Programmerade system
TDA143, 2013-2014
Lecture on Databases

Graham Kemp
kemp@chalmers.se
Room 6475, EDIT Building
http://www.cse.chalmers.se/~kemp/

Material in course textbook

"Computer Science: An Overview"
9th /10th /11th Edition, J. Glenn Brookshear

Chapter 9

Why study databases?

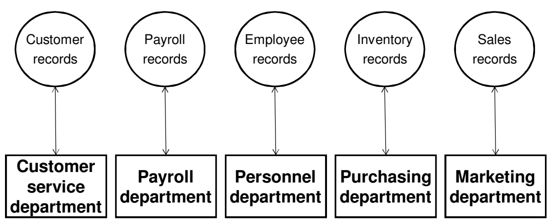
Banking, ticket reservations, customer records, sales records, product records, inventories, employee records, address books, demographic records, student records, course plans, schedules, surveys, test suites, research data, genome bank, medical records, time tables, news archives, sports results, e-commerce, user authentication systems, web forums, www.imdb.com, the world wide web, …

Databases are everywhere!

Examples

- Banking
  - Drove the development of DBMS
- Industry
  - Inventories, personnel records, sales …
  - Production Control
  - Test data
- Research
  - Sensor data
  - Geographical data
  - Laboratory information management systems
  - Biological data (e.g. genome data)

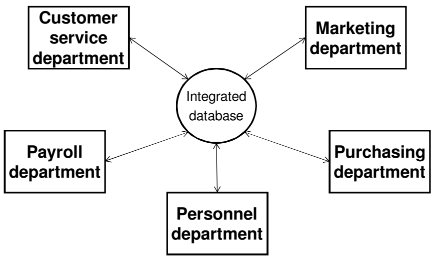
File-oriented information system



Problems with working with files

- Redundancy
  - Updates
  - Wasted space
- Changing a data format will require all application programs that read/write these files to be changed.
- Sharing information between departments can be difficult.

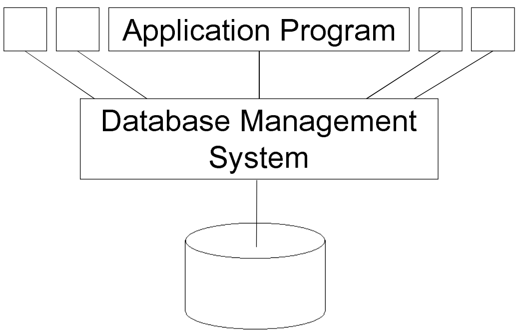
Database-oriented information system



A database is …

- a collection of data
- managed by specialised software called a **database management system** (DBMS) (or, informally, a “database system”)
- needed for large amounts of persistent, structured, reliable and shared data

Using a DBMS: an overview



Centralised control of data

- amount of redundancy can be reduced
  - less inconsistency in the stored data
- stored data can be shared
- standards can be enforced
- security restrictions can be applied
- data integrity can be maintained
  - validation done in one place
- conflicting requirements can be balanced
- provides **data independence**
  - can change storage structure without affecting applications

Motivation for database systems

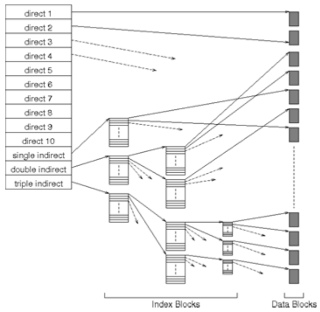
Needed for large amounts of persistent, structured, reliable and shared data (Ted Codd, 1973)

- Large amounts:
  - needs indexing for fast access
  - needs a load utility
- Persistent:
  - needs schema definition of types which evolves
- Structured:
  - storage schema held with data
  - query language (e.g. SQL) independent of storage
- Shared:
  - locking mechanism for concurrent update
  - access control via DBMS
  - centralised integrity checking
- Reliable:
  - changes to disc pages are logged
  - commit protects against program of disc crash
  - can undo (rollback) uncommitted updates

Traditional File Structures

A short digression …

UNIX file management



Actual organisation is hidden

- Just as the file management system in an operating system gives the users the illusion that a text file is stored on disc as a long consecutive sequence of characters …
- … a database management system gives the users the illusion that their data are stored on disc in accordance with a **data model**.

Data models

- Storing data in a computer system requires describing the data according to some **data model**, in a form which can be represented directly within the computer.
- A **data model** specifies the rules according to which data are structured and also the associated operations that are permitted.

