

Introduction to Data Access and Storage

Thomas Rosenthal - DSI @ UofT

Module 01

Getting Started:

Welcome

What is SQL?

Data Modelling

Getting Started:

Welcome

What is SQL?

Data Modelling

Welcome

About Me

Course Content

Quick Technical Check

Welcome

About Me

Course Content

Quick Technical Check

About Me

- Fell into SQL in my first job out of undergraduate in 2013
- Worked as a Data Analyst in US healthcare until moving to Canada in 2018
 - Challenging data
 - Easy to see impact of good queries
 - I was pretty naive about data, probably lucky I didn't break anything 🤪
- Worked as a Data Engineer at Plan Canada managing a CRM Data Warehouse backend
- Went back to UofT to do my Master of Information
 - Wanted to do less SQL, more R and python
 - SQL seemed like it was on its way out...NOPE, I was just wrong
- Reluctant Data Scientist (*do we even need this model?*)
 - Currently doing Data Governance and Ethics implementation at Thomson Reuters
- This is Alto👉



Welcome

About Me

Course Content

Quick Technical Check

Course Content

- Data Modelling
- Data Structures, Schemas, Basic Data Management, Normal Forms
- Basic SQL Syntax
- Advanced SQL Syntax
- Importing and Exporting Data to and from SQL
- SQL's relationship to the Machine Learning Pipeline
- Data Stewardship, Ethics, and SQL in the Wild
- Industry Case Study with Zia Barbar

Github Repo

https://github.com/UofT-DSI/02-intro_sql

- Schedule
- These slides (HTML & PDF)
- Our database for living coding
- All in class code
- Homework (and answers, posted afterwards)
- Assignment details and rubrics

Course Content

- This course is an *Introduction* to SQL
- At the end of the course, I hope you will:
 - Feel comfortable with SQL
 - Know how to search for the right thing on Stack Overflow
 - Read documentation
- We won't cover advanced topics like:
 - Stored Procedures, Triggers, Jobs
 - DBA work (monitoring, server setup, etc)
 - Complex ETL or tooling



The screenshot shows the SQLite Database Browser interface with a list of employees. The code at the top defines a table named 'Employee' with columns: id, first_name, last_name, city, and hire_date. Below the table definition, several SQL queries are run against the database:

```
1 select * from employee;
2 where city = 'Redmond';
3 and date = '2008-05-17';
4 and type = 'Manager';
5
6 -40100128 inserted Security storage stores that there were 0 witnesses.
7 -The first witness line at the last time in 'Northwestern Dr'.
8 -The second witness, named Accused, lives somewhere in 'Practis Ave'. Redmond
9
10 select * from person;
11 where address_number < 1;
12 select address_number from person;
13 where address_street_name like '%Northwestern Dr%';
14 order by address_number desc;
15 limit 1;
16
17 -40071 Accused Miller 400378 105 Practis Ave 02877140
18 -14887 Murry Schapiro 110000 4008 Northwestern Dr 02888840
19
20 select * from customer;
21 where person_id is (40071);
22
23 -I inserted a person and then inserted a customer with the same name. Now they're linked.
24 -The customer is customer and the person is 'Miller'. Only gold members have silver traps.
25 -The person has a wife and she's a silver member.
26
27 select * from customer;
28 where id like '400%';
29 and membership_level = 'gold';
30
31 -40071A 028612 Joe Gerwinkle 02800000 gold
32 -400550 07512 Jeremy Stevens 02800000 gold
33
34 select * from person;
35 where location_id is null;
36 select id from driver_license;
37 where place_holder like '00042P%';
38
39 -401798 Thelma Choate 0847981012 Phi Dn 1277888871
40 -401818 Jeremy Stevens 40037817 000 Washington Pl Apt 5A 0210000071
41 -781000 Maurice Whaley 12077791 100 Park Rd 1277888871
42
43 -401818 I was forced by a witness with a lot of money
```

Software: DB Browser for SQLite

Course Content

Homework (ungraded, but worth doing)

- At the end of each SQL module, I will provide you with 3-10 queries to write
 - Covers the topics of the modules
- Review answers in Tutorials with Salaar
 - I may go over one or two queries at the start of class if desired
- Designed to be relatively easy
 - Reaffirms what we wrote together
 - Doing work on your own helps reinforce the learning

Course Content

Grading (specific rubrics tbd)

- Three Assignments
 1. Data Model Design, due at the end of week 2, start in week 1: 40%
 2. SQL Skills, due at the end of week 3, start in week 2: 40%
 3. Class Participation (reflection), due at the end of week 3, work on any time: 10%
- Class Attendance: 10%
 - Let Salaar or myself know if you are unable to attend a lesson
 - Code along!! Best way to learn.

