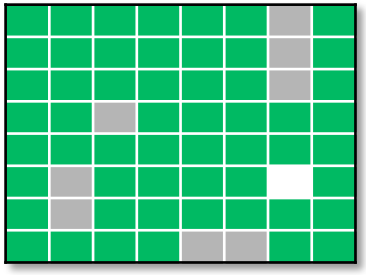


Completeness of Queries over SQL Databases

Werner Nutt and Simon Razniewski

Introduction

- Data Quality research investigates how good data is

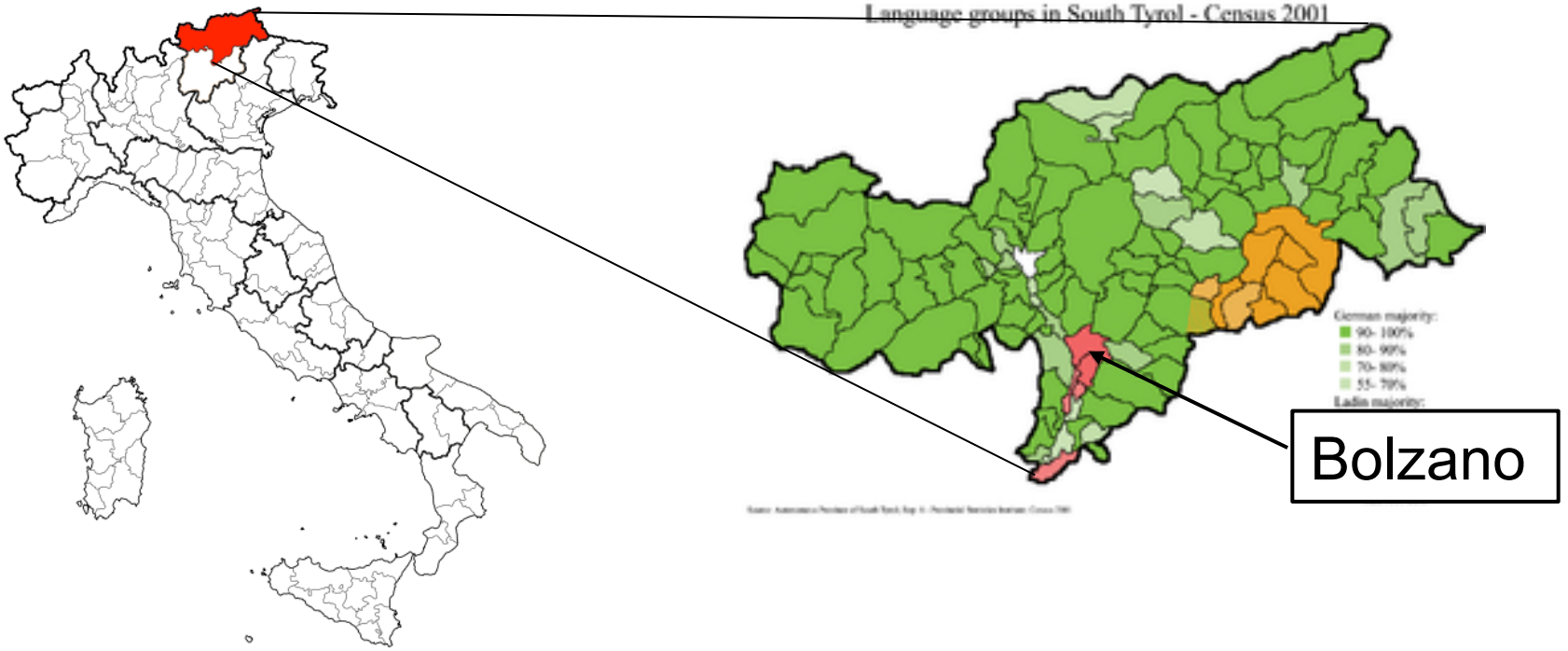
- Dimensions of Data Quality are:
  - Correctness
  - Timeliness
  - Completeness

Completeness

- Query answering over incomplete data: extensively studied
  - Codd: NULL values [1975]
  - Imielinski/Lipski: Representation systems [1984]

