Computing

CS2102: Database Systems

Lecture 3 — Entity Relationship Model (ER Model)

Quick Recap: SQL for Creating Databases

- Data Definition Language (DDL)
  - Create, modify and drop tables to implement a given DB schema
  - Specify integrity constraints (e.g., **NOT NULL**, **PRIMARY KEY**, **FOREIGN KEY**, **CHECK**)

- Data Manipulation Language (DML)
  - Insert, update and delete data from tables

Employees (id: **integer**, name: **text**, age: **integer,** role**: text**)

```
CREATE TABLE Employees (
        id      INTEGER PRIMARY KEY,
        name    VARCHAR(50) NOT NULL,
        age     INTEGER,
        role    VARCHAR(50)
);
```

Employees

id	name	age	role

```
INSERT INTO Employees VALUES
        (101, 'Sarah', 25, 'dev')
        (102, 'Judy', 35, 'sales');
```

Employees

id	name	age	role
101	Sarah	25	dev
102	Judy	35	sales

We Sneakily Skipped a Step

- Open questions:
  - Where does the database schema come from?
  - What tables with which attributes do we need?
  - What data integrity constraints are required?
  - Table names, attribute names, data types, …?

➜ **Database Design Process**

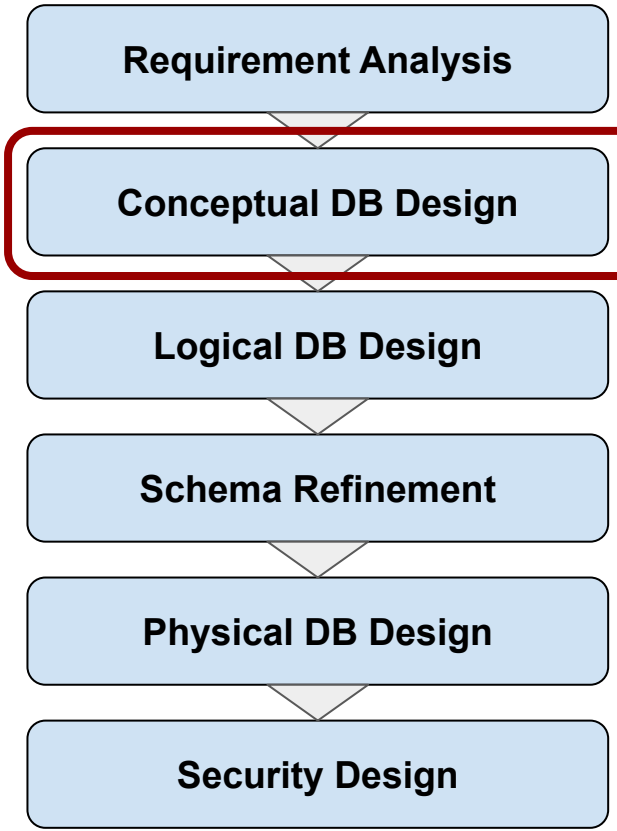
Quick Quiz: Which table is "better"?

```
CREATE TABLE Employees (
        id      INTEGER PRIMARY KEY,
        name    VARCHAR(50) NOT NULL,
        age     INTEGER,
        role    VARCHAR(50)
);
```

or

```
CREATE TABLE Employees (
        id      INTEGER PRIMARY KEY,
        name    VARCHAR(50) NOT NULL,
        dob     DATE,
        role    VARCHAR(100)
);
```

Database Design Process — 6 Common Steps



- Identification and collection of user needs
- e.g., data / application / performance requirements

- Capturing requirements using a conceptual model
- RDBMS: **Entity Relationship Model (ER Model)**

- Mapping conceptual model to logical schema of DBMS
- RDBMS: Entity Relationship Model → Relational Schema

- Checking schema / tables for redundancies and anomalies

- Implementing database based on final data schema
- Consideration of performance requirements

- Identification of users and user groups and their permissions to access which parts of the data

Overview

- **Entity Relationship Model**
  - **Overview + ER diagrams**
  - Entity sets and attributes
  - Relationship sets
  - Cardinality & participation constraints
  - Dependency constraints: weak entity sets
  - Aggregation
- Relational Mapping
  - From ER diagram to database tables
- Summary

Requirement Analysis: Online Airline Reservation System (OARS)

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, 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.

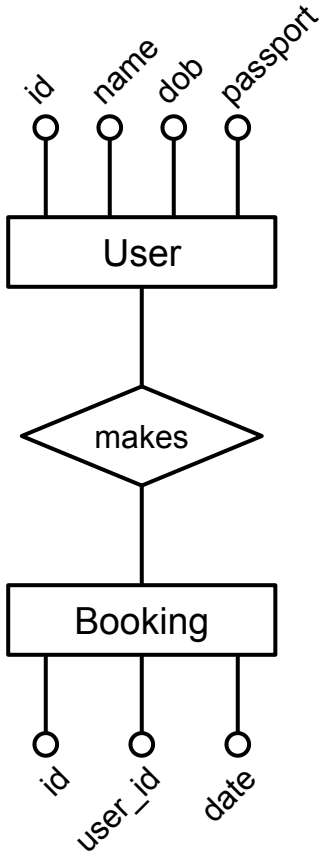
A flight instance is the actual scheduled flight on a given day together with the assigned aircraft type. For example, flight SQ231 flies daily from Singapore to Sydney, typically with a Boeing 777-300ER (code: B77W).

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 Model

- ER Model
  - Most common model for conceptual database design
  - Developed by Peter Chen (1976)
  - Visualized using **ER diagrams**
    (Important: many revised version – no one single set of notations!)

- Core concepts
  - All data is described in terms of **entities** and their **relationships**
  - Information about entities & relationships are described using **attributes**
  - Certain data constraints can be described using additional annotations



Overview

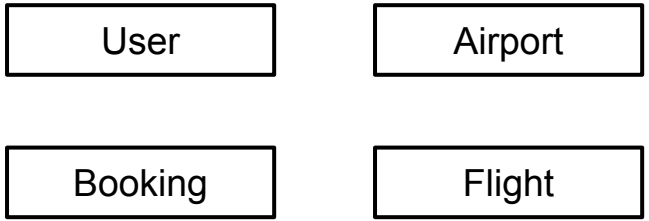
- **Entity Relationship Model**
  - Overview + ER diagrams
  - **Entity sets and attributes**
  - Relationship sets
  - Cardinality & participation constraints
  - Dependency constraints: weak entity sets
  - Aggregation
- Relational Mapping
  - From ER diagram to database tables
- Summary

Entities and Entity Sets

- **Entity**
  - Real-world things or objects that are distinguishable from other objects
    (e.g., an individual user, airport, flight, or booking)

- **Entity Set**
  - Collection of entities of the same type
  - Represented by rectangles in ER diagrams
  - Names are typically nouns

__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, [...]

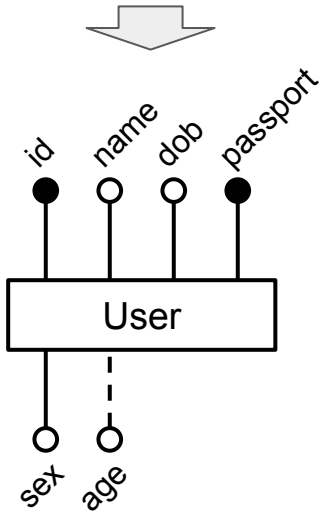


Attributes

- **Attribute**:
  - specific information describing an entity
  - represented by a small circle in ER diagrams

- 2 main subtypes of attributes
  - **Key attribute(s)**: uniquely identifies each entity
    - Indicated by a filled circle in ER diagram
    - Different attributes may uniquely identify an entity
    - Multiple attributes may form a composite key
  - **Derived attribute**: derived from other attributes
    - Indicated by a dashed line in ER diagram
    - Example: derive "age" from "dob"

*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. *[...]*

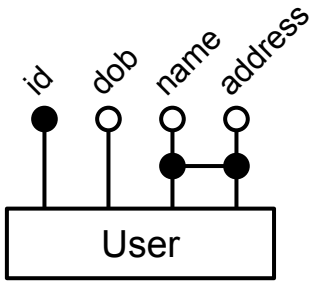


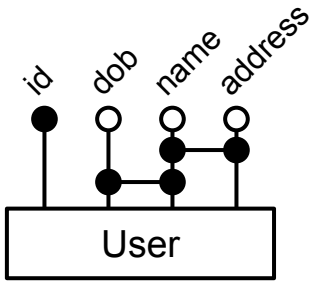
What about address and phone numbers?

Key Attributes

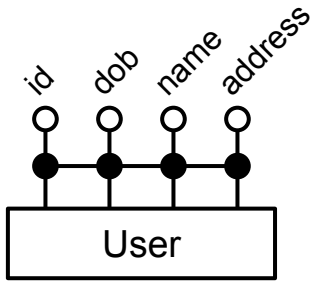
- **Composite** key attributes:
  - 2 or more attributes together uniquely identify each entity
  - An entity may have multiple composite key attributes
  - Representation in ER diagram: additional connecting line

- Examples (for illustration purposes; not necessarily realistic!)



