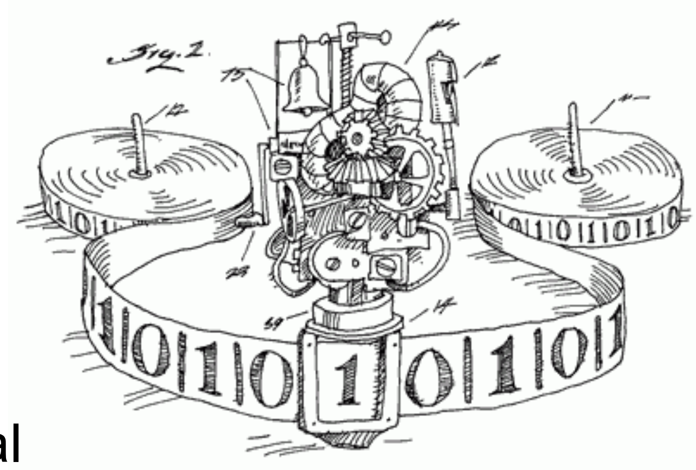
INFO 101 – Introduction to Computing and Security

[2020 - Week 6 / 1]



Prof. Dr. Rui Abreu

IST, University of Lisbon, Portugal

rui@computer.org

@rmaranhao

What we are going to discuss…

- Introduction to databases including data base design
- Basic principles behind relational databases

Data and Information

- “Data are facts”
- “Information is data grouped together and put into a context.”

Here is data: 18.43, 8.21, 210.00, 59.81

Let us add more, e.g.: $18.43, $8.21, $210.00, $59.81

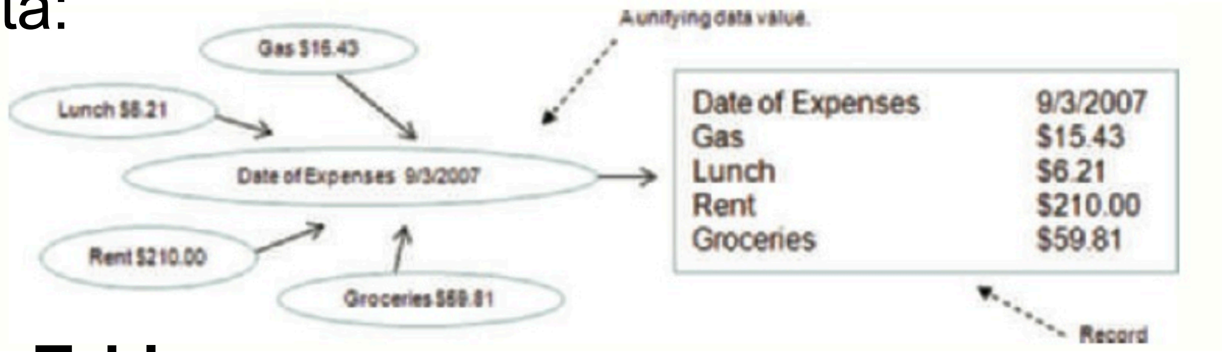
But we still do not know the meaning of given data!

Gas	$18.43
Lunch	$8.21
Rent	$210.00
Groceries	$59.81

Given the context, we now know the meaning of the given data!

Data and Information

- Adding more information about the context further increases our understanding of data:

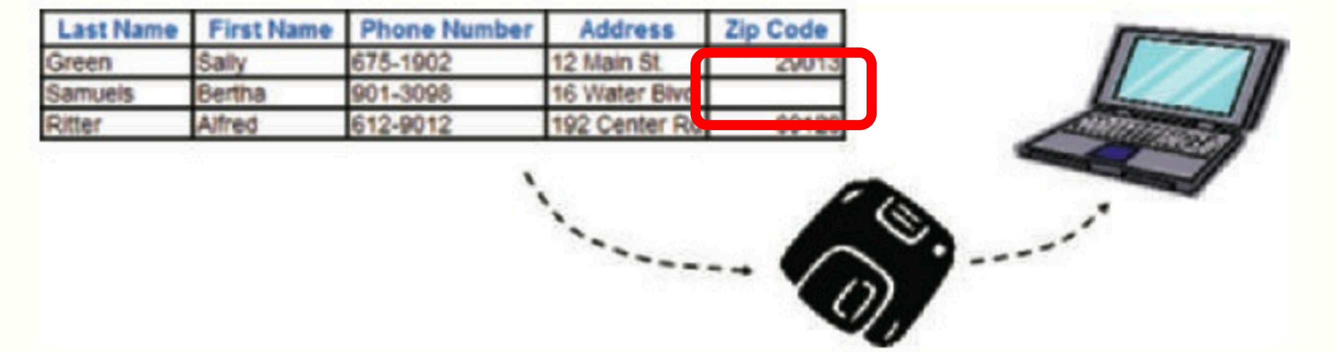


- Information stored in **Tables**:

	Customer	Item	Price	Quantity	Extended Cost	Shipping	Total
Header							
Records	Winston	Chair	$ 99.99	$ 2.00	$ 199.98	$ 32.00	$ 231.98
	Rudy	Desk	$ 219.00	$ 1.00	$ 219.00	$ 30.00	$ 249.00
	Samuels	Chair	$ 99.99	$ 6.00	$ 599.94	$ 60.00	$ 659.94
	Rudy	Rug	$ 107.00	$ 1.00	$ 107.00	$ 25.00	$ 132.00
	Jones	Clock	$ 25.67	$ 2.00	$ 51.34	$ 12.00	$ 63.34
	Winston	Desk	$ 301.00	$ 1.00	$ 301.00	$ 50.00	$ 351.00

Tables

- Often tables are used to store data. The added context using the name of the table, and the column labels supports extracting a meaning.
- Tables might contain NULL values (if information is not known)
- Tables Are Persistently Stored in Files on Disks or Other Storage Devices



Last Name	First Name	Phone Number	Address	Zip Code
Green	Sally	675-1902	12 Main St.	20013
Samuels	Bertha	901-3098	16 Water Blv[illegible]	
Ritter	Alfred	612-9012	192 Center R[illegible]	[illegible]

Tables

- We may select records (=rows) from tables
- Or query the content, hence filtering the relevant pieces of information for us

