Week 6

Database Design

Fundamentals

- **Database (DB):** Organized collection of data
  - Data can be structured or unstructured, depending on the DB type
- **Relational DB:** DB one or more tables that are related through shared fields
- **Non-Relational DB:** Typically for large volumes of data and/or unstructured data, like files of different types
- **Tables:** Data stored in columns and rows
  - **Tables** represent an **entity**
  - **Rows:** Each row of a table represents a single *record* of that entity
  - **Columns:** Each column represents a single *field* of that entity

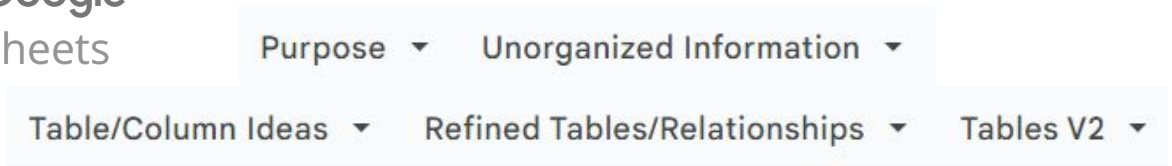
The Design Process[1]

1. **Determine the purpose of your database**
   - What is the high level purpose of this database?
2. **Find and organize the information required**
   - What information should be stored to fulfill the high level purpose?
3. **Divide the information into tables**
   - What are the entities to store?
4. **Turn information items into columns**
   - What are the individual data pieces that make up those entities (tables)?
5. **Specify primary keys**
   - What is the unique field (column) for each entity (table)?
6. **Set up the table relationships***
   - Do these tables relate? If so what is the connection?
7. **Refine your design***
   - Look for errors to inform improvements.
8. **Apply the normalization rules***
   - Review normalization rules and adjust as necessary.

[1] Microsoft - DB Design Basics - The design process

The Design Process: Strategize w/ Documents

- We can use pen&paper, docs, or diagrams to plan our database before implementing it
- The result from this planning is a database **Schema**
- **Schema:** Defines the structure and organization of your database
  - Includes logical constraints such as table names, fields, data types, keys, etc.
- The schema does not contain any data, it is simply a blueprint
- Should document the process step by step
  - Example:



Determining the Purpose of a Database[1]

Who will use the database? How do you expect to use the database?

Movie DB - Class Use - Simple

- Used by Instructor (or class members)
- Used to store information about movies for instructional purposes

Movie DB - IMDB Use - Specific

- Used by Employees and Accessed by the public
  - Used to store information about movies all over the world, to act as the go to international movie information source
  - Employees must be able to access only portions of the database they are authorized for
  - Public users should have READ ONLY access, accessible through the IMDB web app or authorized APIs

This step should define your guiding **mission statement** that informs your goals and decision making as you develop your DB

[1] Microsoft - DB Design Basics - The design process

Finding and Organizing the Information[1]

- Identify the information that is already present
  - What data do I currently track or store?
- Consider use cases of your software
  - What may my users or I use this software for? What data is needed for those use cases?
- Ask human questions about your scenario
  - What would you like to learn about your data? What are you curious about?
- Don’t limit yourself

Find any data ⟶ Organize later

[1] Microsoft - DB Design Basics - The design process

Group Information to determine Tables[1]

- Identify entities from your list of data ideas
  - What data is actually a representation of a thing with many data fields?

OR

- Group smaller related fields into tables
  - Are some of these fields individual properties? Can they be combined to create an entity?

[1] Microsoft - DB Design Basics - The design process

Turn Information into Columns[1]

- With the entities identified, round out the table with other related fields
  - What information in your list relates to the tables you've created?
- Refine your columns
  - Are some of your fields heavy with information?
  - Could you split them up into multiple columns?
  - Data with multiple parts, like an address, can be stored in multiple columns instead of just 1
  - Example:

address
205 Humber College Blvd, Etobicoke, ON M9W 5L7

number	name	city	province	postal_code
205	Humber College Blvd	Etobicoke	ON	M9W 5L7

- Don't include calculated data
  - Data that could be counted, sumed, averaged, or calculated in any way are typically too dynamic to be stored. It is best to calculate this information using SQL queries when needed.

[1] Microsoft - DB Design Basics - The design process

Column Data Types[1]

- There are 3 main data type categories in MySQL
- Every column in SQL must be declared with a type from these 3 categories
- The common types are listed here, descriptions are on the following slides

String Types

VARCHAR(*size*)

CHAR(*size*)

ENUM(*val1, val2, val3, ...*)

Number Types

INT(*size*)

DECIMAL(*size, d*)

FLOAT(*size, d*)

BOOL

Date/Time Types

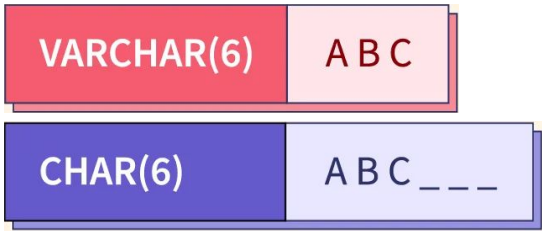
DATE

DATETIME(*fsp*)

TIME(*fsp*)

