Introduction to Database Systems

What's in a Database?

Stéphane Bressan

Contact

Stéphane Bressan

COM1-03-20

6516 3543

steph@nus.edu.sg

Lee Mong Li

COM1-03-19

6516 2905

leeml@comp.nus.edu.sg

Final assessment (60%)

The final assessment is a two hours individual test. It covers all topics. It is a closed book assessment, although one double-sided A4 page of hand-written or typed notes is allowed. Check with Registrar Office for the official date, time and location of the final assessment.

Continuous assessment (40%)

Midterm assessment

The midterm assessment is a one hour individual test. It is a closed book assessment, although one double-sided A4 page of hand-written or typed notes is allowed. The midterm assessment takes place during lecture time in Week 7. The location will be indicated in IVLE lesson plan.

Project

The objective of the project is to apply the concepts and techniques learned in class for the design and programming of a Web database application. The deliverables are a pre-alpha demonstration, an alpha demonstration, a brief report and demonstration of your application. Team of 4 students (no constraint on tutorial membership or option).

Homework

You will receive instruction for your homework.

We want to develop a sales analysis application for our online gaming store. We would like to store several items of information about our customers: their first name, last name, date of birth, e-mail, date and country of registration on our online sales service and the username that they have chosen . We also want to manage the list of our products, the games, their version and price. The price is fixed for each version of each game. Finally, our customers buy and download games. So we must remember which version of which game each client has downloaded. It is not important to keep the download date for this application.

Let us create a table to store customer information. Let us call this table "`customers`". Each record in the table represents a customer. Each record contains a field to record the customer's first name, "`first_name`", a field for saving the last name of the client, "`last_name`", a field for saving the customer's email, "`email`", a field for saving the customer's date of birth, "`dob`", a field to record the customer's registration date, "`since`", a field to record the customer's identifier, "`customerid`" and finally a field to register the country of registration of the customer, "`country`".

1. Go to `www.sqlite.org`
2. Go to `www.sqlite.org/download.html`
3. Download the command-line shell for accessing and modifying SQLite databases ("`A bundle of …`")
4. Extract the executable
5. You will find a short documentation at www.sqlite.org/sqlite.html

```
›  .open cs2102.db
›  .mode column
›  .headers on
›  .help
›  ...
›  ...

›  .quit
```

You may skip the `".open cs2102.db`“ for now. That step creates a persistent database but also may slow down some update operations.

The SQL "CREATE TABLE" command creates the table. The name of the table and the name and domain (type) of each field must be specified: string of variable length, "VARCHAR ()" and date, "DATE".

```
CREATE TABLE customers (
     first_name VARCHAR(64),
     last_name VARCHAR(64) ,
     email VARCHAR(64),
     dob DATE,
     since DATE,
     customerid VARCHAR(16) ,
     country VARCHAR(16));
```

Let us create a table to save the information about the games. Let us call this table "`games`". Each record in this table represents a version of a game. It contains a field to save the name of the game, "`name`", a field to save the version of the game, "`version`" and a field to record the sale price of the game, "`price`".

We see two new domain examples: fixed character strings "`CHAR ()`" and "`NUMERIC`" numbers. There are many others.

```
CREATE TABLE games (
     name VARCHAR(32),
     version CHAR(3),
     price NUMERIC );
```

Let us create a table to save information about downloads. Let us call this table "`downloads`". Each record in this table represents the download of a version of a game by a customer. It contains a field to record the customer ID, "`customerid`", a field to save the name of the game, "name" and a field to save the version of the game, "`version`".

Note that these three fields have the same names and domains as the corresponding fields in the "`games`" and "`customers`" tables.

```
CREATE TABLE downloads (
     customerid VARCHAR(16),
     name VARCHAR(32),
     version CHAR(3));
```

We can now start inserting data. Let us create a record for the customer Carole Yoga. Carole was born on August 1, 1989. She registered with our online service on September 15, 2016 from France with the identifier "`Carole89`".

```
INSERT INTO customers VALUES(
     'Carole',
     'Yoga',
     'cyoga@glarge.org',
     '1989-08-01',
     '2016-09-15',
     'Carole89',
     'France');
```

The files below contain the SQL commands for creating the "`customers`", "`games`" and "`downloads`" tables as well as the data insertion commands, respectively.

