Course overview and background

Cours 1 - BDD

Angela Bonifati

Dept. Informatique
Claude Bernard Lyon 1

Welcome to the BDD!

Today

- Introduction to deductive databases
- Review of relational data model and its query languages

Course overview

- general topic: logic-based query languages and their semantics
- Logics is everywhere in our daily life

Course overview

- general topic: logic-based query languages and their semantics
- Logics is everywhere in our daily life

- our focus: foundations of databases and query languages
- Database management systems (DBMS) as microcosm of CS

Course overview

Classical to Deductive Database Systems

- Data are stored in tables as tuples which can be seen as facts about some individuals.
  - What if we want to store the following rule "If you score more than 10 then you pass the course"?
  - Why not!! let us call it Deductive Database Systems (DDS).

Course overview

Classical to Deductive Database Systems

- Data are stored in tables as tuples which can be seen as facts about some individuals.
  - What if we want to store the following rule "If you score more than 10 then you pass the course"?
  - Why not!! let us call it Deductive Database Systems (DDS).
- We use SQL to query our data in RDBMS.
  - What about querying DDS? we need to define the interaction between **rules** and **facts**.
  - Well, that's what we call **deduction** or **inference**.

Course overview

- Key idea: add *deductive* capabilities to relational databases, the deductive database contains:
  - Facts (intensional relations), e.g. "Bill is the father of Marry".
  - Rules to generate derived facts (extensional relations), e.g. "Any person has a father and mother".

Course overview

- Key idea: add *deductive* capabilities to relational databases, the deductive database contains:
  - Facts (intensional relations), e.g. "Bill is the father of Marry".
  - Rules to generate derived facts (extensional relations), e.g. "Any person has a father and mother".
- Logic as an underlying mechanism for representing, querying and inferring knowledge from DDS.

Course overview

- Key idea: add *deductive* capabilities to relational databases, the deductive database contains:
  - Facts (intensional relations), e.g. "Bill is the father of Marry".
  - Rules to generate derived facts (extensional relations), e.g. "Any person has a father and mother".
- Logic as an underlying mechanism for representing, querying and inferring knowledge from DDS.
- Datalog as a logical language of deductive databases which is a fragment of first-order logic.
  - Similar to Prolog.
  - Has facts and rules.
  - Rules define -possibly recursive- queries (extend classical SQL querying).

Admin

Staff (CM & TD)

- Pr. Angela Bonifati (responsable)

  web: angela.bonifati[at]univ-lyon1.fr
- Pr. Mohand-Said Hacid

  web: mohand-said.hacid[at]univ-lyon1.fr
- Dr. Laureline Pinault

  web: laureline.pinault[at]ens-lyon.fr

Admin

Course website

Go to

`Spirals`

Choose 'M1-MIF14-Bases de Données Déductives'

Admin

Textbooks

Abiteboul Serge, Richard Hull, and Victor Vianu. *Foundations of Databases, 1995*.

Greco Sergio, and Cristian Molinaro. *Datalog and logic databases*, Synthesis Lectures on Data Management 7.2 (2015): 1-169.

Jeffrey D. Ullman, *Principles of Database & Knowledge-Base Systems, Vol. 1: Classical Database Systems*, Chapter 3, 1990.

Admin

Grading criteria

- projet TP
- intermediate exam (TD)
- final exam (DS)
- combination of the former (35% TP, 15% TD, 50% DS)

Outline of today's lecture

- Review of the relational model
- Review of relational algebra, relational calculus, datalog, and SQL

a quick refresher on first order logic

First order logic:

$\exists, \forall, \neg, \wedge, \vee, R(\overline{x}), x = y, \varphi \rightarrow \psi$

The query languages we will consider are fragments of first order logic (FO)

that is, the queries (i.e., computable mappings from database instances to database instances) expressible in these languages are equivalently expressible as formulas in FO

we next review some basics of FO

FO review: data

Fix some universe U of atomic values.

- A relation schema consists of a name R and finite set of attribute names $\mathit{attributes}(R) = \{A_1, \ldots, A_k\}$. The arity of R is $\mathit{arity}(R) = k$.
- A fact over relation schema R of arity k is a term of the form $R(a_1, \ldots, a_k)$, where $a_1, \ldots, a_k \in U$.
  - alternatively, a tuple over R is a function from $\mathit{attributes}(R)$ to U.
- An instance of relation schema R is a finite set of facts/tuples over R.

FO review: data

Fix some universe U of atomic values.