Origin	Destination	Ticket Cost
San Diego	Boston	$ 59.00
Denver	Philadelphia	$ 198.00
San Diego	Tampa	$ 215.00
Portland	Boston	$ 112.00
San Diego	Tampa	$ 205.00
Denver	Chicago	$ 157.00
Tampa	San Diego	$ 200.00

Origin	Destination	Ticket Cost
San Diego	Tampa	$ 215.00
San Diego	Tampa	$ 205.00

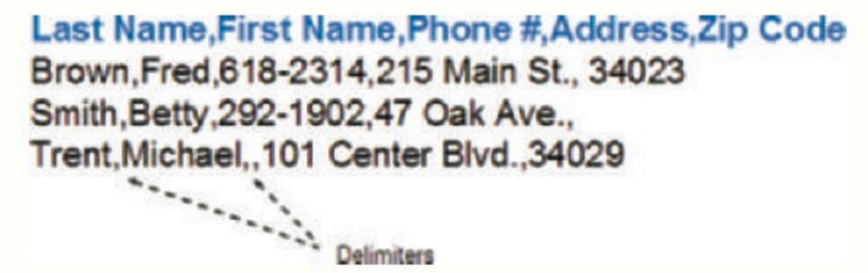
- Which is often used for **decision making**

Rules for Records and Tables

- Some rules apply:
  - The order of rows does not matter
  - The order of columns does not matter
  - All records (i.e. rows) must have the same number of fields (i.e. columns)
  - All data in a column (i.e. field) must be of same type.

How to store records (and tables)?

- In a flat (textual) file
  - E.g. CSV (Comma separated values)



- In a binary file (needs coding)

- In a spreadsheet like Excel (currently using XML)

- In a database

How information is stored in a computer?

- Unstructured
  - E.g. in a text file

There is no specific rule about where pieces of information are stored.

- Semi-structured
  - E.g. in a XML file

The stored information follows certain rules, e.g., the XML grammar

- Structured
  - E.g. in a database

The stored information is structured (like in a table). Keys, fields, and their types are known.

Why using structure in storing information?

- Performance – There is no redundancy in information stored
- Access time can be optimized (or even minimized)
- Transaction management can be implemented

- Writing and searching for Queries is easier!

    - Which persons in a company have currently an income of $5.000 or more per month?
    - Is there a teacher who is visiting his own lecture as a student?
    - ...

Databases

- A **database** is a collection of records of information which are stored so that a computer program can access it to answer questions.
- A **database management system (DBMS)** is a program (software) designed to manage a database

- A **relational database** is a collection of tables:
  - There are many DBMS available, like: Oracle, Microsoft SQL Server, IBM DB2, mySQL, PostgreSQL, …

Database types

- **Relational** — A relational database is implemented as data tables with logical pointers (keys) that connect records in various tables.
- **Object-oriented** — In this type of database data is arranged in objects.
- **Multidimensional** — This type of database is a data aggregator which combines data from a multitude of data sources. It can also be known as a meta-database.
- **Hierarchical** — This type of database was favored by IBM in the past and is tree structured. It is generally not in use any longer.
- **Network** — This is an older type of database structure that used physical pointers to connect records. It is generally not in use any longer.

Relational Databases

- Relational Database (RDB) schema: Set of tables

Columns, attributes, fields

Records, tuples, rows

Lect_num	Type	Name	Hours	Teacher	Room
117.101	VO	Software Maintenance	3.0	Franz Wotawa	LH i7
117.102	VO	Compiler Construction	2.0	Franz Wotawa	LH i11
117.111	UE	Compiler Construction	1.0	Birgit Hofer	LH i11
....	......	....	.....	....	....

Table, relation

How to obtain a good RDB schema?

1

Lect_num	Type	Name	Hours	Teacher	Room	Address
117.101	VO	Software Maintenance	3.0	Franz Wotawa	LH i7	Inffeldgasse 25
117.102	VO	Compiler Construction	2.0	Franz Wotawa	LH i11	Infeldgasse 16
117.111	UE	Compiler Construction	1.0	Birgit Hofer	LH i11	Inffeldgasse 16
…………..	……	………………..	…..	………….	……..	

WHICH IS BETTER?

2

Lect_num	Type	Name	Hours	Teacher	Room
117.101	VO	Software Maintenance	3.0	Franz Wotawa	LH i7
117.102	VO	Compiler Construction	2.0	Franz Wotawa	LH i11
117.111	UE	Compiler Construction	1.0	Birgit Hofer	LH i11
…………..	……	………………..	…..	………….	……….

Room	Address
LH i7	Inffeldgasse 25
LH i11	Infeldgasse 16
……….	

Theoretical background

Relational schema / relational calculus

- A database is a set of relations (i.e. tables).
- Every relation R has a corresponding schema
- A **relational schema** is a set of attributes $A_1,..,A_n$
- Every **relation** *r* has a schema *R* and we write *r(R)*

- **Example**: The relation *LectureHall* has a schema *LectureHall(Room,Address).* Note that *(i7,Inffeldgasse 25*) is a record of relation *LectureHall* (also called tuple)

Room	Address
LH i7	Inffeldgasse 25
LH i11	Infeldgasse 16
………..	

Relational Calculus

- Relations their schema and operations
- Operations / functions:
    - Selection of tuples (select)
    - Selection of fields (projection)
    - Combine data (union, intersection,...)
    - ...

Union of tables / relations

- Combine 2 tables based on the same schema
- The result comprises tuples from both tables

$$R \cup S := \{t | t \in R \lor t \in S\}$$

Table 1

Id	Given name	Surname
1000	Ludwig	Beethoven
1010	Wolfgang	Mozart

Table 2

Id	Given name	Surname
2000	Isaac	Newton
1010	Wolfgang	Mozart

Result table

Id	Given name	Surname
1000	Ludwig	Beethoven
1010	Wolfgang	Mozart
2000	Isaac	Newton

Intersection of tables / relations

- Combine 2 tables based on the same schema
- The result comprises tuples that are in both tables only

Table 1

Id	Given name	Surname
1000	Ludwig	Beethoven
1010	Wolfgang	Mozart

Table 2

Id	Given name	Surname
2000	Isaac	Newton
1010	Wolfgang	Mozart

$$R \cap S := \{t | t \in R \wedge t \in S\}$$

Result table

Id	Given name	Surname
1010	Wolfgang	Mozart

Cartesian product (cross product)