`customers.sql`

`games.sql`

`downloads.sql`

```
CREATE TABLE downloads(
        customerid VARCHAR(16),
        name VARCHAR(32),
        version CHAR(3));

INSERT INTO downloads VALUES ('Adam1983', 'Biodex', '1.0');
INSERT INTO downloads VALUES ('Adam1983', 'Domainer', '2.1');
```

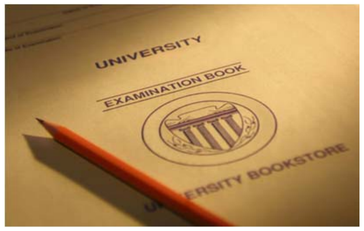
Excerpt from the file `downloads.sql`

```
› DROP TABLE downloads;
› DROP TABLE customers;
› DROP TABLE games;
› .read customers.sql
› .read games.sql
› .read downloads.sql
› INSERT INTO customers VALUES(
› 'Carole', 'Yoga', 'cyoga@glarge.org', '1989-08-01',
› '2016-09-15', 'Carole89', 'France');
›
```

A simple SQL query must include the "`SELECT`" clause, which specifies the fields to be printed, the "`FROM`" clause, which indicates the tables to be queried, and possibly the "`WHERE`" clause, which contains a possible condition on the records to be considered.
For example, the following query displays the first and last names of registered customers from Singapore.

```
SELECT first_name, last_name
FROM customers
WHERE country = 'Singapore';
```

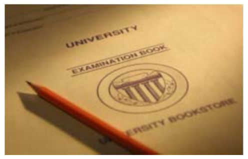
Why are we here?

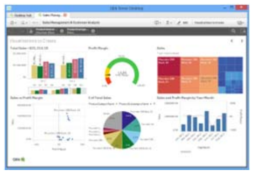


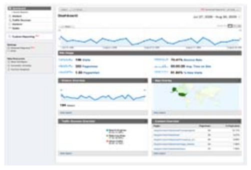
Can you give examples of applications typically requiring a database?

The management of the accounts of a retail bank,
Ticket bookings of one or more airlines,
The management of a university,
The management of an online store,
Managing my address book ...

What operations do these applications perform on the database?
What properties these applications require from the database?







There are two main categories of database applications: transactional applications (we talk about online transaction processing or OLTP) and analytical applications (we talk about online analytical processing or OLAP) . The term "database" is typically used for transactional applications and the expression "data warehouse" is reserved for analytical applications. The difference is mainly due to the respective specifications and limitations of the technology. It is difficult to organize data access so that both complex updates and queries are fast. Great efforts are made by researchers and manufacturers to invent and develop technologies that allow the convergence of both types of applications.

The life cycle of a database includes the design of the database, updating the database, querying the database and administering it.

The design of the database corresponds to the identification of the fields, the definition of the schema and the creation of the tables.
Updating the database includes adding, deleting, and editing records.
The querying of the database consists of the submission of queries which are questions of the type "What are the first and last names of the registered customer in Singapore? ".
The administration of the database includes the management of user access rights, backups and optimization of the physical organization of the data.

We do not program database applications from scratch. We use Database Management Systems (DBMS). DBMS are generic platforms for the design, implementation and management of database applications