Welcome

About Me

Course Content

Quick Technical Check

Quick Technical Check

(a slide where we make sure everyone has DB Browser for SQLite installed)

If not, please download it here: sqlitebrowser.org/dl/

For live coding:

- Please download/fork the FarmersMarket.db from our GH repo:
 - https://github.com/UofT-DSI/02-intro_sql/tree/main/SQL
- Open it in SQLite with the "Open Database" button and navigate to wherever you have saved it

Good to go?

If not, please message Salaar

Getting Started:

Welcome

What is SQL?

Data Modelling

What is SQL?

SQL

Flavours

Environments for SQL

What is SQL?

SQL

Flavours

Environments for SQL

SQL

SQL Fundamentals

- SQL: Structured Query Language
 - Pronounced as either S.Q.L. (ess-cue-ell) or “sequel”
- SQL is a *query* language rather than a programming language
 - Querying is closer to telling a computer *what you want*, rather than *what is has to do*
 - SQL code is often less reproducible than other programming languages because it's domain specific
 - Some SQL code, especially more advanced procedural code, is reproducible within the same flavour
 - SQL's domain is databases and is based on set theory
- Designed to manage data within Relational Database Management Systems (RDBMs), e.g.
 - MSSQL
 - Oracle DB
 - MySQL/MariaDB
 - PostgreSQL

SQL

SQL Formatting

- Like other programming/query languages, SQL has reserved keywords/commands to perform instructional operations
 - Generally, these keywords are written in all caps: `SELECT`
 - Most modern interpreters no longer require this, but it is the expected standard
- All statements/queries should be ended with a semicolon
 - A few SQL constructs (like common table expressions, we'll get to these later) require them, otherwise they are optional
 - I'll almost certainly forget to use them
 - There's some debate on whether or not it's best practice

SQL

SQL Formatting

- In SQL, white space and/or line breaks do not matter
 - Readability is important
 - Try to keep SQL statements to a reasonable screen width
 - Use sensible line breaks
 - Offset subqueries with indents
- Code is commented in/out with `--` rather than `#`
- Code blocks can be commented out with `/* */`

```
/*
somecode spanning
multiple lines
*/
```

What is SQL?

SQL

Flavours

Environments for SQL

Flavours

- RDBMs differ from one to the next. This is manifested in many ways:
 - different keywords
 - e.g. return only 10 rows: `SELECT TOP 10...` vs `SELECT ... LIMIT 10`
 - different syntax
 - e.g. not equal: `!=` or `<>` (or both)
 - other, more nuanced/complex differences
 - e.g. architecture, data types, etc
- We are using SQLite:
 - Super easy to get setup
 - Requires almost no overhead
 - Open source, *free*
 - Similar enough in syntax to learn on
 - Used all over the world and in many applications
 - e.g. Firefox uses a SQLite backend to write a user's history locally

Flavours

- Broad observations about Open Source:
 - Excellent at what they are designed for
 - Varying data types (SQLite has some unique ones!)
 - Not every command exists, but workarounds are usually possible
 - Some utilize RDBMs that feel extremely outdated
- Broad observations about enterprise systems:
 - Powerful and designed to handle edge cases
 - Feel a bit more refined
 - Can be version dependent
 - Tend to "lock in" business/organizations
 - Migration is costly, sometimes outrageously so
 - Newer players (Snowflake, Databricks, etc) and cloud providers (Azure, AWS, Google, etc) offer a lot more functionality than just database querying

What is SQL?

