



732A54

Big Data Analytics

6hp

<http://www.ida.liu.se/~732A54>



Relational databases



Literature

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



Database methods

1. Representation and storage of data



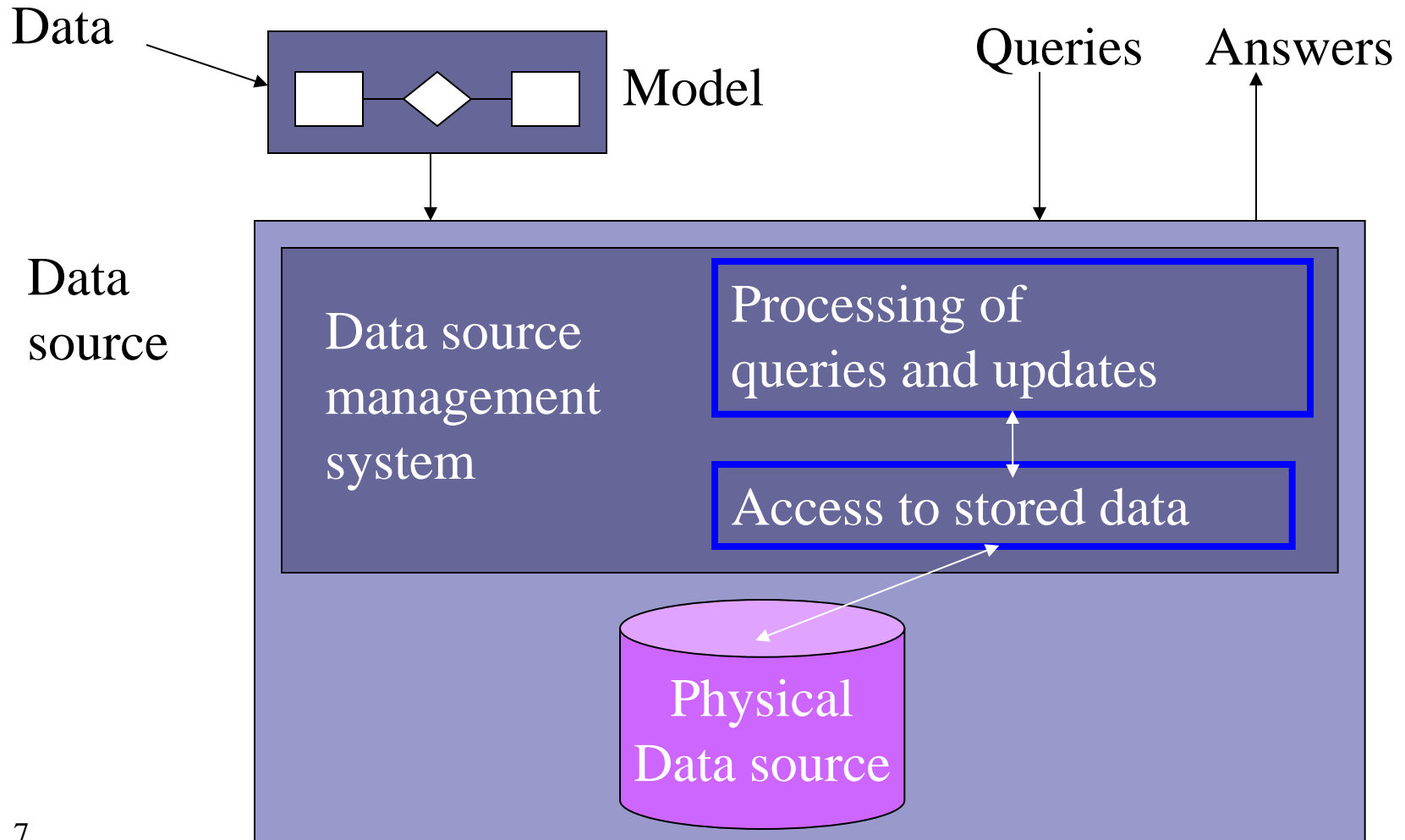
Databases

- One (of many) way(s) to store data in electronic form
- Used in every-day life: bank, reservation of hotel or journey, library search, shops

Databases

- Database management system (DBMS): a collection of programs that supports a user to create and maintain a database
- database system = database + DBMS

Data Sources





Persons

- Data source administrator
 - Data source designer
 - 'end user'
 - application programmer
-
- DBMS designer
 - tool developer
 - operator, maintenance

Issues - this course

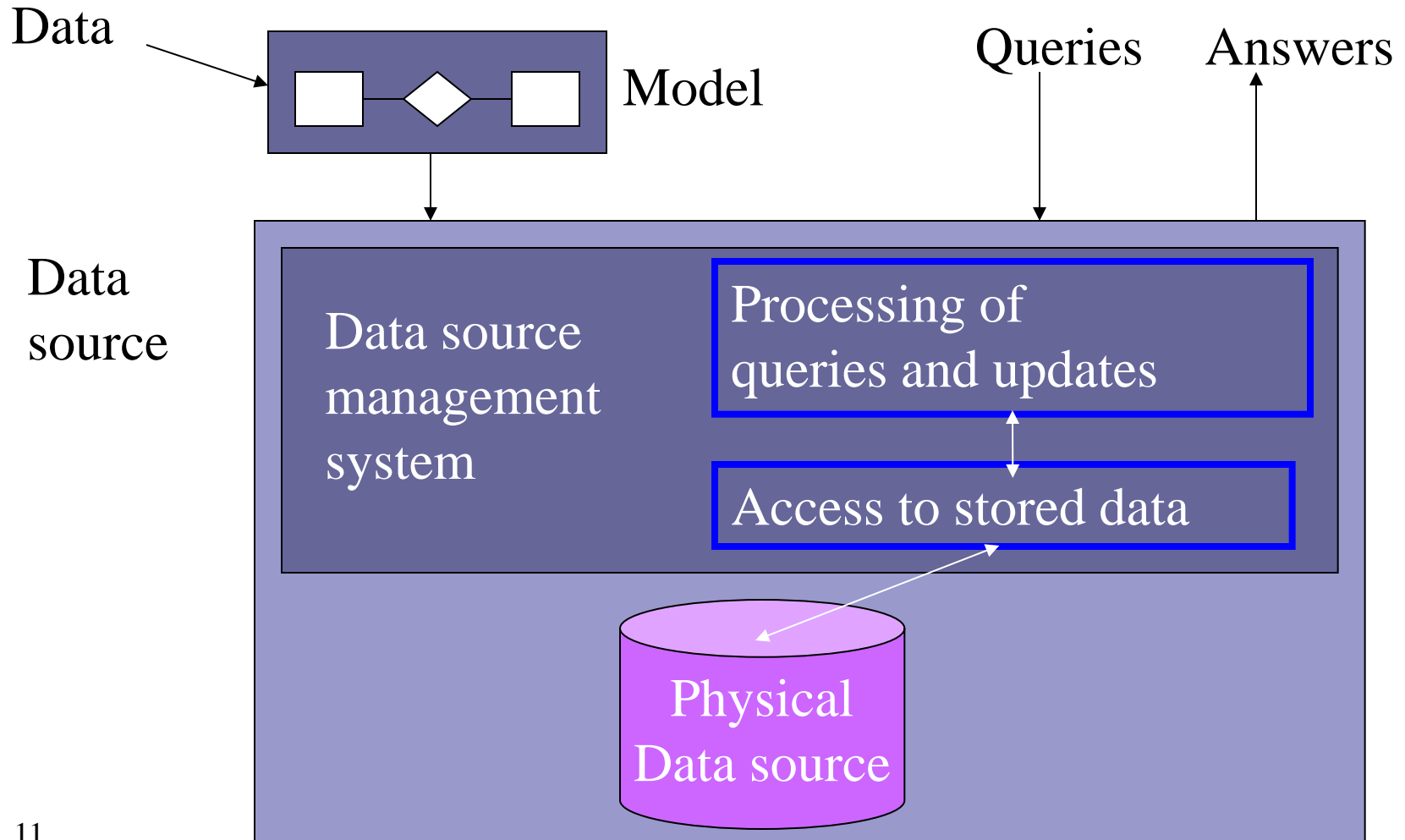
- What information is stored?
- How is the information stored?
(high level and low level)
- How is the information accessed?
(user level and system level)



Other issues

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

Data Sources





Which information is stored?