- Combines two arbitrary tables
- The result comprises all different combination of tuples!

$$R \times S := \{(a_1, a_2, ..., a_n, b_1, b_2, ..., b_m) | (a_1, a_2, ..., a_n) \in R \wedge (b_1, b_2, ..., b_m) \in S\}$$

R:

A	B	C	D
1	2	3	4
4	5	6	7
7	8	9	0

S:

E	F	G
1	2	3
7	8	9

R × S:

A	B	C	D	E	F	G
1	2	3	4	1	2	3
4	5	6	7	1	2	3
7	8	9	0	1	2	3
1	2	3	4	7	8	9
4	5	6	7	7	8	9
7	8	9	0	7	8	9

Projection

- Selection of columns in a table
- Given: $R(A_1,..,A_n)$ and $\beta \subseteq \{A_1,..,A_n\}$

$$\pi_\beta(R) := \{t_\beta | t \in R\}$$

Id	Given name	Surname
1000	Ludwig	Beethoven
1010	Wolfgang	Mozart
2000	Isaac	Newton

$\pi_{\{\text{Given name,Surname}\}}$

Given name	Surname
Ludwig	Beethoven
Ludwig	Attersee
Isaac	Newton

$\pi_{\{\text{Given name}\}}$

Given name
Ludwig
Wolfgang
Isaac

Selection

- Selection of rows that fulfill a given condition
- Given: $R(A_1,..,A_n)$ and a condition C

$$\sigma_C(R) := \{t \mid t \in R \text{ and } t \text{ fulfills } C\}$$

Id	Given name	Surname
1000	Ludwig	Beethoven
1010	Wolfgang	Mozart
2000	Isaac	Newton
3010	Ludwig	Attersee

$\sigma_{\{\text{Given name}=„\text{Ludwig}“\}}$

SVNR	Vorname	Nachname
1000	Ludwig	Beethoven
3010	Ludwig	Attersee

Natural Join

- Combine tables that share some common attributes
- Given: $R(A_1,..,A_n,B_1,..,B_n)$ and $S(B_1,..,B_n, C_1,..,C_n)$

$$R \bowtie S := \{r \cup s_{[C_1,\ldots,C_n]} | r \in R \wedge s \in S \wedge r_{[B_1,\ldots,B_n]} = s_{[B_1,\ldots,B_n]}\}$$

R:

A	B	C	D
1	2	3	4
4	5	6	7
7	8	9	0

S:

A	F	G
1	2	3
7	8	9

NATURAL JOIN (R, S):

A	B	C	D	F	G
1	2	3	4	2	3
7	8	9	0	8	9

Semi-Join

- Like the Natural Join but using projection on the attributes of the left relation.
- Given: $R(A_1,..,A_n,B_1,..,B_n)$ and $S(B_1,..,B_n, C_1,..,C_n)$

$$R \ltimes S := \{r | r \in R \land s \in S \land r_{[B_1,...,B_n]} = s_{[B_1,...,B_n]}\}$$

R:

A	B	C	D
1	2	3	4
4	5	6	7
7	8	9	0

S:

A	F	G
1	2	3
7	8	9

SEMIJOIN (R, R.A = S.A, S):

A	B	C	D
1	2	3	4
7	8	9	0

Summary of the Relational Calculus

- Foundation for database queries!
  - E.g., with selection and projection we can represent a lot of queries!

- There are more operations
- Many operations can be defined using some basic operators

Functional dependencies

- Functional dependency (FD) is a relationship between attributes of a database relation (or table)
- Used in database design for normalization and identification of the database keys

- **Example**: In a personnel database people are usually stored based on their unique social security number (SSN) !
  - *Personnel(SSN, Given name, Surname) SSN* uniquely determines *Given name* und *Surname*!
  - We write: *{SSN}* $\rightarrow$ *{Given name, Surname}*

SSN	Given name	Surname
1000	Ludwig	Beethoven
1010	Wolfgang	Mozart
2000	Isaac	Newton
3010	Ludwig	Attersee
4001	Franz	Wotawa
4002	Gerhard	Wotawa

Formal definition of FDs

- Given a relational schema R. Let α and β be subsets of the attributs of R. β is functional dependent on α ($\alpha \rightarrow \beta$), if the following condition is true:

$$\forall t_1, t_2 \in r: t_1[\alpha]=t_2[\alpha] \rightarrow t_1[\beta]=t_2[\beta]$$

SSN	Given name	Surname
1000	Ludwig	Beethoven
1010	Wolfgang	Mozart
2000	Isaac	Newton
3010	Ludwig	Attersee
4001	Franz	Wotawa
4002	Gerhard	Wotawa

We see that {SSN} $\rightarrow$ *{Given name, Surname}*
BUT NOT:
{Given name} $\rightarrow$ *{SNN} because Ludwig has two SSNs*
OR
{Surname} $\rightarrow$ *{Given name} because Wotawa leads to Franz and Gerhard*

Primary keys

- Is there a set of attributes α that identify each record of a table, then α *is called a **super key***.
- A **candidate key** is a minimal super key (i.e., this is a super key where not attributes can be anymore removed).
- A **primary key** is of a table is one selected candidate key (there may be more).

Because of {SSN} $\rightarrow$ *{Given name, Surname}* {SSN} is a super key and also a primary key!

{SSN} *identifies each record in the database table on the right. {SSN} is also minimal and can be, therefore, used as a primary key for this table.*

NOTE: There is no record in this table with the same SSN.

SSN	Given name	Surname
1000	Ludwig	Beethoven
1010	Wolfgang	Mozart
2000	Isaac	Newton
3010	Ludwig	Attersee
4001	Franz	Wotawa
4002	Gerhard	Wotawa

Normal forms

- **Goal**:
  - Remove redundancies in databases!
  - Improve data consistency in databases!

- Data inconsistencies usually are revealed in case of changes. In case of schemes without redundancies there are no consistency problems!

Example of data inconsistencies

- Change of the address of a lecture hall!

Lect_num	Type	Name	Hours	Teacher	Room	Address
117.101	VO	Software Maintenance	3.0	Franz Wotawa	LH i7	Inffeldgasse 25
117.102	VO	Compiler Construction	2.0	Franz Wotawa	LH i11	Infeldgasse 16
117.111	UE	Compiler Construction	1.0	Birgit Hofer	LH i11	Inffeldgasse 16
....	......	....	.....	....	....	

