INDUSTRIAL ENGINEERING AND BUSINESS INFORMATION SYSTEMS (IEBIS)
FACULTY OF BEHAVIOURAL, MANAGEMENT AND SOCIAL SCIENCES

UNIVERSITY
OF TWENTE.

INTRODUCTION TO DATABASES

DR. DANIEL BRAUN

OUTLINE

- INTRODUCTION
- TYPES OF DATABASES
- DESIGN AND MODELLING
- NORMALIZATION

What is a database? What do we need it for?

1 INTRODUCTION

PERSISTING DATA

- Computer programs store the value of variables in the RAM
- When the program is closed (or the power is switched off), the data is lost
- To keep data between sessions, it must be persisted (= saved in a permanent storage)

Filesystem:

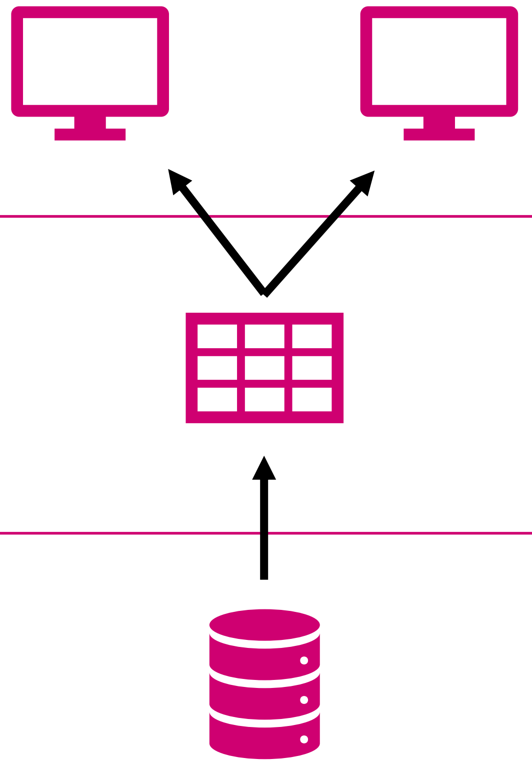
- We can store the data in a file
- However, this comes with problem, at least for larger business information systems:
  - Limited support for multiple users
  - Inconsistencies
  - Data loss
  - …

DATABASE MANAGEMENT SYSTEM (DBMS)

- A **database** is an organized collection of structured information
- A **database management system (DBMS)** is a software for creating and managing databases
- Properties we usually expect from DBMS:
  - **Data independence (physical and logical)**
  - **Data integrity**
  - **Query language**
  - Concurrency
  - Scalability
  - Multi-user support
  - Efficiency
  - Reliability
  - …
- Examples of DBMS: MySQL, Microsoft Access, MongoDB, H2, …

DATA INDEPENDENCE

How does the user see the data?	External Level
How is the data structured?	Logical Level
How is the data stored?	Physical Level

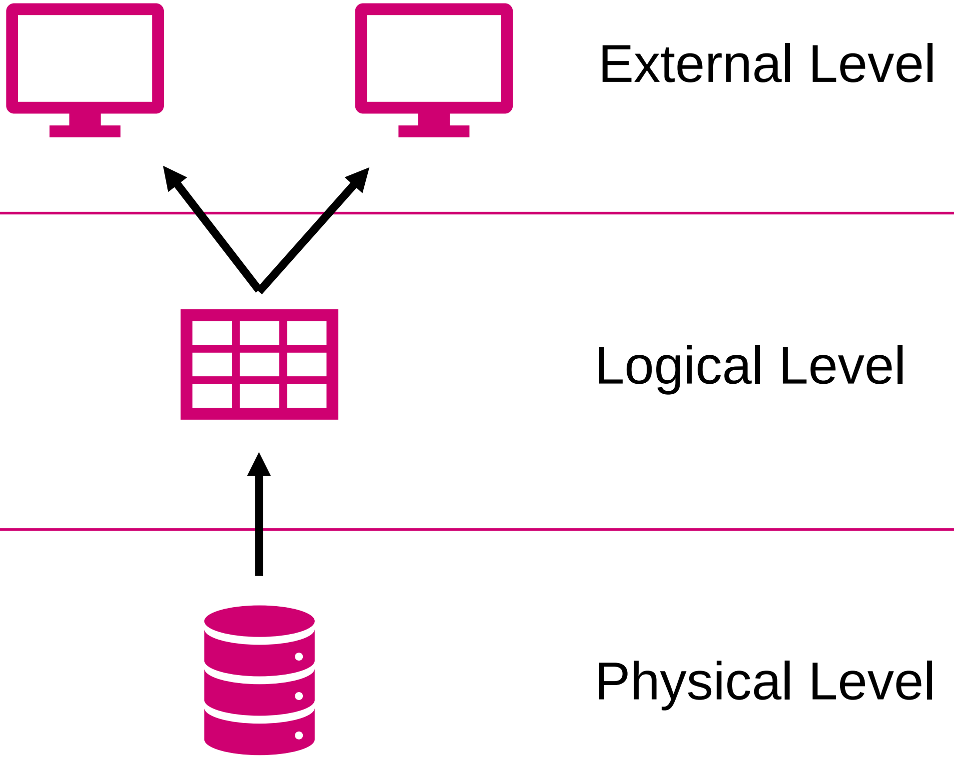


DATA INDEPENDENCE

Name	Email	Address
Alice Hopper	alice@utwente.nl	Hallenweg 17, 7522NH Enschede
Bob Turing	bob@utwente.nl	Hallenweg 17, 7522NH Enschede

Name	Email	Address
Alice Hopper	alice@utwente.nl	Hallenweg 17, 7522NH Enschede
Bob Turing	bob@utwente.nl	Hallenweg 17, 7522NH Enschede

```
Alice Hopper;alice@utwente.nl;Hallenweg 17, 7522NH Enschede
Bob Turing;bob@utwente.nl;Hallenweg 17, 7522NH Enschede
```



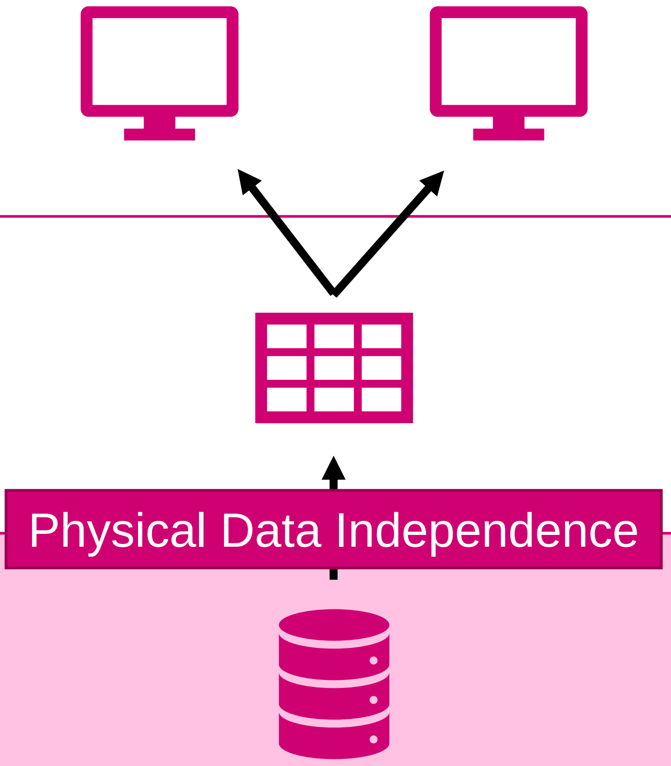
DATA INDEPENDENCE

Name	Email	Address
Alice Hopper	alice@utwente.nl	Hallenweg 17, 7522NH Enschede
Bob Turing	bob@utwente.nl	Hallenweg 17, 7522NH Enschede

External Level

Name	Email	Address
Alice Hopper	alice@utwente.nl	Hallenweg 17, 7522NH Enschede
Bob Turing	bob@utwente.nl	Hallenweg 17, 7522NH Enschede

Logical Level
Unaffected




```
1;Alice Hopper
2;Bob Turing
```

```
1;alice@utwente.nl
2;bob@utwente.nl
```

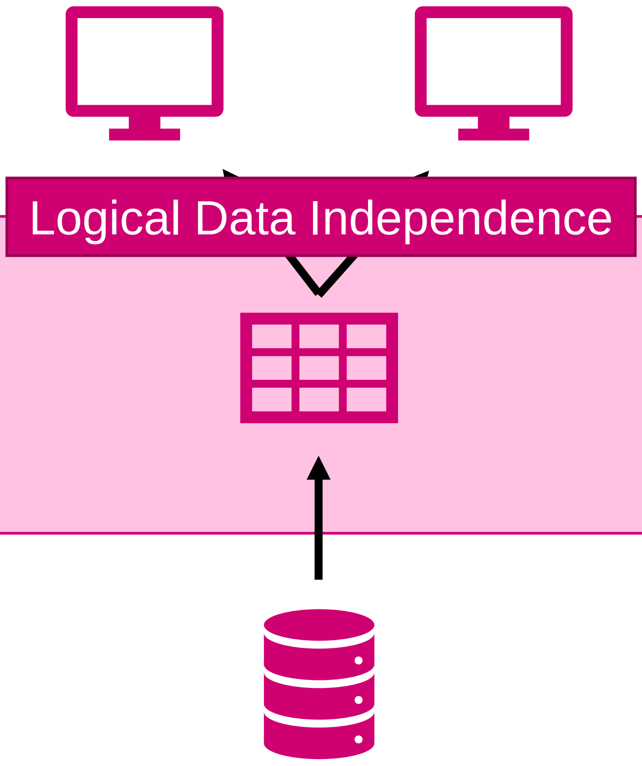
```
1;Hallenweg 17, 7522NH Enschede
2;Hallenweg 17, 7522NH Enschede
```

Physical Level
Changed

DATA INDEPENDENCE

Name	Email	Address
Alice Hopper	alice@utwente.nl	Hallenweg 17, 7522NH Enschede
Bob Turing	bob@utwente.nl	Hallenweg 17, 7522NH Enschede

External Level
Unaffected



Firstname	Lastname	Email	Address
Alice	Hopper	alice@utwente.nl	Hallenweg 17, 7522NH Enschede
Bob	Turing	bob@utwente.nl	Hallenweg 17, 7522NH Enschede

Logical Level
Changed

```
1;Alice;Hopper
2;Bob;Turing
```

```
1;alice@utwente.nl
2;bob@utwente.nl
```

