Course overview and background

Lecture 1
2ID35, Spring 2015

George Fletcher

Faculteit Wiskunde & Informatica
Technische Universiteit Eindhoven

22 April 2015

Welcome to the Database Technology Course!

Today

- Introduction to database systems
- Syllabus
- Review of relational data model and its query languages

Course overview

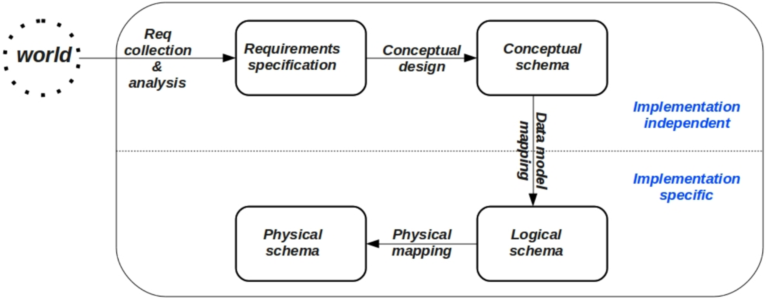
- general topic: Big data; engineering for of data science
- Data is everywhere, outlasting code (which is also data ...)

Course overview

- general topic: Big data; engineering for of data science
- Data is everywhere, outlasting code (which is also data ...)

- our focus: core technologies behind principled management of massive collections of data
- Database management systems (DBMS) as microcosm of CS

Course overview



DB design process

Course overview

"Goodness" in DB design

- Conceptual. Accurately reflect the semantics of use in the modeled domain.

Course overview

"Goodness" in DB design

- Conceptual. Accurately reflect the semantics of use in the modeled domain.
- Logical. Accurately reflect conceptual model and disallow redundancies and update anomalies, as best possible.

Course overview

“Goodness” in DB design

- Conceptual. Accurately reflect the semantics of use in the modeled domain.
- Logical. Accurately reflect conceptual model and disallow redundancies and update anomalies, as best possible.
- Physical. Accurately reflect logical model and efficiently and reliably support use of data.
  - *Our focus in this course*

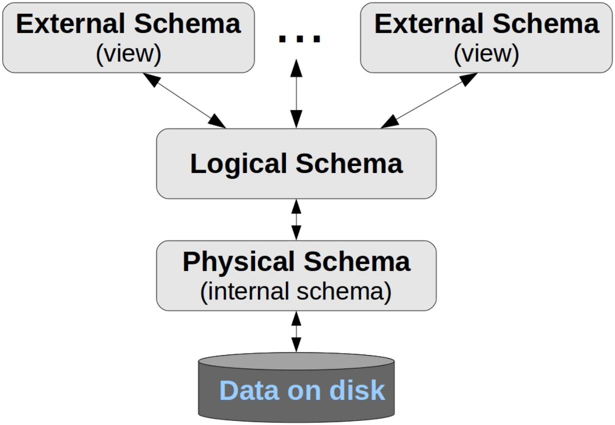
Course overview

- Why not filesystem and/or virtual memory (i.e., use the OS)?

Course overview

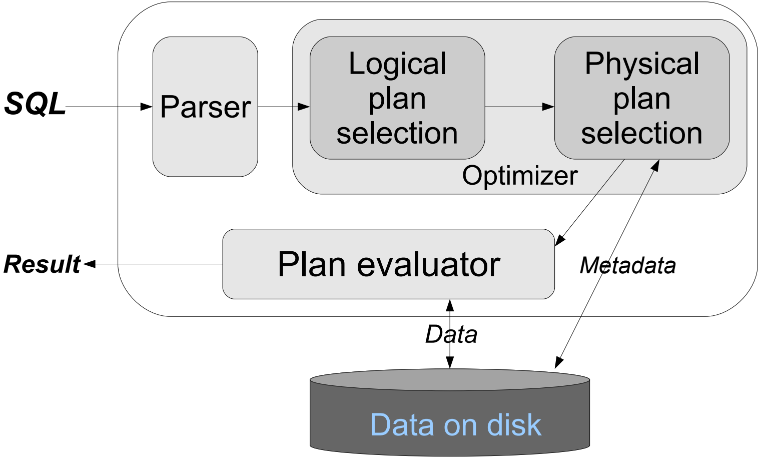
- Why not filesystem and/or virtual memory (i.e., use the OS)?
- Key idea: data independence
  - insulation from changes in the way data is structured and stored
- physical vs. logical data independence

