Lecture 1 — Database Systems

Jeff Zarnett
jzarnett@uwaterloo.ca

Department of Electrical and Computer Engineering
University of Waterloo

August 11, 2023

Course Syllabus

As our first order of business, let's go over the course syllabus.

Collaborative Course

The source material for the ECE 356 notes and slides is open-sourced via Github.

If you find an error in the notes/slides, or have an improvement, go to `https://github.com/jzarnett/ece356` and open an issue.

If you know how to use `git` and LaTeX, then you can go to the URL and submit a pull request (changes) for me to look at and incorporate!

Databases

Databases and database systems are everywhere.

It is safe to say that it would not be possible for you to read this pdf (or attend the lecture in which it is used) without the involvement of databases.

They store data in an orderly fashion and allow for its retrieval at a later date.

The more traditional examples relate to everyday life for a common person: banking, airline reservations, e-commerce websites...

Database “Alternatives”

A database is by no means the only way to store data;
data may be stored in any format one wishes.

Simple text files are often used to record data.

A surprising amount of the world operates on the basis of Microsoft Excel.

This is not because of the mathematical functions or intelligence of Excel but rather the gridlines. Seriously.

Structure

For small amounts of data it may not matter very much that the data is not structured and not organized.

For larger amounts of data, this quickly becomes impractical.

Some sort of structure is needed and the first approach to dealing with this was to use files.

Why Not Files?

- **Data Redundancy**
- **Data Inconsistency**
- **Data Isolation**
- **Integrity Problems**
- **Atomicity Problems**
- **Concurrent-Access Problems**
- **Security Problems**

The DBMS

A database management system (DBMS) is a solution to these problems.

Of course, a database, if poorly designed, will not fix all of those problems.

There are three levels of abstraction we are interested in when discussing a database, from least abstraction to most:

1. Physical Level
2. Logical Level
3. View Level

Tip Your Server

A database is typically accessed via a database server.

Thus we see the client-server model in use: an application program then accesses the database through some request interface and receives answers.

We can also issue statements that change the data directly.

An incorrect statement can do a lot of damage and lead, perhaps, to unrecoverable data loss.

Schemas & Instances

A database schema defines the structure of the database.

This includes what data is to be stored in what format.

An instance of a database reflects both the schema and the particular content of that database.

Sample Logical Schema

VEHICLE

VIN	`string(17) not null, primary key`
year	`integer not null`
make	`string(64) not null`
model	`string(64) not null`
license_plate_number	`string(8)`

LICENSE_PLATE

number	`string(8) not null, primary key`
expiry	`date`
owner_address_id	`integer`

OWNER_ADDRESS

id	`integer not null, primary key`
name	`string(64) not null`
street	`string(64) not null`
city	`string(32) not null`
province	`string(2) not null`
postal_code	`string(7) not null`

Meta-Data in a Schema

There are also some items of meta-data (data about data) in the shown schema.

`id` is shown as “primary key”, which provides additional information and additional restrictions.

There are also indexes that are used in searches.

They can be added, removed, modified without affecting the underlying data.

Physical Schema

No information is shown here about how the physical schema looks.

How is this data stored in a practical sense?

From the logical view it is irrelevant.

Creating an Instance

If we create a new database and apply this schema, we created a new instance but it is empty.

This is the initial state of the database and each modification to the database creates a new state of the database.

At any time, we can take a snapshot of the current database (export all its data).

Making Changes

The most common type of modification would be to manipulate data in it.
Add, modify, delete...

This kind of change does not modify the schema, but it does modify the data.

If we attempt to modify the data in a way that is not consistent with the schema (e.g., try to put “ABC” in an integer field), that modification will be rejected.

An update can overwrite or permanently delete important data!

Careful schema design, as we will see, will prevent that sort of error, such as introducing a requirement that VINs be unique.

If that rule exists, the update is rejected.

Design cannot prevent all such errors, unfortunately.

There can be two people with the name “Thomas Anderson”[1], so no rule of unique names can be introduced.

An erroneous update command could set all names to exactly that even if the schema is designed perfectly.

[1] Whoah.

Schema Changes

Textbook authors sometimes tell you that schema changes will be rare, but much depends on your application.

If it is a stable application then schema changes are rare.

If it is a startup company or an app with ever-expanding functionality then the the schema will change regularly.

Schema changes are design changes.

Schema Changes

Modifications to the schema can be made in a way that does not affect the data or require changes to applications that access it.

If a new field “country” is added to the address entity, then something that looks only for the name will not be affected.

Similarly, a new entity, unrelated to these, such as “Service Ontario locations” can be created with no impact on the existing data.

Changer Danger

Modifying a schema, can, of course, be dangerous.

If we decide to remove a data element, such as the province from the address, then data may be lost.

If we add another property, such as a country, by default it contains nothing.

If this property is to be defined as “not null” then a default value is needed.

Multiple Instances and Schemas

The database server can, of course, have multiple instances, and multiple schemas, all of which are independent of one another.

Two databases may use the same schema but have completely different data.

Two databases can begin with the same schema but evolve in different directions over time based on the changes that are made.

A change in the schema may result in changes to some data, but may not.

Relationship Goals

The entities themselves are only a part of the picture.

There are also the relationships between these, which are partially reflected in the fields like `owner_address_id`.

This takes us to our next subject, which is to examine the relational model.