user queries (RA, SQL, etc.)

relational database

Flight | origin | destination | airline
--- | --- | --- | ---
VIE | LHR | BA
LHR | EDI | BA
LGW | GLA | U2
LCA | VIE | OS

Airport | code | city
--- | --- | ---
VIE | Vienna
LHR | London
LGW | London
LCA | Larnaca

a standard database system
Flight origin destination airline
VIE LHR BA
LHR EDI BA
LGW GLA U2
LCA VIE OS

Airport code city
VIE Vienna
LHR London
LGW London
LCA Larnaca

...but, we live in the era of big data
**Volume**
size does maters
(thousands of TBs of data)

**Variety**
many data formats
(structured, semi-structured, etc.)

**Veracity**
data is often incomplete/inconsistent

**Velocity**
data often arrives at fast speed
(updates are frequent)
Volume
size does matter
(thousands of TBs of data)

Variety
many data formats
(structured, semi-structured, etc.)

Veracity
data is often incomplete/inconsistent

Velocity
data often arrives at fast speed
(updates are frequent)

the rest of this course
Volume Challenges

- Many standard algorithms for data processing do not scale

- We may not even have what can realistically be called an algorithm
  - Data must be at least scanned (classical assumption in databases)
  - The best case is a linear time algorithm
  - But, consider a linear scan on the best available device (6GB/s)
    - 1 PetaByte (PB) = 10^6 GBs is scanned in about 2 days
    - 1 ExaByte (EB) = 10^9 GBs is scanned in about 5 years
  - We have PB data sets, while EB data sets are not far away

⇒ linear time, let alone polynomial time, is not good enough
Possible Approaches

• **Scale Independence** – find queries that can be answered regardless of scale

• **Replace** the query with one that is much faster to execute
Scale Independence
Query Answering on Big Data

- Answer a query on a big database using a small subset of it

![Diagram showing a large database on the left and a small subset on the right, with the equation Q = \_]

- Then, exploit existing database technology to answer queries on big data
Scale Independence

- Armbrust et al. considered the notion of scale independence

  The evaluation of queries using a number of “operations” that is independent of the size of data


Scale Independence: Facebook Example

Find all friends of a person who live in NYC

Q(p,n) :- FriendOf(p,id), Person(id,n,NYC)
Scale Independence: Facebook Example

Find all friends of a person who live in NYC

Person | id | name | city
-------|----|------|-----

FriendOf | id₁ | id₂ |
---------|-----|-----|
P₀       |
...

Q(P₀,n) :- FriendOf(P₀,id), Person(id,n,NYC)

- We are interested in a certain person P₀
[**Scale Independence: Facebook Example**]

**Find all friends of a person who live in NYC**

<table>
<thead>
<tr>
<th>Person</th>
<th>id</th>
<th>name</th>
<th>city</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FriendOf</th>
<th>id₁</th>
<th>id₂</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P₀</td>
<td></td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td></td>
<td>P₀</td>
<td></td>
</tr>
</tbody>
</table>

\[ Q(P₀, n) :\text{- FriendOf}(P₀, id), \text{Person}(id, n, \text{NYC}) \]

- We are interested in a certain person \( P₀ \)
- Cardinality constraint: Facebook has a limit of 5000 friends per user
Scale Independence: Facebook Example

Find all friends of a person who live in NYC

We are interested in a certain person $P_0$

Cardinality constraint: Facebook has a limit of 5000 friends per user

Key constraint: id is the key attribute of Person

$Q(P_0,n) \ :- \ \text{FriendOf}(P_0,id), \ \text{Person}(id,n,NYC)$

$\Rightarrow 10000$ tuples in total are needed

…and these tuples can be fetched efficiently by using indices on id attributes
Scale Independence: Facebook Example

Find all friends of a person who live in NYC

\[ Q(P_0, n) : \text{FriendOf}(P_0, id), \text{Person}(id, n, \text{NYC}) \]

- We are interested in a certain person \( P_0 \)
- Cardinality constraint: Facebook has a limit of 5000 friends per user
- Key constraint: id is the key attribute of Person

For a given person, this query can be answered using a bounded number of tuples, independent of the size of the Facebook graph
Towards a Theory on Scale Independence

• The previous example shows that it is feasible to answer a query $Q$ in a big database $D$ by accessing a bounded amount of data.

