FIB - Disseny de Bases de Dades

Relational translation - 1

Knowledge objectives

- Name the two meanings of the NULL value

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. Explain the general way to translate classes and associations into relational model

Application Objectives

- Translate a DB query from natural language into SQL, given a DB schema, taking into account the presence of `NULL` values.

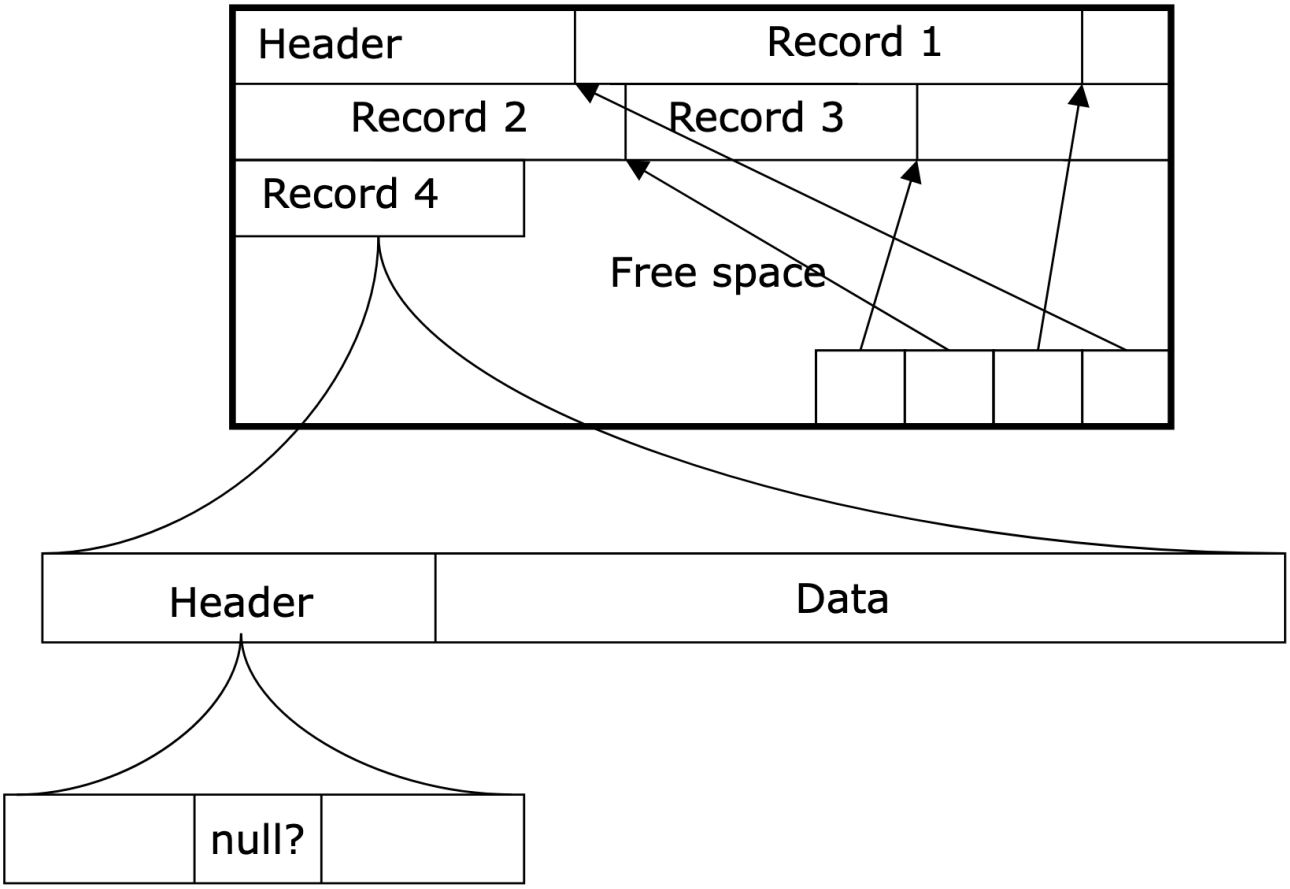
Which rows do we get?

```
SELECT  *
FROM    R
WHERE   A = 10 OR A <> 10
```

