Databases TDA357/DIT620

Niklas Broberg

niklas.broberg@chalmers.se

What’s a database anyway?

A database is …

- Structured
- Persistant
- Changable
- Digital

- True to integrity constraints

DBMS

Database

==

Data collection managed by a

specialized software called a

Database Management System (DBMS)

Why a whole course in 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, medicinal 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)

Why not a file system?

File systems are

- Structured
- Persistant
- Changable
- Digital

… but oh so inefficient!

Modern DBMS

- Handle *persistent* data
- Give *efficient* access to huge amounts of data
- Give a *convenient* interface to users
- Guarantee *integrity* constraints

- Handle transactions and concurrency

Database Management Systems

- Hierarchical databases:
  - ”Easy” to design if only one hierarchy
  - Efficient access
  - Low-level view of stored data
  - Hard to write queries
- Network databases:
  - ”Easy” to design
  - Efficient access
  - Low-level view of stored data
  - Very hard to write queries

Database Management Systems

- Relational databases:
  - **Hard to design**
  - Use specialized storage techniques
  - Efficient access
  - Provides high-level views of stored data based on mathematical concepts
  - **Easy to write queries**
  - Not all data fit naturally into a tabular structure
- Other databases (”NoSQL”):
  - Some based on semantic data models
  - Object-oriented database management systems (OODBMS)
  - XML-based, Key-value based, …