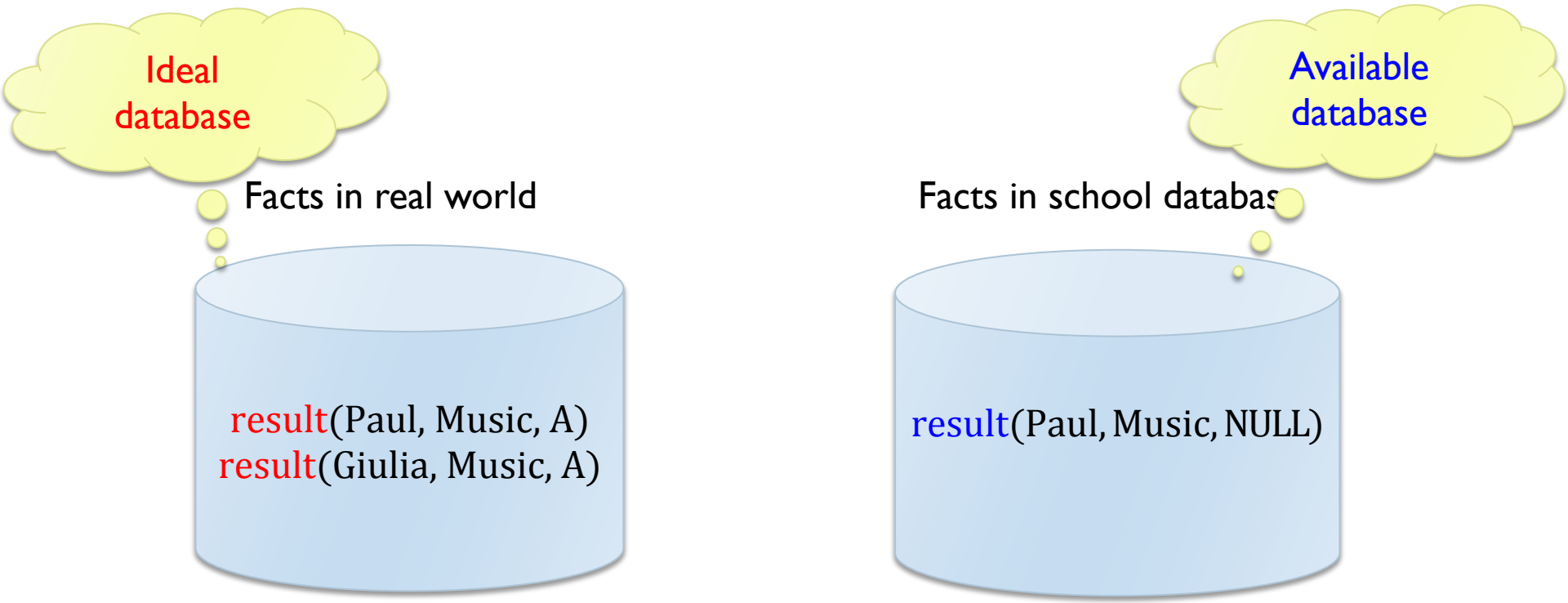
- Query completeness: Little attention
  - Razniewski/Nutt: Only on missing records [VLDB 2011]

Bolzano is in the Province of South Tyrol



- Trilingual province in the north of Italy
  - Has its own school administration

Incompleteness in the school data



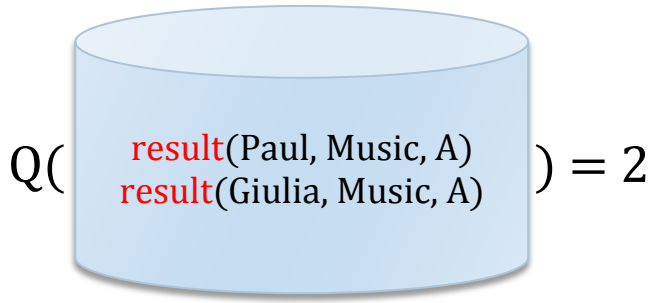
Missing information in the school database:

- no grade for Paul (missing value)
- no entry for Giulia (missing record)

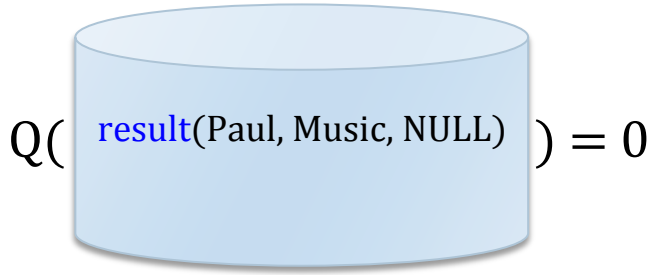
Consequence: Query answers are incorrect

Query Q: *"How many pupils have grade A in Music?"*

According to ideal database:



According to available database:



→ If data is incomplete, query answers become incorrect.

Use Metadata to guarantee completeness!