• However, to make practical use of scale independence, several fundamental questions have to be answered:
  1. Given $Q$ and $D$, can we decide whether $Q$ is scale independent in $D$?
  2. If such an identification is expensive, can we find sufficient conditions?
  3. If $Q$ is scale independent in $D$, can we effectively identify a small $D_Q \subseteq D$?
  4. Can we achieve reasonable time bounds for finding $D_Q$ and computing $Q(D_Q)$?
Scale Independence: Definition

we refer to first-order queries (FO)* and conjunctive queries (CQ)

• A query $Q$ is scale independent in a database $D$ w.r.t. $M \geq 0$ if there exists a subset $D_Q \subseteq D$ such that:
  1. $|D_Q| \leq M$
  2. $Q(D_Q) = Q(D)$

• We say that $Q$ is scale independent w.r.t. $M \geq 0$ if $Q$ is scale independent in $D$ w.r.t. $M$, for every database $D$

*notice that $\text{FO} = \text{RA} = \text{DRC} = \text{TRC}$
Scale Independence: Algorithmic Problems

\[ \text{QDSI}(L) \]

**Input**: a database \( D \), a query \( Q \in L \), and \( M \geq 0 \)

**Question**: is \( Q \) scale independent in \( D \) w.r.t. \( M \)?

**Data complexity**, i.e., fixed \( Q \) – this gives rise to the problem \( \text{QDSI}[Q](L) \) for a fixed query \( Q \in L \)

\[ \text{QSI}(L) \]

**Input**: a query \( Q \in L \), and \( M \geq 0 \)

**Question**: is \( Q \) scale independent w.r.t. \( M \)?
## Complexity of QDSI(L)

<table>
<thead>
<tr>
<th>L</th>
<th>Non-Boolean</th>
<th>Boolean</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Combined, Data</td>
<td>Combined, Data</td>
</tr>
<tr>
<td>CQ</td>
<td>Σ₃,ₚ⁻c, NP⁻c</td>
<td>O(1)-time, O(1)-time</td>
</tr>
<tr>
<td>FO</td>
<td>PSPACE⁻c, NP⁻c</td>
<td>PSPACE⁻c, NP⁻c</td>
</tr>
</tbody>
</table>

third level of the polynomial hierarchy

NP ⊆ Σ₃,ₚ ⊆ PSPACE

assuming that |Q| ≤ M

### Proof idea (upper bounds):

- Given Q, D, M and D' ⊆ D such that |D'| ≤ M, decide whether Q(D) = Q(D')
- Solve the complement of QDSI(L) by calling the algorithm for the above problem
Complexity of QSI(L)

• Conjunctive queries are never scale independent w.r.t. some \( M \geq 0 \), unless the query is trivial
  – This is due to monotonicity, i.e., \( D \subseteq D' \Rightarrow Q(D) \subseteq Q(D') \)
  – Example of a trivial query: returns a constant tuple over all databases

• QSI(FO) is undecidable. Why? (hint: consider the case when \( M = 0 \))
  – This holds even for Boolean queries
  – The class of scale independent FO queries is not recursively enumerable
Facebook Example Revisited

Find all friends of a person who live in NYC

<table>
<thead>
<tr>
<th>Person</th>
<th>id</th>
<th>name</th>
<th>city</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FriendOf</th>
<th>id₁</th>
<th>id₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>P₀</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P₀</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ Q(P₀, n) \leftarrow \text{FriendOf}(P₀, id), \text{Person}(id, n, NYC) \]

- We are interested in a certain person \( P₀ \)
- Cardinality constraint: Facebook has a limit of 5000 friends per user
- Key constraint: id is the key attribute of Person

For a given person, this query can be answered using a bounded number of tuples, independent of the size of the Facebook graph.
Find all friends of a person who live in NYC

Q(P₀, n) :- FriendOf(P₀, id), Person(id, n, NYC)

- We are interested in a certain person P₀.
- Cardinality constraint: Facebook has a limit of 5000 friends per user.
- Key constraint: id is the key attribute of Person.

For a given person, this query can be answered using a bounded number of tuples, independent of the size of the Facebook graph.
Access Schemas: Definition

- Consider a relational schema $R = \{R_1, \ldots, R_n\}$. An access schema $A$ over $R$ is a set of tuples $(R, X, N, T)$ where
  - $R \in R$
  - $X$ is a set of attributes of $R$
  - $N, T$ are natural numbers

- A database $D$ (over $R$) conforms to $A$ if for each tuple $(R, X, N, T) \in A$ the following hold:
  - Size bound: for each tuple $t$ of values for the attributes $X$, $|\sigma_{X=t}(D)| \leq N$
  - Time bound: $\sigma_{X=t}(D)$ can be retrieved in time at most $T$
Facebook Example – Access Schemas

