Database Management Systems INFO 210

Summary - Part 1 (DBMS, ER, SQL)

Franz Wotawa

wotawa@ist.tugraz.at

1

Remarks – Assignment 4

- If your `createDB.sql` or `insertDB.sql` from Assignment 3 was not working, provide a new one as well!
- The bonus example has also to be provided this Friday!
- Please do not forget to test all files you send to me:
  - Try all the SQL statements or the files using the PostgreSQL shell!!!
  - Do not use the PostgreSQL dump file feature!

2

What we have discussed in our journey

Motivation

Information systems
Big data (volume, velocity, variety)

Relational DBs

- **Conceptual modeling & design** (ER diagrams)
- **DB theory** (relational algebra & calculus, functional dependencies, normal forms)
- **SQL** (modifying DB and data, querying)
- **Other concepts** (transactions, query management, ACID)

New directions of DB research and practice

- **NoSQL**

3

Motivation

- Why do we need databases?
- Why do we consider relational databases!

4

We Live in a World of Data

- Nearly 500 Exabytes per day are generated by the Large Hadron Collider experiments (not all recorded!)
- 2.9 million emails are sent every second
- 20 hours of video are uploaded to YouTube every minute
- 24 PBs of data are processed by Google every day
- 50 million tweets are generated per day
- 700 billion total minutes are spent on Facebook each month
- 72.9 items are ordered on Amazon every second

5

Data and *Big* Data

- The value of data as an organizational asset is widely recognized
- Data is literally exploding and is occurring along three main dimensions
  - "Volume" or the amount of data
  - "Velocity" or the speed of data
  - "Variety" or the range of data types and sources
- What is Big Data?
  - It is the proliferation of data that floods organizations on a daily basis
  - It is *high* volume, *high* velocity, and/or *high* variety information assets
  - It requires new forms of processing to enable *fast* mining, enhanced decision-making, insight discovery and process optimization

6

What Do We Do With Data and Big Data?

We want to do these *seamlessly* and *fast*...

7

Why Studying Databases?

- Data is *everywhere* and is *critical* to our lives
- Data need to be recorded, maintained, accessed and manipulated *correctly*, *securely*, *efficiently* and *effectively*
  - At the "low end": scramble to web-scale (a mess!)
  - At the "high end": scientific applications
- Database management systems (DBMSs) are indispensable software for achieving such goals
- The principles and practices of DBMSs are now an integral part of computer science curricula
  - They encompass OS, languages, theory, AI, multimedia, and logic, among others

As such, the study of database systems can prove to be richly rewarding in more ways than one!

8

Learning Outcomes

❖ After finishing this course you will be able to:

1. Describe a wide range of data involved in real-world organizations using the entity-relationship (ER) data model
2. Explain how to translate an ER diagram into a relational database
3. Analyze and apply a formal query language, relational calculus and algebra
4. Indicate how SQL builds upon relational calculus and algebra and effectively apply SQL to create, query and manipulate relational databases
5. Design and develop multi-tiered, full-fledged standalone and web-based applications with back-end databases
6. Appreciate how DBMSs create, manipulate and manage files of fixed-length and variable-length records on disks

9

Data Base Management Systems

- A special software is accordingly needed to make the preceding tasks easier
- This software is known as Data Base Management System (DBMS)
- DBMSs provide automatic:
  - Data independence
  - Efficient data access
  - Data integrity and security
  - Data administration
  - Concurrent access and crash recovery
  - Reduced application development and tuning time

10

Some Definitions

- A database is a collection of data which describes one or many real-world enterprises
  - E.g., a university database might contain information about entities like students and courses, and relationships like a student enrollment in a course
- A DBMS is a software package designed to store and manage databases
  - E.g., DB2, Oracle, MS SQL Server, MySQL and Postgres
- A database system = (Big) Data + DBMS + Application Programs