Course overview



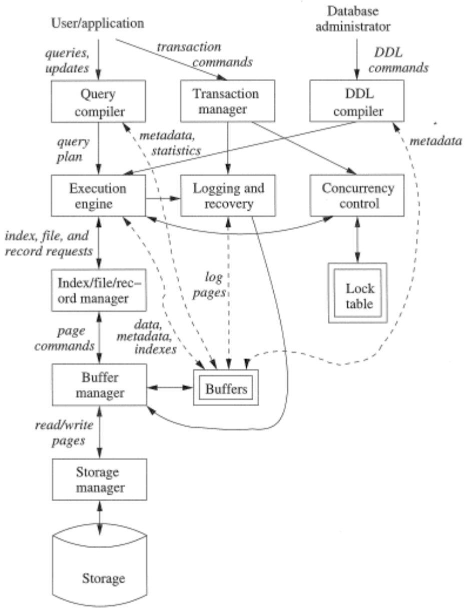
Layers of Abstraction

Course overview



The life of a query

Course overview



Admin: staff

Staff

- dr. George Fletcher
  web: `www.win.tue.nl/~gfletche/`
  office: MF7.063

Admin: website

Course website

Google "`George Fletcher TUE teaching`"

or, go to

`http://wwwis.win.tue.nl/2ID35/`

Admin: textbook

Textbook

A. Silberschatz, H. Korth, and S. Sudarshan. *Database System Concepts*, 6th Edition, McGraw-Hill, 2011.

Admin: responsibilities

Student responsibilities

- One multi-phase team project,
- One written individual assignment, and
- One final individual exam.

Admin: responsibilities

Student responsibilities

- One multi-phase team project,
- One written individual assignment, and
- One final individual exam.

Grading criteria

- 50% project
- 10% written assignment(s)
- 40% final exam

Admin: responsibilities

Student responsibilities

- One multi-phase team project,
- One written individual assignment, and
- One final individual exam.

Grading criteria

- 50% project
- 10% written assignment(s)
- 40% final exam

Note: Late work will lose $\frac{1}{3}$ of its original point-value for each day it is late, and once solutions are posted or discussed, late submissions will not be accepted. Full details and other policies can be found found in the syllabus, on the course webpage.

Admin: Course project

Carefully study a recent DB research paper

- As a team, implement main idea and repeat experiments from the paper
- Goal: validation (or refutation) and extension (and/or correction) of claimed results

Admin: Course project

Carefully study a recent DB research paper

- As a team, implement main idea and repeat experiments from the paper
- Goal: validation (or refutation) and extension (and/or correction) of claimed results
- Your first assignment: go the the course webpage, and read details (first item due before the end of this Sunday, 26 April)

Admin: Syllabus

- (22 April) Course introduction and background review (relational model, data independence).
- (24 April) Storage, the I/O computational model, & external sorting
  - Sunday 26 April: project part 1 due (paper selection)

Admin: Syllabus

- (22 April) Course introduction and background review (relational model, data independence).
- (24 April) Storage, the I/O computational model, & external sorting
  - Sunday 26 April: project part 1 due (paper selection)
- (29 April) Indexing: B-trees, R-trees, and GiST
- (1 May) Indexing: hashing and evaluation of indexes

Admin: Syllabus

- (22 April) Course introduction and background review (relational model, data independence).
- (24 April) Storage, the I/O computational model, & external sorting
  - Sunday 26 April: project part 1 due (paper selection)

- (29 April) Indexing: B-trees, R-trees, and GiST
- (1 May) Indexing: hashing and evaluation of indexes

- (6 May) Query processing
- (8 May) Data statistics & Answering queries using views
  - Sunday 10 May: project part 2 due (first report)

Admin: Syllabus

- (13 May) Query optimization
  - written assignment posted
- (15 May) *Hemelvaart – no lecture*

Admin: Syllabus

- (13 May) Query optimization
  - written assignment posted
- (15 May) *Hemelvaart – no lecture*

- (20 May) *no lecture – work on team project*
- (22 May) Distributed query processing
  - Sunday 24 May: project part 3 due (second report)

Admin: Syllabus

- (13 May) Query optimization
  - written assignment posted
- (15 May) *Hemelvaart – no lecture*

- (20 May) *no lecture – work on team project*
- (22 May) Distributed query processing
  - Sunday 24 May: project part 3 due (second report)

- (27 May) *no lecture – team project meetings with instructor*
- (29 May) *no lecture – team project meetings with instructor*

Admin: Syllabus

- (3 June) Transaction management
- (5 June) Graph and tree (XML) query processing

Admin: Syllabus

- (3 June) Transaction management
- (5 June) Graph and tree (XML) query processing

- (10 June) Course summary and exam review
- (12 June) Project presentations I
  - in class: written assignment due

Admin: Syllabus

- (3 June) Transaction management
- (5 June) Graph and tree (XML) query processing

- (10 June) Course summary and exam review
- (12 June) Project presentations I
  - in class: written assignment due

- (17 June) Project presentations II
- (19 June) Project presentations III
  - Sunday 21 June: project part 4 (final submission) due

Admin: Syllabus

- (3 June) Transaction management
- (5 June) Graph and tree (XML) query processing

- (10 June) Course summary and exam review
- (12 June) Project presentations I
  - in class: written assignment due

- (17 June) Project presentations II
- (19 June) Project presentations III
  - Sunday 21 June: project part 4 (final submission) due

- (26 June, 13:30-16:30) Final exam

After the break ...

