CS2102: Database Systems

ER Model

Announcement

Announcement

Project

- Groups are set, please check on Canvas and report any discrepancy
- Project (Part 1) is out on Canvas (`Project (Part 1).pdf`)
  - **Deadline:** *Week 7*, Saturday, 4 March 2023 (*12:00 Noon*)

Office Hours

- Pre-booked consultation session
- Schedule available on Canvas
  - Will be updated as more TAs indicate their slots

Midterm

- **Week 7:** Monday, 27 February 2023 (*12:30 - 13:30*)
- MPSH 2A & MPSH 2B

Recap

Recap

SQL

What Is It?

- The standard language for RDBMS
  - Different language groups: DDL, DML, DQL, DCL, TCL

- **DDL:** `CREATE TABLE`, `ALTER TABLE`, `DROP TABLE`
- **DML:** `INSERT`, `UPDATE`, `DELETE`

Key Challenge

- Specification of integrity constraints
  - `NOT NULL`, `UNIQUE`, `PRIMARY KEY`, `FOREIGN KEY`, `CHECK`

- Specification actions in case of foreign key constraint violations *(`ON UPDATE`/`ON DELETE`)*
- Relaxed checks of violations with deferrable constraints

Recap

SQL
- What
- **Table**
- Rows

Constraints
Principles

SQL

Table Syntax

CREATE

```
CREATE TABLE <table_name> (
  <attr> <type> [<column_constraint>],
  <attr> <type> [<column_constraint>],
    :
  [<table_constraint>],
  [<table_constraint>],
    :
);
```

DROP

```
DROP TABLE [IF EXISTS] <table_name>;
```

Recap

SQL

Rows Syntax

INSERT

```
INSERT INTO <table_name> [(<attr>, ...)]
  VALUES (<values>, ...) [, (<values>, ...)];
```

UPDATE

```
UPDATE <table_name>
  SET <attr> = <value> [, <attr> = <value>]
  [WHERE <condition>];
```

DELETE

```
DELETE FROM <table_name> [WHERE <condition>];
```

Recap

SQL
Constraints
Principles

Integrity Constraints

Types

Type	Column	Table	Condition
Not-NULL	NOT NULL	-	IS NOT NULL
Unique	UNIQUE	UNIQUE(A_1,A_2,...)	x.A_i <> y.A_i
Primary Key	PRIMARY KEY	PRIMARY KEY(A_1,A_2,...)	UNIQUE & NOT NULL
Foreign Key	REFERENCES R_1(B)	FOREIGN KEY (A_1,A_2,...) REFERENCES R_1(B_1,B_2,...)	The tuple exists in R_1 or the tuple contains NULL value
General	CHECK (c)	CHECK (c)	Condition c does not evaluate to False

Recap

Principles

Principle of Acceptance

Definition

Perform the operation if the condition ***evaluates to True***.

- Used in `WHERE` clause *(and relational algebra)*
  - *e.g., perform the update/delete/query*

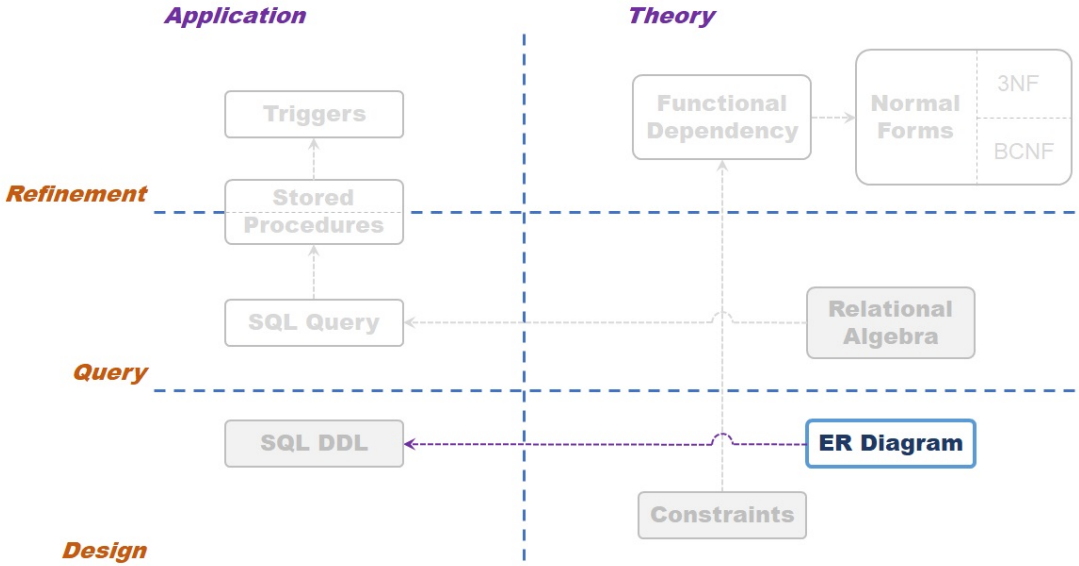
Principle of Rejection

Definition

Reject the operation if the condition ***evaluates to False***.

- Used in integrity constraints
  - *e.g., perform the insertion*

Roadmap


Application
Theory
Triggers
Functional Dependency
Normal Forms
3NF
BCNF
Refinement
Stored Procedures
SQL Query
Relational Algebra
Query
SQL DDL
ER Diagram
Constraints
Design

Roadmap

Overview
Question
Design

Overview

Entity Relationship

- ER Diagrams
- Entity Sets and Attributes
- Relationship Sets

Relationship Constraints

- Cardinality
- Participation

Relational Mapping

- ER Diagram to Schema

Extended Notations

- ISA Hierarchies
- Aggregation

Roadmap

Overview
Question
Design

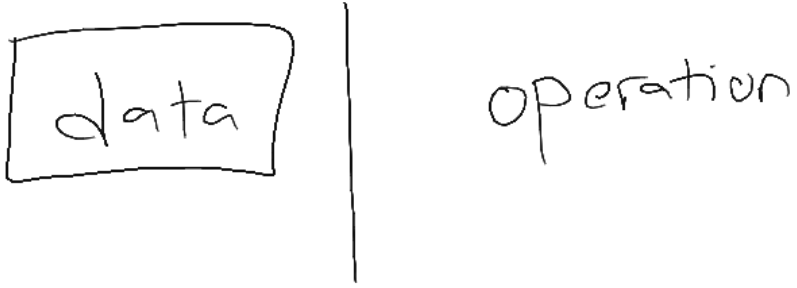
model ↔

Open Questions

We *Sneakily* Skipped a Step

Flight

- Where does the database schema come from?
- What tables with which attributes do we need?
- What data integrity constraints are required?
- What table names, attribute names, data types, *etc* do we need?



Roadmap

xkcd Little Bobby Table

Overview
Question
Design

Database Design Process

- **Requirement Analysis**
  - Identification and collection of user needs
    (e.g., data / application / performance requirements)
- **Conceptual Database Design**
  - Capturing requirements using conceptual model
  - **RDBMS:** *Entity Relationship Model (ER Model)*
- **Logical Database Design**
  - Mapping conceptual model to logical schema
  - **RDBMS:** *ER Model → Relational Schema*
- **Schema Refinement**
  - Checking schema / tables for redundancies and anomalies
- **Physical Database Design**
  - Implementing database based on final schema
  - Consideration of performance requirements
- **Security Design**
  - Identification of users and user groups and their permissions to access which parts of the data

Entity Relationship

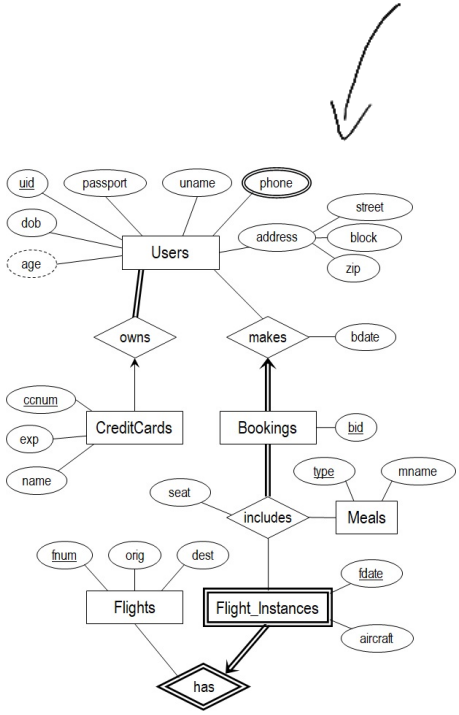
Entity Relationship

Requirements
ER Model
Entities
Attributes
Relationships

Requirements Analysis

Online Airline Reservation Systems

> Users need to be able to make bookings from an origin to a destination airport which may comprise multiple connecting flights. We record the booking date. Each flight has a flight number, the origin and destination airport, the distance in kilometers, the departure and arrival time, and the days of the week the flight is in operation.
>
> A flight instance is the actual scheduled flight on a given day together with the assigned aircraft type. For example, flight SQ231 flies daily from Singapore to Sydney, typically with a Boeing 777-300ER *(code: B77W)*.
>
> For a valid booking, we need the user's name, sex, address, phone number(s), and the passport number. Users are only able to pay via credit card. When making a booking, the user can select the class, the seat number, as well as meal preferences *(if available)*.



