INF 202
Intro to Data and Databases

Spring 2023
Prof. M Abdullah Canbaz

Class 4
January 31, 2023

Housekeeping

- Quiz 1 will be available this afternoon
  - You have to finish it within 45 minutes,
  - Please take it before Friday 11:59 PM,

Recap - Database System Lifecycle

- Initial Problem and Mission Statement
- Analysis of Use Cases
- Simple Data Model
- Iterate Until Requirements are Set
- Design
- Implementation
- Maintenance
- Repeat

Recap - Data Model

- A data model is a representation of how the different types of data interact. (Churcher p. 12)
- It will not be an exact depiction, it will always be based on assumptions and definitions.
- It is based on your problem statement and use cases.

Recap - Data Modeling Process

- Identify classes
- Identify relationships and cardinalities
- Identify and associate attributes
- Determine attribute domains (possible values)
- Determine candidate and primary keys
- Check for redundancy
- Validate against user transactions
- Review with users

Identify classes

A class is a template for storing data about a set of similar things

They can correspond to actual things or objects in the real world:

- Physical Items (Plant, Museum, Tennis Court, Inventory Item)
- Places (Country, City)
- People (Customer, Employee, Museum Patron, Tennis Player)

They can also be more abstract things:

- Categories (Plant Use, Museum Type)
- Operations/Events (Museum Visit, Tennis Match)
- Transactions (Purchase Order)

Customer				
First Name	**Last Name**	**Address**	**Age**	**Is Preferred**
Joe	Smith	100 Main St	32	Yes
Jenny	Jones	200 Oak Ave	45	Yes
Steve	Miller	50 Second St	22	No

Travel Package				
Starting City	**Ending City**	**Starting Date**	**Ending Date**	**Price**
NYC	Miami	10/1/2019	10/15/2019	$3,000
Los Angeles	Austin	10/20/2019	11/2/2019	$4,000
NYC	Miami	11/9/2019	11/21/2019	$3,000
NYC	Mexico City	11/30/2019	12/17/2019	$5,000

Package Order	
Date Ordered	**Confirmation Number**
4/1/2019	XCS32821
4/4/2019	FKQ28592
4/6/2019	VJI88549
4/9/2019	KZP00182

Data Modeling Process

- Identify classes
- Identify relationships and cardinalities
- Identify and associate attributes
- Determine attribute domains (possible values)
- Determine candidate and primary keys
- Check for redundancy
- Validate against user transactions
- Review with users

Data Modeling: Attribute Domains

- An attribute domain is “the set of allowable values for” that attribute.
- For example, an attribute domain of “T/F” means the possible values of that attribute are “true” or “false”. No other value can exist in that attribute (other than NULL).
- Other examples: a number; a string up to 256 characters in length; a dollar value; a date; a date and time

Data Modeling: Types of Attributes

- A simple attribute is indivisible; a composite attribute is made up of simple attributes.
- “Waist size” is a simple attribute -- it is one number.
- “Full name” is a composite attribute -- it is composed of first name, last name, and possible middle name.

Data Modeling: Types of Attributes

- A single attribute has one value for each entity; a multi-valued attribute has more than one.
- “Date Of Birth” is single. People can only have one date of birth; the value can’t be more than one thing.
- “Email” is multi-valued. People can have two emails: “john.doe@gmail.com” and “jdoe@albany.edu”.

Data Modeling: Types of Attributes

- A derived attribute is one that can be derived from other attributes.
  - In their simplest form, these can be an attribute plus a constant. For example, if **date of birth** is an attribute, ***age*** can be a derived attribute.
  - Derived attributes can summarize data - these are known as aggregate attributes
  - Derived attributes can be derived from other derived attributes.

Data Modeling Process

- Identify classes
- Identify relationships and cardinalities
- Identify and associate attributes
- Determine attribute domains (possible values)
- Determine candidate and primary keys
- Check for redundancy
- Validate against user transactions
- Review with users

Data Modeling: Candidate Keys

- A candidate key is “a minimal set of attributes that uniquely identifies each entity (class object) occurrence.”
- If the attribute value of an attribute is different for each class instance, that attribute can be a candidate key
  - ***Think uniqueness***, Is the value of the attribute really unique?
  - Can there be duplicates where we cannot distinguish one object from another?
- For example, my SSN is unique to me. Thus the “SSN” attribute can be a candidate key for the class “person”.

