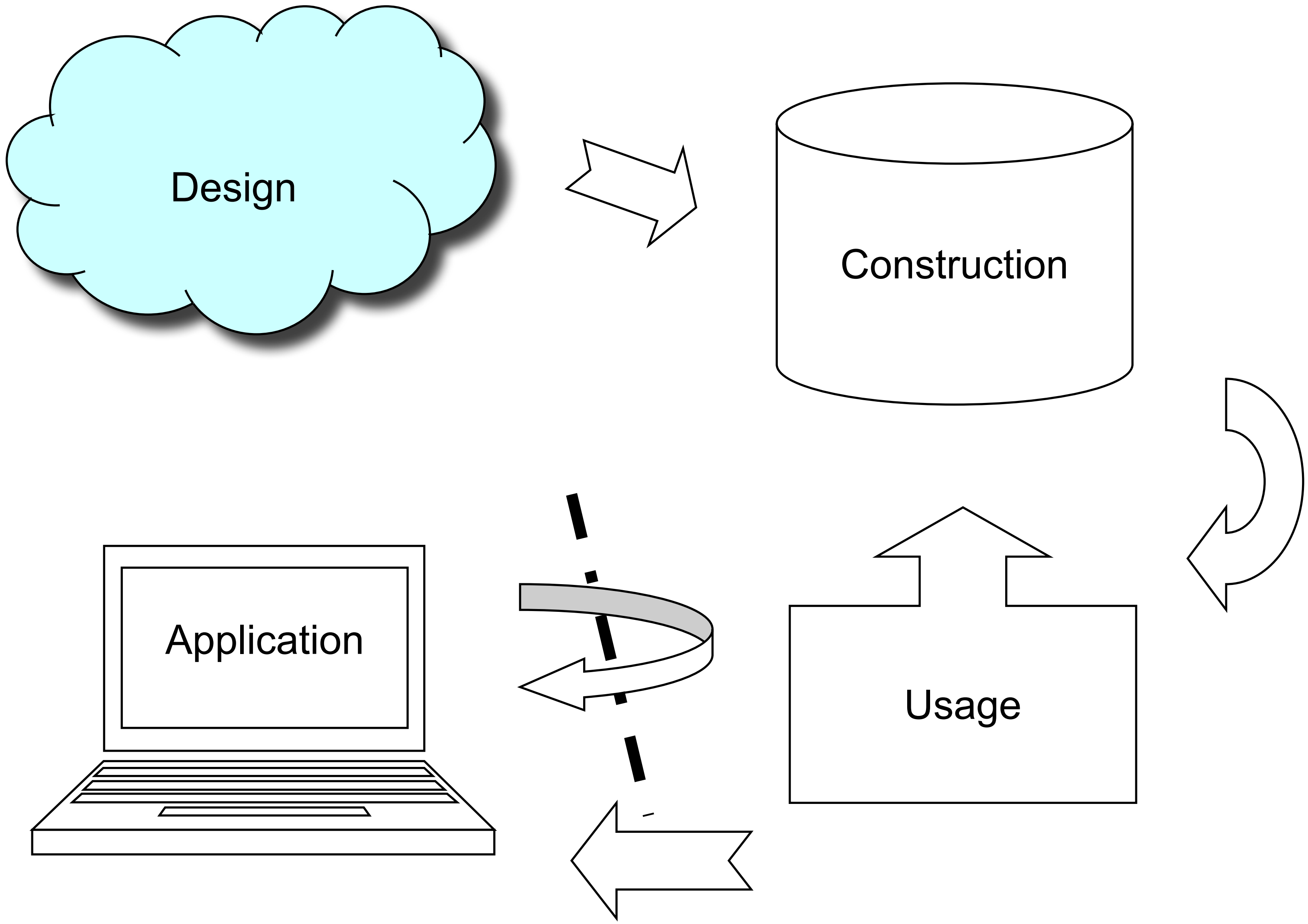
Lecture 1

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: …”

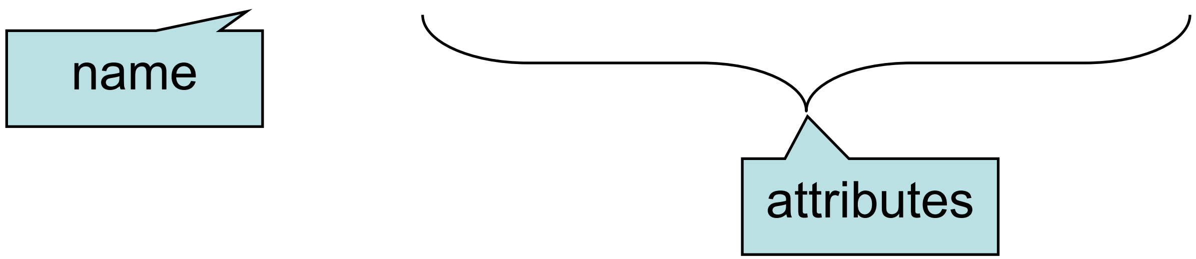
Designing a database

- ”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', 'Steven Van Acker'),
  ('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’	’Steven Van Acker’
’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(<u>code</u>, name, teacher)

key

```
{('TDA357', 'Databases', 'Steven Van Acker'),
 ('TDA357', 'Algorithms', 'Devdatt Dubhashi')}
```

