Relational translation-I (Difficulties, Criteria and Tools)

Knowledge objectives

1. Recognize the difficulty of a DB design depending on the size and origin (either new or coming from fusion) of the company
2. Enumerate the different alternatives to perform design
3. Enumerate the three mandatory phases of the design of an operational database
4. Enumerate the five optional phases of the design of an operational database
5. Name the two meanings of the NULL value
6. Enumerate the four kinds of Generalization/Specialization
7. Give three reasons to create a surrogate of a primary key

Understanding objectives

1. Explain the storage space problem generated by NULL values, and how DBMSs solve it
2. Explain the access time problem generated by NULL values, and how we should take it into account in the DB design
3. Implement a Generalization/Specialization in any of the three alternative ways to do it
4. Write the query to retrieve all the data corresponding to any of the classes in a Generalization/Specialization
5. Create a surrogate of the primary key in a table

Application objectives

1. Give an example of semantic relativism
2. Translate a DB query from natural language into SQL, given a DB schema, taking into account the presence of NULL values and ternary logic, as well as the possibility of using outer joins and tables with inheritance

Design phases

Conceptual

Logical

Physical

Design depending on the company

- New company (or just without computers)
  - Small
  - Large

- Existing company
  - Department integration
  - Merge/Absorption of companies

Design methods

- From scratch, it can be:
  - Manual (by using a semantic data model)
  - Automatic (normalization theory and algorithms)
    - Analysis
    - Synthesis

- From existing databases, we must perform
  - [Reverse engineering]
  - View integration

Design phases of an operational RDB (Mandatory)

0. Capture and abstraction of user requirements
   - Study of opportunities and specification (with pen and paper)
1. Conceptual design:
   - Create a Conceptual Schema (CS) in a semantic data model (UML)
2. Logical design:
   - Translate from the CS to the Logic Schema (LS) following the classical relational model (standard SQL)
     - **Use normalization theory**
3. Physical design:
   - Fit the LS into the RDBMS we have

Design phases of an operational RDB (Optional)

4. Quantifying data volumes and frequency of critical processes
5. Considerations regarding:
   - Response time
   - Concurrency
   - Recovery
   - Security
6. Tuning (physical structures and system parameters)
7. Performance control (monitors and query plan)
8. Report from the designer to the DBA