Change 16 to 17!

Lect_num	Type	Name	Hours	Teacher	Room	Address
117.101	VO	Software Maintenance	3.0	Franz Wotawa	LH i7	Inffeldgasse 25
117.102	VO	Compiler Construction	2.0	Franz Wotawa	LH i11	Infeldgasse 17
117.111	UE	Compiler Construction	1.0	Birgit Hofer	LH i11	Inffeldgasse 16
....	......	....	.....	....	....	

$\neq$

1. Normal form (1NF)

- *“A relation is in first normal form if and only if the* ***domain*** *of each attribute* ***contains only atomic (indivisible) values****, and the value of each attribute contains only a single value from that domain.”*

- The 1NF enables queries and sorting!

- There is one extension: *“... and each relation must have a primary key.”*

Example 1NF

Lect_num	Type	Name	Hours	Teacher	Room	Address
117.101	VO	Software Maintenance	3.0	Franz Wotawa, Birgit Hofer	LH i7	Inffeldgasse 25
117.102	VO	Compiler Construction	2.0	Franz Wotawa	LH i11	Infeldgasse 16
117.111	UE	Compiler Construction	1.0	Birgit Hofer	LH i11	Inffeldgasse 16
…………..	……	……………….	…..	………….	……..	

The relation above is not in 1NF!!!

Lect_num	Type	Name	Hours	Teacher	Room	Address
117.101	VO	Software Maintenance	3.0	Franz Wotawa	LH i7	Inffeldgasse 25
117.101	VO	Software Maintenance	3.0	Birgit Hofer	LH i7	Inffeldgasse 25
117.102	VO	Compiler Construction	2.0	Franz Wotawa	LH i11	Infeldgasse 16
117.111	UE	Compiler Construction	1.0	Birgit Hofer	LH i11	Inffeldgasse 16
…………..	……	……………….	…..	………….	……..	

2. Normal form (2NF)

- *“A relation in 1NF is in 2NF if and only if no **non-prime attribute** is **dependent** on any **proper subset** of any candidate key of the relation.*

  ***A non-prime attribute of a relation** is an attribute that is not a part of any candidate key of the relation.“*

Every non-prime attribute has to be dependent on the key only

- A database that is not in 2NF comprises redundancies!

Example 2NF

- Let us consider the following table with primary key {Lect_num,SSN}:

Lect_num	Type	Name	Hours	SSN	Teacher	Room	Address
117.101	VO	Software Maintenance	3.0	1000	Franz Wotawa	LH i7	Inffeldgasse 25
117.101	VO	Software Maintenance	3.0	1002	Birgit Hofer	LH i7	Inffeldgasse 25
117.102	VO	Compiler Construction	2.0	1000	Franz Wotawa	LH i11	Infeldgasse 16
117.111	UE	Compiler Construction	1.0	1002	Birgit Hofer	LH i11	Inffeldgasse 16
....	......	....	.....		....	....	

- Dependencies:
  - {Lect_num} → *{Type, Name, Hours, Room}*
  - {SSN} → *{Teacher}*
  - *{Room}* → *{Address}*

Example 2NF

This table is not in 2NF!!!

- Type, name, and hours of a lecture do only depend on the lecture number (Lect_num) and not on the teacher!

Redundancy

Lect_num	Type	Name	Hours	SSN	Teacher	Room	Address
117.101	VO	Software Maintenance	3.0	1000	Franz Wotawa	LH i7	Inffeldgasse 25
117.101	VO	Software Maintenance	3.0	1002	Birgit Hofer	LH i7	Inffeldgasse 25
117.102	VO	Compiler Construction	2.0	1000	Franz Wotawa	LH i11	Infeldgasse 16
117.111	UE	Compiler Construction	1.0	1002	Birgit Hofer	LH i11	Inffeldgasse 16
....	......	....			....	....	

Make 3 tables!!!

Lect_num	Type	Name	Hours	Room	Address
117.101	VO	Software Maintenance	3.0	LH i7	Inffeldgasse 25
117.102	VO	Compiler Construction	2.0	LH i11	Infeldgasse 16
117.111	UE	Compiler Construction	1.0	LH i11	Inffeldgasse 16
....	......	....	.....	....	

Lect_num	SSN
117.101	1000
117.101	1002
117.102	1000
117.111	1002
....	....

SSN	Teacher
1000	Franz Wotawa
1002	Birgit Hofer
....	....

3. Normal form (3NF)

- *“A relation in 2NF is in 3NF if and only if all the attributes in a table are determined only by the candidate keys of that relation and not by any non-prime attributes.*

  ***No non-prime attributes are allowed to be transitive dependent on a prime attribute**!”*

- Eliminates problems occurring when changing information!

Example 3NF

{Lect_num} → *{Type, Name, Hours, Room}*
{Room} → *{Address}*

- **This table is not in 3NF!!!**
  - The room of the lecture depends functionally on the lecture number (Lect_num). The address of the room depends functionally on the room itself. Hence, we have a transitive dependency between a key and an attribute that is not in the key!

Lect_num	Type	Name	Hours	Room	Address
117.101	VO	Software Maintenance	3.0	LH i7	Inffeldgasse 25
117.101	VO	Software Maintenance	3.0	LH i7	Inffeldgasse 25
117.102	VO	Compiler Construction	2.0	LH i11	Infeldgasse 16
117.111	UE	Compiler Construction	1.0	LH i11	Inffeldgasse 16
....	......	....	.....	....	

Make 2 tables!!!

Lect_num	Type	Name	Hours	Room
117.101	VO	Software Maintenance	3.0	LH i7
117.101	VO	Software Maintenance	3.0	LH i7
117.102	VO	Compiler Construction	2.0	LH i11
117.111	UE	Compiler Construction	1.0	LH i11
....	......	....	.....	....

Room	Address
LH i7	Inffeldgasse 25
LH i7	Inffeldgasse 25
LH i11	Infeldgasse 16
LH i11	Inffeldgasse 16
....	

