Databases

Lecture 3 - Entity-Relationship-Model

Emily Lucia Antosch

HAW Hamburg

07.04.2025

Contents

1. Introduction 2
2. Entity-Relationship-Model 6
3. The relational model 34
4. License Notice 97

1. Introduction

1.1 Where are we right now?

- Last time, we looked at SQL as the language in which we define our database.
- We learnt about different database objects and how they can help us achieve our business requirements.
- Today, we'll look at
  - what an ERM (Entity-Relationship-Model) is,
  - how we can use it to effectively conceptually design databases and
  - why conceptually designing a database prior to implementation can save us a lot of headache.

1.1 Where are we right now?

1. Introduction
2. Basics
3. SQL
4. **Entity-Relationship-Model**
5. Relationships
6. Constraints
7. More SQL
8. Subqueries & Views
9. Transactions
10. Database Applications
11. Integrity, Trigger & Security

1.2 What is the goal of this chapter?

- At the end of this lesson, you should be able to
  - design a database using the ER-model,
  - decide about which attributes, constraints and relations will help you achieve your requirements.

2. Entity-Relationship-Model

What is an ERM

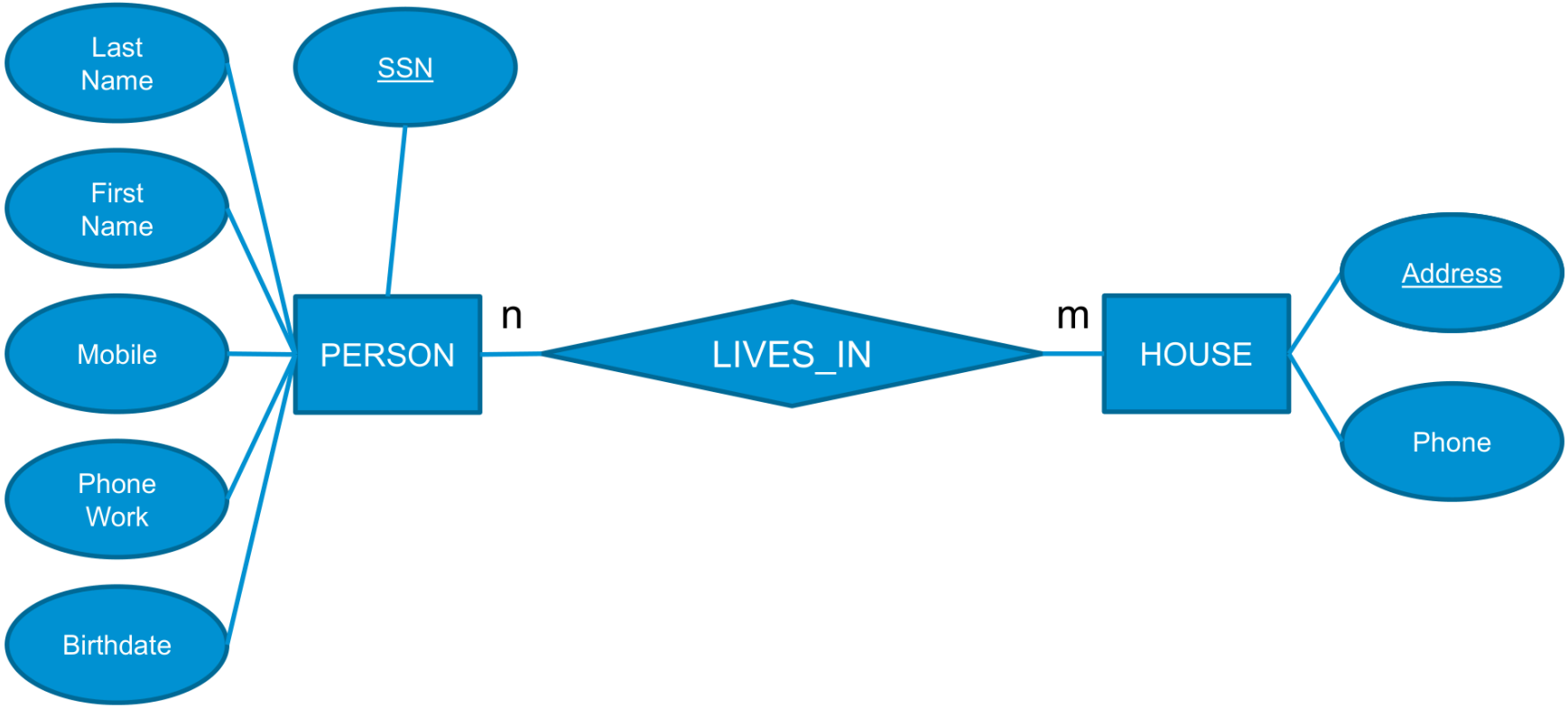
- Entity-Relationship-Model is model/diagram for the logical draft of the database
- The focus is on the business requirements
- This language is not implemented in any DBMS

A quick history of the ERM

- Introduced by Peter Chen in 1976.
- An ERM describes interrelated things of interest in a specific domain of knowledge.
- A basic ERM is composed of entity types (which classify the things of interest) and specifies relationships that can exist between entities (instances of those entity types).
- Elements:
  - Entity: A distinguishable thing existing in the real world.
  - Relationship: Between entities.
  - Attribute: Property of an entity or relationship.

2.1 Basics

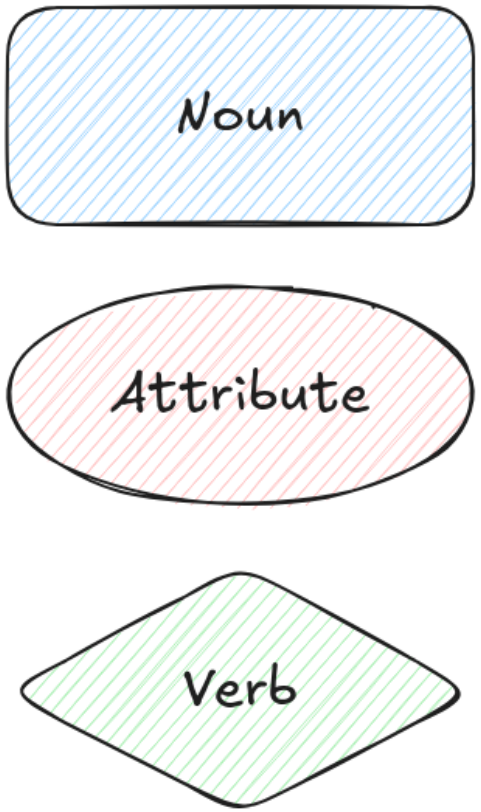
Conceptual Design with ERM



2.1 Basics

Conceptual Design with ERM

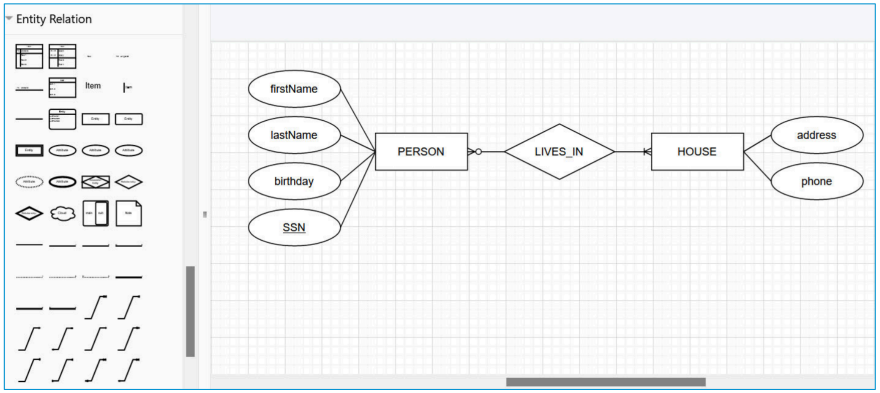
- Entity Type
  - Represented as a rectangle
  - Singular Noun
- Attribute Type
  - Represented as ovals
  - Noun
- Relationship Type
  - Represented as diamond
  - Always between entities
  - Verb & has cardinalities



