732A54
Big Data Analytics

Topic: Relational Databases

Huanyu Li
huanyu.li@liu.se

LINKÖPING UNIVERSITY

Literature

- Elmasri, Navathe, Fundamentals of Database Systems, 7th edition, Addison Wesley, 2016. Chapters 3-6 and 9; section 7.1.

Slide

- This slide is based on Patrick’s previous slides, and Olaf’s 732A57 slides.
- https://www.ida.liu.se/~732A57/fo/index.en.shtml

Outline

- Basic Terminology
- The Database Approach
- The Lecture's Focuses
  - Representing and Storing data
  - EER to Database Schema
  - Querying RDBs using SQL
- Lab RDB

Basic Terminology

LINKÖPING UNIVERSITY

Most Basic Terminology

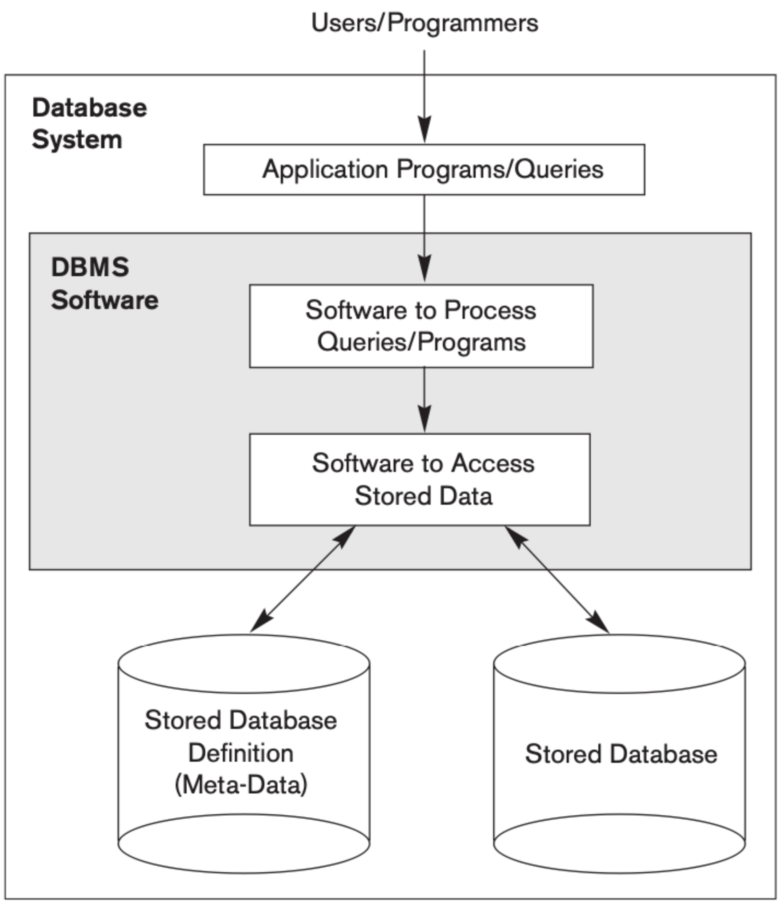
- **Data**: known facts that can be records and that have implicit meaning
- **Database**:
  - collection of related data
  - Represents some aspect of the real world
  - Built for a specific purpose
- Examples of databases (used in every-day life):
  - Product data of e-commerce platforms (e.g., Amazon)
  - Transaction data of a bank account

Terminology (cont'd)

- **Database management system (DBMS)**:
  - A collection of computer programs or cloud services
  - Enables users to create and maintain a database (DB)
  - Supports concurrent access to a database by multiple users and programs
  - Interfaces: query languages, graphical tools, APIs, etc.
  - …
- Examples of DBMS:
  - Microsoft's SQL Server
  - MySQL
  - Oracle
- **Database System = DB + DBMS**

Terminology (cont'd)

- **Database System = DB + DBMS**



Elmasri, R. & Navathe, S.B. (2016). *Fundamentals of Database Systems* (7th ed.). Pearson. Chapter 1.

Terminology (cont'd)

- **Persons:**
  - Database Administrators
    - Manage resources
  - Database Designers
    - Design structures
  - End Users
    - Manipulate data (create, query. update)

  - System Analysts and Application Programmers
  - DBMS designers and implementers
  - Tool developers
  - Operators and maintenance personnel

Database Approach

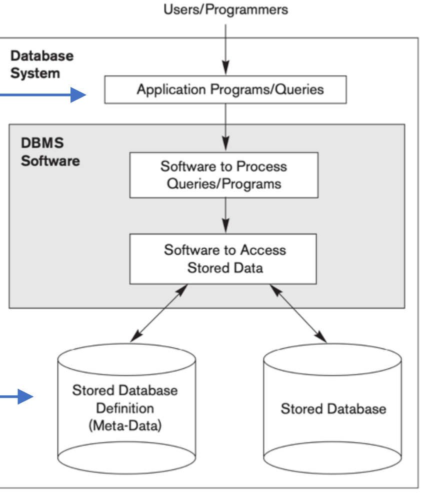
LINKÖPING UNIVERSITY

Database Approach

- Use a single repository to maintain data that is defined and accessed by various users
- To enable a database approach, we need to design a database system
  - **Database design** focuses on defining the database
  - **Application design** focuses on the programs and interfaces that access the database (out of scope of this lecture)

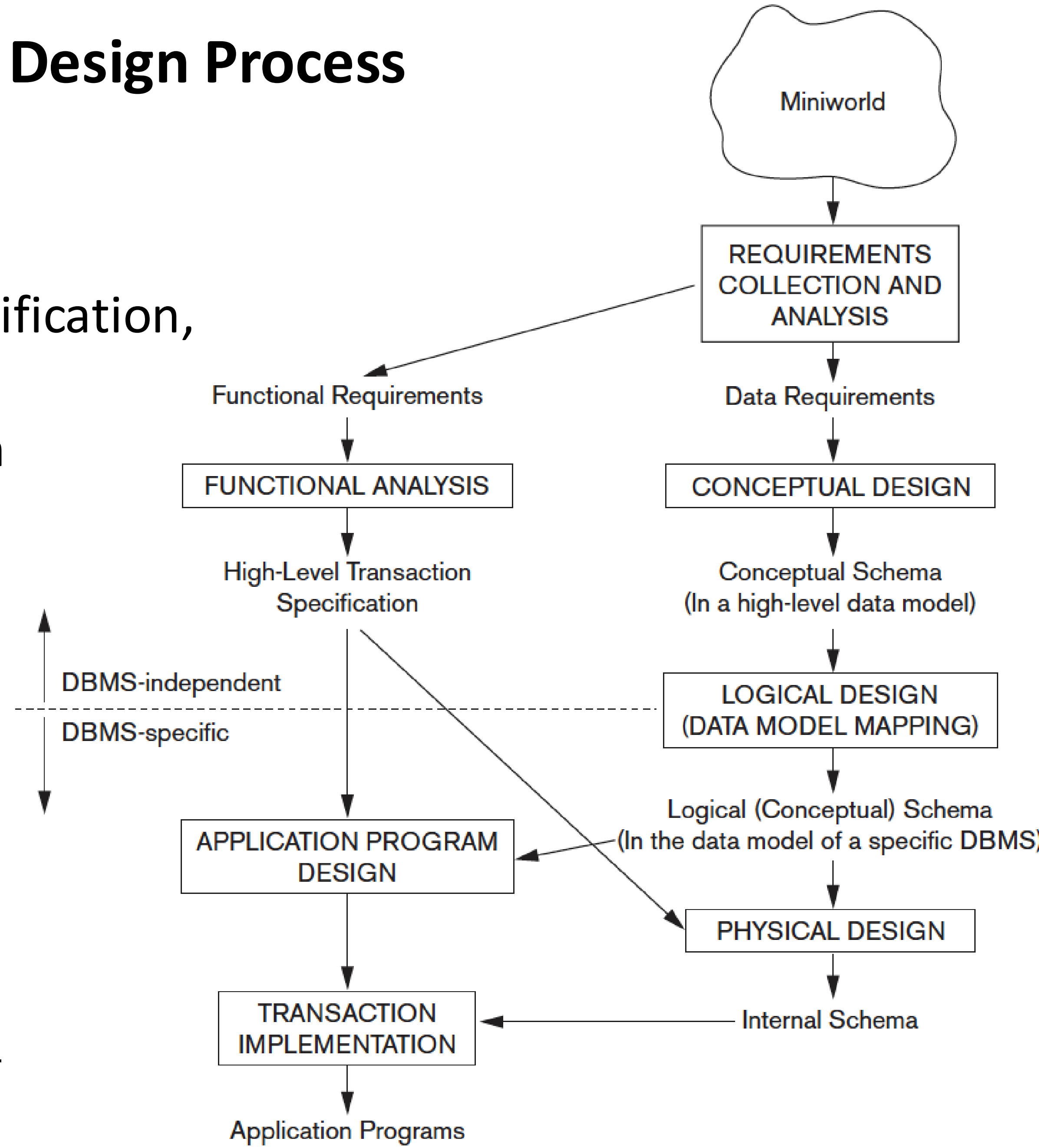
application design

database design



Database System Design Process

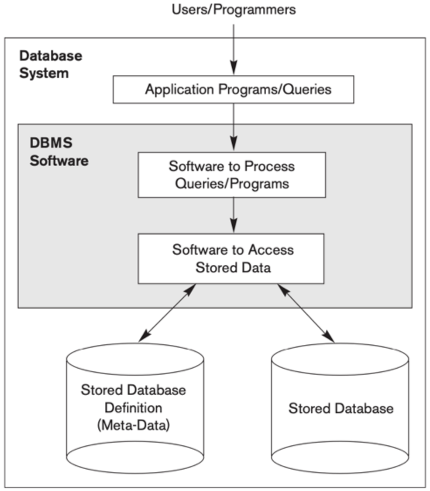
- Database design
  - Requirement specification, analysis
  - Conceptual design
  - Logical design
  - Physical design

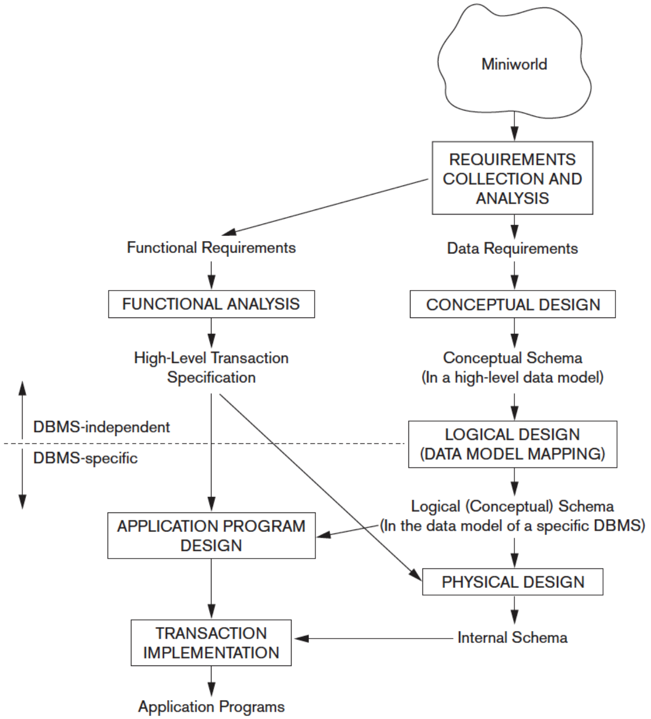


The Lecture’s Focuses

This lecture's focuses

- What information is stored?
- How is the information stored?
  - (high level and low level)
- How is the information accessed?
  - (user level and system level)





Other issues out of scope of this lecture

- How to optimize performance of a data source?
- How to recover of a data source after crash?
- How to access information from multiple data sources?
- How to allow and control multiple users to access a data source?

