**Data Modeling Process**

**What is Data Modeling and the Design Process?**

Data modeling is the process of analyzing data and creating a design

Two Types of Databases:

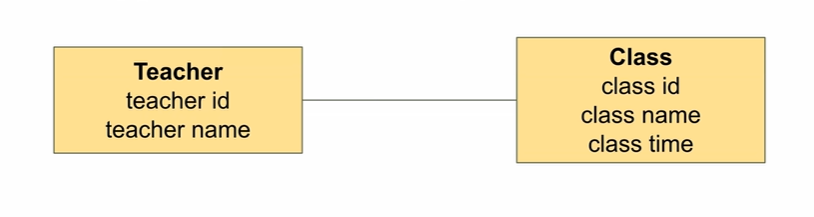
1. Relational Database: common, used for updating
2. Data Warehouse: used for reporting, separate course

Data Model Types:

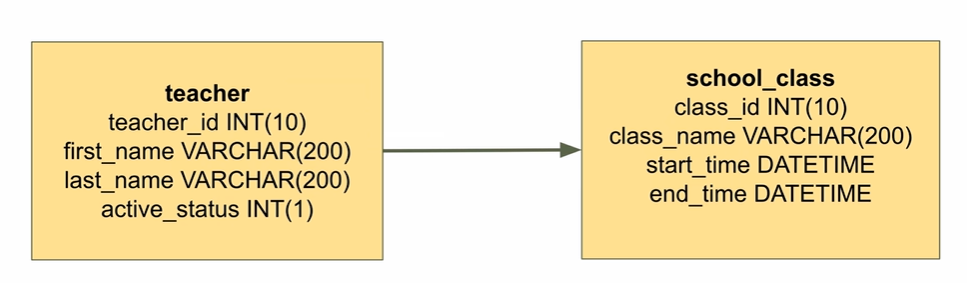
1. Conceptual: high level, shows relationships between objects, usually just names or concepts



1. Logical: objects at a more detailed level, information for each object



1. Physical: describes the internal schema of the database, table ames, column names, keys, relationships



**Determine the goal of the database:**

* What is the database trying to achieve?
* What is the purpose?

**Consider the current system, ask these questions:**

1. Find out if there is a current system or database

There may be a current system that your database is replacing or enhancing. The current system is a good source of requirements.

If yes, how is data stored for the current system? How well does it work?

1. Find out what the problems are with the current system or database (e.g. data quality, missing data)
   * Data Problems:
     + System may have bad quality data
     + Missing or inconsistent
     + Data retrieval and updating cab be slow
   * Designing from scratch
     + “Designing from scratch”, or designing where you have no system to start from
     + If the database is well designed, it should have minimal data quality issues
2. Ask if any historical data needs to be added to the database
   * Using Historical Data
     + May be an existing database
     + What about data not stored in the database?
     + Does it need to be added?
     + Definitions: people, subjects, or products
     + Transactions: enrollments, sales

**Gathering requirements of the Database**

Gathering requirements is all about finding out what the needs of the database are

1. Find out who you need to speak to for your database

* Personal project? Yourself
* Store? Store Manager, owner
* Work project? Business representatives, other team members

1. Find out what needs to be stored. Not everything needs to be stored
2. Find out if a history of changes needs to be stored

* What records have changed
* When they have changed
* Who changed them
* What changed them

**Finding Exceptions to the Rules**

“Usually” is very dangerous to the database. It describes something that is almost always true, but we need to know definitively true or false. Expand on it, find the details.

1. **Determine if there are any exceptions to your requirements**
2. **Question any specific field length or type restrictions**

Length such as customer ID

(10001st customer in a 0~9999 table, storing 2 digits for years)

Data Types:

* Assign different data types to fields when you design a database
* Types have restrictions
* Only be restricted if the rule can’t be broken

**Identifying Entities**

Use LucidChart for entity planning.[Documents (lucid.app)](https://lucid.app/documents#/documents?folder_id=recent)

* **Entities are things we want to store data about**
* **A good way to identify them is to look for nouns in our goal statement**

**Actions:**

1. Look at the goal statement for your database

“To help a **university** keep track of **students** that have enrolled, what **subjects** they are taking, and the **teachers** of those **subjects**”

1. Identify the **nouns** in the statement
2. Convert them to entities (remove duplicates singular form)

Use the singular of the word for now: university, student, subject, teacher

1. Create a new diagram with your entities

**Defining the Attributes:**

An attribute is something that is stored for each entity

Will likely be a column or field in a database table

**Action:**

1. For each of your entities, create a list of attributes for them
2. Determine what data type they would be

generic like text, number, or date, does not have to be database-specific

1. Update the diagram with the data types

**Normalization**

* A process of converting a database design into a standard format; conversion is done into a “normal form”, a standard of designing a database
* Many Benefits:

eliminating inconsistencies

removing duplicate information

Improve performance

* Three common normal forms are available
  + First Normal Form:
    - First stage of the normalization process

“Each set of columns must uniquely identify a row”

* + - A combination of all columns must be unique
    - More than one row? It’s not unique
    - Primary Key:
      * A field or column in a table that contains a unique value for the row
      * Value can only be stored once in that column
      * Not good if primary key must change
      * Customer numbers or student IDs are common examples
      * Names are not: they are not always unique

Relationships:

1. One to Many: one teacher can teach many subjects (bad example)
2. One to One: One person only has one spouse
3. Many to Many: Student & subjects. We use a joining table to manage this relationship.
4. Self-relationship: A table joins itself. Two records on one table. For example, an employee has a manager, and a manager also has a manager because each manager is an employee as well.

* Second Normal Form:
* Second stage of the normalization process “fulfill the requirements of the first normal form” and “each non-key attribute must be functionally dependent on the primary key”
* Third Normal Form:
  + “Fulfill the second normal form” and “has no transitive functional dependency”
  + Transitive Functional Dependency means every non-key attribute must be dependent on the primary key and the primary key only.

Example:

* Column A determines column B
* Column B determines column C
* Therefore: A>B>C and column A determines column C
* This dependency should be removed, and the relationship between B and C should be on a different table.

--------------------------------------------------------------------------------------------------------------------------

Yanhui example

* To keep records for monthly reporting of Yanshui’s rental apartments.

1. Keep track of tenants’ information for each room: **room**, **name**, **birthday**, **rental duration**, **rent**, **deposit, contract date**
2. Keep track of the **expenses** and transactions in organized categories
3. Calculate the monthly rent, electricity bill, remaining rent if moving out, how many days until the contract expires (so we can start to find new tenants)
4. Keep tracks of monthly payments for **rent** and **electricity**, figure out who hasn’t made the payment