2.1 Basics

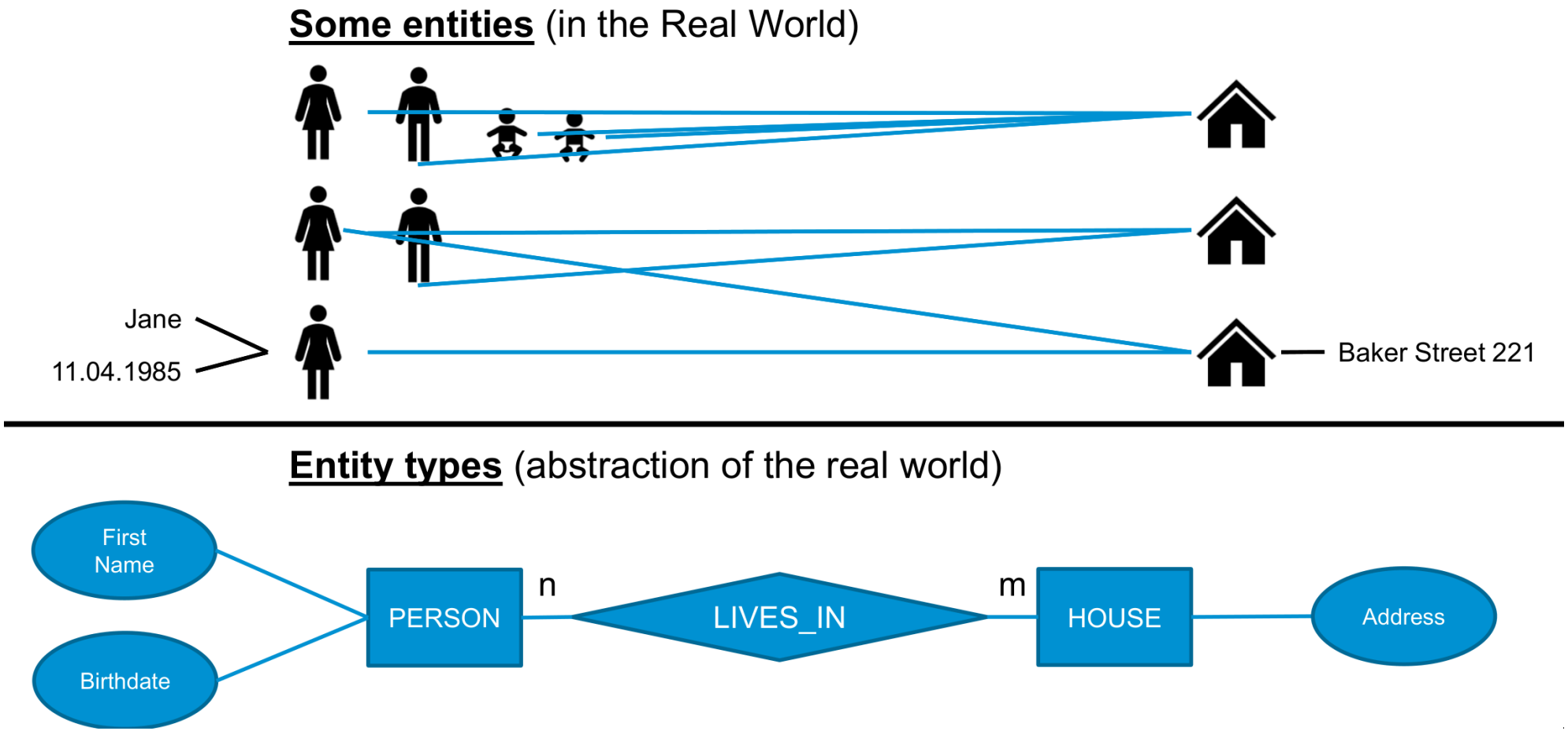
Online-Tools for ERM

- Creating ERMs can be done by using any drawing tool or just a piece of paper and a pen.
- Examples of drawing tools:
  - Excalidraw (Recommended)
  - Draw.io
  - Lucidchart
  - Creately



2.1 Basics

Entity Abstraction



Entity Abstraction

! Memorize

- An **entity** is a distinguishable thing that exists in the real world.

- An abstraction of entities would be an **entity type** (comparable to classes in OOP)
- Several entities make up an **entity set**
- An abstraction of relationships is called a **relationship type**

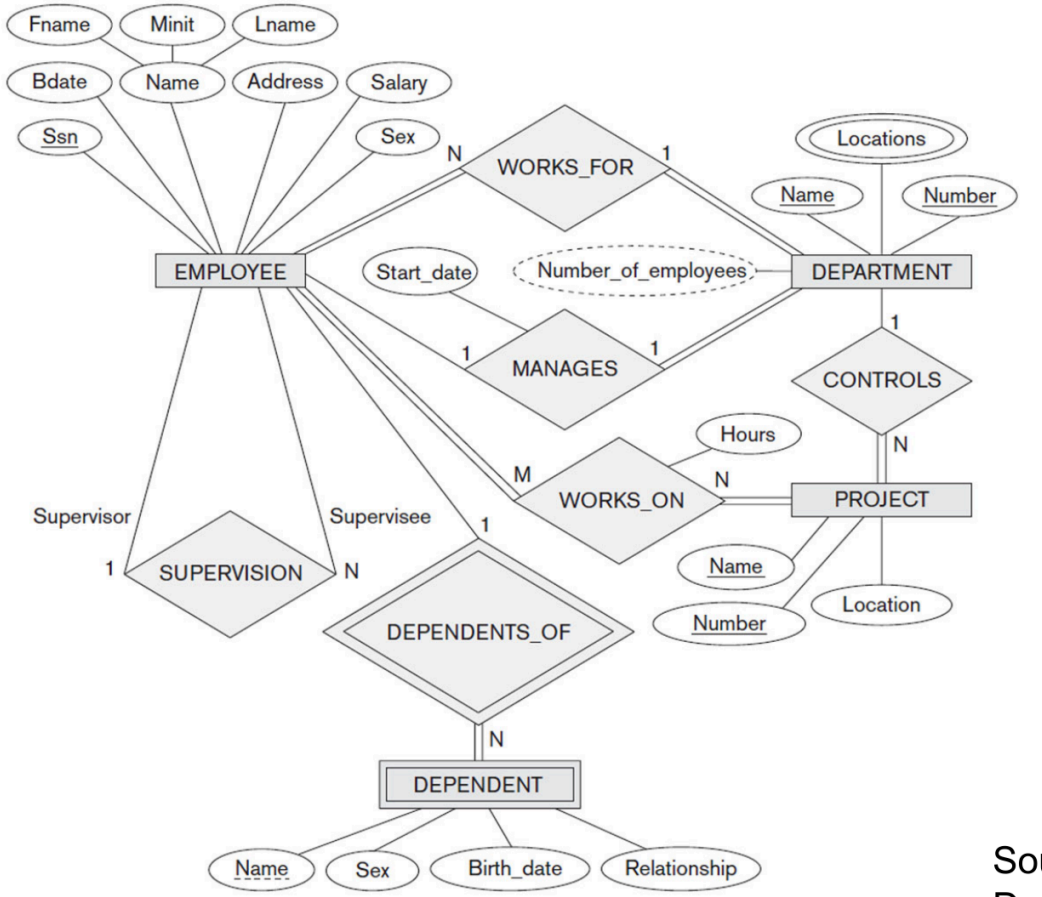
Entity Abstraction

Example

Imagine a company:

- A company is made-up of departments and each department has a unique name, a number and a manager.
- Each employee's name, social security number, address, salary and birth date is stored within our database.
- We also want to keep track of the hours per week per project, keep track of the supervisor.