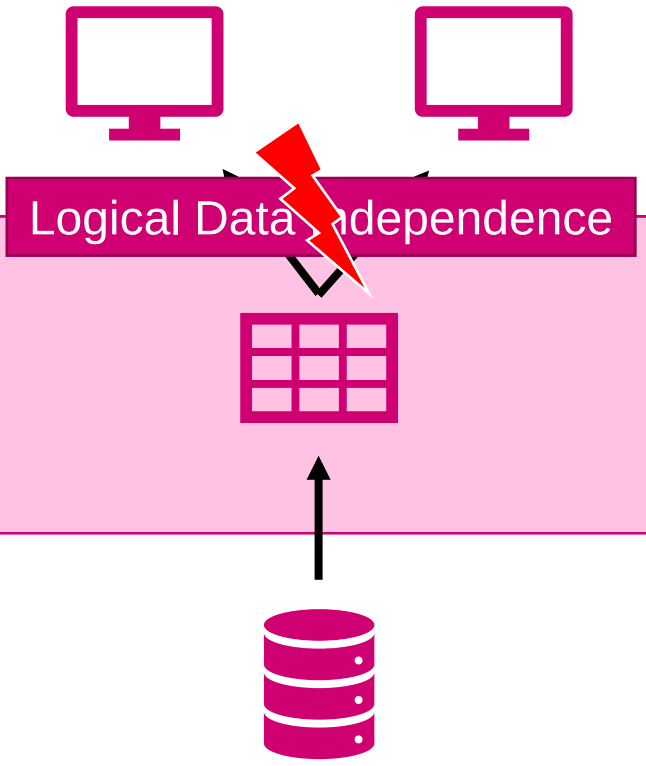
```
1;Hallenweg 17, 7522NH Enschede
2;Hallenweg 17, 7522NH Enschede
```

Physical Level
Changed

DATA INDEPENDENCE

Name	[redacted]	Address
Alice Hopper	[redacted]	Hallenweg 17, 7522NH Enschede
Bob Turing	[redacted]	Hallenweg 17, 7522NH Enschede

External Level
Affected



Firstname	Lastname	[redacted]	Address
Alice	Hopper	[redacted]	Hallenweg 17, 7522NH Enschede
Bob	Turing	[redacted]	Hallenweg 17, 7522NH Enschede

Logical Level
Changed

```
1;Alice;Hopper
2;Bob;Turing
```

```
1;Hallenweg 17, 7522NH Enschede
2;Hallenweg 17, 7522NH Enschede
```

Physical Level
Changed

DATA INTEGRITY

- **Data Integrity** is a property whereby data is guaranteed to be accurate, complete, and consistent over its whole life-cycle.
- Examples of data integrity checks:
  - A birthdate field may only contain date values (domain integrity)
  - References to other data is kept up-to-date (referential integrity)
- Not easy to implement, but straightforward… in normal operation
- What if things go wrong? Loss of internet connection, power outage, software crash, …

SCENARIO: MONEY TRANSFER

Transfer 500€ from account A to account B:

1. Read balance of account A
2. Check if balance of account A >= 500
3. Subtract 500 from balance of account A
4. Write new balance of account A
5. Read balance of account B
6. Add 500 to balance of account B
7. Write new balance of account B

SCENARIO: MONEY TRANSFER

Transfer 500€ from account A to account B:

1. Read balance of account A
2. Check if balance of account A >= 500
3. Subtract 500 from balance of account A
4. Write new balance of account A
5. Read balance of account B
6. Add 500 to balance of account B
7. Write new balance of account B

TRANSACTION

- A (database) **transaction** is a single logical unit of work, that can consist of multiple operations
- Transactions must fulfill the **ACID** properties, i.e., they must be:
  - Atomic
  - Consistent
  - Isolated
  - Durable

ACID PROPERTIES

- **Atomic:** Each transaction is either completely executed or everything stays unchanged
- **Consistent:** Each transaction brings the database from one valid state to a new one
- **Isolated:** Concurrent transactions do not influence each other
- **Durable:** Once a transaction is completed, the change is permanent, even in case of a system failure (usually means the result is recorded in non-volatile memory)

General note: Not all DBMS guarantee these (or other integrity) properties

UNIVERSITY OF TWENTE.

2 TYPES OF DATABASES

TYPES OF DATABASES

- **Relational databases**
- Object-oriented databases
- XML databases
- Hierarchical databases
- NoSQL databases

RELATIONAL DATABASES

- In a **relational database**, data is stored in a set of tables that can be connected with each other. We call them **relations**.
- The number of attributes (columns) in a relation (table) is called the **arity**.
- The number of tuples (rows) in a relation (table) is called **cardinality**.
- Every relation has a **primary key**, which distinctly identifies each element in the relation. The primary key can consist of multiple attributes.
- Connections between tuples are established by referencing the primary key of another tuple. If a tuple contains the primary key (PK) of another relation, this key is called a **foreign key** (FK).

Adaption of: Florian Matthes. Software Engineering for Business Applications - Bachelor's Course. TU Munich, 2021.

RELATIONAL DATABASES

PK PK

Purchase

CustomerId	Ean
1	5449…
2	5449…

FK FK

PK

Customer

CustomerId	FirstName	LastName
1	Alice	Hopper
2	Bob	Turing

PK

Product

Ean	Name
5449…	Coca Cola Vanilla
…	…

NOSQL DATABASES

- **NoSQL databases** (non-SQL, non-relational, not only SQL) are databases that use non-relational data models
- In comparison to relational databases, they usually are
  - more scalable
  - schemaless or at least only use a weak schema
  - provide easy data replication
- Most systems favor performance over consistency (no ACID transactions)
- NoSQL is an umbrella term for many different technologies, like:
  - Key-value databases (e.g., ArangoDB)
  - Document-oriented databases (e.g., MongoDB and CouchDB)
  - Wide-column databases (e.g., Bigtable)
- Backbone of the “Web 2.0”

RELATIONAL VS. NOSQL

- Relational databases are more efficient for frequent but small transactions and mostly read-transactions
- NoSQL databases are more efficient for large numbers of read and write requests with large payload
- Relational databases offer stronger consistency; there is middleware for NoSQL systems to support ACID transaction (e.g., CloudTPS)
- NoSQL databases are more scalable and can more easily offer redundancy

RELATIONAL VS. NOSQL

You build a new social media network for students of the UT,
where they can share pictures of their assignments and solutions.

What kind of database do you use?

A: Relational

B: NoSQL

RELATIONAL VS. NOSQL

You build an auction platform, on which
lecturers can sell exam solutions to the highest bidder.

What kind of database do you use?

A: Relational

B: NoSQL

RELATIONAL VS. NOSQL

A BI Application that helps a chain of stores to understand, which product sells best in a specific season?

What kind of database do you use?

A: Relational

B: NoSQL

RELATIONAL VS. NOSQL

- BI Applications often:
  - Perform mostly read transactions
  - Have a fixed and limited set of users
  - Use transactions with small payloads
  - Benefit from the power of SQL-queries

=> In this course, we will work with relational databases

3 DESIGN AND MODELLING

DATABASE DESIGN

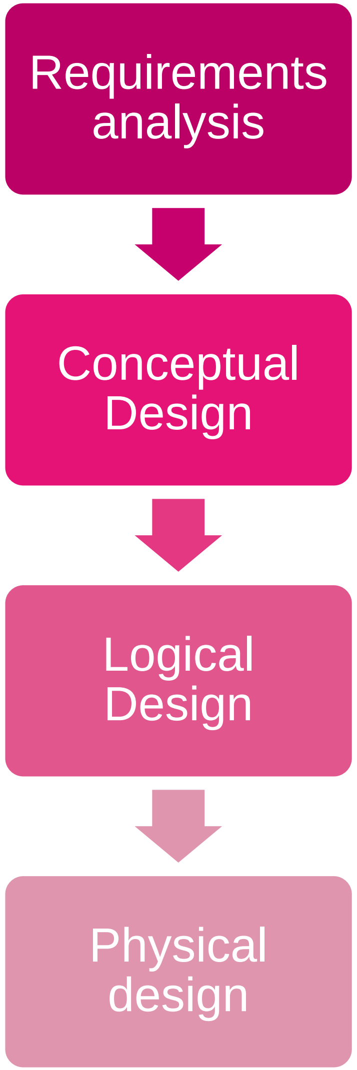
- For most problems (e.g., handling purchase data), there is almost an infinite number of possible database designs to handle the problem:
  - You can use different types of databases (Relational, NoSQL)
  - Even for the same type of database, you can use different DBMS (MySQL, H2, …)
  - Data can be represented in different ways

Customer		
CustomerId	FirstName	LastName
1	Alice	Hopper
2	Bob	Turing

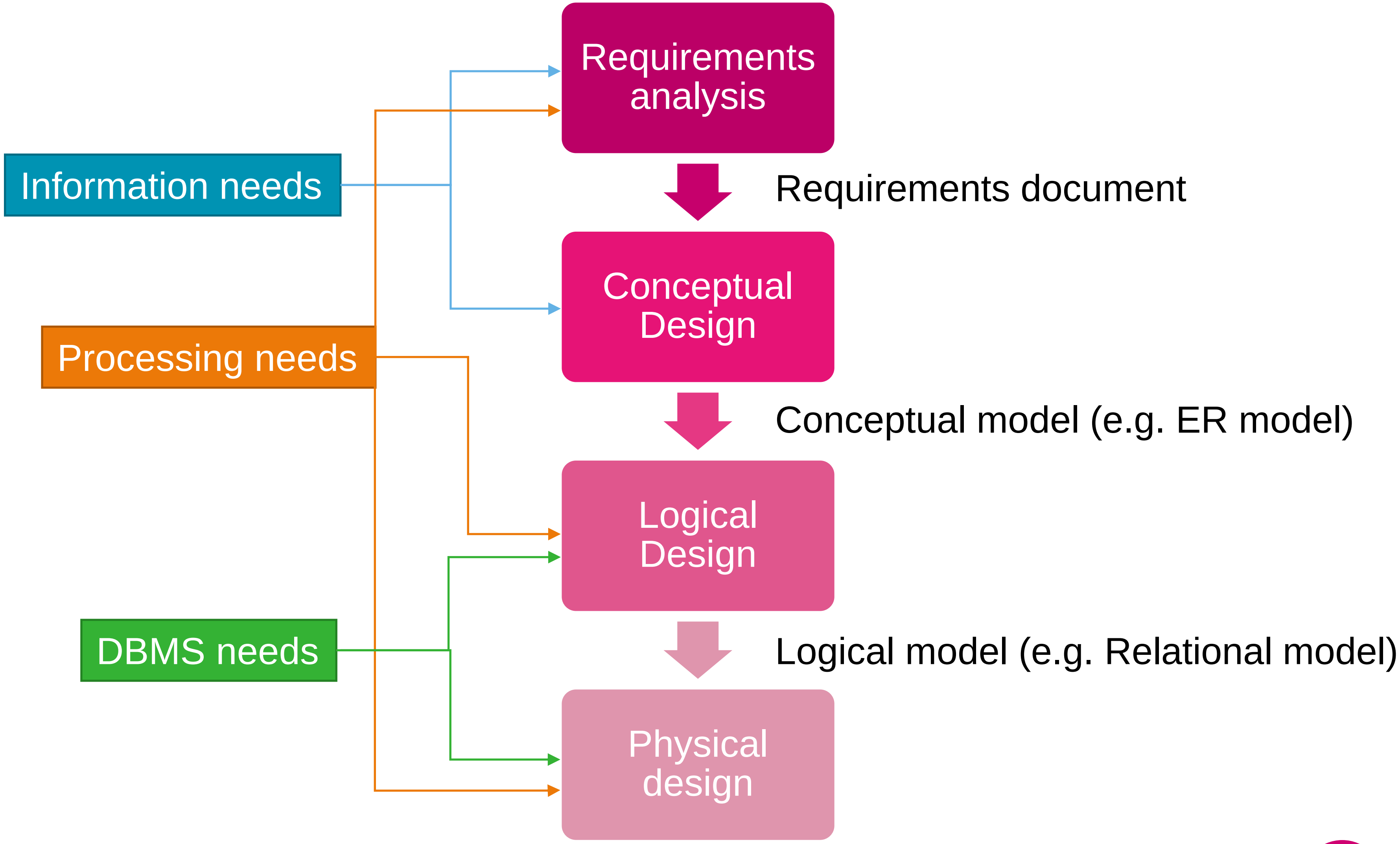
Customer	
CustomerId	Name
1	Alice Hopper
2	Bob Turing

  - ….
- While a lot of solutions might work, some are betters than other
- During database design, we try to identify a good solution

DATABASE DESIGN STAGES



DATABASE DESIGN STAGES



UNIVERSITY OF TWENTE.