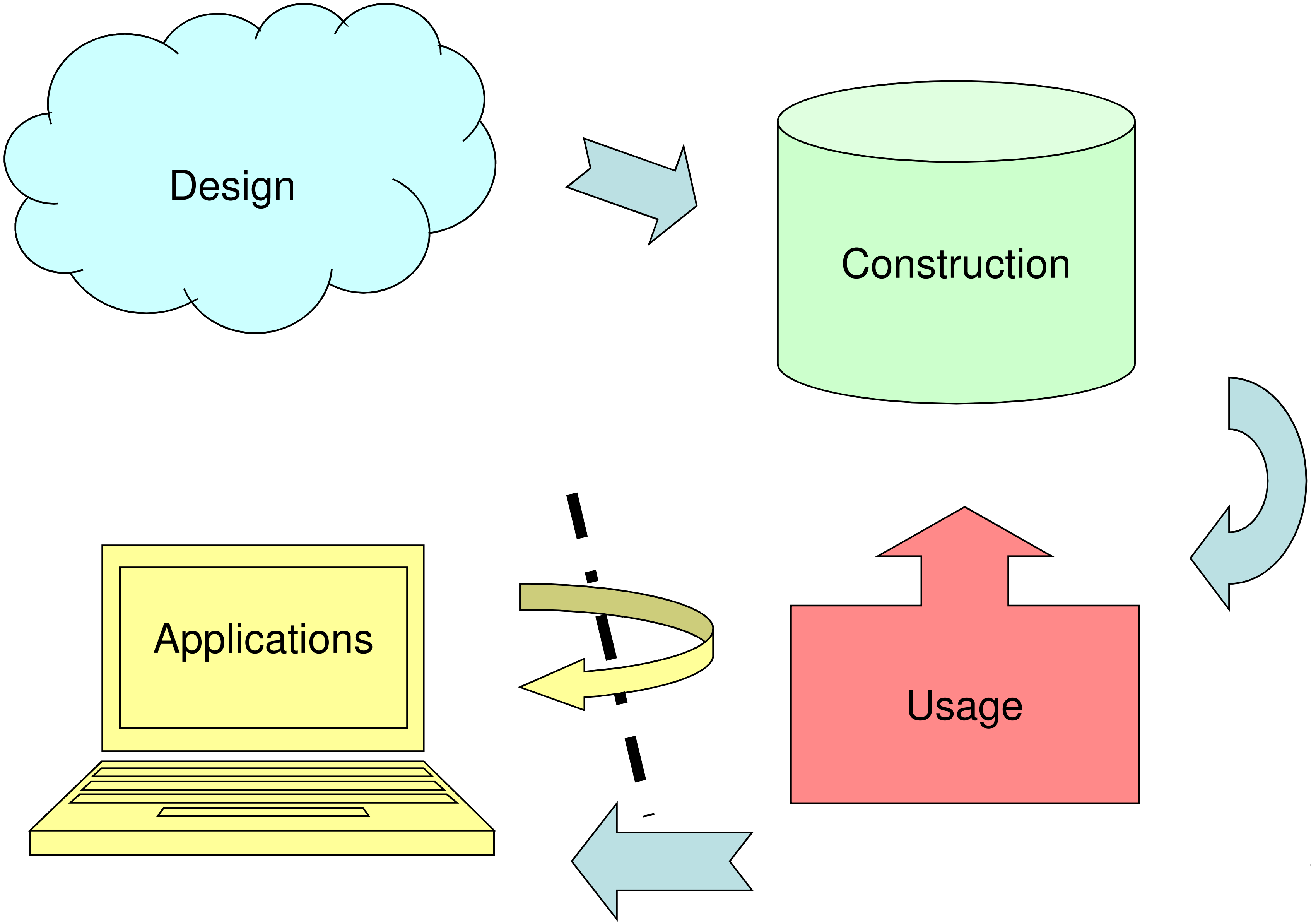
Relational DBMSs

- 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

Database system studies

1. Design of databases, e.g.
   - Entity-Relationship modelling
   - relational data model
   - dependencies and normalisation
   - XML and its data model
2. Database programming, e.g.
   - relational algebra
   - data manipulation and querying in SQL
   - application programs
   - querying XML
3. Database implementation, e.g.
   - indexes, transaction management, concurrency control, recovery, etc.

Course Objectives



Course Objectives – Design

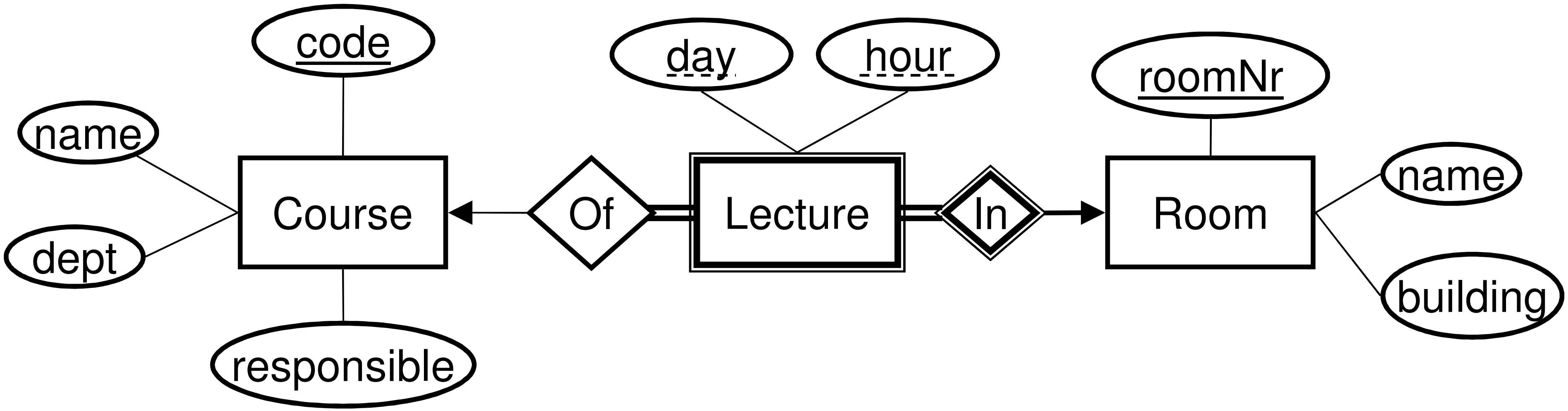
When the course is through, you should

- Given a domain, know how to design a database that correctly models the domain and its constraints