Entity Relationship

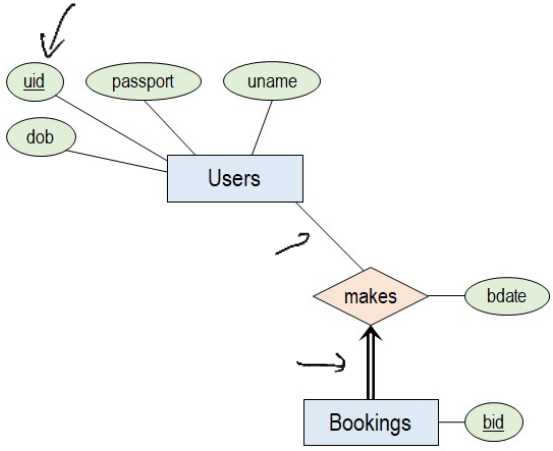
ER Model

Preliminary

- Most common model for conceptual database design
- Developed by Peter Chen in 1976
- Visualized using **ER Diagrams**

Core Concepts

- All data is described in terms of
  - **entities** *(rectangle, nouns)*
  - **relationships** *(diamond, verbs)*
- **Attributes** describes information about entities and relationships
- Elements are connected via lines
- Certain data constraints can be described using *additional annotations*



Entity Relationship

Person { Persons / People

Entities and Entity Sets

Entity

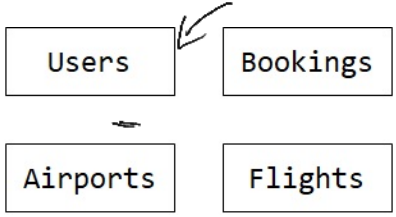
An **entity** is a *representation* of real-world objects that are *distinguishable* from other objects *(e.g., user, airport, flight, or booking)*.

Users need to be able to make ***bookings*** from an origin to a destination ***airport*** which may comprise multiple connecting ***flights***. Each flight has a flight number, *[...]*

Entity Set

An **entity set** is a collection of entities of the *same type*

- Represented by a ***rectangle*** in ER diagrams
- Names are typically ***nouns***
  - Should not be *proper nouns* *(e.g., Albert Einstein)*
  - As a good practice, make the name *plurals*



Entity Relationship

Attributes

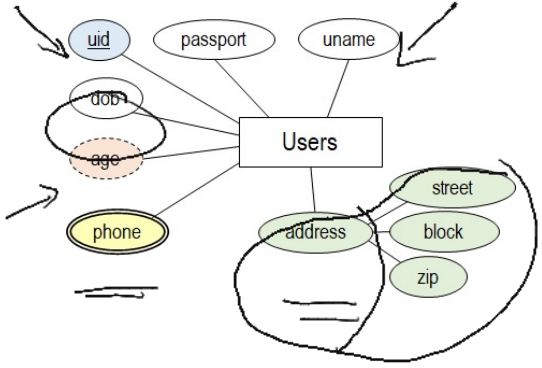
Attribute

Attributes are specific information *describing an entity*
- Represented by an **oval** in ER diagrams

For a valid booking, we need the ***user's name***, ***sex***, ***address***, ***phone number(s)***, and the ***passport number***. Users are only able to pay via credit card. *[...]*

Subtypes:

1. **Key Attributes** : ***underlined***
   - Uniquely identifies each entity
2. **Composite Attributes** : ***composed of other ovals***
   - Composed of multiple other attributes
3. **Multivalued Attributes** : ***double-lined***
   - One or more values for a given entity
4. **Derived Attributes** : ***dashed line***
   - Derived from other attributes



Entity Relationship

Requirements
ER Model
Entities
Attributes
Relationships
- Roles
- Degree
- General

Relationships and Relationships Sets

Relationship

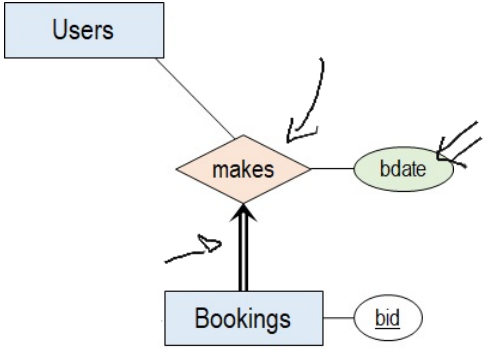
A **relationship** is an association among *one or more* entities

Users need to be able to ***make bookings*** *[...]* We record the ***booking date****. [...]*

Relationship Set

A **relationship set** is a collection of relationships of the *same type*

- Represented by a **diamond** in ER diagrams
- May have their own attributes *(further describe the relationship)*
- Names are typically ***verbs***



Additional Annotations

- ***Roles***
- ***Degrees***
- ***Cardinalities***
- ***Participation***
- ***Dependencies***

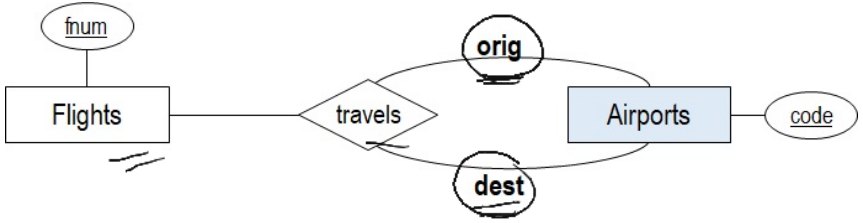
Entity Relationship

Relationships and Relationships Sets

Relationship Roles

- Descriptor of an entity set's participation in a relationship
- Most of the time *implicitly* given by the name of the entity sets
- Explicit role label only in case of *ambiguities*

> *[...]* from an ***origin*** to a ***destination airport*** which may comprime multiple connecting ***flights*** *[...]*



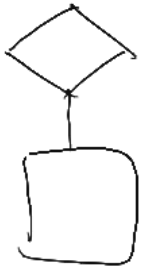
[#]This is missing from the ER shown in the *requirements analysis* due to space limitation.