- A relation schema consists of a name R and finite set of attribute names $\mathit{attributes}(R) = \{A_1, \ldots, A_k\}$. The arity of R is $\mathit{arity}(R) = k$.
- A fact over relation schema R of arity k is a term of the form $R(a_1, \ldots, a_k)$, where $a_1, \ldots, a_k \in U$.
  - alternatively, a tuple over R is a function from $\mathit{attributes}(R)$ to U.
- An instance of relation schema R is a finite set of facts/tuples over R.
- A database schema (aka "signature") is a finite set D of relation schemas.
- An instance of database schema D is a set of relation instances, one for each $R \in D$.

FO review: syntax

Fix a database schema D and let $S_1, \ldots, S_m \in D$.

SQL

```
SELECT A1,...,Ak
FROM S1,...,Sm
WHERE Cond
```

where `Cond` is a well-formed selection condition over $\mathcal{A}$, and $A_1, \ldots, A_k \in \mathcal{A}$, for $\mathcal{A} = \bigcup_{S \in \{S_1, \ldots, S_m\}} \mathit{attributes}(S)$.

FO review: syntax

Fix a database schema D and let $S_1, \ldots, S_m \in D$.

SQL

```
SELECT A1,...,Ak
FROM S1,...,Sm
WHERE Cond
```

where `Cond` is a well-formed selection condition over $\mathcal{A}$, and $A_1, \ldots, A_k \in \mathcal{A}$, for $\mathcal{A} = \bigcup_{S \in \{S_1, \ldots, S_m\}} \mathit{attributes}(S)$.

Relational Algebra (RA)

$$\pi_{A_1,\ldots,A_k}(\sigma_{Cond}(S_1 \times \cdots \times S_m))$$

FO review: syntax

Datalog

$$result(A_1, \ldots, A_k) \leftarrow S_1(\overline{A^1}), \ldots, S_m(\overline{A^m}), C_1, \ldots, C_j$$

where $\overline{A^i}$ is a list of variables of length $arity(S_i)$ (for $1 \leq i \leq m$), C_i is a well-formed selection condition over $\mathcal{A}$ (for $1 \leq i \leq j$), and $A_1, \ldots, A_k \in \mathcal{A}$, for the set $\mathcal{A}$ of all variables appearing in $\overline{A^1}, \ldots, \overline{A^m}$.

FO review: syntax

author(<u>authorID</u>, name, birthdate, language)
book(<u>bookID</u>, title, authorID, publisher, language, year)
store(<u>storeID</u>, address, phone)
sells(<u>storeID</u>, <u>bookID</u>)

FO review: syntax

author(<u>authorID</u>, name, birthdate, language)
book(<u>bookID</u>, title, authorID, publisher, language, year)
store(<u>storeID</u>, address, phone)
sells(<u>storeID</u>, <u>bookID</u>)

List the titles of all books written by Italian speakers.

FO review: syntax

author(authorID, name, birthdate, language)
book(bookID, title, authorID, publisher, language, year)
store(storeID, address, phone)
sells(storeID, bookID)

In SQL

```
SELECT book.title
FROM author, book
WHERE book.authorID = author.authorID
      AND
      author.language = 'Italian'
```

FO review: syntax

author(<u>authorID</u>, name, birthdate, language)
book(<u>bookID</u>, title, authorID, publisher, language, year)
store(<u>storeID</u>, address, phone)
sells(<u>storeID</u>, <u>bookID</u>)

In RA

$$\pi_{book.title}(\sigma_C(author \times book))$$

where C is

$$book.authorID = author.authorID \wedge author.language = \text{`Italian'}$$

FO review: syntax

author(<u>authorID</u>, name, birthdate, language)
book(<u>bookID</u>, title, authorID, publisher, language, year)
store(<u>storeID</u>, address, phone)
sells(<u>storeID</u>, <u>bookID</u>)

In Datalog

$$result(T) \leftarrow author(A, N, D, LA), book(B, T, A, P, LB, Y), \\ LA = \text{`Italian'}$$

FO review: syntax

author(<u>authorID</u>, name, birthdate, language)
book(<u>bookID</u>, title, authorID, publisher, language, year)
store(<u>storeID</u>, address, phone)
sells(<u>storeID</u>, <u>bookID</u>)

List the IDs of all authors writing in Italian or born in 1985.