SQL

Flavours

Environments for SQL

Environments for SQL

Databases

- Relational databases are a collection of tables, views, procedural code, and other SQL-assisting artefacts
 - Generally the data stored in a database will related to a real-world concept
 - Backends to data-collecting systems are often databases
 - e.g. CRMs, EMR software, ERPs, web-based applications
 - Usually not connected to other databases unless deemed necessary
 - Often transactional, meaning data is actively being written to by frontend systems
 - Tables are normalized (we'll get into Normalization Forms later in the course)
- There are also non-relational databases, often referred to as NoSQL
 - We won't cover these
 - Common tools include: Amazon DynamoDB, Azure CosmosDB, MongoDB, Google Cloud Datastore

Environments for SQL

Data Warehouses and Data Marts

- Data Warehouses are highly structured collections of (usually tabular) data
 - Data has been processed for a specific purpose, e.g. analytics
 - Data has been centralized
 - Often with the assistance of ETL (Extract, Transform, Load) tools
 - Have a lot of overhead, require governance, and strict rigidity
 - Tables are denormalized
 - Very common for enterprises, but losing traction in many industries
- Data Marts are created from Data Warehouses to focus on a single subject
 - Designed to make Data Warehouses easier to use
 - Data is structured, but flexibility is driven by the purpose of the Data Mart
 - Some denormalized tables may be normalized or even undergo even greater normalization
 - *The subject/purpose of the Data Mart might drive these types of decisions*
 - Common for enterprises that have Data Warehouses

Environments for SQL

Data Lakes and Data Swamps

- Data Lakes allow on-demand access to raw, semi-structured, structured, and unstructured data
 - Not defined by a specific purpose
 - Highly scalable
 - Can be transactional, if systems are designed to produce outputs into data lakes
 - Commonly data sources for machine learning pipelines live in data lakes
 - Inexpensive compared to Data Warehouses
 - Increasingly common for enterprises to shift towards Data Lakes, especially with support from newer tools like Snowflake, Databricks, etc which can maximize the analytical value of a Data Lake
- Data Swamps...are poorly governed Data Lakes
 - Lack of documentation, lack of governance, poorly designed Data Lakes become Swamps
 - Avoid building these

Getting Started:

Welcome

What is SQL?

Data Modelling

Data Modelling

Relational Database Management Systems

Data Models

Structure of Data

Constraints

(...)

Data Modelling (continued)

Entity Relationship Diagrams

Attributes of an ERD: Entities & Relationships

Relationship Examples

Conceptual, Logical, Physical Models

Assignment 1: Design a Logical Model

Data Modelling

Relational Database Management Systems

Data Models

Structure of Data

Constraints

(...)

Data Modelling

Relational Database Management Systems

- Relational Database Management Systems (RDBMs) are software designed to:
 - Store large amounts of data
 - Utilize a query language to allow easy retrieval of the data
 - Allow multiple users to access the data simultaneously
 - Manage permissions for data access
 - Mitigate data corruption and unauthorized access
- Generally, data is stored in a *database*
 - a database is a collection of information
 - within a database, a collection of objects (e.g. tabular data "tables") is stored
- RDBMs allow users to define interactions between these objects, such as:
 - Establish the relationship between objects
 - Define procedural scripts to query specific data or trigger an action
 - Schedule routine work (e.g. procedures to run, maintenance, etc)

Data Modelling

Relational Database Management Systems

Data Models

Structure of Data

Constraints

(...)

Data Modelling

Data Models

- A data model is a notation for describing data or information
- Data models consist of:
 - Structure of the data
 - Operations
 - Constraints on the data
 - Relationships

Data Modelling

Relational Database Management Systems

Data Models

Structure of Data

Constraints

(...)

Data Modelling

Structure of Data

- SQL is comprised of tables

Breed	Affectionate w/ Family	Good w/ Other Dogs	Shedding	Coat Type	Coat Length	Playfulness	Energy
Pugs	5	4	4	Smooth	Short	5	3
Akitas	3	1	3	Double	Medium	3	4
Samoyeds	5	3	3	Double	Long	5	4

- Tables have Attributes and Observations
 - In SQL we call Attributes "Columns"
 - e.g. Breed, Coat Type, Coat Length
 - and Observations "Rows"
 - e.g. Samoyed, Double, Long
- SQL databases require tables to be named
 - e.g. We can call this table "breed_traits"