[1] W3Schools - MySQL Data Types

Column Data Types[1]



String Types

VARCHAR(*size*)

- Most common string data type for columns of string data
- *size* specifies maximum length in characters, which can be from 0 to 65535
- VARCHAR only uses the memory space required, it is a VARIABLE length string

CHAR(*size*)

- *size* specifies maximum length in characters, which can be from 0 to 255
- CHAR always uses the memory space requested, it is a FIXED length string
- Useful for strictly formatted columns, like 3 character country codes or postal codes

ENUM(*val1, val2, val3, ...*)

- Only allows column to hold strings that are defined in the list
- Useful for restricted columns, like province lists or dropdown menus

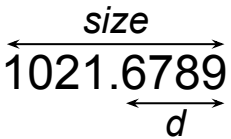
[1] W3Schools - MySQL Data Types

Column Data Types[1]

Number Types

INT(*size*) – Signed range: -2147483648 to 2147483647 – Unsigned range: 0 to 4294967295

- Most common data type for columns of whole number data
- *size* specifies maximum digit length of the number
- Used for numerical data, including id column values



DECIMAL(*size, d*)

- Most common data type for columns of decimal point numbers
- *size* limited to 65 digits, *d* specifies the number of digits right of the decimal point (max 30)
- Use when working with financial or accounting data, like currency values

FLOAT(*size, d*) – Range: -1.79E+308 to 1.79E+308

- Used where extreme numeric values are encountered, like scientific applications

BOOL – Zero is considered as false, nonzero values are considered as true

- Used to represent on/off, true/false, is/is not values

[1] W3Schools - MySQL Data Types

Column Data Types[1]

Date/Time Types

DATE – Range: '1000-01-01' to '9999-12-31'
- Format: YYYY-MM-DD — Ex: 2024-10-09

DATETIME(*fsp*) – Range: 1000-01-01 00:00:00' to '9999-12-31 23:59:59'
- Format: YYYY-MM-DD hh:mm:ss
- A date and time combination — Ex: 2024-10-10 02:00:00 (2am)

TIME(*fsp)* **–** Range: '-838:59:59' to '838:59:59'
- Format: hh:mm:ss — Ex: 14:00:00.000000 (2pm)

fsp
(fractional seconds precision)

fsp (**f**ractional **s**econds **p**recision)

- number of digits following the decimal point for fractional parts of seconds, from 0 to 6

[1] W3Schools - MySQL Data Types

Keys - Specify Primary Keys

- Every table should include a column that uniquely identifies each row
- The unique column is the **primary key** for the table
  - Primary key is referenced *(by FK*)* in **creating** the **links** (relationships) **between tables**
  - The primary key must be different in every row, there **cannot be duplicated values**
  - Primary keys **should not change** their value
- Primary Keys can be any column in your table, as long as they are unique
  i. The key can be a column of data that is unique
  ii. The key can also be a combination of columns that when combined make a unique value
  iii. Typically creating an “id” column as the unique identifier is easiest

i. **Primary Key**

student_number	name
n01234567	Auston Matthews

ii. **(Primary) Composite Key**

course_code	instructor
HTTP5126-A	Matthew Bebis

iii. **Primary Key**

bus_id	route
1	508

Keys - Specify Foreign Keys[1]

- **Foreign Key:** a column(s) in one table **that refers to the Primary Key** in another table
- Foreign Keys can reference any type of primary key from other tables
  - i. Unique Data PK*
  - ii. Composite PK*
  - iii. id PK*

i.

PK

student_number	name
n01234567	Auston Matthews

Foreign Key

student_number	address
n01234567	40 Bay St.

ii.

(Primary) Composite Key

course_code	instructor
HTTP5126	Matthew Bebis

(Foreign) Composite Key

course_code	instructor
HTTP5126	Matthew Bebis

iii.

PK

bus_id	route
1	508

FK

bus_id	driver
1	Sean Doyle

[1] What are Foreign Keys?

Keys - Specify Foreign Keys

- Purpose of foreign keys is to create a relationship between two tables that:
  - Maintain Consistency
    - Ensure values in one table correspond to valid rows in another table, preventing data inconsistencies
  - Can Prevent Invalid Data
    - Blocks attempts at adding data with foreign key that does not have matching primary key
  - Can Enable Cascading Actions
    - Changes in the parent table automatically apply to the related rows in the child table

Relationships will fall into 1 of 3 categories*

one-to-one relationship	one-to-many relationship	many-to-many relationship

Keys - Naming Conventions

- **Primary Key**
  - Should follow all column naming conventions (Week 2) *AND*
  - Should be named: table it is in, followed by *_id*
- **Foreign Key**
  - Should follow all column naming conventions (Week 2) *AND*
  - Should be named: table it is referring to, followed by *_id*

Primary Key ↓ | **Foreign Key** ↓

movie

movie_id (PK)	title	director_id (FK)
1	The Banshees of Inisherin	1
2	The Truman Show	2
3	Eternal Sunshine of the Spotless Mind	3
4	The Dark Knight	4

Primary Key ↓

director

director_id (PK)	name
1	Martin McDonagh
2	Peter Weir
3	Michel Gondry
4	Christopher Nolan