ORACLE

Microsoft SQL Server

IBM DB2

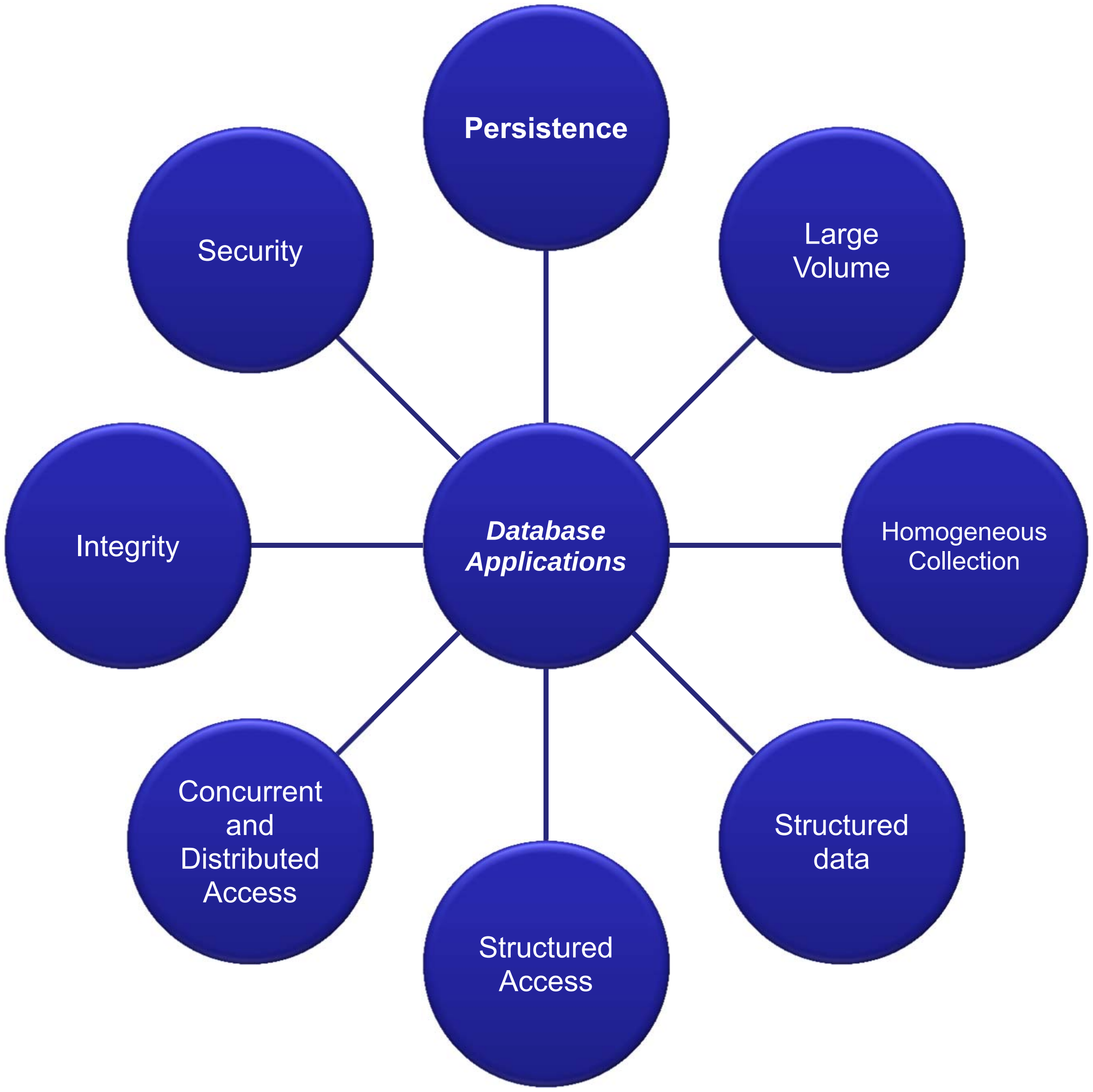
IBM

MySQL

PostgreSQL

SYBASE

SQLite

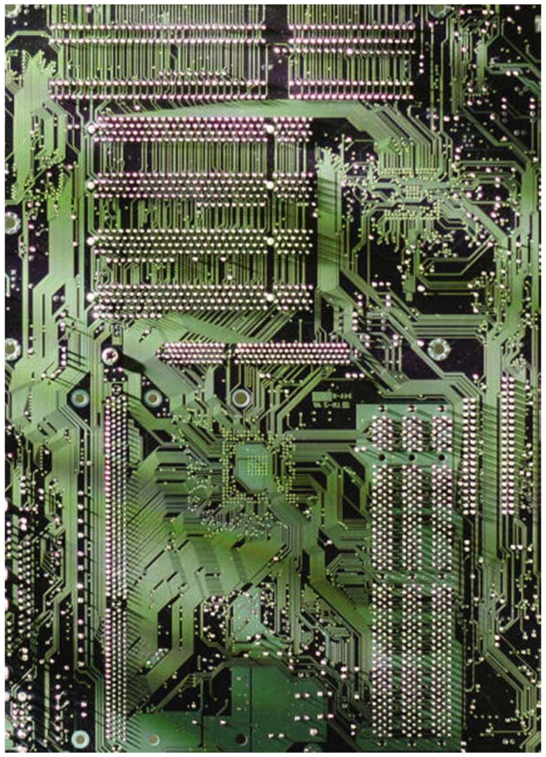


Typical properties and requirements of database applications

Data must Persist

Primary memory is volatile

Secondary and tertiary memories are persistent



Data Comes in Large Amounts

There were 190 million registered voters in the 2014 Indonesian elections

Where could one store the names, identification numbers, and electoral districts of voters?

Data Comes in Large Amounts

There were 190 million registered voters in the 2014 Indonesian elections

How could one sort them by alphabetical order of electoral districts and names?