- id is a key attribute for Person (size bound)
- it takes time $T_1$ to retrieve a tuple based on its key value (time bound)

$$A = \{(Person, \{id\}, 1, T_1), (FriendOf, \{id\}, 5000, T_1)\}$$

the Facebook graph conforms to $A$

- if $id_1$ is provided, at most 5000 tuples with such an id exist (size bound)
- it takes time $T_2$ to retrieve those tuples (time bound)
Facebook Example – Access Schemas

Find all friends of a person who live in NYC

<table>
<thead>
<tr>
<th>Person</th>
<th>id</th>
<th>name</th>
<th>city</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FriendOf</th>
<th>id_1</th>
<th>id_2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P_0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td></td>
<td>P_0</td>
<td></td>
</tr>
</tbody>
</table>

\[Q(P_0, n) \; :- \; \text{FriendOf}(P_0, id), \; \text{Person}(id, n, \text{NYC})\]

\[A = \{(\text{Person}, \{id\}, 1, T_1), \; (\text{FriendOf}, \{id_1\}, 5000, T_1)\}\]

By only looking at the access schema we can tell whether we can efficiently answer the given query
Scale Independence Under Access Schemas

- Given a schema $R$, access schema $A$ over $R$, and a query $Q(x,y)$, we say that $Q$ is $x$-scale independent under $A$ if for each database $D$ that conforms to $A$, and each tuple of values $t$ for $x$, the answer to $Q_t = Q(t,y)$ over $D$ can be computed in time that depends only on $A$ and $Q$, but not on $D$.

- For a fixed query $Q(x,y)$, $Q$ is efficiently $x$-scale independent under $A$ if for each database $D$ that conforms to $A$, and each tuple of values $t$ for $x$, the answer to $Q_t = Q(t,y)$ over $D$ can be computed in polynomial time in $A$. 
Facebook Example – Access Schemas

Find all friends of a person who live in NYC

<table>
<thead>
<tr>
<th>Person</th>
<th>id</th>
<th>name</th>
<th>city</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FriendOf</td>
<td>id₁</td>
<td>id₂</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ Q(p,n) \triangleq \text{FriendOf}(p,\text{id}), \text{Person}(\text{id},n,\text{NYC}) \]

\[ A = \{(\text{Person}, \{\text{id}\}, 1, T₁), (\text{FriendOf}, \{\text{id₁}\}, 5000, T₁)\} \]

\[ Q \text{ is efficiently } \{p\}\text{-scale independent under } A \]
Can we Characterize Such Queries?

• It is an **undecidable** problem whether a query is $x$-scale independent under an access schema

• The lack of effective syntactic characterizations of semantic classes of queries is common in databases $\Rightarrow$ isolate practically relevant sufficient conditions

• **Goal**: provide a **syntactic class** of queries such that
  - Is sufficiently large to cover interesting queries
  - Guarantees that the queries are efficiently scale independent
Controllability and Scale Independence

**Define:** a syntactic class of so-called $x$-controllable FO queries for a given access schema, where $x$ is a subset of the free variables of a query

**Show:** each $x$-controlled query under access schema $A$ is efficiently $x$-scaled independent under $A$

an $x$-controlled query under $A$ can be answered efficiently on big databases that conform to $A$
**x-controllability: Atom Rule**

\[
\text{IF } (R, X, N, T) \in A \\
\text{THEN } R(y) \text{ is } x\text{-controlled under } A, \text{ where } x \text{ is the subtuple of } y \text{ corresponding to } X
\]

<table>
<thead>
<tr>
<th>Atomic Query</th>
<th>Access Schema A</th>
<th>Controlling Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>FriendOf(p,id)</td>
<td>(FriendOf, {p}, 5000, T_1)</td>
<td>{p}</td>
</tr>
<tr>
<td>Visit(id,rid,yy,mm,dd)</td>
<td>(Visit, {id,yy,mm,dd}, 1, T_2)</td>
<td>{id, yy, mm, dd}</td>
</tr>
<tr>
<td>Person(id,pn,NYC)</td>
<td>(Person, {id}, 1, T_3)</td>
<td>{id}</td>
</tr>
<tr>
<td>Dates( yy, mm, dd)</td>
<td>(Dates, {yy}, 366, T_4)</td>
<td>{yy}</td>
</tr>
<tr>
<td>Restaurant(rid,rn,NYC,A)</td>
<td>(Restaurant, {rid}, 1, T_5)</td>
<td>{rid}</td>
</tr>
</tbody>
</table>

We underline the controlling variables: FriendOf(p,id), Visit(id,rid,yy,mm,dd), etc.
**x-controllability: Conjunction Rule**

