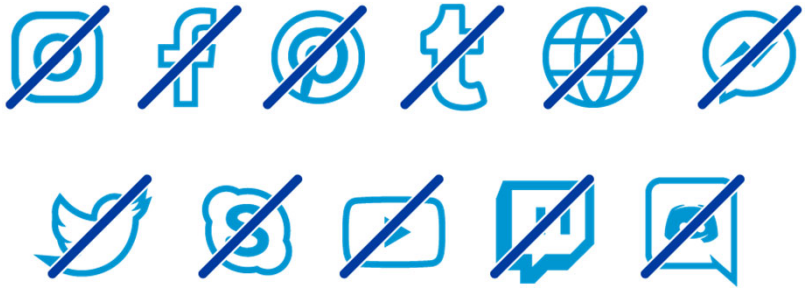
DATABASES

Source: https://en.itpedia.nl/2017/11/26/wat-is-een-database/

Prof. Dr. Ulrike Herster
Hamburg University of Applied Sciences

HAW HAMBURG

COPYRIGHT



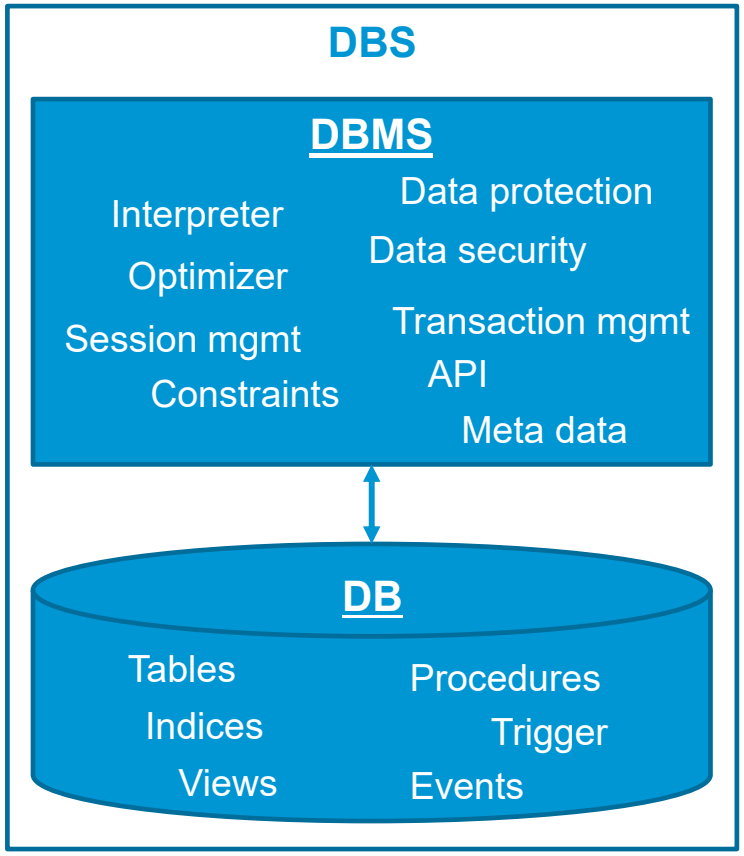
The publication and sharing of
slides, images and sound recordings of this
course is not permitted

BASICS
DATABASE EXAMPLES

Repetition

Source: www.unitop-welt.de

BASICS
DEFINITION



Repetition

DB

→ Manage data logically and physically

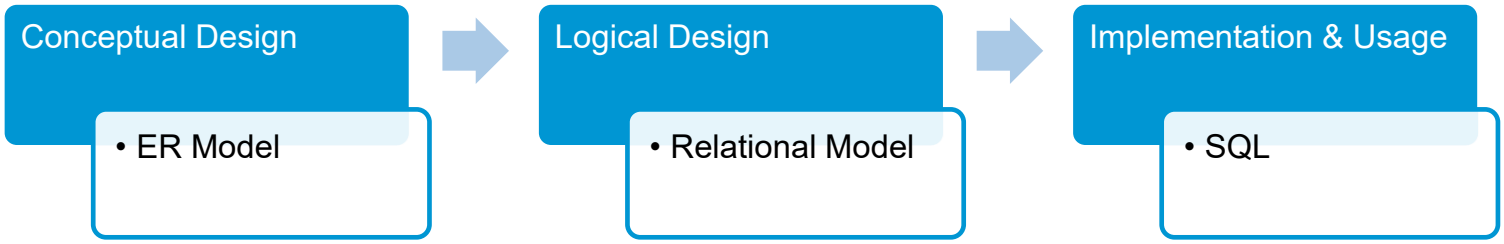
DBMS

→ Offers tools for managing, editing, and evaluating data

Source: Adams „SQL“

BASICS
DATABASE DESIGN

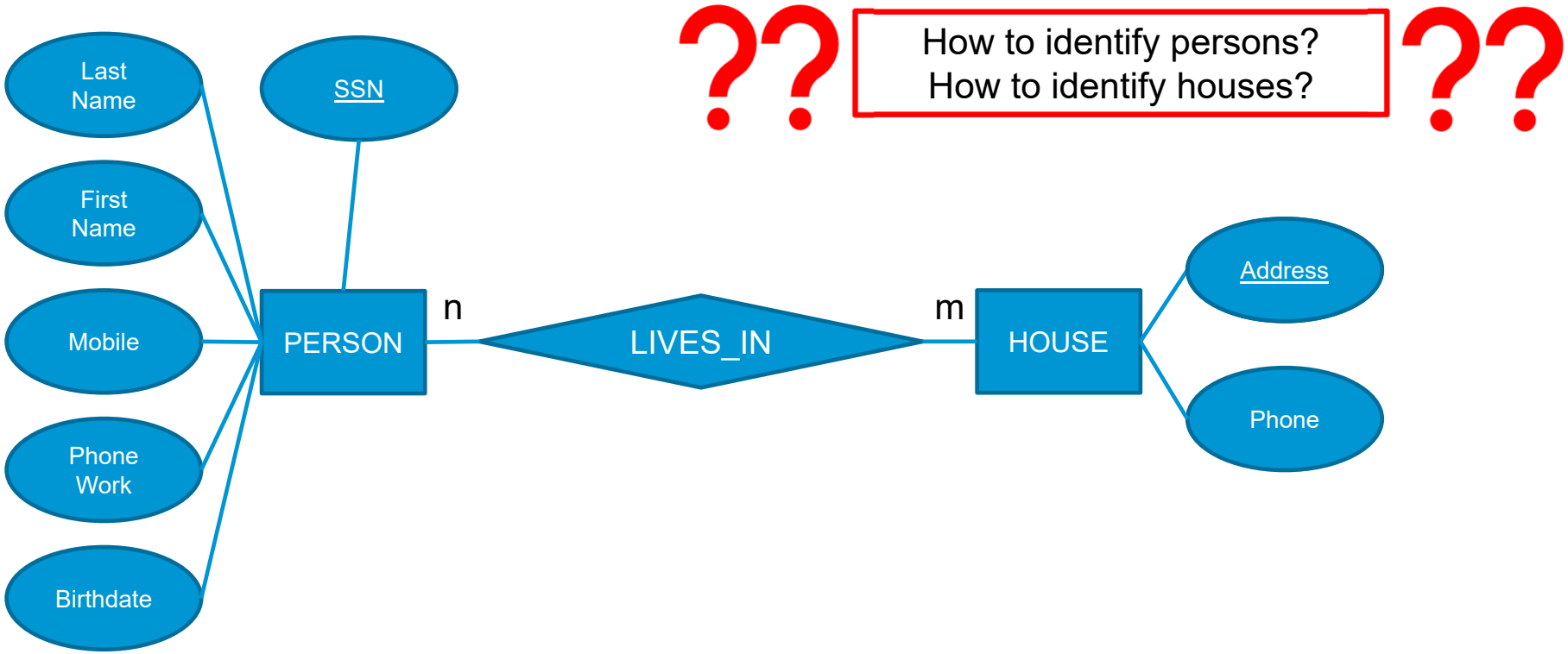
Repetition



BASICS
EXAMPLE: CONTACT LIST

Repetition

1. Conceptual design with ER Model



BASICS
EXAMPLE: CONTACT LIST

Repetition

2. Logical design with Relational Data Model

PERSON	SSN	FirstName	LastName	Mobile	PhoneWork	Birthdate

LIVES_IN	SSN (FK)	Address (FK)

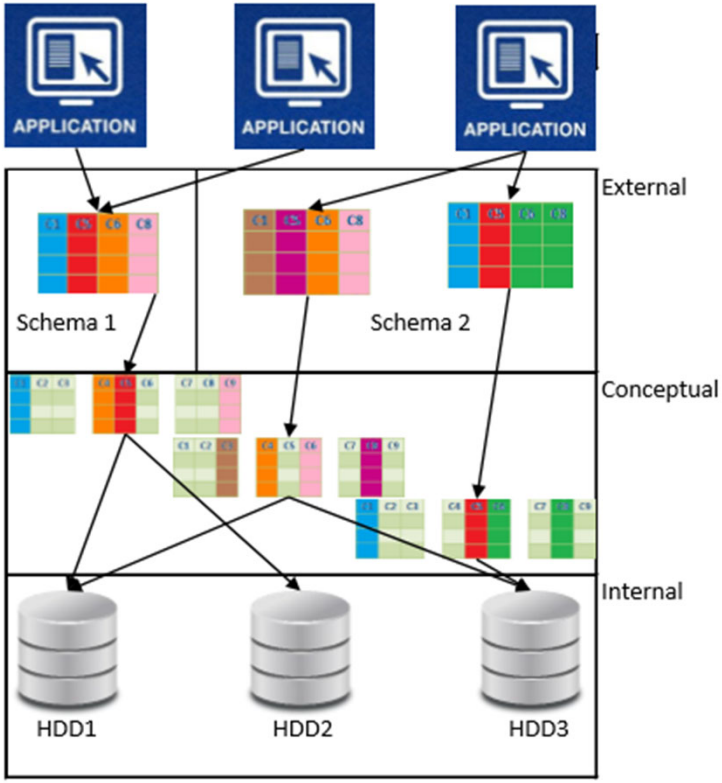
HOUSE	Address	Phone

BASICS
EXAMPLE: CONTACT LIST

3. Implementing in database

```
CREATE TABLE Person(
      SSN           CHAR(9)       NOT NULL ,
      FirstName     VARCHAR(15)   NOT NULL ,
      LastName      VARCHAR(15)   NOT NULL ,
      Mobile        VARCHAR(30) ,
      PhoneWork     VARCHAR(30) ,
      Birthdate     DATE ,
      PRIMARY KEY ( SSN )
);
```

BASICS
THE ANSI-SPARC ARCHITECTURE



Source: https://stackoverflow.com/questions/9771884/ansi-sparc-practical-explanation

BASICS
DATA INDEPENDENCE


Repetition

- Database Systems (and its data) must be accessible for a long time
  - Example: Insurance company (decades)
  - Often longer than the lifetime of applications, operating systems, hardware

