

EDA216 Database Technology

Lecture 1

Christian.Soderberg@cs.lth.se

January 16, 2017



Christian.Soderberg@cs.lth.se

1 / 29

Today's Lecture

- ▶ Short introduction to the course itself
- ▶ Some background on databases
- ▶ Introduction to relations and SQL

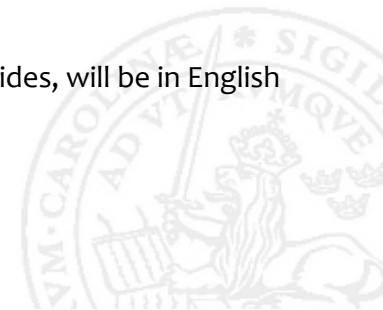


Christian.Soderberg@cs.lth.se

3 / 29

Administration

- ▶ From the formal course description (see web site):
 "Language of instruction: *The course will be given in Swedish*"
- ▶ So, I'll lecture in Swedish...
- ▶ ...but the written material, including these slides, will be in English

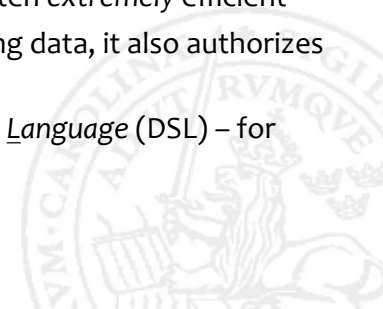


Christian.Soderberg@cs.lth.se

2 / 29

Database Management Systems

- ▶ A *Database* is just an organized collection of data
- ▶ A *Database Management System* (DBMS) is some software which allows us to separate the code for managing data from the rest of our code
- ▶ A DBMS typically runs as a server, and it's often extremely efficient
- ▶ The server takes care of storing and retrieving data, it also authorizes access to the data
- ▶ We talk to the server using a *Domain Specific Language* (DSL) – for many DBMS's, that language is SQL



Christian.Soderberg@cs.lth.se

5 / 29

Historical development

- ▶ In the 60ies, several ideas for DBMS were tried, many of them were based on linking data
- ▶ In 1970, Edgar F. Codd invented the relational model, which has been tremendously successful, and will be the basis of this course – SQL is based on his relational model
- ▶ In the 80ies several attempts were made to create 'object-oriented databases', but without much success
- ▶ In the early 2000s, a new category of databases emerged, with *key-value stores* and *document-orientation* – the category became known as NoSQL, but the 'No'-part is often thought of as 'Not Only'
- ▶ The latest fad is what's become known as NeoSQL
- ▶ As we turn into 2017, SQL databases dominate the market

Christian.Soderberg@cs.lth.se

6 / 29

About the course

- ▶ EDA216 – Database Technology
- ▶ Credits: 7.5 hp
- ▶ Level: G2
- ▶ Required for: C2
- ▶ Elective for: BME4, D4, E4, F4, I4, L4, and π 4

Christian.Soderberg@cs.lth.se

8 / 29

SQLite

- ▶ Most widely used DBMS, such as Oracle, MySQL, SQL Server, PostgreSQL, MongoDB, Cassandra, and MariaDB, run as servers
- ▶ One notable exception is SQLite, which is normally linked into our programs instead
- ▶ We're going to use SQLite in this course, since it's very easy to set up, and still implements most of the SQL standard
- ▶ SQLite is probably the world's most used database – it's linked into programs such as Chrome, Opera, Safari, Firefox, Skype, and it's also used in many, many mobile apps

Christian.Soderberg@cs.lth.se

7 / 29

Course Aim

The course gives basic theoretical and practical knowledge about database systems and their organisation. The emphasis is on relational databases.