11

Data Models

- The user of a DBMS is ultimately concerned with some real-world enterprises (e.g., a University)
- The data to be stored and managed by a DBMS *describes* various aspects of the enterprises
  - E.g., The data in a university database describes students, faculty and courses entities and the relationships among them
- A data model is a collection of high-level data description constructs that hide many low-level storage details
- A widely used data model called the entity-relationship (ER) model allows users to pictorially denote entities and the relationships among them

12

The Relational Model

- The relational model of data is one of the most widely used models today
- The central data description construct in the relational model is the relation
- A relation is basically a table (or a set) with rows (or records or tuples) and columns (or fields or attributes)
- Every relation has a schema, which describes the columns of a relation
- Conditions that records in a relation must satisfy can be specified
  - These are referred to as integrity constraints

13

The Relational Model: An Example

- Let us consider the student entity in a university database

Students Schema

Students(<u>*sid*</u>: string, *name*: string, *login*: string, *dob*: string, *gpa*: real)

An attribute, field or column

Integrity Constraint: Every student has a unique *sid* value

A record, tuple or row

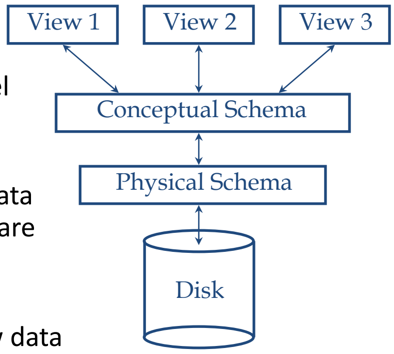
sid	***name***	***login***	***dob***	***gpa***
512412	Khaled	khaled@lzh.edu	18-9-1995	3.5
512311	Jones	jones@lzh.edu	1-12-1994	3.2
512111	Maria	maria@lzh.edu	3-8-1995	3.85

An *instance* of a Students *relation*

14

Levels of Abstraction

- The data in a DBMS is described at three levels of abstraction, the conceptual (or logical), physical and external schemas
- The conceptual schema describes data in terms of a specific data model (e.g., the relational model of data)
- The physical schema specifies how data described in the conceptual schema are stored on secondary storage devices
- The external schema (or views) allow data access to be customized at the level of individual users or group of users (views can be 1 or many)



15

Queries in a DBMS

- The ease with which information can be queried from a database determines its value to users
- A DBMS provides a specialized language, called the query language, in which queries can be posed
- The relational model supports powerful query languages
  - Relational calculus: a formal language based on mathematical logic
  - Relational algebra: a formal language based on a collection of operators (e.g., selection and projection) for manipulating relations
  - Structured Query Language (SQL):
    - Builds upon relational calculus and algebra
    - Allows creating, manipulating and querying relational databases
    - Can be embedded within a host language (e.g., Java)

16

Concurrent Execution and Transactions

- An important task of a DBMS is to *schedule* concurrent accesses to data so as to improve performance

T1	T2
	R(A) W(A)
R(B) W(B)	
	R(C) W(C)

An *atomic* sequence of database actions (read/writes) is referred to as *"transaction"*

- When several users access a database *concurrently*, the DBMS must order their requests carefully to avoid conflicts
  - E.g., A check might be cleared while account balance is being computed!
- DBMS ensures that conflicts do not arise via using a locking protocol
  - Shared vs. Exclusive locks