... vocational schools use
the information system of the province
to manage grades

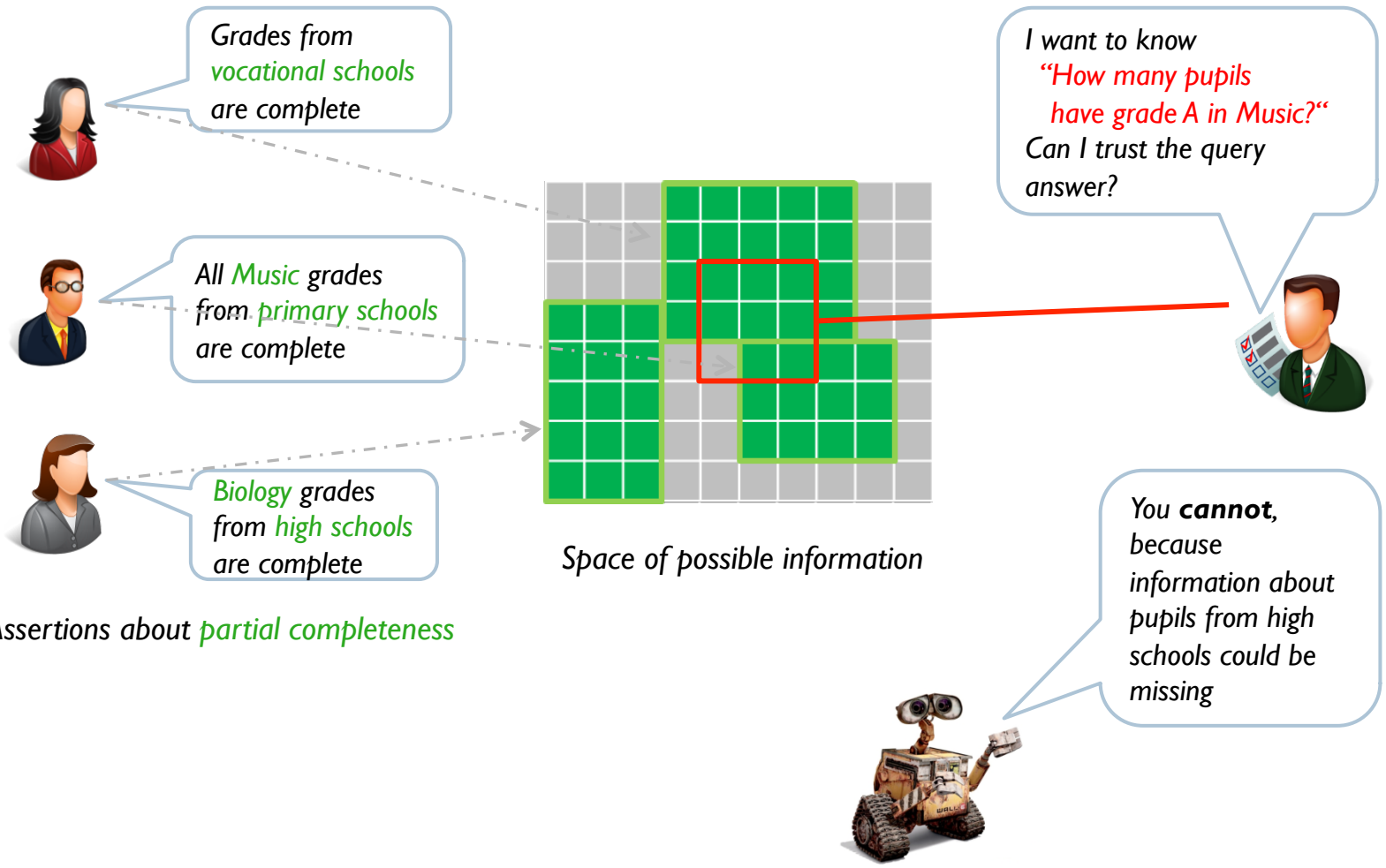
... primary schools
took part in a survey
of music education

However, we may know [illegible] parts [illegible]