REQUIREMENTS ANALYSIS

- **Requirements analysis** in Software Engineering, is the process of identifying the needs and expectations of stakeholders towards a software project
- It includes various tasks, like:
  - Stakeholder analysis
  - Identification of tasks that should be supported
  - Formalization of requirements
  - …
- Not a BI or DB specific task
- But important for us: identification of information and processing needs

REQUIREMENTS ANALYSIS

- A simple technique that can help to identify relevant information in requirements texts: noun-verb analysis (a.k.a. Abbott's technique, Abbott 1983)
- Not originally designed for database design, but still applicable:
  - A noun can be a hint for an entity or attribute [object / class]
  - A verb can be a hint for a relationship [method]
- Example:

"As a module coordinator, I want to be able to assign a student grades for each course in the module and the module itself."

Abbott RJ (1983) Program Design by Informal English Descriptions. Communications of the ACM 26(11):882–894

REQUIREMENTS ANALYSIS

- A simple technique that can help to identify relevant information in requirements texts: noun-verb analysis (a.k.a. Abbott's technique, Abbott 1983)
- Not originally designed for database design, but still applicable:
  - A noun can be a hint for an entity or attribute [object / class]
  - A verb can be a hint for a relationship [method]
- Example:

"As a module coordinator, I want to be able to assign a student grades for each course in the module and the module itself."

Abbott RJ (1983) Program Design by Informal English Descriptions. Communications of the ACM 26(11):882–894

UNIVERSITY OF TWENTE.

CONCEPTUAL DESIGN

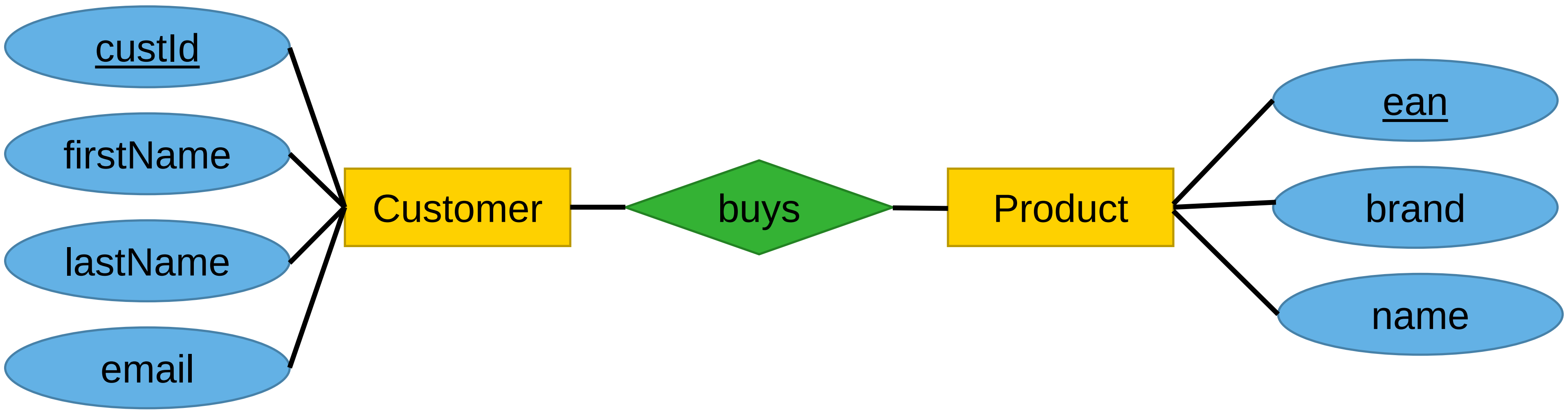
In the conceptual design phase, we want to answer questions like:

- What objects, individuals, or concepts are relevant for the application? (~ entities)
- Which relationships exist among these entities?
- What information about these entities and relationships needs to be stored? (~ attributes)
- Which rules govern these entities and relations? And which constraints can be derived from them.

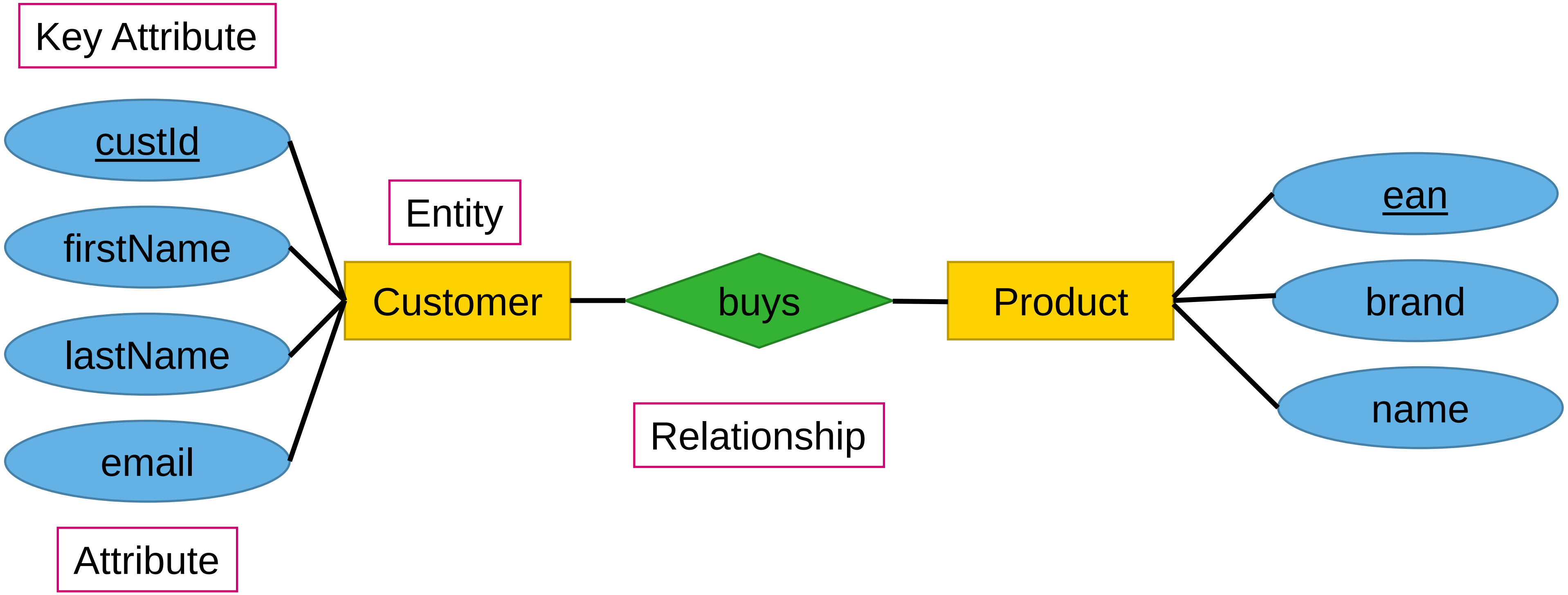
The most common way to represent this are Entity-Relationship-Diagram (ER diagram).

ER DIAGRAMS

There are different notations to represent ER diagrams (IDEF1X, Bachman, …), we will use the widely adopted **Chen-Notation**.

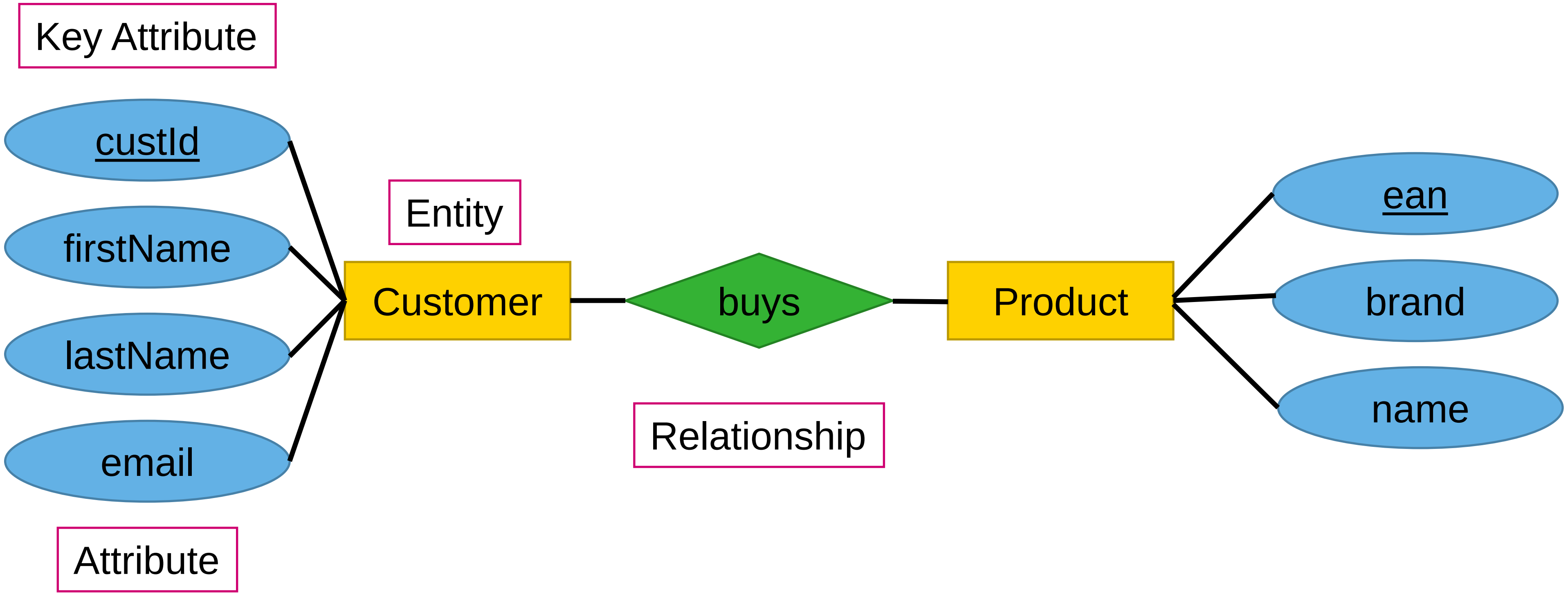


ER DIAGRAMS


Key Attribute
custId
firstName
lastName
email
Attribute
Entity
Customer
buys
Relationship
Product
ean
brand
name

ER DIAGRAMS

In a conceptual design, we have key attributes, but **NO** primary or foreign keys.