”We want a database that we can use for scheduling courses and lectures. This is how it’s supposed to work: …”

Course Objectives – Design

- Entity-relationship (E-R) diagrams
- Functional Dependencies
- Normal Forms



Course Objectives – Construction

When the course is through, you should

– Given a database schema with related constraints, implement the database in a relational DBMS

```
Courses(code, name, dept, examiner)
Rooms(roomNr, name, building)
Lectures(roomNr, day, hour, course)
  roomNr -> Rooms.roomNr
  course -> Courses.code
```

Course Objectives – Construction

- SQL Data Definition Language (DDL)

```
CREATE TABLE Lectures (
   lectureId INT PRIMARY KEY,
   roomId REFERENCES Rooms(roomId),
   day INT check (day BETWEEN 1 AND 7),
   hour INT check (hour BETWEEN 0 AND 23),
   course REFERENCES Courses(code),
   UNIQUE (roomId, day, hour)
);
```

Course Objectives – Usage

When the course is through, you should

- Know how to query a database for relevant data using SQL
- Know how to change the contents of a database using SQL

”Add a course ’Databases’ with course code ’TDA357’, given by …”

”Give me all info about the course ’TDA357’”

Course Objectives – Usage

- SQL Data Manipulation Language (DML)

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

- Querying with SQL

```
SELECT * FROM Courses WHERE code = 'TDA357';
```

Course Objectives – Applications

When the course is through, you should

– Know how to connect to and use a database from external applications

”We want a GUI application for booking rooms for lectures …”

Course Objectives – Applications

- JDBC

```
// Assemble the SQL command for inserting the
// newly booked lecture.
String myInsert = ″INSERT INTO Lectures ″
  + ″VALUES (″ + room + ″, ″
  + day + ″, ″ + hour + ″, ″ + course + ″)″;

// Execute the SQL command on the database
Statement stmt = myDbConn.createStatement();
stmt.executeUpdate(myInsert);
```

Course Objectives - Summary

You will learn how to

- design a database
- construct a database from a schema
- use a database through queries and updates
- use a database from an external application

Examination

- Written exam: Mar 14 (Fri) 8:30-12:30
  - 60 points (3/4/5 = 24/36/48, G/VG = 24/42)
  - Divided into 7 distinct blocks:
    - E-R diagrams (12)
    - FDs and Normal Forms (12)
    - SQL DDL (8)
    - Relational Algebra (6)
    - SQL (8)
    - Transactions (6)
    - XML (8)

Non-standard Exam Structure

- Each block will have two or three sections: A, B and possibly C. Everyone is expected to know the A questions, while B and C questions are intended for those seeking higher grades.
- You can only get points from one section within each block!
  - Less time spent on blocks that you know well.
  - Harder to get ”stray” points.
  - A’s give ~30

Exam – E-R diagrams (12)

”A small train company wants to design a booking system for their customers. …”

- Given the problem description above, construct an E-R diagram.
- Translate an E-R diagram into a database schema.

Exam – FDs and NFs (12)

”A car rental company has the following, not very successful, database. They want your help to improve it. …”

- Identify all functional dependencies you expect to hold in the domain.
- Indicate which of those dependencies violate BCNF with respect to the relations in the database.
- Do a complete decomposition of the database so that the resulting relations are in BCNF.

Exam – SQL DDL (8)

”A grocery store wants a database to store information about products and suppliers. After studying their domain you have come up with the following database schema. …”

- Write SQL statements that create the relations as tables in a DBMS.
- Write a trigger that, whenever the quantity in store of an item drops below a given threshold, inserts an order for that product with the supplier.

Exam – SQL (8)

”The grocery store wants your help in getting proper information from their database. …”

- Write a query that finds the total value of the entire inventory of the store.
- List all products with their current price, i.e. the discount price where such exists, otherwise the base price.