ERM: Company Example



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

Entity Type

- An **entity type** is a basic object in an ERM.
- Represents a thing in the real world, like a car, a job or a person.
- An entity type has attributes, such as a name, an address or an age.
- A particular entity of that type will have values for each of these attributes.

Entity Type

! Memorize

- An entity type therefore defines a collection of entities, that have the same attributes.
- Each entity type can be defined by its name and its attributes.
- The collection of all entities of a particular entity type, so all the instances of this entity type, is called an **entity set**.

ERM: Entity Example

- Categories for entities could be
  - actual physical objects, people, roles, organizations,
  - actions, interfaces or general information
- An element is not an entity type
  - if it has neither attributes nor relationships,
  - only contains attributes that another entity type already has

? Question

What is a good name for an entity type?

ERM: Entity Example

Task 1

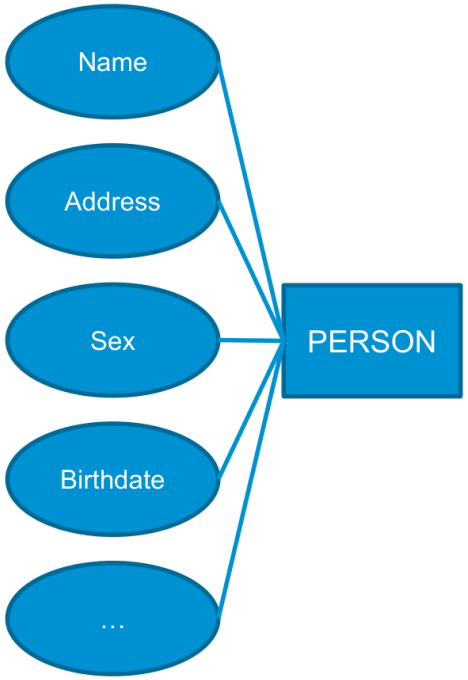
What are the entity types in the following examples?

- A company is organized in departments.
- Departments have a unique name, a unique number, a manager.
- A department oversees a number of projects, each with a name and a number.
- The company may store information about each employee like their name, their social security number and their salary.

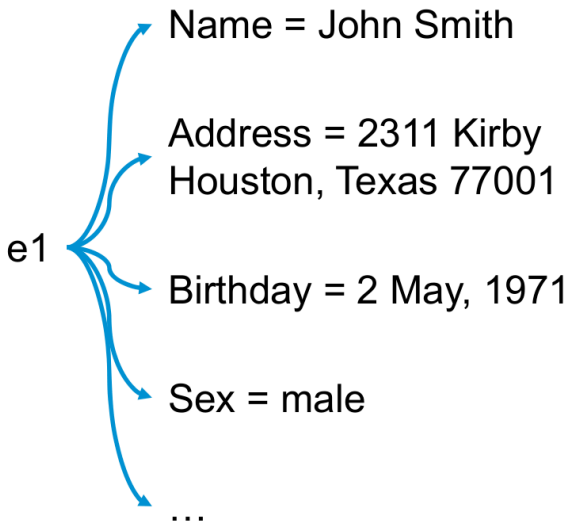
2.1 Basics

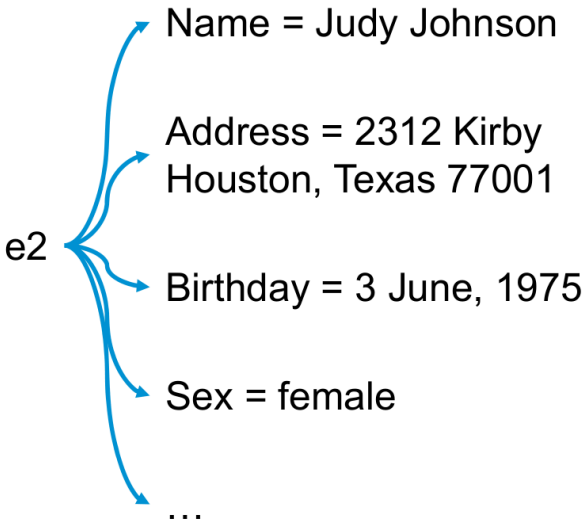
Entity Type

Entity Type



Entities





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

Attributes

- Is the attribute relevant to the problem you are trying to solve?
- An attribute must belong to an entity type (or a relationship type).
- Some of the attributes of an entity are important in identifying the entity. These are called **key attributes**.
- A good name for an attribute is unique within the entity type, but not necessarily across the entire model.

Attributes

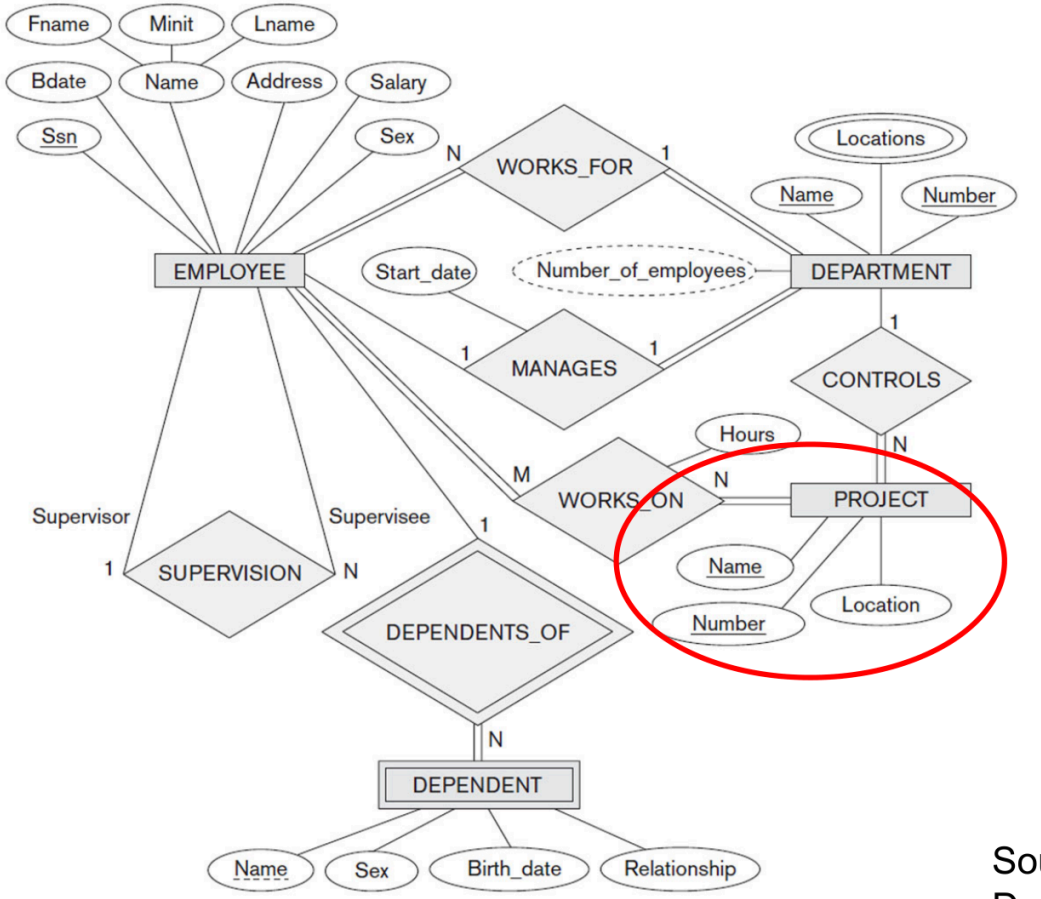
Task 2

When you look at the attributes of the entity project, what could be identifying or key attributes?