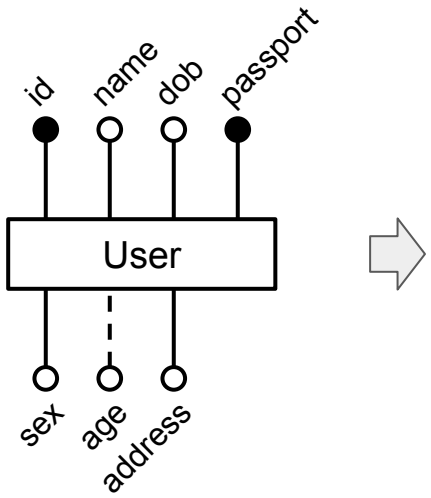


- At least all attributes uniquely identify an entity
- We typical prefer a minimum set of attributes



"Composite" Attributes

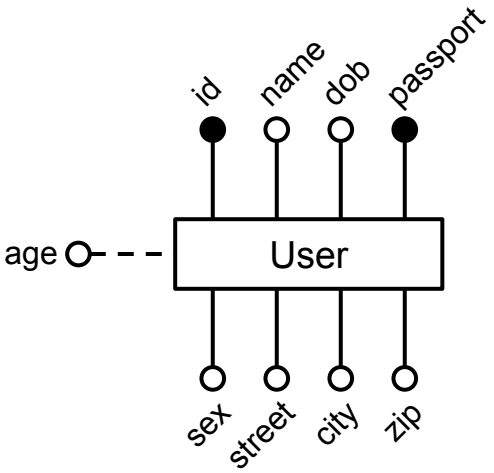
- Common: requirement analysis often vague / ambiguous / unclear
  - Not always obvious how certain attributes should be modeled
  - Example "address": single string attribute vs. multiple attributes



id	name	dob	age	sex	passport	address
101	Alice	15-02-2000	26	f	KEJR4A90	15 Computing Drive, Singapore 117418

"Composite" Attributes

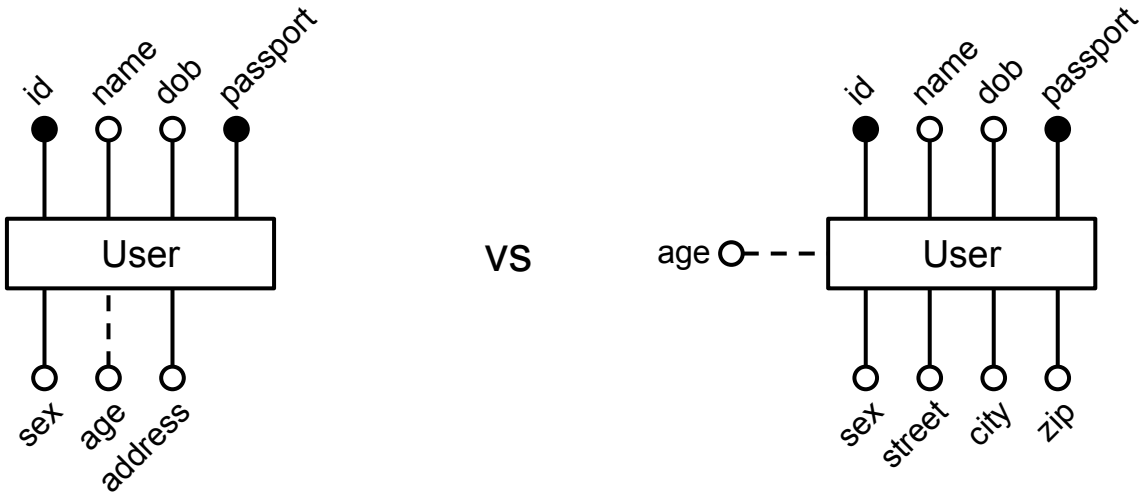
- Common: requirement analysis often vague / ambiguous / unclear
  - Not always obvious how certain attributes should be modeled
  - Example "address": single string attribute vs. multiple attributes



id	name	dob	age	sex	passport	street	city	zip
101	Alice	15-02-2000	26	f	KEJR4A90	15 Computing Drive	Singapore	117418

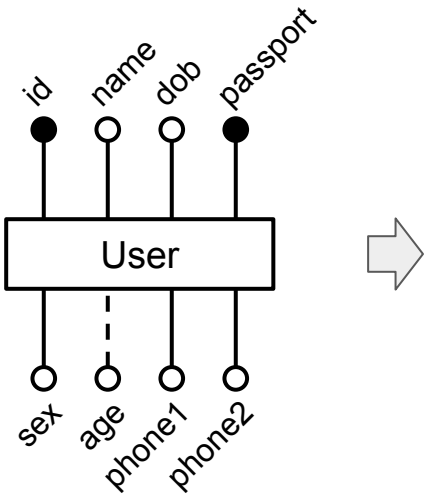
Quick Quiz

Which solution is typically the **preferred** one?
But always? And why?



Multivalued Attributes

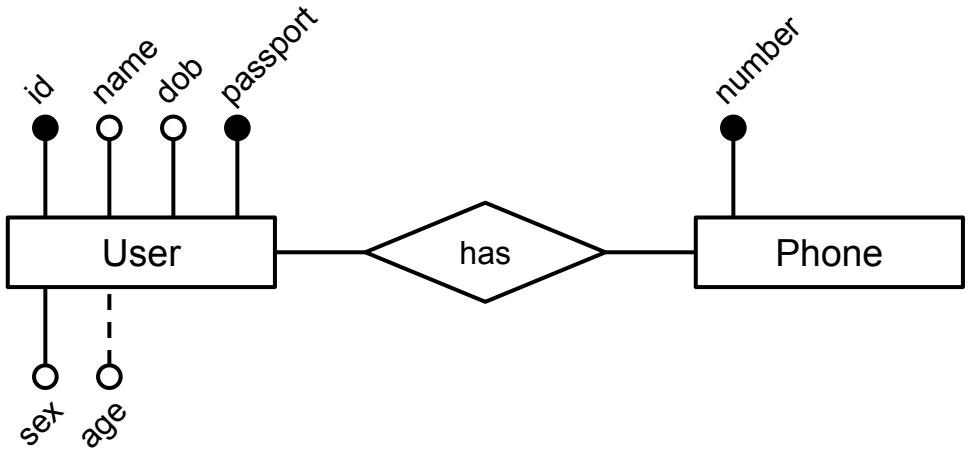
- Common: an attribute may refer to a set/list of values
  - Examples: phone numbers, hobbies, tags/keywords
  - However: all attributes must be single-valued
  - Example "phone numbers": fixed number of single-valued attributes vs. dedicated entity set



id	name	dob	age	sex	passport	phone1	phone2
101	Alice	15-02-2000	26	f	KEJR4A90	+65-1234-5678	+65-8765-4321

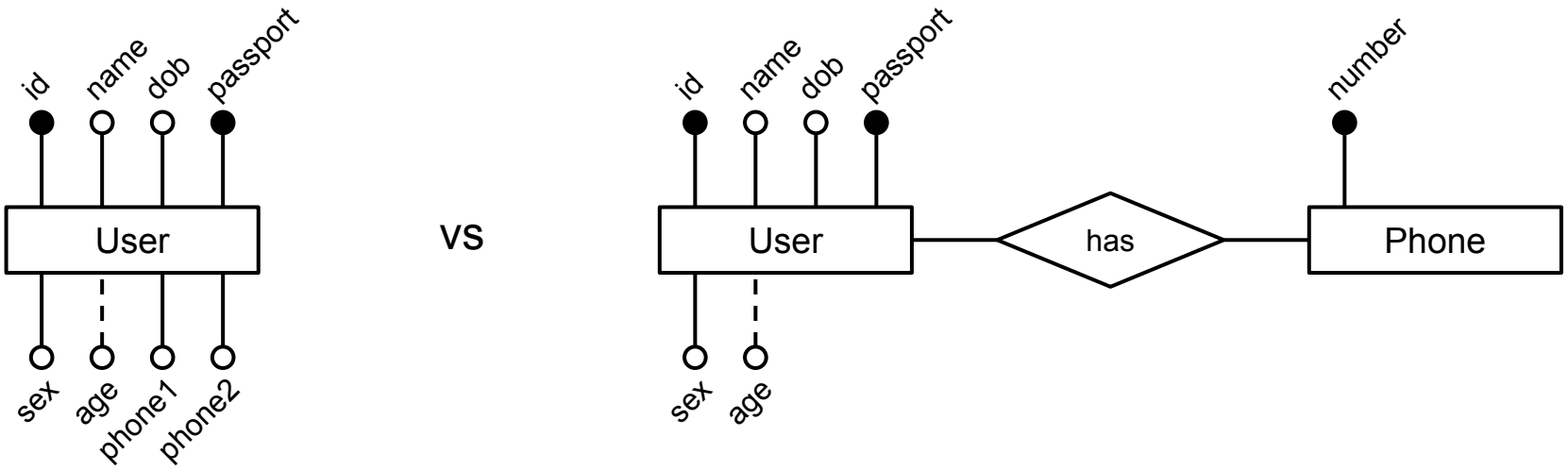
Multivalued Attributes

- Common: an attribute may refer to a set/list of values
  - Examples: phone numbers, hobbies, tags/keywords
  - However: all attributes must be single-valued
  - Example "phone numbers": fixed number of single-valued attributes vs. dedicated entity set



Quick Quiz

Which solution is typically the **preferred** one?
But always? And why?