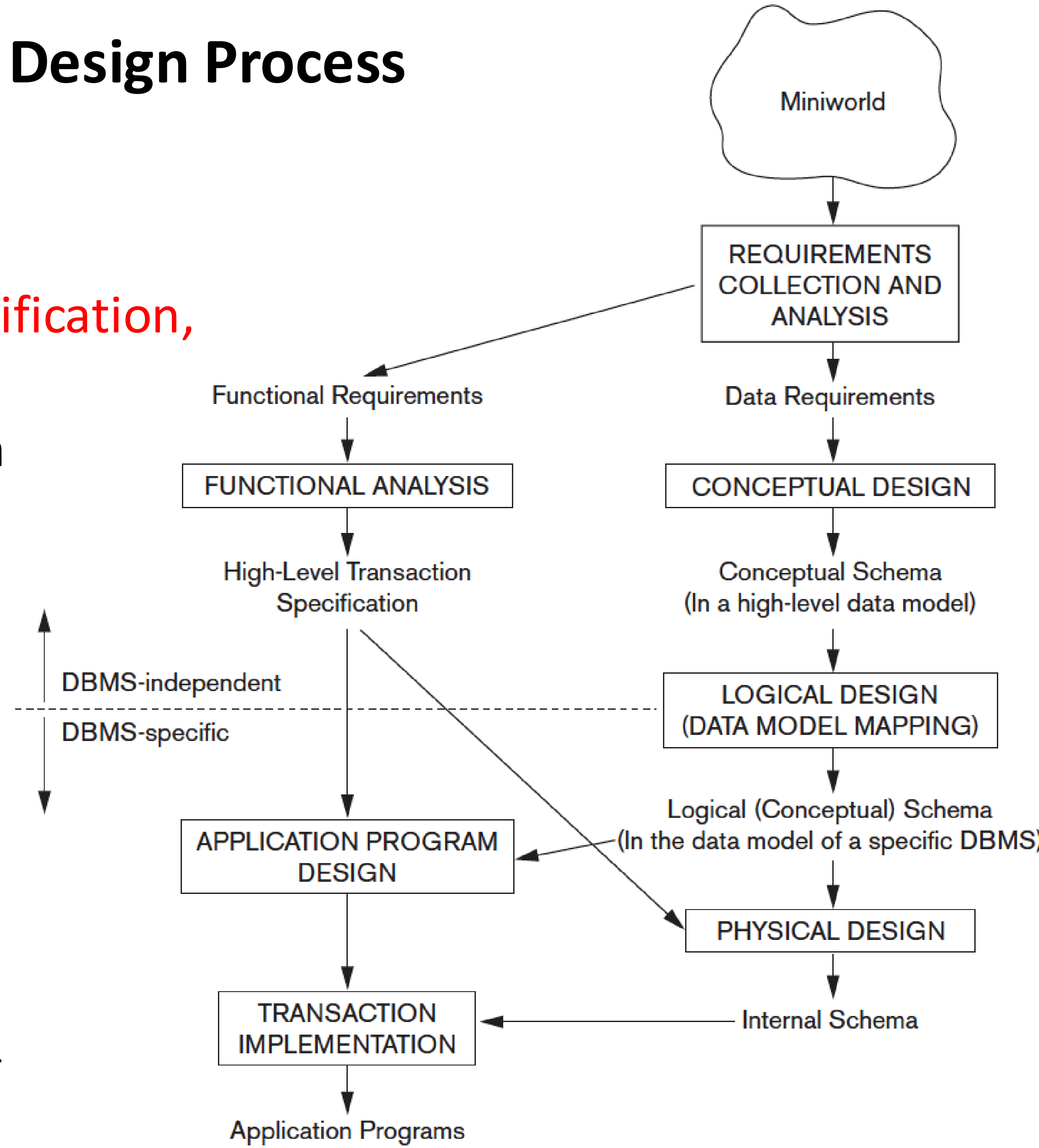
Representing and Storing data

-- What information is stored?
-- How is the information stored?

LINKÖPING
UNIVERSITY

Database System Design Process

- Database design
  - Requirement specification, analysis
  - Conceptual design
  - Logical design
  - Physical design



Mini-world Example

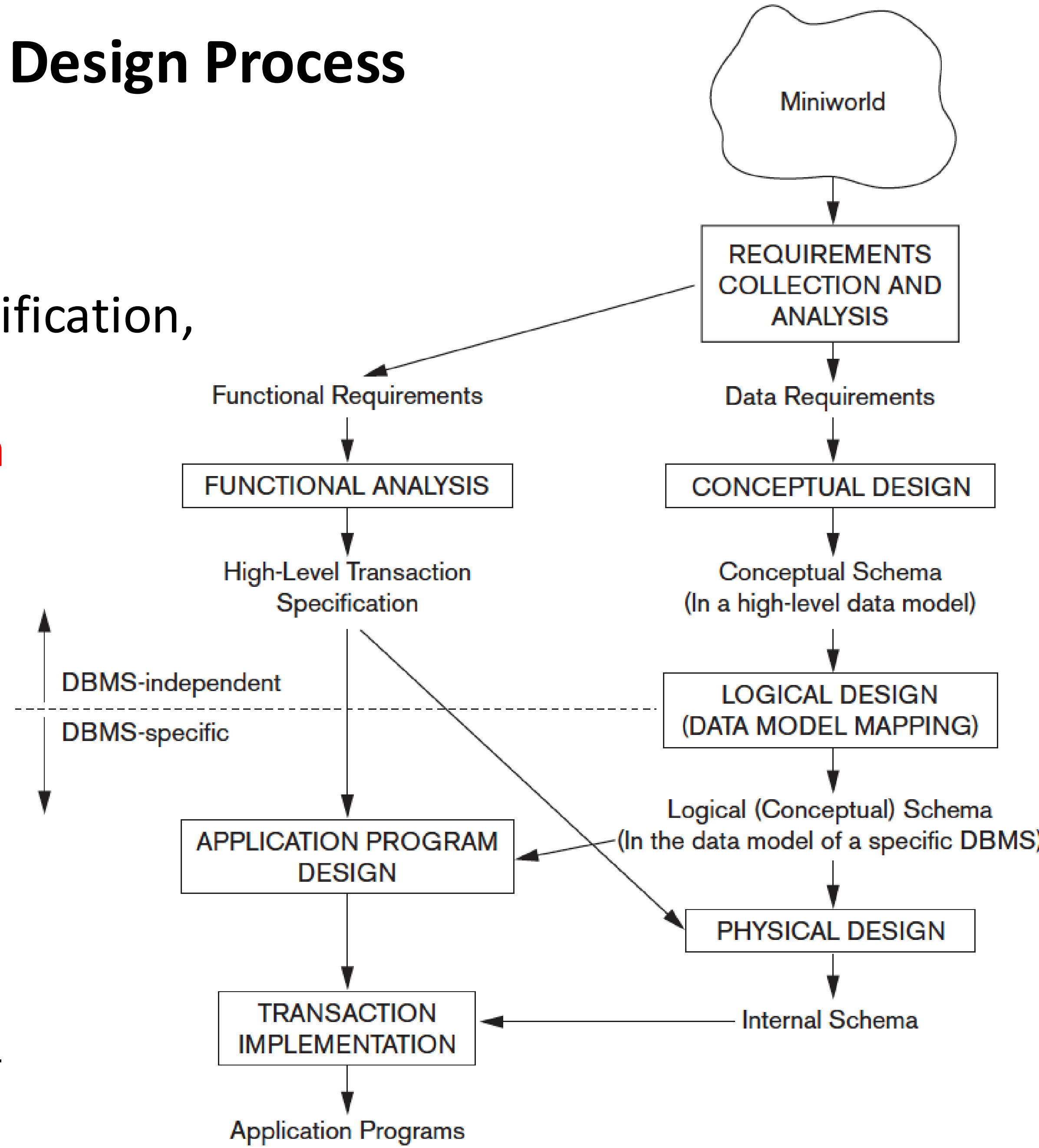
- A taxi company needs to model their activities.
- There are two types of employees in the company: drivers and operators. For drivers it is interesting to know the date of issue and type of the driving license, and the date of issue of the taxi driver's certificate. For all employees it is interesting to know their personal number, address and the available phone numbers.
- The company owns a number of cars. For each car there is a need to know its type, year of manufacturing, number of places in the car and date of the last service.
- The company wants to have a record of car trips. A taxi may be picked on a street or ordered through an operator who assigns the order to a certain driver and a car. Departure and destination addresses together with times should also be recorded.

Requirement analysis, what information is stored?

- A taxi company needs to model their activities.
- There are two types of employees in the company: drivers and operators. For drivers it is interesting to know the date of issue and type of the driving license, and the date of issue of the taxi driver's certificate. For all employees it is interesting to know their personal number, address and the available phone numbers.
- The company owns a number of cars. For each car there is a need to know its type, year of manufacturing, number of places in the car and date of the last service.
- The company wants to have a record of car trips. A taxi may be picked on a street or ordered through an operator who assigns the order to a certain driver and a car. Departure and destination addresses together with times should also be recorded.

Database System Design Process

- Database design
  - Requirement specification, analysis
  - Conceptual design
  - Logical design
  - Physical design

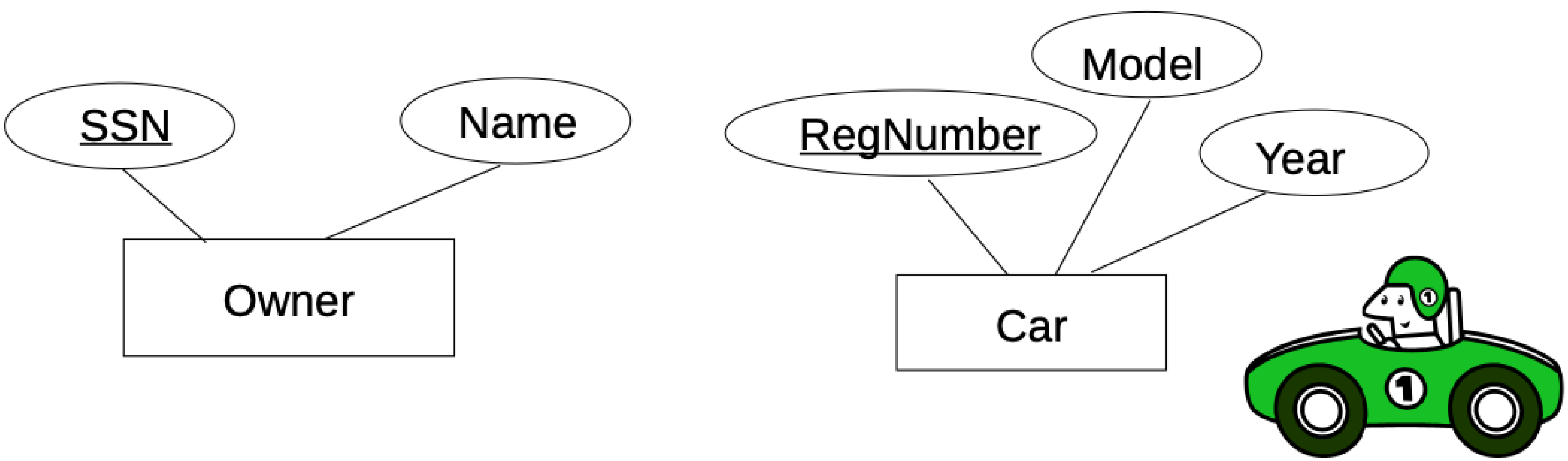


ER/EER Model ((Enhanced) Entity Relationship)

- High-level conceptual data model
  - An overview of the database
  - Independent of type of data source
  - Easy to discuss with non-database experts
  - Easy to translate to data model of DBMS
- ER diagram (diagrammatic notion):
  - entities and entity type
  - attributes
  - key attributes
  - relationships and cardinality constraints
  - sub-types (EER)

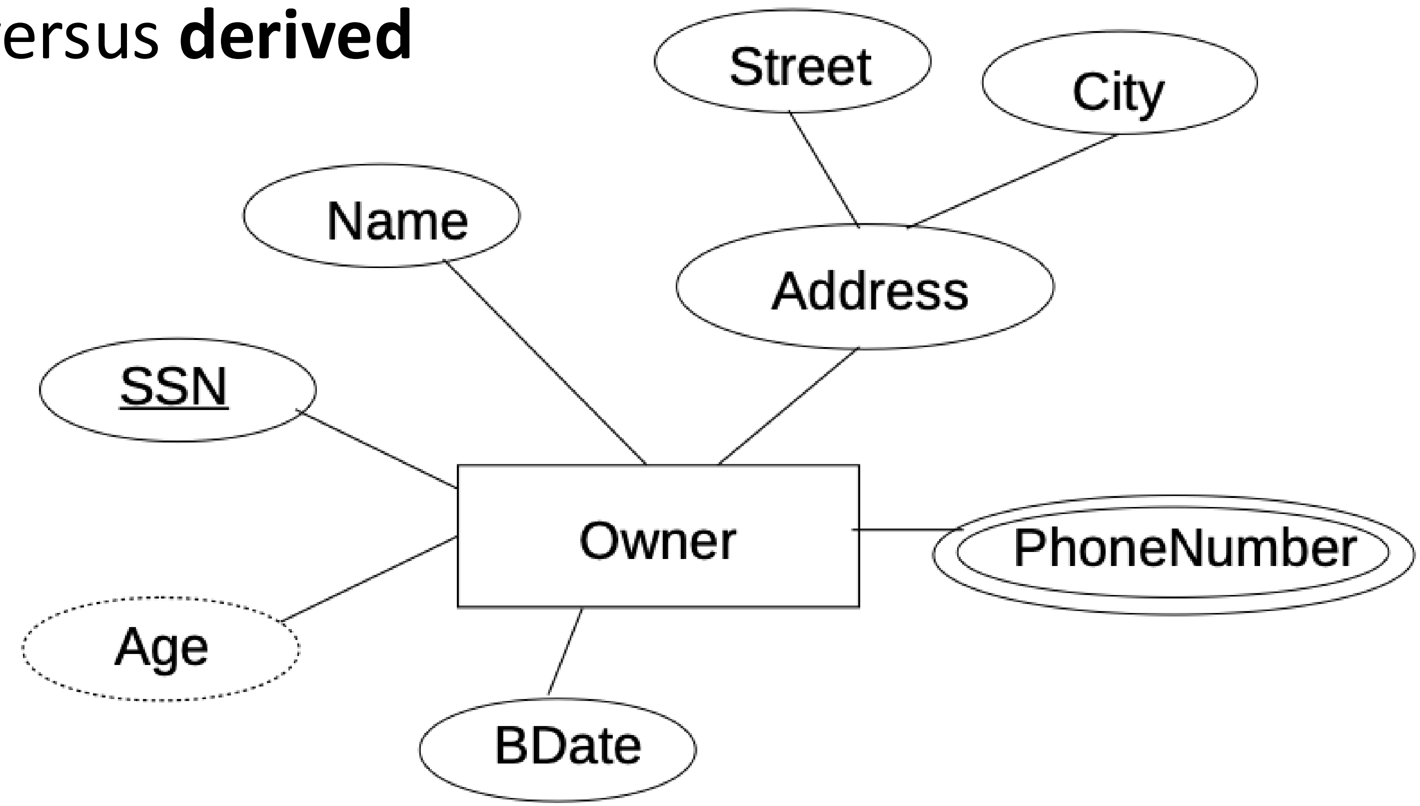
Entities and Entity Types

- **Entity**: a ‘thing’ in the real world with an independent existence
- **Attributes**: Properties that describe an entity
- **Entity type**: A collection of entities that have the same set of attributes