- Need for data independence
  - Avoid tight coupling of applications, data, and operating environment
  - Physical data independence
  - Logical data independence

SIMPLE ENTITIES AND ATTRIBUTES
ERM: ENTITY-RELATIONSHIP MODEL

Repetition

- Entity Type
  - Represented as rectangle in ERM
  - Singular noun
- Attribut Type
  - Represented as ovals
  - Noun
- Relationship Type
  - Represented as Diamond in ERM
  - Always between entities
  - Verb
  - Has cardinalities

SIMPLE ENTITIES AND ATTRIBUTES
ERM: SIMPLE ENTITIES – EXAMPLE COMPANY

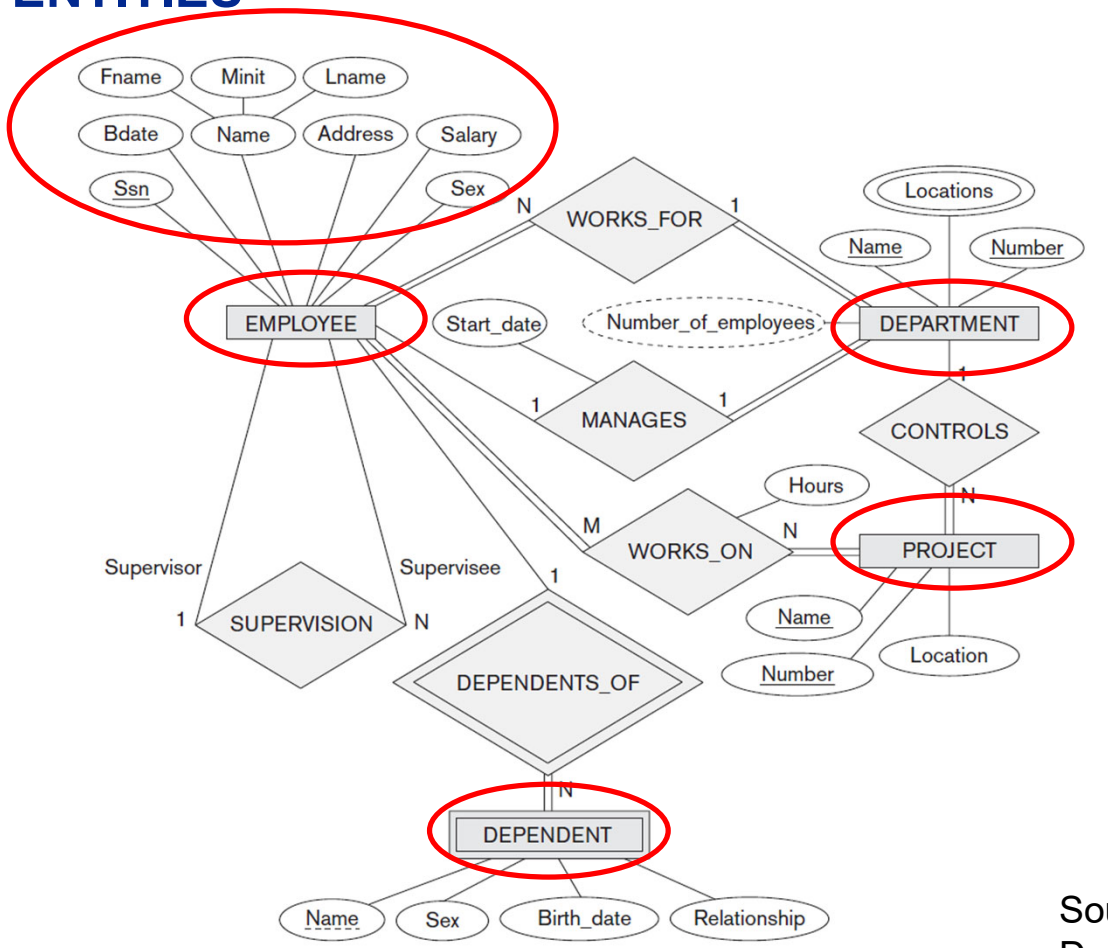
Repetition

- The company is organized into departments
- Each department has a unique name, a unique number, a manager (employee) with start date, and several locations
- A department controls a number of projects, each with unique name, unique number, single location
- We store each employee's name, ssn, address, salary, sex, birthdate
- An employee is assigned to one department, but may work on several projects, also from other departments
- We keep track of the hours per week per project
- We also keep track of the supervisor
- We want to keep track of each employee's dependents for insurance purposes, namely first name, sex, birth date, and relationship to employee.

Source: Elmasri, Fundamentals of Database Systems, Page 204 ff

SIMPLE ENTITIES AND ATTRIBUTES

ERM: SIMPLE ENTITIES



Repetition

Source: Elmasri, Fundamentals of Database Systems, Page 204 ff

HOMEWORK

Each student must install a DBMS on
their home computer
(mySQL preferred, MariaDB, PostgresSQL, Oracle,...)!!!

Source: Foto von K8 auf Unsplash

SIMPLE ENTITIES AND ATTRIBUTES
ERM: ATTRIBUTES – KEY ATTRIBUTES (IDENTIFYING ATTRIBUTES)

- How can we identify an actual entity within an entity set?
- Attributes must be used
  → Key Attributes (also called identifying attributes)
- Sometimes several attributes together form a key attribute (identifying attribute), meaning that the combination of the attribute values must be distinct for each entity
  - If a set of attributes possesses this property, the proper way to represent this in the ER model that is to define a *composite attribute* and designate it as a key attribute of the entity type
  - Notice that such a composite key attributes must be minimal; that is, all component attributes must be included in the composite attribute to have the uniqueness property
- Key attributes are underlined
- If two attributes are underlined separately, then each is an identifying attribute on its own

Source: Elmasri, Fundamentals of Database Systems, Page 204 ff

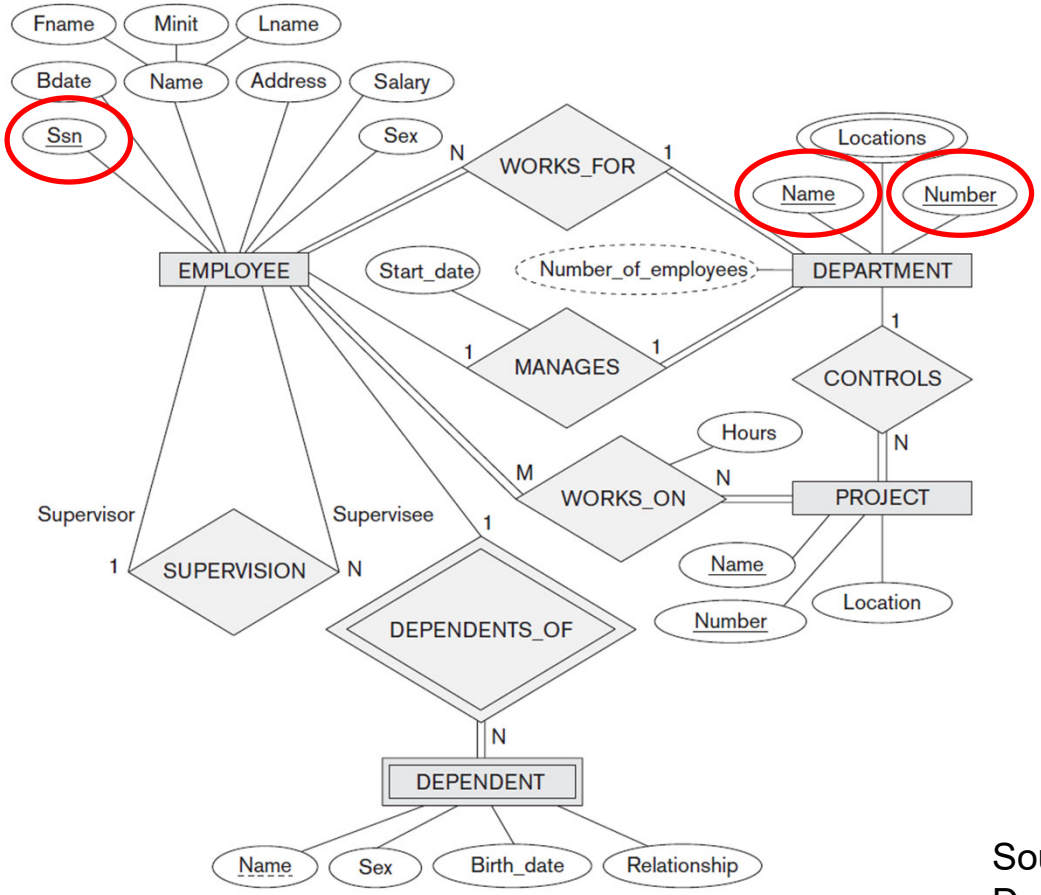
SIMPLE ENTITIES AND ATTRIBUTES
ERM: ATTRIBUTES – KEY ATTRIBUTES (IDENTIFYING ATTRIBUTES)

What are key attributes for entity typs EMPLOYEE and DEPARTMENT?

- The company is organized into departments
- Each department has a unique name, a unique number, a manager (employee) with start date, and several locations
- A department controls a number of projects, each with unique name, unique number, single location
- We store each employee's name, ssn, address, salary, sex, birthdate
- An employee is assigned to one department, but may work on several projects, also from other departments
- We keep track of the hours per week per project
- We also keep track of the supervisor
- We want to keep track of each employee's dependents for insurance purposes, namely first name, sex, birth date, and relationship to employee.