Entity-Relationship (ER) Diagrams

Objective behind ER Diagrams

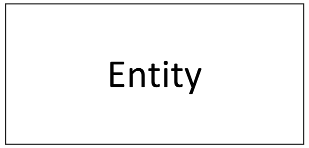
- Simple graphical method for the design of database schemes

- Method:
  1. Identification of Entities and their relationships
  2. Extraction of relation scheme from the graphical representation
  3. Normalization of the relation scheme

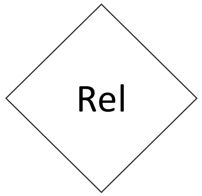
- ER Diagrams are very much similar to UML Class Diagrams
  We even can use the noun method to extract the entities!

ER Diagrams

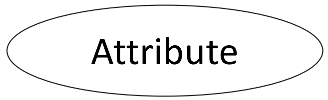
- **Entity**: similar to a class in UML



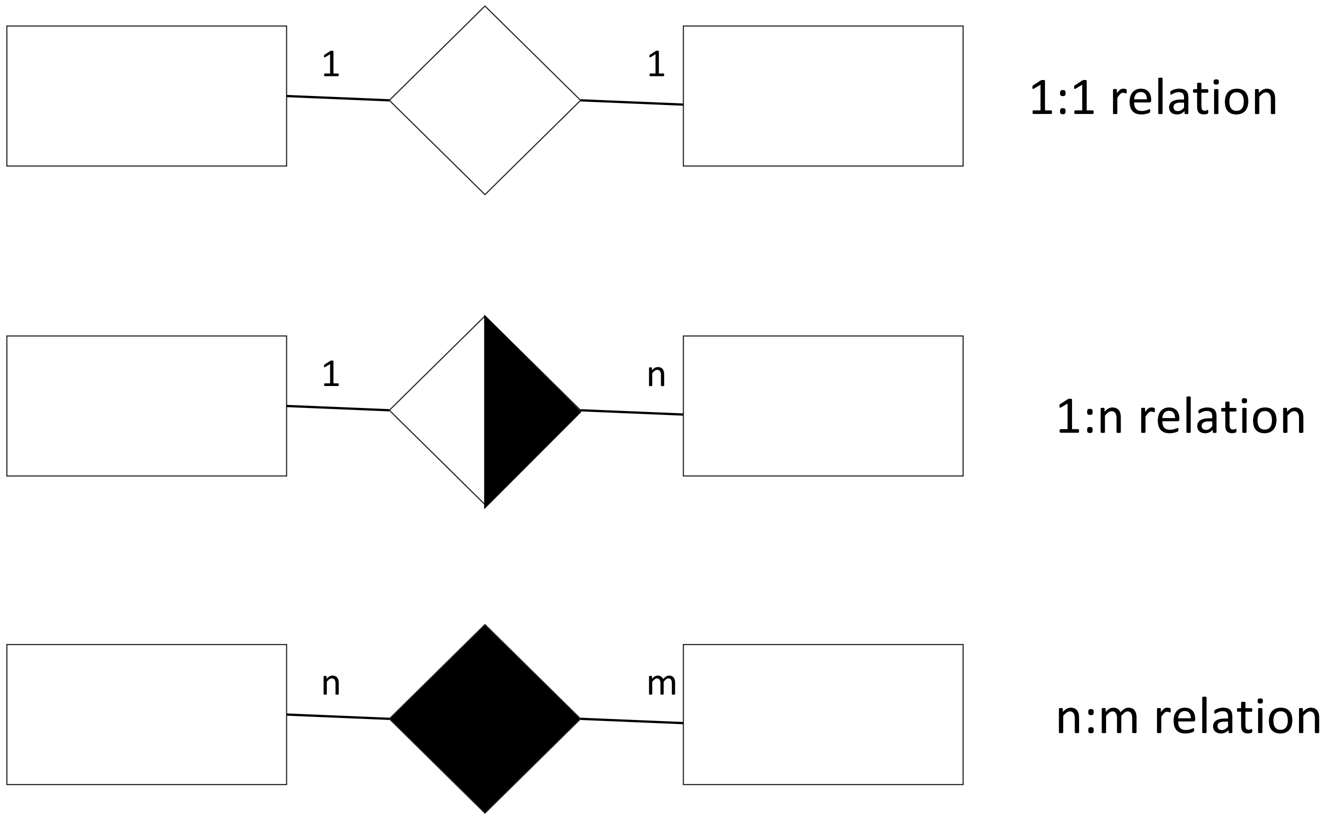
- **Relationships** among entities



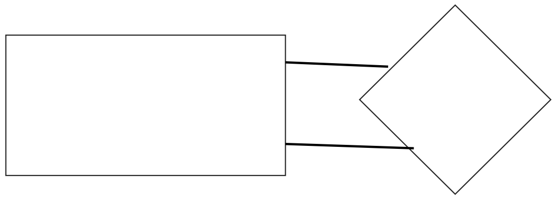
- **Attributes** store properties of entities



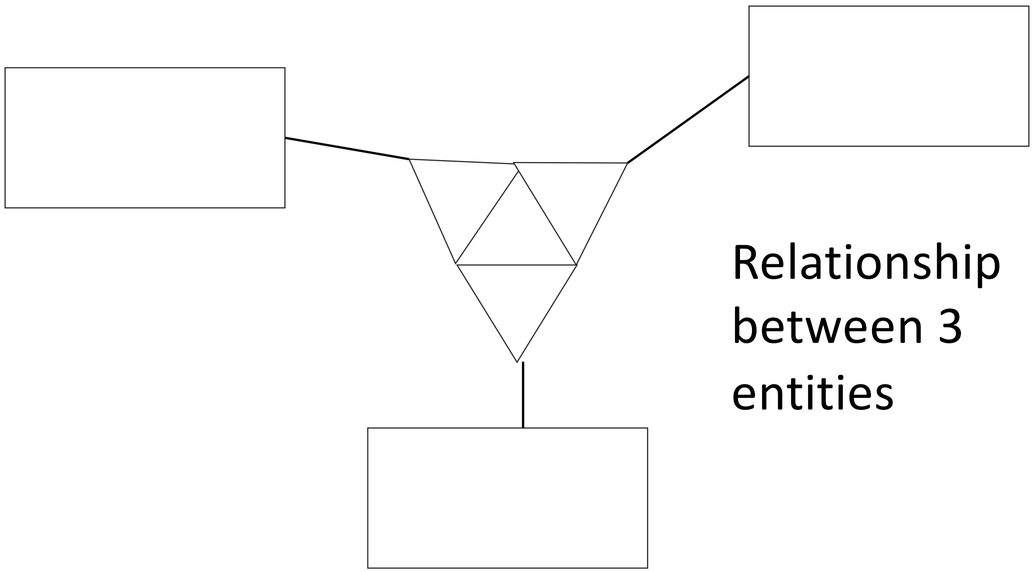
ER Diagram relationships



ER Diagram relationships



Reflexive relation



Relationship between 3 entities

Using ER-Diagrams

Let us consider the following text:

“A lecture has an unique number, a name, a type, and weekly hours. Each lecture is assigned to a lecture hall. Lecture halls have an unique name, an address, and a number of available seats. For each lecture we have a teacher. Students can apply for lectures.”