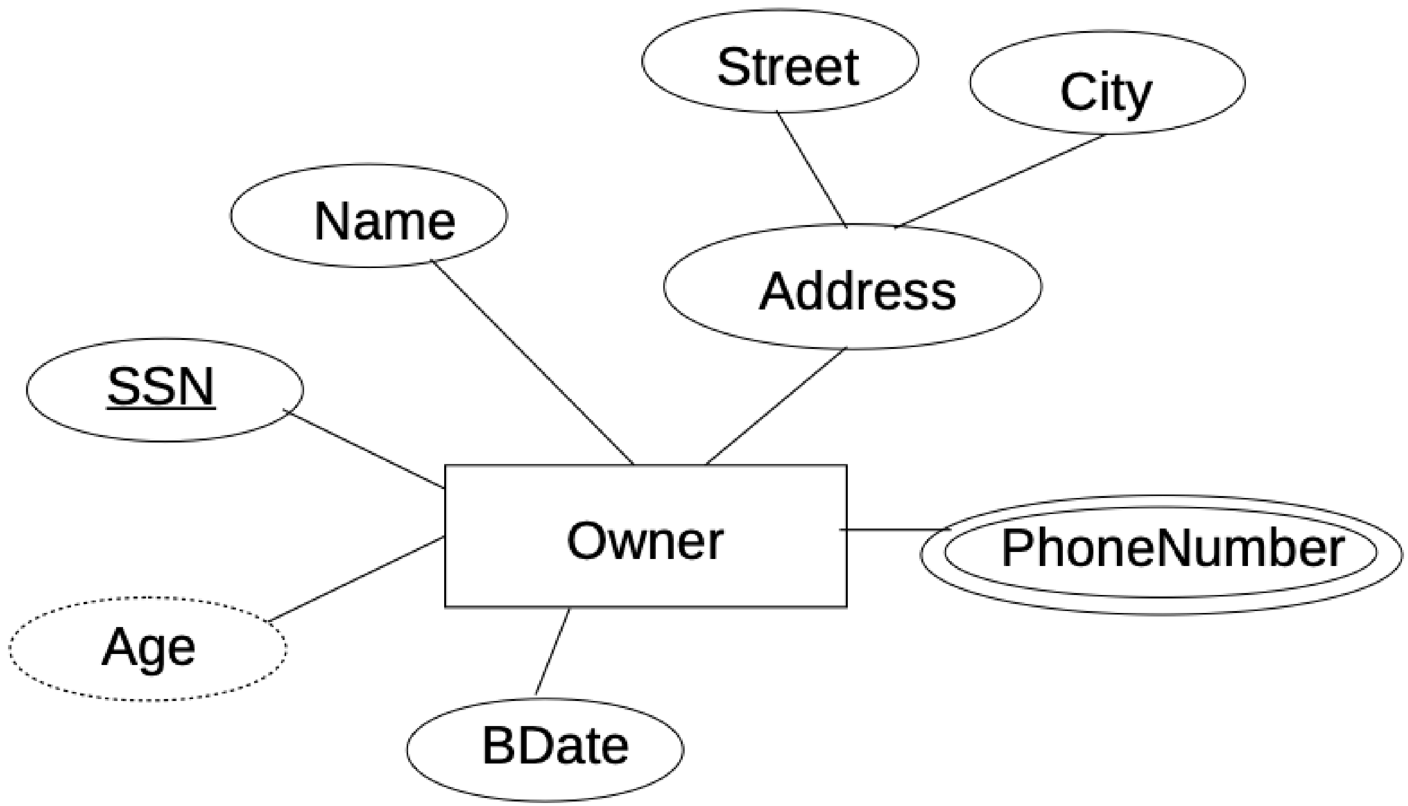
Attributes

- **simple** versus **composite**
- **single-valued** versus **multivalued**
- **stored** versus **derived**



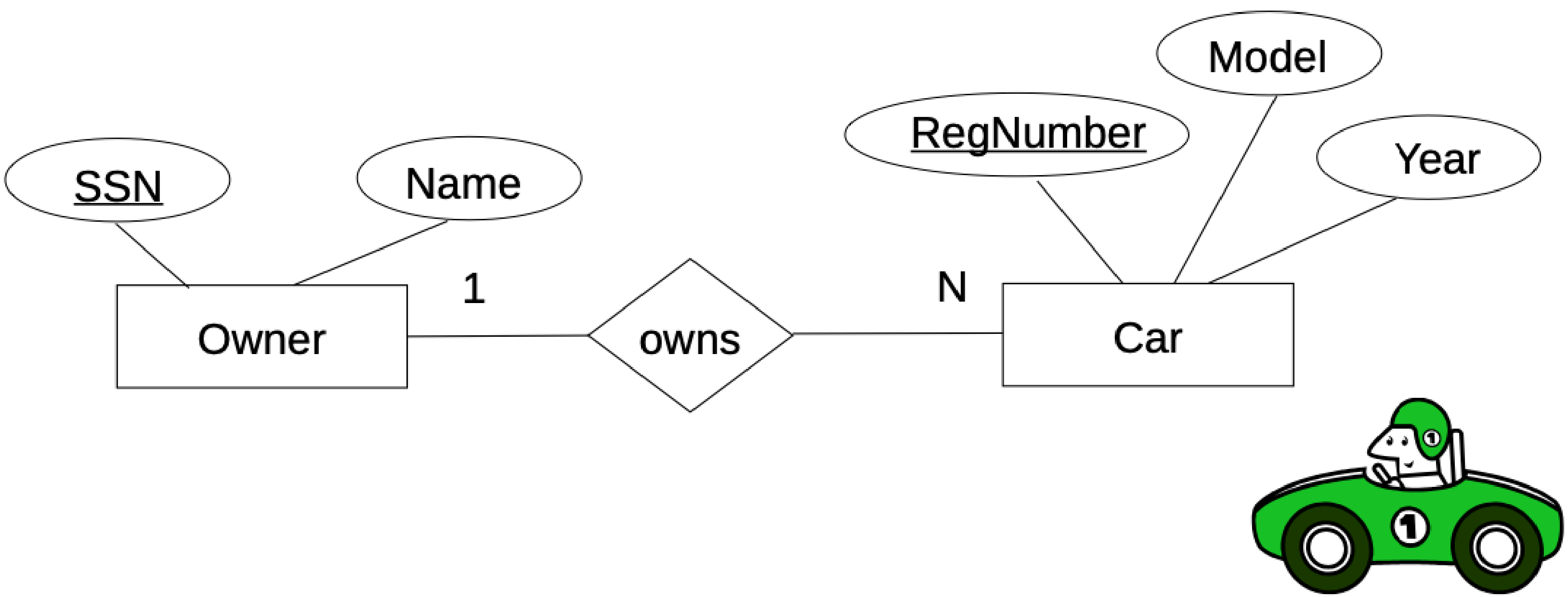
Constraints on Attributes

- Value sets (domains) of attributes
- Key attributes



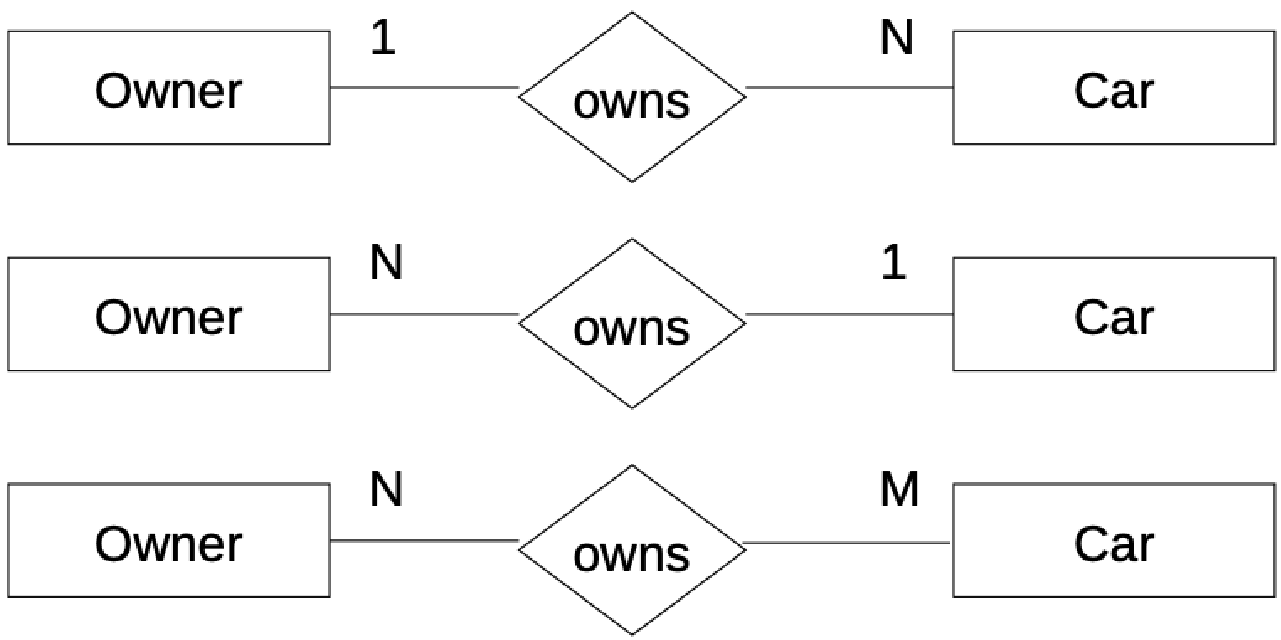
Relationship Types

- Relationship type: Association among entity types



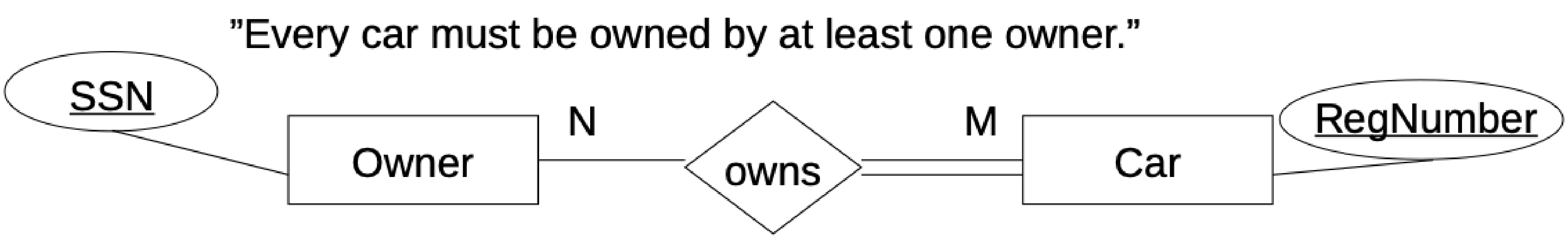
Constraints on Relationship Types

- Cardinality ratio: Maximum number of relationships an entity can participate in
- Possible cardinality ratio: *1:1*, *1:N*, *N:1*, and *N:M*



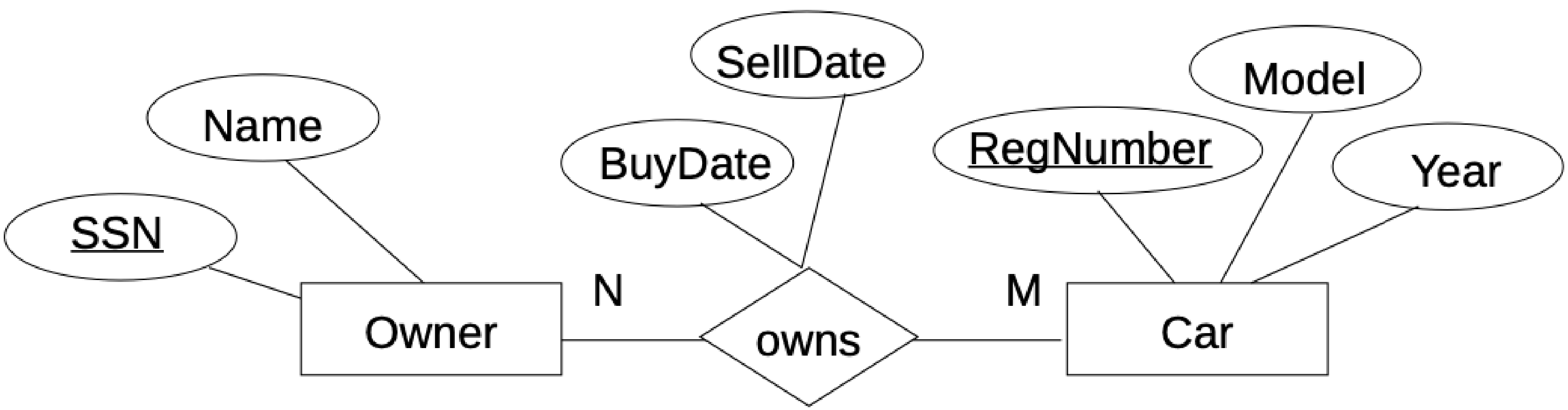
Constraints on Relationship Types

- Participation constraint
  - **Total participation**: Every entity participates in at least one relationship with another entity
  - **Partial participation**: Not all entities are involved in the relationship