Side Note

- PostgreSQL (and most modern RDBMS)
  - Not limited to basic single-valued data types
  - Support for complex / composite data types
  - Support for user-defined composite types

Quick Quiz: What are potential downsides of this more complex data types?

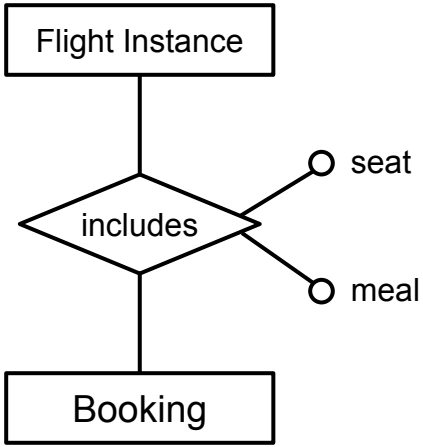
Source: PostgreSQL docs

Overview

- **Entity Relationship Model**
  - Overview + ER diagrams
  - Entity sets and attributes
  - **Relationship sets**
  - Cardinality & participation constraints
  - Dependency constraints: weak entity sets
  - Aggregation
- Relational Mapping
  - From ER diagram to database tables
- Summary

Relationships and Relationship Sets

- **Relationship**
  - Association among two or more entities
- **Relationship Set**
  - Collection of relationships of the same type
  - Represented by diamonds in ER diagrams
  - Can have their own attributes that further describe the relationship
  - Names are typically verbs

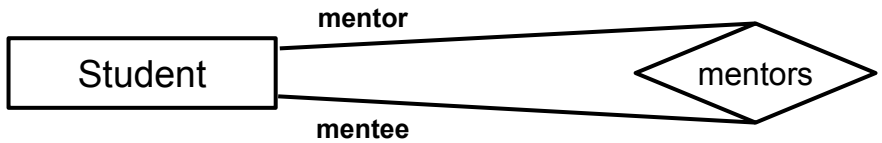


- Additional annotations to further specify relationships
  - Roles, degree, cardinalities, participation, dependencies

Relationship Roles

- **Role**
  - 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 labels only common in case of ambiguities
    (typically in case the same entity sets participate in the same relationship more than once)

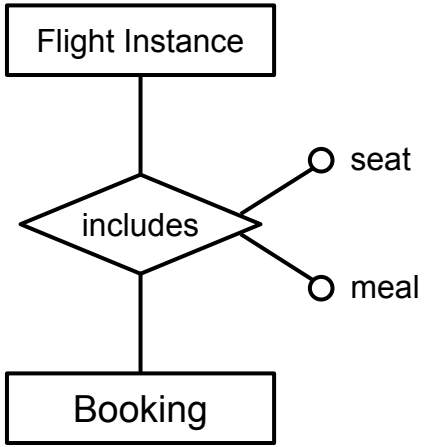
- Example: Students can mentor other students



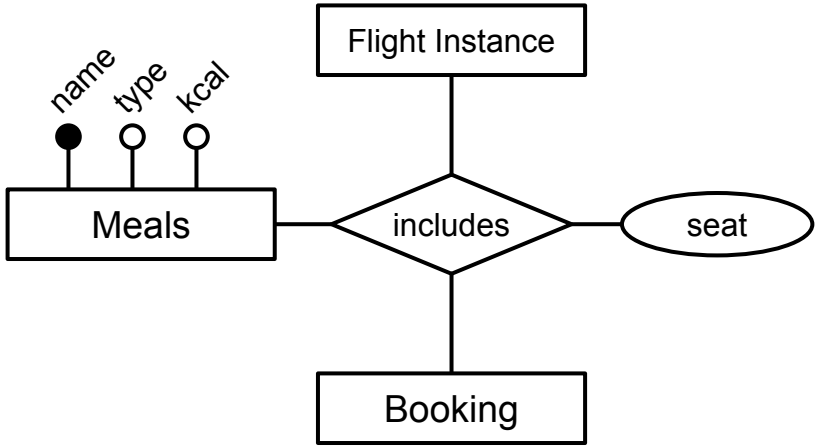
Degree of Relationship Sets

- **Degree**
  - In principle, no limitation on how many entity roles participate in a relationship
  - An *n*-ary relationship set involves *n* entity roles ➜ *n* = degree of relationship set

n = 2 ➜ binary relationship set

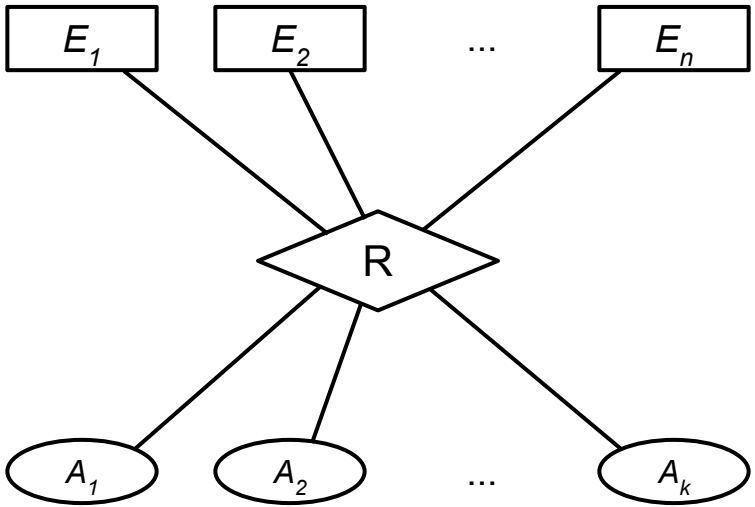


n = 3 ➜ ternary relationship set



Degree of Relationship Sets

- General n-ary relationship set R
  - *n* participating entity sets E_1, E_2, …, E_n
  - *k* relationship attributes A_1, A_2, …, A_k



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

Overview

- **Entity Relationship Model**
  - Overview + ER diagrams
  - Entity sets and attributes
  - Relationship sets
  - **Cardinality & participation constraints**
  - Dependency constraints: weak entity sets
  - Aggregation
- Relational Mapping
  - From ER diagram to database tables
- Summary

Cardinality & Participation Constraints

- Cardinalities of Relationship Sets
  - Describe how often an entity can participate in a relationship at most

upper bound

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

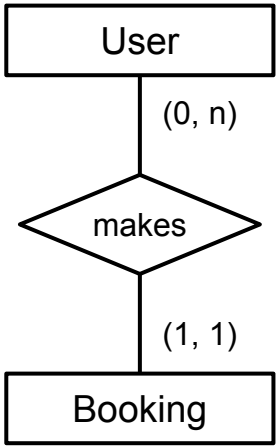
- Participation constraints
  - Describe how often an entity has to participate in a relationship at least

lower bound

  - Is the participation of an entity in a relationship even mandatory?

Cardinality & Participation Constraints

- Representation in ER diagram
  - (min,max) label at connections between entity and relationship sets

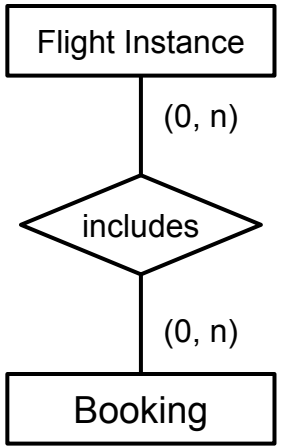


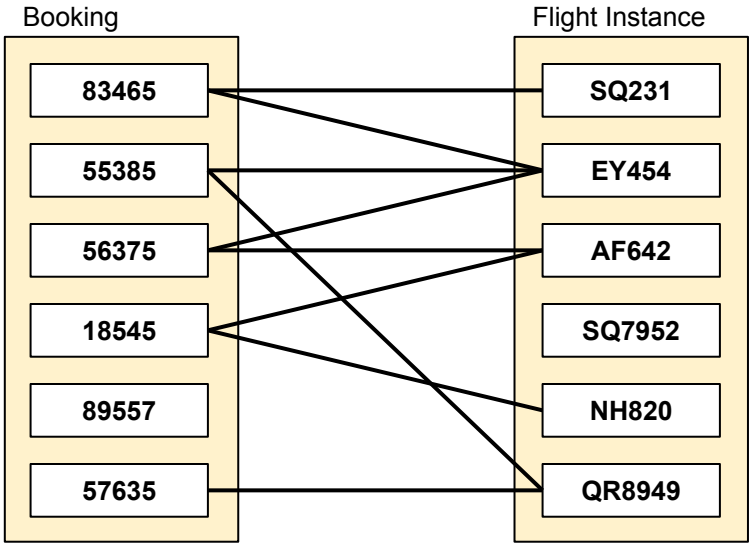
Interpretation

- Each user can make multiple bookings
  (but not every user must have made a booking)
- Each booking was done by exactly one user
  (implies that each booking is associated with a user)

Cardinality: Many-to-Many (no mandatory participation)