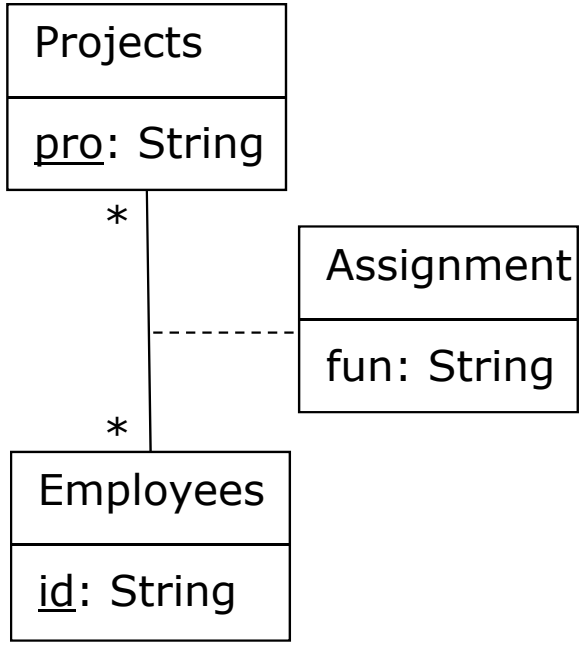
Interpretation

Conceptual Design

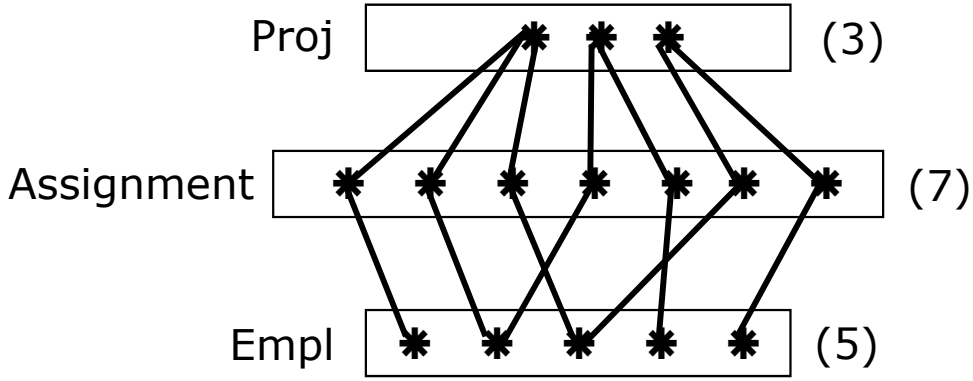
Logical Design

UML model and instantiation (Conceptual Design)

Standard UML



Instances



Relational model representation (Logical Design)

- Long

  CREATE TABLE Projects (pro CHAR(25), …);
  CREATE TABLE Employees (id CHAR(9), …);
  CREATE TABLE Assignments (…);

- Short

  Projects(pro, …)

  Assignments(empl,pro,function)

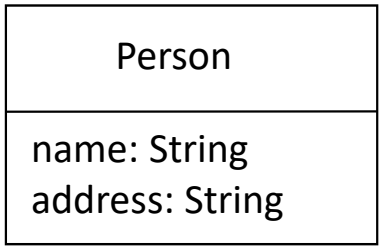
  Employees(id, …)

Only correct instantiations of the conceptual schema should be allowed in the corresponding relational translation

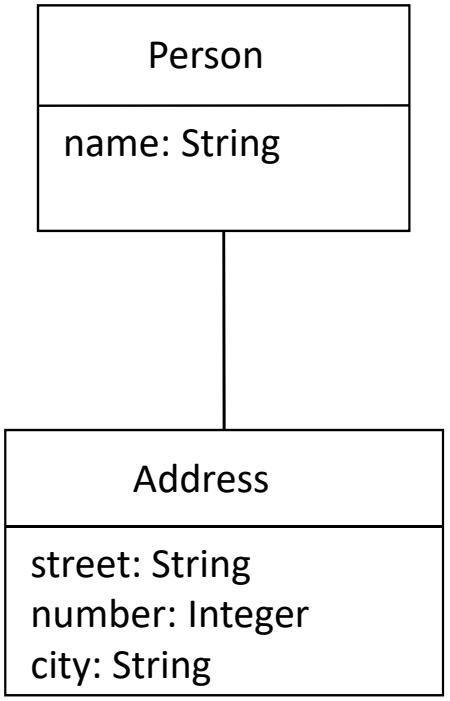
Semantic relativism

Example of semantic relativism with UML (I)

Option 1



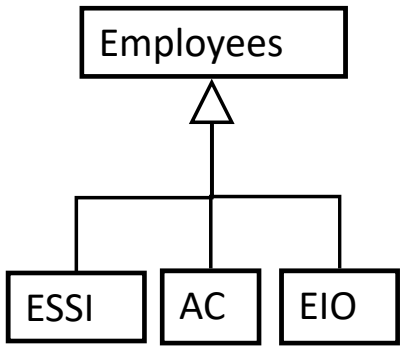
Option 2



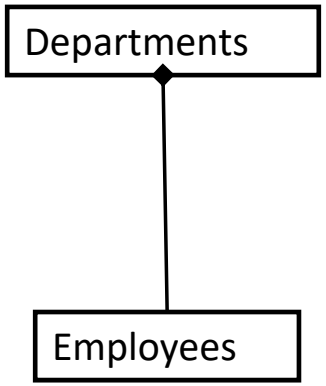
The class “Person” is the same, even if it has different attributes and associations

Example of semantic relativism with UML (II)

