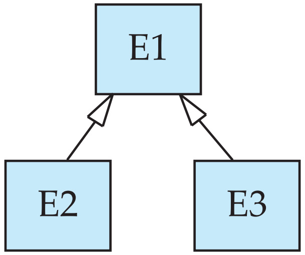
one-to-one relationship



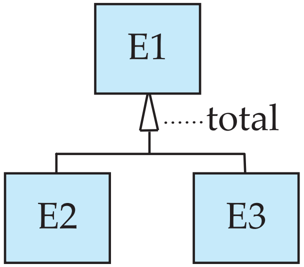
cardinality limits



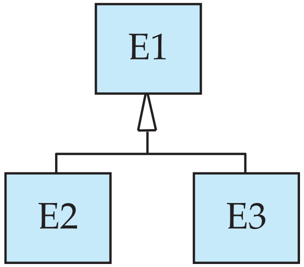
role indicator



ISA: generalization or specialization



total (disjoint) generalization



disjoint generalization

Alternative ER Notations

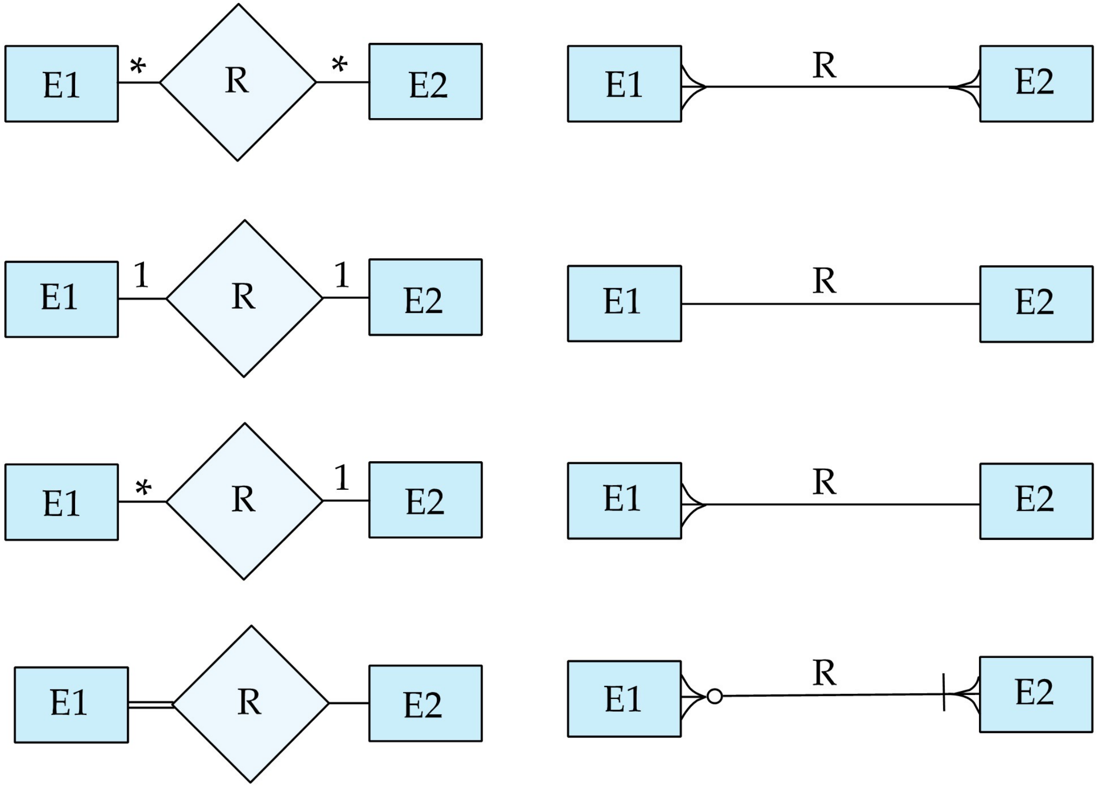
Chen | IDE1FX (Crows feet notation)

many-to-many relationship

one-to-one relationship

many-to-one relationship

participation in R: total (E2) and partial (E1)



Lab: Developing E-R Diagram

TO SUBMIT: part 2 of the Lab (only) upload to Canvas a Word Doc

Answer the following question: What does this diagram enforce and what it does not enforce in relationships between package, sender and receiver.

Grading rubric:

a. Full credit: explanation has to be understandable and make sense

b. Partial credit: It does not make total sense

c. Zero credits: not submitted or submitted a set of words that TA cannot understand after they read it 2 times