- Model of reality
 - Entity-Relationship model (ER)
 - Unified Modeling Language (UML)

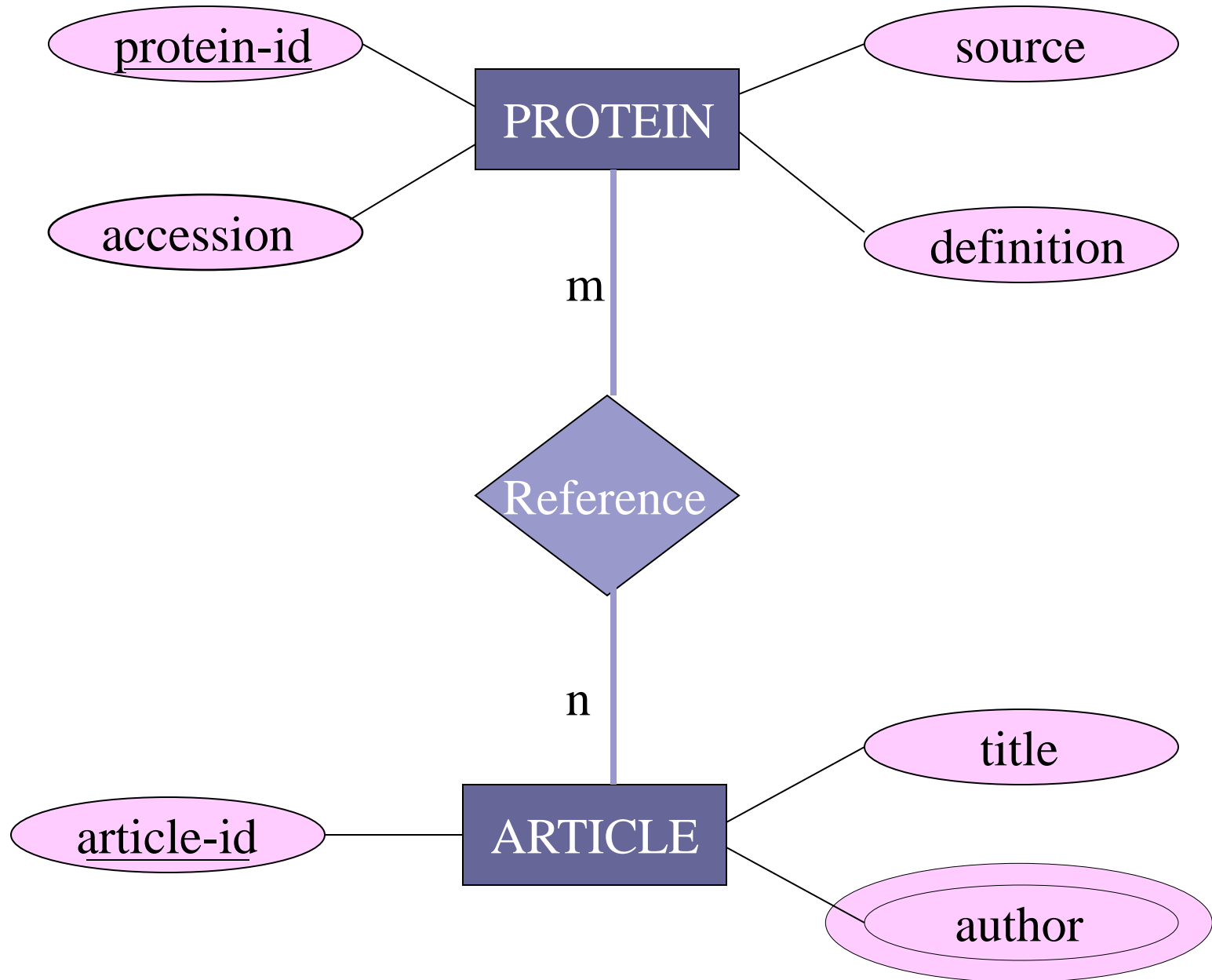


ER/EER diagram

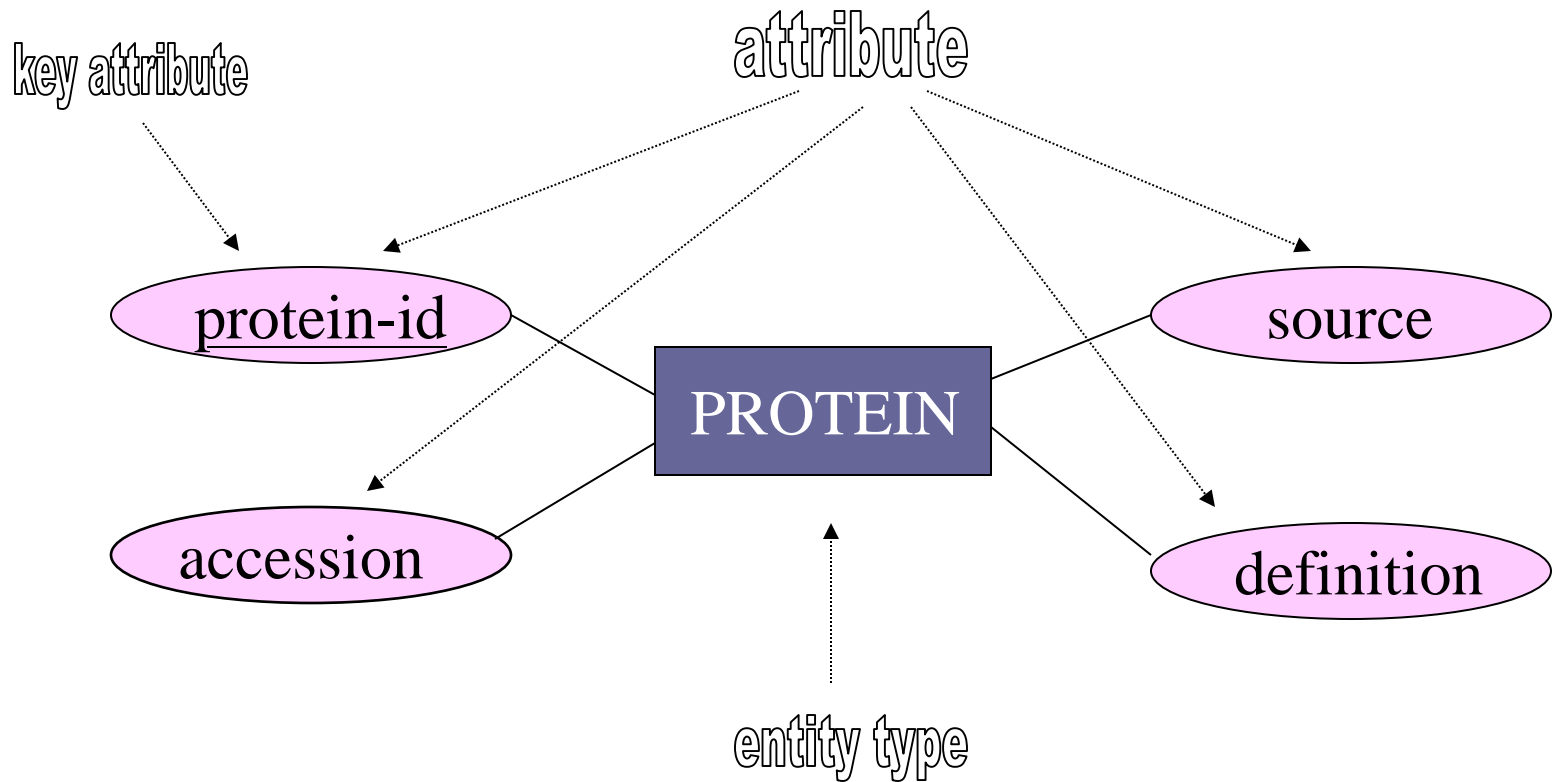
- structured way to model data, independent of type of data source
- notions:
 - entities and entity types
 - attributes
 - key attributes
 - relationships and cardinality constraints
 - sub-types (EER)