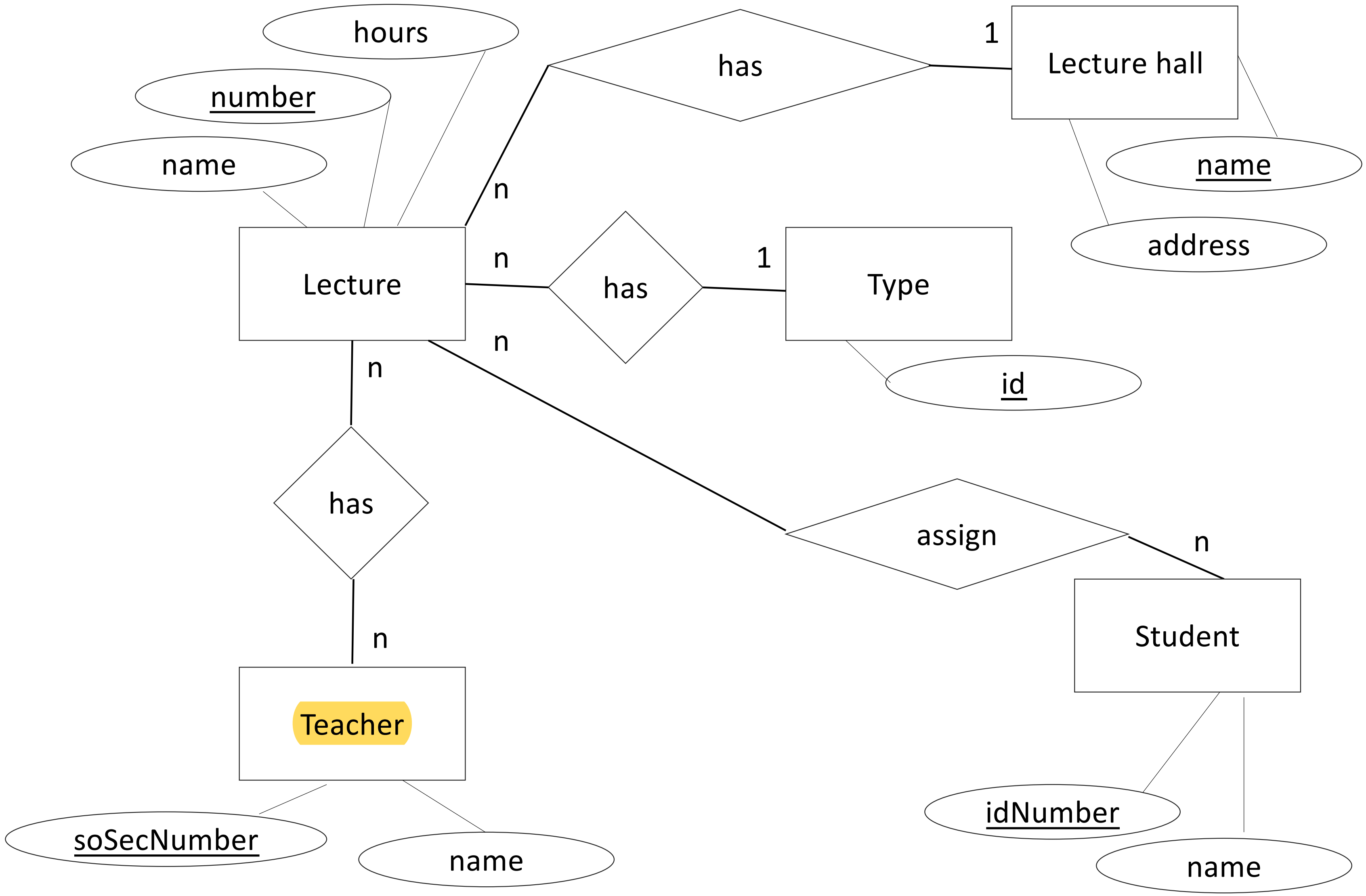
1. Step: Identify entities, their attributes and relationships (noun method)

*"A **lecture** has an unique number, a name, a **type**, and weekly hours. Each lecture is assigned to a **lecture hall**. Lecture halls have an unique name, an address, and a number of available seats. For each lecture there must be a **teacher**. **Students** can apply for lectures."*

- **Notes**:
  - Nouns are either entities or attributes
  - Verbs are usually capturing relationships

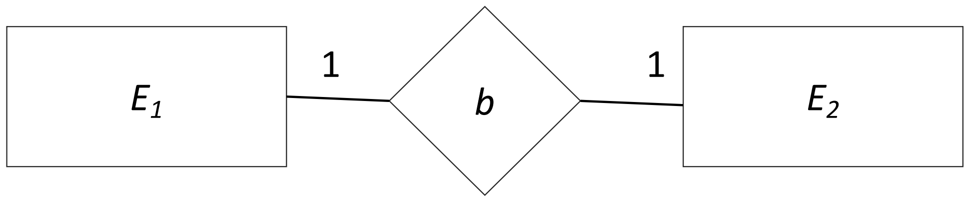
2. Step: First version of an ER Diagram

- We need further information!
  - Which attributes are unique?
  - Are there missing attributes?
  - Are there open questions regarding relationships or entities?
    - What type of relationship do we have? 1:1 or n:m?


hours
number
name
has
1
Lecture hall
name
address
n
n
Lecture
has
1
Type
n
n
id
has
assign
n
Student
n
Teacher
idNumber
soSecNumber
name
name