Option 1



Option 2



These two alternatives allow to represent basically the same reality, but what is considered metadata (i.e., subclasses of “Employees”) at LHS is considered data (i.e., instances of “Departments”) at RHS

Example of semantic relativism with UML (III)

Option 1

Mothers
child: Person mother: Person

Fathers
child: Person father: Person

Option 2

Parenthood
child: Person father: Person mother: Person

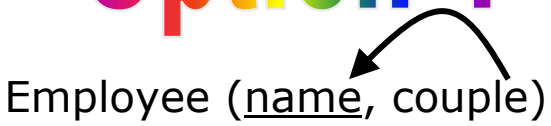
Option 3

Parenthood
child: Person parent: Person kind: {Father, Mother}

These three alternatives represent parenthood just with some subtle differences

Example of semantic relativism with tables

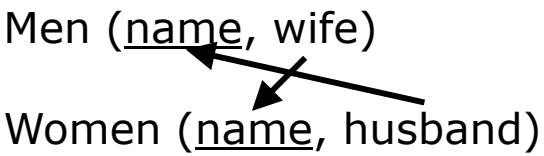
Option 1



Option 2

Marriage (husband, wife)

Option 3



Null values

Meaning

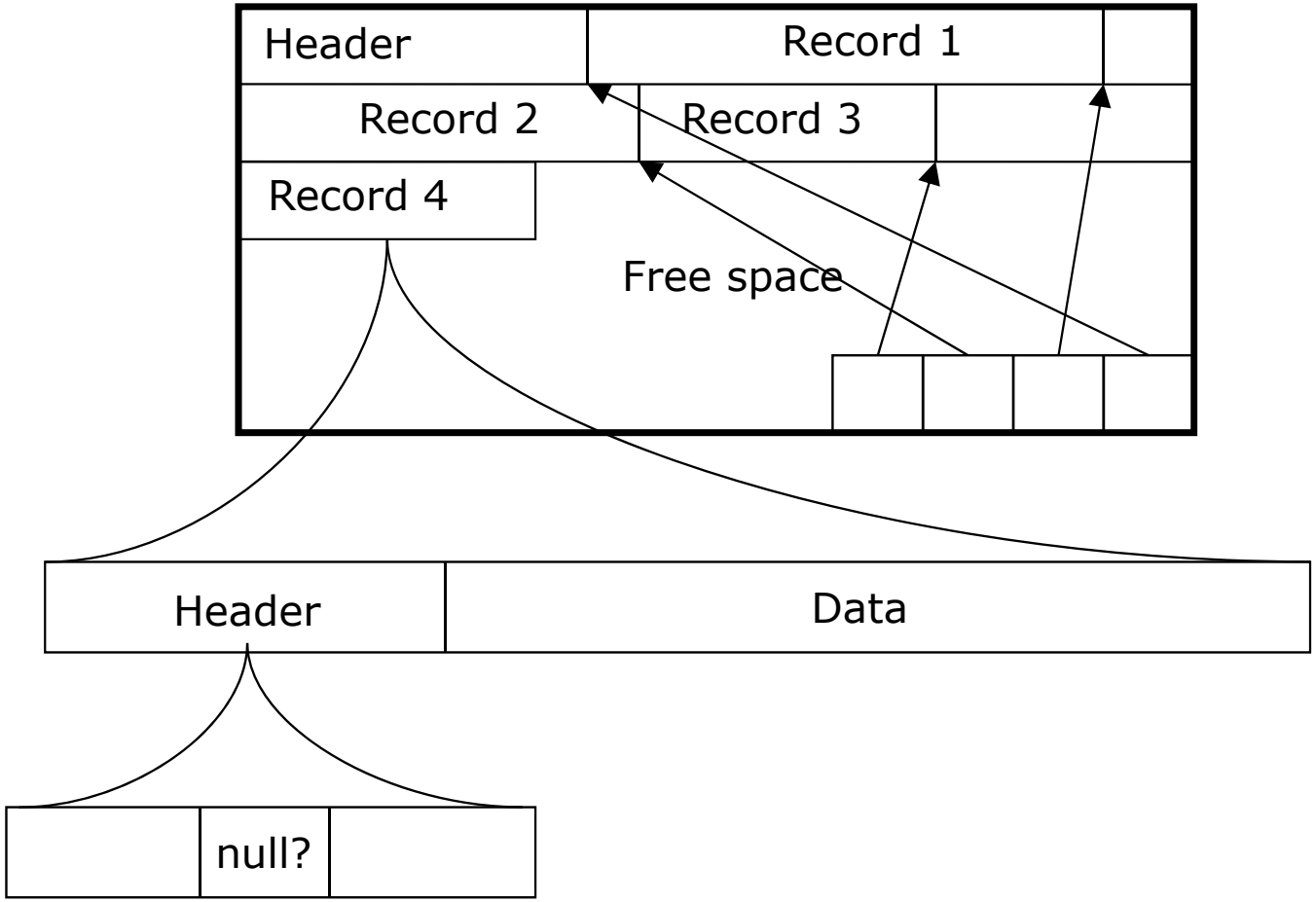
Ternary logic

```
SELECT *
FROM Employees
WHERE age=40
        OR age<>40;
```