FO review: syntax

author(<u>authorID</u>, name, birthdate, language)
book(<u>bookID</u>, title, authorID, publisher, language, year)
store(<u>storeID</u>, address, phone)
sells(<u>storeID</u>, <u>bookID</u>)

```
SELECT A.authorID
FROM Author A
WHERE A.Language = 'Italian'
UNION
SELECT A.authorID
FROM Author A
WHERE A.Birthdate = 1985
```

FO review: syntax

author(<u>authorID</u>, name, birthdate, language)
book(<u>bookID</u>, title, authorID, publisher, language, year)
store(<u>storeID</u>, address, phone)
sells(<u>storeID</u>, <u>bookID</u>)

```
SELECT A.authorID
FROM Author A
WHERE A.Language = 'Italian'
UNION
SELECT A.authorID
FROM Author A
WHERE A.Birthdate = 1985
```

$$\pi_{authorID}(\sigma_{language=\text{`I...'}}(author)) \cup \pi_{authorID}(\sigma_{birthdate=1985}(author))$$

FO review: syntax

author(<u>authorID</u>, name, birthdate, language)
book(<u>bookID</u>, title, authorID, publisher, language, year)
store(<u>storeID</u>, address, phone)
sells(<u>storeID</u>, <u>bookID</u>)

```
SELECT A.authorID
FROM Author A
WHERE A.Language = 'Italian'
UNION
SELECT A.authorID
FROM Author A
WHERE A.Birthdate = 1985
```

$$\pi_{authorID}(\sigma_{language=\text{`I...'}}(author)) \cup \pi_{authorID}(\sigma_{birthdate=1985}(author))$$

$$\begin{aligned} result(A) &\leftarrow author(A, N, B, L), L = \text{`Italian'} \\ result(A) &\leftarrow author(A, N, B, L), B = 1985 \end{aligned}$$

FO review: syntax

author(<u>authorID</u>, name, birthdate, language)
book(<u>bookID</u>, title, authorID, publisher, language, year)
store(<u>storeID</u>, address, phone)
sells(<u>storeID</u>, <u>bookID</u>)

List the IDs of all authors writing in Italian and born in 1985.

FO review: syntax

author(<u>authorID</u>, name, birthdate, language)
book(<u>bookID</u>, title, authorID, publisher, language, year)
store(<u>storeID</u>, address, phone)
sells(<u>storeID</u>, <u>bookID</u>)

```
SELECT A.authorID
FROM Author A
WHERE A.Language = 'Italian'
INTERSECT
SELECT A.authorID
FROM Author A
WHERE A.Birthdate = 1985
```

FO review: syntax

author(<u>authorID</u>, name, birthdate, language)
book(<u>bookID</u>, title, authorID, publisher, language, year)
store(<u>storeID</u>, address, phone)
sells(<u>storeID</u>, <u>bookID</u>)

```
SELECT A.authorID
FROM Author A
WHERE A.Language = 'Italian'
INTERSECT
SELECT A.authorID
FROM Author A
WHERE A.Birthdate = 1985
```

$$\pi_{authorID}(\sigma_{language=\text{`I...'}}(author)) \cap \pi_{authorID}(\sigma_{birthdate=1985}(author))$$

FO review: syntax

author(<u>authorID</u>, name, birthdate, language)
book(<u>bookID</u>, title, authorID, publisher, language, year)
store(<u>storeID</u>, address, phone)
sells(<u>storeID</u>, <u>bookID</u>)

```
SELECT A.authorID
FROM Author A
WHERE A.Language = 'Italian'
INTERSECT
SELECT A.authorID
FROM Author A
WHERE A.Birthdate = 1985
```

$$\pi_{authorID}(\sigma_{language=\text{`I...'}}(author)) \cap \pi_{authorID}(\sigma_{birthdate=1985}(author))$$

$$\begin{aligned} I(A) &\leftarrow author(A, N, B, L), L = \text{`Italian'} \\ B(A) &\leftarrow author(A, N, B, L), B = 1985 \\ result(A) &\leftarrow I(A), B(A) \end{aligned}$$

FO review: syntax

author(<u>authorID</u>, name, birthdate, language)
book(<u>bookID</u>, title, authorID, publisher, language, year)
store(<u>storeID</u>, address, phone)
sells(<u>storeID</u>, <u>bookID</u>)

List the IDs of all authors writing in Italian not born in 1985.