ER DIAGRAMS - NOTATION

Entities

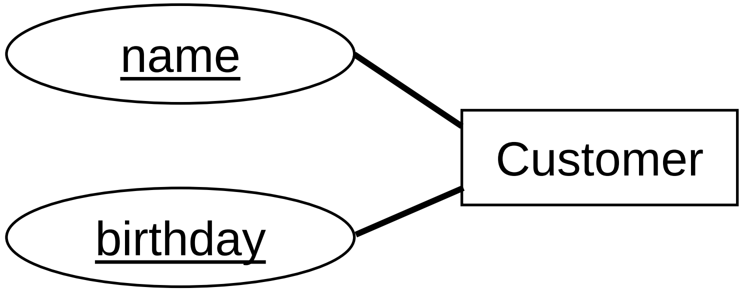
Entity

Weak Entity

ER DIAGRAMS - NOTATION

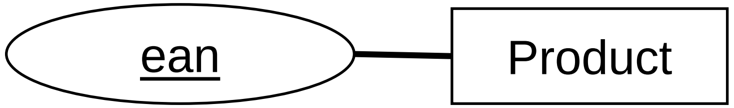
Attributes

Composite Key

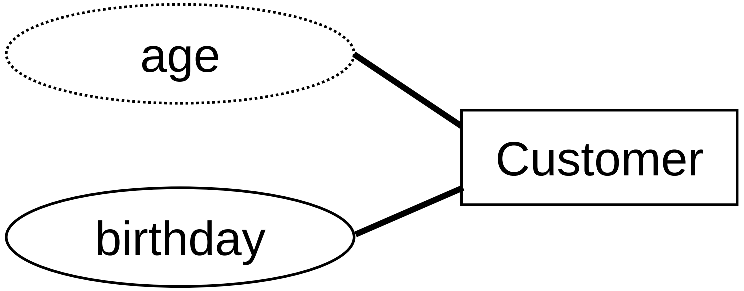


Attribute

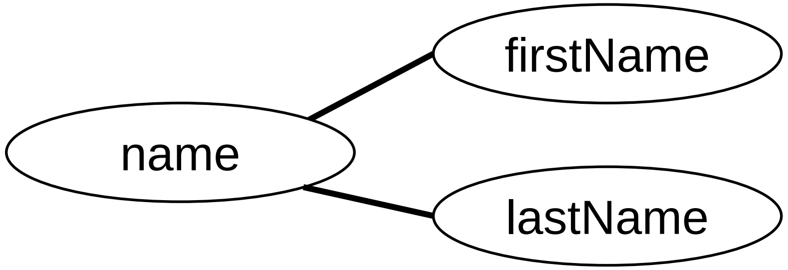
Key Attribute



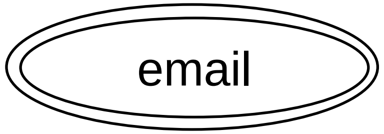
Derrived Attribute



Composite Attribute



Multi-value Attribute



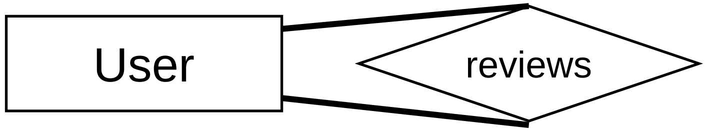
ER DIAGRAMS - NOTATION

Relationships

Binary Relationship

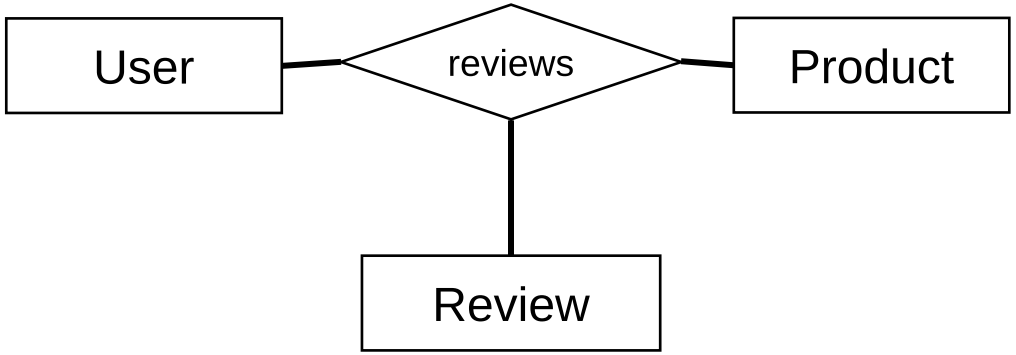
Degree of the relationship = 2

Unary / Recursive Relationship



Degree of the relationship = 1

Ternary Relationship



Degree of the relationship = 3

N-ary Relationship

…

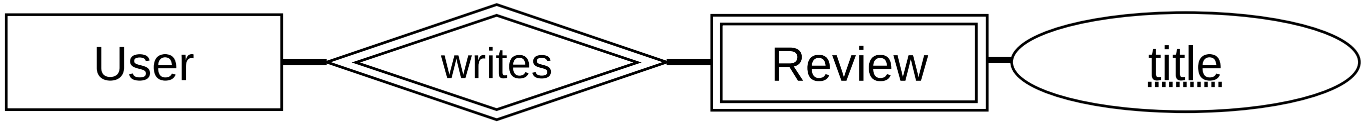
UNIVERSITY OF TWENTE.

ER DIAGRAMS - NOTATION

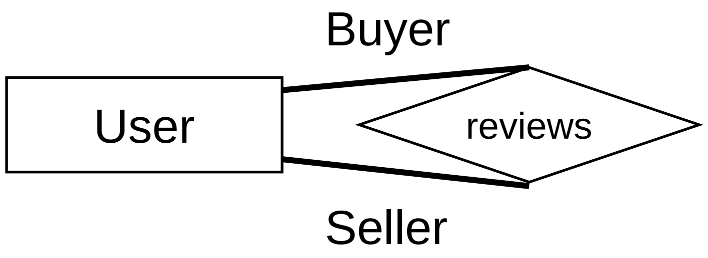
Relationships

Weak Relationship

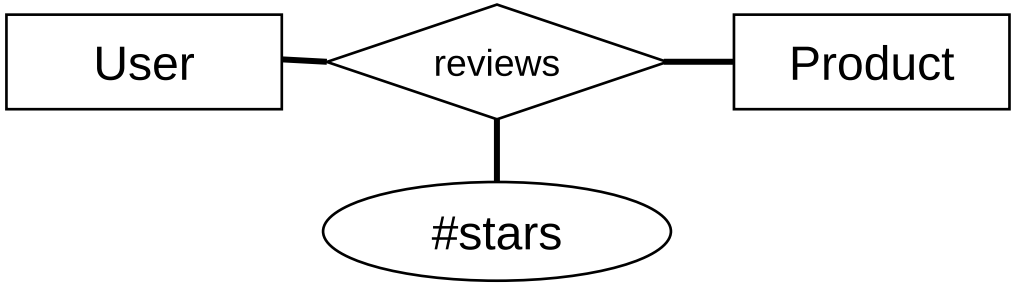
Partial Key



Role

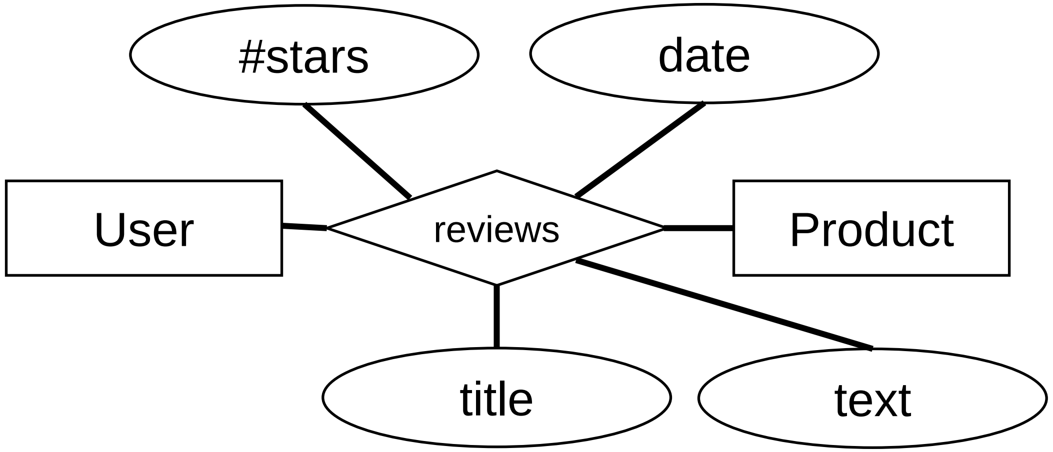


Relationship with attribute



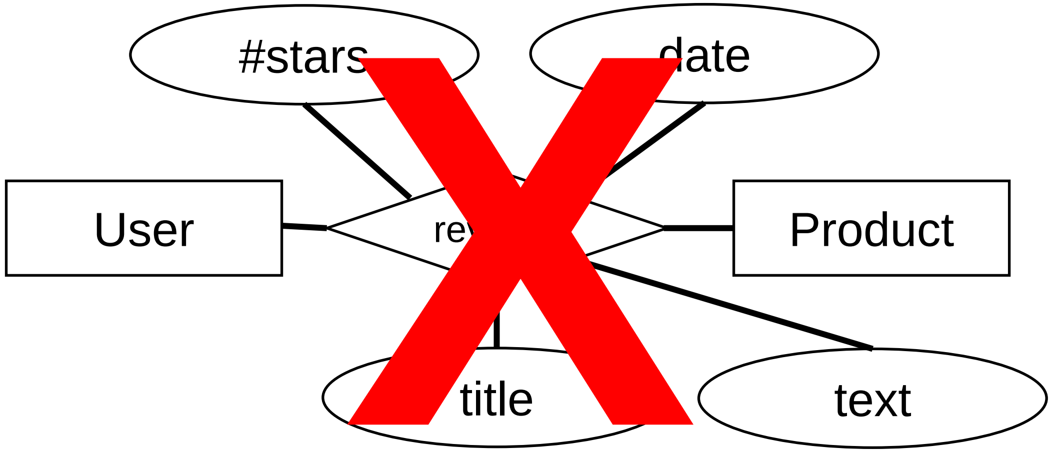
UNIVERSITY OF TWENTE.