- A department controls a number of projects, each with a unique name, unique number and a single location.

2.1 Basics

Attributes



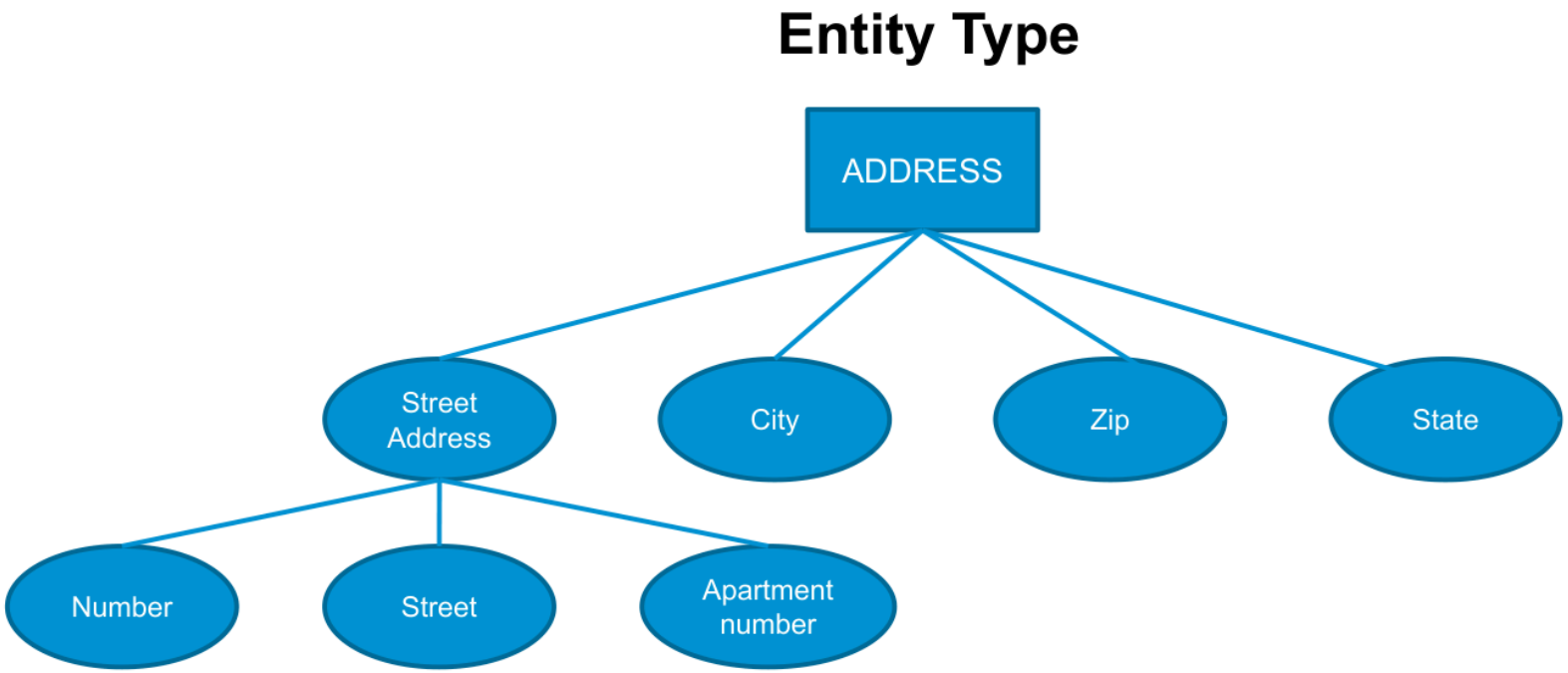
Source: Elmasri, Fundamentals of Database Systems, Page 204 ff

Attributes

- Composite vs. Simple (atomic) attributes
  - Attributes which are not divisible are called simple or atomic attributes
  - Composite attributes can form a hierarchy
  - Composite attributes are useful to model situations in which a user sometimes refers to the composite attribute as a unit but at other times refers specifically to its components
  - If the composite attribute is referenced only as a whole, there is no need to subdivide it into component attributes

- Composite attributes are attached to their component attributes by straight lines

Attributes in Entity Types



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

Key Attributes in entity sets

- 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

Key Attributes in entity sets

Task 3

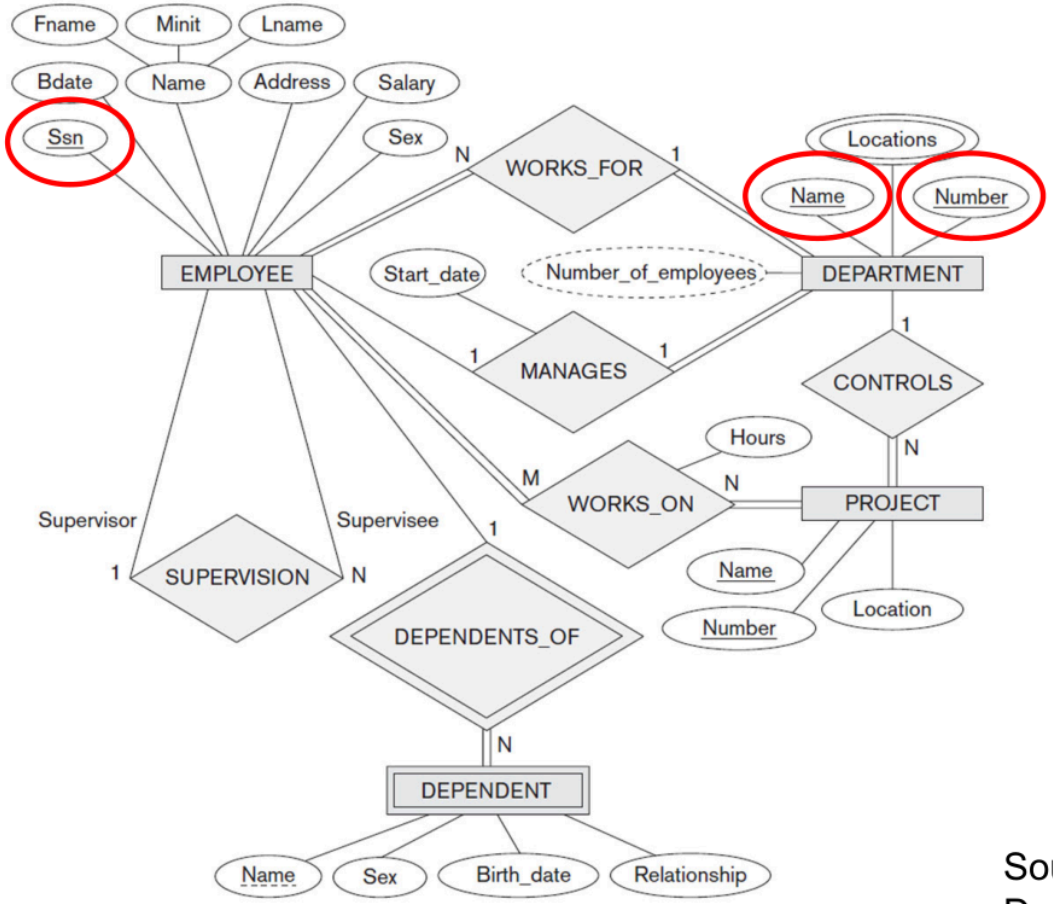
What are key attributes for entity type `EMPLOYEE` and `DEPARTMENT`?