DEFINITION	Homo sapiens adrenergic, beta-1-, receptor
ACCESSION	NM_000684
SOURCE ORGANISM	human
REFERENCE	1
AUTHORS	Frielle, Collins, Daniel, Caron, Lefkowitz, Kobilka
TITLE	Cloning of the cDNA for the human beta 1-adrenergic receptor
REFERENCE	2
AUTHORS	Frielle, Kobilka, Lefkowitz, Caron
TITLE	Human beta 1- and beta 2-adrenergic receptors: structurally and functionally related receptors derived from distinct genes

Entity-relationship

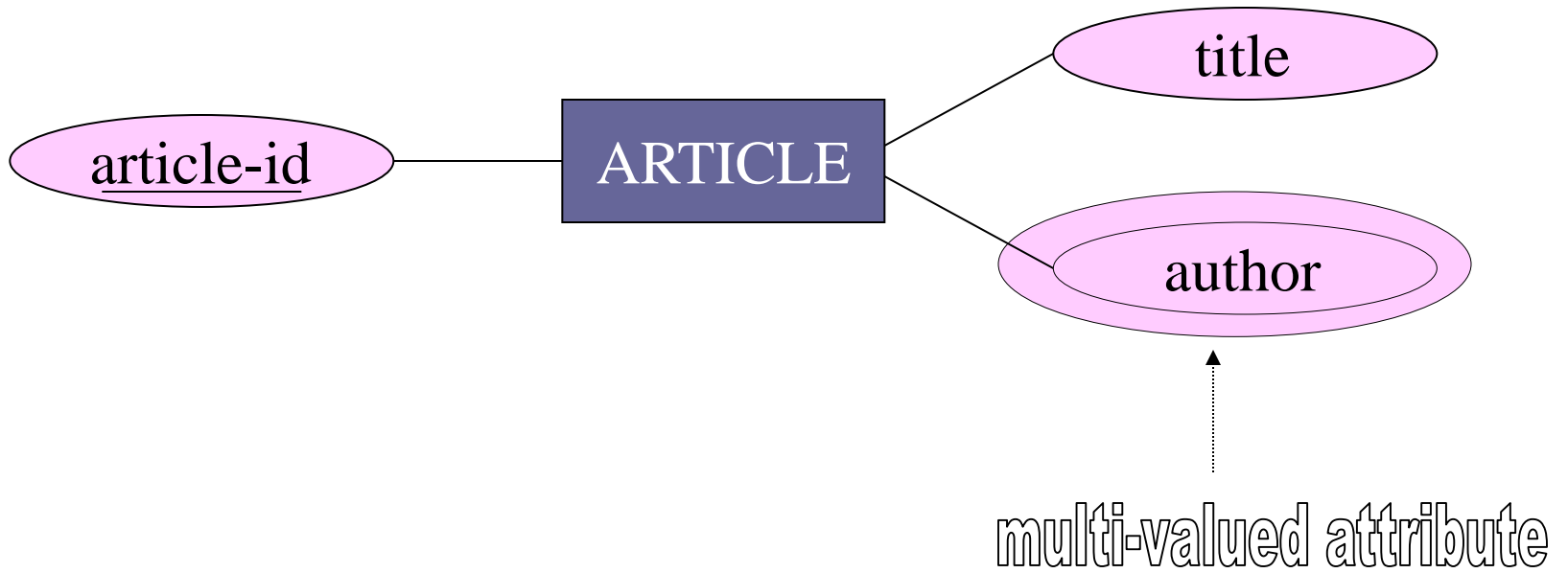


Entity-relationship

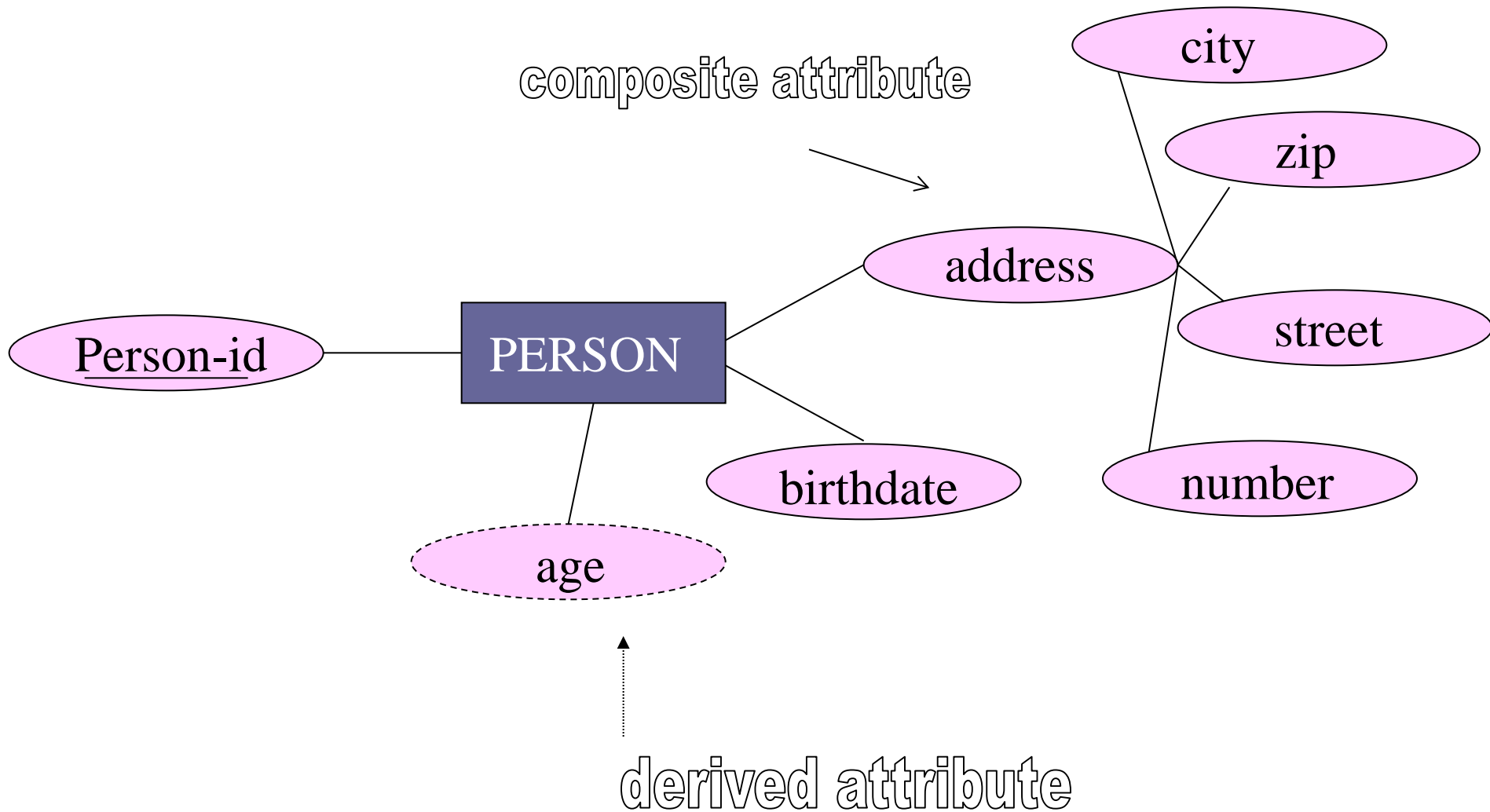


Weak entity type: type without key attribute

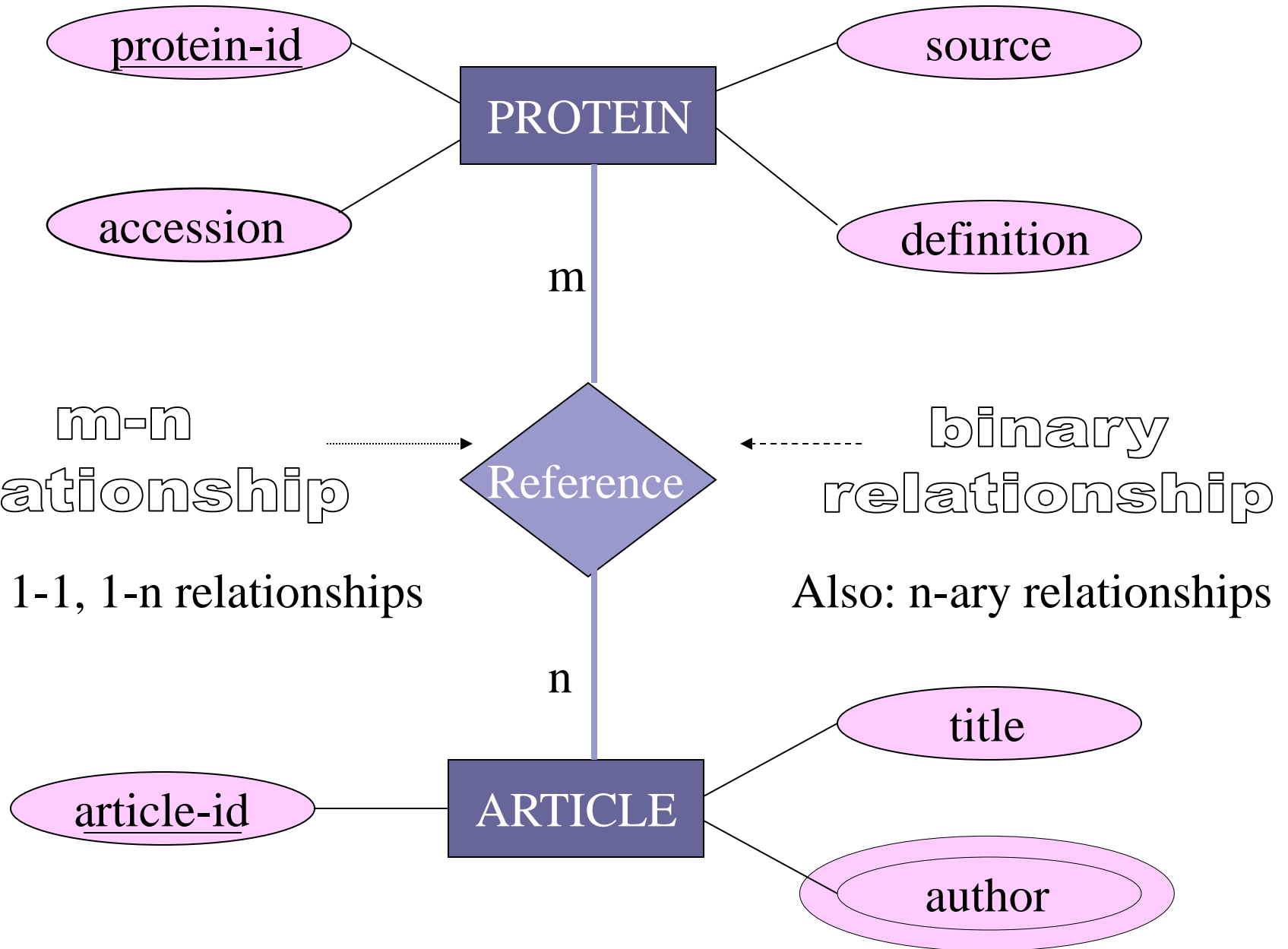
Entity-relationship



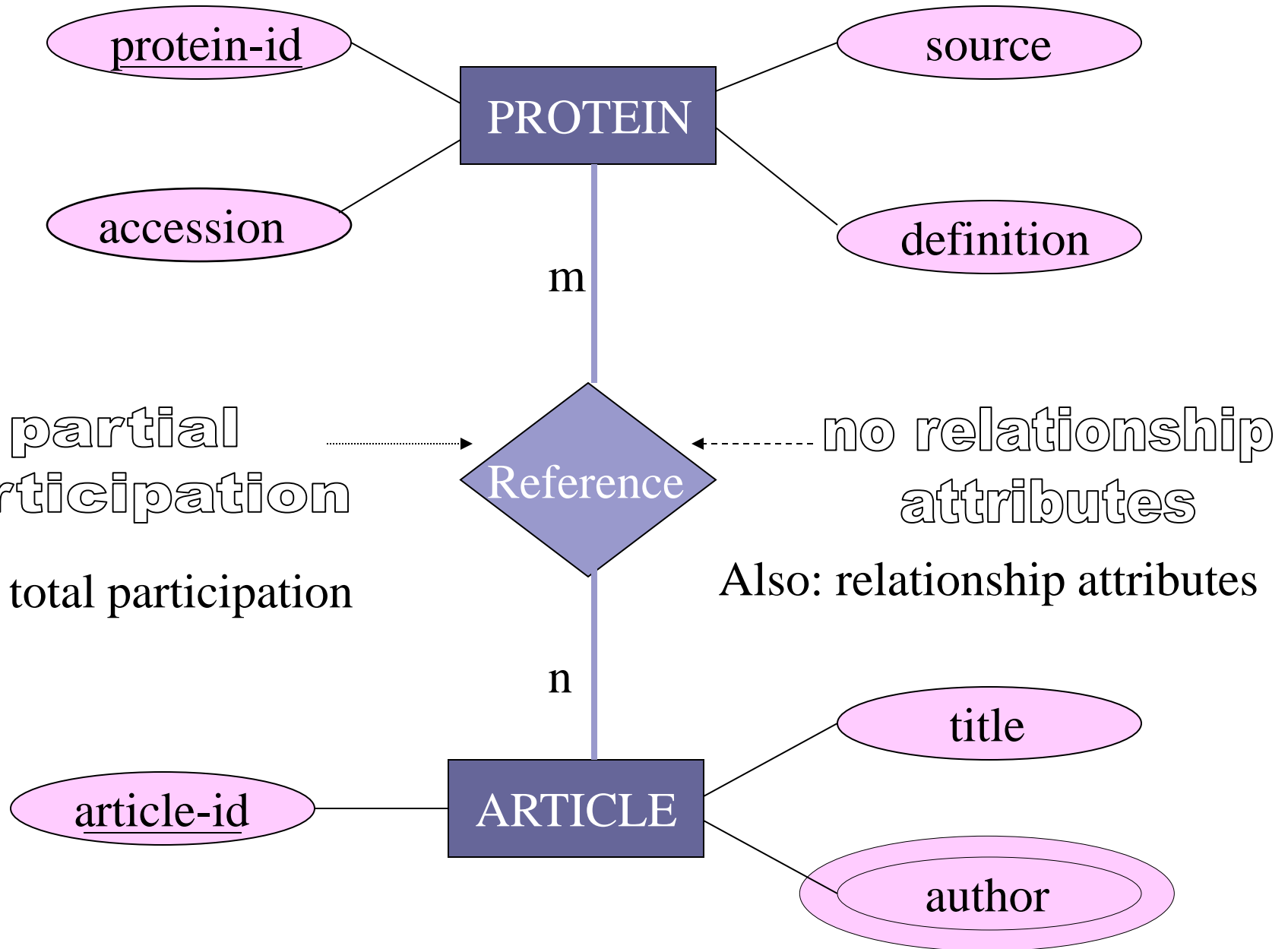
Entity-relationship



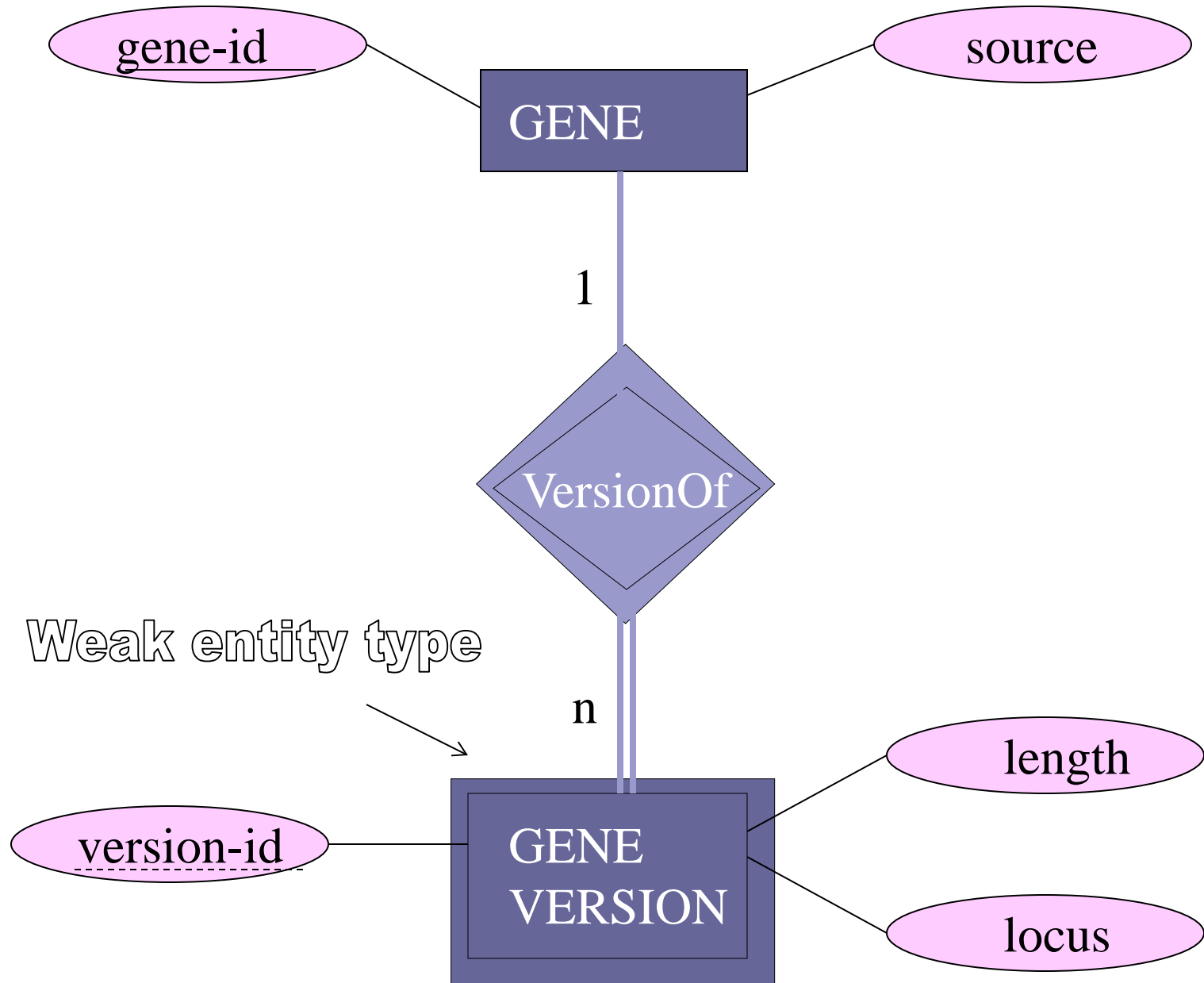