Data models: brief overview

- “No data model”
  - Flat files
- “Classical” data models
  - Hierarchical (tree)
  - Network (e.g. CODASYL) (graph)
  - **Relational** (Codd, 1970) (tables)
- Semantic data models, e.g.
  - Entity-Relationship model (Chen, 1976)
  - Functional Data Model (Shipman, 1981)
  - SDM (Hammer and McLeod, 1981)

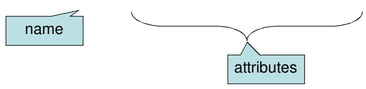
Relational DBMSs

- Very simple model
- Familiar tabular structure
- Has a good theoretical foundation from mathematics (set theory)
- Industrial strength implementations, e.g.
  - Oracle, Sybase, MySQL, PostgreSQL, Microsoft SQL Server, DB2 (IBM mainframes)
- Large user community

Relation Schemas

- In the relational data model, a design consists of a set of **relation schemas**.
- A relation schema has
  - a name, and
  - a set of attributes (+ types):

`Courses(code, name, teacher)`



Schema vs Instance

- ***Schema*** (or ***intension*** or a relation)
  - name and attributes of a relation

```
Courses(code, name, teacher)
```

- ***Instances*** (or ***extension*** of a relation)
  - the actual data
  - a set of **tuples**:

```
{ ('TDA357', 'Databases', 'Niklas Broberg'),
  ('TIN090', 'Algorithms', 'Devdatt Dubhashi') }
```

tuples

From schema to database

- The relations of the database schema become the tables when we implement the database in a DBMS. The tuples become the rows:

```
Courses(code, name, teacher)
```

relation schema

table instance

code	*name*	*teacher*
'TDA357'	'Databases'	'Niklas Broberg'
'TIN090'	'Algorithms'	'Devatt Dubhashi'

Keys

- Relations have **keys** – attributes whose values uniquely determine the values of all other attributes in the relation.

Courses(<u>code</u>, name, teacher)

key

{('TDA357', 'Databases', 'Niklas Broberg'),
~~**('TDA357', 'Algorithms', 'Devdatt Dubhashi')**~~**}**

Composite keys

- Keys can consist of several attributes

Courses(<u>code</u>, <u>period</u>, name, teacher)

```
{('TDA357', 2, 'Databases', 'Graham Kemp'),
 ('TDA357', 3, 'Databases', 'Niklas Broberg')}
```

Schemas and subschemas

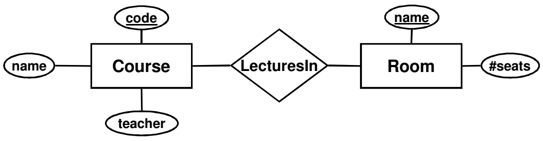
- A <u>schema</u> is a description of the entire database structure.
- A <u>subschema</u> is a description of only a part of the database structure.
  - Tailored to the needs of a user group
  - Controls access to data

Database design

- We design the conceptual model for our database using a high-level data model like the Enitity-Relationship model …
- … then we translate this design to the relational model (for implementation in an RDBMS).

Enitity-Relationship Diagram

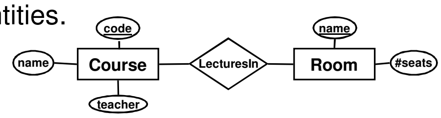
Example:



- A course has lectures in a room.
- A course is related to a room by the fact that the course has lectures in that room.
- A relationship is often named with a verb (e.g. HasLecturesIn)

Translation to relations

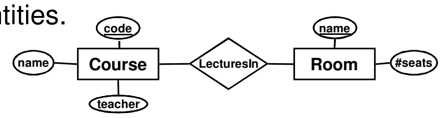
- A relationship between two entities is translated into a relation, where the attributes are the *keys* of the related entities.



⇨ **What?**

Translation to relations

- A relationship between two entities is translated into a relation, where the attributes are the *keys* of the related entities.



⇨
```
Courses(code, name, teacher)
Rooms(name, #seats)
LecturesIn(code, name)
```

Relational operators (1)

- Selection
  - Choose <u>rows</u> from a relation
  - State condition that rows must satisfy

$\sigma_{condition}(T)$

Examples:

$\sigma_{seats>100}(Rooms)$

$\sigma_{(code=\text{"TDA143" AND day="Friday"})}(Lectures)$

Relational operators (2)

- Projection
  - Choose <u>columns</u> from a relation
  - State which columns (attributes)

$\pi_A(T)$

Examples:

$\pi_{code}(Courses)$

$\pi_{name,seats}(Rooms)$

Relational operators (3)

$R_1 \times R_2$
- Cartesian product
- Combine each row of R_1 with each row of R_2

$R_1 \bowtie_{condition} R_2$
- join operator
- Combine row of R_1 with each row of R_2 if the condition is true

$R_1 \bowtie_{condition} R_2 = \sigma_{condition}(R_1 \times R_2)$

SQL

- SQL = Structured Query Language
- A very high-level *declarative* language.
  - Specify *what* information you want, not *how* to get that information (like you would in e.g. Java).
- Based on Relational Algebra

SELECT-FROM-WHERE

- Basic structure of an SQL query:

```
SELECT attributes
FROM   tables
WHERE  tests over rows
```

```
SELECT A
FROM   T
WHERE  C
```

⟺ $\pi_A(\sigma_C(T))$

Example:

GivenCourses =

course	per	teacher
TDA357	2	Niklas Broberg
TDA357	4	Rogardt Heldal
TIN090	1	Devdatt Dubhashi

```
SELECT *
FROM   GivenCourses
WHERE  course = 'TDA357';
```

Result =

What?