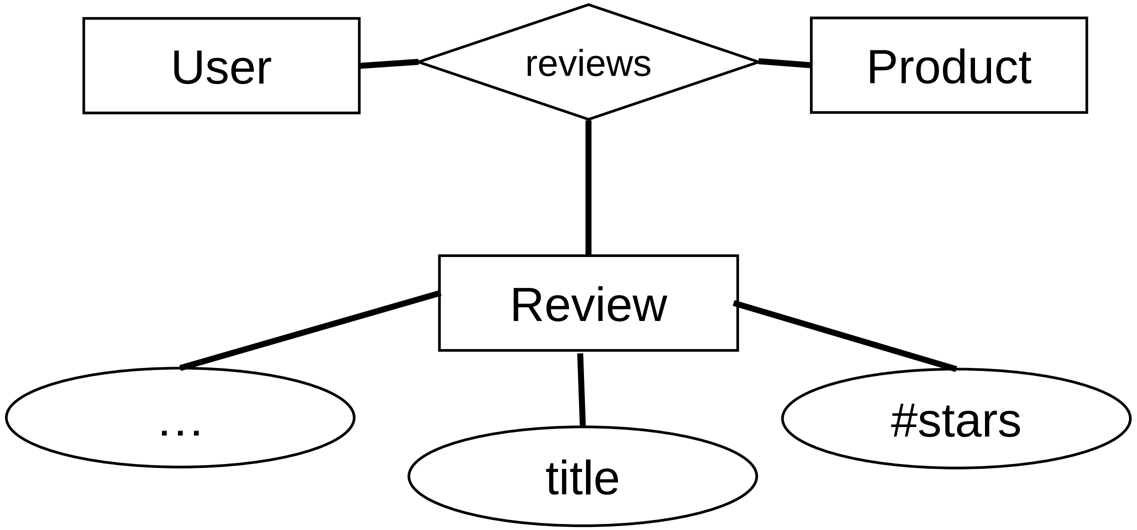
ER DIAGRAMS - NOTATION



UNIVERSITY OF TWENTE.

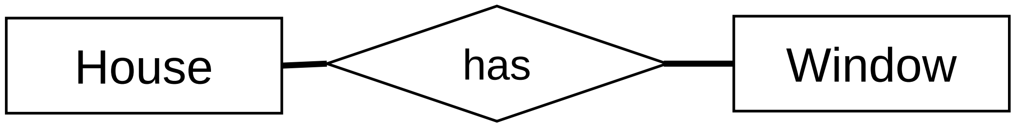
ER DIAGRAMS - NOTATION





UNIVERSITY OF TWENTE.