FO review: syntax

author(<u>authorID</u>, name, birthdate, language)
book(<u>bookID</u>, title, authorID, publisher, language, year)
store(<u>storeID</u>, address, phone)
sells(<u>storeID</u>, <u>bookID</u>)

```
SELECT A.authorID
FROM Author A
WHERE A.Language = 'Italian'
EXCEPT
SELECT A.authorID
FROM Author A
WHERE A.Birthdate = 1985
```

FO review: syntax

author(<u>authorID</u>, name, birthdate, language)
book(<u>bookID</u>, title, authorID, publisher, language, year)
store(<u>storeID</u>, address, phone)
sells(<u>storeID</u>, <u>bookID</u>)

```
SELECT A.authorID
FROM Author A
WHERE A.Language = 'Italian'
EXCEPT
SELECT A.authorID
FROM Author A
WHERE A.Birthdate = 1985
```

$$\pi_{authorID}(\sigma_{language=\text{`I...'}}(author)) - \pi_{authorID}(\sigma_{birthdate=1985}(author))$$

FO review: syntax

author(<u>authorID</u>, name, birthdate, language)
book(<u>bookID</u>, title, authorID, publisher, language, year)
store(<u>storeID</u>, address, phone)
sells(<u>storeID</u>, <u>bookID</u>)

```
SELECT A.authorID
FROM Author A
WHERE A.Language = 'Italian'
EXCEPT
SELECT A.authorID
FROM Author A
WHERE A.Birthdate = 1985
```

$$\pi_{authorID}(\sigma_{language=\text{`I...'}}(author)) - \pi_{authorID}(\sigma_{birthdate=1985}(author))$$

$$\begin{aligned} I(A) &\leftarrow author(A, N, B, L), L = \text{`Italian'} \\ B(A) &\leftarrow author(A, N, B, L), B = 1985 \\ result(A) &\leftarrow I(A), not\ B(A) \end{aligned}$$

FO review: syntax

author(<u>authorID</u>, name, birthdate, language)
book(<u>bookID</u>, title, authorID, publisher, language, year)
store(<u>storeID</u>, address, phone)
sells(<u>storeID</u>, <u>bookID</u>)

List the IDs of all books which are sold in every store.

FO review: syntax

author(<u>authorID</u>, name, birthdate, language)
book(<u>bookID</u>, title, authorID, publisher, language, year)
store(<u>storeID</u>, address, phone)
sells(<u>storeID</u>, <u>bookID</u>)

In RA

$$\pi_{bookID}(book) - \pi_{bookID}\left((\pi_{storeID}(store) \times \pi_{bookID}(book)) - sells\right)$$

FO review: syntax

author(<u>authorID</u>, name, birthdate, language)
book(<u>bookID</u>, title, authorID, publisher, language, year)
store(<u>storeID</u>, address, phone)
sells(<u>storeID</u>, <u>bookID</u>)

In SQL

```
SELECT B.bookID
FROM book B
WHERE NOT EXISTS
  (SELECT bookID, storeID
   FROM book, store
   WHERE bookID=B.bookID
   EXCEPT
   sells)
```

FO review: syntax

author(<u>authorID</u>, name, birthdate, language)
book(<u>bookID</u>, title, authorID, publisher, language, year)
store(<u>storeID</u>, address, phone)
sells(<u>storeID</u>, <u>bookID</u>)

In Datalog

$$
\begin{aligned}
all(S, B) &\leftarrow book(B, T, A, P, L, Y), store(S, Ad, Ph) \\
missing(B) &\leftarrow all(S, B), not\ sells(S, B) \\
result(B) &\leftarrow book(B, T, A, P, L, Y), not\ missing(B)
\end{aligned}
$$

FO review: syntax

author(<u>authorID</u>, name, birthdate, language)
book(<u>bookID</u>, title, authorID, publisher, language, year)
store(<u>storeID</u>, address, phone)
sells(<u>storeID</u>, <u>bookID</u>)

in TRC, the Tuple Relational Calculus (i.e., essentially straight FO)

$$\{t \mid \exists b \in book(t.bookID = b.bookID \wedge \forall s \in store \exists \ell \in sell(\ell.storeID = s.storeID \wedge \ell.bookID = b.bookID))\}$$

FO review: query evaluation

Fact. SQL (without aggregation and other bells-and-whistles), RA, TRC, and (safe non-recursive) Datalog are all equivalent in expressive power.

FO review: query evaluation

Fact. SQL (without aggregation and other bells-and-whistles), RA, TRC, and (safe non-recursive) Datalog are all equivalent in expressive power.

Let

- *FO* denote the full TRC
  - i.e., any of the above languages

FO review: query evaluation

Fact. SQL (without aggregation and other bells-and-whistles), RA, TRC, and (safe non-recursive) Datalog are all equivalent in expressive power.

Let

- *FO* denote the full TRC
  - i.e., any of the above languages
- *Conj* denote the TRC using only $\exists$ and $\wedge$
  - i.e., the "conjunctive" queries
  - corresponds in SQL to basic SELECT-FROM-WHERE blocks
  - corresponds to the $\{\sigma, \pi, \times\}$ fragment of RA
  - corresponds to single positive datalog rules

FO review: query evaluation

Fact. SQL (without aggregation and other bells-and-whistles), RA, TRC, and (safe non-recursive) Datalog are all equivalent in expressive power.