Entity-relationship



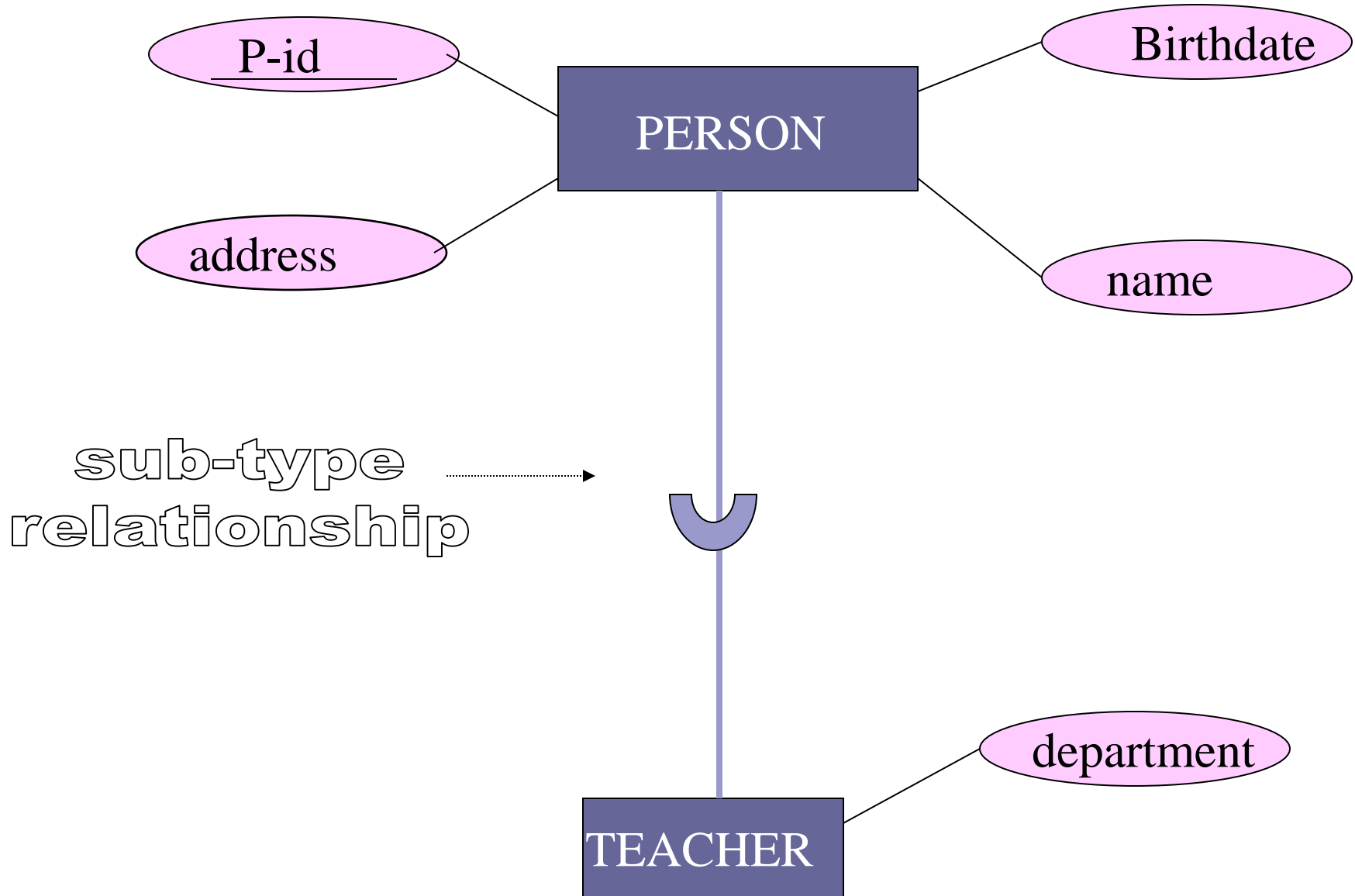
Entity-relationship



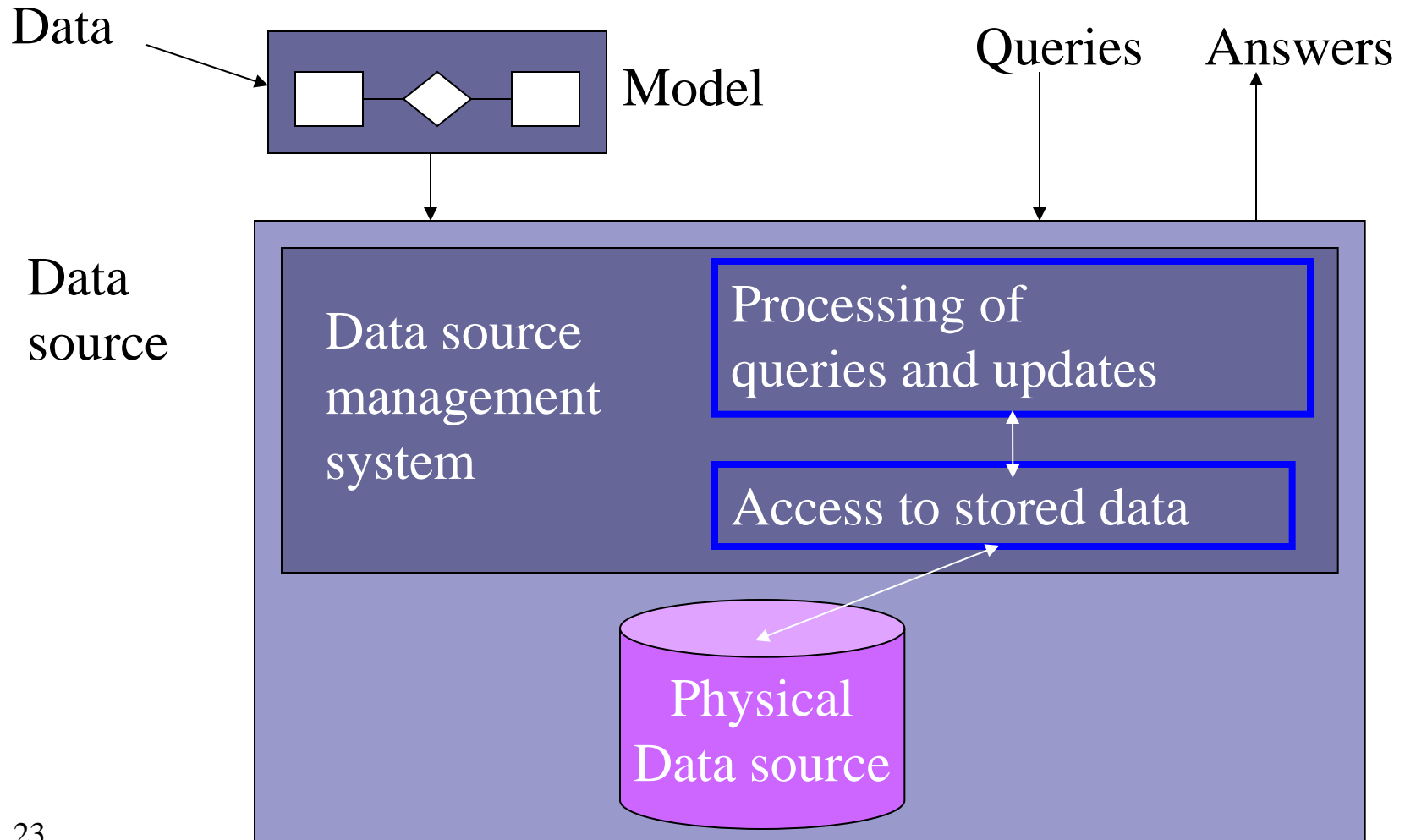
Entity-relationship



Enhanced Entity-relationship



Data Sources



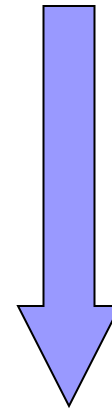
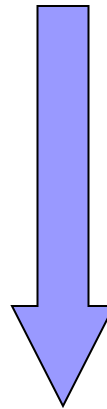
How is information stored? (high level)

How is information accessed? (user level)

