Introduction to Database Design

KBL chapters 1-3

Rasmus Pagh

Today's lecture

- Who are we? Why are we here?
- Overview of intended learning outcomes.
  - Introduction to the relational model.
  - Brief SQL primer.

Who am I?

- Rasmus Pagh, associate professor.
  - Office 4D26, e-mail: lastname@itu.dk
- PhD from Aarhus University, 2002. Topic: Hashing.
- Worked at ITU since then
  - Research in algorithms and data structures, databases (indexing), and data mining.
  - Previous teaching in databases and algorithms.

Who are you?

- About 70% BSWU, 30% SDT.
- We are too many for a presentation round…
- **Instead**: During exercises, please spend a few minutes to talk to at least one person that you did not know before (good to do this in small groups).

Teaching

- Lectures, 8.00-9.50
  - No preparation expected
  - Problem sessions
- Exercises, 10-12
  - In 2A14 and nearby, computers in 3A50
  - Current week, no preparation
  - Previous week, homework
- Project work – 4 mandatory hand-ins (more info next week)

- You must check the course pages in LearnIT and itu.dk/people/pagh/idb11/ regularly for news and resources.

Terminology 1: Database

- "a usually large collection of data organized especially for rapid search and retrieval (as by a computer)" - from m-w.com
- "a collection of data items related to some enterprise" – KBL