Null values

- Two meanings
  - Unknown
  - Nonexistent
- Reasons to use them:
  - Inserting a tuple with an unknown value
  - Adding a new attribute to a non-empty relation
  - Special aggregation cases
  - Avoiding exceptions in aggregations with unknown values
- Representation:
  - Different from any non-null value

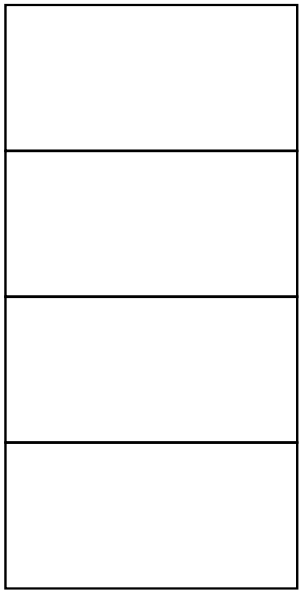
Null values storage



Null values query

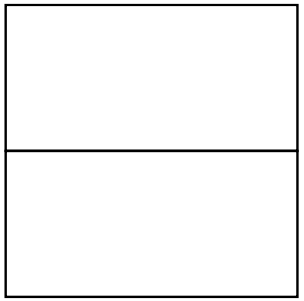
a) SELECT name
FROM People
WHERE #labour>0;

People

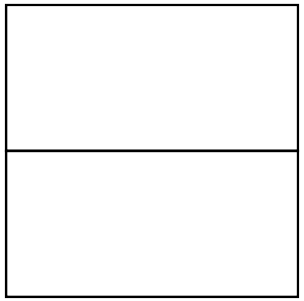


b) SELECT name
FROM Women
WHERE #labour>0;

Men



Women



Specific comparison for nulls

~~SELECT id FROM T WHERE a=NULL;~~ 

vs

SELECT id FROM T WHERE a IS NULL;

Ternary logic for null values

NULL θ X -> UNKNOWN
NULL=NULL -> UNKNOWN

NOT	
TRUE	F
UNKNOWN	U
FALSE	T

AND	T	U	F
TRUE	T	U	F
UNKNOWN	U	U	F
FALSE	F	F	F

OR	T	U	F
TRUE	T	T	T
UNKNOWN	T	U	U
FALSE	T	U	F

Interpretation of ternary logic

- Queries
  - Return rows when the predicate evaluates to true
    - Do not return those evaluating unknown

- Constraints
  - Raise an exception when the predicate evaluates to false
    - Do not raise anything when evaluating unknown