Data Modelling

Structure of Data

- Columns are defined (and restricted, i.e. constrained) by data types
- Common ones include:
 - **INT** (integers: 1,2,3,-1,-2,-3)
 - most systems conserve storage space specifying their range
 - **FLOAT, DECIMAL, REAL** (decimal: 5.5, 3.333333)
 - **VARCHAR, NVARCHAR, TEXT** (text strings, with a maximum length associated: 'abc')
 - **DATE, DATETIME, TIME** (dates and times: '2023-01-09",11:11:11.000')
- These may vary slightly by flavour (in SQLite they are simpler and less restricted)
- Data types are important:
 - They affect operation speed, storage size, data validity
 - Speed: it's much computationally less expensive to compute smaller values
 - Storage: small is usually better, but the wrong size will affect systems, for example Psy's Gangnam Style exceeded 2,147,483,647 (32-bit signed, $2^{32}/2-1$) views, causing YouTube to expand the view counter to 64-bit (~9.2b if signed)
 - Validity: ensures columns contain the right type of data for operations, e.g. avoiding $5 + 'ten' = ??$

Data Modelling

Relational Database Management Systems

Data Models

Structure of Data

Constraints

(...)

Data Modelling

Constraints

- Data Models also specify constraints
- Constraints are rules that must be followed:
 - Referential-Integrity constraints
 - Ensure that values in one table have corresponding values in another table
 - Attribute Constraints
 - Ensure that certain types of values are always consistent within columns
 - May also ensure whether values are unique, not missing, etc

Data Modelling

Constraints

- **NULL** and **NOT NULL**
 - if a value can be missing or not
- **UNIQUE**
 - all values are different
- **PRIMARY KEY (PK)**
 - Ensures each value in a column is unique within the table (e.g. an ID field)
 - One PK per table
 - Cannot be NULL
 - Ensures database integrity by restricting record deletion
- **FOREIGN KEY (FK)**
 - Creates a linkage between a column in one table and a column in another table
 - Generally, foreign keys are linked to primary keys
 - Sometimes share the same name as the linked column, but this isn't required
 - Linkage requires data types to be the same
 - As many FKS as needed per table
 - May be NULL
 - Record can be deleted

Data Modelling (continued)

Entity Relationship Diagrams

Attributes of an ERD: Entities & Relationships

Relationship Examples

Conceptual, Logical, Physical Models

Assignment 1: Design a Logical Model

Data Modelling

Entity Relationship Diagrams

- Entity Relationship Diagrams (ERDs) are diagrams depicting the structure of tables within database
 - This both *identifies the tables* and *describes their relationships*
- ERDs are useful for:
 - database design
 - debugging
 - writing logical, consistent, and efficient queries
- There three levels of details for ERD depictions:
 - Conceptual model
 - Logical model
 - Physical model

Data Modelling (continued)

Entity Relationship Diagrams

Attributes of an ERD: Entities & Relationships

Relationship Examples

Conceptual, Logical, Physical Models

Assignment 1: Design a Logical Model

Data Modelling

Attributes of an ERD Entity

- For a given table:
 - Name
 - Relationship to another table
 - Column Names
 - Column Types
 - Primary Keys (if present)
 - Foreign Keys (if present)

Attributes of a Relationship

- Defines which columns are related
- Defines what type of relationship exists:
 - One-to-One:
 - One-to-Many
 - Many-to-Many

Data Modelling (continued)

Entity Relationship Diagrams (ERDs)

Attributes of an ERD: Entities & Relationships

Relationship Examples

Conceptual, Logical, Physical Models

Assignment 1: Design a Logical Model

Data Modelling

Relationship Examples

One-to-One: where a given row within a table is associated with only a single row in another table

Table 1: Country – Table 2: Capital City

Table 1:Country	Table 2:Capital
Canada	1:1 Ottawa
USA	1:1 Washington DC
Mexico	1:1 Mexico City

Data Modelling

Relationship Examples

One-to-Many: where a given row within a table can be referenced by multiple rows in another table

Table 1: Country – Table 2: States

Table 1:Country	Table 2:States
Canada	1: ∞ Alberta
Canada	1: ∞ British Columbia
Canada	1: ∞ ... (11 more rows)
USA	1: ∞ Alabama
USA	1: ∞ Alaska
USA	1: ∞ ... (48 more rows)
Mexico	1: ∞ Aguascalientes
Mexico	1: ∞ Baja California
Mexico	1: ∞ ... (30 more rows)

Data Modelling

Relationship Examples

One-to-Many: where multiple rows within a table can be referenced by multiple rows in another table

Table 1: Employee – Table 2: Employee Type

For this example, imagine the following:

Previously, A was an instructor at DSI (python) *and* a TA at DSI (SQL). Thomas remains the instructor for SQL. Salaar is now the TA. If a table contained Employee ID and Employee Type ID, A would be listed twice.