- Other more involved queries than just search and retrieval

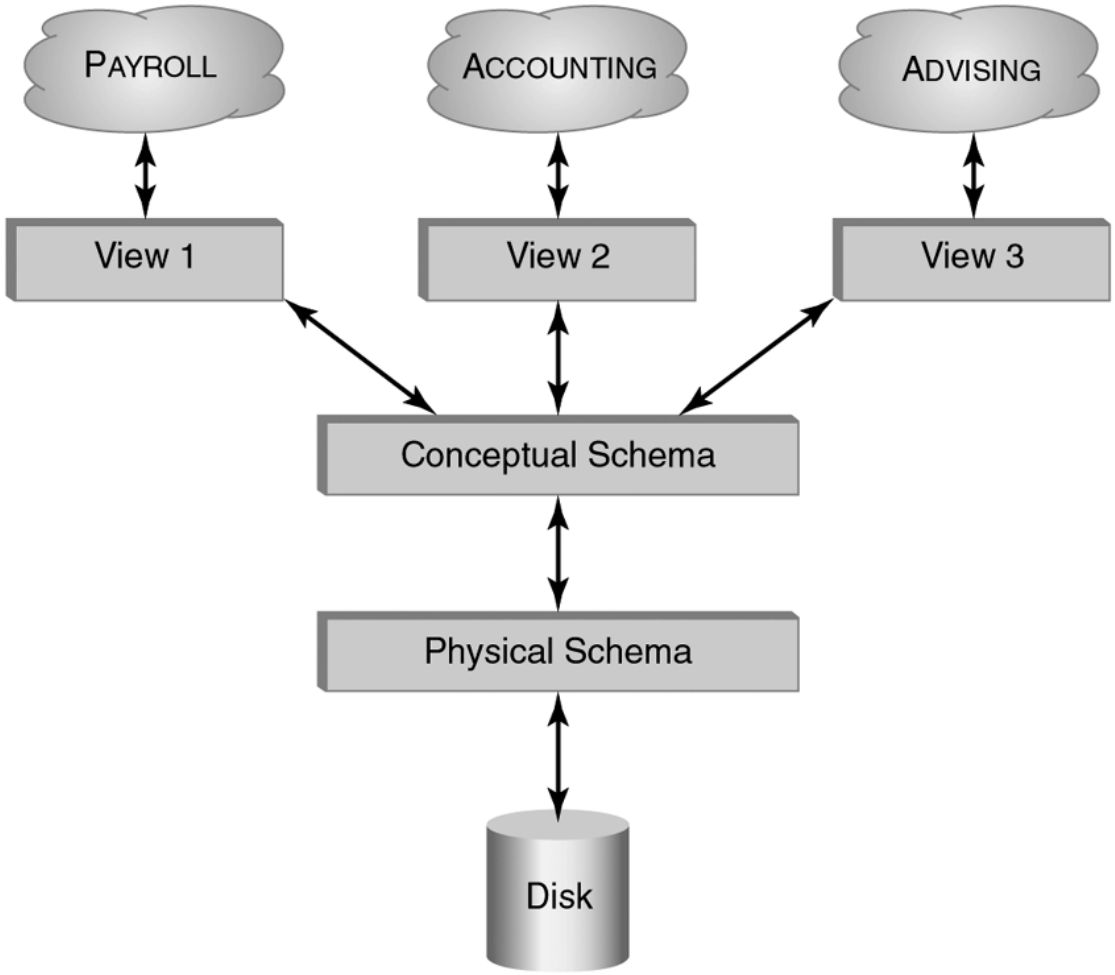
Terminology 2: DBMS

- DataBase Management System
- Software system used when implementing databases.
- Provides *efficient*, *convenient*, and *safe* storage of, and *multiuser* access to (possible *massive*) amounts of *persistent* data.
- Supports a high-level language for access (queries and updates) to the data.

Motivation

ORACLE		Ticker Symbol:	ORCL
Shares Outstanding:	5,029,523,000	**Market Value:**	$ 112,761,905,660

Levels of data independence



Data independence example

Idea: Use Excel for course planning.

First try: One view.

Line	Course	Semester	Group	Teacher	Contribution	ECTS
BSWU	BDLF	E2010	EC	Rasmus Pagh	100%	7.5
SDT	FOC	E2010	EC	Rasmus Pagh	50%	15
SDT	FOC	E2010	PLS	Marco Carbone	50%	15
BSWU	IDB	E2011	EC	Rasmus Pagh	100%	7.5
SDT	DM	E2011	PLS	Carsten Schürmann	100%	7.5

Second try: Multiple views

EC group						
BSWU	BDLF	E2010	EC	Rasmus Pagh	100%	7.5
SDT	FOC	E2010	EC	Rasmus Pagh	50%	15
BSWU	IDB	E2011	EC	Rasmus Pagh	100%	7.5
Sum:					250%	30
PLS group						
SDT	FOC	E2010	PLS	Marco Carbone	50%	15
SDT	DM	E2011	PLS	Carsten Schürmann	100%	7.5
					150%	22.5

SDT study line						
SDT	FOC	E2010	EC	Rasmus Pagh	50%	15
SDT	FOC	E2010	PLS	Marco Carbone	50%	15
SDT	DM	E2011	PLS	Carsten Schürmann	100%	7.5
					200%	37.5
BSWU study line						
BSWU	BDLF	E2010	EC	Rasmus Pagh	100%	7.5
BSWU	IDB	E2011	EC	Rasmus Pagh	100%	7.5
					200%	15

Were are trying to use a data model (spreadsheet) that does not separate logical organization from views on data

The result is a clash of conflicting goals

Relational databases

E. F. Codd, 1970

- In relational databases data is *logically* stored in tables (aka. relations).
- A table is a **set** of rows (aka. tuples).
- Columns (aka. attributes) have a name and a type.

Id	Name	Address	Status
111111111	John Doe	123 Main St.	Freshman
666666666	Joseph Public	666 Hollow Rd.	Sophomore
111223344	Mary Smith	1 Lake St.	Freshman
987654321	Bart Simpson	Fox 5 TV	Senior
023456789	Homer Simpson	Fox 5 TV	Senior
123454321	Joe Blow	6 Yard Ct.	Junior

FIGURE 2.1 The table STUDENT. Each row describes a single student.

Relational databases, cont.

- *Conceptual* difference from tables in C, Java, …: There is **no order** of tuples and attributes.
- In a pure relational model, values are **atomic** (think primitive type).
- **In modern relational DBMSs** (object relational): Values can be objects.