Let

- *FO* denote the full TRC
  - i.e., any of the above languages
- *Conj* denote the TRC using only $\exists$ and $\wedge$
  - i.e., the "conjunctive" queries
  - corresponds in SQL to basic SELECT-FROM-WHERE blocks
  - corresponds to the $\{\sigma, \pi, \times\}$ fragment of RA
  - corresponds to single positive datalog rules
- *AConj* denote the "acyclic" conjunctive queries
  - queries with join trees

Books schema

author(<u>authorID</u>, name, birthdate, language)
book(<u>bookID</u>, title, authorID, publisher, language, year)
store(<u>storeID</u>, address, phone)
sells(<u>storeID</u>, <u>bookID</u>)

Books schema

author(<u>authorID</u>, name, birthdate, language)
book(<u>bookID</u>, title, authorID, publisher, language, year)
store(<u>storeID</u>, address, phone)
sells(<u>storeID</u>, <u>bookID</u>)

1. List all author names and the books they have written, i.e., all pairs of (name, bookID).

Books schema

author(<u>authorID</u>, name, birthdate, language)
book(<u>bookID</u>, title, authorID, publisher, language, year)
store(<u>storeID</u>, address, phone)
sells(<u>storeID</u>, <u>bookID</u>)

1. List all author names and the books they have written, i.e., all pairs of (name, bookID).
2. List all books authored by a native speaker.

Books schema

author(<u>authorID</u>, name, birthdate, language)
book(<u>bookID</u>, title, authorID, publisher, language, year)
store(<u>storeID</u>, address, phone)
sells(<u>storeID</u>, <u>bookID</u>)

1. List all author names and the books they have written, i.e., all pairs of (name, bookID).
2. List all books authored by a native speaker.
3. List all authors who only have books appearing in their native language.

Books schema

author(<u>authorID</u>, name, birthdate, language)
book(<u>bookID</u>, title, authorID, publisher, language, year)
store(<u>storeID</u>, address, phone)
sells(<u>storeID</u>, <u>bookID</u>)

1. List all author names and the books they have written, i.e., all pairs of (name, bookID).
2. List all books authored by a native speaker.
3. List all authors who only have books appearing in their native language.
4. List all books which are not available in stores.

Books schema

author(<u>authorID</u>, name, birthdate, language)
book(<u>bookID</u>, title, authorID, publisher, language, year)
store(<u>storeID</u>, address, phone)
sells(<u>storeID</u>, <u>bookID</u>)

1. List all author names and the books they have written, i.e., all pairs of (name, bookID).
2. List all books authored by a native speaker.
3. List all authors who only have books appearing in their native language.
4. List all books which are not available in stores.
5. List all books sold in more than one store.

Books schema

author(<u>authorID</u>, name, birthdate, language)
book(<u>bookID</u>, title, authorID, publisher, language, year)
store(<u>storeID</u>, address, phone)
sells(<u>storeID</u>, <u>bookID</u>)

1. List all author names and the books they have written, i.e., all pairs of (name, bookID).
2. List all books authored by a native speaker.
3. List all authors who only have books appearing in their native language.
4. List all books which are not available in stores.
5. List all books sold in more than one store.
6. List all books sold in exactly one store.

Books schema

author(<u>authorID</u>, name, birthdate, language)
book(<u>bookID</u>, title, authorID, publisher, language, year)
store(<u>storeID</u>, address, phone)
sells(<u>storeID</u>, <u>bookID</u>)

1. List all author names and the books they have written, i.e., all pairs of (name, bookID).
2. List all books authored by a native speaker.
3. List all authors who only have books appearing in their native language.
4. List all books which are not available in stores.
5. List all books sold in more than one store.
6. List all books sold in exactly one store.
7. List all (bookId, storeId) pairs where the given book is not for sale at the given store.

Books schema

author(authorID, name, birthdate, language)
book(bookID, title, authorID, publisher, language, year)
store(storeID, address, phone)
sells(storeID, bookID)

1. List all author names and the books they have written, i.e., all pairs of (name, bookID).
2. List all books authored by a native speaker.
3. List all authors who only have books appearing in their native language.
4. List all books which are not available in stores.
5. List all books sold in more than one store.
6. List all books sold in exactly one store.
7. List all (bookId, storeId) pairs where the given book is not for sale at the given store.
8. List all books which are sold in every store.