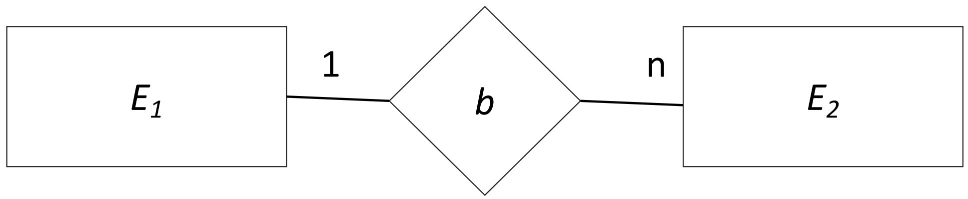
Converting ER Diagram into a scheme

- **Entity** E with attributes $A_1,..,A_n$:
  Relation $E(A_1,..,A_n)$ with a
  key $k \subseteq \{A_1,..,A_n\}$
- **1:1 relationship**: Extend one of the entities E_1 or E_2 with the key of the other.

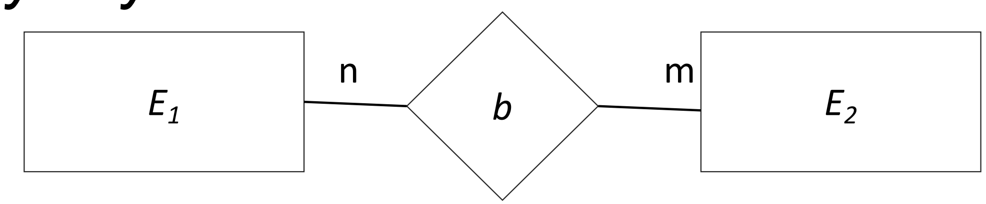


Converting ER Diagram into a scheme

- **1:n relationship**: Extend the relation E_2 with the key of E_1

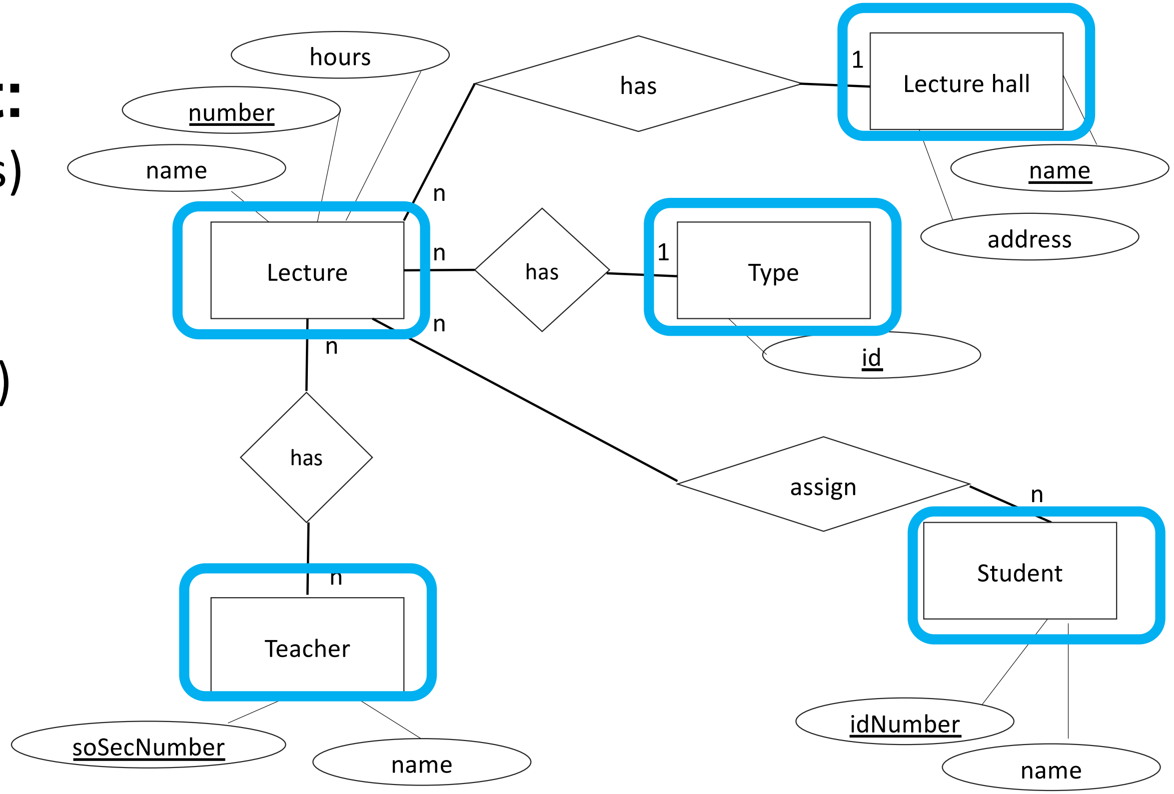


- **n:m relationship**: Introduce a new relation *b(...) having attributes from b* (if available) and both primary keys from E_1 and E_2 *as a primary key!*