Source: Elmasri, Fundamentals of Database Systems, Page 204 ff

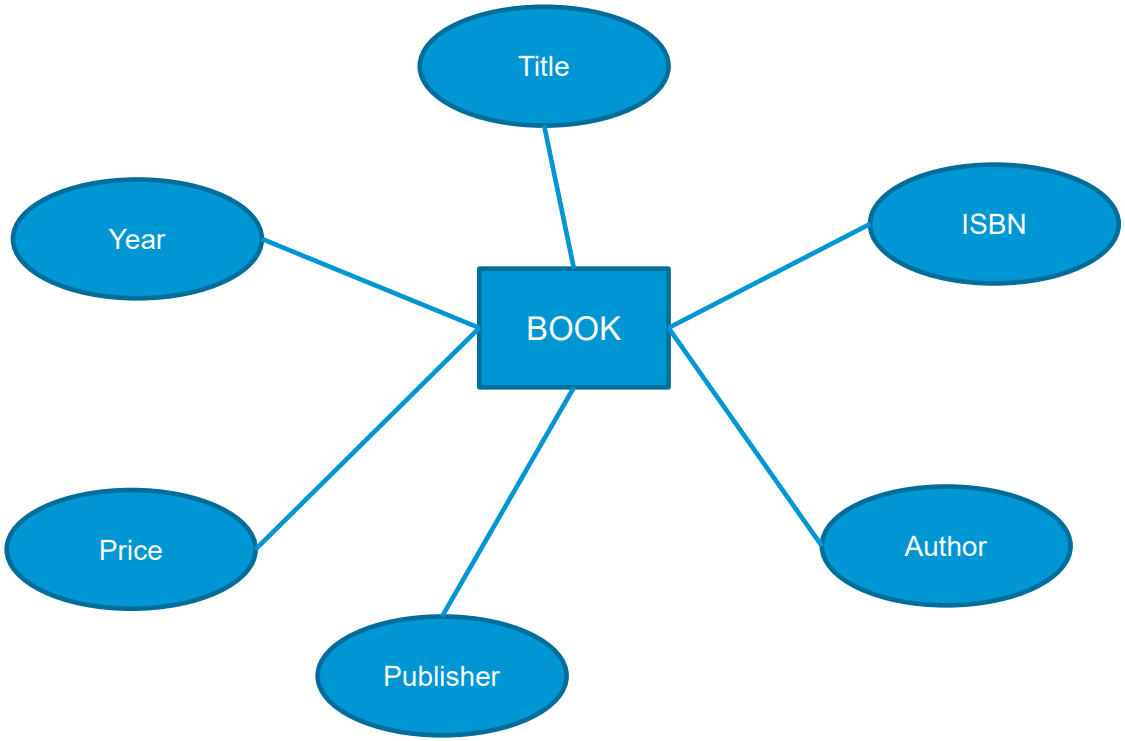
SIMPLE ENTITIES AND ATTRIBUTES
ERM: ATTRIBUTES – KEY ATTRIBUTES (IDENTIFYING ATTRIBUTES)



Source: Elmasri, Fundamentals of Database Systems, Page 204 ff

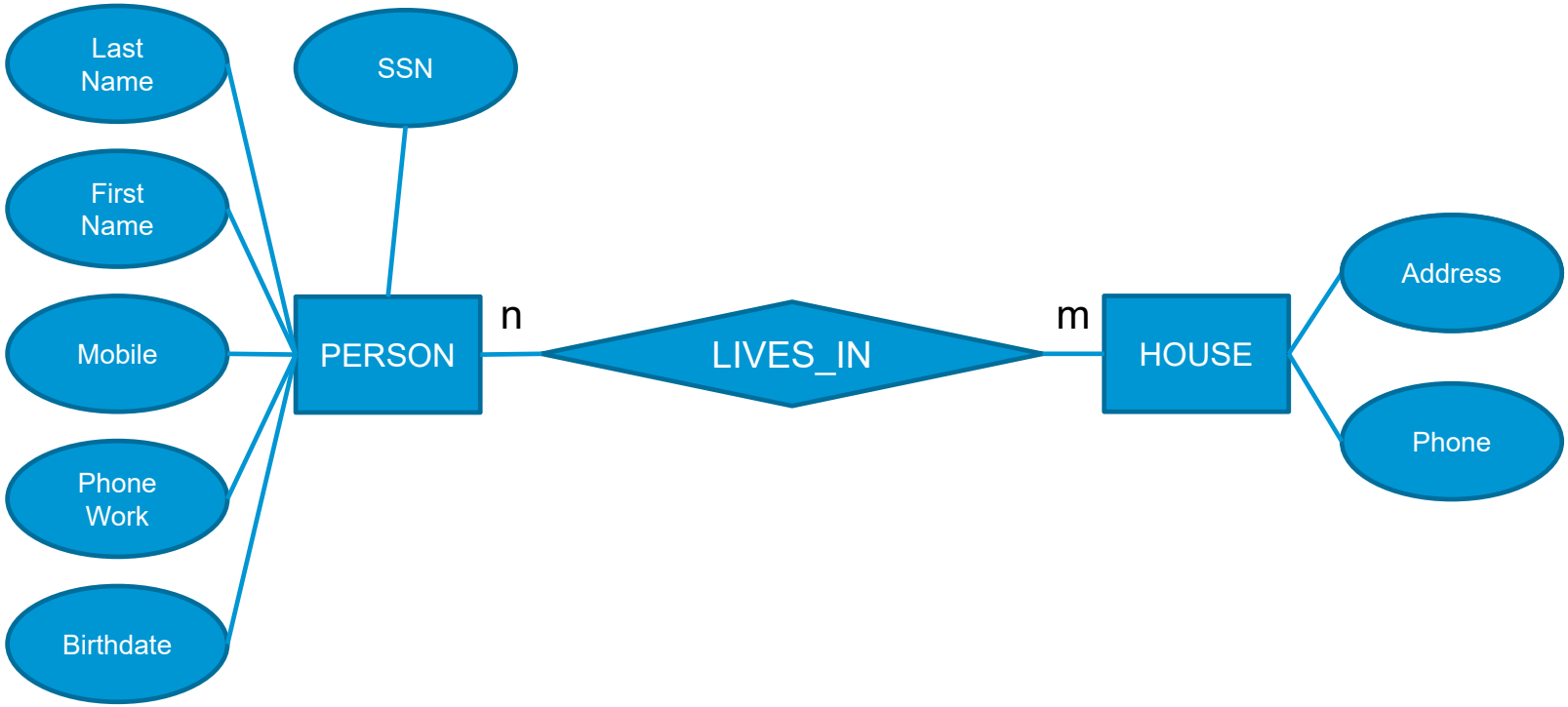
SIMPLE ENTITIES AND ATTRIBUTES
ERM: ATTRIBUTES – KEY ATTRIBUTES (IDENTIFYING ATTRIBUTES)

What are key attributes for entity type BOOK?

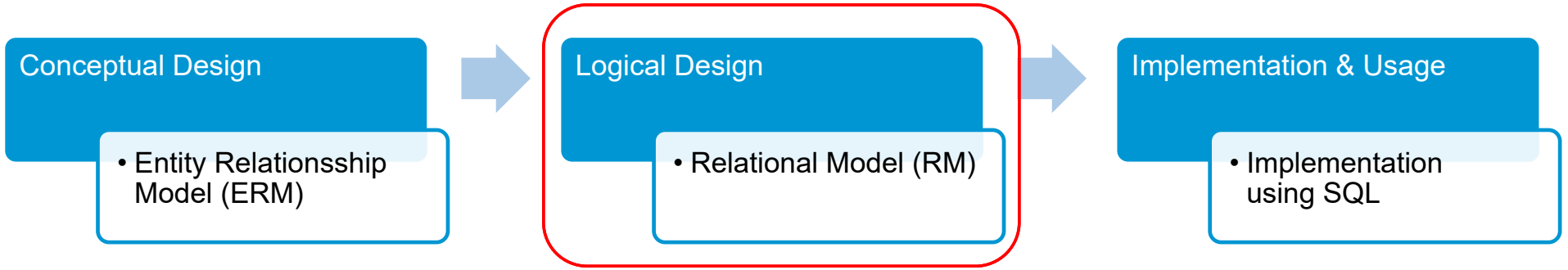


SIMPLE ENTITIES AND ATTRIBUTES
ERM: ATTRIBUTES – KEY ATTRIBUTES (IDENTIFYING ATTRIBUTES)

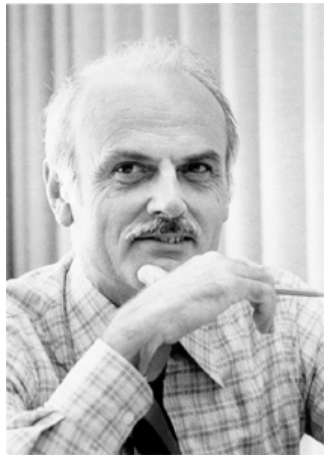
What are key attributes for entity types PERSON and HOUSE?



SIMPLE ENTITIES AND ATTRIBUTES
RM: DATABASE DESIGN



SIMPLE ENTITIES AND ATTRIBUTES
RM: RELATIONAL MODEL


Source: www.wikipedia.org

- Edgar F. Codd invented the relational model in 1970 and won the Turing Prize (Nobel Prize for Computer Science) for it
- The model has become widely accepted in practice and has a mathematical basis
- The model is based on relations, which are subsets of the Cartesian product
- Simpler approach:

  a relation is a table, a tuple is a row
- Everything is modelled in a table and stored in a row of this table

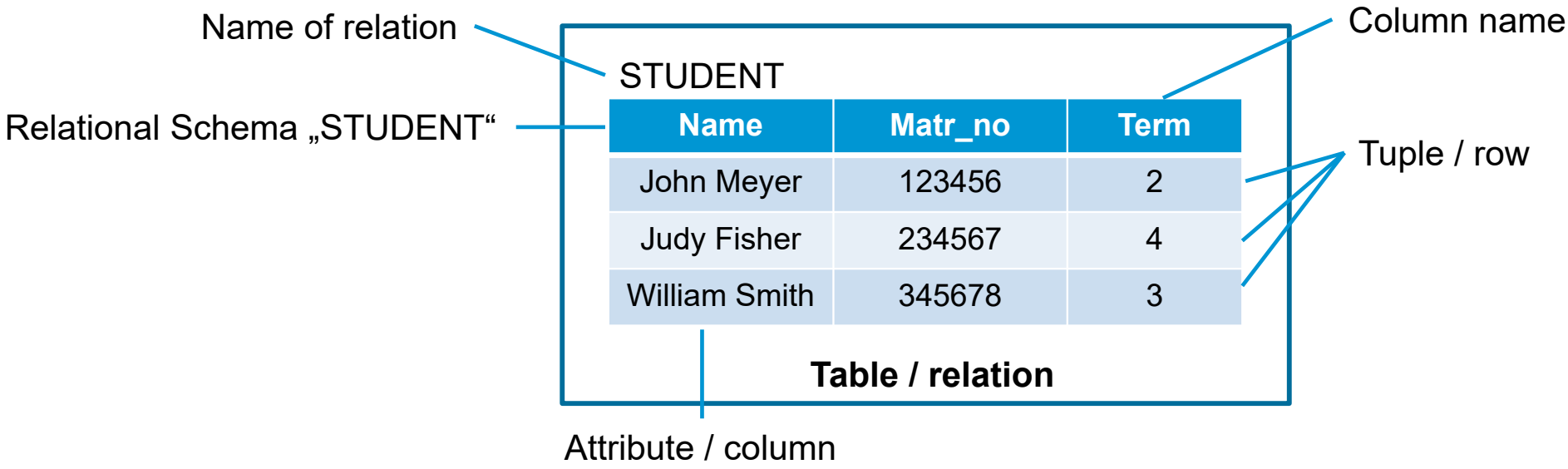
Name	Matr_no	Term
John Meyer	123456	2
Judy Fisher	234567	4
William Smith	345678	3

SIMPLE ENTITIES AND ATTRIBUTES
RM: RELATIONAL MODEL

- The relational model describes objects and relationships as a relational schema
- A *relational schema* consists of a set of attributes
- Each attribute belongs to a value range/type
- A *database schema* consists of a set of relational schemas
- A *relation* displays the current data for the relational schema
- The set of relations is called the *database* (or the state of the DB)
- An element of a relation is called a *tuple*, which is simply a *row*

Name	Matr_no	Term
John Meyer	123456	2
Judy Fisher	234567	4
William Smith	345678	3

SIMPLE ENTITIES AND ATTRIBUTES
RM: RELATIONAL MODEL



Attribute	Type
Name	String
Matr_no	Integer
Term	Integer

SIMPLE ENTITIES AND ATTRIBUTES
RM: RELATIONAL MODEL

Name of relation

Relational Schema „STUDENT“

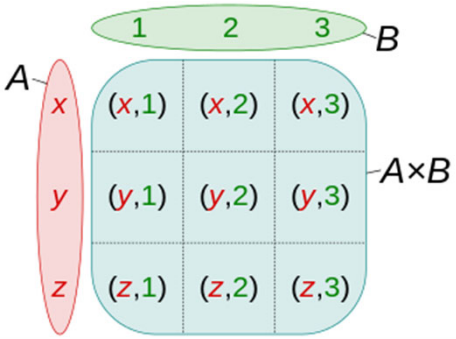
Column name

Tuple / row

STUDENT

Name	Matr_no	Term
John Meyer	123456	2
Judy Fisher	234567	4

A table can be created by taking the Cartesian product of a set of rows and a set of columns.
If the Cartesian product rows × columns is taken, the cells of the table contain ordered pairs of the form (row value, column value).