Employee	Employee ID	Type	Type ID
Thomas	001	Instructor	10
A	002	TA	11
Salaar	003		

Employee ID	Type ID
001	∞:∞ 10
002	∞:∞ 10
002	∞:∞ 11
003	∞:∞ 11

We'd create additional Many-to-Many relationships if we created a course table, because A and Salaar are involved in multiple courses!

Data Modelling (continued)

Entity Relationship Diagrams

Attributes of an ERD: Entities & Relationships

Relationship Examples

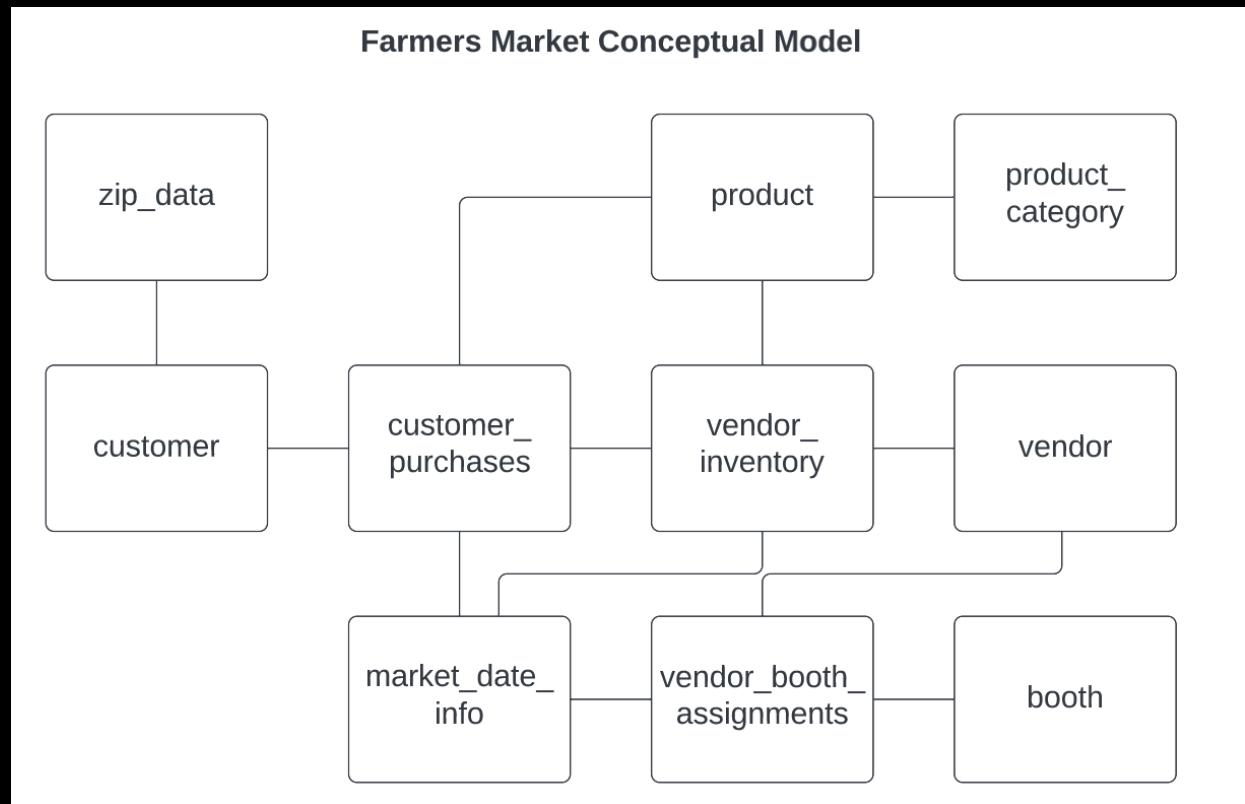
Conceptual, Logical, Physical Models

Assignment 1: Design a Logical Model

Data Modelling

Conceptual Models

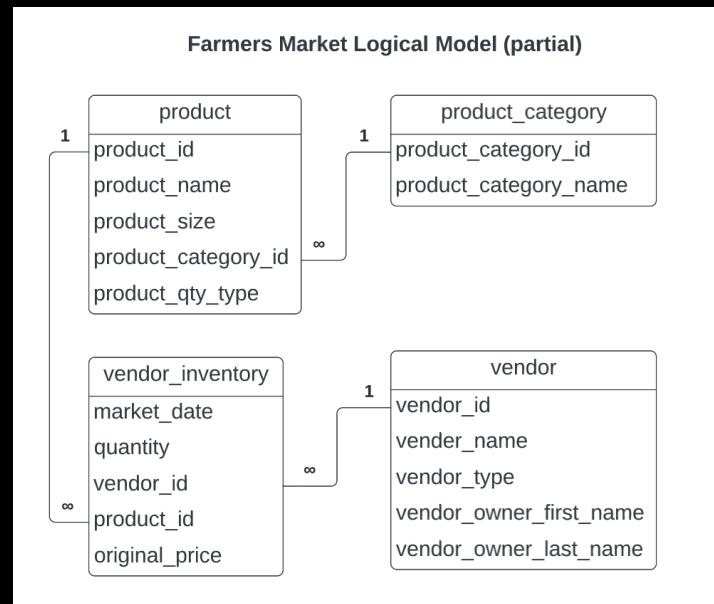
- Defines the tables (objects/entities) and their relationships
- Our Farmers Market database:
 - Ten tables
 - Relationships between these tables
 - e.g. product and product_category: *what type of thing a product is*
 - product and customer_purchases: *what product a customer has bought*
 - product and vendor_inventory: *what products each vendor has on available*
 - Not all tables share a relationship to one another, but all tables have at least one relationship



Data Modelling

Logical Models

- Adds additional detail to the conceptual model by adding column names for each table
- Often indicates the type of relationship
 - One-to-One
 - One-to-Many
 - Many-to-Many
- Our (partial) Farmers Market database:
 - product shares a One-to-Many relationship with vendor_inventory on product_id
 - product_category shares a One-to-Many relationship with product on product_category_id
 - _____ shares a _____ relationship with _____ on vendor_id



Data Modelling

Physical Models

- Adds additional detail to the logical model by adding key type and column data type
- Our (partial) Farmers Market database:
 - vendor_id is the PK for vendor, which shares a One-to-Many relationship with vendor_inventory on vendor_id (FK)
 - product_id is a FK for vendor_inventory (*so elsewhere in this diagram, we'd connect this to a PK of another table*)