Data Modeling: Primary Keys

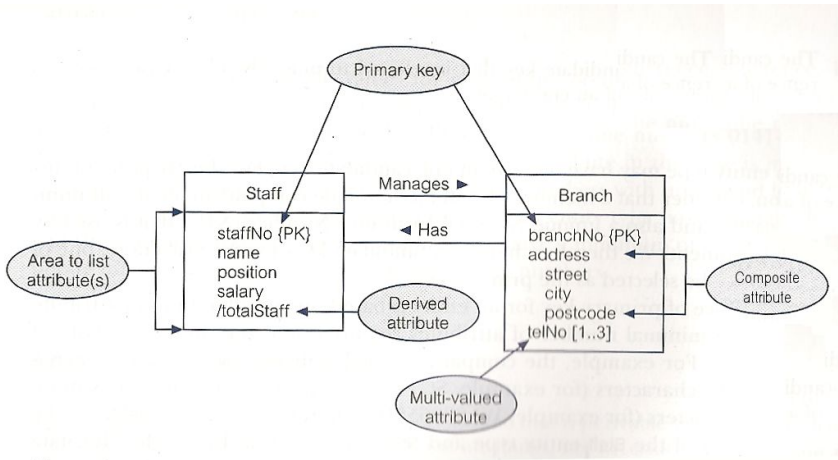
- A primary key is chosen from among the candidate keys (if more than one). Others are called alternate keys.
- Your full name can be a candidate key for purposes of identifying you. Your SSN could also be a candidate key. Only one can be the primary key.
- Primary keys are used as references in other tables, thus establishing relationships.

Data Modeling: Primary Keys

- Where only one candidate key exists for an class, the choice of primary key is easy.
- Some thought must be given to the choice of primary key where more than one candidate key exists. Need to set out the rationale for choosing one over the other.

Data Modeling: Attributes and Keys

- We see two classes
  - Staff & Branch
- Each class has a set of identified attributes that describe the class
  - Staff Class
    - name, position, salary
    - staffNo is a unique # generated to distinguish between staff members who may have same name.
    - staffNo satisfies uniqueness and hence candidate for primary key
- address is a composite attribute in Branch class since it is comprised of street, city, postcode.



Data Keys: Natural vs. Surrogate Keys

- A natural key is derived from application data. It has some meaning in the domain.
- A surrogate (or artificial) key is not derived from application data, and has meaning only to the database.
- Surrogate keys may be visible to the user or application, or they may be visible only to the database.

Data Keys: Advantages of Surrogate Keys

- Immutability
  - The key will never change, whereas a key modeled after a real-world attribute may
  - Applications will not lose their reference to a row
  - The uniqueness of a natural key may change due to new requirements
- Performance
  - An integer, or even a GUID, takes up less space than a string of characters
- Queries may be less complex

Data Keys: Disadvantages of Surrogate Keys

- Dissociation
  - The key has no relationship to any real-world concept. Other attributes must always be brought in to establish any meaning
- Potential duplication of natural keys. At the very least, ensuring uniqueness on the other keys may be more difficult.
- A discrepancy will exist between your physical implementation and the logical data model
- Queries may be less intuitive to understand