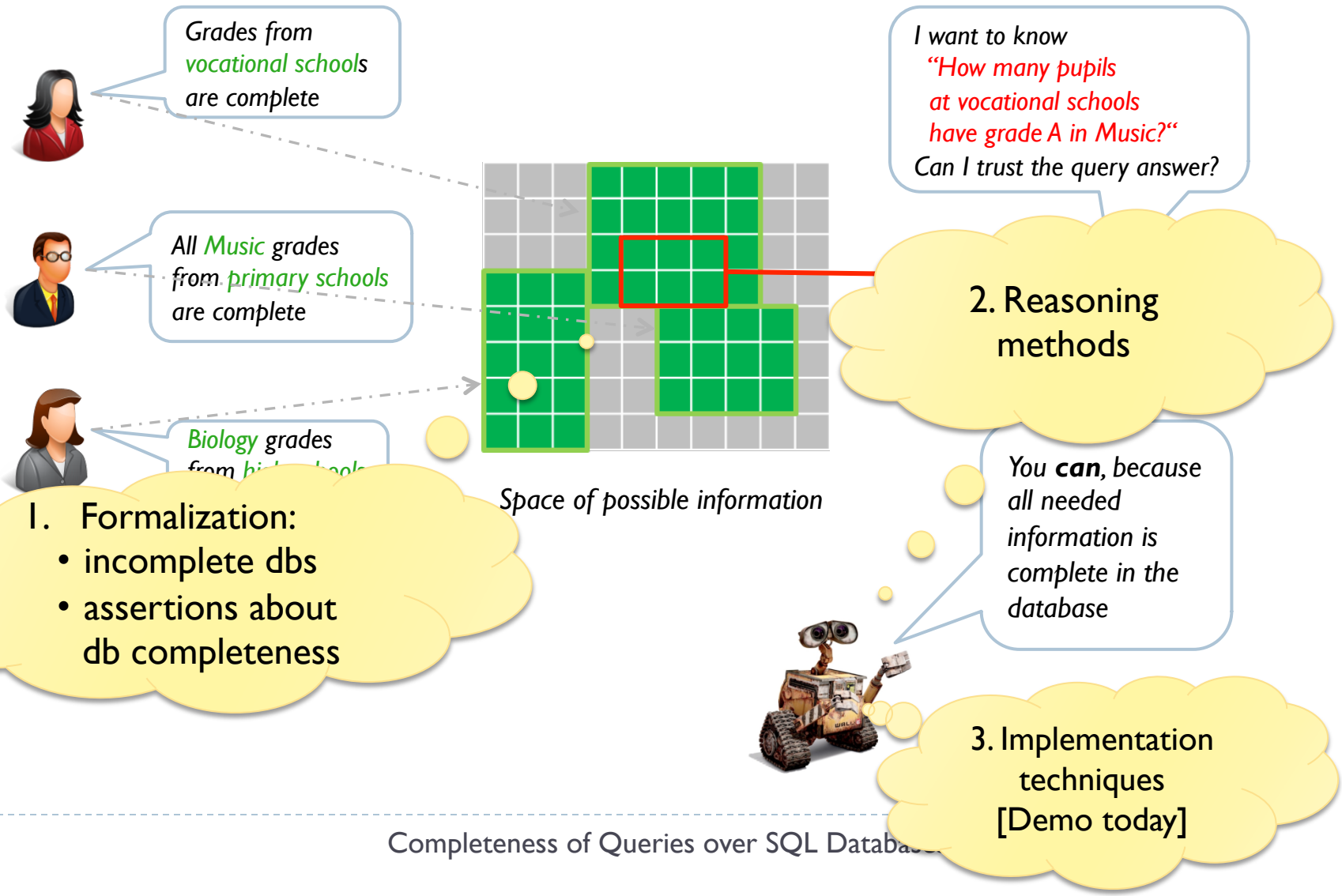
- *"The grades from vocational schools are [illegible]mplete"*
- *"The Music grades from primary schools are complete"*

➔ Idea: Assess completeness of a query
using completeness assertions for (parts of) tables

Reasoning about query completeness



Reasoning about query completeness (2)



Running example: Schema

result(name, subject, grade)

pupil(name, schoolName, schoolType)