Exam – Relational Algebra (6)

”Here is a schema for a database over persons and their employments. …”

- What does this relational-algebraic expression compute? …

- (Write a relational-algebraic expression that computes … .)

Exam – Transactions (6)

”Here are some transactions that run in parallel. …”

- What will the end results given by the transactions be?
- What could happen if they were not run as transactions?

Exam –XML (8)

”A medical research facility wants a database that uses a semi-structured model to represent different degrees of knowledge regarding the outbreak of epidemic diseases. …”

- Suggest how to model this domain as a DTD.
- Discuss the benefits of the semi-structured model for this particular domain.
- What does this XQuery expression compute?

Exam – Total (60)

- Design ~30 points
- Construction ~10 points
- Usage ~20 points

- Applications = 0 points (?)

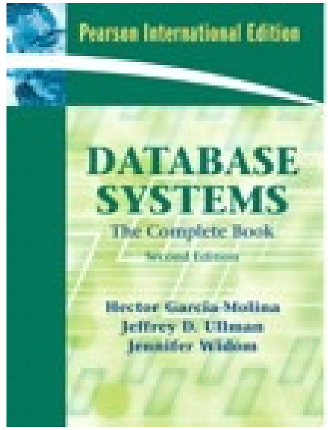
Lab Assignment

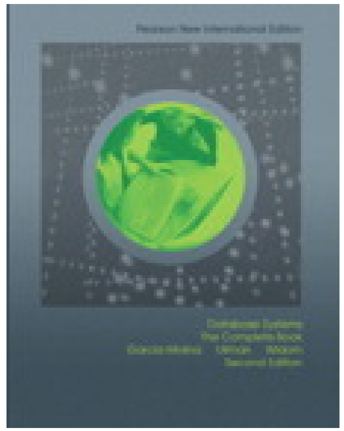
- Write a ”student portal” application in Java
  - Part I: Design
    - Given a domain description, design a database schema using an E-R diagram.
  - Part II: Design
    - Given a domain description, find and act on the functional dependencies of the domain to fix the schema from Part I.
  - Part III: Construction and Usage
    - Implement the schema from Part II in Oracle.
    - Insert relevant data.
    - Create views to support key operations.
  - Part IV: Construction
    - Create triggers to support key operations.
  - Part V: Interfacing from external Application
    - Write a Java application that uses the database from Part III.

Course Book

"Database Systems:
The Complete Book, 2E",
by Hector Garcia-Molina,
Jeffrey D. Ullman,
and Jennifer Widom

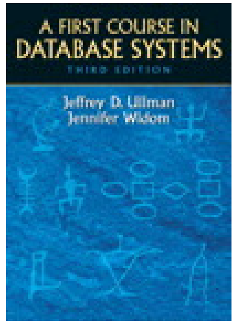
Approx. chapters 1-12



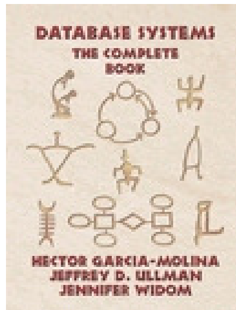


Alternative versions

"First Course in Database Systems, A, 3/E" by Jeffrey D. Ullman and Jennifer Widom



"Database Systems: The Complete Book", by Hector Garcia-Molina, Jeffrey D. Ullman, and Jennifer Widom



Approx. chapters 1-8

Schedule

	Mon	Tue	Wed	Thu	Fri
8-10			Tutorial, EL43	Lab, ED3507	
10-12	Lab, ED3507		Tutorial, EL43	Lecture, HC2	
13-15			[Lecture, HC2] Tutorial, EL43		Lab, ED6225
15-17	Lecture, HC2				

No tutorials or lab sessions in week 1!