Terminology:

Relation schema: description of the columns (names, types) of a relation

Relation instance: a relation with a *specific* set of rows and named columns

Relation schema example

```
CREATE TABLE CAR (
  Regnr VARCHAR(8),
  Ownerid INTEGER,
  Color VARCHAR(15))
```

Course goal

After the course the students should be able to:

- suggest a database design according to the relational model, and present it as an SQL schema, using the concepts key, type, and constraint.

Problem session

(In small groups, about 5 minutes)

Discuss and suggest ways to represent a teaching plan using one or more relations.

As we saw, using a single relation is *not* a good idea:

Line	Course	Semester	Group	Teacher	Contribution	ECTS

Can you avoid (or reduce) duplication of information?

Normalization

Redundant information is a problem:

- Extra storage
- Hard to update

Normalization theory helps to refine the design to get a more efficient way to organize data in relations.

Course goal

After the course the students should be able to:

- find functional dependencies in a relation and perform decomposition to eliminate unwanted dependencies.

E-R Modeling

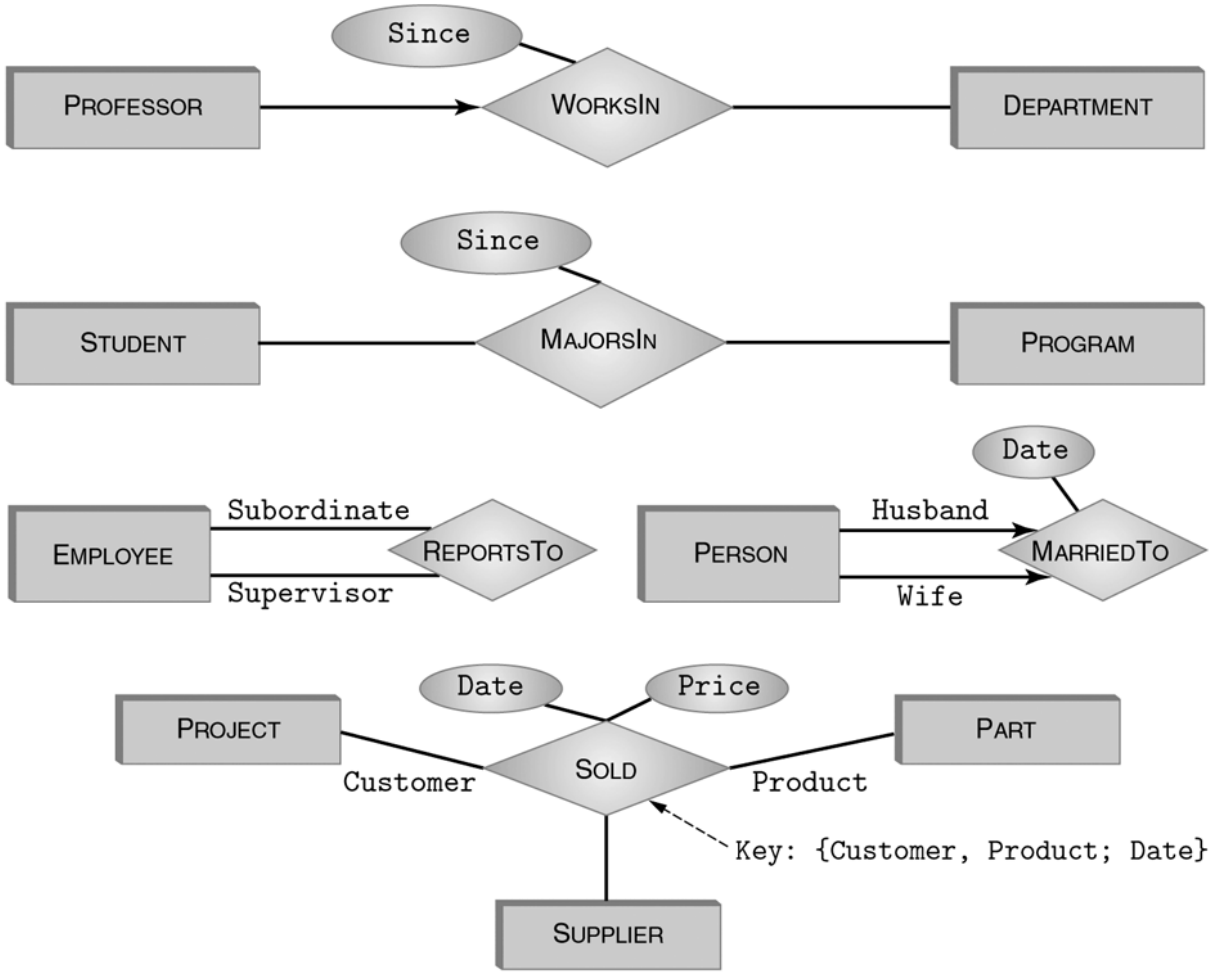


FIGURE 4.2 E-R diagrams for several relationship types.

Course goal

After the course the students should be able to:

- define a database design by E-R modeling, using the concepts entity, attribute, key, cardinality, and relationship

SQL

- The most important programming language for databases
- Structured Query Language ("sequel")
- **Declarative**: specify *what* you want, not *how* to get it
- SQL queries takes one or more tables as arguments and produces a table as a result
- Not only queries, also updates and schema definition

Course goal

After the course the students should be able to:

- write SQL queries, involving multiple relations, compound conditions, grouping, aggregation, and subqueries.

SELECT statement