Formalization: Incomplete database

When talking about incompleteness, we need a complete reference

An *incomplete database* D is a pair of

an ideal database D^i and

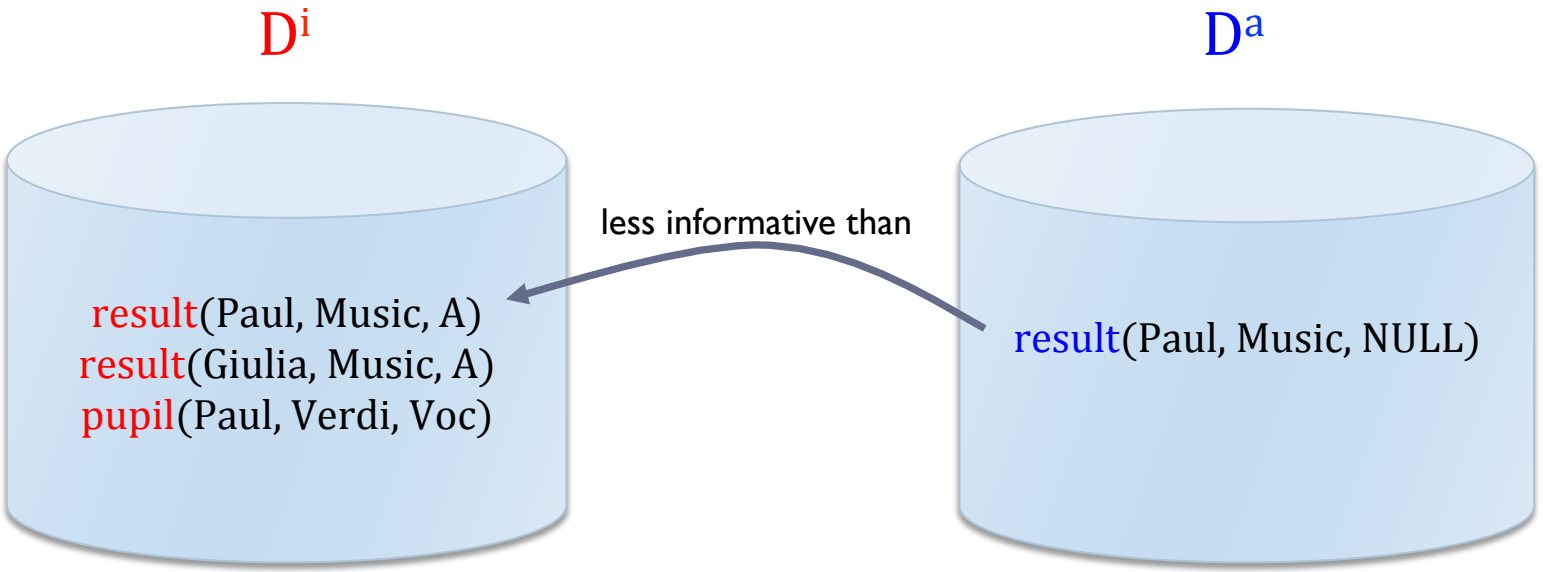
an available database D^a

$$D = (D^i, D^a)$$

such that

each record in D^a is less informative than some record in D^i

Example: Incomplete database



Formalization: Query completeness

[Motro1989]

Query Q

"The answer to Q is complete"

Notation: $\mathrm{Compl}(Q)$

Semantics:

$$(D^i, D^a) \models \mathrm{Compl}(Q) \quad \text{iff} \quad Q(D^i) = Q(D^a)$$

Formalization: DB completeness

Table completeness statement assert
partial completeness of a db table. E.g.,

"The available database contains
all subjects taken by pupils at vocational schoo

This is a tuple-generating dependency (TGD)

Formally:

$$result^i(n,s,g),\ pupil^i(n,sn,Voc) \rightarrow \exists g'\ result^a(n,s,g')$$

Every result of a pupil from a vocational school
according to the ideal db
is also in the available db
(but the grade may be missing)

Example: DB completeness

$result^i(n,s,g), pupil^i(n,sn,Voc) \rightarrow \exists g'\ result^a(n,s,g')$

holds over the incomplete database (D^i, D^a)