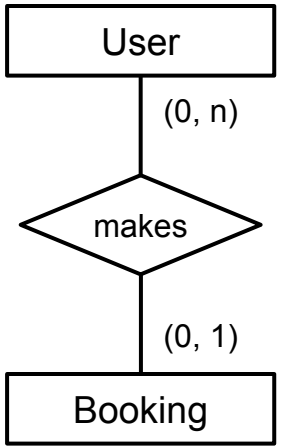
- Many-to-many relationship between bookings and flight instances
  - Each booking can include 0 or more flight instances
    (note that a booking with 0 flights might not meaningful; we will improve on that)
  - Each flight instance can be part of 0 or more bookings

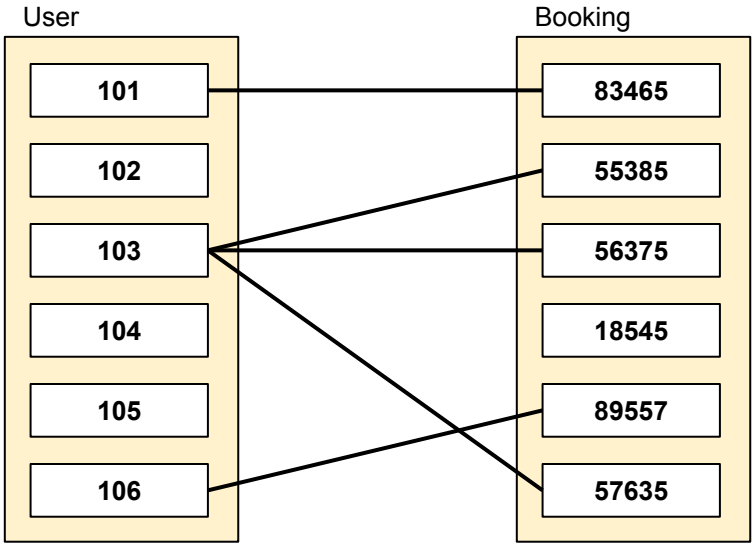




Cardinality: Many-to-One (no mandatory participation)

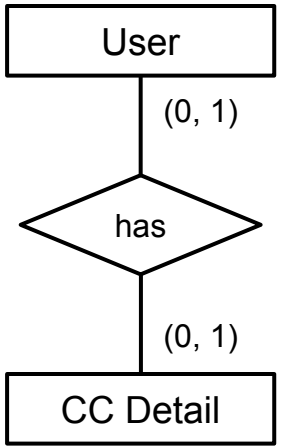
- Many-to-one relationship between users and bookings
  - Each user can make 0 or more bookings
  - Each booking is done by one 1 user <u>at most</u>
    (again, not perfect yet, and we will improve on that)

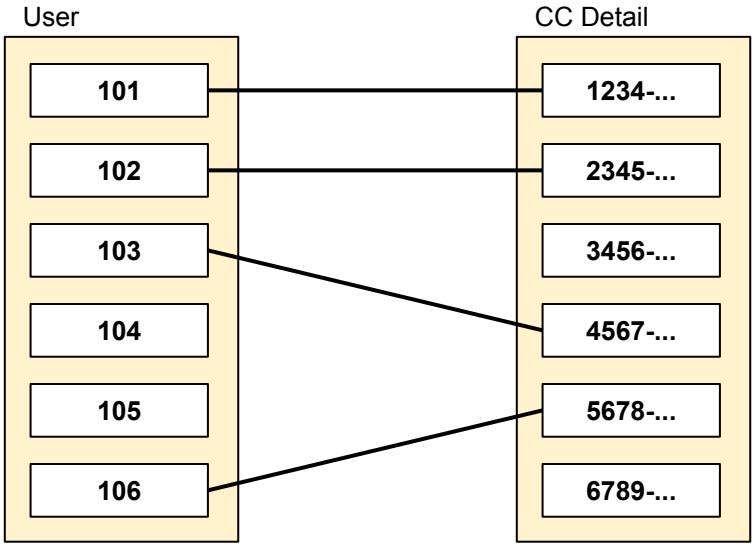




Cardinality: One-to-One (no mandatory participation)

- One-to-one relationship between users and credit card details
  - Each user can provide only 1 set of credit card details at most
  - Each set of credit card details is associated with 1 user at most





Participation Constraints

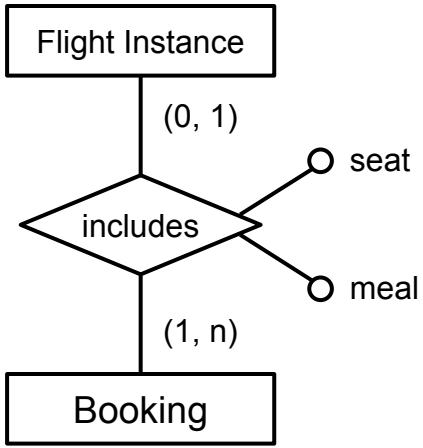
- Limitation of (basic) cardinality constraints from previous examples
  - A booking can include 0 flights
  - A booking can be done by 0 users
  - A set of credit card details does not need to be associated with a user

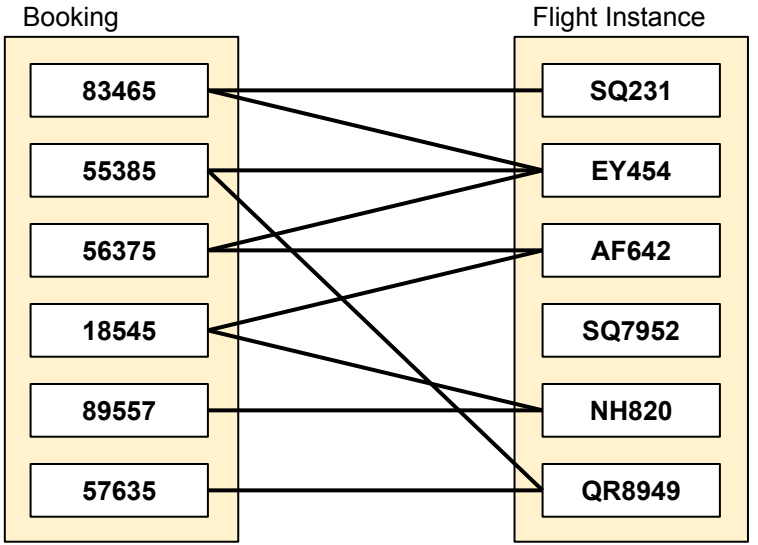
} an entity does not have to participate in a relation

➜ Let's include **participation constraints**

Cardinality & Participation Constraints

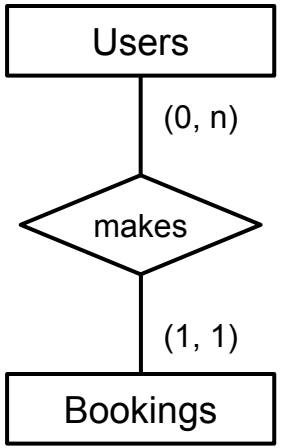
- Many-to-many relationship between bookings and flight instances
  - Each booking includes 1 or more flight instances
  - Each flight instance can be part of 0 or more bookings

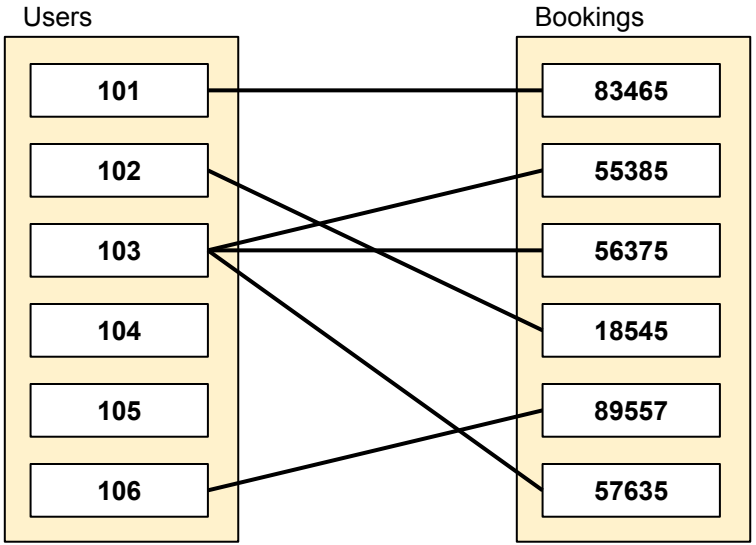




Cardinality & Participation Constraints

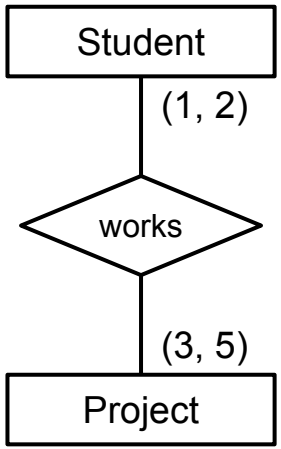
- Many-to-one relationship between users and bookings
  - Each user can make 0 or more bookings
  - Each booking is done by <u>exactly 1</u> user





Cardinality & Participation Constraints

- Flexibility of (min,max) notation
  - Minimum not limited to 0 or 1; maximum no limited to n
  - Arbitrary specific values to capture real-world constraint



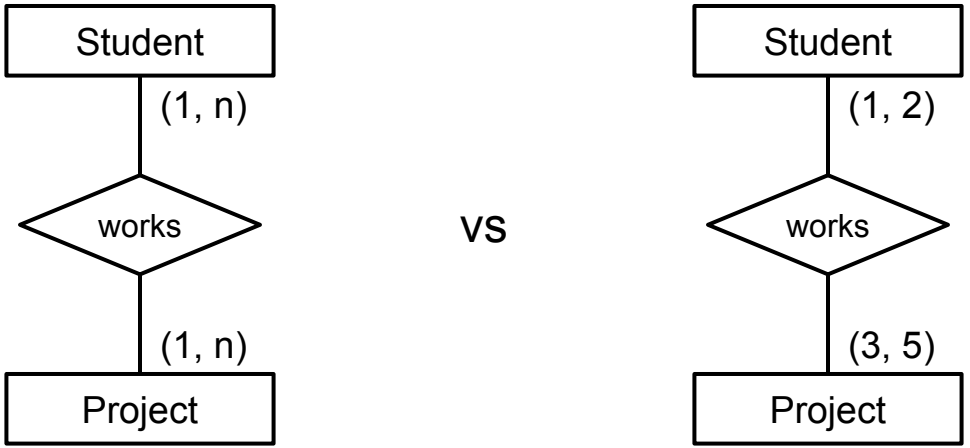
Interpretation

- Each student must work on at least 1 project
- Each student may not work on more than 2 projects
- Each project consists of at least 3 students
- Each project may not consist of more than 5 students

Quick Quiz

Why do values other than 0/1/n add significant complexity?