Behaviour of algebraic operations with nulls

R(	A,	B)
	?	?
	a	?
	a	1
	?	1

S(	A,	B)
	?	?
	a	?
	a	1

T(	A,	B,	C)
	?	?	z
	a	?	y
	?	1	x
	a	?	w
	?	1	v

RUS	(A,	B)
	?	?
	a	?
	a	1
	?	1

R∩S	(A,	B)
	?	?
	a	?
	a	1

R-S	(A,	B)
	?	1

T[A,B]	(A,	B)
	?	?
	a	?
	?	1

T[A]	(A)
	?
	a

R(B=B)T	(A,	B,	A',	B',	C)
	a	1	?	1	x
	?	1	?	1	x
	a	1	?	1	v
	?	1	?	1	v

"?" corresponds to the null value

Behaviour of aggregates with nulls

- COUNT
  - With "*", counts all rows
  - With a column name, counts those with non-null value for the column
- SUM
  - Adds only non-null values
  - Returns null if there are not non-null values
- AVG
  - Its result always coincides with $\frac{SUM(a)}{COUNT(a)}$
- MIN/MAX
  - Returns null if there are not non-null values

Content	COUNT(*)	COUNT(a)	SUM(a)	AVG(a)	$\frac{SUM(a)}{COUNT(a)}$	$\frac{SUM(a)}{COUNT(*)}$	MIN(a)
empty	0	0	null	null	null	null	null
null	1	0	null	null	null	null	null
null 0 1	3	2	1	0.5	0.5	0.3333	0

Behaviour in other clauses with nulls

- V IN (X_1, X_2, ...)
  - NULL IN (X_1, X_2, ...) -> UNKNOWN
    - Same behaviour as V = X_1 OR V = X_2 OR ...

- GROUP BY
  - Null values generate a single group

- UNIQUE
  - Unique constraints treat null as different from anything
    - Null is different from any other value, but also different from null
      - This makes every null a different null instance so that unique constraints accept multiple null values

Query example with nulls

Teachers living in a city where no student lives

~~SELECT id~~
~~FROM teachers~~
~~WHERE city NOT IN (SELECT city FROM students);~~ 

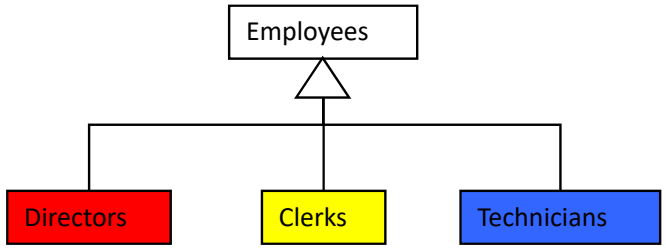
vs

```
SELECT id
FROM teachers
WHERE NOT EXISTS (    SELECT *
                      FROM students
                      WHERE teachers.city = students.city);
```

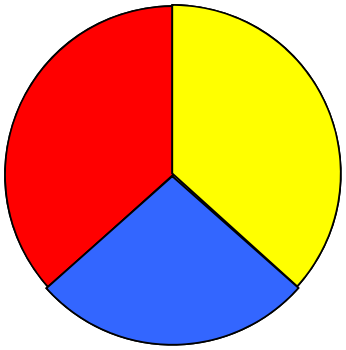
Generalization/Specialization

Kinds of Generalization/Specialization

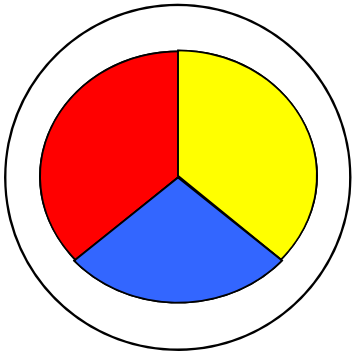
- Complete/Incomplete
- Disjoint/Overlapping