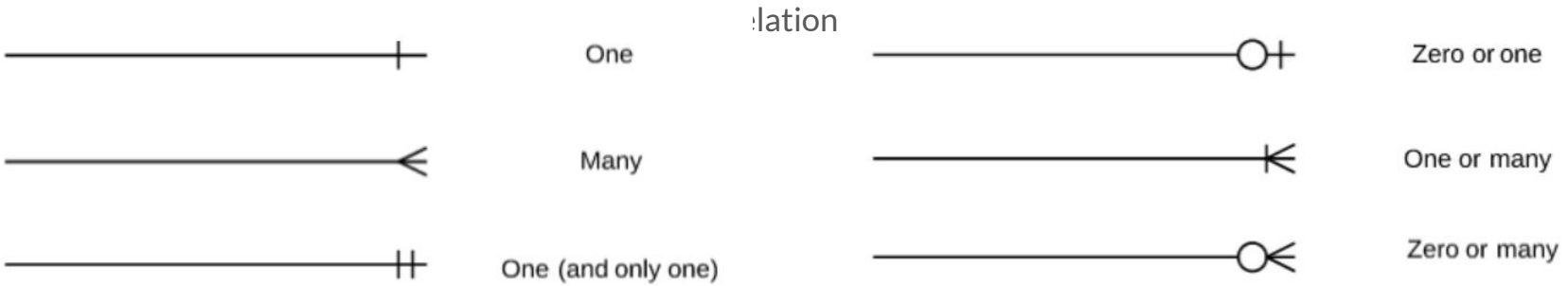
Data Modeling: Foreign Keys

- Let's recap some basics when representing relationships:
  - Each class (entity) is represented as a table
  - Each class (entity) has one to many attributes also known as fields or columns in a table
  - Each class (entity) has some candidate keys/fields which could become our primary key(s)

- We can use the identified primary keys to represent relationships between two classes using the concept of "foreign keys"
  - A foreign key is a field(s) that refers to the primary key field(s) in some other class/table

Class Diagrams vs. Entity-Relationship Diagrams

- A class diagram is a logical representation of the model, and an entity-relationship diagram (ERD) is a representation of how it will map to the database.
- A class diagram should not include surrogate keys (this is not part of the logical model) or foreign key fields.

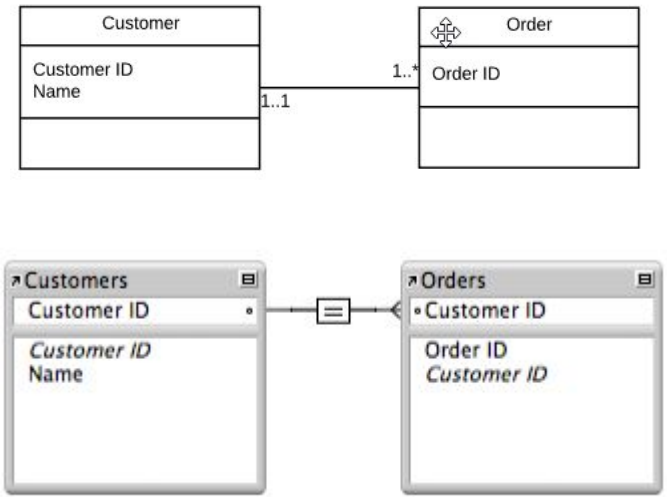


Data Modeling: 1-to-Many Relationships

- For a 1-Many relationship, the key field from the table representing the class at 1 end is added as a foreign key in the table representing the class at the Many end

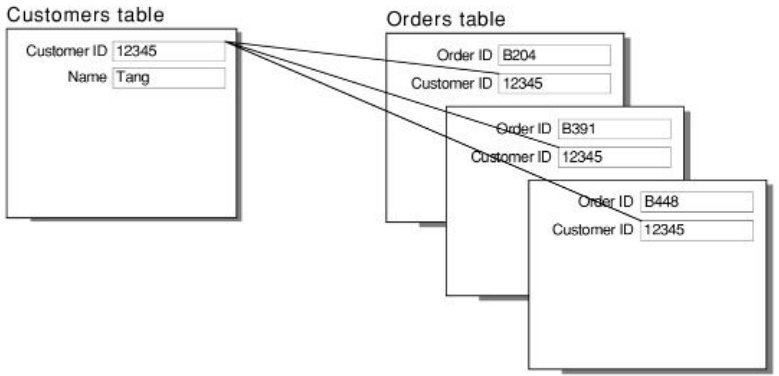
Data Modeling: 1-to-Many Relationships

- This represents a 1-to-Many relationship where one record in a table can be associated with one or more records in another table.
- Each customer (entity) can have many sales orders
  - This is represented by either 0…* or 1…*
- In this example, the primary key in the Customers table is designed to have unique values
- You add the primary key from the left side (Customers) as a foreign key in the other end of the table or class (Orders) to represent a 1-to-Many relationship.