Mathematically:
Relation STUDENT ⊆ String × Integer × Integer

Cartesian Product

Source: www.wikipedia.org

SIMPLE ENTITIES AND ATTRIBUTES
RM: RELATIONAL MODEL

- Objects are described using relations
  - Relations can be viewed as tables
  - But: Not like a spreadsheet table!
- There can be links between relations
- Attributes describe properties
- Possible attribute values are defined by the domain

Source: Elmasri, Fundamentals of Database Systems, Page 59ff

SIMPLE ENTITIES AND ATTRIBUTES
RM: RELATIONAL MODEL

Informally

- A *relational model* represents the database as a collection of *relations*
- Each relation resembles a *table* of values or, to some extent, a flat file of records
- When a relation is thought of as a table of values,
  each *row* in the table represents a collection of related data values
- A row represents a fact that typically corresponds to a real-world entity or relationship
- The *table name* and *column names* are used to help to interpret
  the meaning of the values in each row
- All values in a column are of the same data type

Source: Elmasri, Fundamentals of Database Systems, Page 59ff

SIMPLE ENTITIES AND ATTRIBUTES
RM: RELATIONAL MODEL

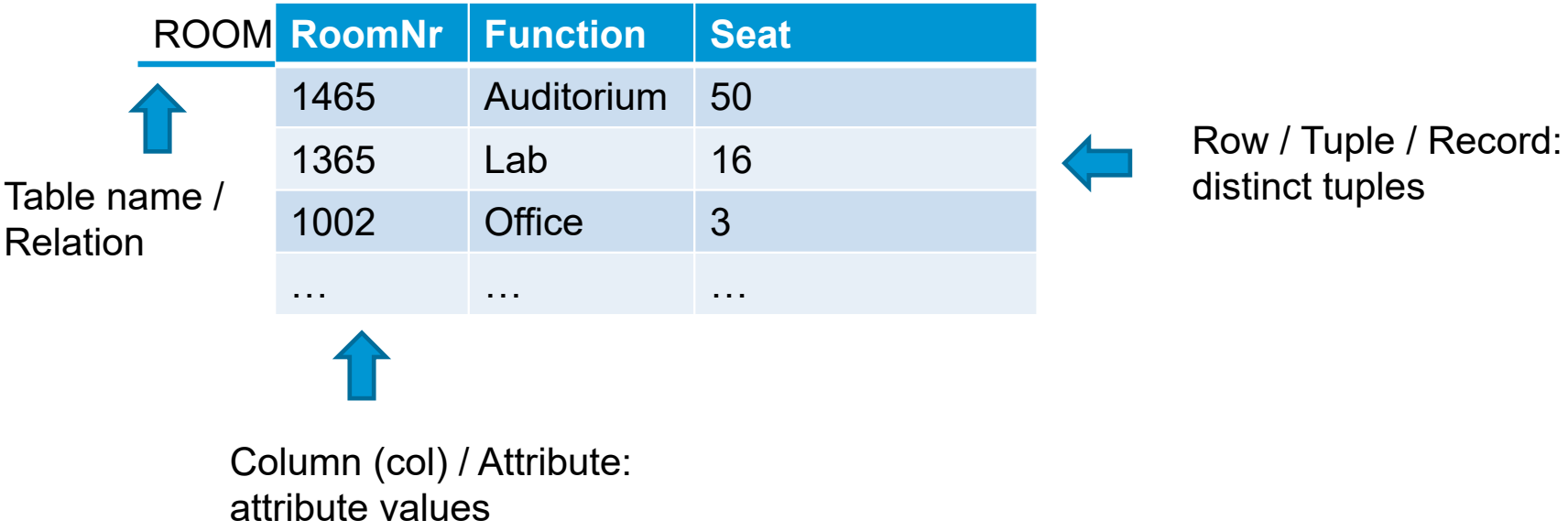
Formally

- A row is called a *tuple*
- A column header is called an *attribute*
- The table is called a *relation*
- The data type describing the types of values that can appear in each column is represented by a domain of possible values

Source: Elmasri, Fundamentals of Database Systems, Page 59ff

SIMPLE ENTITIES AND ATTRIBUTES
RM: RELATIONAL MODEL - EXAMPLE

- Example:
  - ROOM(RoomNr, Function, Seats)
  - Function = $\{Auditorium, Lab, Office, Administration\}$



SIMPLE ENTITIES AND ATTRIBUTES
RM: RELATIONAL MODEL

- Relation Schema $R\ (A_1, A_2, \ldots\ A_n)$ is made up of
  - A relation name R and
  - A list of attribute names $A_1, A_2, \ldots\ A_n$
- Each attribute A_i is the name of a role played by some domain D in the relation schema R
- D is called the domain of A_i and is denoted by $dom(A_i)$
- A relation schema is used to describe a relation
- R is called the name of this relation
- The *degree* (or arity) of a relation is the number of attributes n of its relation schema

BOOK

ISBN	Title	Author	Publisher	Year	Price
978-1-292-09761-9	Fundamentals of Database Systems	Ramez Elmasri	Prentice Hall	2016	59.99
978-0321197849	An Introduction to Database Systems	C. J. Date	Pearson	2003	69.92
…	…	…	…	…	…

Source: Elmasri, Fundamentals of Database Systems, Page 59ff

SIMPLE ENTITIES AND ATTRIBUTES
RM: RELATIONAL MODEL - EXAMPLE

- Relational schema:
  BOOK (ISBN,
  Title,
  Author,
  Publisher,
  Year,
  Price)

  → Relation BOOK is of degree six

- Relational schema with data types:
  BOOK (ISBN : integer,
  Title : string,
  Author : string,
  Publisher : string,
  Year : integer,
  Price : decimal(6,2))

BOOK	ISBN	Title	Author	Publisher	Year	Price

Source: Elmasri, Fundamentals of Database Systems, Page 59ff

SIMPLE ENTITIES AND ATTRIBUTES
RM: RELATIONAL MODEL

- A relation (or relation state) r of the relation schema $R\ (A_1, A_2, \ldots\ A_n)$, also denoted by $r(R)$, is a set of m-tuples

$$r = \ (t_1, t_2, \ldots\ t_m)$$

- Each n-tuple t is an ordered list of n values $t = <\ v_1, v_2, \ldots\ v_n >$, where each value $v_i, 1\ \leq i\ \leq n$, is an element of $dom\ (A_i)$ or is a special NULL value

- The i^{th} value in tuple t, which corresponds to the attribute A_i, is referred to as $t[A_i]$ or $t.A_i$ (or $t[i]$ if we use the positional notation)

Source: Elmasri, Fundamentals of Database Systems, Page 59ff

A relation is a SET (of rows)

- no order, no row number
- no duplicates

SIMPLE ENTITIES AND ATTRIBUTES
RM: RELATIONAL MODEL - EXAMPLE

Table name / Relation

BOOK

ISBN	Title	Author	Publisher	Year	Price
978-1-292-09761-9	Fundamentals of Database Systems	Ramez Elmasri	Prentice Hall	2016	59.99
978-0321197849	An Introduction to Database Systems	C. J. Date	Pearson	2003	69.92
…	…	…	…	…	…

Row / Tuple: distinct tuples

Column (col) / Attribute: attribute values

SIMPLE ENTITIES AND ATTRIBUTES
RM: RELATIONAL MODEL

- A relation (or relation state) $r(R)$ is a mathematical relation of degree n on the domains dom(A_1), dom(A_2), ..., dom(A_n), which is a subset of the Cartesian product (denoted by $\times$) of the domains that define R :

$$r(R) \subseteq (\text{dom}(A_1) \times \text{dom}(A_2) \times \ldots \times \text{dom}(A_n))$$

- If $|D|$ is the total number of values in a domain D, the total number of tuples in the Cartesian poduct is

$$|dom\,(A_1)| \times |dom\,(A_2)| \times \ldots \times |dom\,(A_n)|$$

Source: Elmasri, Fundamentals of Database Systems, Page 59ff

SIMPLE ENTITIES AND ATTRIBUTES
RM: RELATIONAL MODEL

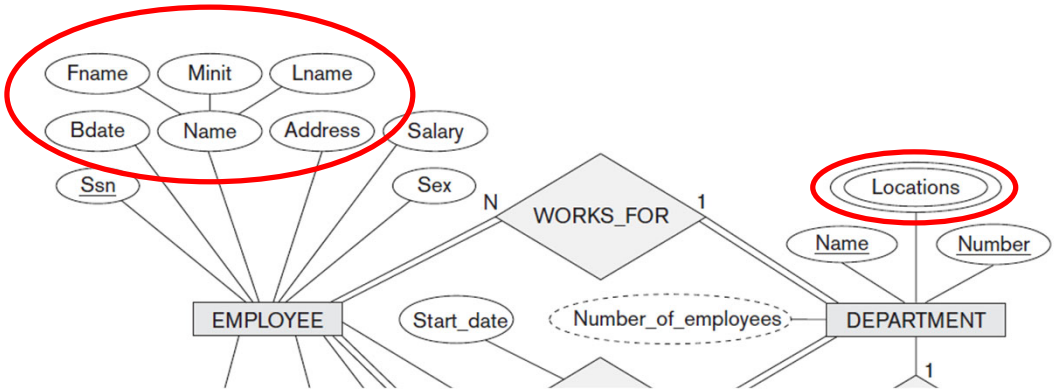
- Ordering of tuples
  - A relation is defined as a set of tuples
  - Thus, tuples in a relation do not have any order
  - In a file, records are physically stored on disk and thus have an order

- Ordering of values within a tuple
  - An n-tuple is an ordered list of n values, so the ordering of values in a tuple is important

Source: Elmasri, Fundamentals of Database Systems, Page 59ff

SIMPLE ENTITIES AND ATTRIBUTES
RM: RELATIONAL MODEL

- Values and NULLs in tuples
  - Each value in a tuple is an atomic value
  - Hence, composite (and multivalued) attributes are not allowed
  - This model is sometimes called the *flat relational model*