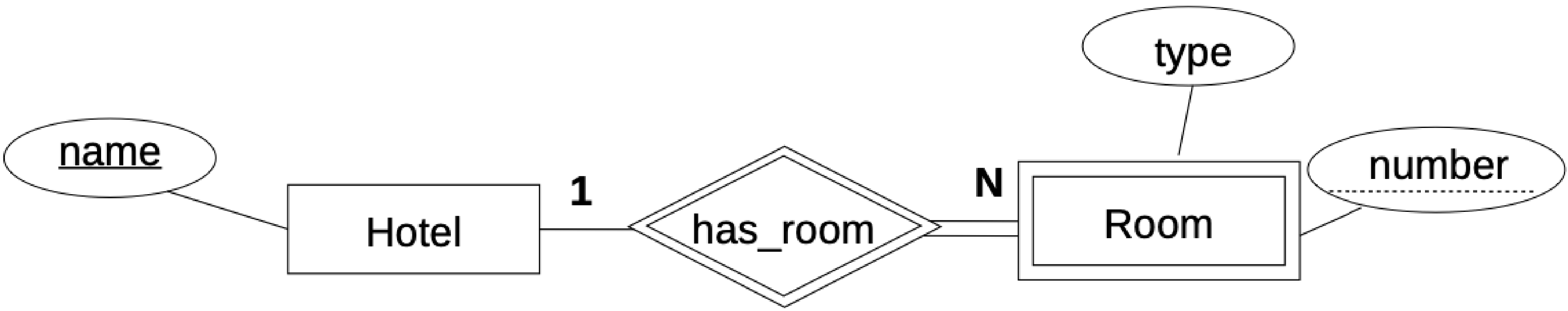
Attributes of Relationship Types

"Store information on who owned which car and during which period of time"



Weak Entity Types

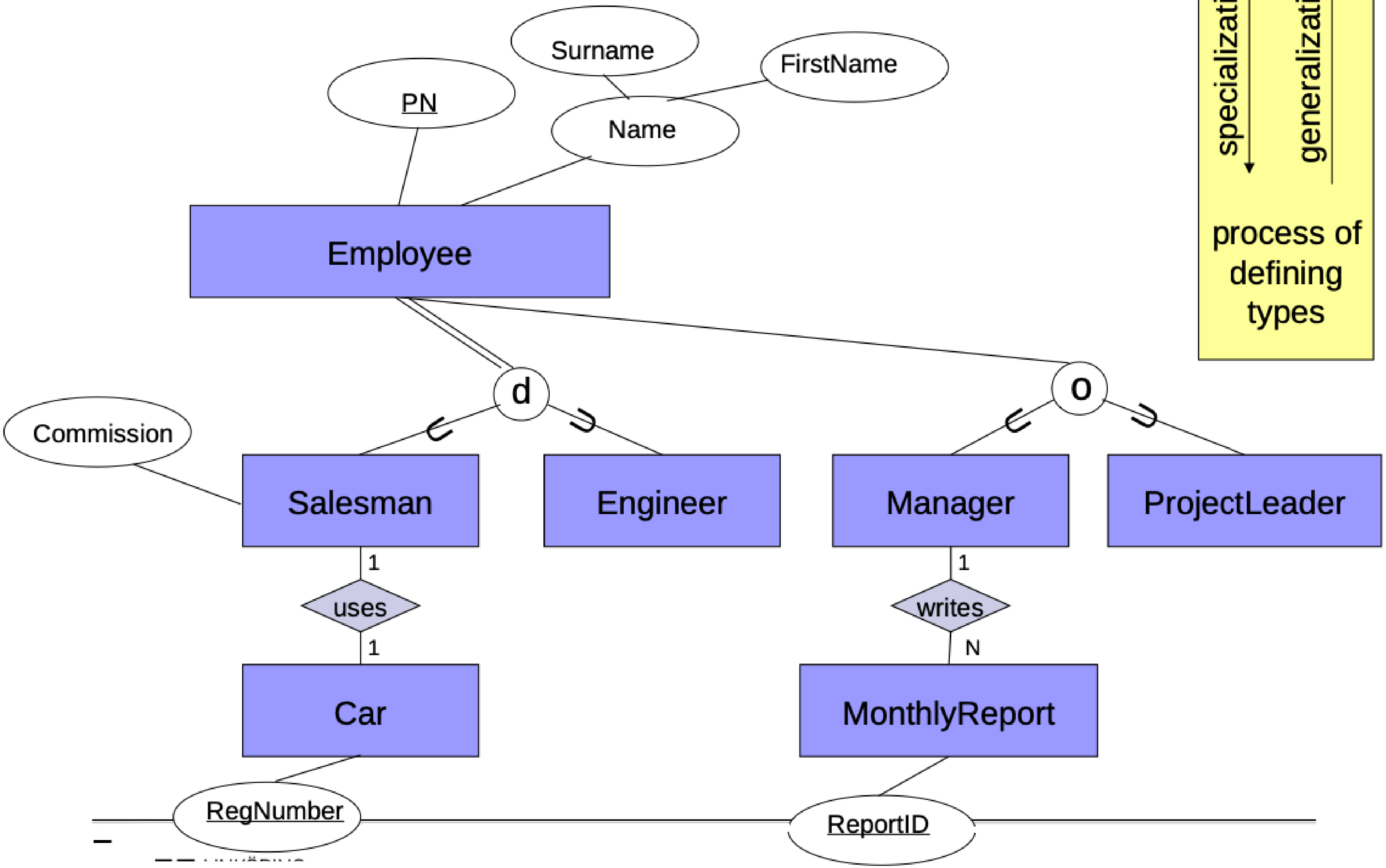
- Identified by their relationship to a specific entity from another entity type
- Do not have key attributes of their own
  - Only partial key
  - The identifying entity has the rest of the key



Enhanced ER (EER) Model

- Why more? (Enhanced)
  - To support more complex data requirements
  - Example: Only some employees can use a company car, only managers have to write monthly report, but all employees have assigned personal number, salary account and a place in the office
- What more?
  - Specialization / generalization (Subtype / supertype)
  - Union subtypes
  - Attributes and relationship inheritance

Subtype / Supertype



specialization
generalization

process of defining types

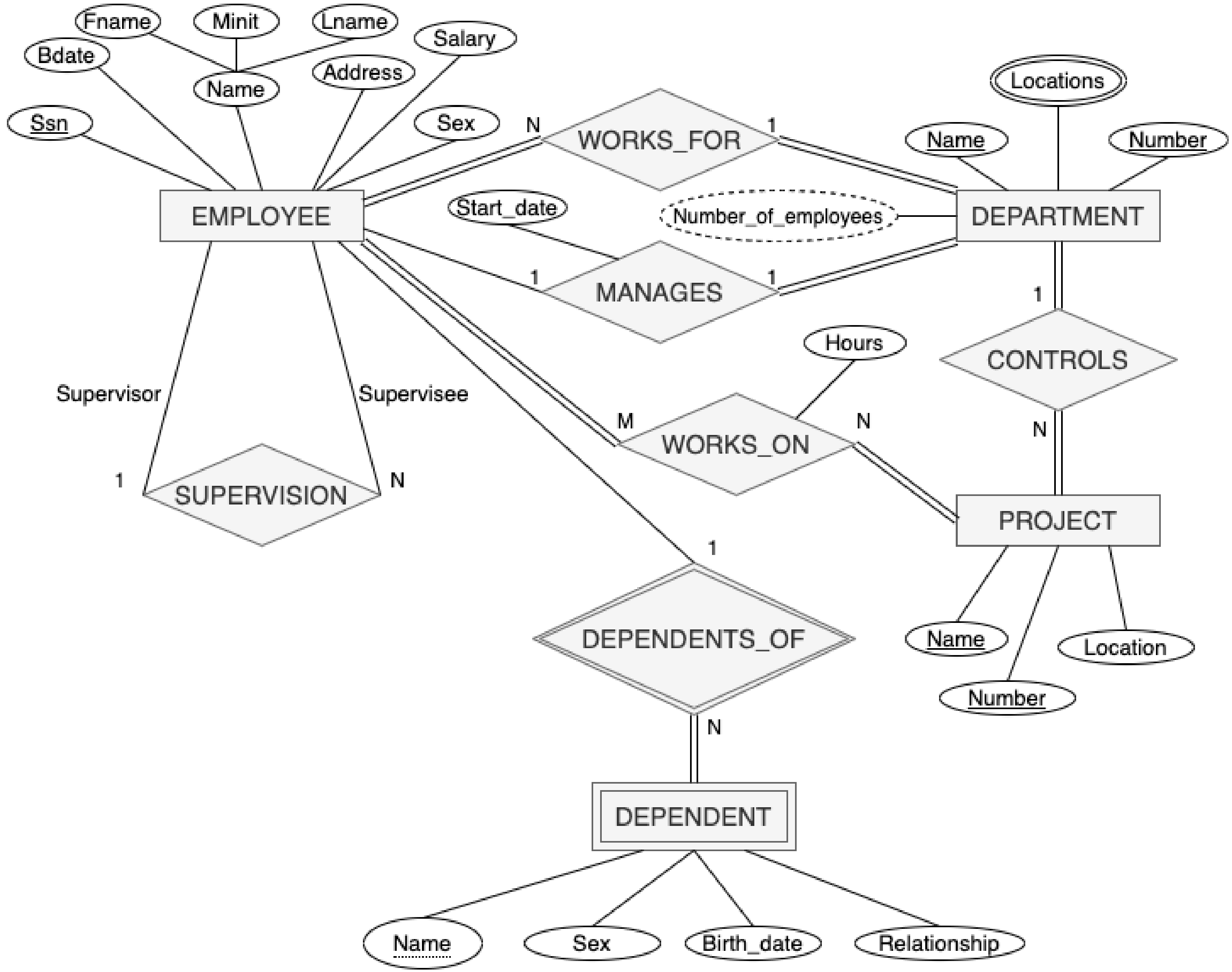
Exercise: a Miniworld

- The company is organized into departments. Each department has a unique name, a unique number, and a particular employee who manages the department. We keep track of the start date when that employee began managing the department. A department may have several locations.
- A department controls a number of projects, each of which has a unique name, a unique number, and a single location.
- The database will store each employee’s name, social security number, address, salary, sex (gender), and birth date. An employee is assigned to one department, but may work on several projects, which are not necessarily controlled by the same department. It is required to keep track of the current number of hours per week that an employee works on each project, as well as the direct supervisor of each employee (who is another employee).
- The database will keep track of the dependents of each employee for insurance purposes, including each dependent’s first name, sex, birth date, and relationship to the employee.

Exercise: a Miniworld

- The company is organized into departments. Each department has a unique name, a unique number, and a particular employee who manages the department. We keep track of the start date when that employee began managing the department. A department may have several locations.
- A department controls a number of projects, each of which has a unique name, a unique number, and a single location.
- The database will store each employee's name, Social Security number, address, salary, sex (gender), and birth date. An employee is assigned to one department, but may work on several projects, which are not necessarily controlled by the same department. It is required to keep track of the current number of hours per week that an employee works on each project, as well as the direct supervisor of each employee (who is another employee).
- The database will keep track of the dependents of each employee for insurance purposes, including each dependent's first name, sex, birth date, and relationship to the employee.