(just think about it for a minute here; we will cover it later)



Overview

- **Entity Relationship Model**
  - Overview + ER diagrams
  - Entity sets and attributes
  - Relationship sets
  - Cardinality & participation constraints
  - **Dependency constraints: weak entity sets**
  - Aggregation
- Relational Mapping
  - From ER diagram to database tables
- Summary

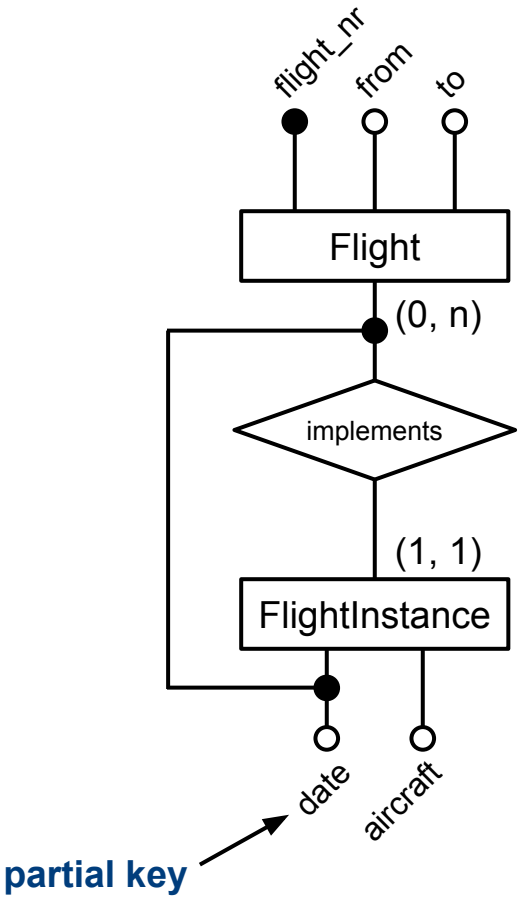
Dependency Constraints

- **Weak entity sets**
  - Entity set that does not have its own key
  - A weak entity can only be uniquely identified by considering the primary key of the **owner entity**
  - A weak entity's existence depends on the existence of its owner entity
  - Weak entity set and identifying relation set are represented via double-lined rectangles / diamonds

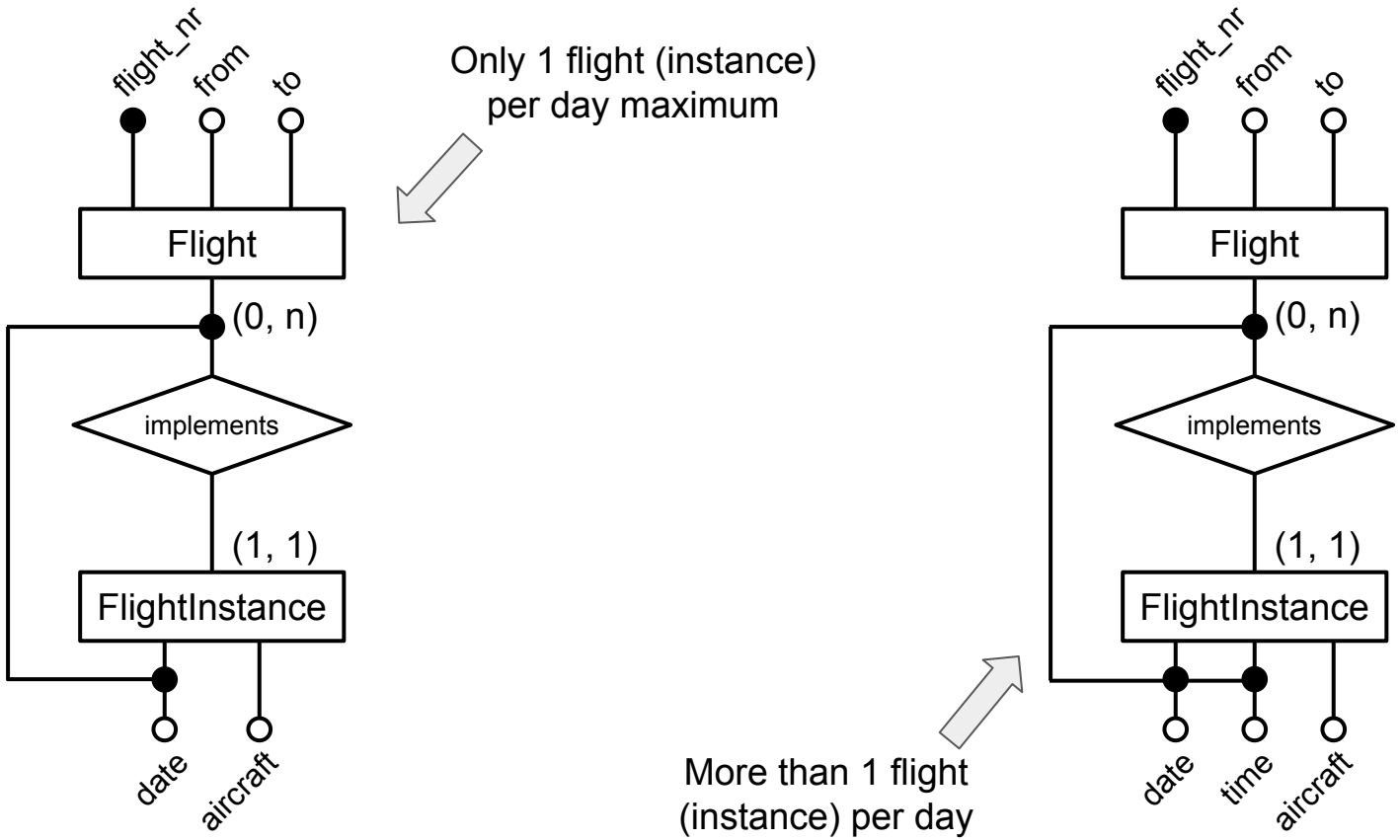
- Requirements
  - Many-to-one relationship (identifying relationship) from weak entity set to owner entity set
    (one-to-one possible but less common)
  - Weak entity set must have (1, 1) attached to identifying relationship

Dependency Constraints

- Example
  - A flight instance is the actual scheduled flight (with a unique flight number) on a given day
    - Each flights instance is identified by the "flight_nr and the "date"
    - "date" is a **partial key**
  - A flight instance cannot "exist" without the flight



Dependency Constraints



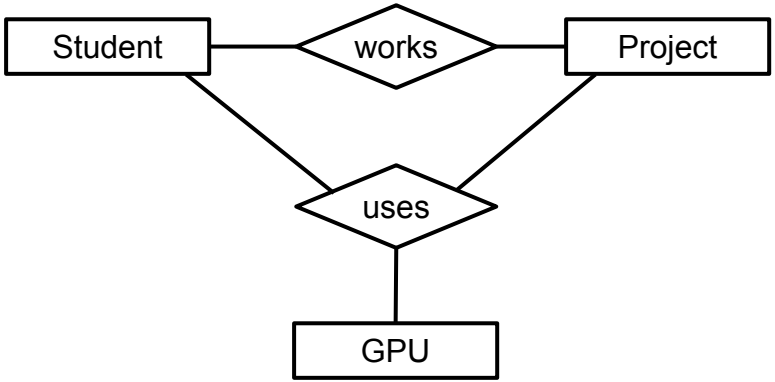
Overview

- **Entity Relationship Model**
  - Overview + ER diagrams
  - Entity sets and attributes
  - Relationship sets
  - Cardinality & participation constraints
  - Dependency constraints: weak entity sets
  - **Aggregation**
- Relational Mapping
  - From ER diagram to database tables
- Summary

Extended Concepts — Aggregation

- Concepts of ER diagrams so far
  - Only relationships between entity sets
  - No relationships between entity sets and relationship sets

- Motivating example



Limitations:

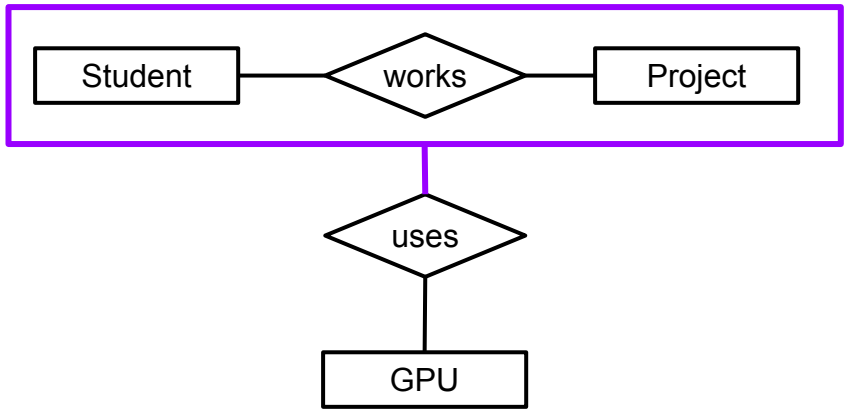
- Relationship between "works" and "uses" not explicitly captured
- "works" and "uses" are kind of redundant relationships

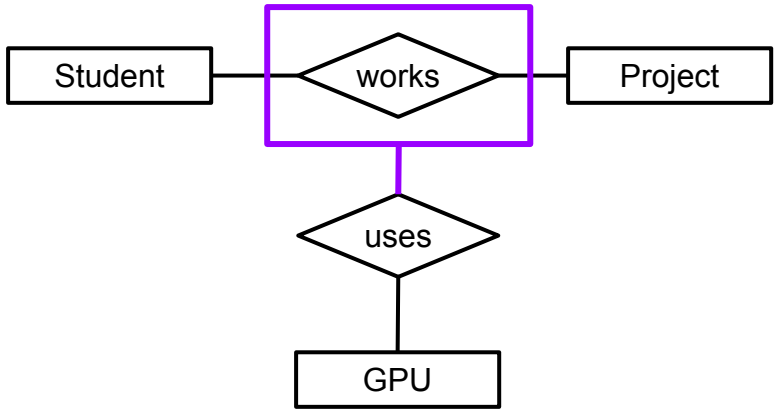
➜ **Aggregation**

Extended Concepts — Aggregation

- Aggregation — basic idea
  - Abstraction that treats relationships as higher-level entities
  - Example: treat Students-works-Projects as an entity set

- Notation in ER diagram (2 equivalent alternatives)

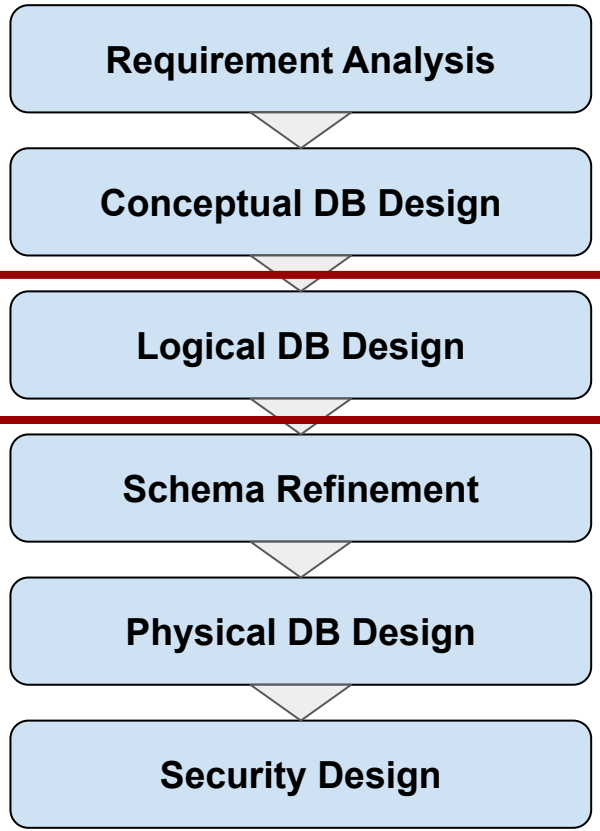




Overview

- Entity Relationship Model
  - Overview + ER diagrams
  - Entity sets and attributes
  - Relationship sets
  - Cardinality & participation constraints
  - Dependency constraints: weak entity sets
  - Aggregation
- **Relational Mapping**
  - **From ER diagram to database tables**
- Summary

Database Design Process — 6 Common Steps



- Identification and collection of user needs
- e.g., data /application / performance requirements

- Capturing requirements using a conceptual model
- RDBMS: **Entity Relationship Model (ER Model)**

- Mapping conceptual model to logical schema of DBMS
- RDBMS: Entity Relationship Model → Relational Schema

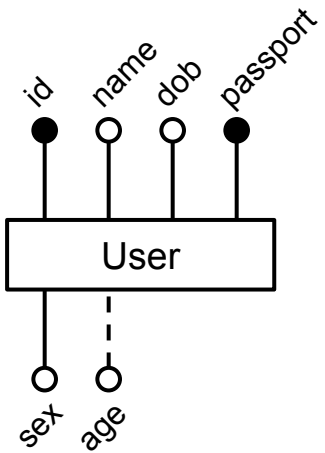
- Checking schema / tables for redundancies and anomalies

- Implementing database based on final data schema
- Consideration of performance requirements

- Identification users and user groups and their permissions to access which parts of the data

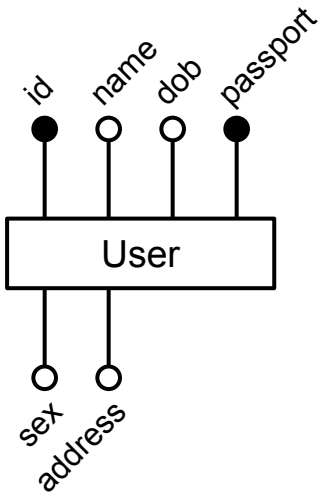
Entity Sets

- Straightforward mapping from entity sets to tables (except for composite & multivalued attributes)
  - Name of entity set ➜ name of table
  - Attributes of entity set ➜ attributes of table
  - Key attributes of entity set ➜ primary key of table

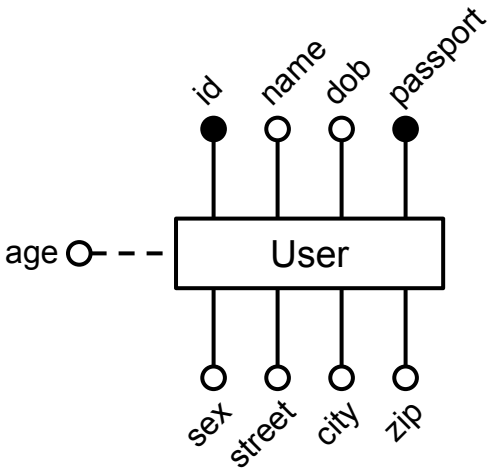



```
CREATE TABLE Users (
        id              INTEGER,
        name            VARCHAR(100),
        dob             DATE,
        sex             CHAR(1)
        age             INTEGER,
        passport        VARCHAR(20),
        PRIMARY KEY (id),
        UNIQUE (passport)
);
```

Note: PostgreSQL supports Generated Column but there are come caveats when used in practice that are beyond our scope.



VS

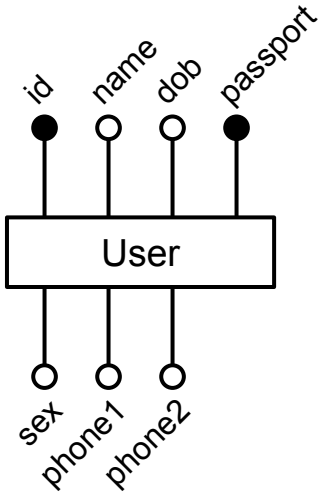



```
CREATE TABLE Users (
        id              INTEGER,
        name            VARCHAR(100),
        dob             DATE,
        sex             CHAR(1)
        passport        VARCHAR(20),
        address         VARCHAR(200),
        PRIMARY KEY (id),
        UNIQUE (passport)
);
```

```
CREATE TABLE Users (
        id              INTEGER,
        name            VARCHAR(100),
        dob             DATE,
        sex             CHAR(1)
        passport        VARCHAR(20),
        street          VARCHAR(100),
        city            VARCHAR(100),
        zip             VARCHAR(10),
        PRIMARY KEY (id),
        UNIQUE (passport)
);
```