Course assistants:

- Jonas Almström Duregård
- Gregoire Detrez
- Hamid Ebadi Tavallaei
- Evgenii Kotelnikov

Web resources

- Course webpage:

`http://www.cse.chalmers.se/edu/course/TDA357/`

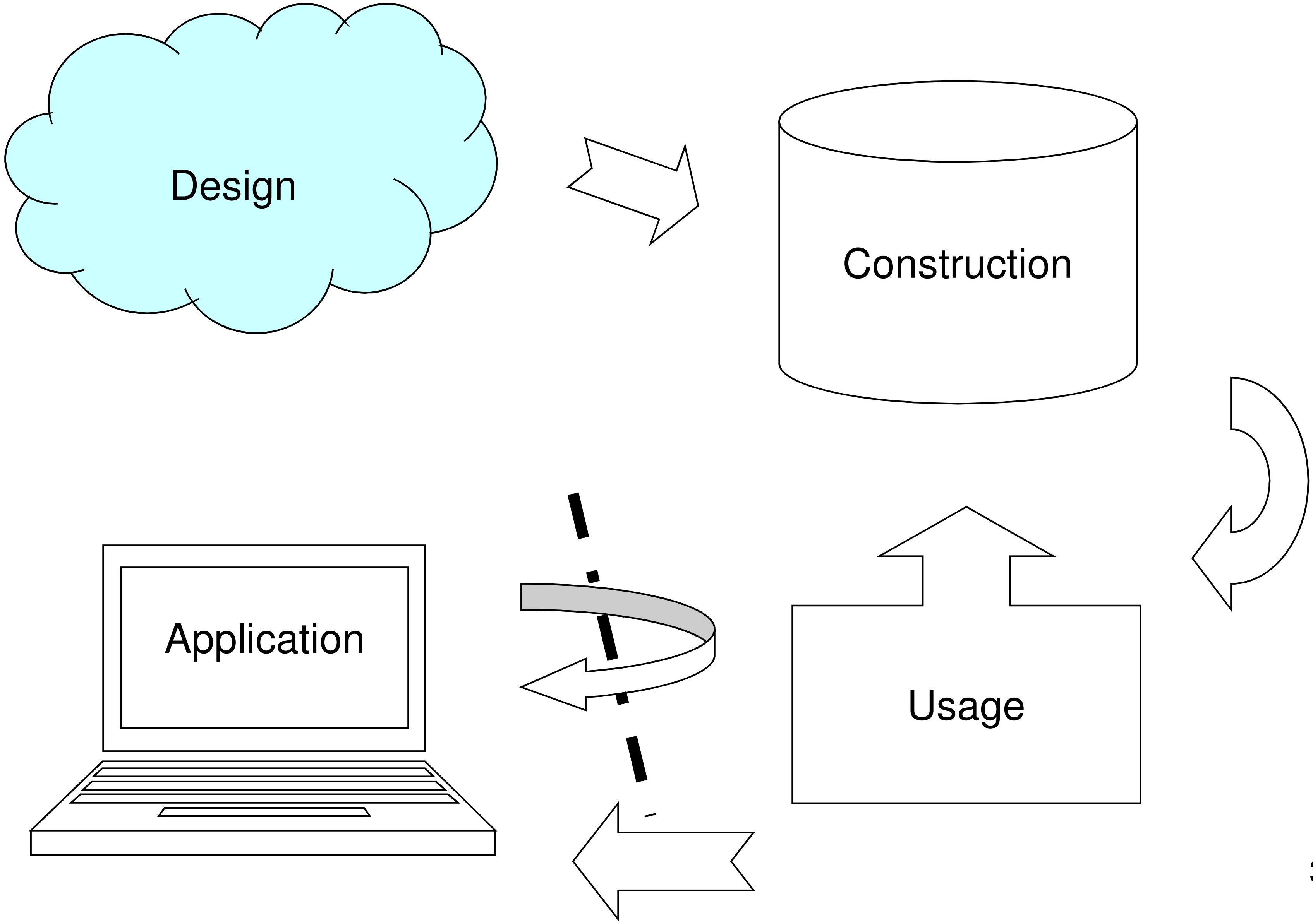
- Google discussion group:

`http://groups.google.com/group/tda357-vt2014`

Database design

Relations

Course Objectives



Course Objectives – Design

When the course is through, you should

- Given a domain, know how to design a database that correctly models the domain and its constraints

”We want a database that we can use for scheduling courses and lectures. This is how it’s supposed to work: …”

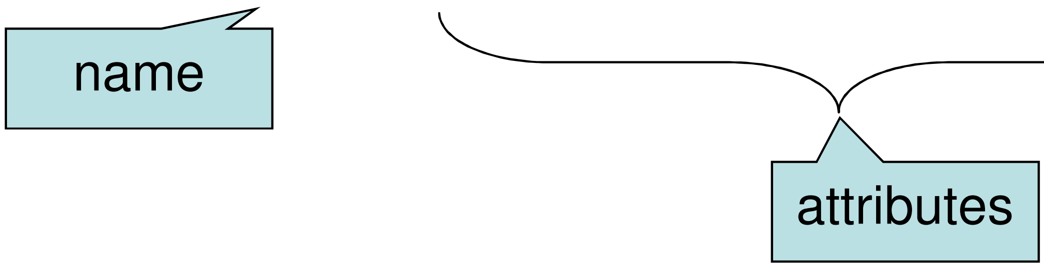
Designing a database

- ”Map” the domain, find out what the database is intended to model
  – The database should accept all data possible in reality
  – The database should agree with reality and not accept impossible or unwanted data
- Construct the ”blueprint” for the database – the database ***schema***

Database Schemas

- A database schema is 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*** – the logical structure of the relation (or database)
  - `Courses(code, name, teacher)`
- ***Instance*** – the actual content at any point in time

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

tuples

(like a blueprint for a house, and the actual house built from it.)

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'

Why relations?

- Relations often match our intuition regarding data
- Very simple model
- Has a good theoretical foundation from mathematics (set theory)

- The abstract model underlying SQL, the most important database language today

Keys

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

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

key

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

Composite keys

- Keys can consist of several attributes

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

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

In the schema, **code** and **period** are underlined.

Quiz time!

What's wrong with this schema?

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

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

Redundancy!

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

Scheduler database

”We want a database for an application that we will use to schedule courses. …”

- Course codes and names, and the period the courses are given
- The number of students taking a course
- The name of the course responsible
- The names of all lecture rooms, and the number of seats in them
- Weekdays and hours of lectures

Naive approach

- Not using a structured design method means it’s easy to make errors.
- Learn from the mistakes of others, then you won’t have to repeat them yourself!

First attempt

- Course codes and name, and the period the course is given
- The number of students taking a course
- The name of the course responsible
- The names of all lecture rooms, and the number of seats in them
- Weekday and hour of lectures

```
Schedules(code, name, period, numStudents,
  teacher, room, numSeats, weekday, hour)
```

Quiz: What’s a key of this relation?

First attempt

Schedules(<u>code</u>, name, <u>period</u>, numStudents, teacher, <u>room</u>, numSeats, <u>weekday</u>, hour)

code	*name*	*per.*	*#st*	*teacher*	*room*	*#seats*	*day*	*hour*
TDA357	Databases	3	87	Niklas Broberg	HC1	126	Monday	15:15
TDA357	Databases	3	87	Niklas Broberg	HC2	94	Thursday	10:00
TDA357	Databases	2	93	Graham Kemp	HC4	216	Tuesday	10:00
TDA357	Databases	2	93	Graham Kemp	VR	228	Friday	10:00
TIN090	Algorithms	1	64	Devdatt Dubhashi	HC1	126	Wednesday	08:00
TIN090	Algorithms	1	64	Devdatt Dubhashi	HC1	126	Thursday	13:15

Quiz: What’s wrong with this approach?