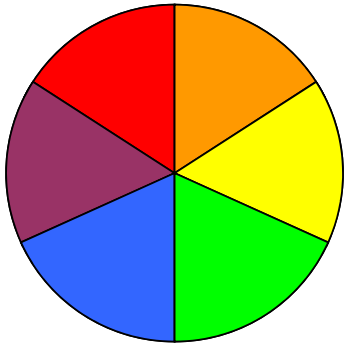
Complete Disjoint



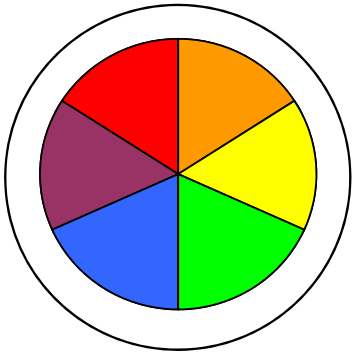
Incomplete Disjoint



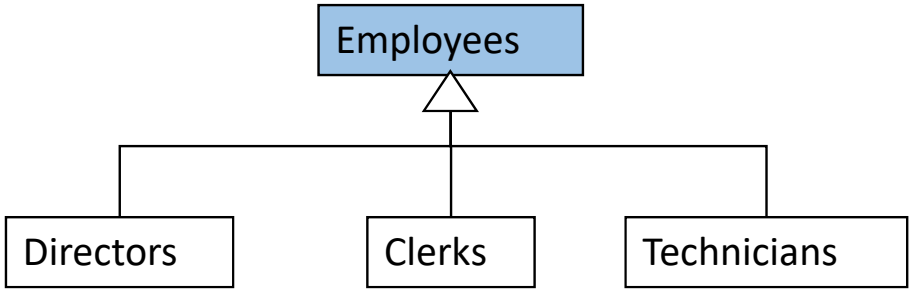
Complete Overlapping



Incomplete Overlapping



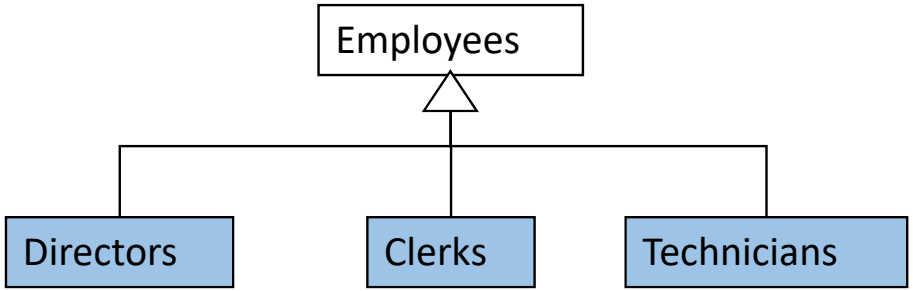
Generalization/Specialization implementation (I)



Use only one table for the superclass:

Employees (emp, employee attributes, director attributes, clerk attributes, technician attributes[, **discriminant]**)

Generalization/Specialization implementation (II)



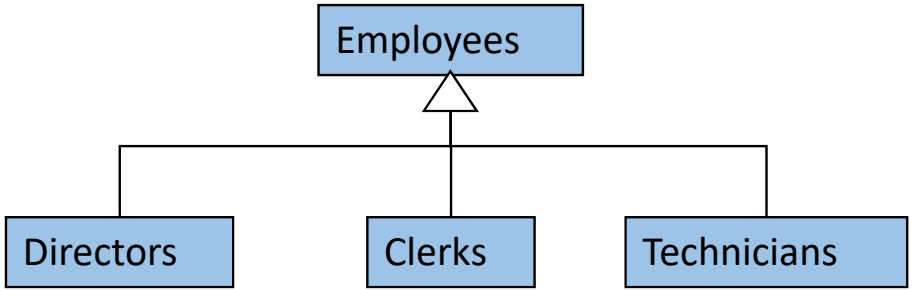
Use only one table for each of the subclasses:

Directors (emp, employee attributes, director attributes)

Clerks (emp, employee attributes, clerk attributes)

Technicians (emp, employee attributes, technician attributes)

Generalization/Specialization implementation (III)



Use one table for each class, and declare the corresponding foreign keys:

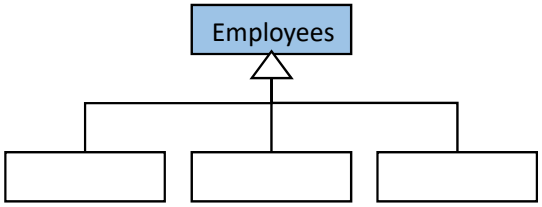
Employees (<u>emp</u>, employee attributes[, **discriminant**])

Directors (<u>emp</u>, director attributes)

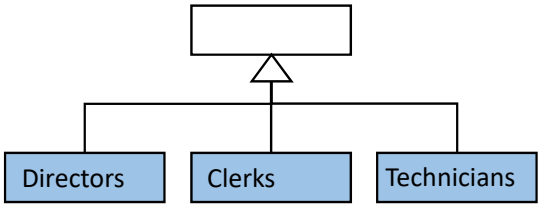
Clerks (<u>emp</u>, clerk attributes)

Technicians (<u>emp</u>, technician attributes)

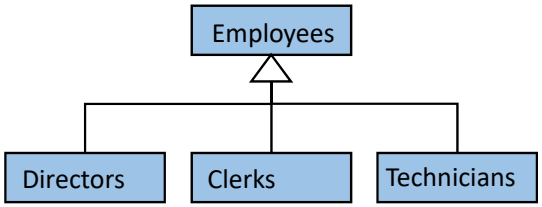
Generalization/Specialization queries




```
SELECT *
FROM employees;
```

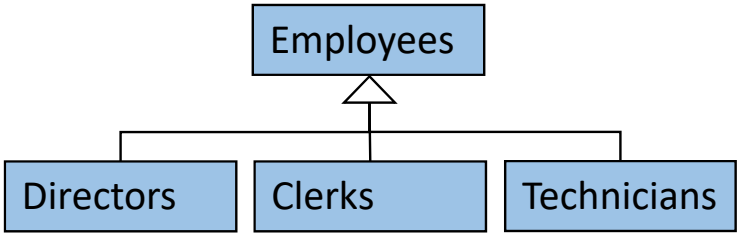



```
SELECT *
FROM directors
UNION
SELECT *
FROM clerks
UNION
SELECT *
FROM technicians;
```



~~SELECT *~~
~~FROM employees e, directors d, clerks c, technicians t~~
~~WHERE e.id=d.id AND e.id=c.id AND e.id=t.id;~~

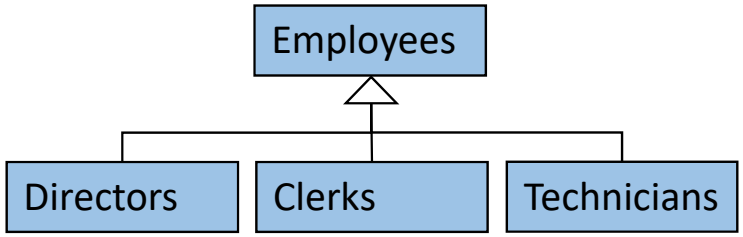
Inheritance queries




```
SELECT e.a1, …, an, c.a1, c.am
FROM employees e, clerks c
WHERE e.id=c.id
UNION
SELECT e.a1, e.an, NULL, …, NULL
FROM employees e
WHERE e.id NOT IN (SELECT c.id
                   FROM clerks c);
```

With 2 subclasses, we need three unions (i.e., four subqueries)!
With 3 subclasses, we need seven unions (i.e., eight subqueries)!
…

Left outer join




```
SELECT *
FROM employees e
    LEFT OUTER JOIN directors d ON e.id=d.id
    LEFT OUTER JOIN clerks c ON e.id=c.id
    LEFT OUTER JOIN technicians t ON e.id=t.id;
```

Left/Right/Full outer join

R(	a,	b)
	1	a
	2	b
	3	?