Multivalued Attributes

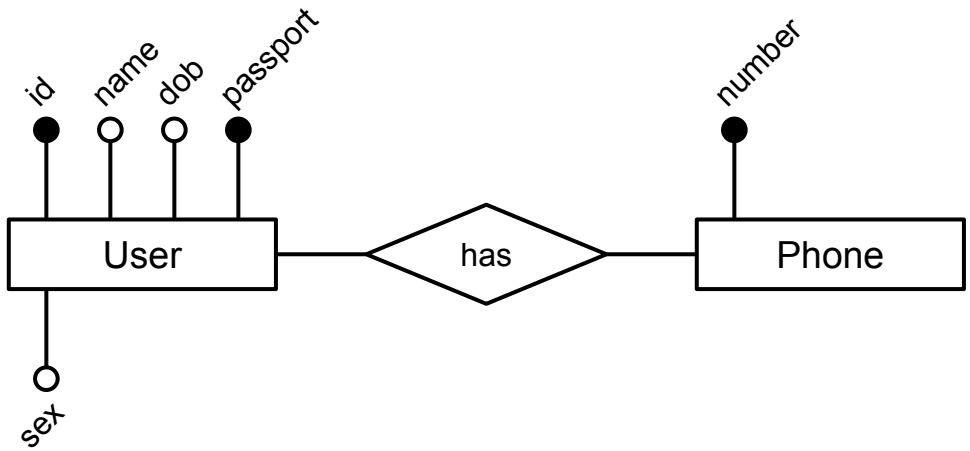
- Fixed number of single-valued attributes




```
CREATE TABLE Users (
        id              INTEGER,
        name            VARCHAR(100),
        dob             DATE,
        sex             CHAR(1)
        passport        VARCHAR(20),
        phone1          VARCHAR(20),
        phone2          VARCHAR(200),
        PRIMARY KEY (id),
        UNIQUE (passport)
);
```

Multivalued Attributes

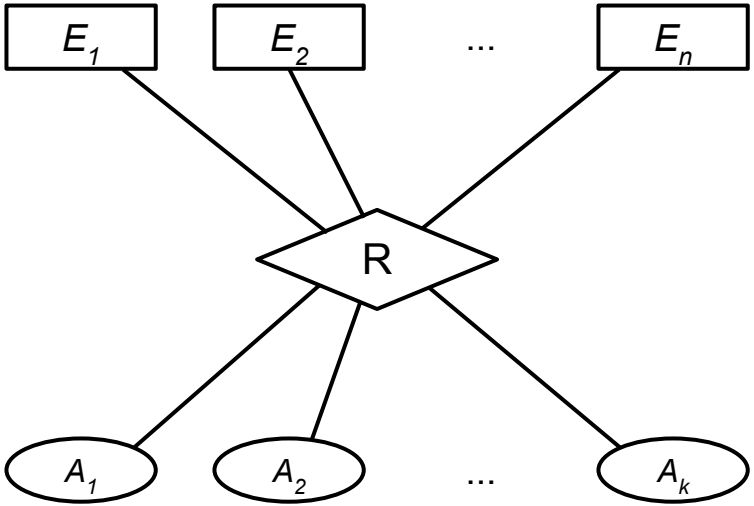
- Separate entity set for phone numbers




```
CREATE TABLE Users (
        id              INTEGER,
        name            VARCHAR(100),
        dob             DATE,
        sex             CHAR(1)
        passport        VARCHAR(20),
        PRIMARY KEY (id),
        UNIQUE (passport)
);
```

```
CREATE TABLE Phones (
        number          VARCHAR(20),
        user_id         INTEGER,
        PRIMARY KEY (number),
        FOREIGN KEY (user_id) REFERENCES Users (id)
);
```

Relationship Sets



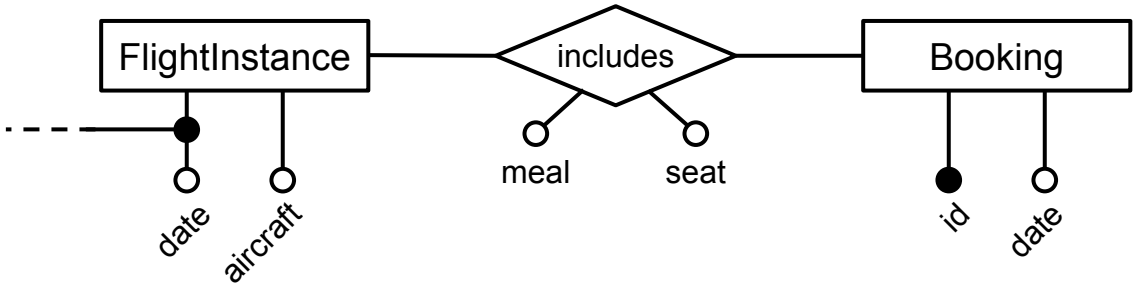
- General n-ary relationship set R
  - *n* participating entity sets E_1, E_2, …, E_n
  - *k* relationship attributes A_1, A_2, …, A_k
  - Let $Key(E_i)$ be the attributes of the selected key of entity set E_i

➜ **Attributes of relationship set *R***

- $Key(E_1)$, $Key(E_2)$, …, $Key(E_n)$ — key attributes of all participating entity sets E_i
- A_1, A_2, …, A_k — all relationship attributes of *R*

Cardinality: Many-to-Many (no mandatory participation)

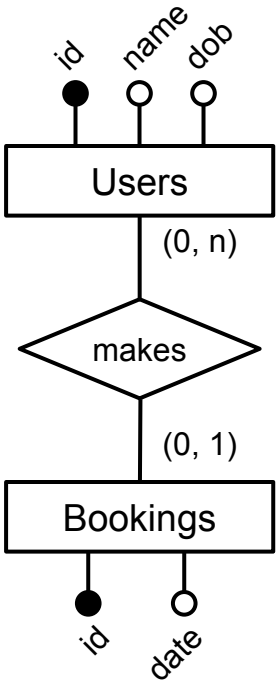
Quick Quiz: Where does "flight_nr" come from?




```
CREATE TABLE Includes (
        flight_nr       VARCHAR(10),
        flight_date     DATE,
        booking_id      INTEGER,
        seat            VARCHAR(10),
        meal            VARCHAR(50),
        PRIMARY KEY (flight_nr, flight_date, booking_id),
        FOREIGN KEY (flight_nr, flight_date) REFERENCES FlightInstances (flight_nr, date),
        FOREIGN KEY (booking_id) REFERENCES Bookings (id),
);
```

Cardinality: Many-to-One (no mandatory participation)

- **Approach 1**: Represent "makes" with a separate table
  - Similar to Many-to-Many but with different primary key!

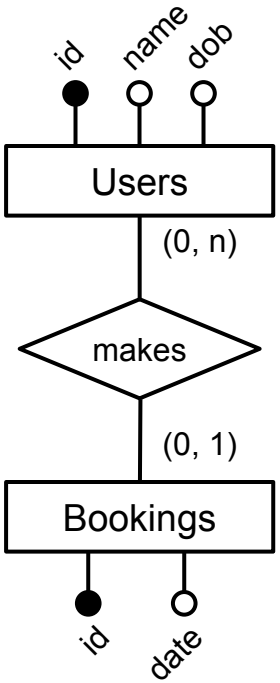



```
CREATE TABLE Makes (
        user_id         INTEGER,
        booking_id      INTEGER,
        PRIMARY KEY (booking_id),
        FOREIGN KEY (user_id) REFERENCES Users (id),
        FOREIGN KEY (booking_id) REFERENCES Bookings (id)
);
```

Cardinality: Many-to-One (no mandatory participation)

Quick Quiz: What is typically the preferred approach? 1 or 2?

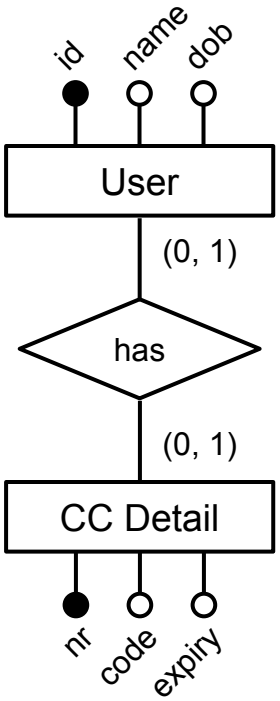
- **Approach 2:** Combine "makes" and "Bookings" into one table
  - Possible because given a booking, we can uniquely identify the user who made it




```
CREATE TABLE Bookings (
        id              INTEGER,
        date            DATE,
        user_id         INTEGER,
        PRIMARY KEY (id),
        FOREIGN KEY (user_id) REFERENCES Users (id)
);
```

Cardinality: One-to-One (no mandatory participation)

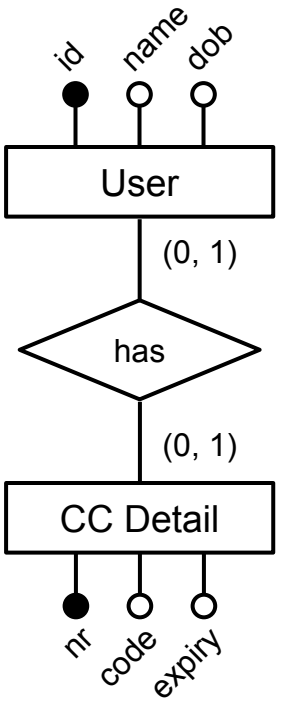
- **Approach 1**: Represent "has" with a separate table
  - Similar to Many-to-One but primary key can be chosen




```
CREATE TABLE Has (
        user_id         INTEGER,
        cc_nr           CHAR(16) UNIQUE,
        PRIMARY KEY (user_id),
        FOREIGN KEY (user_id) REFERENCES Users (id),
        FOREIGN KEY (cc_nr) REFERENCES CCDetails (id)
);
```

Cardinality: One-to-One (no mandatory participation)

- **Approach 2**: Combine "has" and "Users" or "has" and "CC Details"