Anomalies

code	*name*	*per.*	*#st*	*teacher*	*room*	*#seats*	*day*	*hour*
TDA357	Databases	3	87	Niklas Broberg	HC1	**126**	Monday	15:15
TDA357	Databases	3	87	Niklas Broberg	HC2	94	Thursday	10:00
TDA357	Databases	2	93	Graham Kemp	HC4	216	Tuesday	10:00
TDA357	Databases	2	93	Graham Kemp	**VR**	**228**	Friday	10:00
TIN090	Algorithms	1	64	Devdatt Dubhashi	HC1	**126**	Wednesday	08:00
TIN090	Algorithms	1	64	Devdatt Dubhashi	HC1	**126**	Thursday	13:15

- Redundancy – same thing stored several times
- Update anomaly – we must remember to update all tuples

- Deletion anomaly – if no course has lectures in a room, we lose track of how many seats it has

Second attempt

`Rooms(`<u>`room`</u>`, numSeats)`

`Lectures(`<u>`code`</u>`, name, `<u>`period`</u>`, numStudents, teacher, `<u>`weekday`</u>`, hour)`

room	*#seats*
HC4	216
VR	228
HC1	126
HC2	94

code	*name*	*per*	*#st*	*teacher*	*day*	*hour*
TDA357	Databases	3	87	Niklas Broberg	Monday	15:15
TDA357	Databases	3	87	Niklas Broberg	Thursday	10:00
TDA357	Databases	2	93	Graham Kemp	Tuesday	10:00
TDA357	Databases	2	93	Graham Kemp	Friday	10:00
TIN090	Algorithms	1	64	Devdatt Dubhashi	Wednesday	08:00
TIN090	Algorithms	1	64	Devdatt Dubhashi	Thursday	13:15

Better? No!

Lost connection between **`Rooms`** and **`Lectures`**!

… and still there's redundancy in **`Lectures`**

Third attempt

```
Rooms(room, numSeats)
Courses(code, name)
CourseStudents(code, period, numStudents)
CourseTeachers(code, period, teacher)
Lectures(code, period, room, weekday, hour)
```

room	*#seats*
HC4	216
VR	228
HC1	126
HC2	94

code	*name*
TDA357	Databases
TIN090	Algorithms

code	*per*	*#st*
TDA357	3	87
TDA357	2	93
TIN090	1	64

code	*per*	*room*	*day*	*hour*
TDA357	3	HC1	Monday	15:15
TDA357	3	HC2	Thursday	10:00
TDA357	2	HC4	Tuesday	10:00
TDA357	2	VR	Friday	10:00
TIN090	1	HC1	Wednesday	08:00
TIN090	1	HC1	Thursday	13:15

code	*per*	*teacher*
TDA357	3	Niklas Broberg
TDA357	2	Graham Kemp
TIN090	1	Devdatt Dubhashi

Fourth attempt

Rooms(<u>room</u>, numSeats)

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

CoursePeriods(<u>code</u>, <u>period</u>, numStudents, teacher)

Lectures(<u>code</u>, <u>period</u>, room, <u>weekday</u>, hour)

room	*#seats*
HC4	216
VR	228
HC1	126
HC2	94

code	*name*
TDA357	Databases
TIN090	Algorithms

code	*per*	*#st*	*teacher*
TDA357	3	87	Niklas Broberg
TDA357	2	93	Graham Kemp
TIN090	1	64	Devdatt Dubhashi

code	*per*	*room*	*day*	*hour*
TDA357	3	HC1	Monday	15:15
TDA357	3	HC2	Thursday	10:00
TDA357	2	HC4	Tuesday	10:00
TDA357	2	VR	Friday	10:00
TIN090	1	HC1	Wednesday	08:00
TIN090	1	HC1	Thursday	13:15

Yeah, this is good!

Things to avoid!

- Redundancy
- Unconnected relations
- Too much decomposition

Next Lecture

More on Relations

Entity-Relationship diagrams