Null values

- Two meanings
- 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

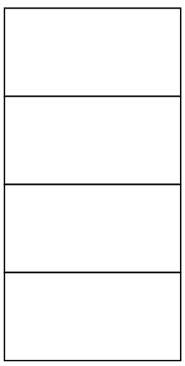
Null values storage



a

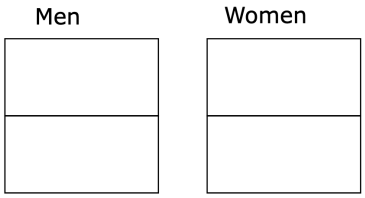
```
1 SELECT name
2 FROM People
3 WHERE #labour>0;
```

People



b

```
1 SELECT name
2 FROM Women
3 WHERE #labour>0;
```



Ternary logic for null values

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

NOT	
T	F
U	U
F	T

AND	**T**	**U**	**F**
T	T	U	F
U	U	U	F
F	F	F	F

OR	**T**	**U**	**F**
T	T	T	T
U	T	U	U
F	T	U	F

Consequences of ternary logic

- Queries
  - Return rows when the predicate is true
  - Do not return those evaluating unknown
- Constraints
  - Raise an exception when the predicate is false
  - Do not raise anything when evaluates unknown

Effect of null values in aggregates

- `COUNT`
  - With “*”, counts all tuples
  - With “a”, counts those with non-null value for the attribute
- `SUM`

    - Adds only non-null values
    - Returns null if there are not non-null values
- `MIN/MAX`
    - Returns null if there are not non-null values
- `AVG`
    - Its result always coincide with SUM(a)/COUNT(a

Content	SUM(a)	COUNT(*)	COUNT(a)	AVG(a)	SUM(a) / COUNT(a)	SUM(a) / COUNT(*)	MIN(a)
empty	null	0	0	null	null	null	null
null	null	1	0	null	null	null	null
null 0 1	1	3	2	0.5	0.5	0.3333	0

Effect of null values in usual other cases

- `V IN (X1, X2, ...)` is the same as `V = X2 OR V = X2 OR...`
    - `NULL IN (X1, X2, ...) -> UNKNOWN`
- `GROUP BY`
    - Null values are put onto a single group
- `UNIQUE`
    - Unique constraints treat null as different to everything (different to null and different to other values)
    - This makes every null a different null so that unique constraints accept multiple null values