Quick tutorial on what we are actually implementing this quarter:

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

But first .... a 5 minute paper

But first .... a 5 minute paper

What are the top 2 things you hope to take away from this course?

a quick refresher on first order logic

First order logic: $\exists, \forall, \neg, \wedge, \vee, R(\bar{x}), x = y, \varphi \rightarrow \psi$

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 $\mathit{arity}(R) = k$.

FO review: data

Fix some universe U of atomic values.

- A relation schema consists of a name R and finite set of attribute names $attributes(R) = \{A_1, \ldots, A_k\}$. The arity of R is $arity(R) = k$.
- 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 $attributes(R)$ to U.

FO review: data

Fix some universe U of atomic values.

- A relation schema consists of a name R and finite set of attribute names $attributes(R) = \{A_1, \ldots, A_k\}$. The arity of R is $arity(R) = k$.
- 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 $attributes(R)$ 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 $attributes(R) = \{A_1, \ldots, A_k\}$. The arity of R is $arity(R) = k$.
- 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 $attributes(R)$ 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.

FO review: data

Fix some universe U of atomic values.

- A relation schema consists of a name R and finite set of attribute names $attributes(R) = \{A_1, \ldots, A_k\}$. The arity of R is $arity(R) = k$.
- 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 $attributes(R)$ 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 $R_1, \ldots, R_m \in D$.

SQL

```
SELECT A1, ..., Ak
FROM R1, ..., Rm
WHERE Cond
```

where `Cond` is a well-formed selection condition over $\mathcal{A}$, and $A_1, \ldots, A_k \in \mathcal{A}$, for $\mathcal{A} = \bigcup_{R \in \{R_1, \ldots, R_m\}} \mathit{attributes}(R)$.

FO review: syntax

Fix a database schema D and let $R_1, \ldots, R_m \in D$.

SQL

```
SELECT A1, ..., Ak
FROM R1, ..., Rm
WHERE Cond
```

where `Cond` is a well-formed selection condition over $\mathcal{A}$, and $A_1, \ldots, A_k \in \mathcal{A}$, for $\mathcal{A} = \bigcup_{R \in \{R_1, \ldots, R_m\}} \mathit{attributes}(R)$.

Relational Algebra (RA)

$$\pi_{A_1, \ldots, A_k}(\sigma_{Cond}(R_1 \times \cdots \times R_m))$$

FO review: syntax

Datalog

$$\mathit{result}(A_1, \ldots, A_k) \leftarrow R_1(\overline{A^1}), \ldots, R_m(\overline{A^m}), C_1, \ldots, C_j$$

where

- $\overline{A^i}$ is a list of variables of length $\mathit{arity}(S_i)$ (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(authorID, name, birthdate, language)
book(bookID, title, authorID, publisher, language, year)
store(storeID, address, phone)
sells(storeID, bookID)

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(<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 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

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

FO review: syntax

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

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

FO review: syntax

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

```
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
```

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

$$result(A) \leftarrow author(A, N, B, L), L = \text{‘Italian’}$$
$$result(A) \leftarrow author(A, N, B, L), B = 1985$$

FO review: syntax

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

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

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

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

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: complexity of 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: complexity of 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: complexity of 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: complexity of 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 "acylic" conjunctive queries
  - queries with join trees
  - full def in a later lecture

FO review: complexity of query evaluation

Fact. The complexity of query evaluation is as follows

	FO	*Conj*	*AConj*
combined	PSPACE-complete	NP-complete	LOGCFL-complete
data	Logspace	Logspace	Linear time

where *combined* means in the size of the query and the database and *data* means in the size of the database (i.e., for some fixed query).

Books schema

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

Books schema

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

1. List (the bookIDs of) 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 (the bookIDs of) all books authored by a native speaker.
2. List (the authorIDs of) all authors who only have books appearing in their native language.

Books schema

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

1. List (the bookIDs of) all books authored by a native speaker.
2. List (the authorIDs of) all authors who only have books appearing in their native language.
3. List all books which are not available in stores.

Books schema

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

1. List (the bookIDs of) all books authored by a native speaker.
2. List (the authorIDs of) all authors who only have books appearing in their native language.
3. List all books which are not available in stores.
4. 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 (the bookIDs of) all books authored by a native speaker.
2. List (the authorIDs of) all authors who only have books appearing in their native language.
3. List all books which are not available in stores.
4. List all books sold in more than one store.
5. List all books sold in exactly one store.

Books schema

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

1. List (the bookIDs of) all books authored by a native speaker.
2. List (the authorIDs of) all authors who only have books appearing in their native language.
3. List all books which are not available in stores.
4. List all books sold in more than one store.
5. List all books sold in exactly one store.
6. List all (authorID, language) pairs where the given author has not published a book in the given 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 (the bookIDs of) all books authored by a native speaker.
2. List (the authorIDs of) all authors who only have books appearing in their native language.
3. List all books which are not available in stores.
4. List all books sold in more than one store.
5. List all books sold in exactly one store.
6. List all (authorID, language) pairs where the given author has not published a book in the given language.
7. List all authors which have a book in every known language (i.e., every language occuring in the author or book tables). Note that this can be a different book for each language.