Data Modeling: 1-to-Many Relationships

- This represents real records for a customer with ID 12345.

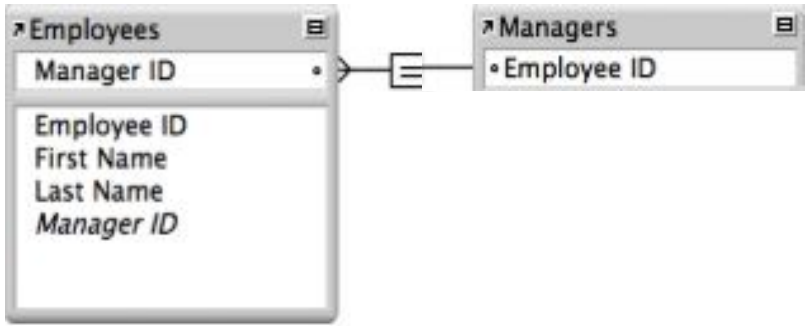


Self-Joins

- “A self-joining relationship (or self-join) is a relationship in which both match fields are defined in the same table/class
  - Match fields → primary and foreign key is the same field but with a different name to depict the relationship
- For example:
  - You have an “Employees” class/entity which maintains a list of all employee attributes for a company such as:
    - Employee ID (Primary Key)
    - First and Last Name
    - Contact Number
  - Let’s say you want to know which employees in the “Employee” table/class are also Managers
  - Every other attribute about the Manager is the same as a regular employee except he/she is playing the role of “Manager”
  - We solve this by adding a new attribute named “Manager ID” whose value is going to be same as the “Employee ID”
  - The “Manager ID” is going to be empty or NULL for employees who are not Managers.

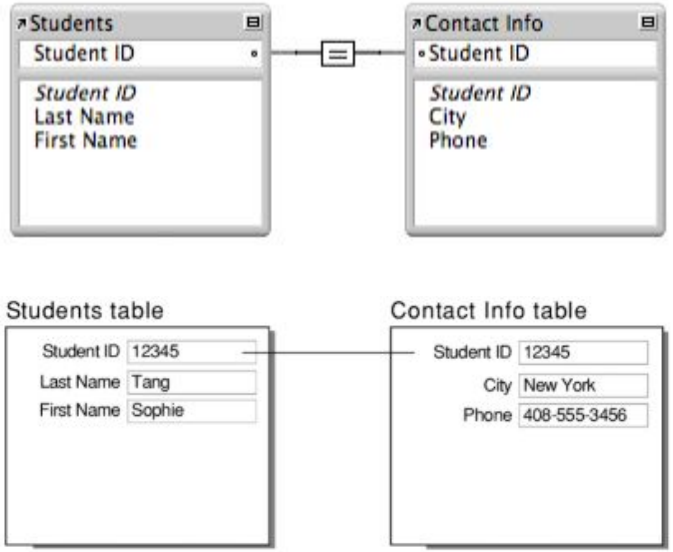
Self-Joins

- Another way to depict this would be to create a new class or table named “Managers” with Employee ID as foreign key.



One to One Relationships

- In a one-to-one relationship between two classes (tables), one record in a table/class is associated with one and only one record in another table/class
  - Represented by 1..1
- For example in a school database
  - each Student has only one Student ID, and each Student ID is assigned to only one person (Student) → 1..1
- This ensures that throughout your database, each Student always gets a unique Student ID and is represented as a “foreign key” in every table/class where it is needed.



Lucidchart Exercise

- Diagram the following scenario with both a class diagram and an entity-relationship diagram:

  > A hotel has 100 rooms and caters mainly to company retreat groups. A retreat group consists of a number of that company's employees as hotel guests, and the system should track which room each guest stayed in.

- The classes/entities are:
  - Retreat Group (Attributes: Company Name, Company Address, Dates of Retreat)
  - Guest (Name)
  - Room (Room Number, Size, Floor, # of Beds)