Entity Relationship

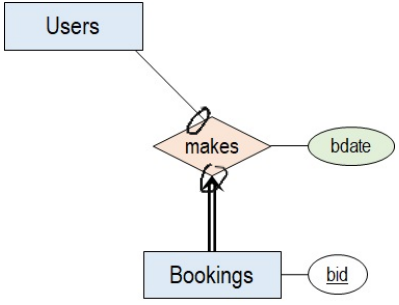
Relationships and Relationships Sets

Degree of Relationship Sets (Arity)

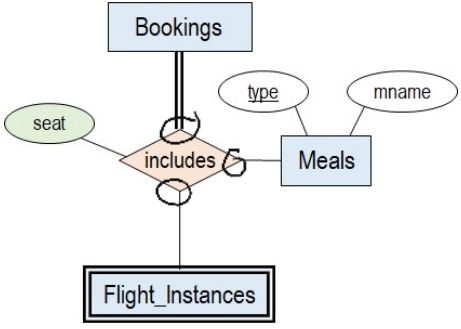
- The number of entity sets involved in the relationship set
  - *n*-ary = *n* entity
  - In principle, there is no limitation on the number of entity sets involved



Binary Relationship Set (n = 2)



Ternary Relationship Set (n = 3)



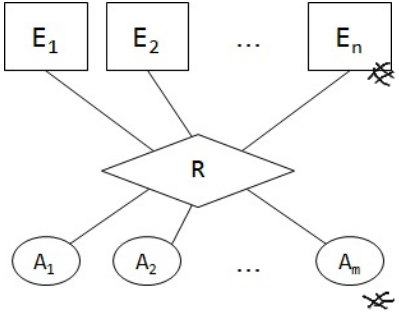
Entity Relationship

Relationships and Relationships Sets

General *n*-ary Relationship Set R

- *n* entity sets involved: E_1, E_2, ..., E_n
- *m* attributes: A_1, A_2, ..., A_m

ER Diagram



> ***"In typical modeling, binary relationships are the most common and relationships with n>3 are very rare"***
>
> -- *Peter Chen* (2009)

Relationship Constraints

Relationship Constraints

Preliminary

Types of Relationship Constraints

1. **Cardinality Constraints**
   - Describes an *upper bound* to the number of times an entity can participate in a relationship *(either 1 or ∞)*
   - Upper limits of 1 are called ***key constraints***
2. **Participation Constraints**
   - Describes a *lower bound* to the number of times an entity can participate in a relationship *(either 0 or 1)*
   - Lower limits of 0 are called ***partial participation constraints***
   - Lower limits of 1 are called ***total participation constraints***
3. **Dependency Constraints**
   - Entity set that does **NOT** have *its own key* *(its keys are called **partial keys**)*
   - Must have *identifying* relationship set connecting to the *owning* entity set

Relationship Constraints

Cardinality Constraint

Basic Cardinality Constraints

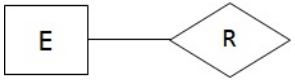
> The following cardinality constraints are only for *binary* relationship

1. **Many-to-Many** *(e.g., a flight can be performed by different aircrafts; an aircraft can perform different flights)*
2. **Many-to-One** *(e.g., a user can make many bookings, but each booking is done by one user)*
3. **One-to-One** *(e.g., a user is associated with one set of credit card details, and vice versa)*

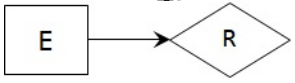
ER Diagram

- Default upper limit is ∞
- Upper limit of 1 is specified by an arrow *(i.e., →)* instead of a line
- These can work with *n*-ary relationship

Upper Limit = ∞



Upper Limit = 1



Relationship Constraints

Cardinality Constraint

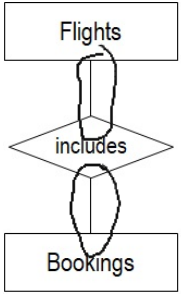
Many-to-Many

Example

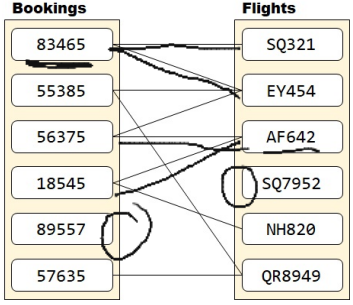
Each booking can include **0 or more** flights* and each flight can be part of **0 or more** bookings.

attributes omitted

ER Diagram



Visualization



Note

- There can be a booking with different flights *(e.g., 83465)*
- There can be a flight with different bookings *(e.g., EY454)*
- There can be a booking without flight *(e.g., 89557)*
- There can be a flight without booking *(e.g., SQ7952)*

*A booking with 0 flights might not be meaningful and we will improve on that.

Relationship Constraints

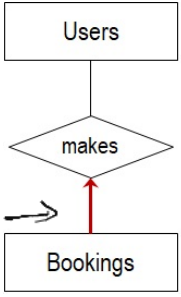
Cardinality Constraint

Many-to-One

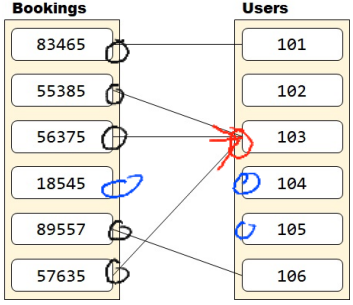
Example

Each user can make **0 or more** bookings and each booking can be made by **at most 1** user*.

ER Diagram



Visualization



Note

- There are no booking with different users
- There can be a user with different bookings *(e.g., 103)*
- There can be a booking without user *(e.g., 18545)*
- There can be a user without booking *(e.g., 102)*

*Still not perfect yet because what is a booking that is made by no user? We will also improve on that.

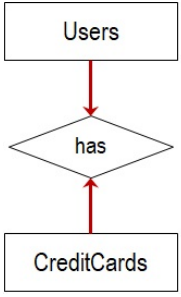
Relationship Constraints

Cardinality Constraint

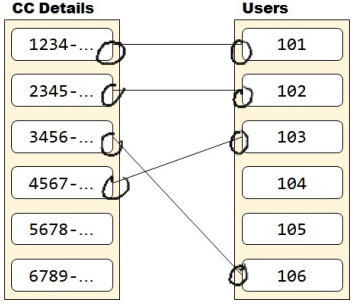
One-to-One

> **Example**
>
> Each user can provide **at most 1** credit card details and each credit card details can is associated to **at most 1** *.

ER Diagram



Visualization



Note

- There are no user with different credit card
- There are no credit card with different user
- There can be a user without credit card *(e.g., 102)*
- There can be a credit card without user *(e.g., 5678-...)*

*May be too strict, because a family may share a single credit card.

Relationship Constraints

Participation Constraint

Basic Participation Constraints

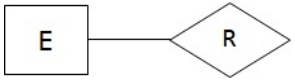
- Cardinality constraints are limited
  - A booking can include 0 flights
  - A booking can be done by 0 users
  - A credit card details does not need to be associated with a user

Limitation

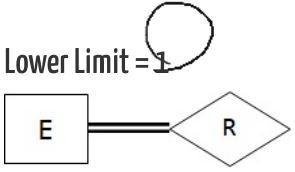
An entity does not have to *participate* in a relation.

- Cardinality constraints only specify *upper bound*
- Participation constraints specify *lower bound*
  - Default lower limit is 0
  - Lower limit of 1 can be specified by double line *(i.e., =)* instead of a single line

Lower Limit = 0



Lower Limit = 1



Relationship Constraints

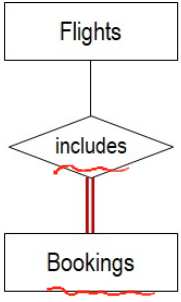
Participation Constraint

Total Participation

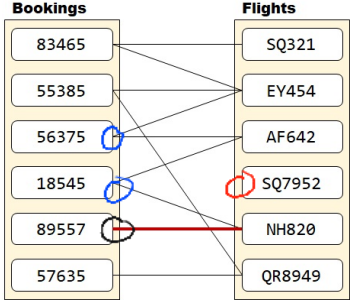
Example

Each user booking includes **at least 1** flight and each flight can be part of **0 or more** bookings.

ER Diagram



Visualization



Note *(comparison)*

- There can be a booking with different flights *(e.g., 83465)*
- There can be a flight with different bookings *(e.g., EY454)*
- **There are no booking without flight**
- There can be a flight without booking *(e.g., SQ7952)*

Relationship Constraints

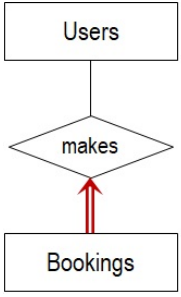
Cardinality Constraint

Key + Total Participation

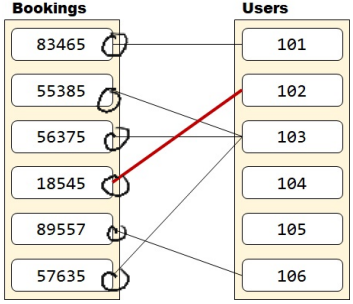
Example

Each user can make **0 or more** bookings and each booking can be made by **exactly 1** user.