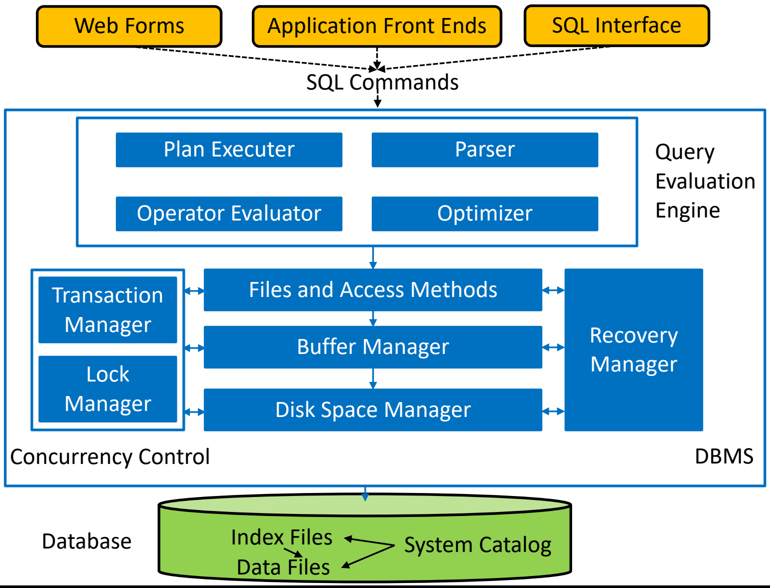
17

Ensuring Atomicity

- Transactions can be interrupted before running to completion for a variety of reasons (e.g., due to a system crash)
- DBMS ensures atomicity (all-or-nothing property) even if a crash occurs in the middle of a transaction
- This is achieved via maintaining a log (i.e., history) of all writes to the database
  - *Before* a change is made to the database, the corresponding log entry is forced to a safe location (this protocol is called Write-Ahead Log or WAL)
  - After a crash, the effects of partially executed transactions are *undone* using the log

18

The Architecture of a Relational DBMS

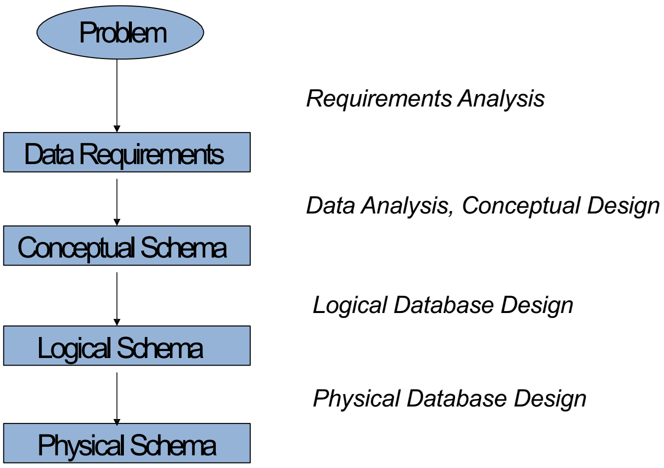


19

CONCEPTUAL DESIGN

20

Database Design Goes Through Stages



2
1

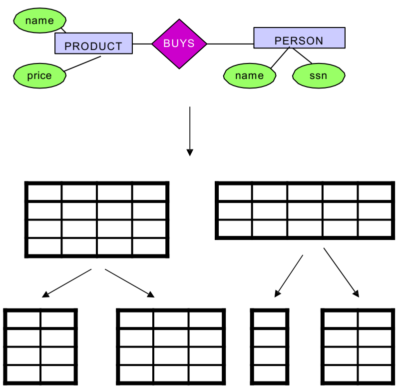
21

Conceptual and Logic Design

Conceptual Model:

Relational Model:
(plus functional
dependencies)

Normalization:



2
2

22

Conceptual Design with the ER Model

Questions to ask:

- What are the *entities* (= objects, individuals) in the organization?
- Which *relationships* exist among the entities?
- What information (= *attributes*) do we want to store about these entities and relationships?
- What are the business rules of the organization?
- Which integrity constraints do arise from them?

The answers are represented in an
Entity Relationship Diagram (ER diagram)

23

23

Entities and Entity Set Types

Entity: An object distinguishable from other objects (e.g., an employee)

- An entity is described by a set of attributes.