D^i

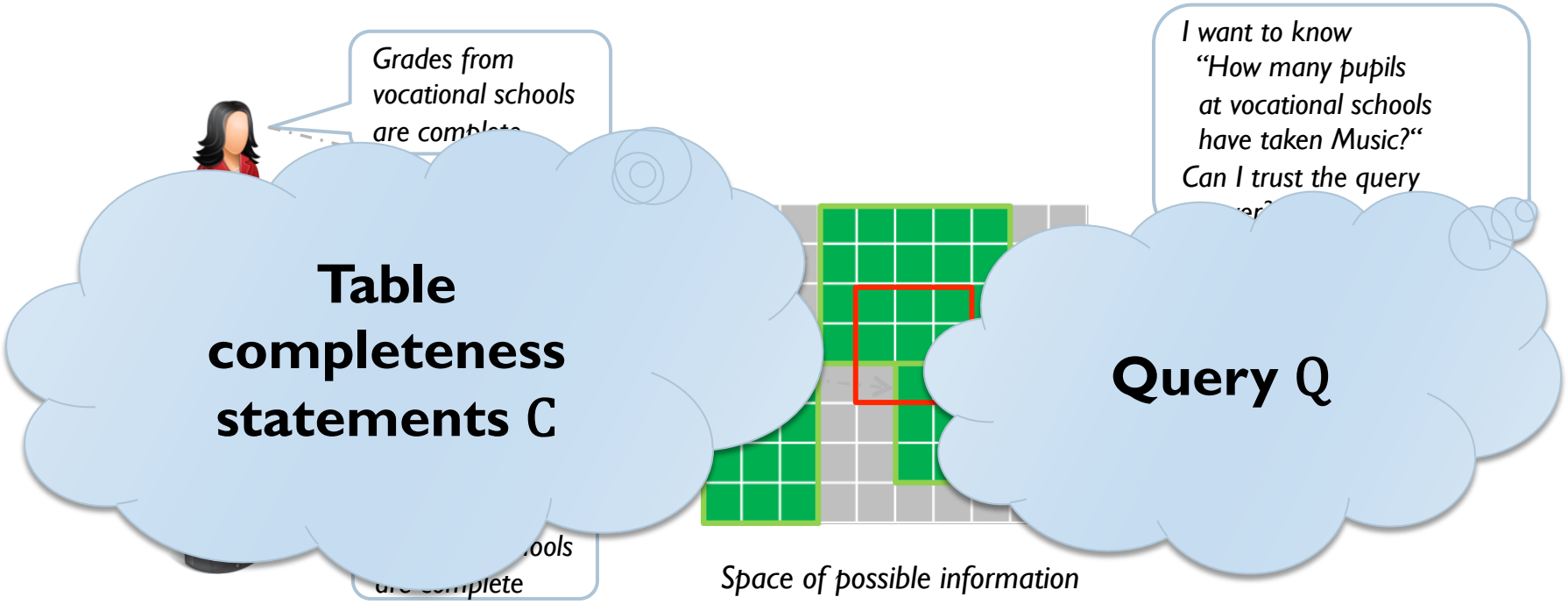
result(Paul, Music, A)
result(Giulia, Music, A)
pupil(Paul, Verdi, Voc)

D^a

result(Paul, Music, NULL)

because result(Paul, Music, NULL) is in D^a

The reasoning problem



Does C imply Compl(Q)?

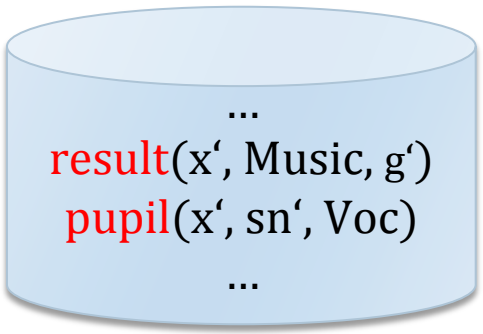
Reasoning: The principle

Query: *"Pupils at vocational schools that took Music"*

$$Q_{pupils}(x) :\text{-} result(x, Music, g), pupil(x, sn, Voc)$$

1. Assume Q_{pupils} returns x' over D^i

2. See which facts must be in D^i



Reasoning: The principle (2)

3. Use table completeness to derive facts in D^a

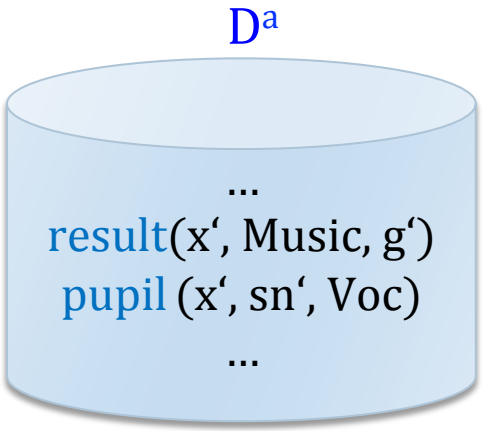
"All subjects taken by pupils at vocational schools there"

$result^i(n, s, g), pupil^i(n, sn, Voc) \rightarrow \exists g'\ result^a(n, s, g')$

"All pupils there"

$pupil^i(n, sn, st) \rightarrow pupil^a(n, sn, st)$

Reasoning: The principle (3)



4. Query the available database *"Pupils at vocational schools that took Music"*

$Q(D^a) = \{x'\}$ → x' is also in $Q(D^a)$

Conclusion: Query Q is complete given the table completeness statements

Reasoning summary

1. Assume, Q returns a generic an
2. See which facts must be in D^i
3. Use table completeness to derive facts in D^a
4. Evaluate $Q(D^a)$
5. If x‘ is returned, the query is c

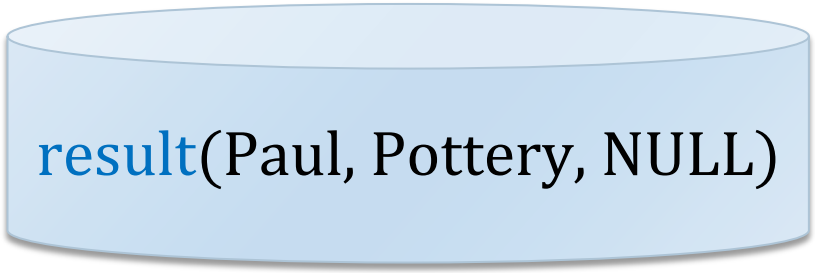
Is that unique?