Data Comes in Large Amounts

When data is to be stored on secondary or tertiary storage, then we need to devise efficient algorithms taking into account the dominant cost of Input/Output operations (I/Os)

Such algorithms are called external algorithms (e.g., external sort)

Sorting Made Simple

```
SELECT *
FROM voters
ORDER BY district, name;
```

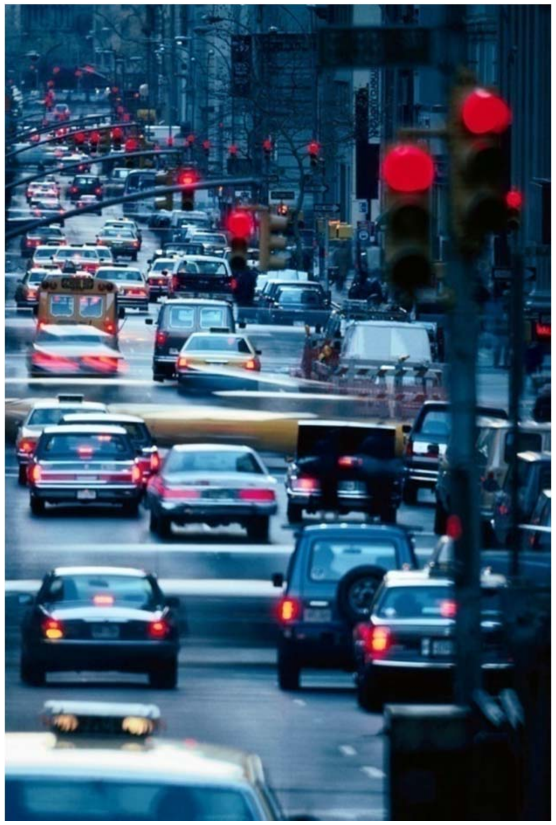
Data Comes in Large Amounts

There were 190 million registered voters in the 2014 Indonesian elections

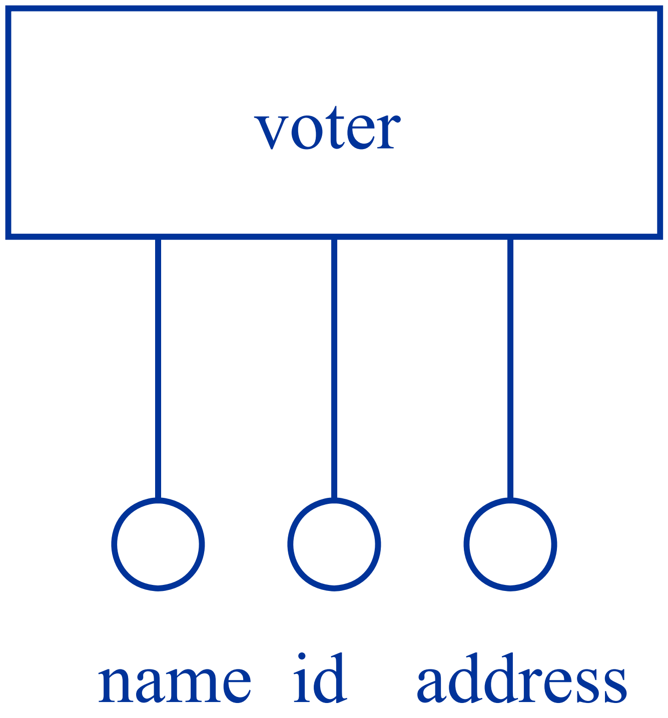
Imagine the original tapes contain duplicate entries

Think about an algorithm to remove the duplicate entries

Data Comes in Homogeneous Collections



Data is Structured



Everything is in a table

```
CREATE TABLE voters
  (first_name char(32),
  last_name CHAR(32),
        district CHAR(64),
        national_id NUMBER);

SELECT last_name
FROM voters
WHERE first_name = ‘Bambang’;
```

SQL

SQL DDL: Data Definition Language. It include statements to create, alter and drop definitions of tables, constraints, stored procedures, triggers, etc.

SQL DML: Data Manipulation Language. It include statements to insert, update, delete and query data.