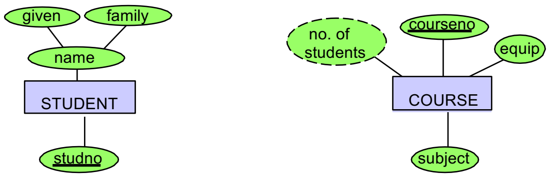
  Examples of entities?
  Examples of things that are not entities?

Entity Set/Entity Type: A collection of similar entities (e.g., all employees)

- All entities in an entity set have the same set of *attributes*.
- Each attribute has a *domain*.
- Each entity set has a *key* (i.e., one or more attributes whose values uniquely identify an entity) 9

24

Graphical Representation of Entity Sets



- Entity Sets are drawn as rectangles
- Attributes are drawn using ovals
- Simple attributes contain atomic values
- Composite attributes combine two or more attributes
- Derived attributes are indicated by dashed lines
- The attributes making up the key are underlined

25

Relationships and Relationships Set/Types

Relationship: An association between two or more entities (e.g., "Joe Smith" is "enrolled" in "CS123")

- Relationships may have attributes

Examples of relationships?

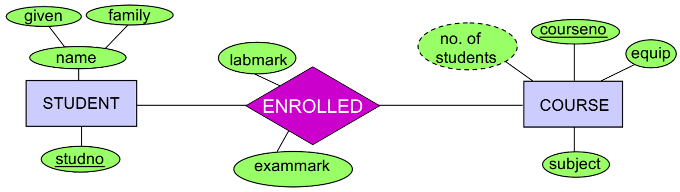
Relationship Set/Type: A collection of similar relationships

- An n-ary relationship type relates n entity types $E_1,\ldots,E_n$
- Each relationship involves n entities $e_1 \in E_1,\ldots,e_n \in E_n$

Examples of relationship sets?

26

Graphical Representation of Relationship Types



- Relationship sets are drawn as diamonds

How many labmarks can a student have?

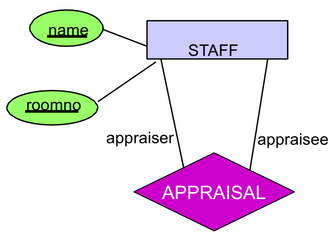
2

27

Roles and Recursive Relationships

An entity type can
- participate in several relationship sets

and
- participate more than once in one relationship set (taking on different "roles")



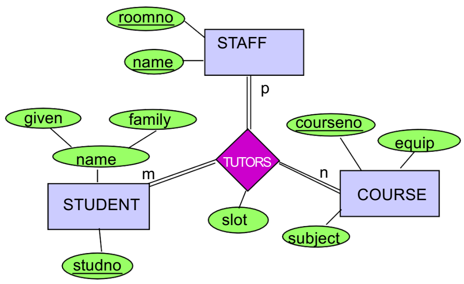
Which are other examples of recursive relationships?

2

28

Multiway (non-binary) Relationship

Relationships can involve more than two entity types…

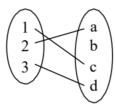


2

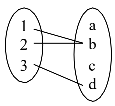
29

Multiplicity/Cardinality of Relationship Types

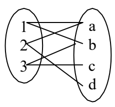
- one-one:

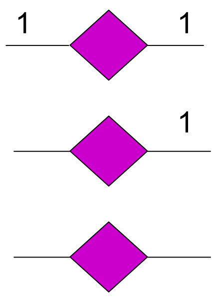


- many-one:



- many-many:





Sometimes the letters m, n are used to indicate the "many" side of relationships.