How to evaluate over databases with NULLs?

Reasoning is NP-complete for DBs without NULLs

[Razniewski/Nutt VLDB 2011]

What is the Meaning of NULL?



- No grades were given in the Pottery course?

  Non-existing value

- Paul received a grade, but the grade was not recorded?

  Unknown value

- It is unknown, which of the two is the case?

  Ambiguous NULL

→ NULLs may indicate *incomplete information*, but *need not*

→ Usage of NULLs is *ambiguous*

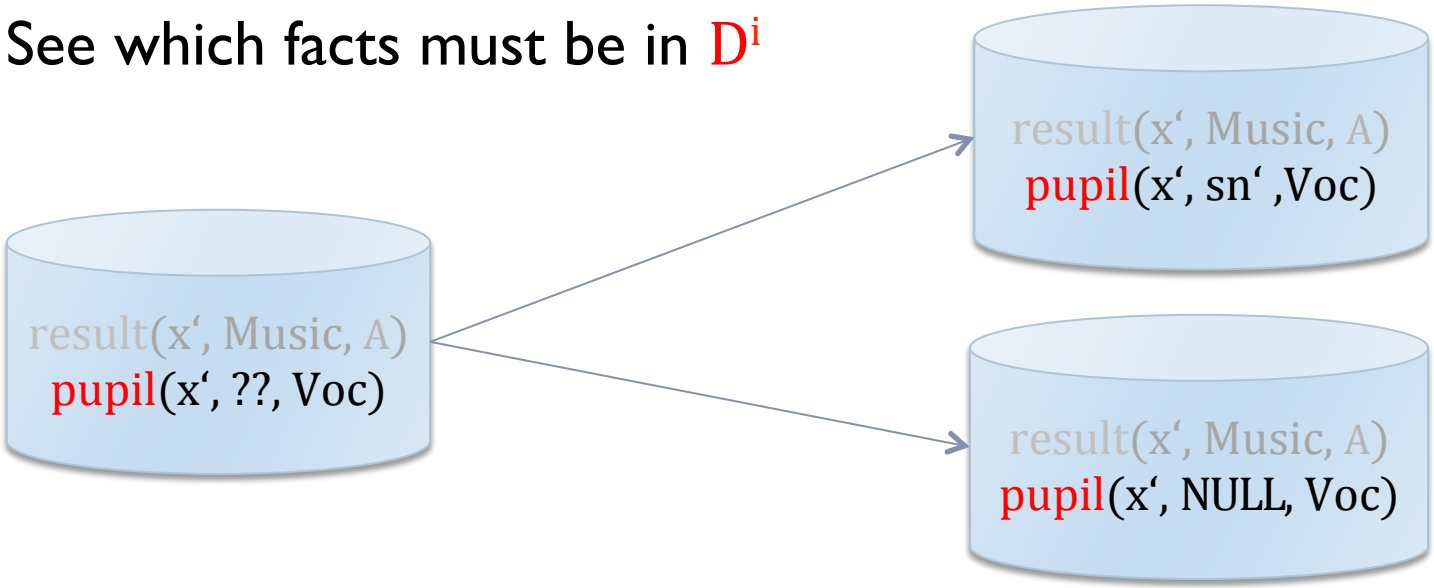
Reasoning over databases w/ NULLs

$Q_{pupils}(x)$:-result(x, Music, A), pupil(x, sn, Voc)