→ multivalued attributes must be represented by separate relations, and composite attributes are represented only by their simple component attributes in the basic relational model

  - NULL values are used for values that may be unknown or may not apply to a tuple

- Relations may represent entitiy types and relationship types from ERM

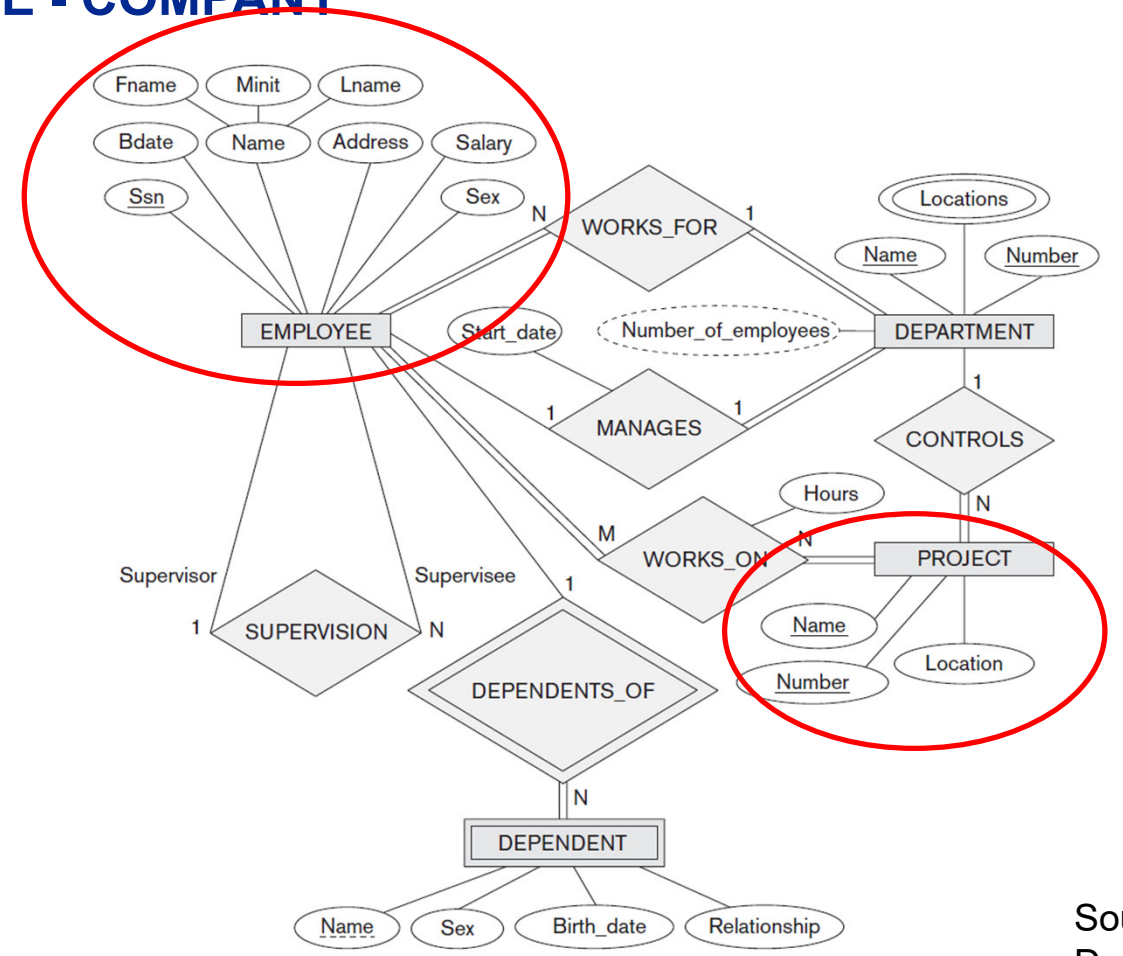
Source: Elmasri, Fundamentals of Database Systems, Page 59ff

SIMPLE ENTITIES AND ATTRIBUTES
RM: RELATIONAL MODEL

- A relational schema R of degree n is denoted by $R\ (A_1, A_2, \ldots\ A_n)$
- The uppercase letters Q, R, S denote relational names
- The lowercase letters q, r, s denote relation states
- The letters t, u, v denote tuples
- In general, the name of a relation schema such as BOOK also indicates the current set of tuples in that relation (the current relation state) whereas STUDENT(Name, Ssn, ...) refers only to the relation schema
- An attribute A can be qualified with the relation's name R to which it belongs by using the dot notation R.A - for example, BOOK.title
- An n-tuple t in a relation $r(R)$ is denoted by $t = <\ v_1, v_2, \ldots, v_n\ >$
  - $t[A_i]$ and $t.A_i$ refer to the value v_i in t
  - $t[A_u, A_w, \ldots, A_z]$ and $t.(A_u, A_w, \ldots, A_z)$ refer to a subtuple in t

Source: Elmasri, Fundamentals of Database Systems, Page 59ff

SIMPLE ENTITIES AND ATTRIBUTES
RM: EXAMPLE - COMPANY



Source: Elmasri, Fundamentals of Database Systems, Page 204 ff

SIMPLE ENTITIES AND ATTRIBUTES
RM: EXAMPLE - COMPANY

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary

PROJECT

Pname	Pnumber	Plocation

Source: Elmasri, Fundamentals of Database Systems, Page 59ff

SIMPLE ENTITIES AND ATTRIBUTES
RM: CONSTRAINTS

Three categories

1. Constraints that are inherent in the data model
   → *inherent model-based constraints* or *implicit constraints*
   Example: no duplicate tuples in a relation
2. Constraints that can be directly expressed in schemas of the data model
   → *schema-based constraints* or *explicit constraints*
   Example: Domain constraints, **key constraints**, constraints on NULL, entity integrity constraints and referential integrity constraints
3. Constraints that cannot be directly expressed in the schemas of the data model, and hence must be expressed and enforced by the application programs
   → *application-based* or *semantic constraints* or *business rules*

Source: Elmasri, Fundamentals of Database Systems, Page 59ff

SIMPLE ENTITIES AND ATTRIBUTES
RM: CONSTRAINTS – KEYS

- There are subsets of attributes of a relation schema R with the property that no two tuples in any relation state r of R should have the same combination of values for these attributes $t_1\,[SK] \neq\ t_2\,[SK]$
- Any such set of attributes SK is called a *super key* of a relation
  → A super key specifies a *uniqueness constraint*
- A minimal super key, that is, a superkey from which we cannot remove any attributes and still have the uniqueness constraint in condition 1 hold, is called candidate key
- For every relation, one of the candidate keys is chosen as the primary key of the relation

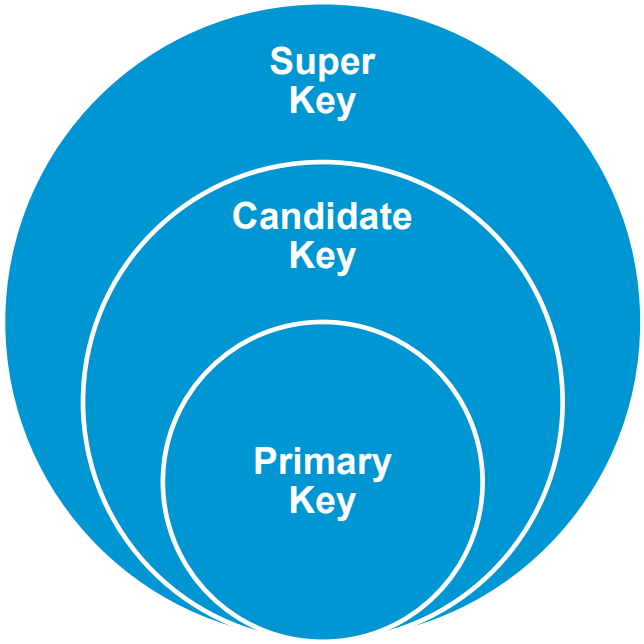
Source: Elmasri, Fundamentals of Database Systems, Page 59ff

SIMPLE ENTITIES AND ATTRIBUTES
RM: KEY ATTRIBUTES

- **Super Key**: An attribute or a set of attributes that uniquely identifies a tuple within a relation

- **Candidate Key (CK):** A super key, so that no proper subset is a super key within the relationship

- **Primary Key (PK):** The candidate key that is selected to identify tuples uniquely within the relation;
  The candidate keys which are not selected as PKs are called ”Alternate Keys”



SIMPLE ENTITIES AND ATTRIBUTES
RM: KEY ATTRIBUTES

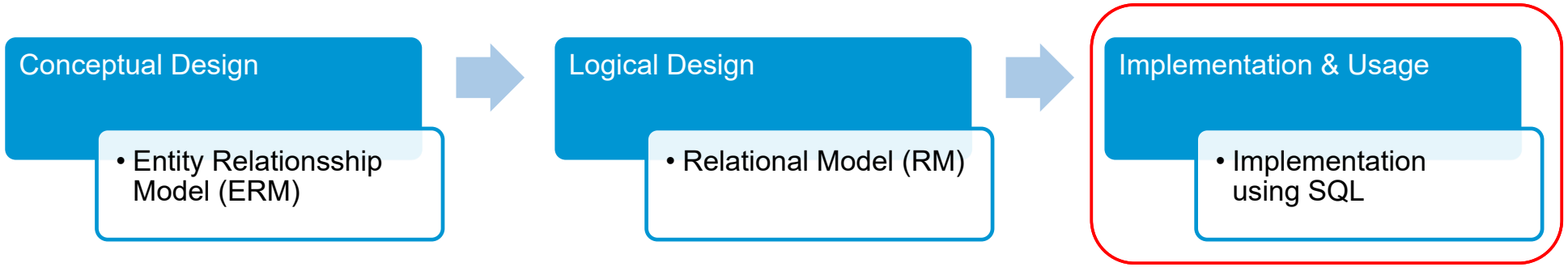
- Primary Key
  - Also called *Entity Integrity Constraint*
  - PK values must be unique and cannot be NULL!
  - Notation: underlined

ISBN	Title	Author	Publisher	Year	Price
978-1-292-09761-9	Fundamentals of Database Systems	Ramez Elmasri	Prentice Hall	2016	59,99
978-0321197849	An Introduction to Database Systems	C. J. Date	Pearson	2003	69,92
…	…	…			

SIMPLE ENTITIES AND ATTRIBUTES
RM: KEY ATTRIBUTES – ARTIFICIAL KEYS

- PNo is an example for an artificial key
  → Also called: surrogate key, technical key
- Key is an attribute not natural for the entity
- Many RDBMS offer identity/serial data types:
  - Number
  - Automatically inserted values
  - Values taken from sequences
- In most cases, business keys are needed, too!
  - A business key is a natural key, i.e., something unique about each tuple
  - Artificial key should be no excuse for not defining unique attributes!
- Artificial Keys may evolve to business keys
  → ISBN, Social Security Number / Passport Number