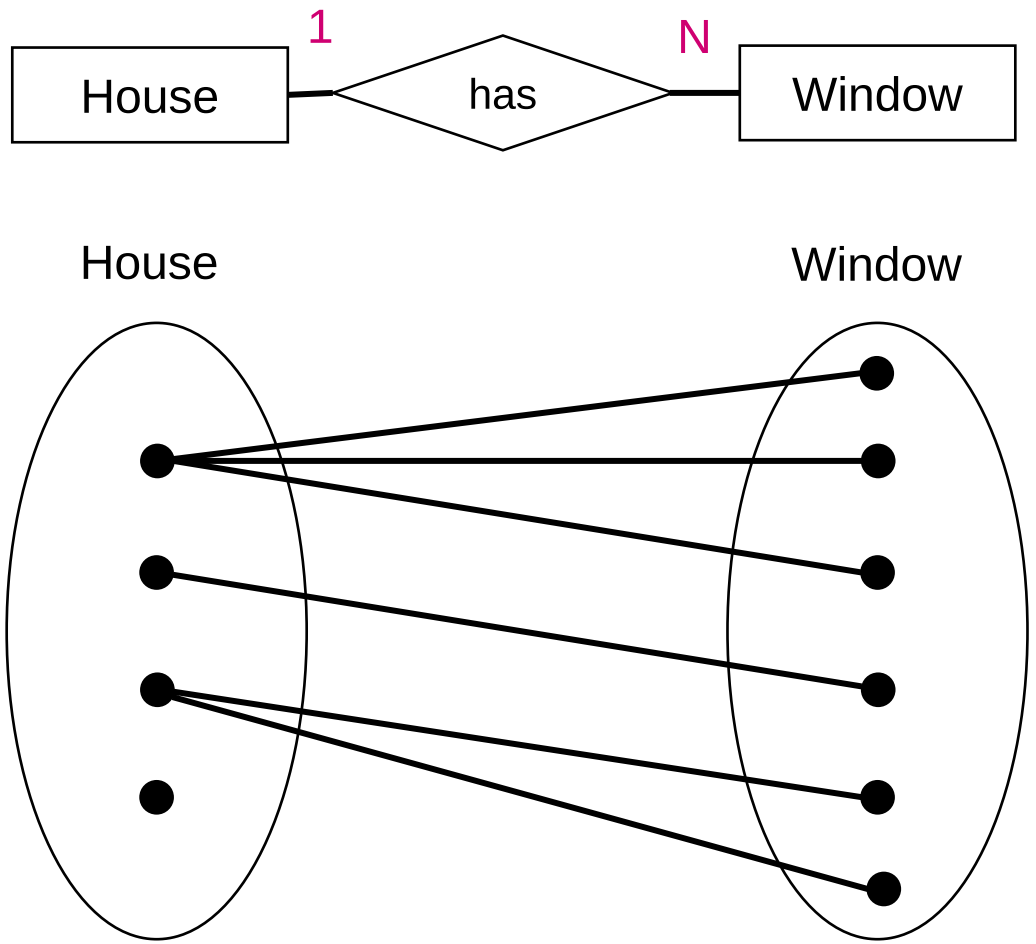
ER DIAGRAMS - NOTATION



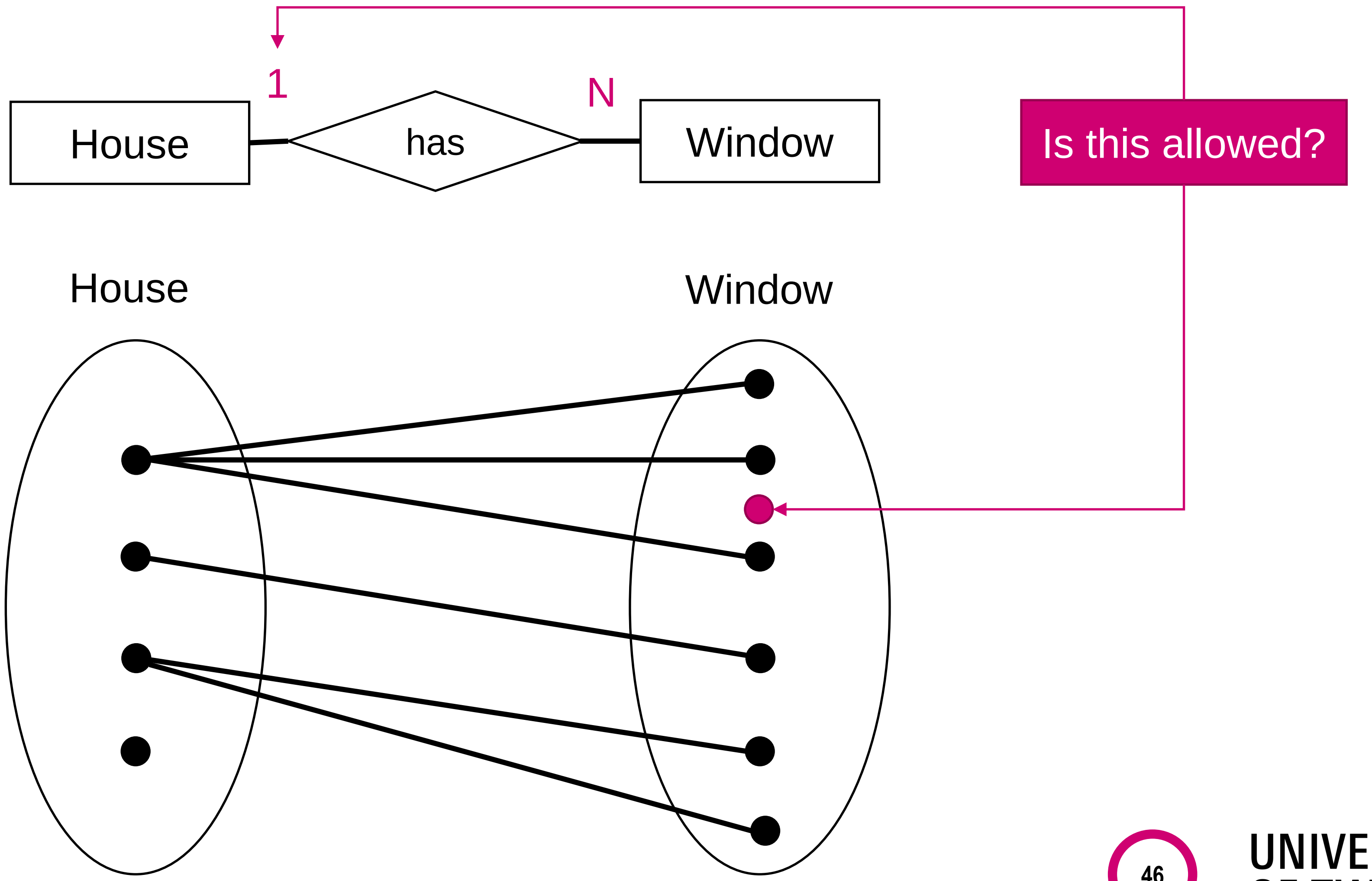
ER DIAGRAMS - NOTATION

Cardinality

One-to-Many

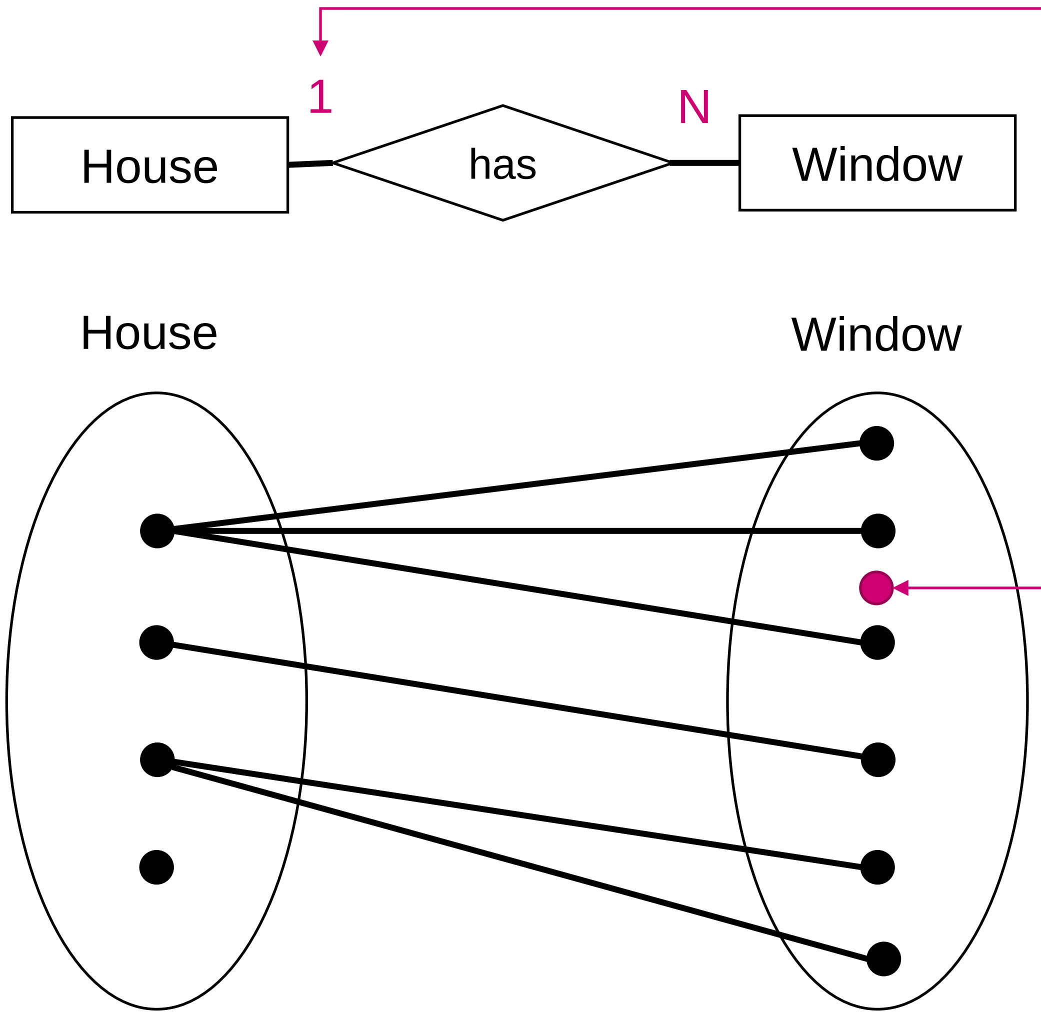


ER DIAGRAMS - NOTATION



UNIVERSITY OF TWENTE.

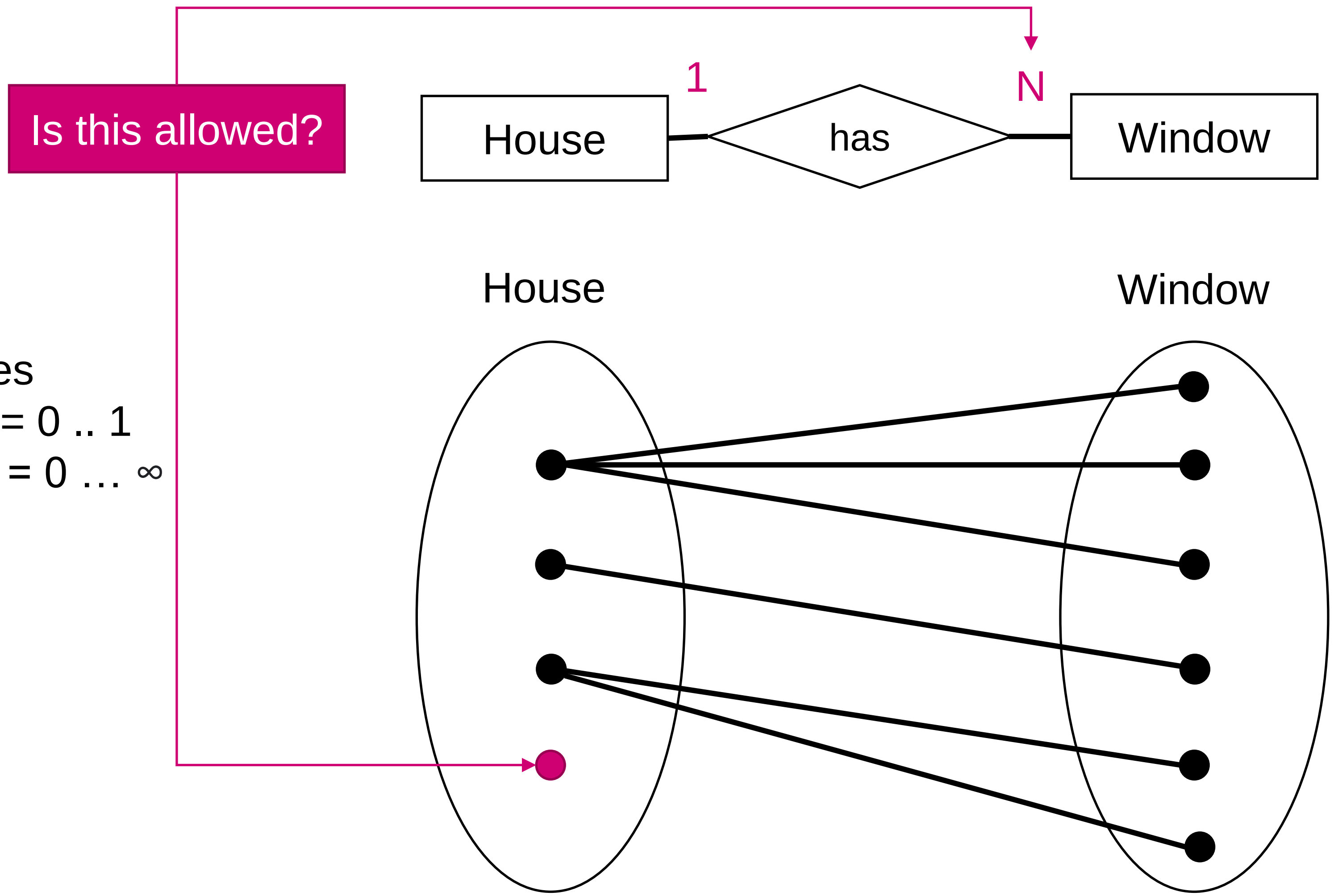
ER DIAGRAMS - NOTATION



Is this allowed?

Yes
1 = 0 .. 1
N = 0 … ∞

ER DIAGRAMS - NOTATION



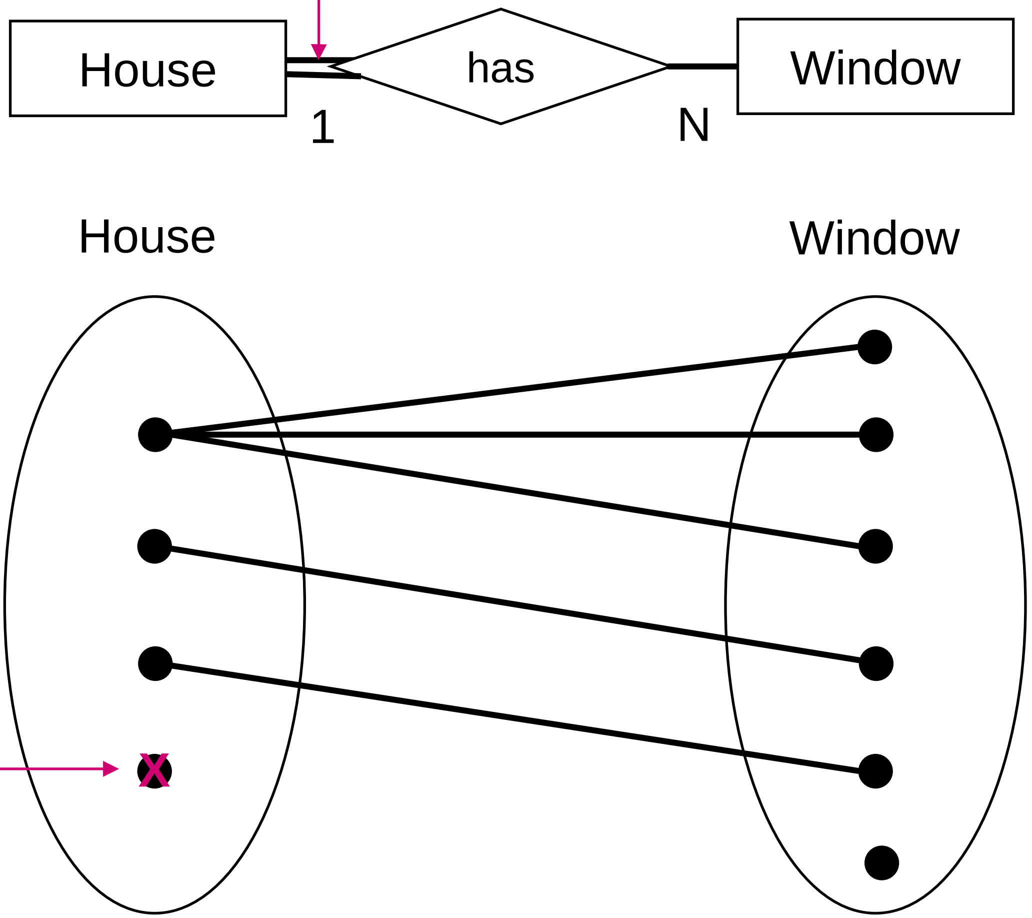
Yes
1 = 0 .. 1
N = 0 … ∞

UNIVERSITY OF TWENTE.

ER DIAGRAMS - NOTATION

Total Participation

Each entity of the type has to participate in at least one relation.

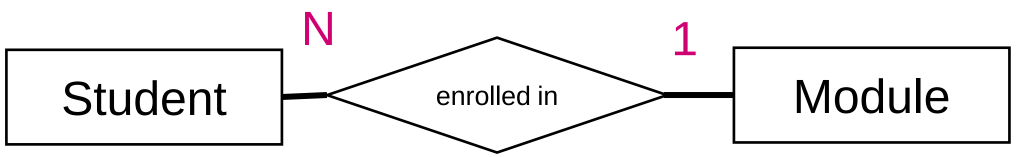


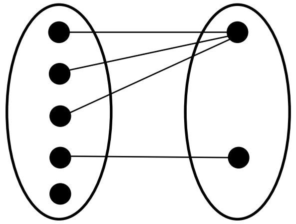
UNIVERSITY OF TWENTE.