ER Diagram



Visualization



Note *(comparison)*

- There are no booking with different users
- There can be a user with different bookings *(e.g., 103)*
- **There are no booking without user**
- There can be a user without booking *(e.g., 102)*

Relationship Constraints

Dependency Constraint

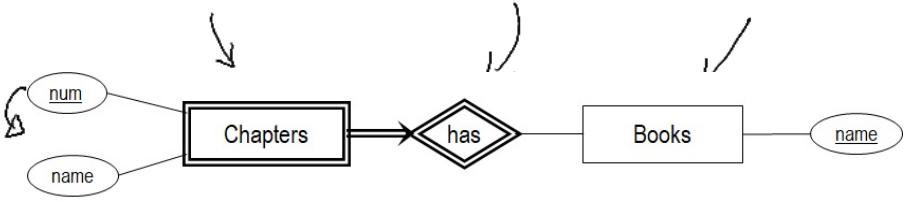
Weak Entity Set

- Entity set that does **NOT** have *its own key*
  - Its own key is called **partial key**
    - Cannot uniquely identify entity
    - Can only uniquely identify with the help of primary key from **owner entity**

- Its existence is **dependent** on the existence of its owner entity
- It is connected to owner entity via **identifying relationship set**

> **Example**
>
> A book chapter is dependent on a book.

ER Diagram



Relationship Constraints

Dependency Constraint

Requirements

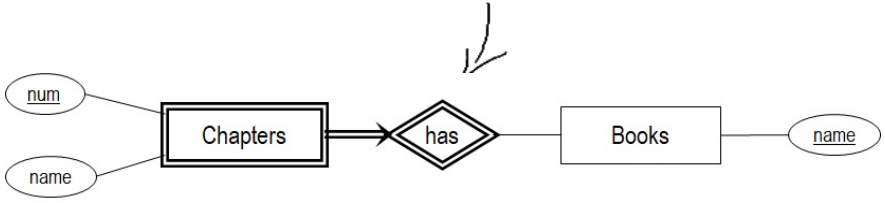
Lines

- **Many-to-one relationship** *(i.e., identifying relationship)* from weak entity set to owner entity set
- Weak entity set must have **total participation constraint** in identifying relationship

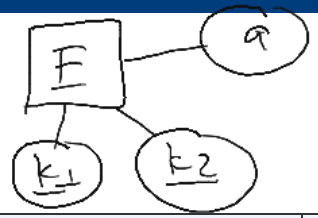
Partial Key

- Set of attributes of weak entity set that uniquely identifies a weak entity for a **a given owner entity**
  - ***Example:*** Chapter number cannot uniquely identify the chapter name *without* knowing the book name

ER Diagram

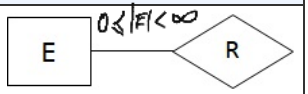
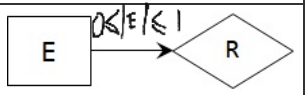
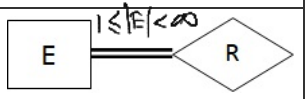
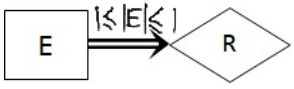


Relationship Constraints



Preliminary
Cardinality
Participation
Dependency
Summary

Summary of Relationship Constraints

Name	Constraint	Diagram
Unconstrained	Each instance of E may participate in *0 or more* instance of R	
Key Constraint	Each instance of E participates in *at most 1* instance of R	
Total Participation	Each instance of E participates in *at least 1* instance of R	
Key + Total Participation	Each instance of E participates in *exactly 1* instance of R	
Weak Entity + Identifying Relationship	E is a weak entity set with identifying owner E' and indentifying relationship set R	

Relational Mapping

Relational Mapping

Entity Sets

Basic Mapping

ER Diagram	Schema
Name of entity set	Name of table
Attribute of entity set	Attribute/column of table
Key attribute of entity set	Primary key of table
Derived attribute of entity set	*should not appear**

Data Type?

ER diagram does not capture the data type. We assume that the data type is as *logical* as possible *(e.g., name is not an integer)*.

Other Constraints?

- ER diagram has a limited expressibility
- Many constraints cannot be encoded *except* with complicated tricks
  - ***Example:*** Not-NULL *(using unary)*, Unique *(one-to-one)*, and data type
    - We assume that they ~~are~~ simply following requirements
    - For types, we assume that they are using the most logical type
- General constraints cannot be encoded

*They are *derived*, it means there is a computation that can be used to generate them. Typically, they frequently change *(e.g., age)*.

Relational Mapping

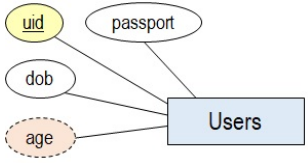
Entity
- Basic
- Key
- Composite
- Multivalued
Relationship
Guidelines

Entity Sets

Key Attributes

- ER diagram can only encode one set of key for each entity set
  - Exception can be made with tricks *(e.g., one-to-one relationship)*

ER Diagram



Schema

```
CREATE TABLE Users (
  uid       INT PRIMARY KEY,
  name      VARCHAR(100),
  dob       DATE,
  passport  VARCHAR(20)
);
```

Table

uid	name	dob	passport

*They are *derived*, it means there is a computation that can be used to generate them. Typically, they frequently change *(e.g., age)*.

Relational Mapping

Entity
- *Basic*
- *Key*
- ***Composite***
- *Multivalued*

Relationship Guidelines

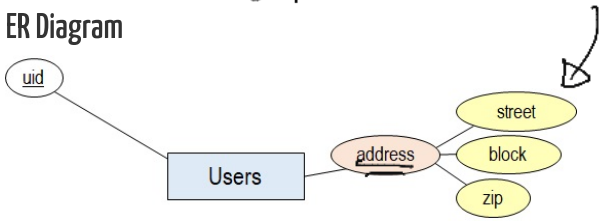
Entity Sets

Composite Attributes

- We assume that tables will only hold ***atomic*** values
- Convert composite attributes into a *set of single-valued* attributes

address
- street
- block
- zip

ER Diagram



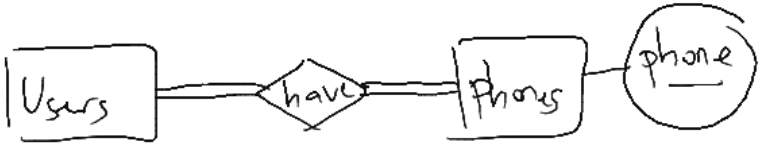
Schema

```
CREATE TABLE Users (
  uid       INT PRIMARY KEY,
    :
  street    VARCHAR(50),
  block     VARCHAR(6),
  zip       INT
);
```

Table

uid	...	street	block	zip

Relational Mapping



Entity
- *Basic*
- *Key*
- *Composite*
- ***Multivalued***

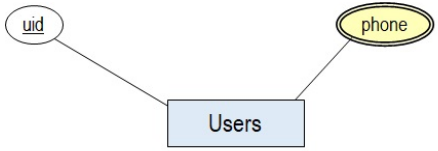
Relationship
Guidelines

Entity Sets

Multivalued Attributes #1

- We assume that tables will only hold ***atomic*** values
- Convert multivalued attributes into a *set of single-valued* attributes
  - ***Limitation:*** *Can only have a fixed number of attributes*

ER Diagram



Schema

```
CREATE TABLE Users (
  uid       INT PRIMARY KEY,
    :
  phone1    INT,
  phone2    INT,
  phone3    INT
);
```

Table

uid	...	phone1	phone2	phone3

Relational Mapping

Entity
- *Basic*
- *Key*
- *Composite*
- ***Multivalued***

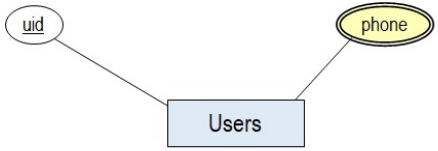
Relationship
Guidelines

Entity Sets

Multivalued Attributes #2

- We assume that tables will only hold ***atomic*** values
- Create *additional table* with *foreign key constraint*
  - ***Limitation:*** *Retrieval requires querying two tables*

ER Diagram



Schema

```
CREATE TABLE Users ( ... );
CREATE TABLE Phones (
  uid       INT,
  phone     INT,
  FOREIGN KEY (uid) REFERENCES Users(uid)
);
```

Table

uid	...