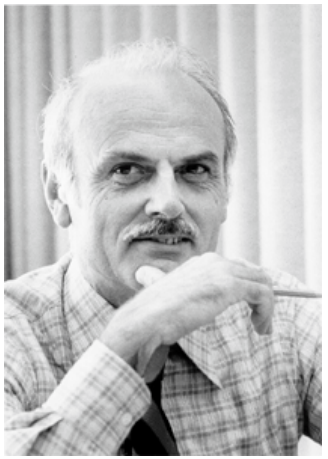
SIMPLE ENTITIES AND ATTRIBUTES
DATABASE DESIGN



SIMPLE ENTITIES AND ATTRIBUTES
SQL: DATABASE DESIGN

- Physical Database Design
  - The primary goal of physical database design is data processing efficiency (as costs for computer technology are decreasing)
  - Implementation of the logical database design in a concrete schema by using SQL including the relational database schema and external views

SIMPLE ENTITIES AND ATTRIBUTES
SQL: DATABASE DESIGN - CODD'S TWELVE RULES



- Define the criteria for a DBMS to be a relational DBMS (RDBMS)
- Very strict (maybe too strict?)
  → None of the popular DBMS fulfils all rules
  - → Especially rules 6, 9, 10, 11, and 12 are difficult to fulfill
  - → Therefore, many manufacturers describe their database as relational if it meets only some of the most important criteria

Source: www.wikipedia.org

SIMPLE ENTITIES AND ATTRIBUTES
SQL: DATABASE DESIGN - CODD'S TWELVE RULES

Rule 0	• **The foundation rule**
Rule 1	• **The information rule**
Rule 2	• **The guaranteed access rule**
Rule 3	• **Systematic treatment of NULL values**
Rule 4	• **Dynamic online catalog based on the relational model**
Rule 5	• **The comprehensive data sublanguage rule**
Rule 6	• The View updating rule
Rule 7	• **Possible for high-level insert, update, and delete**
Rule 8	• **Physical data independence**
Rule 9	• Logical data independence
Rule 10	• Integrity independence
Rule 11	• Distribution independence
Rule 12	• The nonsubversion rule

SIMPLE ENTITIES AND ATTRIBUTES
SQL: SQL HISTORY

- SQL may commercial be considered one of the major reasons for the commercial success relational databases
- SQL → Structured Query Language
- SEQUEL: In 1981, SEQUEL was designed and implemented at IBM Research as the interface for an experimental relational database system called SYSTEM R
- SQL-86 or SQL1: developed by ANSI and ISO
  - Standardized data types, query syntax
  - **`BOOLEAN`**, structured types (classes), recursive queries, ...
- SQL-92 or SQL2
  - **`BLOBS`**, **`VARCHAR`**, **`DATE`**, **`TIME`**, **`TIMESTAMP`**
  - consistence checks
  - modifications of data structures

SIMPLE ENTITIES AND ATTRIBUTES
SQL: SQL HISTORY

- SQL-1999 or SQL3
  - User defined types, object concepts (analogues to classes,…)
- SQL:2003
  - Java: SQLJ + JDBC
  - Stored Procedures (PSM)
  - sequence generator, auto-generated values, MERGE
- SQL-2008
  - SQL:2008: **`TRUNCATE TABLE`**, XML/XQuery support,...
- SQL:2011
  - improved support for temporal databases, Roles, OLAP-Supporting
  - requests: **`ROLLUP`**, **`GROUPING SETS`**, **`CUBE`**

SIMPLE ENTITIES AND ATTRIBUTES
SQL: SQL BASICS

- SQL has facilities for
  - Defining views on the database
  - Specifying security and authorization
  - Defining integrity constraints
  - Specifying transaction controls

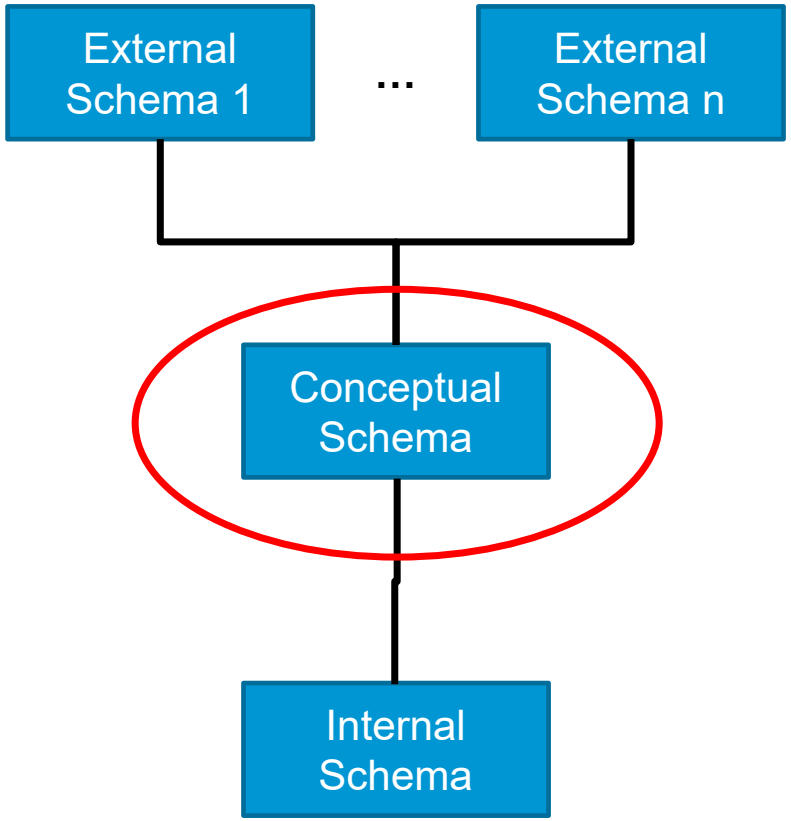
- It also has rules for embedding SQL statements into a general-purpose programming language such as Java, COBOL, or C/C++

Source: Elmasri, Fundamentals of Database Systems, Page 88ff

SIMPLE ENTITIES AND ATTRIBUTES
SQL: SQL BASICS

- Interactive
  - SQL*PLUS, psql, ... GUI: sqldeveloper, pgadmin, squirrel SQL...
- Embedded SQL
  - SQL commands embedded in 3GL (C, Java)
  - Native libraries (vendor specific)
- ODBC (Open Database Connectivity)
  - very popular in MS Windows
  - but can be used under Unix, too
- JDBC (Java Database Connectivity)
  - Part of the standard Java API

SIMPLE ENTITIES AND ATTRIBUTES
SQL: SQL BASICS – EXAMPLES SQL COMMANDS




```
create view
drop view
```

Database:
```
create database
drop database
open database
```

Table:
```
create table
drop table
alter table
```

Domains:
```
create domain
drop domain
alter domain
```

```
create index
drop index
alter index
```

SIMPLE ENTITIES AND ATTRIBUTES
SQL: SQL BASICS

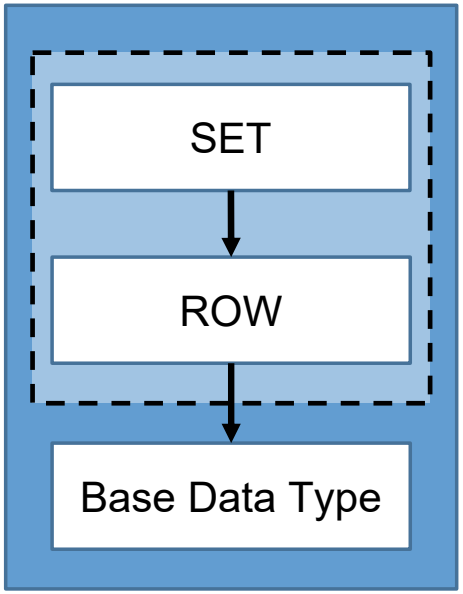
- SQL Keywords: case insensitive

- Convention: Upper Case (e.g., **`SELECT, UPDATE`**)
- Commands end with ;
  (when entered interactively)

- Comments:
  - line comment: −− this is a comment
  - multiline comment: /* comment */

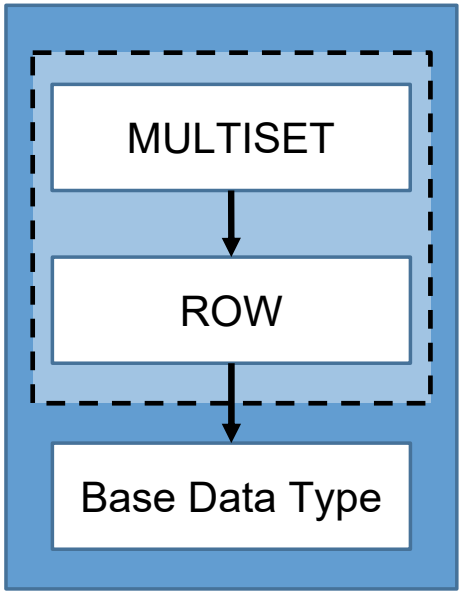
SIMPLE ENTITIES AND ATTRIBUTES
SQL: SQL BASICS

Relational Data Model



SET: no order,
homogeneous elements,
no duplicates

SQL Data Model



MULTISET: no order,
homogeneous elements,
duplicates allowed

SIMPLE ENTITIES AND ATTRIBUTES
SQL: SQL BASICS - EXCURSION

- Syntax Definition: BNF (Backus-Naur Form)

Symbol	Semantics
::=	Is defined by
|	Alternative
{ }	Grouping of alternatives
[ ]	Optional
*	Repeating element ≥ 0
+	Repeating element ≥ 1
< >	Syntactical variable (non-terminal symbol)

SIMPLE ENTITIES AND ATTRIBUTES
SQL: SQL BASICS - EXCURSION

- Syntax Definition: BNF (Backus-Naur Form)

< digit >	::=	0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
< hexit >	::=	< digit > | A | B | C | D | E | F | a | b | c | d | e | f
< sayhello >	::=	Hello { world | IE4 } !
< imtired >	::=	I am [very [, very]*] tired.
< column constraint >	::=	NOT NULL | < unique specification> | < references specification> | < check constraint definition>

SIMPLE ENTITIES AND ATTRIBUTES
SQL: CREATE SCHEMA

- A schema
  - groups together tables and other constructs that belong to the same database application
  - is identified by a schema name
  - includes an authorization identifier and descriptors for each element
- A schema is essentially a namespace
- Schema elements include tables, constraints, views, domains, and other constructs (such as authorization grants) that describe the schema

Source: Elmasri, Fundamentals of Database Systems, Page 88ff

SIMPLE ENTITIES AND ATTRIBUTES
SQL: CREATE SCHEMA

- Syntax:

```
CREATE [ OR REPLACE ]
       { DATABASE | SCHEMA }
       [ IF NOT EXISTS ]
       db_name
       [ create_specification ] ...
```

- Example:

```
CREATE SCHEMA COMPANY ;
```

- Attention: User must be authorized to create schema and schema elements

Source: Elmasri, Fundamentals of Database Systems, Page 88ff

SIMPLE ENTITIES AND ATTRIBUTES
SQL: CREATE TABLE

- A new relation with a name, its attributes and initial constraints
- Each attribute is defined by a name, a data type and constraints (e.g., **`NOT NULL`**)
- Following the attributes, the primary key, entity integrity, and referential integrity constraint can be specified (alternatively, they can be specified with **`ALTER TABLE`**)
- All relations created by **`CREATE TABLE`** are called *base tables*, i.e., the relation and its tuples are created and stored as a file by the DBMS

Source: Elmasri, Fundamentals of Database Systems, Page 88ff

SIMPLE ENTITIES AND ATTRIBUTES
SQL: CREATE TABLE

Syntax for creating an empty table:

```
CREATE TABLE <relationname>
      (<column> <type> [ DEFAULT expr]
            [ [NOT] NULL ] [ colconstraint ] *
      [,{<column> <type> [ DEFAULT expr ]
            [ [NOT] NULL ] [ colconstraint ] *
       | <tableconstraint> } ] *
      ) ;
```

Source: Elmasri, Fundamentals of Database Systems, Page 88ff

SIMPLE ENTITIES AND ATTRIBUTES
MAPPING OF ERM TO RELATIONAL MODEL

Seven Steps

1. Mapping of regular entity types
2. Mapping of weak entity types
3. Mapping of binary 1:1 relationships
4. Mapping of binary 1:n relationships
5. Mapping of binary m:n relationships
6. Mapping of multivalued attributes
7. Mapping of n-ary relationships

→ Later!!!