```
SELECT Id, Name
FROM STUDENT
WHERE Status='Senior'
```

Id	Name
987654321	Bart Simpson
023456789	Homer Simpson

Id	Name	Address	Status
111111111	John Doe	123 Main St.	Freshman
666666666	Joseph Public	666 Hollow Rd.	Sophomore
111223344	Mary Smith	1 Lake St.	Freshman
987654321	Bart Simpson	Fox 5 TV	Senior
023456789	Homer Simpson	Fox 5 TV	Senior
123454321	Joe Blow	6 Yard Ct.	Junior

FIGURE 2.1 The table STUDENT. Each row describes a single student.

SELECT statement

```
SELECT Name
FROM STUDENT
WHERE Id=987654321
```

returns a table with one row and one column

Id	Name	Address	Status
111111111	John Doe	123 Main St.	Freshman
666666666	Joseph Public	666 Hollow Rd.	Sophomore
111223344	Mary Smith	1 Lake St.	Freshman
987654321	Bart Simpson	Fox 5 TV	Senior
023456789	Homer Simpson	Fox 5 TV	Senior
123454321	Joe Blow	6 Yard Ct.	Junior

FIGURE 2.1 The table STUDENT. Each row describes a single student.

SELECT statement

```
SELECT *
FROM STUDENT
WHERE Id=987654321
```

* means “all columns” - but is not a “wildcard” character

returns a table with one row and all 4 columns

Id	Name	Address	Status
111111111	John Doe	123 Main St.	Freshman
666666666	Joseph Public	666 Hollow Rd.	Sophomore
111223344	Mary Smith	1 Lake St.	Freshman
987654321	Bart Simpson	Fox 5 TV	Senior
023456789	Homer Simpson	Fox 5 TV	Senior
123454321	Joe Blow	6 Yard Ct.	Junior

FIGURE 2.1 The table STUDENT. Each row describes a single student.

SELECT statement

```
SELECT COUNT(*)
FROM STUDENT
WHERE Status='Senior'
```

returns a table with the value 2, i.e. #rows with 'Senior'
COUNT is an aggregate function

Id	Name	Address	Status
111111111	John Doe	123 Main St.	Freshman
666666666	Joseph Public	666 Hollow Rd.	Sophomore
111223344	Mary Smith	1 Lake St.	Freshman
987654321	Bart Simpson	Fox 5 TV	Senior
023456789	Homer Simpson	Fox 5 TV	Senior
123454321	Joe Blow	6 Yard Ct.	Junior

FIGURE 2.1 The table STUDENT. Each row describes a single student.