Example:

GivenCourses =

course	per	teacher
TDA357	2	Niklas Broberg
TDA357	4	Rogardt Heldal
TIN090	1	Devdatt Dubhashi

```
SELECT *
FROM   GivenCourses
WHERE  course = 'TDA357';
```

Result =

course	per	teacher
TDA357	2	Niklas Broberg
TDA357	4	Rogardt Heldal

Example:

GivenCourses =

course	per	teacher
TDA357	2	Niklas Broberg
TDA357	4	Rogardt Heldal
TIN090	1	Devdatt Dubhashi

```
SELECT course, teacher
FROM   GivenCourses
WHERE  course = 'TDA357';
```

Result =

What?

Example:

GivenCourses =

course	per	teacher
TDA357	2	Niklas Broberg
TDA357	4	Rogardt Heldal
TIN090	1	Devdatt Dubhashi

```
SELECT course, teacher
FROM   GivenCourses
WHERE  course = 'TDA357';
```

Result =

course	teacher
TDA357	Niklas Broberg
TDA357	Rogardt Heldal

Example:

```
SELECT code, name, period
FROM   Courses, GivenCourses
WHERE  teacher = 'Niklas Broberg'
       AND code = course;
```

Courses

code	*name*
TDA357	Databases
TIN090	Algorithms

GivenCourses

course	*per*	*teacher*
TDA357	2	Niklas Broberg
TDA357	4	Rogardt Heldal
TIN090	1	Devdatt Dubhashi

$\pi_{code,name,period}(\sigma_{teacher='Niklas\ Broberg'\ \&\ code\ =\ course}(Courses \times GivenCourses))$

Example:

```
SELECT code, name, period
FROM Courses, GivenCourses
WHERE teacher = 'Niklas Broberg'
      AND code = course;
```

code	*name*	*course*	*per*	*teacher*
TDA357	Databases	TDA357	2	Niklas Broberg
TDA357	Databases	TDA357	4	Rogardt Heldal
TDA357	Databases	TIN090	1	Devdatt Dubhashi
TIN090	Algorithms	TDA357	2	Niklas Broberg
TIN090	Algorithms	TDA357	4	Rogardt Heldal
TIN090	Algorithms	TIN090	1	Devdatt Dubhashi

$\pi_{code,name,period}(\sigma_{teacher='Niklas\ Broberg'\ \&\ code\ =\ course}(Courses \times GivenCourses))$

Example:

```
SELECT code, name, period
FROM   Courses, GivenCourses
WHERE teacher = 'Niklas Broberg'
      AND code = course;
```

code	*name*	*course*	*per*	*Teacher*
TDA357	Databases	TDA357	2	Niklas Broberg
TDA357	Databases	TDA357	4	Rogardt Heldal
TDA357	Databases	TIN090	1	Devdatt Dubhashi
TIN090	Algorit[illegible]	[illegible]	[illegible]	[illegible]
TIN090	Algorit[illegible]	[illegible]	[illegible]	[illegible]
TIN090	Algorit[illegible]	[illegible]	[illegible]	[illegible]

code	*name*	*course*	*per*	*teacher*
TDA357	Databases	TDA357	2	Niklas Broberg

$\pi_{code,name,period}(\sigma_{teacher='Niklas\ Broberg'\ \&\ code\ =\ course}(Courses \times GivenCourses))$

Example:

```
SELECT code, name, period
FROM   Courses, GivenCourses
WHERE  teacher = 'Niklas Broberg'
       AND code = course;
```

code	*name*	*course*	*per*	*teacher*
TDA357	Databases	TDA357	2	Niklas Broberg

code	*name*	*per*
TDA357	Databases	2

$\pi_{code,name,period}(\sigma_{teacher='Niklas\ Broberg'\ \&\ code\ =\ course}(Courses \times GivenCourses))$

Inserting data

```
INSERT INTO tablename
  VALUES (values for attributes);

INSERT INTO Courses
  VALUES ('TDA357', 'Databases');
```

code	name
TDA357	Databases

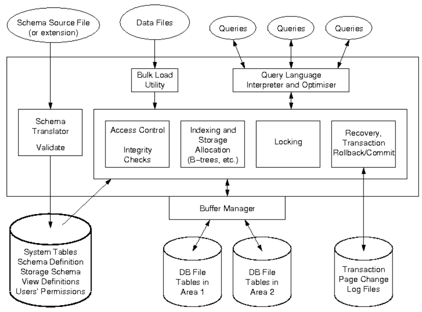
Deletions

```
DELETE FROM tablename
  WHERE test over rows;

DELETE FROM Courses
  WHERE code = 'TDA357';

DELETE FROM Courses;
```

Updates

```
UPDATE tablename
SET    attribute = ...
WHERE  test over rows

UPDATE GivenCourses
SET    teacher = 'Rogardt Heldal'
WHERE  code = 'TDA357'
  AND  period = 4;
```

Database system architecture



More about Databases

TDA357 - Databases

- 7,5 Higher education credits
- Runs twice each year, periods 2 and 3