uid	phone

Relational Mapping

Entity
Relationship
- General
- Many-to-Many
- Many-to-One
- One-to-One
- Key + Total
- Weak Entity
Guidelines

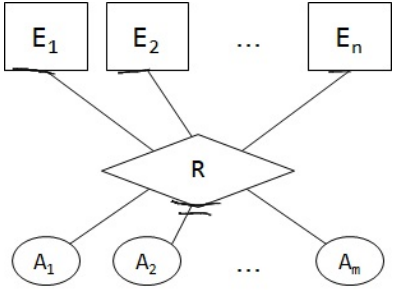
Relationship Sets

General *n*-ary Relationship Set R

- *n* entity sets involved: E_1, E_2, ..., E_n
- *m* attributes: A_1, A_2, ..., A_m

Key (E_1) ⇝ pk_E1
⇝ {pk_E1A, pk_E1B}

ER Diagram



General Schema

```
CREATE TABLE R (
  pk_E1   TYPE REFERENCES ...,
  pk_E2   TYPE REFERENCES ...,
    :       :
  pk_En   TYPE REFERENCES ...,
  A1      TYPE,
  A2      TYPE,
    :       :
  Am      TYPE,
  PRIMARY KEY (pkE1, ..., pk_En)
);
```

FOREIGN KEY (..) REF ...

[#]The general schema is for *unconstrained* relationships.

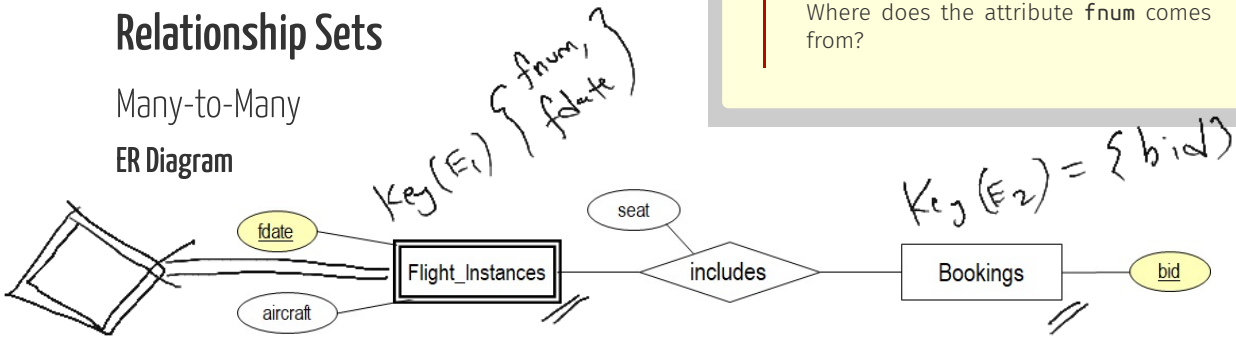
Relational Mapping

Entity
Relationship
- General
- Many-to-Many
- Many-to-One
- One-to-One
- Key + Total
- Weak Entity
Guidelines

Relationship Sets

Many-to-Many

ER Diagram



> **Quiz #1**
>
> Where does the attribute `fnum` comes from?

Schemas

```
CREATE TABLE FlightInstances (
  fnum      INT REFERENCES Flights,
  fdate     DATE,
  aircraft  VARCHAR(10),
  PRIMARY KEY (fnum, fdate)
);
CREATE TABLE Bookings (
  bid   INT PRIMARY KEY
);
```

```
CREATE TABLE Includes (
  fdate DATE,
  fnum  INT,
  bid   INT REFERENCES Bookings,
  seat  VARCHAR(10),
  FOREIGN KEY (fnum, fdate) REFERENCES FlightInstances,
  PRIMARY KEY (fnum, fdate, bid)
);
```

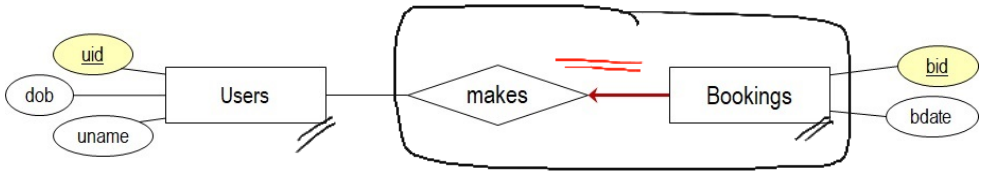
Relational Mapping

Entity
Relationship
- General
- Many-to-Many
- Many-to-One
- One-to-One
- Key + Total
- Weak Entity
Guidelines

Relationship Sets

Many-to-One

ER Diagram



Approach #1

```
CREATE TABLE Users (
  uid   INT PRIMARY KEY,
  uname VARCHAR(100),
  dob   DATE
);
CREATE TABLE Bookings (
  bid   INT PRIMARY KEY,
  bdate DATE
);
```

```
CREATE TABLE Makes (
  uid   INT,
  bid   INT,

  FOREIGN KEY (uid) REFERENCES Users (uid),
  FOREIGN KEY (bid) REFERENCES Bookings (bid),
  PRIMARY KEY (bid)
);
```

Quiz #2

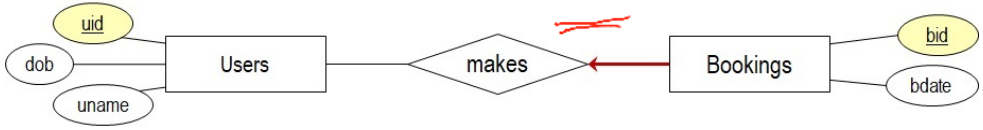
How do we find users who did not make any bookings?

Relational Mapping

Relationship Sets

Many-to-One

ER Diagram



> **Quiz #3**
>
> Which approach is more preferrable?

Approach #2

```
CREATE TABLE Users (
  uid   INT PRIMARY KEY,
  uname VARCHAR(100),
  dob   DATE
);
-- In this approach,
-- we remove "Bookings"
-- and combine with "makes"
-- into one table
```

```
CREATE TABLE MakesBookings (
  uid   INT,
  bid   INT,
  bdate DATE,
  FOREIGN KEY (uid) REFERENCES Users (uid),
  FOREIGN KEY (bid) REFERENCES Bookings (bid),
  PRIMARY KEY (bid)
);
```

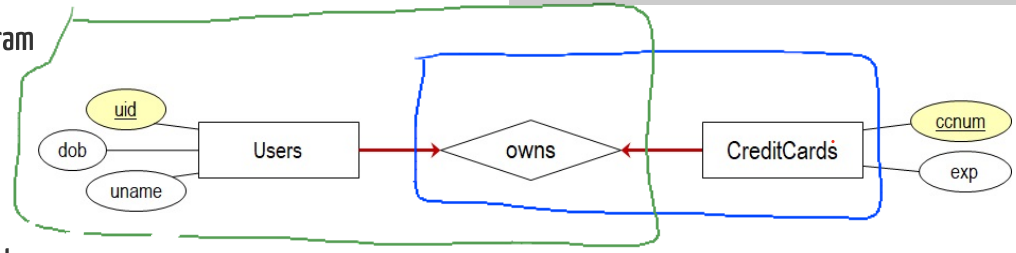
Relational Mapping

Entity
Relationship
- General
- Many-to-Many
- Many-to-One
- One-to-One
- Key + Total
- Weak Entity
Guidelines

Relationship Sets

One-to-One

ER Diagram



Approach #1

```
CREATE TABLE Users (
  uid   INT PRIMARY KEY,
  uname VARCHAR(100),
  dob   DATE,
  ccnum INT UNIQUE,
    REFERENCES CreditCards (ccnum)
);
-- Assume CreditCards is not combined
```

```
CREATE TABLE CreditCards (
  ccnum INT PRIMARY KEY,
  exp   DATE,

  uid   INT UNIQUE,
    REFERENCES Users (uid)
);
-- Assume Users is not combined
```

Note

With this approach, you can only choose one to be combined while the other not combined. Can you figure out the missing tables?