EER


Fname
Minit
Lname
Bdate
Name
Address
Salary
Ssn
Sex
EMPLOYEE
N
WORKS_FOR
1
Locations
Name
Number
DEPARTMENT
Start_date
Number_of_employees
1
MANAGES
1
1
CONTROLS
Hours
Supervisor
Supervisee
M
WORKS_ON
N
N
1
SUPERVISION
N
PROJECT
1
DEPENDENTS_OF
Name
Location
Number
N
DEPENDENT
Name
Sex
Birth_date
Relationship

Storing data with databases

- Relational Databases
  - Model: tables + relation algebra
  - Query language (SQL)
- Object-oriented, NoSQL databases

Storing data with relational databases

- Relational Database
  - Represent data as a collection of relations
  - Think of a relation as a table of values
  - Each row (tuple) represents a record of related data values
    - Unordered
    - No duplicates
  - Each column (attribute) holds a corresponding value for each row
    - Simple atomic values or NULL

Personnel number	Name	Personnel type
1	Karin	Driver
2	Lars	Driver
3	Bo	NULL
4	Lotta	Engineer

Keys

- Consist of one or several columns
- Unique value (for a specific entity type)
- Different kinds of keys
  - Candidate key
  - Primary key
  - Alternate key
  - Foreign key

Personnel number	Personal code number	Name	Personnel type
1	750520-2389	Karin	Driver
2	680913-3871	Lars	Driver
3	500304-7351	Bo	NULL
4	611125-6425	Lotta	Engineer

ID	Name	Role number
1	Karin	1
2	Lars	1
3	Bo	2
4	Lotta	3

Keys

- Consist of one or several columns
- Unique value (for a specific entity type)
- Different kinds of keys
  - Candidate key
  - Primary key
  - Alternate key
  - Foreign key

Personnel number	Personal code number	Name	Personnel type
1	750520-2389	Karin	Driver
2	680913-3871	Lars	Driver
3	500304-7351	Bo	NULL
4	611125-6425	Lotta	Engineer

ID	Name	Role number
1	Karin	1
2	Lars	1
3	Bo	2
4	Lotta	3

Primary key

- Must uniquely identify each record in the table
- No part of the primary-key value can be null
- Should preferably be minimal
- … a new field can be introduced serve as the primary key if necessary!

Example

We know that each row is unique...

Personnel number	Personal code number	Name	Personnel type
1	750520-2389	Karin	Driver
2	680913-3871	Lars	Driver
3	500304-7351	Bo	NULL
4	611125-6425	Lotta	Engineer

Example

What about using all attributes as primary key?

Personnel number	Personal code number	Name	Personnel type
1	750520-2389	Karin	Driver
2	680913-3871	Lars	Driver
3	500304-7351	Bo	NULL
4	611125-6425	Lotta	Engineer

Example

Invalid since personnel type can be null!

Personnel number	Personal code number	Name	Personnel type
1	750520-2389	Karin	Driver
2	680913-3871	Lars	Driver
3	500304-7351	Bo	NULL
4	611125-6425	Lotta	Engineer

Example

Use all attributes except personell type?

Personnel number	Personal code number	Name	Personnel type
1	750520-2389	Karin	Driver
2	680913-3871	Lars	Driver
3	500304-7351	Bo	NULL
4	611125-6425	Lotta	Engineer

Example

Valid but maybe we can use fewer fields for the key?

Personnel number	Personal code number	Name	Personnel type
1	750520-2389	Karin	Driver
2	680913-3871	Lars	Driver
3	500304-7351	Bo	NULL
4	611125-6425	Lotta	Engineer

Example

Personnel number	Personal code number	Name	Personnel type
1	750520-2389	Karin	Driver
2	680913-3871	Lars	Driver
3	500304-7351	Bo	NULL
4	611125-6425	Lotta	Engineer

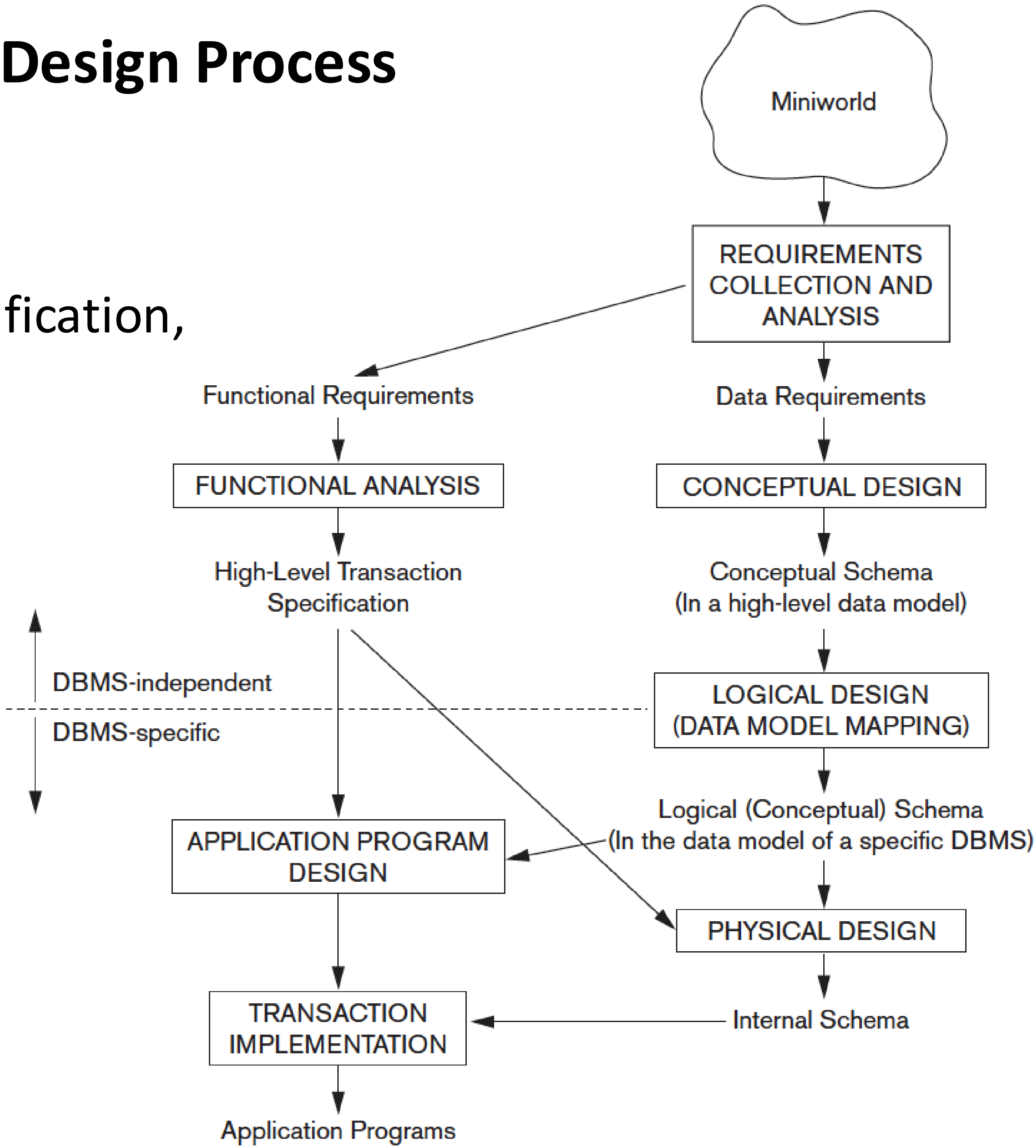
Two alternatives:

Personnel number	Personal code number	Name	Personnel type
1	750520-2389	Karin	Driver
2	680913-3871	Lars	Driver
3	500304-7351	Bo	NULL
4	611125-6425	Lotta	Engineer

EER to Database Schema

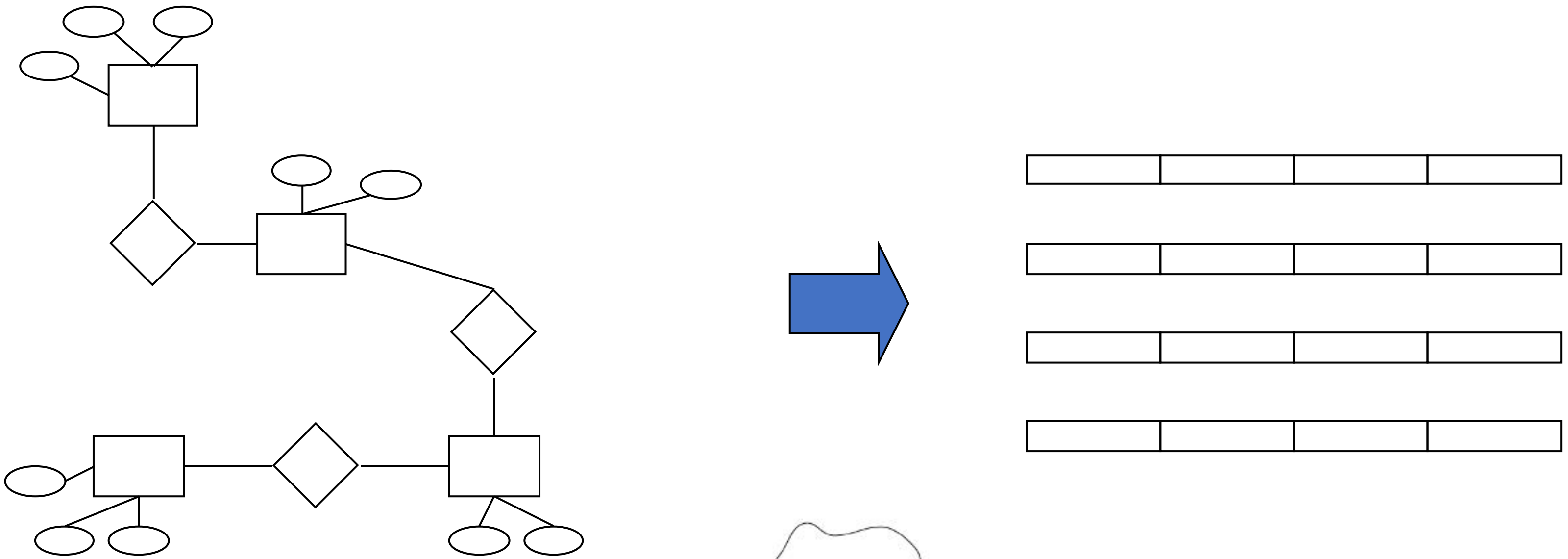
Database System Design Process

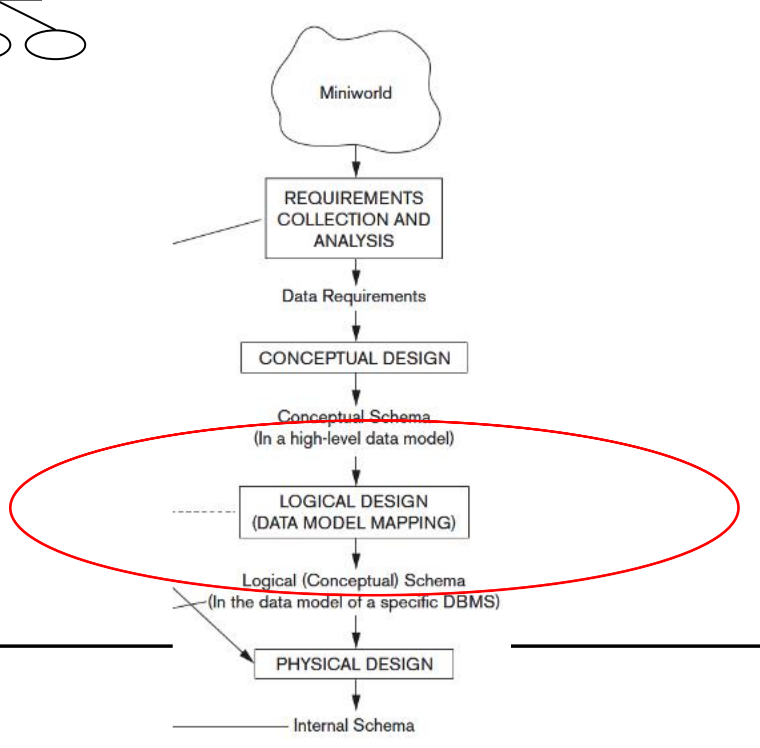
- Database design
  - Requirement specification, analysis
  - Conceptual design
  - Logical design
  - Physical design



LINKÖPING UNIVERSITY

EER to database schema (Logical design)

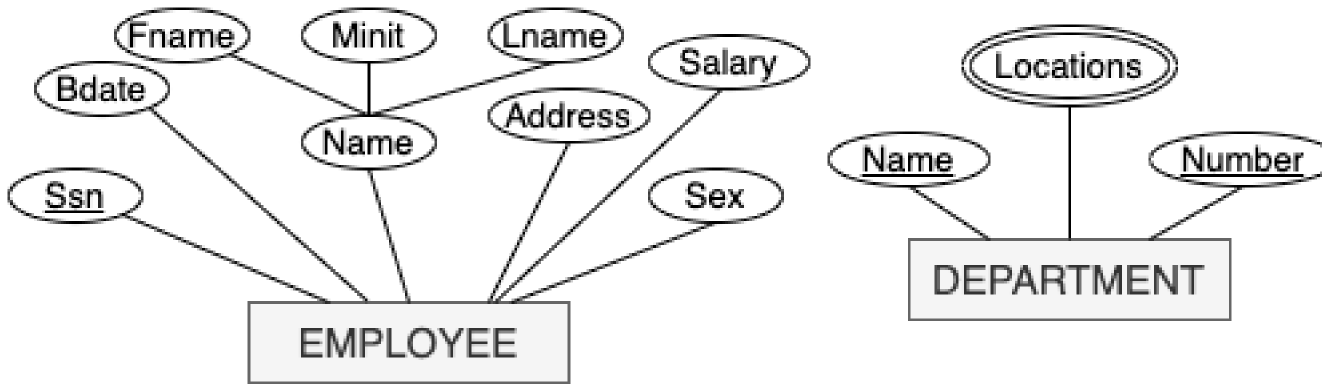




EER to database schema

Step 1: Map Regular Entity Types

- For each regular (strong) entity type *E*, create a relation *R* that includes all the simple attributes of *E*



EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary

DEPARTMENT

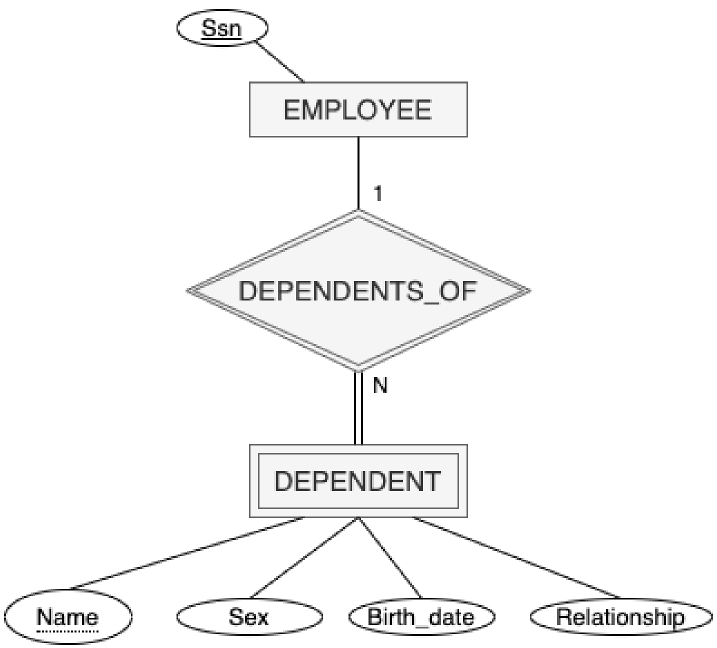
Dname	<u>Dnumber</u>

LINKÖPING UNIVERSITY

EER to database schema

Step 2: Map Weak Entity Types

- For each weak entity type *W* with identifying entity type *E*, create a relation *R* and include all the single-valued attributes of *W* and add the primary key attributes from the relation that corresponds to *E*



DEPENDENT

Essn	Dependent_name	Sex	Bdate

EER to database schema

Step 3: Map Binary 1:1 Relationship Types

- For each binary 1:1 relationship type *R*, between *S* and *T*
- Apply one of three possible approaches:
  - 1. Foreign key approach
  - 2. Merged relationship approach
  - 3. Cross-reference or relationship relation approach (Not preferred)

- 1. Foreign key approach
  - Add primary key of one participating relation as foreign key attribute of the other relation, which will also represent *R*
    - If only one side is total, choose it to represent *R*
  - Declare foreign key attribute as unique
  - Add single-valued attributes of relationship type as attribute of *R*

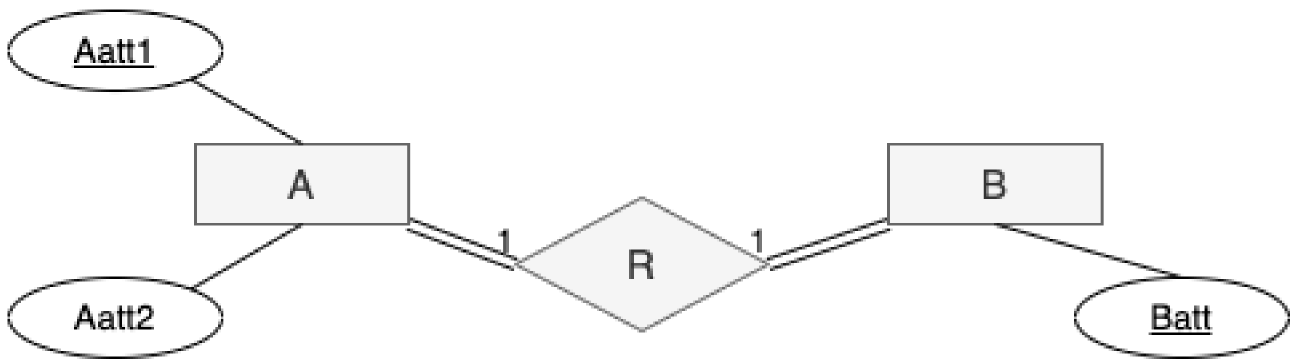
EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

- 2. Merged relationship approach
  - Possible only if both participations are total
  - Combine the two relation schemas into one, which will also represent *R*
  - Make one of the primary keys unique instead
  - Add single-valued attributes of relationship type as attributes of *R*



AB

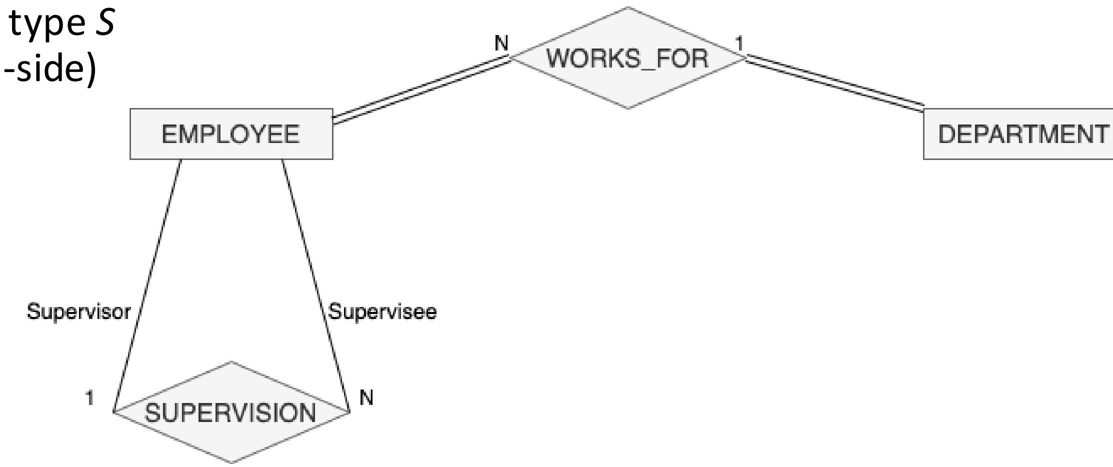
<u>**Aattr1**</u>	**Aatt2**	**Batt**

Step 4: Map 1:N Relationship Types

- For each binary 1:N relationship type *R*, between *S* and *T* (every *S* is related to many *T,* every *T* is related to one *S*)
- Foreign key approach
  - Include primary key of entity type *S* (1-side) as foreign key in *T* (N-side)

EER to database schema

- Foreign key approach
  - Include primary key of entity type *S* (1-side) as foreign key in *T* (N-side)



EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

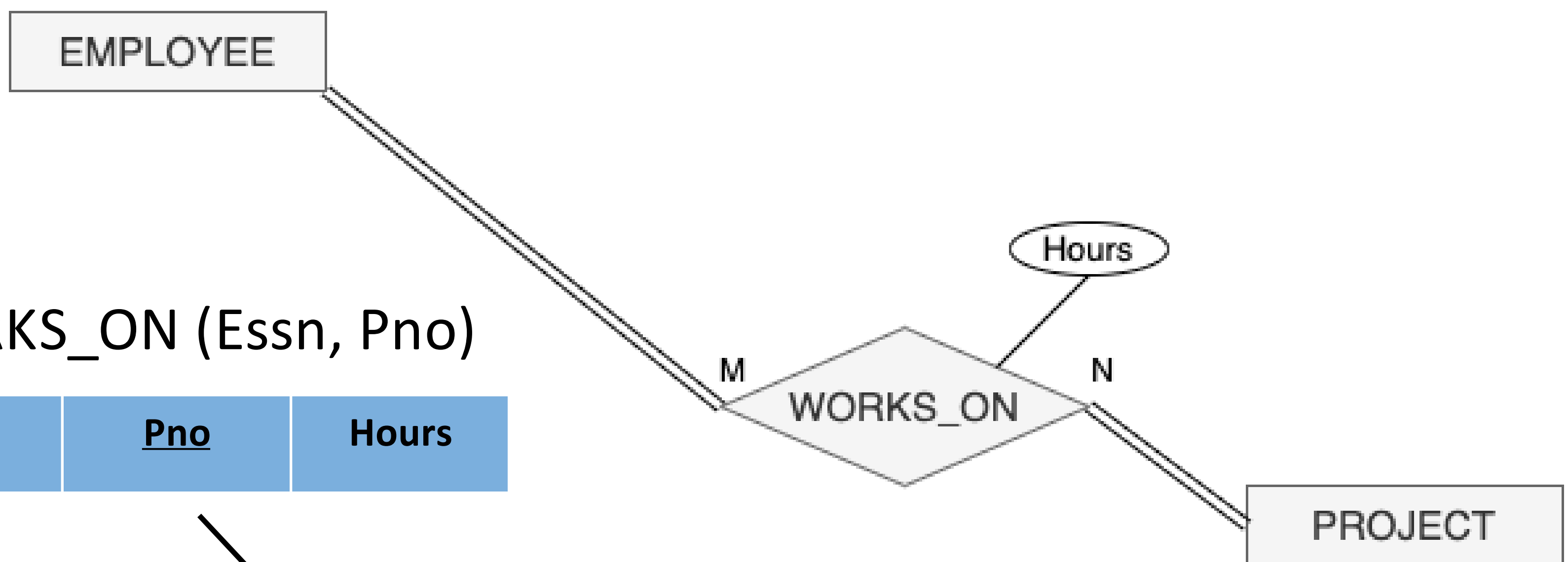
Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date

Step 5: Binary M:N and Higher Order Relationship Types

- For each binary M:N relationship type or ternary or higher order relationship type, create a new relation *S*
- Approach
  - Include primary key of participating entity types as foreign key attributes in *S*
  - Make all these attributes together as a primary key of *S*
  - Include any simple attributes of relationship type in *S*

EER to database schema

- Approach
  - Include primary key of participating entity types as foreign key attributes in *S*
  - Make all these attributes together as a primary key of *S*
  - Include any simple attributes of relationship type in *S*



WORKS_ON (Essn, Pno)

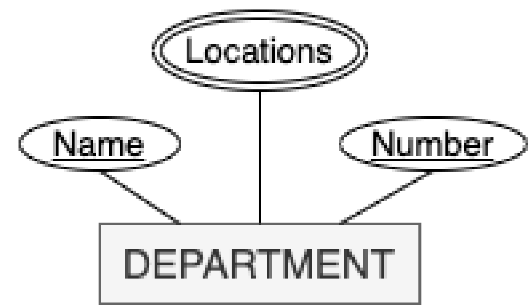
<u>Essn</u>	**<u>Pno</u>**	**Hours**

EMPLOYEE

PROJECT

Step 6: Map Multivalued Attributes

- For each multivalued attribute *A*, create a new relation *R* that contains an attribute corresponding to *A*, and the attribute *K*—as a foreign key in *R* – of the relation that represents the entity type or relationship type that has *A* as a multivalued attribute.
- The primary key of *R* is the combination of all its attributes



DEPT_LOCATIONS

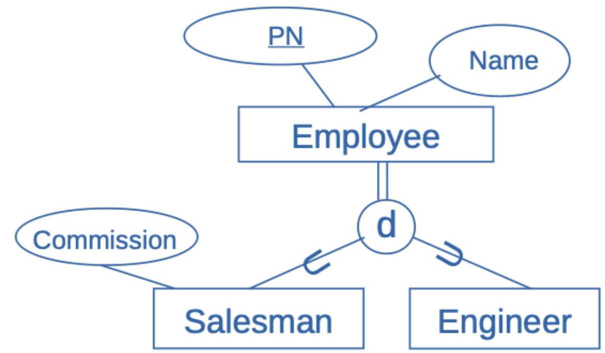
<u>Dnumber</u>	<u>Dlocation</u>

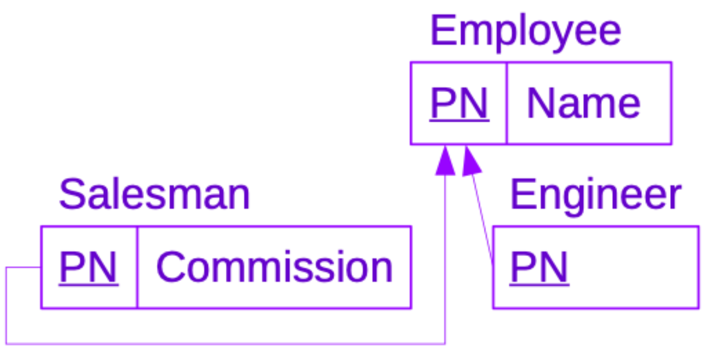
DEPARTMENT

Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date

Options for mapping specialization/generalization

- For any specialization (total or partial, disjoint or overlapping)
  - *1. Separate relation per subtype*





  - *2. Single relation with Boolean type attributes for every subtype, add all attributes of all subtypes*

Employee

PN	Name	isSalesman	isEngineer	Commission

Options for mapping specialization/generalization

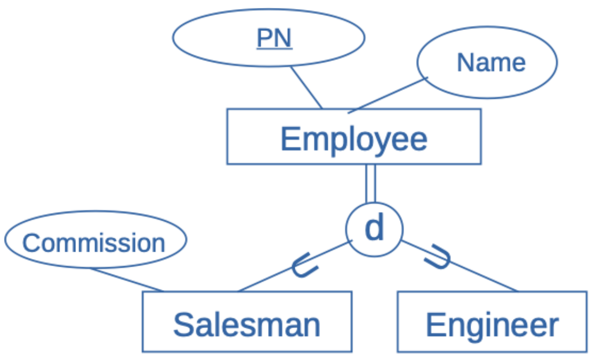
- For total specializations only
  - *Separate relation per subtype only*
  - *Overlapping subtypes will result in multiple tuples per entity*

Salesman

<u>PN</u>	Name	Commission

Engineer

<u>PN</u>	Name

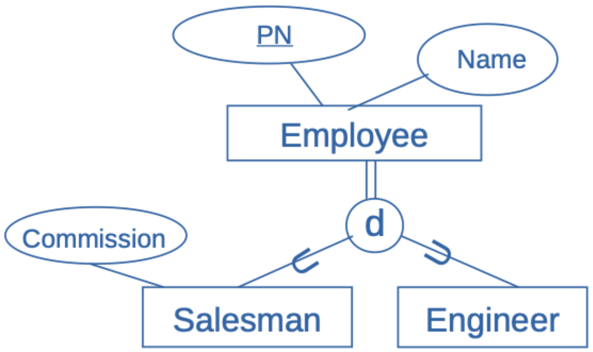


Options for mapping specialization/generalization

- For disjoint specializations only
  - *Single relation with one type attribute*
  - *Type or discriminating attribute indicates subtypes of tuple*
  - *Might require many NULL values if several specific attributes exist in subtypes*

Employee

<u>PN</u>	Name	Type	Commission



Querying RDBs using SQL

-- How is the information accessed?