SELECT statement

```
SELECT *
FROM STUDENT
WHERE Id<66666666 AND NOT (Status='Senior')
```

returns a table with three rows

A condition in WHERE can be any Boolean expression

Id	Name	Address	Status
111111111	John Doe	123 Main St.	Freshman
666666666	Joseph Public	666 Hollow Rd.	Sophomore
111223344	Mary Smith	1 Lake St.	Freshman
987654321	Bart Simpson	Fox 5 TV	Senior
023456789	Homer Simpson	Fox 5 TV	Senior
123454321	Joe Blow	6 Yard Ct.	Junior

FIGURE 2.1 The table STUDENT. Each row describes a single student.

More general form of SELECT

Suppose:

- A_1, A_2, … are column names,
- R_1, R_2, … are tables,
- `<condition>` is a boolean expression involving columns from R_1, R_2, …

Then:

```
SELECT A1,A2,…
FROM R1,R2,…
WHERE <condition>
```

returns the subset of the *cartesian product* of R_1, R_2, … that satisfy <condition>.

Integrity constraint

- An integrity constraint is a statement about legal instances of a database
- Examples:
  - All students have unique ids (a *key*)
  - A student can't change status from senior to freshman
  - Enrolment date is before graduation date

Key constraints, definition

A key constraint **key(K)** associated with a relational schema S, consists of a subset K of attributes in S satisfying:

Uniqueness property:

No instance of S contains a pair of distinct tuples whose values agree on all the attributes in K.

SQL allows specifiction of key constraints. One key is declared to be **primary key**.

SQL key example

```
CREATE TABLE CAR (
  Regnr VARCHAR(8) NOT NULL,
  Ownerid INTEGER,
  Color VARCHAR(15),
  PRIMARY KEY (Regnr),
  UNIQUE (Ownerid, Color) )
```

Here we assume that one person can only have one car of each color.

SQL: CHECK

```
CREATE TABLE CAR (
   Regnr VARCHAR(8) NOT NULL,
   Ownerid INTEGER,
   Color VARCHAR(15),
   PRIMARY KEY (Regnr),
   CHECK (Ownerid>999 AND NOT Color='Lilac' ) )
```

Two semantic constraints that say that Ownerids always have at least 4 digits and that cars can not be lilac.

Note: CHECK is not implemented in MySQL.

Referential integrity

Referential integrity:

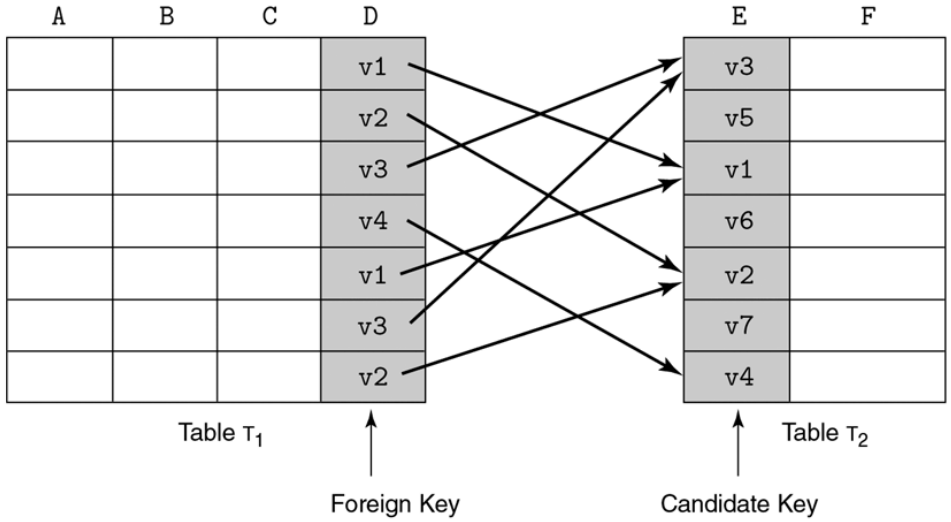
"When a tuple has a reference to another tuple, then the referenced tuple must exist."

```
STUDENT(Id:INT, Name:STRING)
Key:{Id}

TRANSCRIPT(StudId:INT, CrsCode:STRING, Grade:STRING)
Key:{StudId,CrsCode}
```

Often the referenced value is the primary key (a **foreign key**).