Source: Elmasri, Fundamentals of Database Systems, Page 286ff

SIMPLE ENTITIES AND ATTRIBUTES
1. MAPPING OF REGULAR ENTITY TYPES

- For each regular (strong) entity type E in the ER schema,
  create a relation R that includes all the simple attributes of E
- Include only the simple component attributes of a composite attribute
- Choose one of the key attributes of E as the primary key for R
- If the chosen key of E is a composite,
  then the set of simple attributes that form it will together form the primary key of R

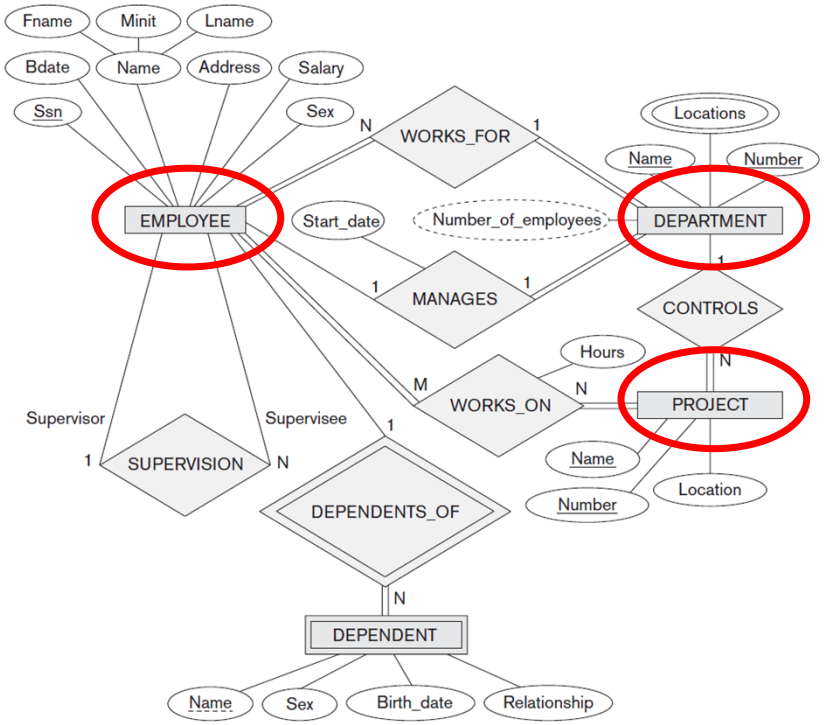
Source: Elmasri, Fundamentals of Database Systems, Page 286ff

SIMPLE ENTITIES AND ATTRIBUTES
1. MAPPING OF REGULAR ENTITY TYPES

EMPLOYEE	Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary

DEPARTMENT	Dname	Dnumber

PROJECT	Pname	Pnumber	Plocation



Source: Elmasri, Fundamentals of Database Systems, Page 286ff

SIMPLE ENTITIES AND ATTRIBUTES
SQL: CREATE TABLE

Example:

```
CREATE TABLE COMPANY.Employee ... ;
```

or

```
USE DATABASE COMPANY ;
CREATE TABLE Employee ... ;
```

Source: Elmasri, Fundamentals of Database Systems, Page 88ff

SIMPLE ENTITIES AND ATTRIBUTES
SQL: CREATE TABLE

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

```
CREATE TABLE Employee
    (Fname       VARCHAR(15)      NOT NULL,
    Minit        CHAR,
    Lname        VARCHAR(15)      NOT NULL,
    Ssn          CHAR(9)          NOT NULL,
    Bdate        DATE,
    Address      VARCHAR(30),
    Sex          CHAR,
    Salary       DECIMAL(10 ,2),

    PRIMARY KEY (Ssn) );
```

Source: Elmasri, Fundamentals of Database Systems, Page 88ff

SIMPLE ENTITIES AND ATTRIBUTES
SQL: CREATE TABLE – DATA TYPES

Syntax for creating an empty table:

```
CREATE TABLE <relationname>
       (<column> <type> [ DEFAULT expr]
              [ [NOT] NULL ] [ colconstraint ] *
       [,{<column> <type> [ DEFAULT expr ]
              [ [NOT] NULL ] [ colconstraint ] *
        | <tableconstraint> } ] *
       ) ;
```

Source: Elmasri, Fundamentals of Database Systems, Page 88ff

SIMPLE ENTITIES AND ATTRIBUTES
SQL: CREATE TABLE – DATA TYPES

- Numeric
- Strings
- Temporal
- SQL-99: CLOB, BLOB, BOOLEAN

Source: Elmasri, Fundamentals of Database Systems, Page 88ff

SIMPLE ENTITIES AND ATTRIBUTES
SQL: CREATE TABLE – DATA TYPES

- Numeric
  - Integer: **`INTEGER`** or **`INT, SMALLINT, BIGINT`**
  - Floating-point: **`FLOAT`** or **`REAL, DOUBLE PRECISION`**
- Formatted numbers: **`DECIMAL`**`(i,j)` or **`NUMERIC`** `(i,j)` where
  - i → precision (total number of decimal digits)
  - j → scale (digits after the decimal point)
  - Oracle: `NUMBER` [(`precision, scale`)]
- Example:
  - **`DECIMAL`** `(10,2)` : values up to 99,999,999.99
  - **`NUMERIC`** `(9,2)` : 1746352.32
  - **`NUMERIC`** `(6)` : not possible

Source: Elmasri, Fundamentals of Database Systems, Page 88ff

SIMPLE ENTITIES AND ATTRIBUTES
SQL: CREATE TABLE – DATA TYPES

- Fixed length:
  - **CHARACTER**(n) or **CHAR**(n)
  - fills up with spaces if not full length is used
- Variable length:
  - **CHARACTER VARYING** (n) or **CHAR VARYING** (n) or **VARCHAR** (n)
  - Oracle: **VARCHAR2**(n)
  - Example: **VARCHAR**(15)
- Value is placed between apostrophes, e.g., 'abc'
- **CHARACTER SET** / **CHARSET** has to be defined or standard charset of DBMS is used → e.g., UNICODE UTF-8

Source: Elmasri, Fundamentals of Database Systems, Page 88ff

SIMPLE ENTITIES AND ATTRIBUTES
SQL: CREATE TABLE – DATA TYPES

- `DATE` (10 positions): `YYYY-MM-DD`
- `TIME` (8 positons): `HH:MM:SS`
- `DATETIME`: `YYYY-MM-DD HH:MM:SS`
- `TIMESTAMP`: `YYYY-MM-DD HH:MM:SS.ssssss`
- Example:
  - `DATE '2008-09-27'`
  - `TIME '09:12:47'`
- Define a point in time
  - Syntax: `TIMESTAMP [(precision)] [WITH TIME ZONE]`
- INTERVAL:
  - Defines a duration or a time difference

Source: Elmasri, Fundamentals of Database Systems, Page 88ff

SIMPLE ENTITIES AND ATTRIBUTES
SQL: CREATE TABLE – DATA TYPES

- CLOB: Character Large Object
  → Very long texts
  → in KB, MB, GB

- BLOB: Binary Large Object
  → Long Binary Data (e.g, pictures, video)

- BOOLEAN
  → MySQL: TINYINT(1)

Source: Elmasri, Fundamentals of Database Systems, Page 88ff

SIMPLE ENTITIES AND ATTRIBUTES
SQL: CREATE TABLE – DATA TYPES

Source: https://docs.oracle.com/cd/B19306_01/gateways.102/b14270/apa.htm

Microsoft SQL Server	Oracle
BIGINT	NUMBER(19)
BINARY	RAW
BIT	NUMBER(3)
CHAR	CHAR
DATETIME	DATE
DECIMAL	NUMBER(p[,s])
FLOAT	FLOAT(49)
IMAGE	LONG RAW
INTEGER	NUMBER(10)
MONEY	NUMBER(19,4)
NCHAR	NCHAR
NTEXT	LONG
NVARCHAR	NCHAR
NUMERIC	NUMBER(p[,s])
REAL	FLOAT(23)
SMALL DATETIME	DATE
SMALL MONEY	NUMBER(10,4)
SMALLINT	NUMBER(5)
TEXT	LONG
TIMESTAMP	RAW
TINYINT	NUMBER(3)
UNIQUEIDENTIFIER	CHAR(36)
VARBINARY	RAW
VARCHAR	VARCHAR2

SIMPLE ENTITIES AND ATTRIBUTES
SQL: CREATE TABLE – DATA TYPES USED IN THIS LECTURE AND LABS

- For text:
  → **`VARCHAR`**`(n)`

- For time information:
  → **`TIMESTAMP, DATE`**

- For numbers:
  → **`INT`**
  → **`DECIMAL`**`(p,s)`

SIMPLE ENTITIES AND ATTRIBUTES
SQL: CREATE TABLE – PRIMARY KEY (PK)

- PK identifies every tuple uniquely
- Entity Integrity
- One PK for each table
- PK is (implicit)
  - **`NOT NULL`**
  - **`UNIQUE`** (no duplicates)

SIMPLE ENTITIES AND ATTRIBUTES
SQL: CREATE TABLE – PRIMARY KEY (PK)

Syntax:

- As Column Constraint
  → Only if the primary key is one single attribute (not combined)

```
[ CONSTRAINT <constraintname> ] PRIMARY KEY
```

- As Table Constraint

```
[ CONSTRAINT <constraintname>] PRIMARY KEY
   ( <column>[ , <column>] * )
```

SIMPLE ENTITIES AND ATTRIBUTES
SQL: CREATE TABLE – PRIMARY KEY (PK)