Composite keys

- Keys can consist of several attributes

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

{('TDA357', 2, 'Databases', 'Steven Van Acker'),
 ('TDA357', 3, 'Databases', 'Aarne Ranta')}
```

Quiz time!

What's wrong with this schema?

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

{('TDA357', 2, 'Databases', 'Steven Van Acker'),
 ('TDA357', 3, 'Databases', 'Aarne Ranta')}
```

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	2	200	Steven Van Acker	HB2	186	Tuesday	10:00
TDA357	Databases	2	200	Steven Van Acker	HB2	186	Wednesday	08:00
TDA357	Databases	3	93	Aarne Ranta	HC4	216	Tuesday	10:00
TDA357	Databases	3	93	Aarne Ranta	VR	228	Friday	10:00
TIN090	Algorithms	1	64	Devdatt Dubhashi	HB2	186	Wednesday	08:00
TIN090	Algorithms	1	64	Devdatt Dubhashi	HB2	186	Thursday	13:15

Quiz: What’s wrong with this approach?

Anomalies

code	*name*	*per.*	*#st*	*teacher*	*room*	*#seats*	*day*	*hour*
TDA357	Databases	2	200	Steven Van Acker	HB2	**186**	Tuesday	10:00
TDA357	Databases	2	200	Steven Van Acker	HB2	**186**	Wednesday	08:00
TDA357	Databases	3	93	Aarne Ranta	HC4	216	Tuesday	10:00
TDA357	Databases	3	93	Aarne Ranta	**VR**	**228**	Friday	10:00
TIN090	Algorithms	1	64	Devdatt Dubhashi	HB2	**186**	Wednesday	08:00
TIN090	Algorithms	1	64	Devdatt Dubhashi	HB2	**186**	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
HB2	186
HA4	182

code	*name*	*per*	*#st*	*teacher*	*day*	*hour*
TDA357	Databases	2	200	Steven Van Acker	Tuesday	10:00
TDA357	Databases	2	200	Steven Van Acker	Wednesday	08:00
TDA357	Databases	3	93	Aarne Ranta	Tuesday	10:00
TDA357	Databases	3	93	Aarne Ranta	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
HB2	186
HA4	182

code	name
TDA357	Databases
TIN090	Algorithms

code	per	#st
TDA357	2	200
TDA357	3	93
TIN090	1	64

code	per	room	day	hour
TDA357	2	HB2	Tuesday	10:00
TDA357	2	HB2	Wednesday	08:00
TDA357	3	HC4	Tuesday	10:00
TDA357	3	VR	Friday	10:00
TIN090	1	HB2	Wednesday	08:00
TIN090	1	HB2	Thursday	13:15

code	per	teacher
TDA357	2	Steven Van Acker
TDA357	3	Aarne Ranta
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
HB2	186
HA4	182

code	*name*
TDA357	Databases
TIN090	Algorithms

code	*per*	*#st*	*teacher*
TDA357	2	200	Steven Van Acker
TDA357	3	93	Aarne Ranta
TIN090	1	64	Devdatt Dubhashi

code	*per*	*room*	*day*	*hour*
TDA357	2	HB2	Tuesday	10:00
TDA357	2	HB2	Wednesday	08:00
TDA357	3	HC4	Tuesday	10:00
TDA357	3	VR	Friday	10:00
TIN090	1	HB2	Wednesday	08:00
TIN090	1	HB2	Thursday	13:15

Yeah, this is good!

Things to avoid!

- Redundancy
- Unconnected relations
- Too much decomposition

Summary

- A database schema is a blueprint
  - Consists of a set of relations e.g. Courses(code, name, teacher) where "Courses" is the relation name and code, name and teacher are attributes.
- A database instance holds actual data
  - Tuples are instances of a relation.
    - E.g. ('TDA357', 'Databases', 'Steven Van Acker')
- In a DBMS, a table holds relations where:
  - Each row holds a tuple
  - Each column stores a different attribute
- Keys uniquely identify the other values of a tuple in a relation
  - Composite keys combine several attributes
- Avoid
  - Redundancy
  - Unconnected relations
  - Too much decomposition

Next time, Lecture 2

More on Relations

Entity-Relationship diagrams