SQL (Structured Query Language)

- Developed by IBM Research
- Used in many database systems
- Statements for data definitions, queries and updates
- Tutorial:
  - https://www.w3schools.com/sql/

Relational Schema

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPER_SSN	DNO

DEPARTMENT

DNAME	DNUMBER	MGR_SSN	MGR_START_DATE

DEPT_LOCATIONS

DNUMBER	DLOCATION

Relational Schema (cont'd)

PROJECT

PNAME	<u>PNUMBER</u>	PLOCATION	DNUM

WORKS_ON

<u>ESSN</u>	<u>PNO</u>	HOURS

DEPENDENT

<u>ESSN</u>	<u>DEPENDENT_NAME</u>	SEX	BDATE

SQL – Creating Tables

```
CREATE TABLE <tablename> (
        <colname> <datatype> [<constraint>],
        …,
        [<constraint>],
        …
   );
```

- Data types: integer, decimal, number, varchar, char, etc.
- Constraints: not null, primary key, foreign key, unique, etc.

SQL – Creating Tables (Example)

```
CREATE TABLE WORKS_ON (
    ESSN     integer,
    PNO      integer,
    HOURS    decimal(3, 1),

    constraint pk_workson primary key (ESSN, PNO),
    constraint fk_works_emp FOREIGN KEY (ESSN) references EMPLOYEE(SSN),
    constraint fk_works_proj  FOREIGN KEY (PNO) references PROJECT(PNUMBER)
);
```

SQL – Modifying Table Definitions(Example)

- Add, delete, and modify columns and constraints

ALTER TABLE *EMPLOYEE* **ADD COLUMN** *JOB VARCHAR(12);*

ALTER TABLE *EMPLOYEE* **DROP COLUMN** *ADDRESS;*

ALTER TABLE WORKS_ON **DROP FOREIGN KEY** fk_works_emp*;*

ALTER TABLE WORKS_ON **ADD CONSTRAINT** fk_works_emp
FOREIGN KEY *(ESSN)* **REFERENCES** *EMPLOYEE*(*SSN*);

- Delete a table and its definition

DROP TABLE *EMPLOYEE*;

SQL Retrieval Queries - Syntax

- Using a SELECT statement

SELECT *<attribute-list>*

FROM *<table-list>*

[WHERE *<condition>* **]** ;

- <attribute-list> is a list of column names (or expressions) whose values are to be retrieved
- <table-list> is a list of table names required to process the query
- <condition> is a Boolean expression that identifies the tuples to be retrieved by the query (if no WHERE clause, all tuples to be retrieved)

SQL Retrieval Queries -Examples

- Q1: List SSN for all employees.

```
SELECT SSN
FROM EMPLOYEE;
```

SSN
123456789
333445555
999887777
987654321
666884444
453453453
987987987
888665555

SQL Retrieval Queries -Examples

- Q2: List birth date and address for all employees whose name is `John B. Smith'.

SELECT BDATE, ADDRESS
FROM EMPLOYEE
WHERE FNAME = 'John'
AND MINIT = 'B.'
AND LNAME = 'Smith';

BDATE	ADDRESS
1965-01-09	731 Fondren, Houston, TX

SQL Retrieval Queries -Examples

- Q3: List all information about the employees of department 5.

SELECT FNAME, MINIT, LNAME,SSN, BDATE, ADDRESS, SEX, SALARY, SUPERSSN, DNO

FROM EMPLOYEE

WHERE DNO = 5;

SQL Retrieval Queries -Examples

- Q4: List all employees that work at the research department.

SELECT *

FROM EMPLOYEE, DEPARTMENT

WHERE DNAME= ‘Research’

AND DNUMBER = DNO;

EMPLOYEE

FNAME	MINIT	LNAME	<u>SSN</u>	BDATE	ADDRESS	SEX	SALARY	SUPER_SSN	DNO

DEPARTMENT

DNAME	<u>DNUMBER</u>	MGR_SSN	MGR_START_DATE

SQL Retrieval Queries -Examples

- Q5: List project number, department number and the name and address of directors of departments, for all projects that are located in Stockholm.

SELECT PNUMBER, DNUM, LNAME, ADDRESS

FROM PROJECT, DEPARTMENT, EMPLOYEE

WHERE PLOCATION = 'Stockholm'

AND DNUMBER = DNUM

AND SSN = MGRSSN;

PROJECT

PNAME	<u>PNUMBER</u>	PLOCATION	DNUM

SQL Retrieval Queries -Examples

```
SELECT PROJECT. PNUMBER, PROJECT.DNUM,
  EMPLOYEE.LNAME, EMPLOYEE.ADDRESS
FROM PROJECT, DEPARTMENT, EMPLOYEE
WHERE PROJECT.PLOCATION = 'Stockholm'
    AND DEPARTMENT.DNUMBER = PROJECT.DNUM
    AND EMPLOYEE.SSN = DEPARTMENT.MGRSSN;
```

SQL Retrieval Queries -Examples

- Q6: List first and last name for all employees together with first and last names of their bosses.

SELECT E.FNAME, E.LNAME, S.FNAME, S.LNAME

FROM EMPLOYEE E, EMPLOYEE S

WHERE E.SUPERSSN = S.SSN;

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPER_SSN	DNO

SQL Retrieval Queries -Examples

- Q7: List all salaries.

SELECT SALARY

FROM EMPLOYEE;

Tables as Sets

- SQL considers a table as a multi-set (bag), i.e. tuples can occur more than once in a table.
- Why?
  - Removing duplicates is expensive.
  - User may want information about duplicates.
  - Aggregation operators.

SALARY
30000
40000
25000
43000
38000
25000
25000
55000

SQL Retrieval Queries -Examples

- Q8: List all salaries without duplicates.

SELECT DISTINCT SALARY

FROM EMPLOYEE;

SALARY
30000
40000
25000
43000
38000
55000

SQL Retrieval Queries -Examples

- Q9: List all project numbers for projects in which an employee with name Smith works or where the leader of the department to which the project belongs is called Smith.

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPER_SSN	DNO

DEPARTMEN
T

DNAME	DNUMBER	MGR_SSN	MGR_START_DATE

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM

WORKS_ON

ESSN	PNO	HOURS

LINKÖPING UNIVERSITY

SQL Retrieval Queries -Examples

(**SELECT DISTINCT** PNUMBER
FROM PROJECT, DEPARTMENT, EMPLOYEE
WHERE DNUM = DNUMBER **AND**
MGRSSN = SSN **AND** LNAME = ‘Smith’)
UNION
(**SELECT DISTINCT** PNUMBER
FROM PROJECT, WORKS_ON, EMPLOYEE
WHERE PNO = PNUMBER **AND**
ESSN = SSN **AND** LNAME = ‘Smith’);

SQL Retrieval Queries -Examples

- Q10: List all employees that live in Stockholm.

SELECT FNAME, LNAME

FROM EMPLOYEE

WHERE ADDRESS **LIKE** ‘%Stockholm%’;

SQL Retrieval Queries -Examples

- Q11: List names for all employees that are born in the 1950s.

SELECT FNAME, LNAME

FROM EMPLOYEE

WHERE BDATE **LIKE** ‘195_’;

SQL Retrieval Queries -Examples

- Q12: List names and salaries for all employees that work with ProductX in case they would receive a raise of 10%.

```
SELECT FNAME, LNAME, 1.1 * SALARY
FROM EMPLOYEE, WORKS_ON, PROJECT
WHERE SSN = ESSN
   AND PNO = PNUMBER
   AND PNAME = 'PRODUCTX';
```

SQL Retrieval Queries -Examples

- Q13: List all employees in department 5 with a salary between 30,000 and 40,000.

SELECT *

FROM EMPLOYEE

WHERE DNO = 5 **AND**

(SALARY **BETWEEN** 30000 **AND** 40000);

SQL Retrieval Queries -Examples

- Q14: List all employees and the projects they work with sorted with respect to department and within the department sorted alphabetically

SELECT DNAME, LNAME, FNAME, PNAME

FROM DEPARTMENT, EMPLOYEE, PROJECT, WORKS_ON

WHERE PNO = PNUMBER **AND** SSN = ESSN **AND** DNO = DNUMBER

ORDER BY DNAME;

SQL Retrieval Queries -Examples

SELECT DNAME, LNAME, FNAME, PNAME
FROM DEPARTMENT, EMPLOYEE, PROJECT, WORKS_ON
WHERE PNO = PNUMBER **AND** SSN = ESSN **AND** DNO = DNUMBER
ORDER BY DNAME **ASC**;

SELECT DNAME, LNAME, FNAME, PNAME
FROM DEPARTMENT, EMPLOYEE, PROJECT, WORKS_ON
WHERE PNO = PNUMBER **AND** SSN = ESSN **AND** DNO = DNUMBER
ORDER BY DNAME **DESC**;

SQL Retrieval Queries -Examples

- Q15: List SSN for all employees that work with the same project at the same times as the person with SSN '123456789' (John Smith).

SELECT ESSN
FROM WORKS_ON
WHERE (PNO, HOURS) **IN**
 (**SELECT** PNO, HOURS
 FROM WORKS_ON
 WHERE ESSN = '123456789');

SQL Retrieval Queries -Examples

SELECT E.ESSN

FROM WORKS_ON E, WORKS_ON JS

WHERE JS.ESSN = '123456789'

  AND E.PNO = JS.PNO

  AND E.HOURS = JS.HOURS;

SQL Retrieval Queries -Examples

- Q16: List all employees whose salary is higher than the salaries of the employees who work at department 5.

```
SELECT LNAME, FNAME
FROM EMPLOYEE
WHERE SALARY > ALL
    (SELECT SALARY
    FROM EMPLOYEE
    WHERE DNO = 5);
```

SQL Retrieval Queries -Examples

- Q17: List all employees whose salary is higher than the salary of some employee who works at department 5.

```
SELECT LNAME, FNAME
FROM EMPLOYEE
WHERE SALARY > SOME
    (SELECT SALARY
    FROM EMPLOYEE
    WHERE DNO = 5);
```

SQL Retrieval Queries -Examples

- Q18: List all employees that do not have a relative at the company.

```
SELECT LNAME, FNAME
FROM EMPLOYEE
WHERE NOT EXISTS
    (SELECT *
    FROM DEPENDENT
    WHERE SSN = ESSN);
```

SQL Retrieval Queries -Examples

- Q19: List all department managers that have at least one relative at the company.

```
SELECT LNAME, FNAME
FROM EMPLOYEE
WHERE EXISTS
    (SELECT *
    FROM DEPARTMENT
    WHERE SSN = MGRSSN)
    AND EXISTS
    (SELECT *
    FROM DEPENDENT
    WHERE SSN = ESSN);
```

SQL Retrieval Queries -Examples

- Q20: List SSN for all employees that work with project 1, 2 or 3.

SELECT DISTINCT ESSN

FROM WORKS_ON

WHERE PNO **IN** (1, 2, 3);

SQL Retrieval Queries -Examples

- Q21: List all employees that do not have a boss.

SELECT FNAME, LNAME
FROM EMPLOYEE
WHERE SUPERSSN **IS NULL**;

SQL Retrieval Queries -Examples

- Q22: List the sum, the highest, lowest and average of the salaries of the employees of the research department.

SELECT SUM(SALARY), **MAX**(SALARY), **MIN**(SALARY), **AVG**(SALARY)

FROM EMPLOYEE, DEPARTMENT

WHERE DNAME = 'Research'

AND DNO = DNUMBER;

SQL Retrieval Queries -Examples

- Q23: List the number of employees.