16

30

Participation Constraints

- Participation constraints specify whether or not an entity must participate in a relationship set
- When there is no participation constraint, it is possible that an entity will not participate in a relationship set
- When there is a participation constraint, the entity must participate *at least once*
- Participation constraints are drawn using a *double line* from the entity set to the relationship set

31

Summary: Properties of Relationship Types

Degree

- The number of participating entity types

Cardinality ratios

- The number of instances of each of the participating entity types which can partake in a single instance of the relationship type:
  1:1, 1:many, many:1, many:many

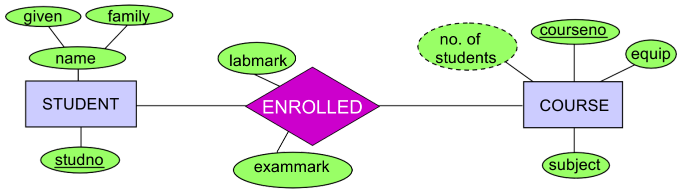
Participation

- Whether an entity instance has to participate in a relationship instance
- Represented with a double line

32

Attributes in ER Modeling

- For every attribute we define
  - *Domain* or *data type*
  - *Format*, *i.e.*, composite or atomic
  - whether it is derived
- Every entity type must have as *key* an attribute or a set of attributes



33

Constraints: Definition

- A constraint is an assertion about the database that must be true at all times
- Constraints are part of the database schema

34

Modeling Constraints

Finding constraints is part of the modeling process.
They reflect facts that hold in the world or
business rules of an organization.

Examples:

Keys: *social security number* uniquely identifies a person

Single-value constraints: a person can have only one father

Referential integrity constraints: if you work for a company,
it must exist in the database

Domain constraints: peoples' ages are between 0 and 150

Cardinality constraints: at most 100 students enroll in a course

3

35

Keys

A key is a set of attributes that
uniquely identify an object or entity:

Person: social-security-number (U.S.)
national insurance number (U.K.)
codice fiscale (Italy)
name
name + address
name + address + dob

(Why not "age"?)

Perfect keys are often hard to find,
so organizations usually invent something.

32

36

Variants of Keys

- Multi-attribute (composite) keys:
  – E.g. name + address

- Multiple keys:
  – E.g. social-security-number, name + address

37

Existence Constraints

Sometimes, the existence of an entity of type X
depends on the existence of an entity of type Y:

Examples:
- Book chapters presume the existence of a book
- Tracks on a CD presume the existence of the CD
- Orders depend on the existence of a customer

We call Y the *dominating* entity type and
X the *subordinate* type
⇒ strong and weak entities

38

Strong and Weak Entities

Dominating and subordinate types are modeled as
- Entities

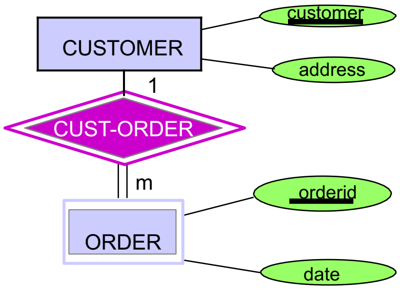
(also "strong", or "identifying" entities)

and
- Weak entities

Identifying entity

Supporting, or identifying relationship

Weak entity

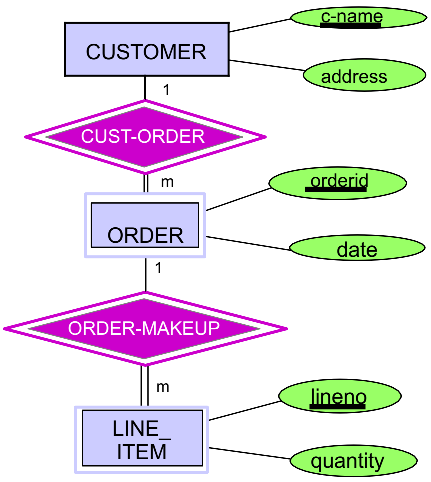


Weak Entities May Depend on Other Weak Entities

- Strong entity type
- Identifying entity for ORDER
- Identifying entity for LINE_ITEM

- Weak entity
- Identifying entity for LINE_ITEM

- Weak entity type



Design Principles for ER Modeling

There are usually several ways to model a real world concept, e.g.:

- entity vs. attribute
- entity vs. relationship
- binary vs. ternary relationships, etc.