Specific comparison for nulls

`SELECT id FROM T WHERE a IS NULL;` vs ~~`SELECT id FROM T WHERE a=NULL;`~~ .💣

Query examples with nulls

teachers living in a city where no student lives

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

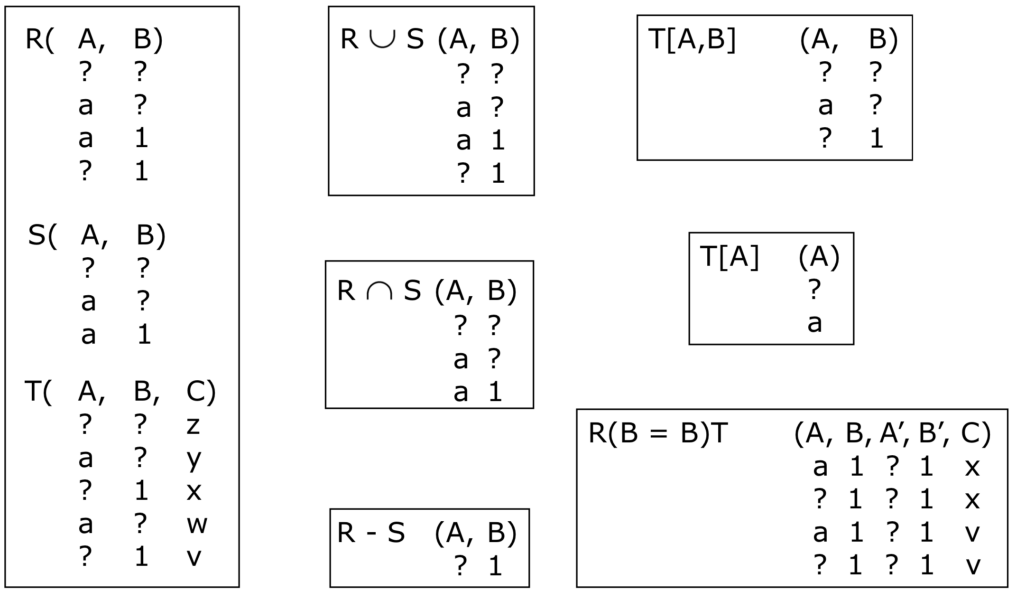
> d.city is null -> professor NO surt

vs

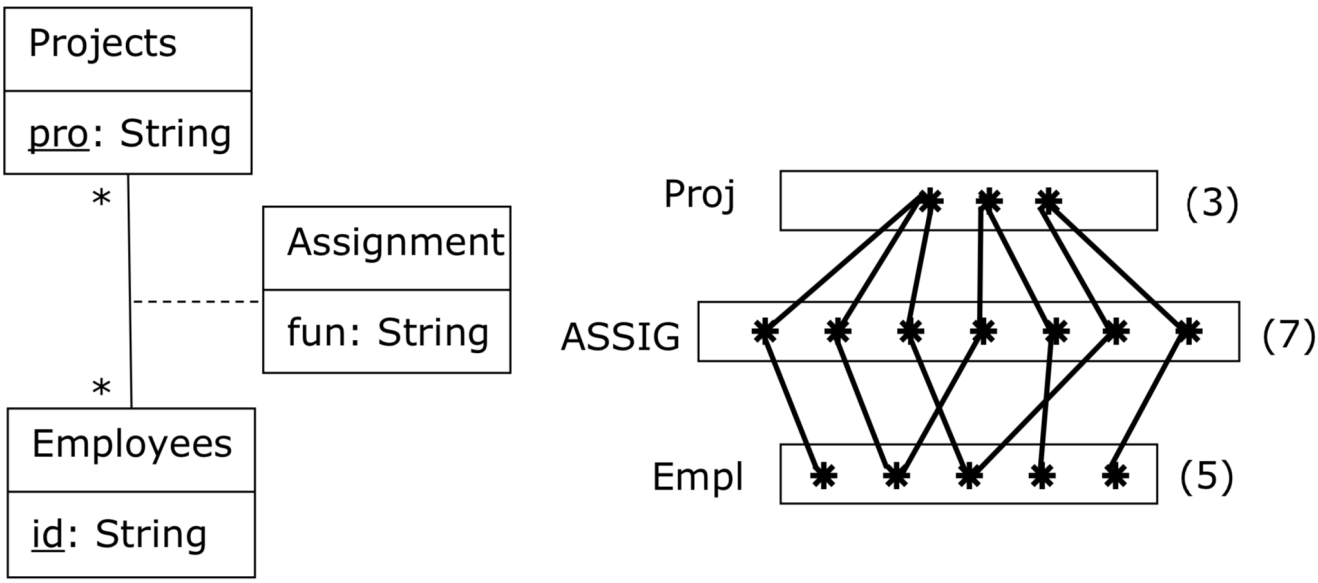
```
SELECT  id
FROM    teachers
WHERE   NOT EXISTS
(
  SELECT  *
  FROM    students
  WHERE   teachers.city = students.city
);
```

> d.city is null -> professor surt

Algebraic operations with nulls



UML model and instantiation!



Relational model representation

- Long

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

- Short

```
Projects    (pro, ...)
Assignments (empl,pro,function)
```

```
3 Employees   (id, ...)
```

WITHOUT OID!!!

Summary

Meaning of NULL values

- Consequences of NULL values
  - Space
  - Time
  - Writing queries
- General way to translate classes and associations