- A company is organized in departments.
- Departments have a unique name, a unique number, a manager.
- A department oversees a number of projects, each with a name and a number.

- The company may store information about each employee like their name, their social security number and their salary.

2.1 Basics

Key attributes in entity sets

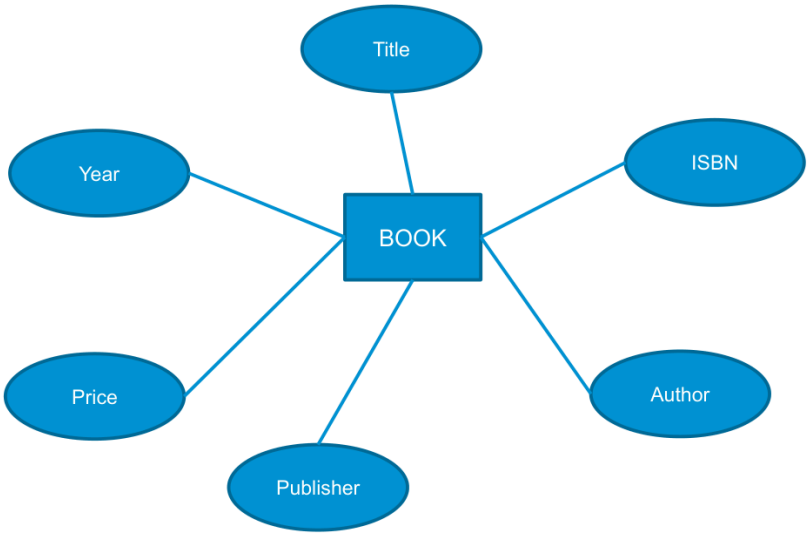


Source: Elmasri, Fundamentals of Database Systems, Page 204 ff

Key attributes in entity sets

Task 4

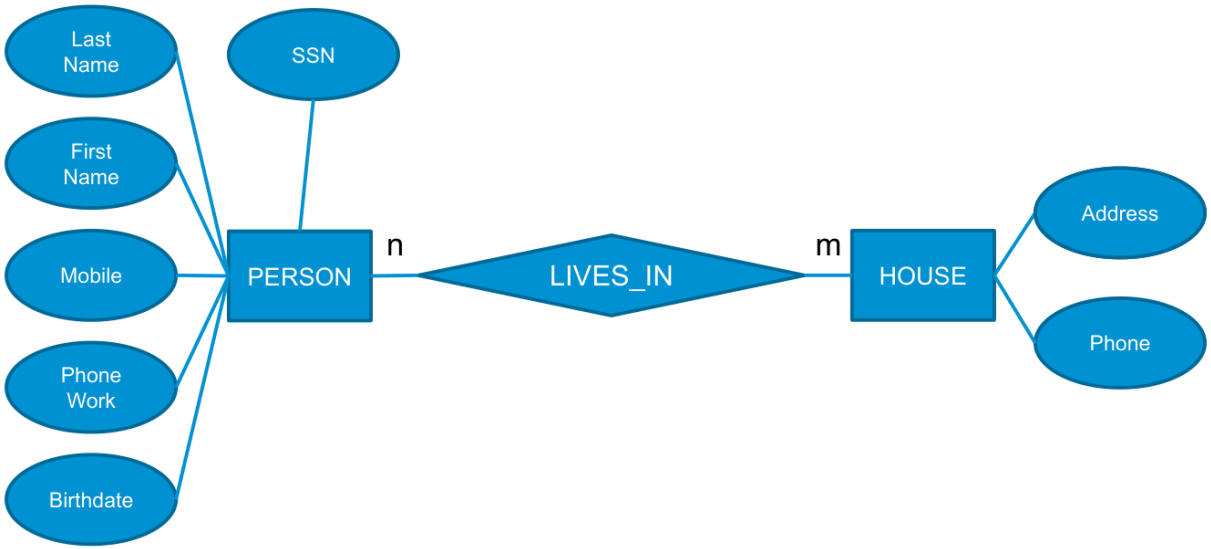
What are key attributes for BOOK?



Key attributes in entity sets

Task 5

What are key attributes for PERSON and HOUSE?

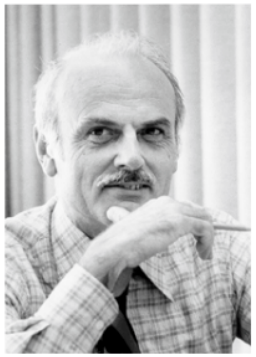


3. The relational model

3.1 What is the relational model

RM: A quick history

- Edgar F. Codd invented the relational model in 1970 and won the Turing price for it.
- The model has become widely accepted.
- The model is based on relations, that are subset of the Cartesian product.
- Everything is modelled in tables.