 - market_date is the PK for vendor_inventory
 - why?

Farmers Market Physical Model (partial)

```
classDiagram
    class vendor_inventory {
        market_date PK
        quantity
        date decimal(16,2)
        vendor_id FK
        product_id FK
        original_price
    }
    class vendor {
        vendor_id PK
        vendor_name
        vendor_type
        vendor_owner_first_name
        vendor_owner_last_name
    }
    vendor_inventory "1" -- "∞" vendor : vendor_id
```

vendor_inventory		
PK	market_date	date
FK	quantity	decimal(16,2)
FK	vendor_id	int
FK	product_id	int
	original_price	decimal(16,2)

vendor		
PK	vendor_id	int
	vendor_name	varchar(45)
	vendor_type	int
	vendor_owner_first_name	varchar(45)
	vendor_owner_last_name	varchar(45)

Data Modelling (continued)

Entity Relationship Diagrams

Attributes of an ERD: Entities & Relationships

Relationship Examples

Conceptual, Logical, Physical Models

Assignment 1: Design a Logical Model

Data Modelling

Assignment 1: Design a Logical Model

Q1) Create an example logical model for a small bookstore. 

At the minimum it should have employee, order, sale, customer, and book entities. Determine sensible column and table design based on what you know about these concepts. Keep it simple, but work out sensible relationships to keep tables reasonably sized. Include a date table. There are several tools online you can do this, I'd recommend [Lucidchart](#).

Q2) We want to create employee shifts, splitting up the day into morning and evening. Add this to the ERD.

Q3) The store wants to keep customer addresses. Propose two architectures for the CUSTOMERS_ADDRESS table, one that will retain changes, and another that will overwrite. Which is type 1, which is type 2? *Hint, search type 1 vs type 2 slowly changing dimensions*

Q4) Review the AdventureWorks ERD here: [link](#)

Highlight at least two differences between it and your ERD. Would you change anything in yours?