- Text (IR)
- Semi-structured data
- Data models (DB)
- Rules + Facts (KB)

structure

precision



Databases

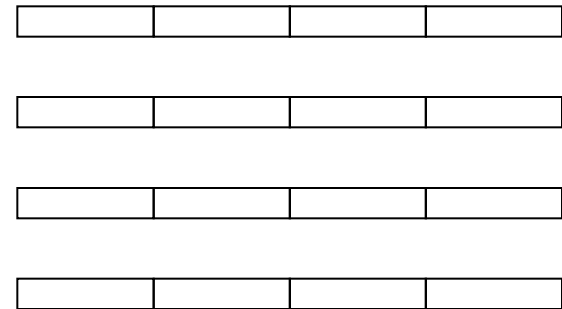
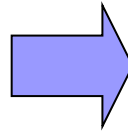
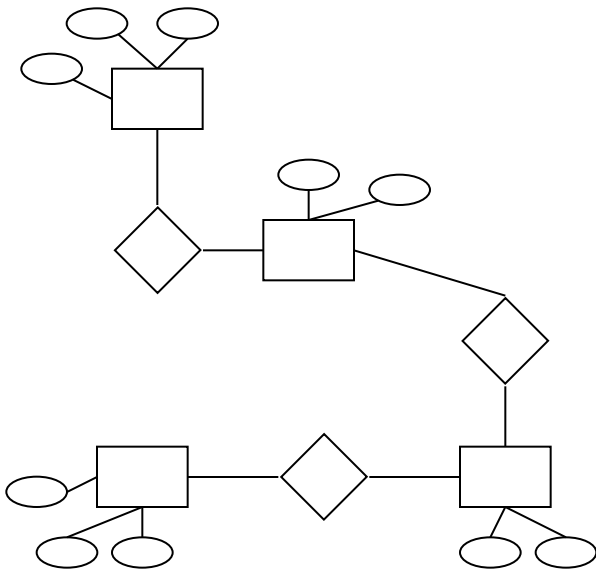
- Relational databases:
 - model: tables + relational algebra
 - query language (SQL)
- Object-oriented, extended-relational, NoSQL databases