Source: www.wikipedia.org

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

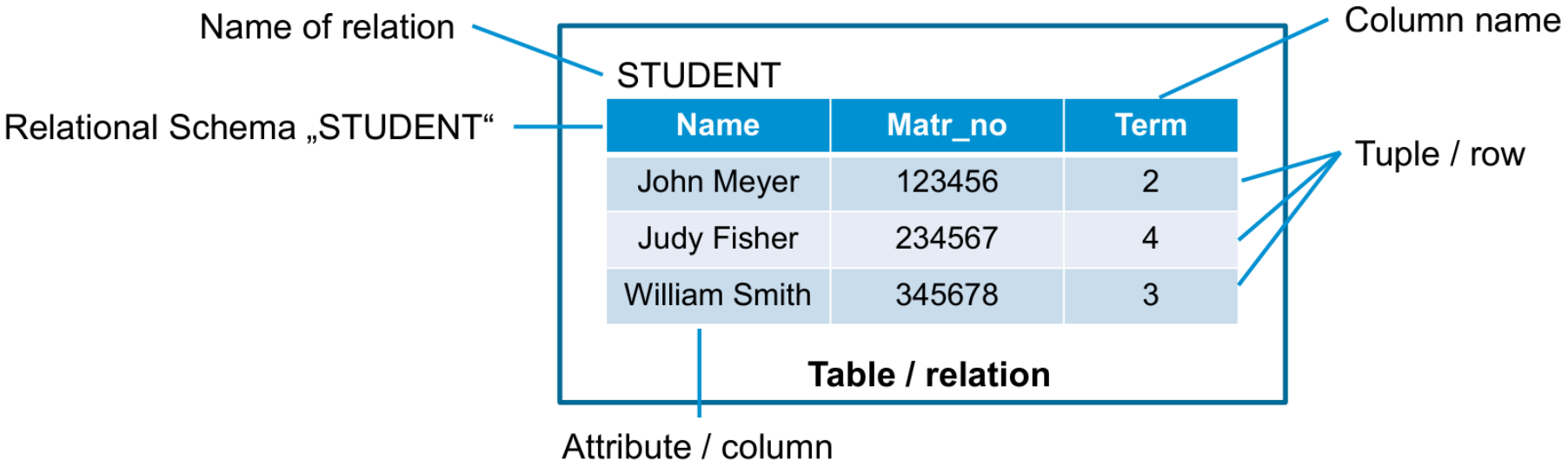
3.1 What is the relational model

RM: The model

- The relational schema 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

3.1 What is the relational model

RM: The model



Attribute	Type
Name	String
Matr_no	Integer
Term	Integer

3.1 What is the relational model

RM: The model

Name of relation

Relational Schema „STUDENT“

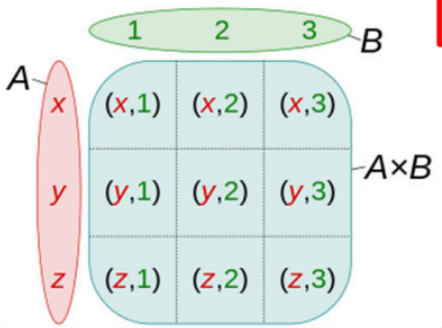
Column name

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

3.1 What is the relational model

RM: The 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

3.1 What is the relational model

RM: The 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

3.1 What is the relational model

- 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

3.1 What is the relational model

RM: The 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

3.1 What is the relational model

RM: The math behind it

- Example:
  - `ROOM(room_num, function, seats)`
  - where $\text{function} = \{\text{auditorium}, \text{lab}, \text{office}, \text{administration}\}$

3.1 What is the relational model

ROOM

RoomNr	Function	Seat
1465	Auditorium	50
1365	Lab	16
1002	Office	3
…	…	…

Table name / Relation

Row / Tuple / Record: distinct tuples

Column (col) / Attribute: attribute values

RM: The math behind it

Definition

- A Relation Schema R is a set of attributes $(A_1, A_2, ..., A_n)$.
- Each attribute A_i is the name of a role played by a certain domain D in the relational schema R.
- A domain D of attribute A_i is denoted as $\mathrm{dom}(A_i)$.
- The degree (or arity) of a relation is the number of attributes n of its relational schema.

3.1 What is the relational model

RM: The math behind it

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

3.1 What is the relational model

RM: The math behind it

- Relational Schema:

```
1  BOOK
2  (ISBN,
3  title,
4  author,
5  publisher,
6  year,
7  price)
```

- Relational Schema with types:

```
1  BOOK
2  (ISBN: integer,
3  title: string,
4  author: string,
5  publisher: string,
6  year: integer,
7  price: real)
```

→ Relation `BOOK` is of degree six.

3.1 What is the relational model

RM: The math behind it

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

$$r = (t_1, t_2, ..., t_m)$$

- Each n-tuple t is an ordered list of n values $t =< v_1, v_2, ..., v_n >$, where each value v_1, $1 \leq i \leq n$, is an element of $\mathrm{dom}(A_i)$ 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).

3.1 What is the relational model

RM: The math behind it

! Memorize

- A relation is a set of rows.
  - meaning: no order, no row number
  - no duplicates

3.1 What is the relational model

RM: The math behind it

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

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

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

$$|\text{dom}(A_1)| \times |\text{dom}(A_2)| \times ... \times |\text{dom}(A_n)|$$

3.1 What is the relational model

RM: The math behind it

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

3.1 What is the relational model

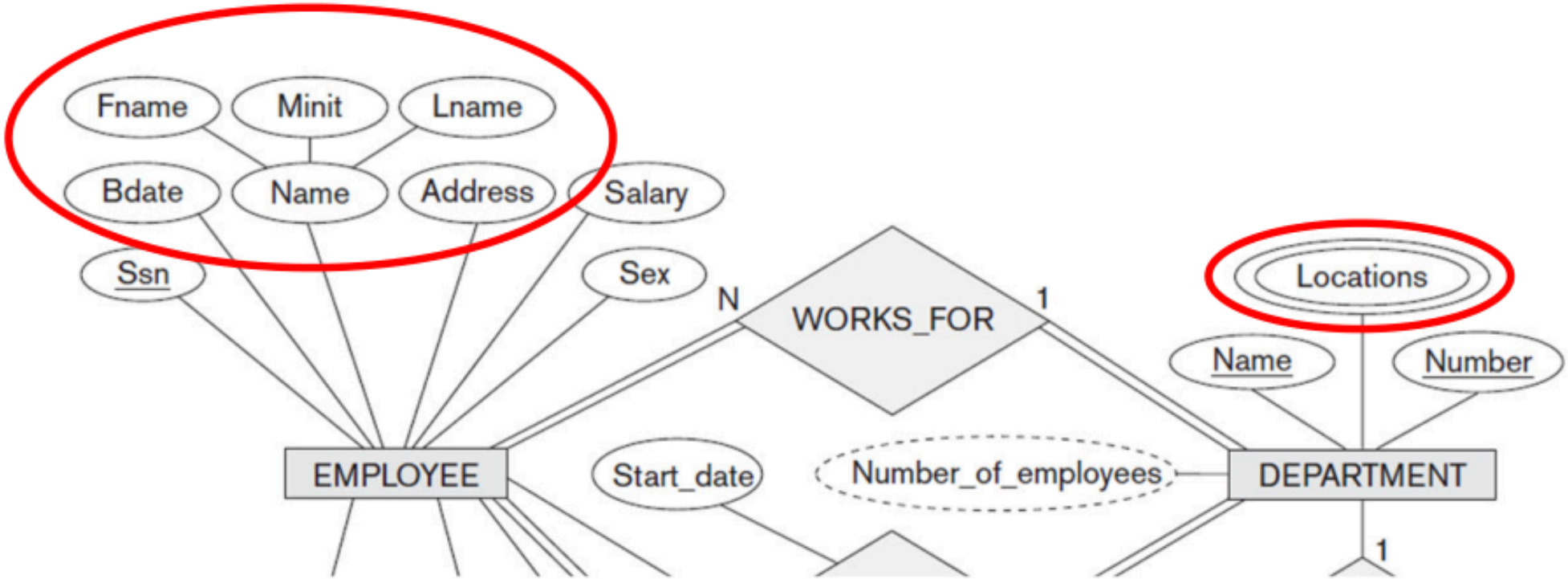
Relations and Values

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

  $\longrightarrow$ multivalued attributes must be represented by separate relations, and composite 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

3.1 What is the relational model

- Relations may represent entity types and relationship types from ERM.