SELECT COUNT(*)
FROM EMPLOYEE;

SQL Retrieval Queries -Examples

- Q24: List for each department the department number, the number of employees and the average salary.

```
SELECT DNO, COUNT(*), AVG(SALARY)
FROM EMPLOYEE
GROUP BY DNO;
```

DNO	COUNT(*)	AVG_SALARY
5	4	33250
4	3	31000
1	1	55000

SQL Retrieval Queries -Examples

- Q25: List for each project the project number, project name and the number of employees that work with the project.

```
SELECT PNUMBER, PNAME, COUNT(*)
FROM PROJECT, WORKS_ON
WHERE PNUMBER = PNO
GROUP BY PNUMBER, PNAME;
```

SQL Retrieval Queries -Examples

- Q26: List for each project with at least 2 employees the project number, project name and number of employees that work with the project.

SELECT PNUMBER, PNAME, **COUNT**(*)

FROM PROJECT, WORKS_ON

WHERE PNUMBER = PNO

GROUP BY PNUMBER, PNAME

HAVING COUNT(*) > 1;

Extended SELECT Syntax

- **SELECT** *<attribute-list and function-list>*

  FROM *<table-list>*
  [**WHERE** *<condition>*]
  [**GROUP BY** *<grouping attribute-list>*]

  [**HAVING** *<group condition>*]

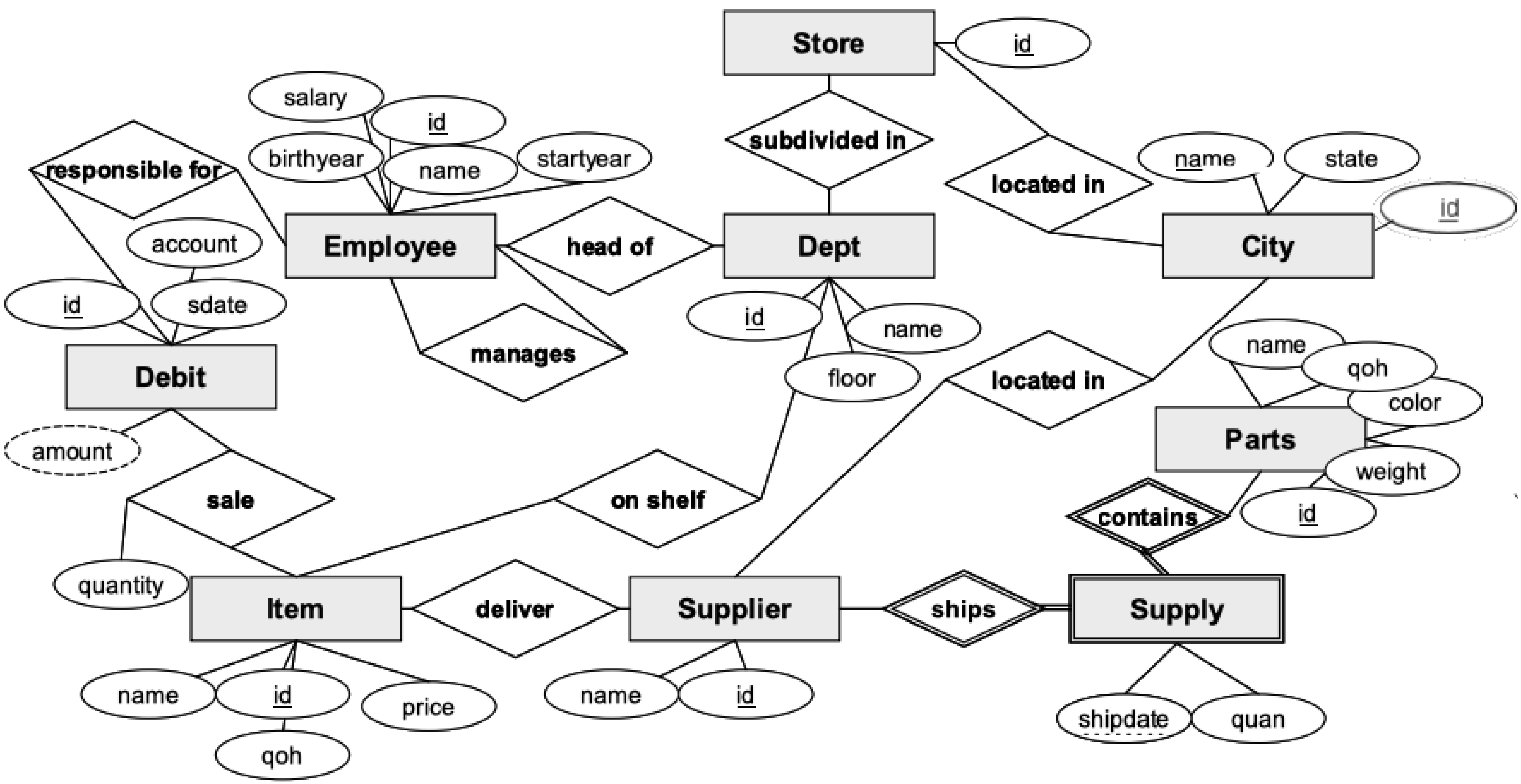
  [**ORDER BY** *<attribute-list>*]**;**

Lab RDB

Lab RDB

- https://www.ida.liu.se/~732A54/lab/rdb.shtml
- The Jonson Brothers' database
  - The lab exercises are based on a database that is used for the business of the Jonson Brothers. The Jonson Brothers is a retail company with department stores in many major US cities. The company has a large number of employees and sells a varied line of products. The company consists of a number of stores that contain a number of departments. The company has employees, who (among other things) sell items at the different stores. Sales are registered in the sale and debit tables. Items are bought from various suppliers, who also supply parts for the company's computer equipment. Deliveries of computer parts are registered in the supply table.
- 14 questions to answer

The Jonson Brothers' ER-Diagram



Tables

jbemployee

An employee is identified by an id and described by name, salary, birthyear and startyear. The id of the manager of each employee is also supplied. A null value means that the employee has no manager.

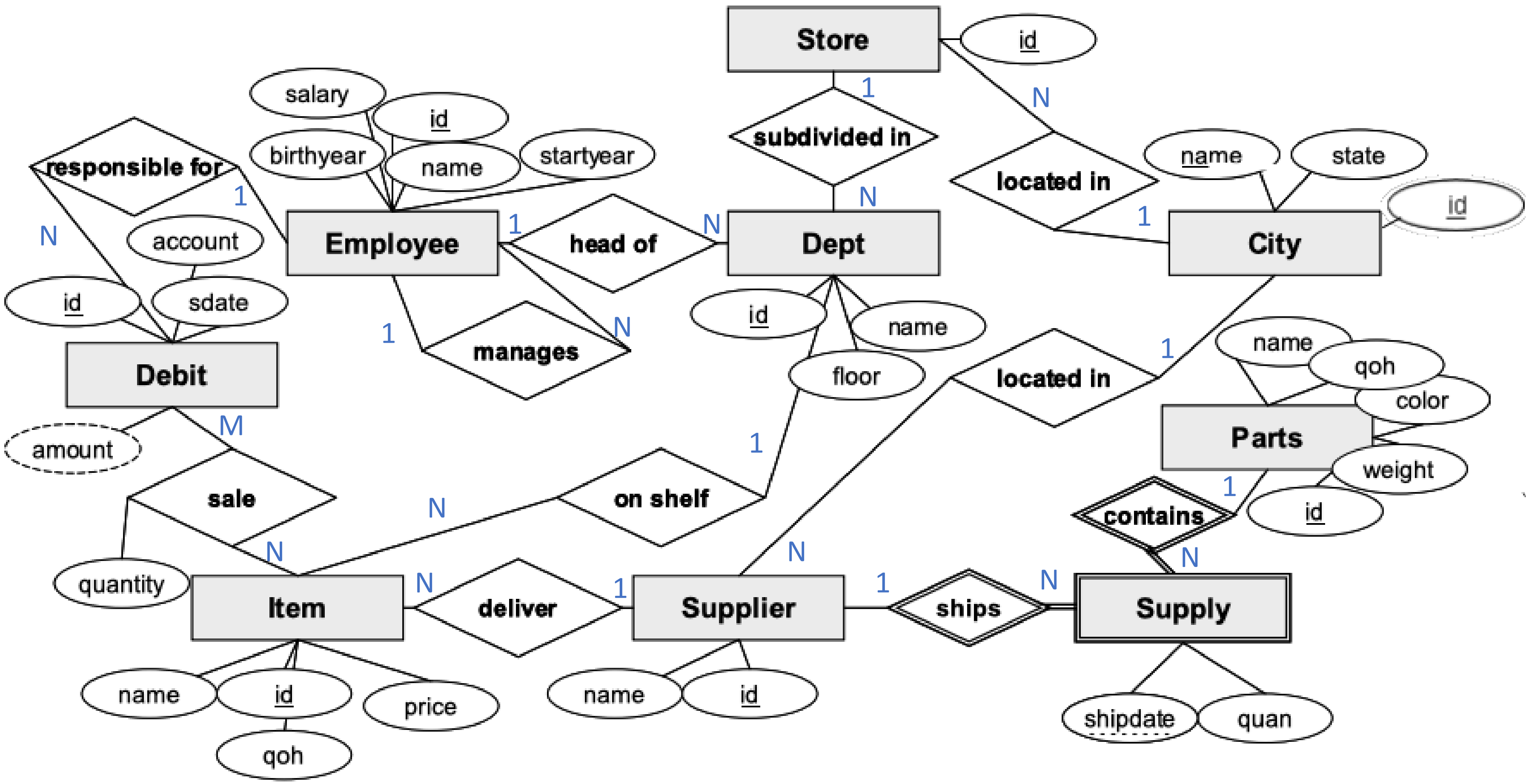
id	name	salary	manager	birthyear	startyear
157	Jones, Tim	12000	199	1940	1960
1110	Smith, Paul	6000	33	1952	1973
35	Evans, Michael	5000	32	1952	1974
129	Thomas, Tom	10000	199	1941	1962
13	Edwards, Peter	9000	199	1928	1958
215	Collins, Joanne	7000	10	1950	1971
55	James, Mary	12000	199	1920	1969
26	Thompson, Bob	13000	199	1930	1970
98	Williams, Judy	9000	199	1935	1969
32	Smythe, Carol	9050	199	1929	1967
33	Hayes, Evelyn	10100	199	1931	1963
199	Bullock, J.D.	27000	NULL	1920	1920
4901	Bailey, Chas M.	8377	32	1956	1975
843	Schmidt, Herman	11204	26	1936	1956
2398	Wallace, Maggie J.	7880	26	1940	1959
1639	Choy, Wanda	11160	55	1947	1970
5119	Bono, Sonny	13621	55	1939	1963
37	Raveen, Lemont	11985	26	1950	1974
5219	Schwarz, Jason B.	13374	33	1944	1959
1523	Zugnoni, Arthur A.	19868	129	1928	1949
430	Brunet, Paul C.	17674	129	1938	1959
994	Iwano, Masahiro	15641	129	1944	1970
1330	Onstad, Richard	8779	13	1952	1971
10	Ross, Stanley	15908	199	1927	1945
11	Ross, Stuart	12067	NULL	1931	1932

jbdept

A department is identified by an id and described by its name as well as its which store and floor it belongs to. The employee id of the manager of the department is also supplied.

id	name	store	floor	manager
35	Book	5	1	55
10	Candy	5	1	13
19	Furniture	7	4	26
20	MajorAppliances	7	4	26
14	Jewelry	8	1	33
43	Children's	8	2	32
65	Junior's	7	3	37
58	Men's	7	2	129
60	Sportswear	5	1	10
99	Giftwrap	5	1	98
1	Bargain	5	0	37
26	Linens	7	3	157
63	Women's	7	3	32
49	Toys	8	2	35
70	Women's	5	1	10
73	Children's	5	1	10
34	Stationary	5	1	33
47	JuniorMiss	7	2	129
28	Women's	8	2	32

The Jonson Brothers' ER-Diagram



Example tables

```
CREATE TABLE jbemployee
( id INT,
 name VARCHAR(20),
 salary INT,
 manager INT,
 birthyear INT,
 startyear INT,
 CONSTRAINT pk_employee PRIMARY KEY(id)
) ENGINE=InnoDB;
```

```
CREATE TABLE jbdept
(  id INT,
  name VARCHAR(20),
  store INT NOT NULL,
  floor INT,
  manager INT,
  CONSTRAINT pk_dept PRIMARY KEY(id)
) ENGINE=InnoDB;
```

```
ALTER TABLE jbdept ADD CONSTRAINT fk_dept_store FOREIGN KEY (store) REFERENCES jbstore(id);
ALTER TABLE jbdept ADD CONSTRAINT fk_dept_mgr FOREIGN KEY (manager) REFERENCES jbemployee(id) ON DELETE SET NULL;
```

Getting started

- https://www.ida.liu.se/~732A54/lab/rdb.shtml
- MySQL

RDB Lab sessions

04-01 17:15-19:00, SU24/25, Huanyu Li
04-07 15:15-17:00, SU24/25, Huanyu Li

Submission

- GitLab repositories will be assigned on April 8th