ER DIAGRAMS - NOTATION

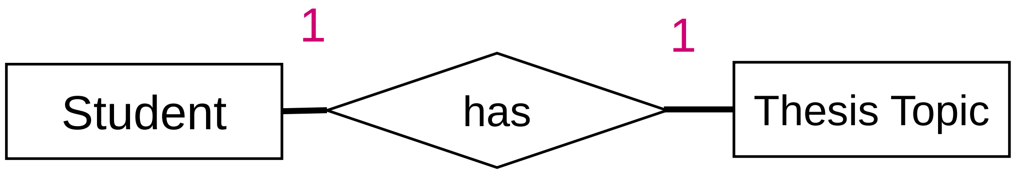
Cardinality

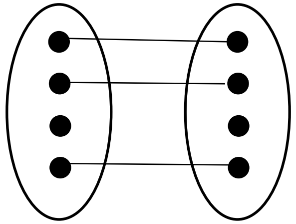
Many-to-One



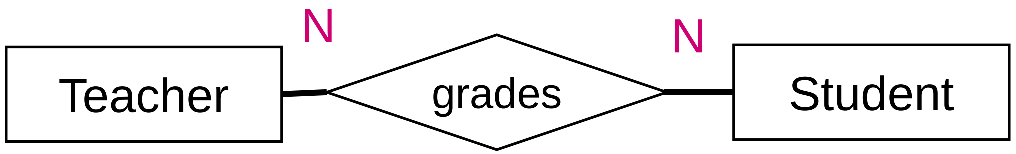


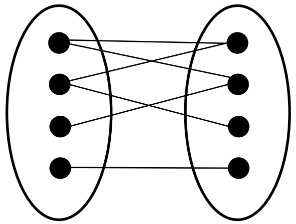
One-to-One





Many-to-Many





NOTATION IS IMPORTANT!

- Natural language is vague (“A customer has a name.”)
- Standardized notations (like the Chen notation) are used to ensure a common understanding
- In order for them to be unambiguous we have to stick to the standards
- Therefore, it does matter if you pick the right type of line, shape, …

DATABASE DESIGN STAGES

Nice! Now we can use the ER diagram to build a database, right?

DATABASE DESIGN STAGES

Nice! Now we can use the ER diagram to build a database, right?

DATABASE DESIGN STAGES

Nice! Now we can use the ER diagram to build a database, right?

But seriously, no.

Requirements analysis

Conceptual Design

Logical Design

Physical design

TABLE - CUSTOMERS

CustomerId	Firstname	Lastname	Email
1234	Alice	Hopper	a.hopper@utwente.nl
1235	Bob	Turing	b.turing@utwente.nl

RELATIONAL MODEL – BACKGROUND

Let D_1, D_2, …, D_n be **Domains** (~ value ranges)

Relation: $R \subseteq D_1 \times D_2 \times \ldots \times D_n$ (i.e., a subset of the cartesian product)

Example: Customers ⊆ int x String x String x String

Tupel: $t \in R$

Example: t = (1234, “Alice”, “Hopper”, “a.hopper@utwente.nl”)

UNIVERSITY OF TWENTE.

RELATIONAL SCHEMA

In Databases, we distinguish between:

- **Schema**, which describes the structure, and
- **Instance**, which describes the content.

A relation schema consists of

- a **name** R
- a set of **attributes** that is not empty $\{A_1, \ldots, A_n\}$
- and a domain D_i for each attribute.

$R: \{[A_1: D_1, A_2: D_2, \ldots, A_n: D_n]\}$

UNIVERSITY OF TWENTE.

RELATIONAL SCHEMA – REPRESENTATION OF ENTITIES

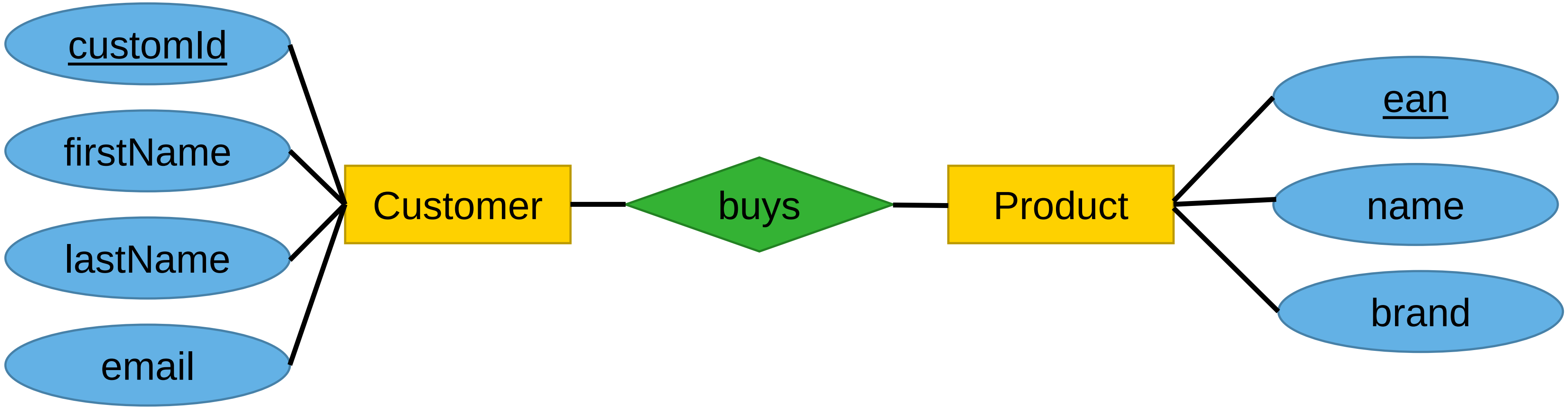
Example:

Customer: {<u>customId: integer</u>, firstname: string, lastname: string, email: string}

Product: {<u>ean: integer</u>, name: string, brand: string}

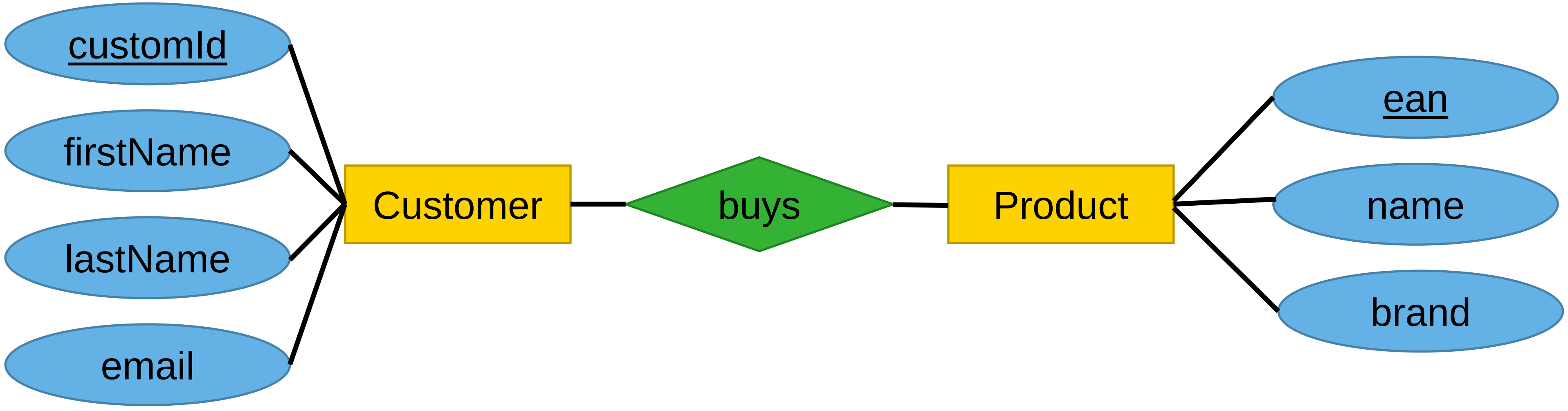
- Underlined attributes are **primary keys**
- A (candidate) **key** is a <u>minimal set</u> of attributes that uniquely identifies a tuple
- One of the (candidate) keys is selected to be the primary key and is used for referencing a tuple

RELATIONAL SCHEMA – REPRESENTATION OF RELATIONS



buys: {<u>customId: integer</u>, <u>ean: integer</u>}

RELATIONAL SCHEMA – REPRESENTATION OF RELATIONS



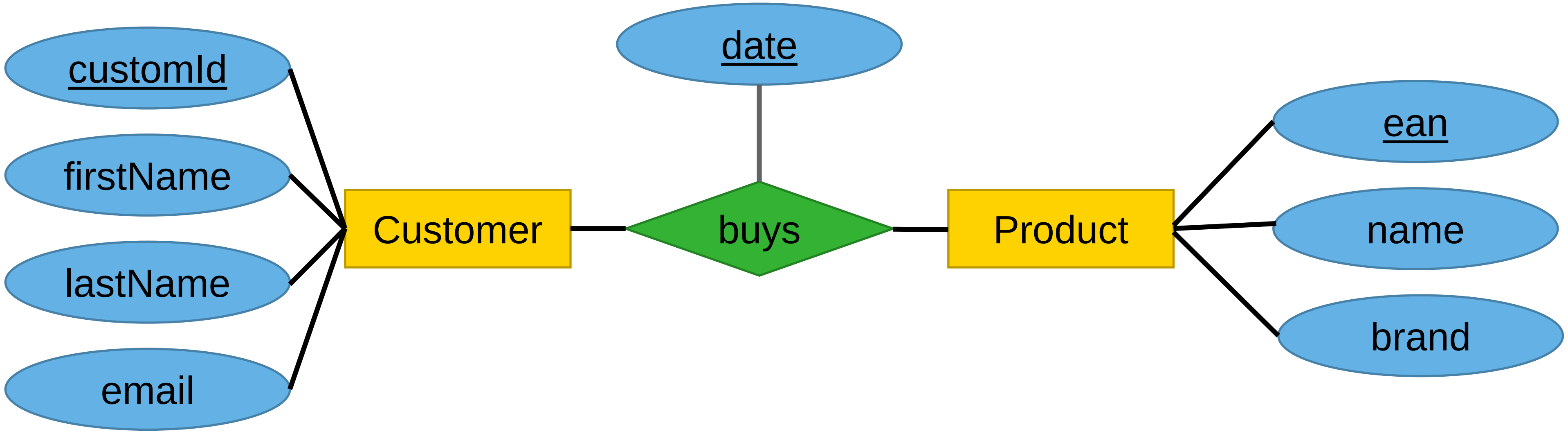
buys: {<u>customId: integer</u>, <u>ean: integer</u>}

Foreign key

Foreign key

UNIVERSITY OF TWENTE.