3.1 What is the relational model

Notation

- A relational schema R of degree n is denoted by $R\,(A_1, A_2, ..., A_n)$
- 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 $t =< v_1, v_2, ..., v_n >$
- $t[A_i]$ and $t.A_i$ refer to the value v_i in t.
- $t[A_u, A_w, ..., A_z]$ and $t.(A_u, A_w, ..., A_z)$ refer to a subtuple in t

3.1 What is the relational model

Constraints

1. Constraints that are inherent in the data model
   - *inherent model-based constraints or implicit constraints*

Example

Example: no duplicate tuples in a relation

Constraints

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

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*

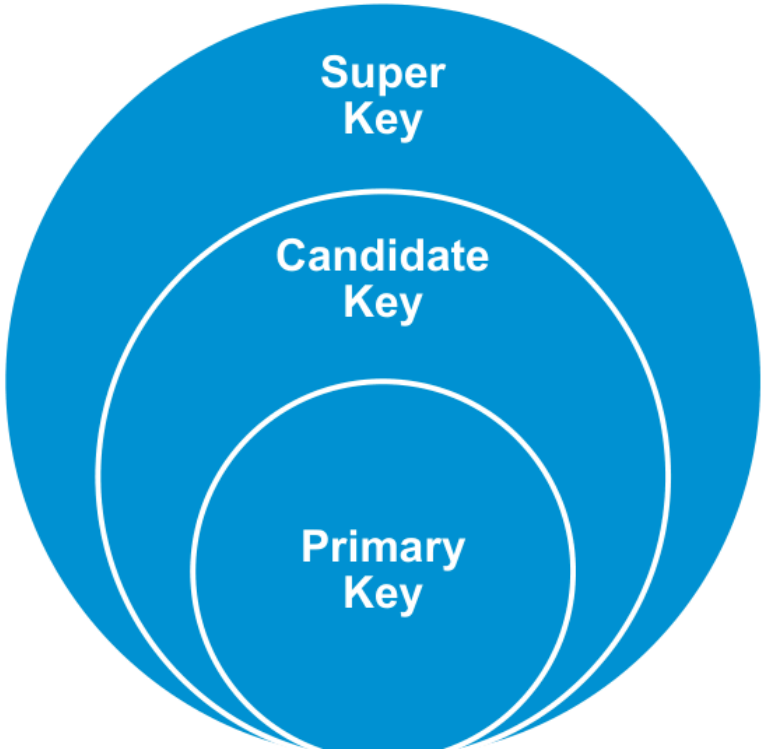
Constraints - Keys

- 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 $t_1[\mathrm{SK}] \neq t_2[\mathrm{SK}]$.
- Any such set of attributes SK is called a super key of a relation $\longrightarrow$ 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

3.1 What is the relational model

Constraints - 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”.



3.1 What is the relational model

Constraints - Key Attributes

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

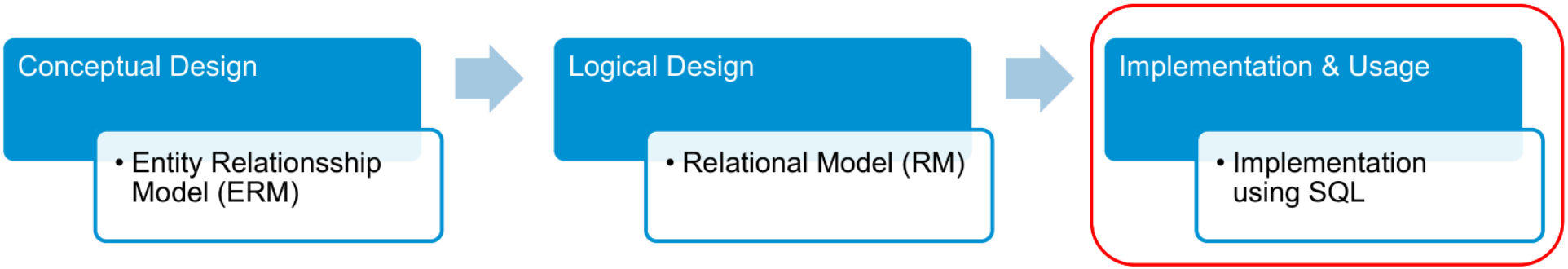
<u>ISBN</u>	Title	Author	Publisher	Year	Price
978-1-292-09761-9	Fundamentals of Database Systems	Ramez Elmasri	Prentice Hall	2016	59,99
978-0321197849	An Introduction to Database Systems	C. J. Date	Pearson	2003	69,92
…	…	…			

Constraints - 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

3.1 What is the relational model

Database Design



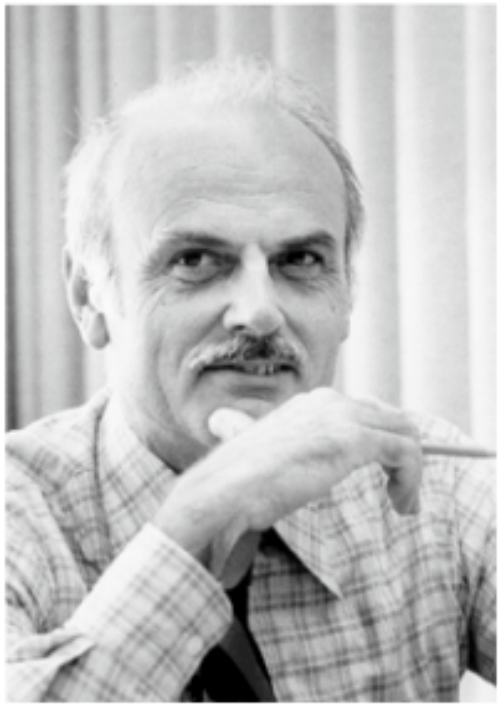
Physical Database Design

- 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.

3.1 What is the relational model

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



Source: www.wikipedia.org

3.1 What is the relational model

relational if it meets only some of the most important criteria

3.1 What is the relational model

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

SQL History

- SQL may commercial be considered one of the major reasons

for the commercial success relational databases

- `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

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

3.1 What is the relational model

- improved support for temporal databases, Roles, OLAP-Supporting
- requests: `ROLLUP`, `GROUPING SETS`, `CUBE`

3.1 What is the relational model

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++

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

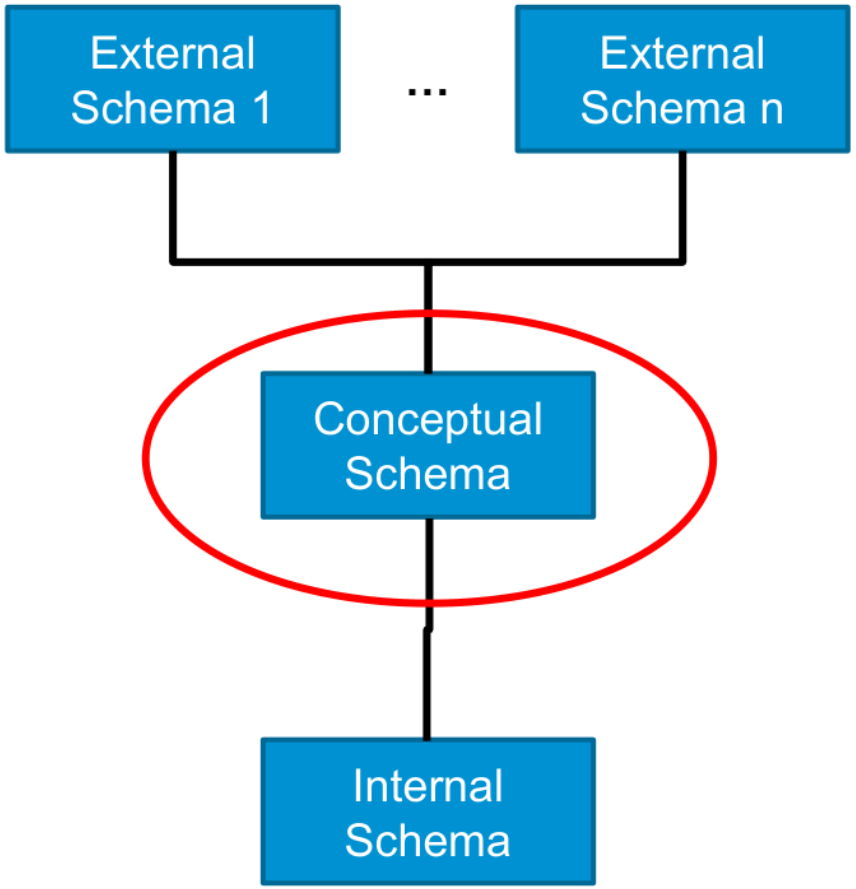
3.1 What is the relational model

SQL Basics - SQL commands

```sql
CREATE VIEW
DROP VIEW
```

```sql
CREATE database
DROP database
OPEN database
```

```sql
CREATE domain
DROP domain
OPEN domain
```