Example Column Constraint:

```
CREATE TABLE Department
(Dname              VARCHAR(15)   NOT NULL,
Dnumber             INT           PRIMARY KEY,
Mgr_ssn             CHAR(9)       NOT NULL,
Mgr_start_date      DATE
);
```

Source: Elmasri, Fundamentals of Database Systems, Page 88ff

SIMPLE ENTITIES AND ATTRIBUTES
SQL: CREATE TABLE – PRIMARY KEY (PK)

Example Table Constraint:

```
CREATE TABLE Department
(Dname              VARCHAR(15)   NOT NULL,
Dnumber             INT           NOT NULL,
Mgr_ssn             CHAR(9)       NOT NULL,
Mgr_start_date      DATE,
PRIMARY KEY ( Dnumber )
);
```

Source: Elmasri, Fundamentals of Database Systems, Page 88ff

SIMPLE ENTITIES AND ATTRIBUTES
SQL: INSERT

- Syntax:

```
INSERT INTO < table >
      [ ( < column > [ , … ] ) ]
      VALUES ( < expression > [ , ...] )
```

- Column list is optional
  - If omitted, values list must match table's attributes
  - If given, we don't have to specify values for all columns
    → Other columns will get the **DEFAULT** value (or NULL)

SIMPLE ENTITIES AND ATTRIBUTES
SQL: INSERT

- There are 2 possibilities for inserting:
  1. Constant tuples
  2. Tuples returned by a query → later…

SIMPLE ENTITIES AND ATTRIBUTES
SQL: INSERT

- There are 2 possibilities for inserting:
  1. Constant tuples
     - Made from literals
     - ... or from function calls, variables
     - For every attribute with a `NOT NULL` constraint and no `default clause`, you must define values
     - E.g., 3+5, current timestamp

Source: Elmasri, Fundamentals of Database Systems, Page 97ff

SIMPLE ENTITIES AND ATTRIBUTES
SQL: INSERT

- There are 2 possibilities for inserting.
  1. Constant tuples
     - Example:

```
INSERT INTO EMPLOYEE
VALUES ( 'Arthur' , 'C', 'Brown' , 323232323,
'1970-12-31', 'London', 'm', 45000, 3334455555, 5 ) ;


INSERT INTO EMPLOYEE ( fname, lname, ssn, super_ssn, dno)
VALUES ( 'Andi' , 'Red', 343434343, 333445555, 5) ;
```

Source: Elmasri, Fundamentals of Database Systems, Page 97ff

SIMPLE ENTITIES AND ATTRIBUTES
SQL: INSERT

Person	PNR	Name	Fname	BornIn
	1	Miller	Olaf	Hamburg
	2	Meier	Stefan	Berlin
	4	Schulz	Olaf	Hamburg
	3	Miller	Karina	Wien

Example Constant tuples:

```
INSERT INTO Person
       VALUES ( 1 , 'Miller' , 'Olaf' , 'Hamburg' ) ;
INSERT INTO Person (PNr, Name, Fname, BornIn)
       VALUES ( 2 , 'Meier' , 'Stefan' , 'Berlin' ) ;
INSERT INTO Person (PNr, BornIn, Fname, Name)
       VALUES ( 4 , 'Hamburg' , 'Olaf' , 'Schulz' ) ;
INSERT INTO Person
       VALUES ( 3 , 'Miller' , 'Karina' , 'Wien' ) ;
```

SIMPLE ENTITIES AND ATTRIBUTES
SQL: UPDATE

- Syntax:

```
UPDATE < table >
SET < column > = < expression >
[ WHERE < condition >]
```

- Used to modify attribute values of one or more selected tuples
- Can modify only tuples of one table at a time
- **WHERE** clause: optional!
  → If left out: Update all tuples
- Note: updating a primary key value may propagate to the foreign key values of tuples in other relations if such a referential triggered action
  is specified in the referential integrity constraints of the DDL

Source: Elmasri, Fundamentals of Database Systems, Page 97ff

SIMPLE ENTITIES AND ATTRIBUTES
SQL: UPDATE

Examples:

```
UPDATE Person
SET    lname= 'Brown' , married = TRUE
WHERE  id = 45 ;
```

```
UPDATE Employee
SET    salary = salary * 1.1 ;
```

```
UPDATE Person
SET    email = NULL
WHERE  email IS NOT NULL ;
```

SIMPLE ENTITIES AND ATTRIBUTES
SQL: SELECT

Data Manipulation Language (DML)

```
INSERT INTO … VALUES ( … )
UPDATE … SET … [ WHERE … ]
DELETE FROM … [ WHERE … ]
```

Data Query Language (DQL)

```
SELECT … FROM … [ WHERE … ] …
```

- Data Query Language is used to extract data from the database
- It doesn't modify any data in the database
- There is only one basic statement: **SELECT**

Source: Elmasri, Fundamentals of Database Systems, Page 145ff

SIMPLE ENTITIES AND ATTRIBUTES
SQL: SELECT

SELECT – Basic form

```
SELECT  <attribute list>
FROM    <table list>
WHERE   <condition>
```

- `<attribute list>` is a list of attribute names (columns) whose values are to be retrieved by the query
- `<table list>` is a list of the relation names (e.g., tables) required to process the query
- `<condition>`: optional conditional (Boolean) expression that identifies the tuples to be retrieved by the query

Source: Elmasri, Fundamentals of Database Systems, Page 97ff

SIMPLE ENTITIES AND ATTRIBUTES
SQL: SELECT

Syntax:

```
SELECT [ DISTINCT | ALL ] < attribute_list >
FROM < table list >
[ WHERE < condition > ]
[ <group by clause > ]
[ <having clause > ]
[ UNION [ ALL ] < query specification> ]
[ < order by clause > ]
```

SIMPLE ENTITIES AND ATTRIBUTES
SQL: SELECT

Example:

What does this statement mean???

```
SELECT Bdate, Address
FROM Employee
WHERE Fname = 'John' AND Minit = 'B' AND Lname = 'Smith';
```

Source: Elmasri, Fundamentals of Database Systems, Page 97ff

SIMPLE ENTITIES AND ATTRIBUTES
SQL: SELECT - ATTRIBUTE LIST

- <attribute list> is a list of attribute names (columns) whose values are to be retrieved by the query

  Example:

```
SELECT fname, lname, ssn
FROM   Employee ;
```

- Asterisk (*) stands for: all attributes
  Example:

```
SELECT *
FROM   Employee ;
```

- Arithmetic expressions and aggregation functions are possible

SIMPLE ENTITIES AND ATTRIBUTES
SQL: SELECT - ATTRIBUTE LIST

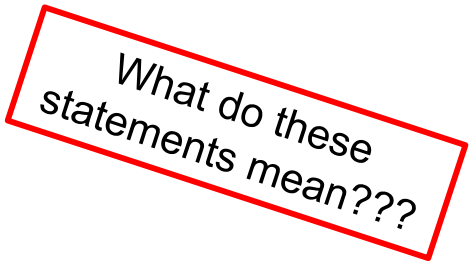
- Syntax:

```
<attribute list>    ::= [<table>.]* <column list>
<column list>       ::= <select sublist>
                        [, <select sublist> ] *
<select sublist>    ::= <element> [[AS] <column name> ]
<element>           ::= [ <table>.] <column_name>
```

- Attributes of the projection can be given directly, if they are unambiguous
- It is always possible to qualify by relation

SIMPLE ENTITIES AND ATTRIBUTES
SQL: SELECT - ATTRIBUTE LIST

Example:




```
SELECT fname AS firstName          -- AS is optional
FROM   Employee ;

SELECT dname
FROM   Department ;

SELECT Employee.ssn, Department.dname
FROM   Employee, Department ;
```

Even if Department has an attribute ”ssn”, the reference is clear

SIMPLE ENTITIES AND ATTRIBUTES
SQL: SELECT - ATTRIBUTE LIST

- SQL uses (mainly) multiset semantics
  - No elimination of duplicates
  - No duplicates wanted: use **DISTINCT**

- Example:

```
SELECT     DISTINCT lname
FROM       Employee ;

SELECT     DISTINCT salary
FROM       Employee ;
```

SIMPLE ENTITIES AND ATTRIBUTES
SQL: SELECT - CONDITION

- <condition> is a conditional (Boolean) expression that identifies the tuples to be retrieved by the query

- Examples:

```
SELECT * FROM Employee WHERE ssn = 333445555 ;
SELECT * FROM Employee WHERE lname IS NULL ;
```

- The **WHERE** clause is optional!
  → If left out: retrieve all tuples
  → If more than one relation is specified in the **FROM** clause and there is no **WHERE** clause, then the Cross Product is selected

Source: Elmasri, Fundamentals of Database Systems, Page 97ff

EXCURSION LOGIC

- Compare two expressions
  - Comparison operators: =, <, <= , >, >=, <> (≠, !=)
  - Expressions could be columns, literals
  - Example:

```
... WHERE age >= 18 ;                -- older than 18
... WHERE last_name <> ’ Miller ’    -- not equals Miller
```

- Check for NULL: **`IS NULL`**

- **`AND`**, **`OR`**, **`NOT`**
  - Example:

```
… WHERE (age >= 18) AND (last_name <> ’Miller’)
```

Source: Elmasri, Fundamentals of Database Systems, Page 97ff

EXCURSION
LOGIC - NULL

- 42 < NULL ?
  → Comparisons against NULL are never true...

- 42 >= NULL ?
  → ... but they are not false, too!

- So, we need a Ternary Logic
  → Values: TRUE, FALSE, NULL

EXCURSION
LOGIC - NULL

- NOT:
  → NOT (NULL) = ? **= NULL**

- AND:
  → TRUE AND NULL = ? **= NULL**
  → FALSE AND NULL = ? **= FALSE**

- OR:
  → TRUE OR NULL = ? **= TRUE**
  → FALSE OR NULL = ? **= NULL**

Source: Elmasri, Fundamentals of Database Systems, Page 97ff

EXCURSION LOGIC - NULL

a	b	a AND b	a OR b	NOT a
0	0	0	0	1
0	1	0	1	1
0	NULL	0	NULL	1
1	0	0	1	0
1	1	1	1	0
1	NULL	NULL	1	0
NULL	0	0	NULL	NULL
NULL	1	NULL	1	NULL
NULL	NULL	NULL	NULL	NULL

HOMEWORK

- Company Example
  - Create for every entity type with its attributes a relation in your DB
  - Insert some sample data in the relations
  - Write some queries to retrieve the data, e.g.,
    - Get all female employees
    - Get all projects located in Hamburg, …
- Think about your own, individual example (e.g., contact list)
  - Create for every entity type with its attributes a relation in your DB
  - Insert some sample data in the relations
  - Write some queries to retrieve the data

Source: Foto von K8 auf Unsplash

ORGANIZATION
OUR JOURNEY IN THIS SEMESTER

- Integrity, Trigger & Security
- Database Applications
- Transactions
- Subqueries & Views
- More Features
- Notations & Guidelines
- Constraints
- **Relationships**
- Simple Entities and Attributes
- Basics

Source: Foto von Justin Kauffman auf Unsplash