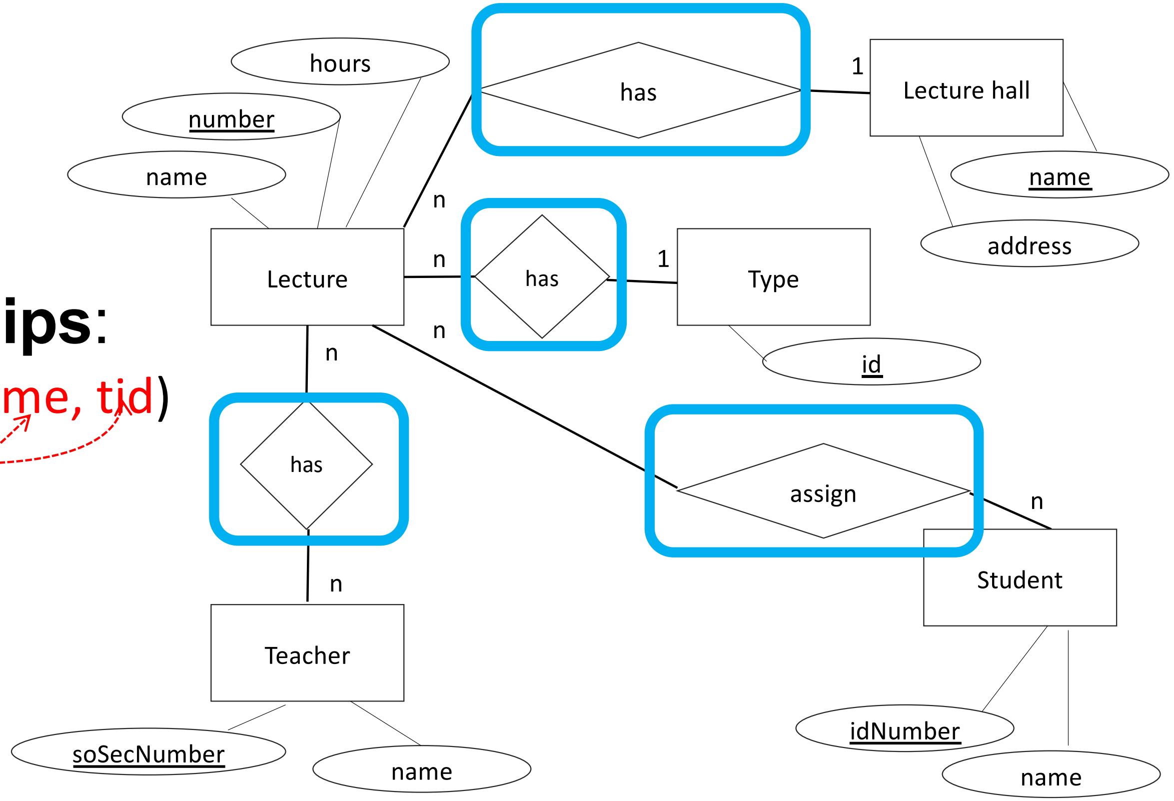
3. Step: Conversion into a DB schema

- **A – Convert entities first:**
  - Lecture(<u>number</u>, name, hours)
  - Type(<u>id</u>)
  - LectureHall(<u>name</u>, address)
  - Teacher(<u>soSecNumber</u>, name)
  - Student(<u>idNumber</u>, name)



3. Step: Conversion into a DB schema

- **B – Conversion of relationships**:
  - Lecture(<u>number</u>, name, hours, lhName, tid)
  - Type(<u>id</u>)
  - LectureHall(<u>name</u>, address)
  - Teacher(<u>soSecNumber</u>, name)
  - Student(<u>idNumber</u>, name)
  - AssignStudent(<u>number,idNumber</u>)
  - HasTeacher(<u>number,soSecNumber</u>)



4. Step – Normalization of relations using 1NF-3NF

- There might be other relations necessary
- For our example the result is in 3NF

- Note:
  - There are other normal forms (BCNF,…), which might be also applied here

SQL – The database language

SQL

- A standardized language almost all relational database have implemented
- Used for database and their tables
  - Create a lecture database:
    - `CREATE DATABASE lectureDB;`
- SQL has data types:

Data type	Notes
bool	Boolean values
int	4 byte integer value
real	Single precision real value (4 byte)
float8	Double precision real (8 byte)
char(N)	Fixed length character string of size N
varchar(N)	Variable length character string up to size N
date	The date
time	The time

SQL

- We can also create tables:

- For example: The relation Lecture(<u>number</u>, name, hours, lhName, tid) can be created in a database lectureDB using the following command:

```
CREATE TABLE lecture (
          number    char(6),
          name      varchar(80),
          hours     int,
          lh_name   varchar(10),
          t_id      char(2)
);
```

SQL

- Can also be used to add data to database tables

- **Example**:

```
INSERT INTO teacher
 VALUES (‘1234567890’,’Franz Wotawa’);
```

SQL

- However, the main purpose of SQL is to answer **QUERIES** to databases!

- **Example**: returning all students from the table `student`

```
SELECT * from students;
```

```
 idNumber  |  name
-----------+---------
 20140001 |  Lucas
 20140002 |  Adam
 20140003 |  Eva
 20140004 |  Sophie
 20140005 |  Marie
(5 rows)
```

SQL Select

- Let us consider the following table:

```
 number |          name            | hours      | lh_name |     t_id
--------+-------------------------+-----------+---------+------------
 117101 | "Software Maintenance" |          3 | i7       | VU
 117102 | "Compilerbau"          |          2 | i11      | VO
 117103 | "Compilerbau"          |          1 | i11      | KU
 117104 | "Seminar DB Dipl"      |          2 | i12      | SE
(4 rows)
```

SELECT **DISTINCT** name FROM lecture
ORDER BY name;

```
          name
------------------------
"Compilerbau"
"Seminar DB Dipl"
"Software Maintenance"
(3 rows)
```

SELECT name FROM lecture;
Implements the **projection** operation

```
          name
------------------------
"Software Maintenance"
"Compilerbau"
"Compilerbau"
"Seminar DB Dipl"
(4 rows)
```

SQL Select

- Let us consider the following table:

```
 number |          name            | hours      | lh_name |     t_id
--------+------------------------+-----------+---------+------------
 117101 | "Software Maintenance" |          3 | i7      | VU
 117102 | "Compilerbau"          |          2 | i11     | VO
 117103 | "Compilerbau"          |          1 | i11     | KU
 117104 | "Seminar DB Dipl"      |          2 | i12     | SE
(4 rows)
```

```
SELECT *
FROM lecture
WHERE hours > 2;
```

Implements the **selection** operation

```
 number |          name            | hours      | lh_name |     t_id
--------+------------------------+-----------+---------+------------
 117101 | "Software Maintenance" |          3 | i7      | VU
(1 row)
```

SQL Select

- We can also combine projection and selection
- The select statement can also be used to join tables
- There are more complicated selection operations possible, e.g., using Boolean operators like AND or OR

Notes on SQL

- We can also add primary keys using SQL
- We can delete data
- We can handle NULL values
- And there is much more…

Summary

- Databases provide the backbone of every information system
- There are different types of databases:
  - Semi-structured
  - Structured
    - Relational databases
    - Object-oriented databases
    - Hierarchical databases
- SQL is the common standard for relational database management systems to create or delete databases, add, modify or delete data, or to search for data using SELECT