Design choices can have an impact on

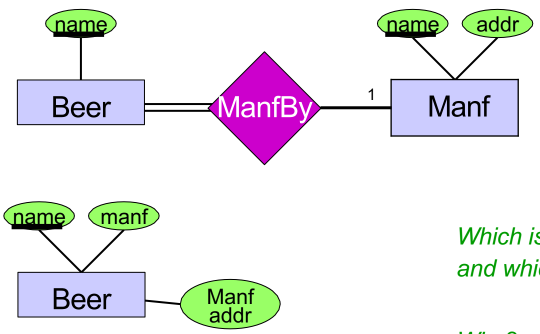
- redundancies among the data that we store
- integrity constraints captured by the database structure

41

"Don't Say the Same Thing More Than Once"

Redundancy wastes space and encourages inconsistency

Example:



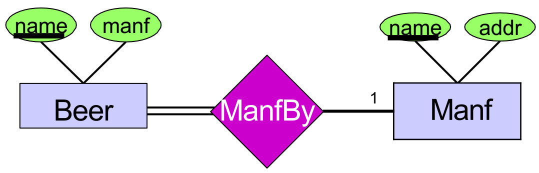
Which is good design, and which is bad?

Why?

42

42

And What About This?

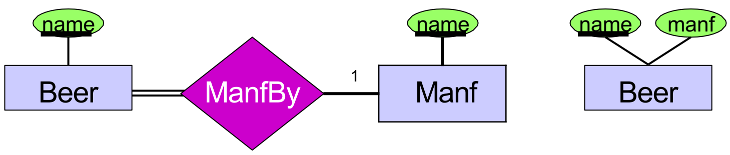


43

Entities Vs. Attributes

Sometime it is not clear
- which concepts are worthy of being entities, and
- which are handled more simply as attributes

Example:
Which are the pros and cons of each of
the two designs below?



44

Entity Vs. Attribute: Rules of Thumb

Only make an entity if either:

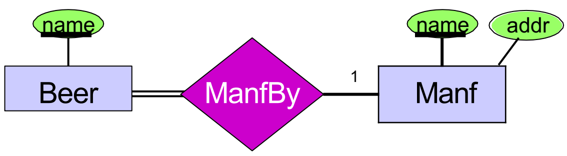
1. It is more than a name of something; *i.e.*,
   it has non-key attributes or relationships with a number of different entities, or

2. It is the "many" in a many-one relationship

45

Entity Vs. Attribute: Example

- The following design illustrates both points:



- *Manfs* deserves to be an entity because we record *addr*, a non-key attribute
- *Beers* deserves to be an entity because it is at the "many" end
- If not, we would have to make "set of beers" an attribute of *Manfs*

46

Hints for ER Modelling

- Identify entity types by searching for nouns and noun phrases
- Assume all entities are strong and check for weak ones on a later pass
- You need an identifier for each strong entity
- Assume all relationships have optional participation and check for mandatory (total) ones on a later pass
- Expect to keep changing your mind about whether things are entities, relationships or attributes
- Keep the level of detail relevant and consistent
  (for example leave out attributes at first)
- Approach diagram through different views …
  … and merge them

47

Use the Schema to Enforce Constraints

- The conceptual *schema* should enforce as many constraints as possible
- Don't rely on future data to follow assumptions

Example:

If the university wants to associate only one instructor with a course,

- don't allow sets of instructors and
- don’t count on departments to enter only one instructor per course

48

Converting ER diagrams into schemas

- Have a look at Moodle (Supplemental material)

TOBY J. TEOREY, DONGQING YANG, and JAMES P. FRY, A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model, ACM, Computing Surveys, Vol. 18, No. 2, June 1986.