RELATIONAL SCHEMA – REPRESENTATION OF RELATIONS



buys: {<u>customId: integer</u>, <u>ean: integer</u>}

buys: {<u>customId: integer</u>, <u>ean: integer</u>, <u>date:date</u>}

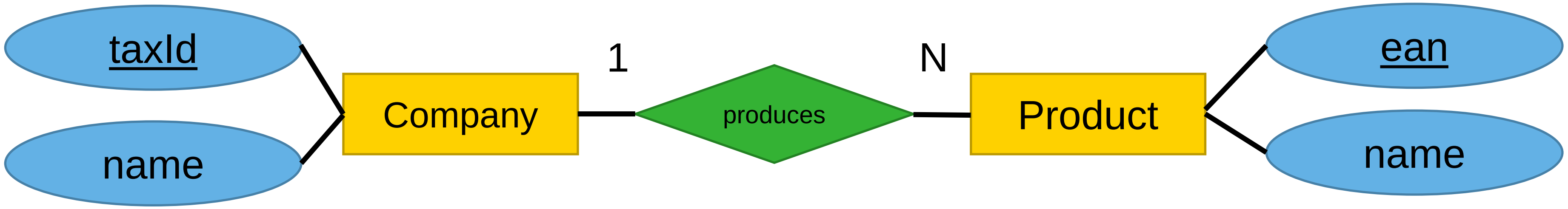
TL;DR: FROM ER MODEL TO RELATIONAL MODEL

1. Create a relation for each entity
2. The attributes of the entity are the attributes of the relation
3. Pick a primary key
4. Create a relation for each relation
5. The attributes of the relation are the primary keys of the participating entities, plus the attributes of the relations (if there are any)
6. Pick a primary key

This will give you <u>a</u> correct solution. Not the only one, not necessarily the optimal one.

OPTIMIZATION

… is a different story.

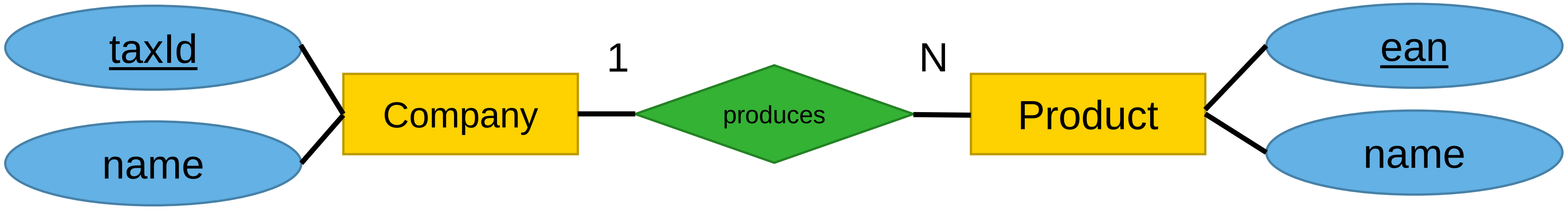


Following last slide:
Company: {TaxId: integer, Name: string}
Product: {Ean: integer, Name: string}
Produces: {TaxId: integer, ean: integer}

FK

OPTIMIZATION

… is a different story.



Following last slide:
Company: {<u>TaxId: integer</u>, Name: string}
Product: {<u>Ean: integer</u>, Name: string}
Produces: {TaxId: integer, <u>ean: integer</u>}

Rule: Relations with the same key can be merged.
Company: {<u>TaxId: integer</u>, Name: string}
Product: {<u>Ean: integer</u>, Name: string, ProducedBy: integer}

FK

UNIVERSITY OF TWENTE.

4 NORMALIZATION

NORMALIZATION

- **Normalization** is a process to optimize the organization of data in a database
  - by **removing redundancy**
  - in order to **avoid inconsistencies**.
- The result of the normalization process is a database in a so-called **Normal Form**
- Depending on the source, five to seven different are differentiated:
  - **Zeroth Normal Form (0NF)**
  - **First Normal Form (1NF)**
  - **Second Normal Form (2NF)**
  - **Third Normal Form (3NF)**
  - **Boyce Codd Normal Form (BCNF)**
  - **Fourth Normal Form (4NF)**
  - **Fifth Normal Form (5NF)**

ZEROTH NORMAL FORM (0NF)

- Data is raw and not normalized, e.g., contains composite values

InvoiceNo	OrderTime	Name	Address	Article	Price	Amount
1	22-01-15 13:37	Alice Hopper	Hallenweg 17, 7522NH Enschede	Pen	1.20	1
1	22-01-15 13:37	Alice Hopper	Hallenweg 17, 7522NH Enschede	Paper	2.00	1
2	22-02-14 12:22	Bob Turing	Hallenweg 15, 7522NH Enschede	Milk	0.80	5

FIRST NORMAL FORM (1NF)

- A relation is in 1NF if it **contains only atomic values**
- I.e., no multi-value or composite attributes

Invoice No	Order Time	First name	Last name	Street	StreetNo	Zip	City	Article	Price	Amount
1	22-01-15 13:37	Alice	Hopper	Hallen weg	17	7522NH	Enschede	Pen	1.20	1
1	22-01-15 13:37	Alice	Hopper	Hallen weg	17	7522NH	Enschede	Paper	2.00	1
2	22-02-14 12:22	Bob	Turing	Hallen weg	15	7522NH	Enschede	Milk	0.80	5

SECOND NORMAL FORM (2NF)

- A relation is in 2NF if it is in 1NF and
- All **non-key attributes are fully functional dependent** on all candidate keys
- An attribute B is **functional dependent** on an attribute A (A → B), iff each A is associated with exactly one B
- **Fully** means: All non-key attributes must depend on **all attributes** of all candidate keys

Invoice No	Order Time	First name	Last name	Street	StreetNo	Zip	City	Article	Price	Amount
1	22-01-15 13:37	Alice	Hopper	Hallen weg	17	7522NH	Enschede	Pen	1.20	1
1	22-01-15 13:37	Alice	Hopper	Hallen weg	17	7522NH	Enschede	Paper	2.00	1
2	22-02-14 12:22	Bob	Turing	Hallen weg	15	7522NH	Enschede	Milk	0.80	5

SECOND NORMAL FORM (2NF)

If the relation is in 1NF but not in 2NF:

1. Create all subsets of attributes of the primary key (~ powerset without the empty set)
2. For each subset, create a new table, with the subset as primary key
3. Add to each new table the attributes that depend on the new primary key
4. Repeat if one of the relations is not yet in 2NF

SECOND NORMAL FORM (2NF)

If the relation is in 1NF but not in 2NF:

1. **Create all subsets of attributes of the primary key (~ powerset without the empty set)**
2. For each subset, create a new table, with the subset as primary key
3. Add to each new table the attributes that depend on the new primary key
4. Repeat if one of the relations is not yet in 2NF

Invoice No	Order Time	First name	Last name	Street	StreetNo	Zip	City	Article	Price	Amount

What is the primary key? What are the subsets of its attributes?

SECOND NORMAL FORM (2NF)

If the relation is in 1NF but not in 2NF:

1. Create all subsets of attributes of the primary key (~ powerset without the empty set)
2. **For each subset, create a new table, with the subset as primary key**
3. Add to each new table the attributes that depend on the new primary key
4. Repeat if one of the relations is not yet in 2NF

Invoice No	Order Time	First name	Last name	Street	StreetNo	Zip	City	Article	Price	Amount

↓

Invoice No	Article

Invoice No

Article

SECOND NORMAL FORM (2NF)

If the relation is in 1NF but not in 2NF:

1. Create all subsets of attributes of the primary key (~ powerset without the empty set)
2. For each subset, create a new table, with the subset as primary key
3. **Add to each new table the attributes that depend on the new primary key**
4. Repeat if one of the relations is not yet in 2NF

Invoice No	Article

Invoice No

Article

Invoice No	Article	Amount

Article	Price

Invoice No	Order Time	First name	Last name	Street	StreetNo	Zip	City

THIRD NORMAL FORM (3NF)

- A relation is in 3NF if it is in 2NF and
- For each functional dependency A → B it must be true that either A is candidate key or B is part of a candidate key (or both)
- That means, we want to remove **transitive dependencies** on candidate keys:
  A → C because A → B → C

Invoice No	Article	Amount

Article	Price

Invoice No	Order Time	First name	Last name	Street	StreetNo	Zip	City

THIRD NORMAL FORM (3NF)

If the relation is in 2NF but not in 3NF:

1. For each attribute that is not a primary key and determines another attribute that is not a candidate key, create a new table with the determining attribute as key
2. Add all attributes that are determined by it

Invoice No	Order Time	First name	Last name	Street	StreetNo	Zip	City

Invoice No	Order Time	First name	Last name	Street	StreetNo	Zip

Zip	City

BOYCE CODD NORMAL FORM (BCNF)

- A relation is in BCNF (3.5NF) if it is in 3NF and
- For each functional dependency A → B it must be true ~~that either~~ **A is candidate key** ~~or B is part of a candidate key (or both)~~
- Example: Database of athletes and their teams (let's assume each team only competes in one sport, per sport an athlete is only part of one team, and there are no two athletes with the same name)

Athlete	Team	Sport
Robben	FC Groningen	Football
Neuer	FC Bayern	Football
Neuer	MSC 1836	Chess

BOYCE CODD NORMAL FORM (BCNF)

- NF1: Only atomic values => Yes
- NF2: Key candidates: {Athlete, Team}, {Athlete, Sport}, there are no non-key attributes => Yes
- NF3: Team → Sport, Team (alone) ist not key candidate, but Sport is part of one => Yes
- BCNF: Team → Sport, Team (alone) ist not key candidate => No

Athlete	Team	Sport
Robben	FC Groningen	Football
Neuer	FC Bayern	Football
Neuer	MSC 1836	Chess

BCNF: For each functional dependency A → B it must be true ~~that either~~
A is candidate key ~~or B is part of a candidate key (or both)~~

GOING FURTHER...

- It is often assumed, though contested (Wu, 1992), that in practice, databases rarely fulfill BCNF but not also the 4NF and 5NF
- 4NF and 5NF are therefore sometimes considered less important

Wu, Margaret S. (1992). "The Practical Need for Fourth Normal Form". *ACM SIGCSE Bulletin*. **24** (1): 19–23. doi:10.1145/135250.134515.

FOURTH NORMAL FORM (4NF)

- A relation is in 4NF if it is in BCNF and
- Does not contain any multivalued dependencies on key attributes
- If we have three attributes A, B, and C, we say B has a **multivalued dependency** on A, if for a single value of A, multiple values of B can exist, independent from C.

Person	Pet	Book
Alice	Rabbit	Sapiens
Alice	Cat	Faust
Alice	Rabbit	Faust
Alice	Cat	Permanent Record

Person	Pet

Person	Book

FIFTH NORMAL FORM (5NF)

- A relation is in 5NF if it is in 4NF and
- it can not be further split without loosing information.

Person	Eyes	Hair
Bob	Blue	Brown
Bob	Green	Blond

PersonId	Eyes	Hair
1	Blue	Brown
2	Green	Blond

PersonId	Eyes

PersonId	Hair

RECAP

RECAP

Today, we discussed:

- What a **DBMS** is and which **properties** we expect from it
- Different **types of databases** and their **advantages** and **disadvantages**
- **Database design** and its **stages**
- **Conceptual design** with **ER diagrams**
- **Logical design** with **relational schemata**
- **Normalization**