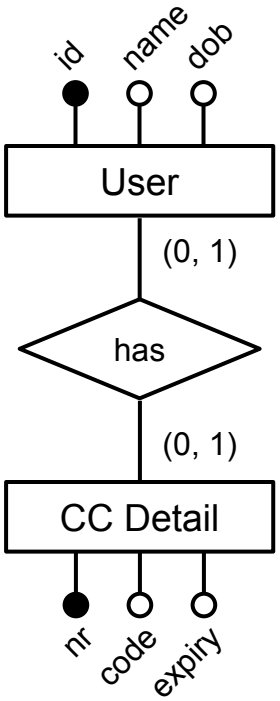

```
CREATE TABLE Users (
        id              INTEGER,
        name            VARCHAR(100),
        dob             DATE,
        cc_nr           CHAR(16) UNIQUE,
        PRIMARY KEY (id),
        FOREIGN KEY (cc_nr) REFERENCES CCDetails (nr)
);
```

```
CREATE TABLE CCdetails (
        nr              CHAR(16),
        code            CHAR(3),
        expiry          DATE,
        user_id         INTEGER UNIQUE,
        PRIMARY KEY (nr),
        FOREIGN KEY (user_id) REFERENCES Users (id)
);
```

Cardinality Constraints: One-to-One

Quick Quiz: What could be a downside of this approach? (Hint: security)

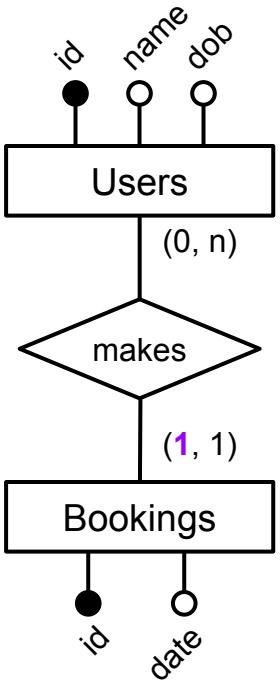
- **Approach 3**: Combine "has", "Users", and "CC Details"




```
CREATE TABLE Users (
        id          INTEGER,
        name        VARCHAR(100),
        dob         DATE,
        cc_nr       CHAR(16) UNIQUE,
        cc_code     CHAR(3),
        cc_expiry   DATE,
        PRIMARY KEY (id)
);
```

Cardinality & Participation Constraints

- **Approach 1** (separate table): fails to capture mandatory participation!

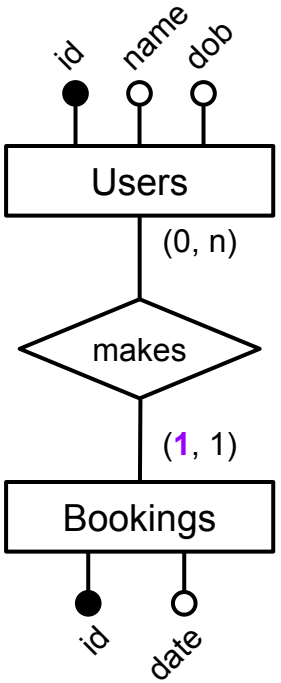



```
CREATE TABLE Makes (
        user_id         INTEGER NOT NULL,
        booking_id      INTEGER,
        PRIMARY KEY (booking_id),
        FOREIGN KEY (user_id) REFERENCES Users (id),
        FOREIGN KEY (booking_id) REFERENCES Bookings (id)
);
```

- Schema does not enforce mandatory participation of "Bookings" w.r.t. "Makes"
- e.g.: "Makes" can be empty while both "Users" and "Bookings" are non-empty

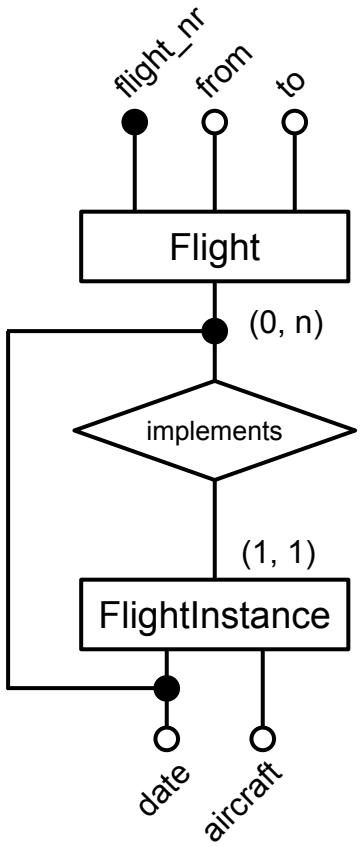
Cardinality & Participation Constraints

- **Approach 2**: Combine "makes" and "Bookings" into one table
  - Enforces total participation via NOT NULL constraint




```
CREATE TABLE Bookings (
        id              INTEGER,
        date            DATE,
        user_id         INTEGER NOT NULL,
        PRIMARY KEY (id),
        FOREIGN KEY (user_id) REFERENCES Users (id)
);
```

Weak Entity Sets



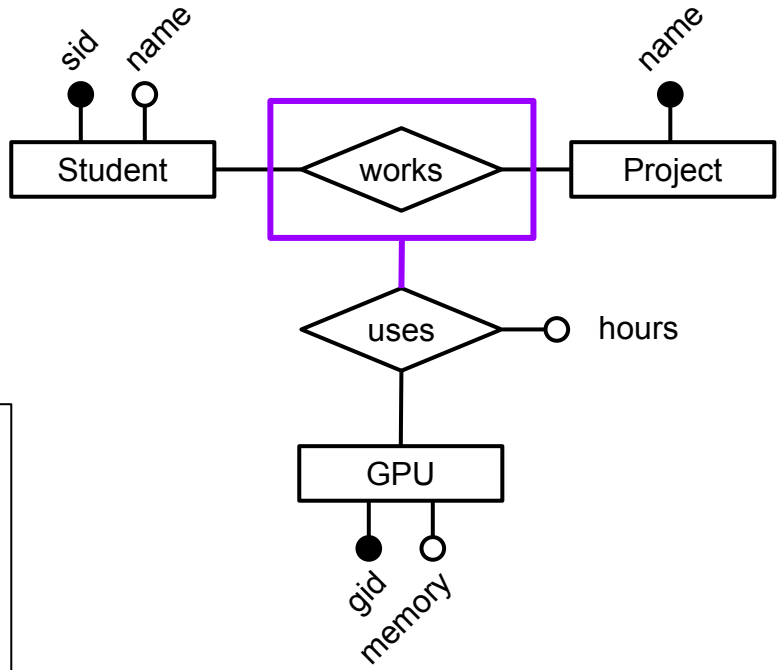

```
CREATE TABLE Flights (
        flight_nr       VARCHAR(10),
        from            VARCHAR(10),
        to              VARCHAR(10),
        PRIMARY KEY (flight_nr)
);
```

```
CREATE TABLE FlightInstances (
        flight_nr       VARCHAR(10),
        date            DATE,
        aircraft        VARCHAR(10),
        PRIMARY KEY (flight_nr, date),
        FOREIGN KEY (flight_nr) REFERENCES Flights (flight_nr)
                ON DELETE CASCADE
                ON UPDATE CASCADE
);
```

Aggregation — Relational Mapping

Schema definition of "uses"

- Primary key of aggregation relationship ➜ (sid, pname)
- Primary key of associated entity set "GPUs" ➜ gid
- Descriptive attributes of "uses" ➜ hours




```
CREATE TABLE Uses (
        gid             INTEGER,
        sid             CHAR(20),
        pname           VARCHAR(50),
        hours           NUMERIC,
        PRIMARY KEY (gid, sid, pname),
        FOREIGN KEY (gid) REFERENCES GPUs (gid),
        FOREIGN KEY (sid, pname) REFERENCES works (sid, pname)
);
```

ER Design & Relational Mapping — Basic Guidelines

- Guidelines for ER design
  - An ER diagram should capture as many of the constraints as possible
  - An ER diagram must not impose any constraints that are not required

- Guidelines for relational mapping
  (i.e., from ER diagram to relational database schema)
  - The relational schema should enforce as many if the constraints as possible using column and/or table constraints
  - The relational schema should not impose and constraints that are not required

Overview

- Entity Relationship Model
  - Overview + ER diagrams
  - Entity sets and attributes
  - Relationship sets
  - Cardinality & participation constraints
  - Dependency constraints: weak entity sets
  - Aggregation
- Relational Mapping
  - From ER diagram to database tables
- **Summary**

Summary

- Entity-Relationship (ER) model
  - Basic concepts: entity sets, relationship sets, attributes
  - Cardinality constraints and participation constraints
  - Extended concepts: ISA hierarchies, aggregation

  } Visualized using **ER diagrams**

- Relational Mapping
  - Mapping ER diagram to database schema
  - Not all constraints of ER diagram may be captured

- Outlook for next lecture
  - SQL for querying a database (recommendation: study RA)

Quick Quiz Solutions

Quick Quiz (Slide 3)

Quick Quiz (Slide 14)

Quick Quiz (Slide 17)

Quick Quiz (Slide 18)

Quick Quiz (Slide 34)

Quick Quiz (Slide 49)

Quick Quiz (Slide 51)

Quick Quiz (Slide 54)