SQL and the Relational Model

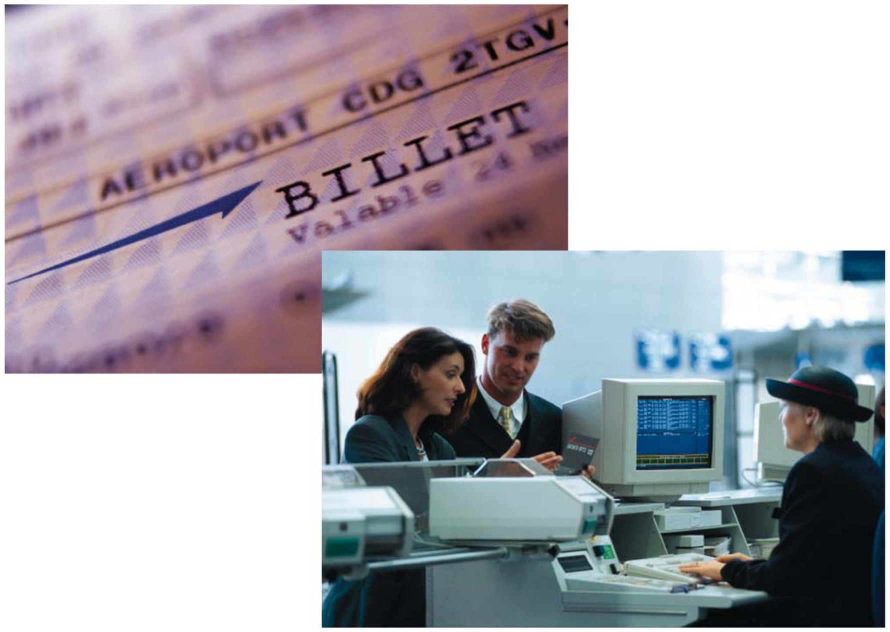
SQL is not meant to be Turing Complete.

SQL is not meant to be a programming language.

SQL is meant to be query language.

Transactions

A transaction is a logical unit of work carried out by a user or an application



Consistent States

A consistent state of the database is a state which complies with the business rules as usually defined by integrity constraints

“students who have not passed cs2102 cannot take cs3223”

If the integrity constraints are violated by a transaction, the transaction is aborted and rolled back, otherwise, it is committed.

Distributed and Concurrent Access

How can data be shared by users and processes that are possibly distributed over a network?

Integrity of Data should be Maintained

How to maintain the integrity of data in spite of possible application, system, or media failures?

ACID Properties

- Recovery
  - Atomicity: all actions in a transaction happen or none happen (transactions are committed or aborted and rolled back)
  - Durability: effects of committed transactions last
- Concurrency Control
  - Isolation: Transactions can be understood independently from each other (to be safe be serializable!)
  - Consistency: If individual transactions leave the application in a consistent state, their concurrent execution should do the same

Security and Access Control of Data is Critical

How to protect the data and define and control access to data?

Grant different access (create, read, write) to different data units (table, row) to different users

SQL

SQL DCL: Database Control Language. It include statements to administer access privileges and transactions properties

Why file systems, or even tools like Excel, can cause problems to manage data?

Persistence must be managed explicitly.

Updating and querying are difficult. Each processing requires a series of operations dependent on the organization of the data in the files which is neither necessarily homogeneous nor necessarily structured.

The organization of data for presentation is often redundant.

Concurrent and distributed access, integrity and security are difficult to manage.

Input errors, hardware errors, system errors or application errors leave the data corrupt and inconsistent.

Relational database management systems (RDBMS) provide solutions to solve or avoid these problems

Data Base Management Systems (DBMS) are platforms that facilitate the design, development, deployment and maintenance of database applications. Relational database management systems (RDBMS) are platforms that provide a database (relational) data model.

The basic data management systems are to the files what a car's automatic driving is to manual driving.

We study:

Relational Database Design

- Design with the entity-relationship model
- Design with SQL and creation of tables with integrity constraints
- Normal Forms and Normalization with Functional Dependencies

Creation, Update, Deletion and Querying of Relational Databases

- Simple and Complex SQL Statements and Queries
- Stored Procedures and Triggers
- Theory of Query Languages: Relational Domain and T-uple Calculus and Relational Algebra

Some bibliographical and Web references

"SQL Tutorial" www.w3schools.com/SQL

"Introduction to Databases" by Jennifer Widom, www.youtube.com or online.stanford.edu

"Database Management Systems" by R. Ramakrishnan et J. Gehrke , McGraw Hill

Introduction to Database Systems " by S. Bressan and B. Catania, McGraw Hill

"An Introduction to Database Systems (8th edition)" by J.C. Date, Pearson

Credits

The content of this lecture is based on chapter 1 of the book "Introduction to database Systems" By S. Bressan and B. Catania, McGraw Hill publisher

Images and clips used in this presentation are licensed from Microsoft Office Online Clipart and Media

For questions about the content of this course and about copyrights, please contact Stéphane Bressan

steph@nus.edu.sg