Foreign key constraint



All non-null values of a foreign key must exist in the referenced table. SQL syntax:

```
FOREIGN KEY StudID references STUDENT(id)
```

Maintaining integrity

What happens when a referenced tuple in STUDENT is changed or deleted?

Three options:

`SET NULL`: Set reference to NULL

`NO ACTION`: Update or delete rejected

`CASCADE`: Delete/update the reference

Problem session

Consider the following relations:

- `PERSON(Cpr,Name,Birthday)`
- `ADDRESS (Id,Street,Number,Zip,City)`
- `LIVESAT(Cpr,AddressId)`
- `PHONE(SubCpr,Number,Type,AddressId)`

What are (probably) the keys?

What are suitable primary/foreign keys?

What should happen when an address is deleted?

Course goal

After the course the students should be able to:

- express simple relational expressions using the relational algebra operators select, project, join, intersection, union, set difference, and cartesian product.

Relational algebra

The mathematical basis of SQL.

SQL expressions can be translated into relational algebra expressions and vice versa.

```
SELECT Lastname, Regnr, Color
FROM Car, Owner
WHERE Id=Ownerid AND Color='Pink'
```

$$\pi_{\text{Lastname,Regnr,Color}}(\sigma_{\text{Color}='\text{Pink}'}(Car \bowtie_{Id=Ownerid} Owner))$$

Relational algebra

SQL is declarative (what)

Relational algebra is procedural (how)

The DBMS translates SQL to relational algebra.

The query optimizer translates the expression to an equivalent expression that can be evaluated more efficiently.

Course goals

After the course the students should be able to:

- decide if a given index is likely to improve performance for a given query.

Transaction

A transaction is a sequence of operations on a database that belong together.

Useful when multiple users update the database in parallel.

Example:

Two persons with a shared bank account try to withdraw 100 kr at the same time.

Transaction:

1) read balance and store in variable B

2) if B>=100 then B:=B-100

3) write B to balance

ACID Properties

Atomicity: A transaction runs to completion or has no effect at all

Consistency: After a transaction completes, the integrity constraints are satisfied

Isolation: Transactions executed in parallel has the same effect as if they were executed sequentially

Durability: The effect of a committed transaction remains in the database even if the computer crashes.

Course goals

After the course the students should be able to:

- identify possible problems in transaction handling, related to consistency,
  atomicity, and isolation.
- apply a simple technique for avoiding deadlocks

Course goals

After the course the students should be able to:

- use SQL in applications (Java).

Course goals

- write simple XML Schemas and simple Xquery expressions.
- explain the meaning of a DTD, and the effect of simple XSLT transformations.

XML

- eXtensible Markup Language
- A format for semistructured data
- Framework for defining markup languages
- Resembles HTML, but “you decide the tags”
- XML describes any content, while HTML describes appearance

XML example

```
<dictionary>
  <entry id=31>
     <word>banana</word>
     <meaning>yellow fruit</meaning>
  </entry>
  <entry id=83>
     <word>milk</word>
     <meaning>white fluid</meaning>
  </entry>
</dictionary>
```

XML Schema

Data definition language for XML documents, i.e. describes the structure of an XML document.

Describes tag names and types, constraints and more.

DTD is another (more limited) data definition language for XML documents.

XQuery

A query language for XML.

Similar to SQL:

```
FOR variable declaration
WHERE condition
RETURN result
```

New: “Path expressions” are used to extract data from the XML document.

Next steps...

- Exercises today:
  - Databases without a DBMS.
  - Modeling exercise.
- Lecture next week:
  - E-R modeling