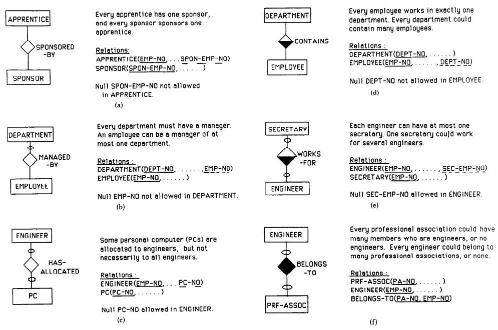
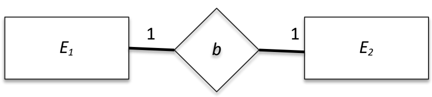


Figure 8. Binary relationship transformation rules.

49

Converting ER Diagram into a scheme

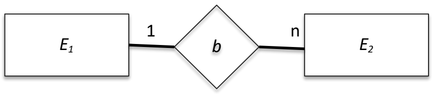
- **Entity** *E* with attributes $A_1,..,A_n$:
  Relation $E(A_1,..,A_n)$ with a
  key $k \subseteq \{A_1,..,A_n\}$
- **1:1 relationship**: Extend one of the entities E_1 or E_2 with the key of the other.



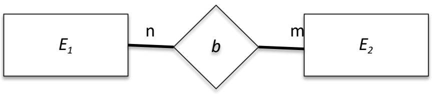
50

Converting ER Diagram into a scheme

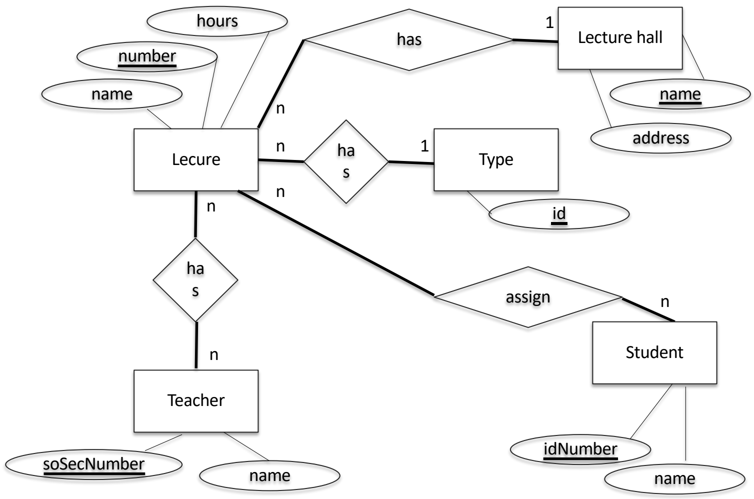
- **1:n relationship**: Extend the relation E_2 with the key of E_1



- **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!*



51

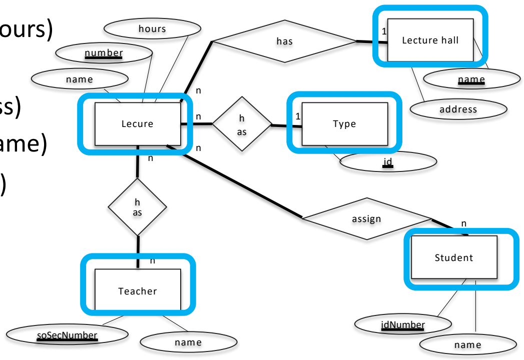


52

3. Step: Conversion into a DB schema

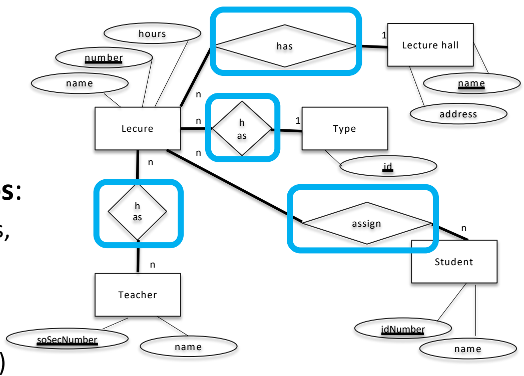
A – Convert entities first:

- 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)



53

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>)

54

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

55