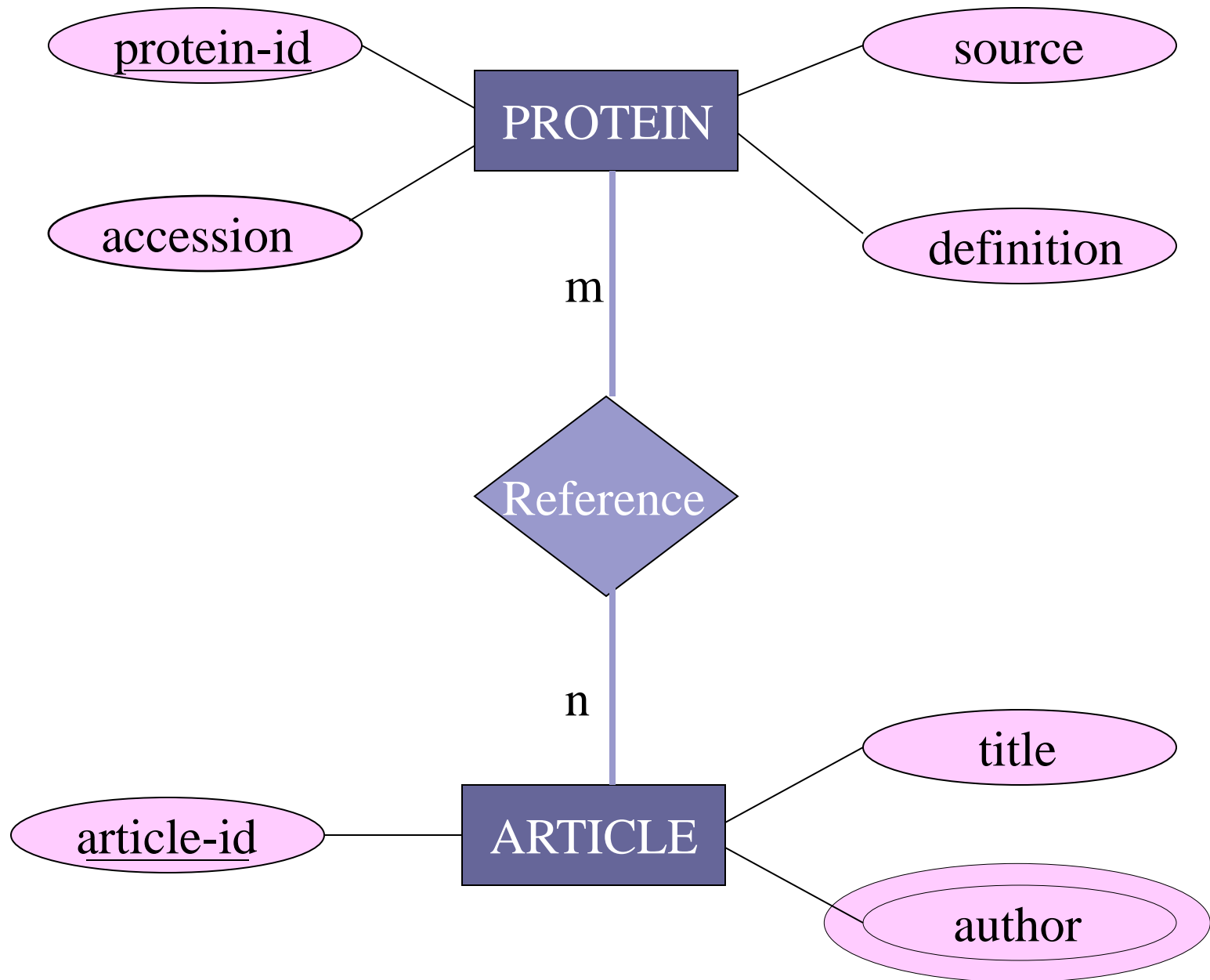
Relational Databases

- Tables = relations (~ entity type)
 - row = tuple (~ entity)
 - column = attribute (~ attribute)
- primary keys
- foreign keys

ER/EER to database schema



Entity-relationship





ER/EER to database schema

Step 1

For each (strong) entity E create a table R with the same simple attributes as the entity.



PROTEIN (PROTEIN-ID, ACCESSION, SOURCE, DEFINITION)



Author is multi-valued attribute.

ARTICLE-TITLE (ARTICLE-ID, TITLE)



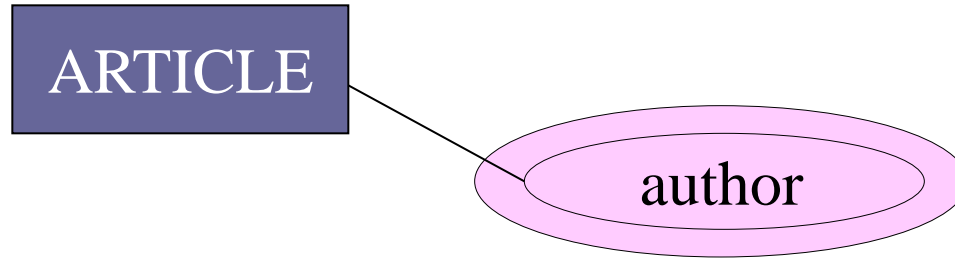
GENE (GENE-ID, SOURCE)



ER/EER to database schema

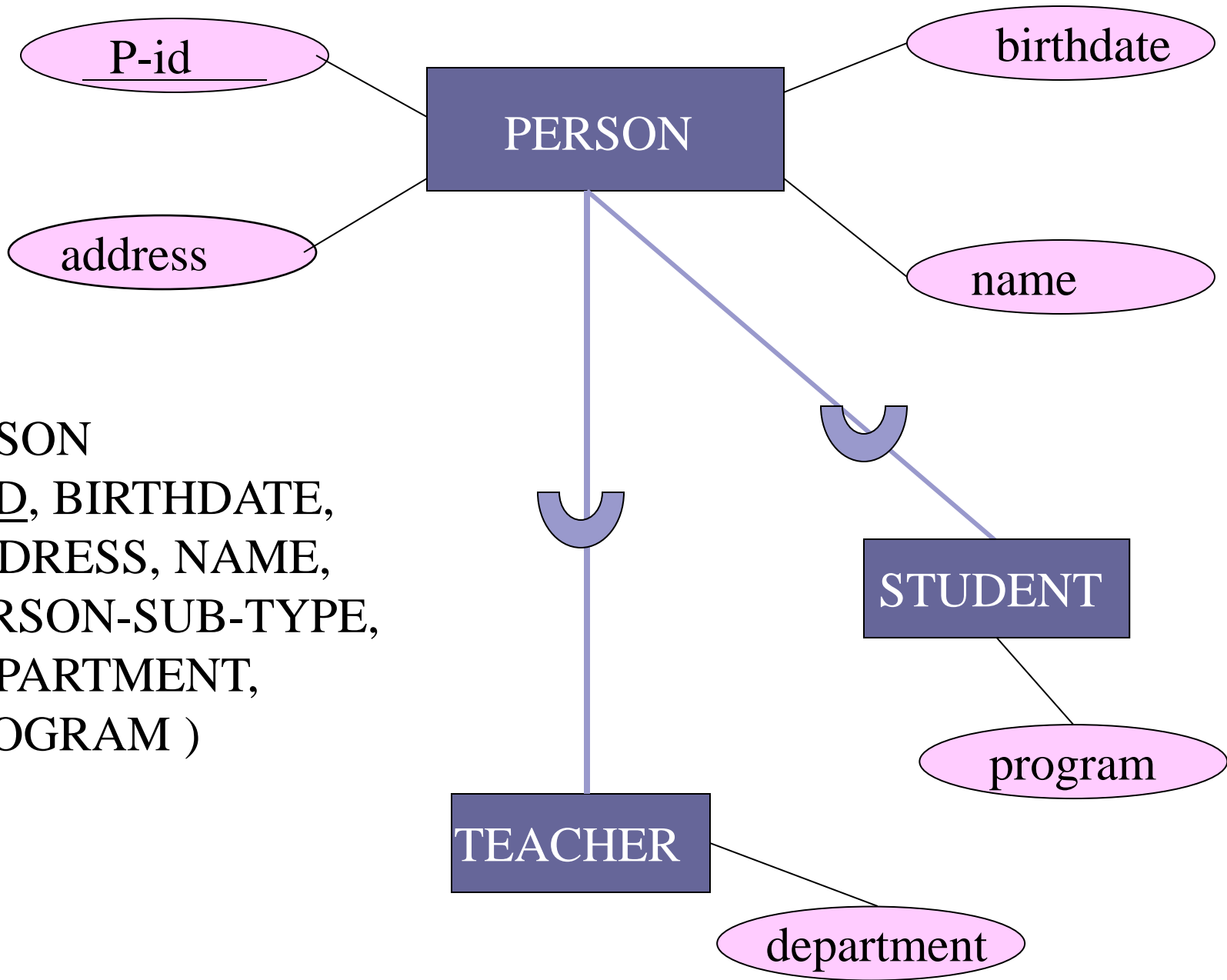
Step 6

For every multi-valued attribute A in R , create a new table that contains an attribute corresponding to A and the primary key of R as foreign key.