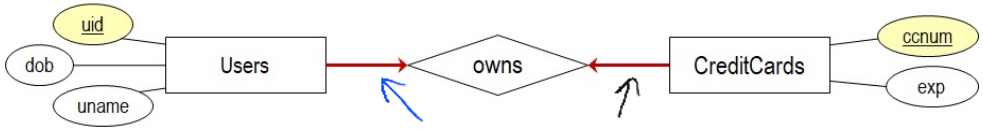
Relational Mapping

Entity
Relationship
- General
- Many-to-Many
- Many-to-One
- One-to-One
- Key + Total
- Weak Entity
Guidelines

Relationship Sets

One-to-One

ER Diagram



> **Quiz #4**
>
> Can we have users without credit cards or credit cards without user?

Partial approach

Approach #2

```
CREATE TABLE Users (
  uid   INT PRIMARY KEY,
  uname VARCHAR(100),
  dob   DATE,
  ccnum INT UNIQUE,
  exp   DATE
);
-- We choose uid to be PK
```

```
CREATE TABLE Users (
  uid   INT UNIQUE,
  uname VARCHAR(100),
  dob   DATE,
  ccnum INT PRIMARY KEY,
  exp   DATE
);
-- Alternatively, ccnum is the PK
```

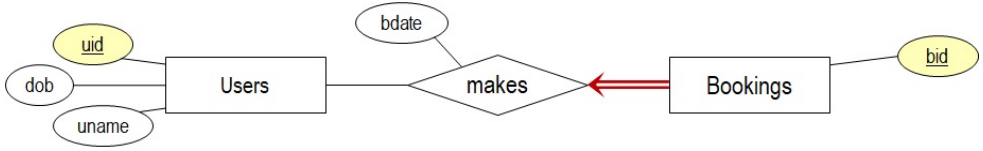
Relational Mapping

Entity
Relationship
- General
- Many-to-Many
- Many-to-One
- One-to-One
- Key + Total
- Weak Entity
Guidelines

Relationship Sets

Key + Total Participation Constraint

ER Diagram



Schema

```sql
CREATE TABLE Users (
  uid   INT PRIMARY KEY,
  uname VARCHAR(100),
  dob   DATE
);
```

```sql
CREATE TABLE MakesBookings (
  bid   INT PRIMARY KEY,
  bdate DATE,
  uid   INT NOT NULL,
  -- NOT NULL ensures that there must be a user!
  FOREIGN KEY (uid) REFERENCES Users (uid)
);
```

> **Quiz #5**
>
> Can we split **Makes** and **Bookings** into separate table?

Relational Mapping

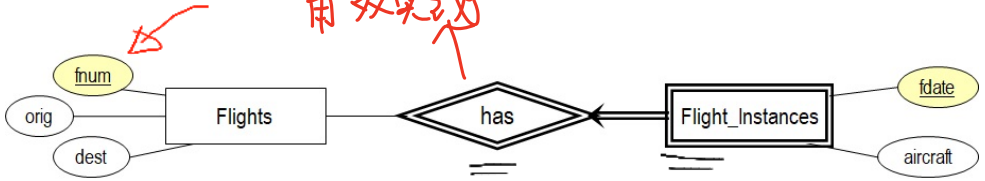
Entity
Relationship
- *General*
- *Many-to-Many*
- *Many-to-One*
- *One-to-One*
- *Key + Total*
- ***Weak Entity***

Guidelines

Relationship Sets

Weak Entity Sets

ER Diagram



Schema

```
CREATE TABLE Users (
  fnum  INT PRIMARY KEY,
  orig  VARCHAR(10),
  dest  VARCHAR(10)
);
```

```
CREATE TABLE FlightInstances (
  fnum     INT REFERENCES Flights (fnum)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  fdate    DATE,
  aircraft VARCHAR(10),
  PRIMARY KEY (fnum, fdate)
);
```

> **Note**
>
> `ON ... CASCADE` depends the on requirement. Do we want to be able to remove flights with flight instances?

Relational Mapping

Guidelines

General Guidelines

- Determine which are ***data*** and which are ***operations*** and captures only the data

Guidelines for ER Design

- An ER diagram should capture as many constraints as possible *(from the requirements)*
- An ER diagram must **NOT** impose any constraints that are not required *(by the requirements)*
- Attributes should have the same name *if and only if* they are *semantically equivalent*
  - Often called the **Universal Schema** assumption *(makes query easier)*

Guidelines for Relational Mapping

- The relational schema should capture as many constraints as possible *(from the requirements)*
  - Using column and/or table constraints *(e.g., `NOT NULL`, `UNIQUE`, `CHECK (...)`, etc)*
- The relational schema must **NOT** impose any constraints that are not required *(by the requirements)*
- The relational schema should use logical data types *(e.g., names should not be `INT`)*

Extended Notations

Extended Notations

Hierarchies
- ***Basic***
- *Interpretation*
- *Constraints*
- *Example*
- *Mapping*
- *Quiz*

Aggregation

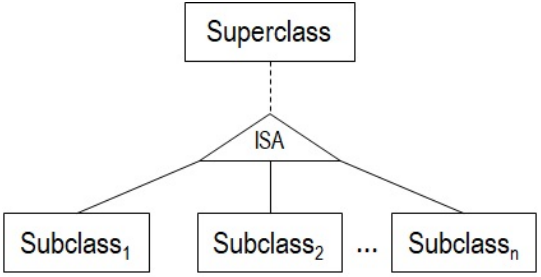
ISA Hierarchies

Basic

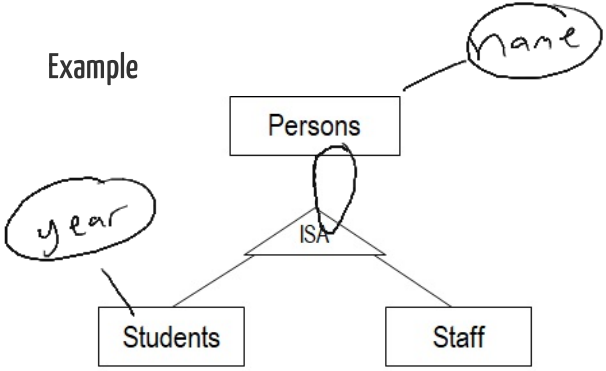
- Special type of relationship: ***"is a"***
- Model generalization/specialization of entity sets *(i.e., superclass/subclass)*

Representations

General Representations



Example



Extended Notations

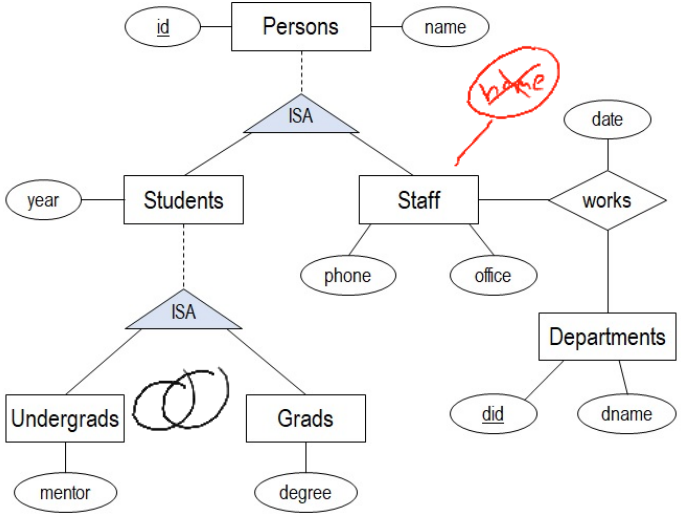
Hierarchies
- *Basic*
- ***Interpretation***
- *Constraints*
- *Example*
- *Mapping*
- *Quiz*

Aggregation

ISA Hierarchies

Interpretation

- Every entity in a subclass *is an* entity in its superclass
  - If the superclass and subclass must be uniquely identified by the same key
  - Subclass key should **NOT** be shown in ER diagram
- Each subclass may have additional attributes and/or relationships
  - While not required, that is the reason why we have ISA hierarchies *(i.e., restrict relationships, etc)*



Extended Notations

ISA Hierarchies

Constraints

Overlap Constraints

Can a superclass entity belong to *multiple* subclasses?

- `TRUE` : A superclass can belong to multiple subclasses
  (e.g., a person can be both student and staff)
- `FALSE`: A superclass cannot belong to multiple subclasses
  (e.g., a student is either graduate or undergraduate)

Covering Constraints

Must a superclass entity belong to *at least one* subclasses?