3.1 What is the relational model

SQL Basics

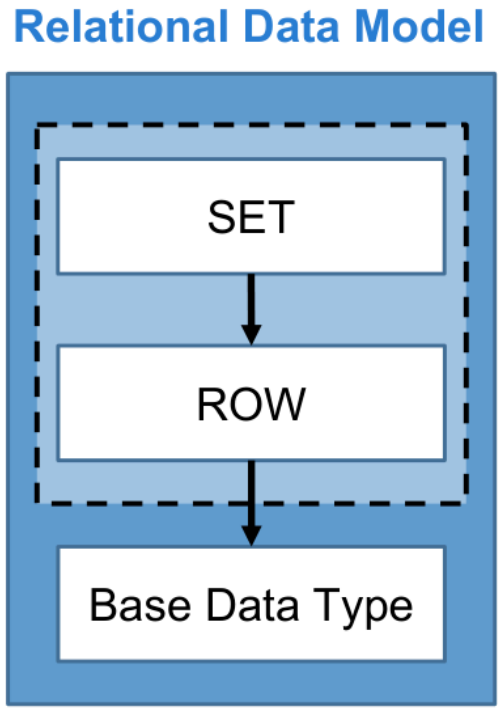
- SQL Keywords: case insensitive
- Convention: Upper Case (e.g., `SELECT`, `UPDATE`)
- Commands end with ;

(when entered interactively)

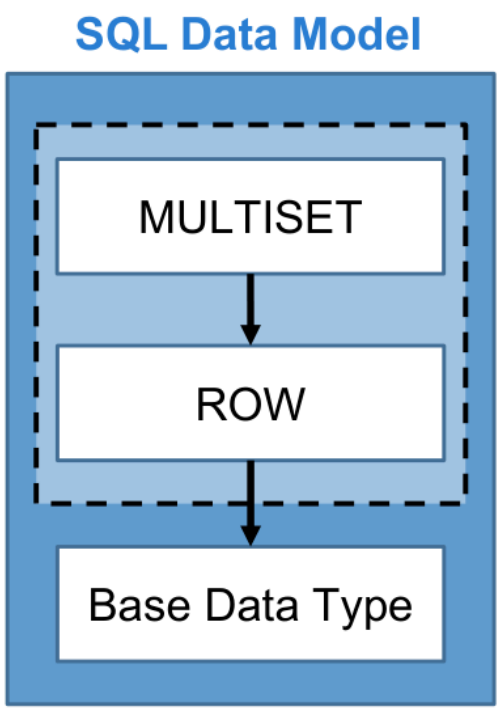
- Comments:
  - line comment: `--` this is a comment
  - multiline comment: `/* comment */`

3.1 What is the relational model

SQL Basics



SET: no order,
homogeneous elements,
no duplicates



MULTISET: no order,
homogeneous elements,
duplicates allowed

3.1 What is the relational model

SQL Basics

- 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)

3.1 What is the relational model

SQL Basics

- 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>

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

Create Schema

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

Example

```SQL
CREATE SCHEMA COMPANY;
```

Memorize

User must be authorized to create schema and schema elements!

3.1 What is the relational model

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

3.1 What is the relational model

Create Table

- Syntax for creating an empty table:

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

Mapping of ERM to Relational Model

Seven steps are involved in mapping the ERM to the RM:

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

Info

We will only look at step 1 here!

Mapping: Step 1

- 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

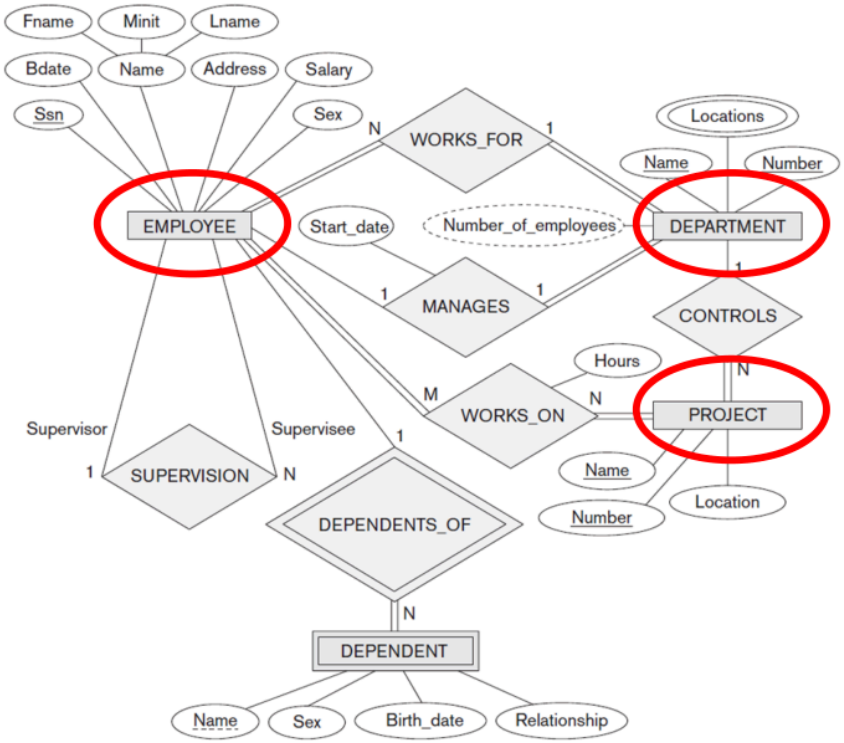
3.1 What is the relational model

Mapping: Step 1

MPLOYEE	Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary

EPARTMENT	Dname	Dnumber

ROJECT	Pname	Pnumber	Plocation



Source: Elmasri, Fundamental Database Systems, Page 286f

Mapping: Step 1

Example

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

3.1 What is the relational model

Create Table EMP

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

3.1 What is the relational model

Data Types: Overview

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

Data Types: Numeric

- 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` ⟶ (digits after the decimal point)

Example

- `INT`: `-2147483648` to `+2147483647`
- `REAL`: 6 decimal digits precision
- `NUMERIC(6,3)`: Up to `999999.999`

Data Types: Character

- 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`

3.1 What is the relational model

Data Types: Date and Time

- `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:

```sql
DATE '2008-09-27'
TIME '09:12:47'
TIMESTAMP [(precision)] [WITH TIME ZONE]
```

3.1 What is the relational model

Data Types: Additional

- `CLOB`: Character Large Object
  - Very long texts
  - in KB, MB, GB
- `BLOB`: Binary Large Object
  - Long Binary Data (e.g, pictures, video)
- `BOOLEAN`
- `JSON` or `JSONB`

3.1 What is the relational model

Data Types: Additional

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

3.1 What is the relational model

Primary Keys

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

Constraint Syntax

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

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

- As Table Constraint:

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

Column Constraint Example

Example Column Constraint:

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

Table Constraint Example

Example Table Constraint:

```sql
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 )
);
```

3.2 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

4. License Notice

4.1 Attribution

The basis of this course stems from: Professor Dr. Ulrike Herster
The following license applies to all unmodified and modified material used throughout this course.

4.1 Attribution

License

The publication or sharing of slides, image and sound recordings of this course are not permitted.