Christian.Soderberg@cs.lth.se

9 / 29

Learning Outcomes: Knowledge and understanding

For a passing grade the student must

- ▶ be able to describe information systems with E/R models and UML notation, and translate such models into relational form
- ▶ be able to normalise database schemas
- ▶ be able to use the query language SQL to create and update a database, and to retrieve information from the database
- ▶ know about alternative ways to organise data in databases and about the design of database management systems

Christian.Soderberg@cs.lth.se

10 / 29

Course Contents

- ▶ Introduction to database systems. Basics of the relational model, the query language SQL. Methods for data modelling and database design, E/R diagrams and UML diagrams. Theory for the relational model: functional dependencies, normalisation, relational algebra. Stored procedures, triggers. Program and web interfaces to databases.
- ▶ Other data models: object-oriented databases, NoSQL-databases, semistructured data (XML).
- ▶ Security and integrity in databases, concurrency, transactions. Implementation of database management systems and query languages.

Christian.Soderberg@cs.lth.se

12 / 29

Learning Outcomes: Competence and skills

For a passing grade the student must

- ▶ be able to use tools to implement a database
- ▶ be able to develop program and web interfaces to databases

Christian.Soderberg@cs.lth.se

11 / 29

Problem

Define data structures to keep track of contacts with phone numbers and email addresses

Christian.Soderberg@cs.lth.se

17 / 29

The Relational Model

- ▶ Everything is represented as *tuples*
- ▶ Each tuple is a row in a *table*
- ▶ Each attribute of the tuples is its own *column* in the table

contacts

name	phone	email
Adam	650-043-1797	adam@life.edu
Emma	347-326-4813	emma@mail.org
Christian	347-326-3154	cs@gmail.com

The Relational Model

contacts

name	phone	email
Adam	650-043-1797	adam@life.edu
Emma	347-326-4813	emma@mail.org
Christian	347-326-3154	cs@gmail.com

- ▶ All values are atomic (i.e., not compound – strings, dates, and timestamps are regarded as being atomic)
- ▶ We have no explicit objects or hierarchies

SQL Queries

- ▶ SQL is short for Structured Query Language, and it's been around since the 70ies
- ▶ It is used to define databases, and to query and update them
- ▶ Asking for the phone number and email of Adam:

```
SELECT phone, email
FROM   contacts
WHERE  name = 'Adam'
```

Example

Try the first notebook!

Questions

contacts

name	phone	email
Adam	650-043-1797	adam@life.edu
Emma	347-326-4813	emma@mail.org
Christian	347-326-3154	cs@gmail.com

- ▶ What do we do if someone hasn't got a phone number?
- ▶ What do we do if someone has more than one email address?

Handling more complex data

- ▶ We can use NULL to denote a missing value
- ▶ We normally avoid having several values in one column (like a string with concatenated email addresses), or several columns with room for extra values – instead we have more tables!
- ▶ It's often useful to have a simple unique id for each row in a table

Using more tables

contacts

name	phone
Adam	NULL
Emma	347-326-4813
Christian	347-326-3154

email_addresses

name	email
Adam	adam@life.edu
Emma	emma@mail.org
Christian	cs@gmail.com
Christian	cso@lth.se

- ▶ Observe that there are no explicit links from contacts to email_addresses, as we probably would have had if we declared a corresponding Java class
- ▶ We would still have problems if someone changed names

Using even more tables

contacts

id	name
101	Adam
102	Emma
103	Christian

email_addresses

id	email
101	adam@life.edu
102	emma@mail.org
103	cs@gmail.com
103	cso@lth.se

phone_numbers

id	phone
102	347-326-4813
103	347-326-3154

- ▶ This also makes it easier to handle several people with the same name (the 'id' number can serve as a key)

Overdoing it

contacts

id
101
102
103

names

id	name
101	Adam
102	Emma
103	Christian

email_addresses

id	email
101	adam@life.edu
102	emma@mail.org
103	cs@gmail.com
103	cso@lth.se

phone_numbers

id	phone
102	347-326-4813
103	347-326-3154

- ▶ This is just too many tables – in a few weeks time we'll learn how to find the sweet spot

Terminology

- ▶ *table*, or *relation*: keeps rows of data, where each row contains a tuple describing something
- ▶ *column*, or *attribute*: describes a property which all our values has (or could have)
- ▶ *row*, or *tuple*: contains all properties of a given value
- ▶ *projection*: selection of some columns from zero or more rows
- ▶ *selection*: selection of zero or more rows
- ▶ *arity*: the number of columns/attributes
- ▶ *cardinality*: the number of rows/tuples