ARTICLE-AUTHOR (ARTICLE-ID, AUTHOR)

ARTICLE (ARTICLE-ID)



PERSON
(P-ID, BIRTHDATE,
ADDRESS, NAME,
PERSON-SUB-TYPE,
DEPARTMENT,
PROGRAM)

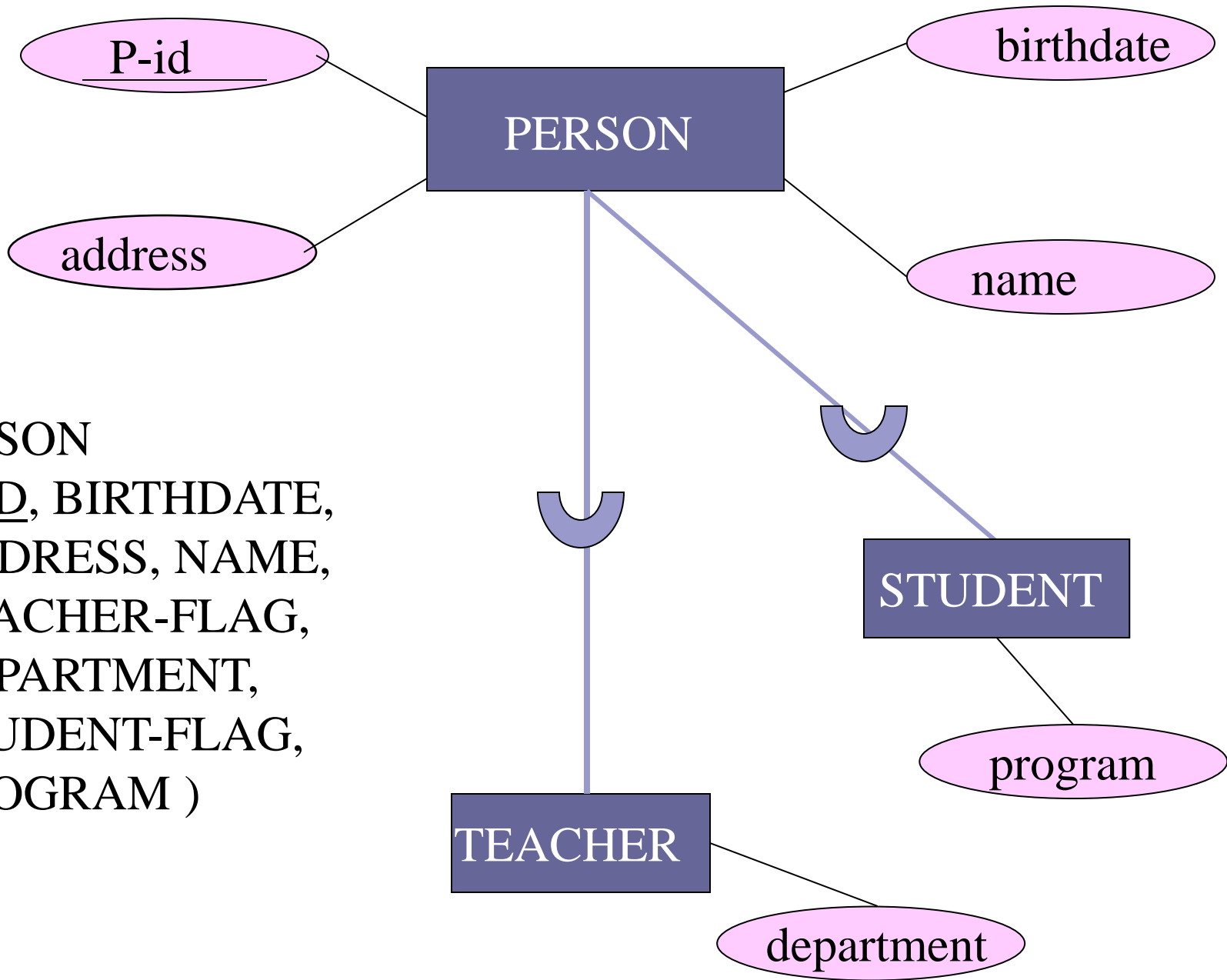


EER to database schema

Sub-type relationship

Assume type C has subtypes S_i

option 4: Create a relation R with all attributes from C and all attributes from the S_i . Add a flag attribute F_i to R for each S_i .



SQL

```
select source  
from protein  
where accession = 'NM_000684';
```

PROTEIN

PROTEIN-ID	ACCESSION	DEFINITION	SOURCE
1	NM_000684	Homo sapiens adrenergic, beta-1-, receptor	human

- WORKS-ON (ESSN, PNO, HOURS)
- PROJECT (PNAME, PNUMBER, PLOCATION, DNUM)
- DEPENDENT (ESSN, DEPENDENT-NAME, SEX, BDATE, RELATIONSHIP)



SQL syntax

SELECT *attribute-list*

FROM *table-list*

WHERE *condition*;

- attribute-list: attributes that are required
- table-list: tables that are needed to process the query
- condition: expression with logical operators (and, or , not) and equality and inequality operators; identifies the tuples that should be retrieved

- Q1: List SSN for all employees.

```
SELECT SSN  
FROM EMPLOYEE;
```

SSN

123456789

333445555

999887777

987654321

666884444

453453453

987987987

888665555

- 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

- 
- 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;
```



```
SELECT *  
FROM EMPLOYEE  
WHERE DNO = 5;
```

- 
- Q4: List name and address for all employees that work at the research department.

```
SELECT FNAME, LNAME, ADDRESS  
FROM EMPLOYEE, DEPARTMENT  
WHERE DNAME = 'Research'  
      AND DNUMBER = DNO;
```



```
SELECT *  
FROM EMPLOYEE, DEPARTMENT  
WHERE DNAME = 'Research'  
AND DNUMBER = DNO;
```

FNAME	DNO	DNAME	DNUMBER ...	MGRSTARTDATE
John	5	Research	5	... 1988-05-22
Franklin	5	Research	5	... 1988-05-22
Ramesh	5	Research	5	... 1988-05-22
Joyce	5	Research	5	... 1988-05-22

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


SELECT PNUMBER, DNUM, LNAME,
ADDRESS

FROM PROJECT, DEPARTMENT,
EMPLOYEE

WHERE PLOCATION = `Stafford'

AND DNUMBER = DNUM


AND SSN = MGRSSN;



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

- 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;
```

- 
- 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.


- 
- Q7: List all salaries.


```
SELECT SALARY  
FROM EMPLOYEE;
```

```
SELECT ALL SALARY  
FROM EMPLOYEE;
```

- 
- Q8: List all salaries without duplicates.

```
SELECT DISTINCT SALARY  
FROM EMPLOYEE;
```

- 
- 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.




**(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');**

- Q10: List all employees that live in Houston.


```
SELECT FNAME, LNAME  
FROM EMPLOYEE  
WHERE ADDRESS LIKE '%Houston%';
```

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

```
SELECT FNAME, LNAME  
FROM EMPLOYEE  
WHERE BDATE LIKE '__ _ 5%';
```

- 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';
```

- 
- 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);
```


- Q14: List all employees and the projects they work with sorted with respect to department and within the department sorted alphabetically with respect to last name and then first name.

SELECT DNAME, LNAME, FNAME,
PNAME

FROM DEPARTMENT, EMPLOYEE,
PROJECT, WORKS-ON

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


ORDER BY DNAME, LNAME, FNAME;



```
SELECT DNAME, LNAME, FNAME,  
PNAME  
FROM DEPARTMENT, EMPLOYEE,  
PROJECT, WORKS-ON  
WHERE PNO = PNUMBER AND SSN =  
ESSN AND DNO = DNUMBER  
ORDER BY DNAME DESC, LNAME  
ASC, FNAME ASC;
```

- 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');
```



```
SELECT E.ESSN  
FROM WORKS-ON E, WORKS-ON JS  
WHERE JS.ESSN = '123456789'  
AND E.PNO = JS.PNO  
AND E.HOURS = JS.HOURS;
```


- 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);
```

- 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);
```

- 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);
```

- 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);
```

- 
- 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);
```

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

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

- 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;
```

- 
- Q23: List the number of employees.

```
SELECT COUNT(*)  
FROM EMPLOYEE;
```


- 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

- 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;
```


- 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;
```



Creating new tables

```
CREATE TABLE DEPTS-INFO  
(DEPT-NUMBER integer primary key,  
DEPT-NAME varchar(15),  
NO-OF-EMPLOYEES integer,  
TOTAL-SAL integer,  
foreign key (DEPT-NUMBER) references  
DEPARTMENT(DNUMBER));
```



```
INSERT INTO DEPTS-INFO  
( DEPT-NUMBER, DEPT-NAME,  
NO-OF-EMPLOYEES, TOTAL-SAL)  
SELECT DNUMBER, DNAME,  
        COUNT (*), SUM(SALARY)  
FROM DEPARTMENT, EMPLOYEE  
WHERE DNUMBER = DNO  
GROUP BY DNUMBER, DNAME;
```