Software System Components Notes

School of Computer Science Students

October 5, 2015

Preface

This is a shared collection of notes for Software System Components. Please visit https://www.github.com/UoB-CS-Students/Year-2-SSC to find out more, and to see other modules. You can contribute to this document by:

- Editing the LaTeX document you wish to contribute to, then submit a pull request to https://www.github.com/UoB-CS-Students/Year-2-SSC.
- Creating a new chapter, placing the LaTeX file in the `tex` folder, and adding a line to `Notes.tex` such as `\input{./tex/MY_CHAPTER.tex}`

Here are some points to follow:

- For the purposes of version control, please try to put each sentence on a new line. (LaTeX treats a single new line as a space, so inserting these extra spaces won't affect the display of your document).
- Place any package imports in `Notes.sty`.
- If you wish to contribute, try to make fairly small changes, and then submit a pull request.
- Use hyphens instead of spaces in your file names, e.g. `My-File.tex` instead of `My File.tex`
- Follow the current naming convention for files/chapters. For example, if the current file names are `1-Alpha`, `2-Beta`, then you should name your file `n-FILENAME`.

Contents

Chapter 1

Introduction to Databases

1.1 How can we store (and communicate) data?

Store	**Share**
• In memory (persistently?) • Java objects • Flat/structured data files: – XML – Interchange formats	• Shared memory • Share files locally • Share files remotely • Document repository • Data server—fetch data as required.

Chapter 2

DBMS

2.1 What is a DBMS?

A DBMS consists of a core database which provides:

1. Representation of data
2. Retrieval and maintenance of data

It also has a **collection of tools**:

- Design and build and maintenance
- E.g. standard applications

And it provides:

- An abstraction from implementation
- Efficient implementation
- Integrity, fault tolerance, security, . . .
- Standard (and bespoke) interfaces, e.g. SQL for querying a database

2.2 Relational databases

Relational databases are the pre-eminent choice for *general purpose* database systems, especially for commercial/enterprise systems since consistency, reliability etc. are critical. Here are some points about relational databases:

- Main model for database systems
- Data in form of sets and relations

- Data connected using basic set theory (selecting, combining, etc.)
- Normally viewed as **tables**
- Queries specify result, but now how it is computed, i.e. it is *declarative*

2.3 Database Models

We don't always need all the benefits of a relational database; we may not even need to write to or update our database. But we may need performance, scaleability, and flexibility.

2.4 What makes a good database?

- It should be an *organised* collection of data
  - For a purpose
  - To facilitate some set of activities
  - Organised so that it can be accessed and maintained efficiently.
- Data independence from internal or physical representation.
- Minimise redundancy (only store data once)
- Maximise consistency (one underlying representation)
- Enable integration and sharing
- Facilitate change
- Logical organisation-

2.5 ACID

Atomicity a transaction happens as a whole, or not at all.

Consistency a database is always in a consistent state (according to the rules defined)

Isolation the effect of two operations happening in parallel is the same as if they had happened sequentially

Durability once something is stored it won't disappear

It should do all this even if the plug is unpulled.

2.6 ANSI-SPARC Architecture

ANSI-SPARC is an abstract design standard for a DBMS. It has three different levels:

External level (user views) a user's view of the database describes a part of the database that is relevant to a particular user.

Conceptual level is a way of describing what data is stored within the whole database and how the data is interrelated.

Internal level involves how the database is physically represented on the computer system. It describes how the data is actually stored in the database on the computer hardware.

2.7 Database applications

Users are usually shielded from the underlying database by application programs and web interfaces; this is for convenience and security.

2.8 Data definition language (DDL)

A data definition language is used to create, modify, and delete parts of the definition of the database.
Here is a sample of SQL code that would create a table called `student` with the fields `sid`, `dob`, `login`, and `course`.

```
CREATE TABLE Student (
        sid     INTEGER
        dob     CHAR(10)
        login   CHAR(20)
        course  CHAR(10)
)
```

2.9 Data manipulation language

A data manipulation language is used to manipulate the data.
Here is a sample of SQL code that would select the `sid` and `login` columns from the `Student` table where the course name is `Se`:

```
SELECT  sid,login
FROM    Student
WHERE   course=‘Se’
```

We can impose constraints on the data. We can define these when we create the table, or we can add them later. Here is an example of imposing restraints:

```
CREATE TABLE Student (
    sid        INTEGER  NOT NULL UNIQUE,
    dob        CHAR(10),
    login      CHAR(20) UNIQUE,
    course     CHAR(10)
)
```

2.10 Relational definitions

Domain an arbitrary (non-empty) set of atomic values.

Attribute name a symbol with an associated domain, dom(A).

Relational schema a finite set of attribute names.

Tuple, t, of a relational schema R a mapping from attributes A of a relational schema R to the union of their domains dom(A) : $t(A) \in$ dom(A)

Relation, r, of a relational schema R a finite set of tuples of relational schema R

Degree (Arity) of a relational schema the number of attributes

Cardinality of a relation the number of tuples

2.11 Keys

Superkey a set of attributes that can *always* be used to differentiate one tuple from another (within a relation).

Key a minimal superkey.

Concatenated key a key with more than one attribute.

Candidate key any key.

Primary key one of the candidate keys.

Foreign key an attribute of the relation which is the key for another relation.

Chapter 3

DML

3.1 Drop and alter

3.1.1 Drop

DROP deletes the table.

```
DROP TABLE Student
```

3.1.2 Alter

ALTER modifies the table definition.

```
ALTER TABLE Student
    ADD COLUMN year_of_study INTEGER
```

3.2 Types

BOOLEAN TRUE, FALSE, or NULL

CHAR(size) or CHARACTER(size)

Strings Several versions

INTEGER or INT Several variations

FLOAT or DOUBLE

REAL, NUMERIC, or DECIMAL

DATE

TIME a time-of-day value