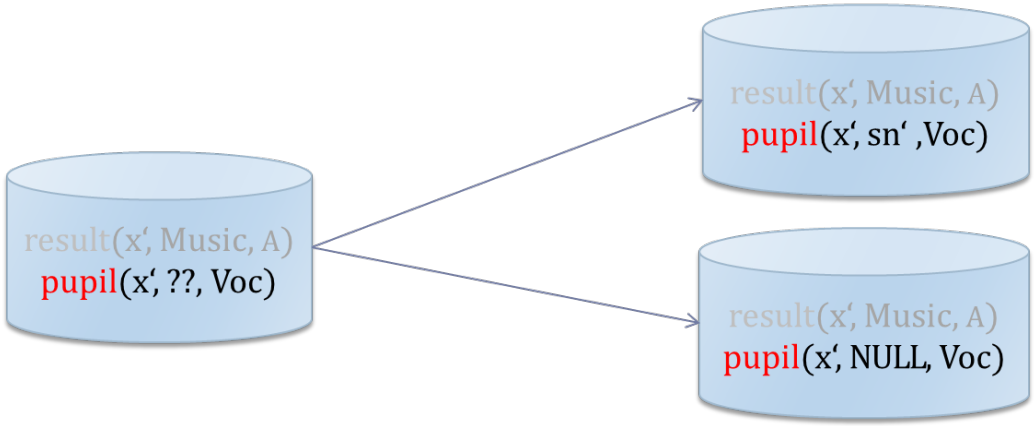
"Pupils at vocational schools with A in Music"

1. Assume Q_{pupils} returns x' over D^i

2. See which facts must be in D^i



Challenge 1: How can we adapt the reasoning to NULLs?



- In general, the reasoning has to be done for both cases
  → Reasoning is in Π^P_2

- If NULLs stand only for unknown values, then no NULLs can appear in D^i and therefore the second case cannot apply
  → Reasoning is NP-complete

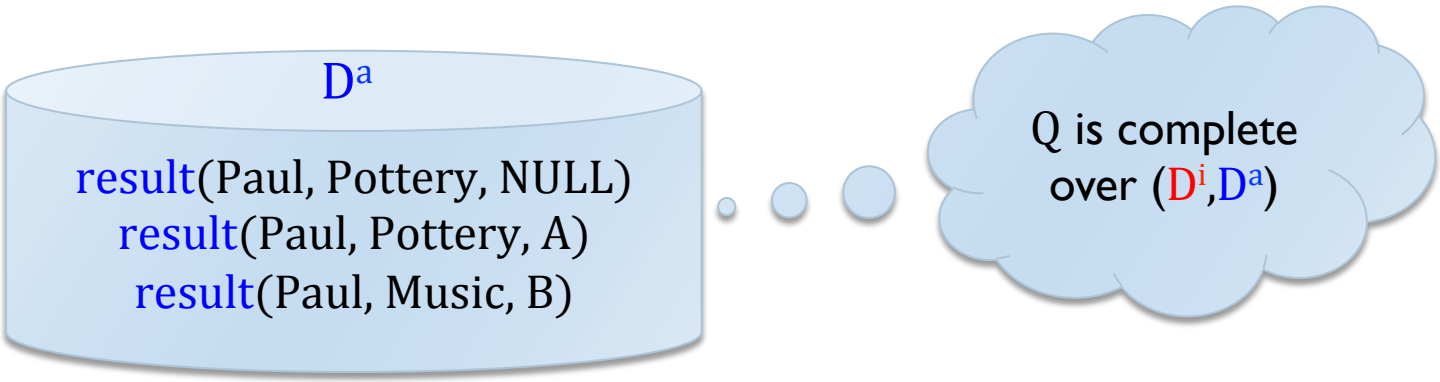
Reasoning with NULLs: Complexity

	NULLs mean unknown values	NULLs mean inapplicable values
Queries w/o selfjoins	PTIME	PTIME
Queries w/ selfjoins	NP-complete	In Π^P_2

Challenge 2: How to compute answers of complete queries?

Q(g) :- result(Paul, s, g)

"All grades of Paul"



$$Q(D^a) = \begin{cases} \{NULL,\ A,\ B\} \\ \{A,\ B\} \end{cases}$$

if NULL stands for a non-existing value

if NULL stands for an unknown value

Reasoning with both kinds of NULLs

Result: Reasoning has the same complexity
as reasoning with NULLs
standing for inapplicable values (in Π^P_2)

Result: If a query is complete,
tuples that contain unknown NULLs
can be forgotten in the query answer

Make different NULLs explicit

Ambiguity can be resolved by boolean guards

result			
name	subject	grade	grade
Paul	…	B	B
Giulia	…	NULL	NULL
Maria	…	NULL	NULL
Andrea	…	NULL	NULL

Unknown

Not applicable

Unknown whether applicable

Allows to count how many pupils received a grade (2-3)

In practice, boolean guards possibly already used where needed

Outcome

- Extended framework
  - Partial databases with NULLs
  - TC statements with projections
  - Reasoning for different meanings of NULL
- Complete queries can be evaluated by standard SQL database engines
- Complexities between PTIME and Π^P_2

Conclusion

- Query completeness assessment is practically relevant
- Reasoning over SQL databases is possible
- Demo at http://magik-demo.inf.unibz.it/

Questions?