- `TRUE` : A superclass must belong to at least one subclasses
  (e.g., there is no student that is neither graduate nor undergraduate)
- `FALSE`: A superclass need not belong to any subclasses
  (e.g., not every person is a student or staff)

Extended Notations

ISA Hierarchies

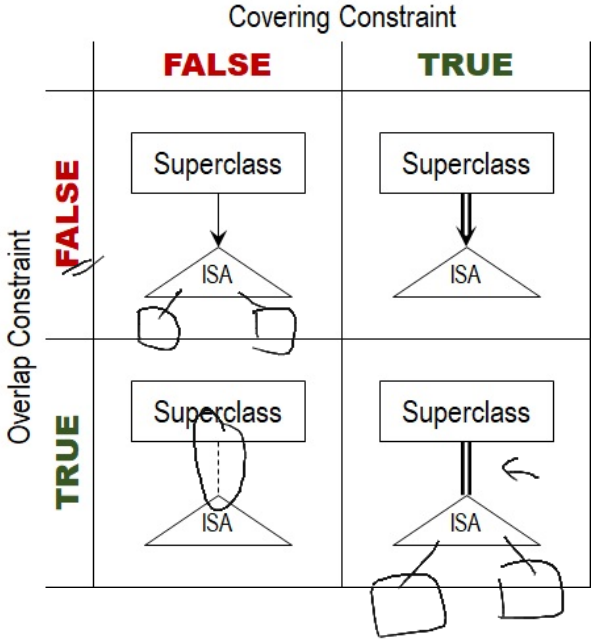
Constraints

Notation

- **TRUE** covering constraint is similar to *participation constraint*
- **FALSE** overlap constraint is similar to *key constraint*

Note

- Subclass is connected to ISA triangle using a solid line
- Conventionally, superclass is above ISA and subclass is below ISA
  - To avoid ambiguity, we use dashed line to connect to superclass when overlap = TRUE and covering = FALSE



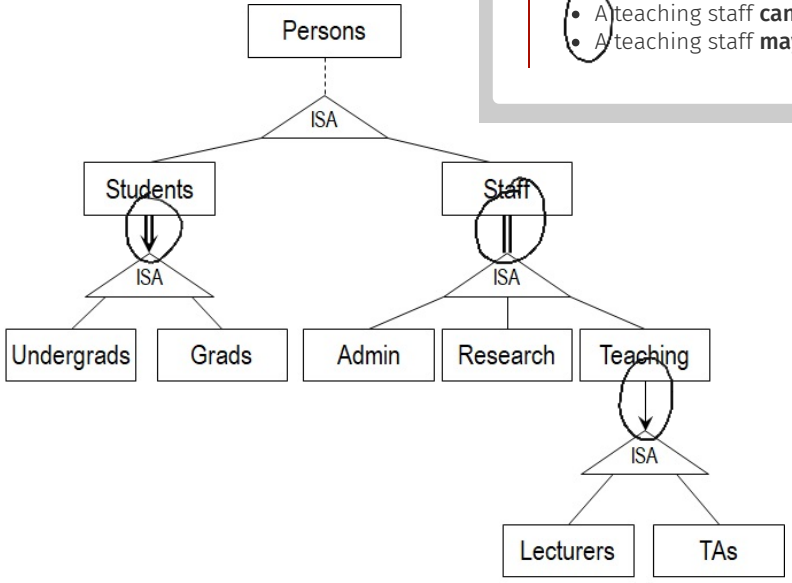
Extended Notations

Hierarchies
- *Basic*
- *Interpretation*
- *Constraints*
- ***Example***
- *Mapping*
- *Quiz*

Aggregation

ISA Hierarchies

ER Diagram



Example

- A person **may** be both student and staff
- A person **may** be neither a student nor staff
- Each student **is** either an undergraduate or a graduate **but not both**
- Each staff **is** either *admin*, *research*, *teaching* or **any combination** of these three roles
- A teaching staff **cannot** be a lecturer and TA
- A teaching staff **may** be neither lecturer or TA

Extended Notations

Hierarchies
- *Basic*
- *Interpretation*
- *Constraints*
- *Example*
- ***Mapping***
- *Quiz*

Aggregation

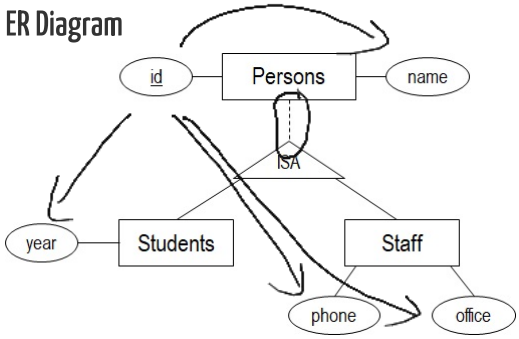
ISA Hierarchies

Relational Mapping

- ***Basic Strategy:*** One relation per superclass and subclass

```
CREATE TABLE Persons (
  id    CHAR(20) PRIMARY KEY,
  name  VARCHAR(50)
);
CREATE TABLE Students (
  id    CHAR(20) PRIMARY KEY,
    REFERENCES Persons (sid) ON DELETE CASCADE
  year  INT, -- or DATE?
);
CREATE TABLE Grads
  id      CHAR(20) PRIMARY KEY
    REFERENCES Persons (sid) ON DELETE CASCADE,
  phone   INT,
  office  VARCHAR(10)
);
```

ER Diagram



Extended Notations

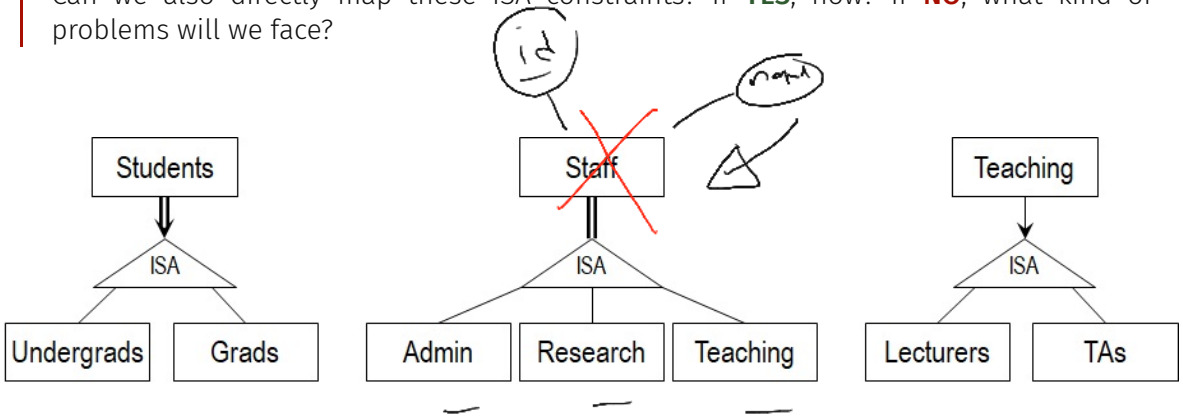
Hierarchies
- *Basic*
- *Interpretation*
- *Constraints*
- *Example*
- *Mapping*
- ***Quiz***

Aggregation

ISA Hierarchies

Quiz #6

Can we also directly map these ISA constraints? If **YES**, how? If **NO**, what kind of problems will we face?



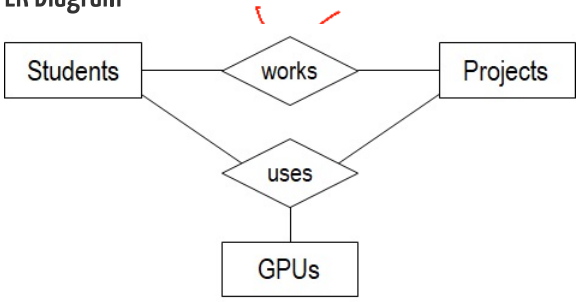
Extended Notations

Aggregation

Motivation

- Relationships only between entity sets
- No relationships between entity sets and relationship sets

ER Diagram



Example

[...] For ***some*** projects worked on by students, GPU may be used.