S(	b,	c)
	a	4
	c	5

R LeftOuterJoin S ON b=b

(	a,	b,	b',	c)
	1	a	a	4
	2	b	?	?
	3	?	?	?

R RightOuterJoin S ON b=b

(	a,	b,	b',	c)
	1	a	a	4
	?	?	c	5

R FullOuterJoin S ON b=b

(	a,	b,	b',	c)
	1	a	a	4
	2	b	?	?
	3	?	?	?
	?	?	c	5

"?" corresponds to the null value

Outer Join in SQL'99 (I)

<table1> [CROSS | INNER | [LEFT|RIGHT|FULL] OUTER] JOIN <table2> [ON <condition>]

- The order in the FROM clause is not commutative now
  - Joins are performed from left to right
- WHERE predicate is evaluated after the joins

Table inheritance in PostgreSQL

```
CREATE TABLE employees (
  name TEXT PRIMARY KEY
  );

CREATE TABLE clerks (
  workingplace TEXT
  ) INHERITS (employees);

INSERT INTO clerks VALUES ('Eva', 'Campus Nord');

SELECT * FROM clerks;
        name |workingplace |
        --------+-----------------+
        Eva    |Campus Nord |

SELECT * FROM employees;
        name  |
        ----------+
        Eva     |
```

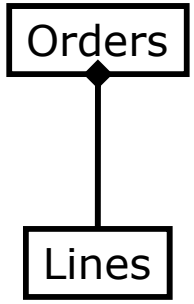
https://www.postgresql.org/docs/current/tutorial-inheritance.html

Surrogates

Surrogates

- Introduced by E. F. Codd in RM/T

- Substitutes the external key if:
  a) There is no such external key
  b) Attributes in the external key change often
  c) The external key demands too much space



Automatic surrogates in PostgreSQL

```
CREATE TABLE <tableName> (
        id SERIAL PRIMARY KEY,
        <otherAttributes> ...);

INSERT INTO <tableName>(<otherAttributes>) VALUES (<otherValues>);
```

- The value for the identifier does not need to be provided on insertion
- We do not have control over it
- Values in the table may not be consecutive if some insertions fail

https://www.postgresql.org/docs/current/datatype-numeric.html#DATATYPE-SERIAL

User-managed surrogates in PostgreSQL

```
CREATE SEQUENCE <seqName>
   INCREMENT BY <int>
   START WITH <int>
   ...;

SELECT * FROM <seqName>;

INSERT INTO <tableName> VALUES (NEXTVAL('<seqName>'), ...);

SELECT CURRVAL('<seqName>');

DROP SEQUENCE <seqName>;
```

- Multiple sequences can be used in the same table
- A sequence can be used in different tables

https://www.postgresql.org/docs/current/sql-createsequence.html

Closing

Summary

- Database design difficulties
- Relational operational DB design phases
- Semantic heterogeneity
- Null values
  - Meaning
  - Consequences
    - Space
    - Time
    - Writing queries
  - Ternary logic
- Implementation of Generalization/Specialization relationships
  - Outer join
  - Inheritance
- Surrogates

Bibliography

- Jaume Sistac et al. *Disseny de bases de dades*. Col·lecció Manuals, number 43. Editorial UOC, 2002
- J. Melton and A. Simon. *SQL 1999*. Morgan Kaufmann, 2002
- P. Gulutzan and T. Pelzer. *SQL-99 Complete, really*. R&D Books, 1999
- R. G. G. Cattell et al. *The Object Data Standard: ODMG 3.0*. Morgan Kaufmann Publishers, 2000
- E. F. Codd. *The Relational Model for Database Management*, version 2. Addison-Wesley, 1990
- Jaume Sistac et al. *Tècniques avançades de bases de dades*. EDIUOC, 2000
- T. Teorey et al. *Database modeling and design*, 4th edition. Morgan Kaufmann Publishers, 2006.
- R. Elmasri and B. Nabathe. *Fundamentals of Database Systems* , 4th edition. Addison-Wesley, 2003