Limitations

- Relationship between "works" and "uses" is not captured
- Using both "works" and "uses" are somewhat redundant

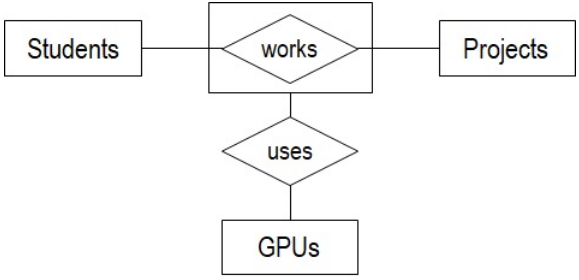
Extended Notations

Aggregation

Basic Idea

> Abstraction that treats relationships as *higher-level* entities
>
> - It is constructed as relationship *(with relationship set)*
> - As entities *(with entity sets)*, it can participate in relationships

ER Diagram



Note

- Aggregate is a **rectangle** *(i.e., entity sets)* around the **diamond** *(i.e., relationship sets)*
- The rectangle should **NOT** touch the diamond
- Entity sets forming the aggregate touches the diamond
- When using the aggregate as an entity set, the line touches the rectangle

Extended Notations

Aggregation

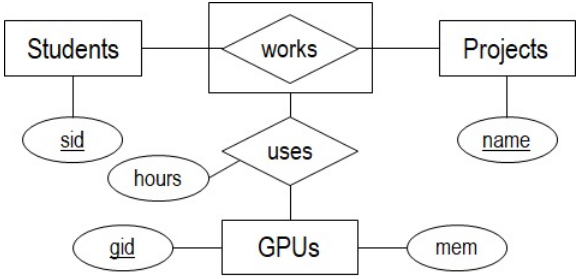
Relational Mapping

- *Students(sid: **INT**)*
- *Projects(name: **VARCHAR(50)**)*
- *GPUs(gid: **INT**, mem: INT)*
- *Works(sid: **INT**, name: **VARCHAR(50)**)*

Foreign Key Constraints

- (*Works.sid*) ⇝ (*Students.sid*)
- (*Works.name*) ⇝ (*Projects.name*)

ER Diagram



Schema

```
CREATE TABLE Uses (
  gid   INT REFERENCES GPUs,
  sid   INT,
  name  VARCHAR(50),
  hours NUMERIC,
  PRIMARY KEY (gid, sid, pname),
  FOREIGN KEY (sid, pname)
    REFERENCES Works (sid, pname)
);
```

Summary

Summary

Summary

Entity-Relationship Model

> Visualized using ER diagrams

- ***Basic Concepts:*** Entity Sets, Relationship Sets, Attributes
- ***Constraints:*** Cardinality Constraints, Participation Constraints
- ***Extended Concepts:*** ISA Hierarchies, Aggregation

Relational Mapping

> Mapping ER diagram to database schema

- Not all constraints of ER diagram can be captured *(capture as much as possible according to requirements)*

Summary

Requirements Analysis

Online Airline Reservation Systems

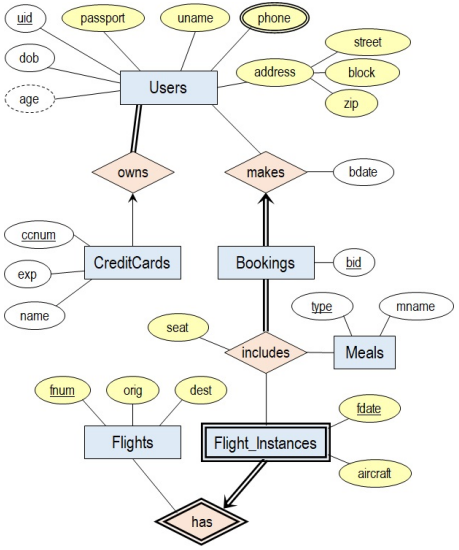
Users need to be able to **make** **bookings** from an origin to a destination airport which may **comprise** multiple connecting **flights**. We record the booking date. Each flight has a **flight number**, the **origin** and **destination** ~~airport~~, the ~~distance~~ in kilometers, the ~~departure~~ and ~~arrival time~~, and the ~~days of the week~~ the flight is in operation.

A **flight instance** is the actual scheduled flight on a **given day** together with the assigned **aircraft type**. For example, flight SQ231 flies daily from Singapore to Sydney, typically with a Boeing 777-300ER *(code: B77W)*.

For a valid booking, we need the user's **name**, ~~sex~~, **address**, **phone number(s)**, and the **passport number**. Users are only able to pay via **credit card**. When making a booking, the user can select the class, the **seat** number, as well as **meal** preferences *(if available)*.

Additional Exercise

Can you complete the ER diagram?



```
postgres=# exit
Press any key to continue . . .
```

Solutions

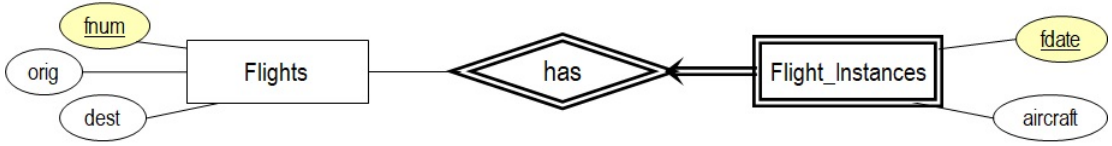
Quiz #1

Quiz #1

Where does the attribute `fnum` comes from?

Solution

The attribute comes from `Flights` because `Flight_Instances` is a weak entity set with `Flights` being the owning entity set.



Solutions

Quiz #2

Quiz #2

How do we find users who did not make any bookings?

Solution

- The table `Users` stores all user information *(whether they have made bookings or not)*
- The table `Bookings` stores all booking information
- The table `Makes` stores only bookings that have been booked by a user
  - In other words, it also stores users who have made bookings
  - So we perform $\pi_{uid}(Users) - \pi_{uid}(Makes)$ *(modify this query to suit the need of the problem)*

Solutions

Quiz #3

Quiz #3

Which approach is more preferrable?

Solution

This is more of a discussion but note that using Approach #2, we do not have to perform join operation between `Makes` and `Bookings`.

Solutions

Quiz #4

Quiz #4

Can we have users without credit cards or credit cards without user?

Solution

If `uid` is the primary key, then we cannot have *credit cards without user* *(since `uid` cannot be `NULL` while `ccnum` can be `NULL`)*. On the other hand, if `ccnum` is the primary key, then we cannot have *users without credit cards* *(since `ccnum` cannot be `NULL` while `uid` can be `NULL`)*.

To prevent both, we add `NOT NULL` constraint on top of `UNIQUE` constraint on the candidate key not selected as primary key.

Solutions

Quiz #5

Quiz #5

Can we split `Makes` and `Bookings` into separate table?

Solution

In general, **NO**. If we split the table, then we cannot enforce *total participation constraint* on `Bookings` with respect to `Makes`.

Why? Simply because we can insert into `Bookings` without making any insertion to `Makes`.

Solutions

Quiz #6

Quiz #6

Can we also directly map these ISA constraints?

Solution

There are "tricks" that we can use to enforce these but it involves adding redundant columns. We will not go into these in details.

Without these tricks, the only other strategy we have is to not create the superclass table. Instead, we can make them into `VIEW` (i.e., a computed table which is computed each time we want to query and adds no additional storage).

The problem with this approach is that the attributes of the superclass have to be stored in the subclass. Firstly, this is redundant storage. Secondly, we no longer can uniquely identify the attributes since they can be different in the subclasses.

The DBMS solution to this is to use triggers, which you will learn in the future.

Solutions

Additional Exercise

> **Additional Exercise**
>
> Can you complete the ER diagram?

Discussion

> This is intended for discussions and the list below is intended only as a rough guide. There may be other similarly correct solutions.
>
> - Is airport an entity set?
> - Where should distance, departure time, arrival time, and days of the week be recorded?
> - Where should sex be recorded?
> - How should the airport be related to what is already shown in the ER diagram?

Final Note

- Modeling using ER diagram is an ***iterative*** process, you may need to revisit your model and revise *(add, remove, or edit elements)*