If $Q_i(x_i, y_i)$ is $x_i$-controlled under $A$ for $i \in \{1,2\}$

Then $Q_1 \land Q_2$ is $(x_1 \cup (x_2 - y_1))$-controlled and $(x_2 \cup (x_1 - y_2))$-controlled under $A$

Consider the queries

$Q_1(id, rid, yy, mm, dd) :- \text{Visit}(id, rid, yy, mm, dd)$  
$Q_2(yy, mm, dd) :- \text{Dates}(yy, mm, dd)$

and the query

$Q(id, rid, yy, mm, dd) :- \text{Visit}(id, rid, yy, mm, dd), \text{Dates}(yy, mm, dd)$

Controlling variables: \{id,yy,mm,dd\} $\cup$ ($\{yy\} - \{rid\}$) = \{id,yy,mm,dd\} or

\{yy\} $\cup$ ($\{id,yy,mm,dd\} - \{mm,dd\}$) = \{id,yy\}

$\Rightarrow$ $Q$ is \{id,yy,mm,dd\}-controlled and \{id,yy\}-controlled under $A$
**x-controllability: Existential Quantification Rule**

\[
\text{IF } Q(y) \text{ is } x\text{-controlled under } A, \text{ and } z \text{ is a subtuple of } y - x \\
\text{THEN } \exists z \text{ } Q \text{ is } x\text{-controlled under } A
\]

Consider the query

\[
Q(id, rid, yy, mm, dd) :\text{- Visit}(id, rid, yy, mm, dd), \text{ Dates}(yy, mm, dd)
\]

and recall that is \( \{id, yy\} \)-controlled under \( A \)

Then, the query

\[
Q(id, yy) :\text{- Visit}(id, rid, yy, mm, dd), \text{ Dates}(yy, mm, dd)
\]

is also \( \{id, yy\} \)-controlled under \( A \). Why?
**x-controllability: Other Rules**

- Similar rules are defined for:
  - Conditions: if \( Q(x) \) is a Boolean combination of \( x_i = x_j \), then \( Q \) is \( x \)-controlled
  - Disjunction - \( Q_1 \lor Q_2 \)
  - (Safe) Negation - \( Q_1 \land \neg Q_2 \)
  - Universal quantification - \( \forall y \ (Q(x,y) \rightarrow Q(z)) \)
  - Expansion: \( Q(y) \) is \( x \)-controlled under \( A \) and \( x \subseteq z \subseteq y \), then \( Q \) is \( z \)-controlled under \( A \)

- In isolation, all the above rules are optimal, i.e., we cannot achieve smaller controlling tuples
$x$-controllability: Example

$$Q(y, p, r) :\text{ FriendOf}(p, id), \text{ Visit}(id, rid, yy, mm, dd), \text{ Person}(id, pn, NYC), \text{ Dates}(yy, mm, dd), \text{ Restaurant}(rid, rn, NYC, A)$$

Is $Q\{yy,p\}$-controllable under $A$?

**Access Schema A**

<table>
<thead>
<tr>
<th>Relation</th>
<th>Attributes</th>
<th>Access</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>FriendOf</td>
<td>{p}</td>
<td>5000</td>
<td>$T_1$</td>
</tr>
<tr>
<td>Visit</td>
<td>{id, yy, mm, dd}</td>
<td>1</td>
<td>$T_2$</td>
</tr>
<tr>
<td>Person</td>
<td>{id}</td>
<td>1</td>
<td>$T_3$</td>
</tr>
<tr>
<td>Dates</td>
<td>{yy}</td>
<td>366</td>
<td>$T_4$</td>
</tr>
<tr>
<td>Restaurant</td>
<td>{rid}</td>
<td>1</td>
<td>$T_5$</td>
</tr>
</tbody>
</table>
x-controllability: Example

Step 1

\[ Q_{\text{FriendOf}}(p, id) :\text{FriendOf}(p, id) \]
\[ Q_{\text{Visit}}(id, rid, yy, mm, dd) :\text{Visit}(id, rid, yy, mm, dd) \]
\[ Q_{\text{Person}}(id, pn) :\text{Person}(id, pn, NYC) \]
\[ Q_{\text{Dates}}(yy, mm, dd) :\text{Dates}(yy, mm, dd) \]
\[ Q_{\text{Restaurant}}(rid, rn) :\text{Restaurant}(rid, rn, NYC, A) \]

---

Step 2

\[ Q_1(id, rid, yy, mm, dd) : Q_{\text{Visit}}(id, rid, yy, mm, dd), Q_{\text{Dates}}(yy, mm, dd) \]

---

Step 3

\[ Q_2(id, rid, yy, mm, dd, pn) : Q_1(id, rid, yy, mm, dd), Q_{\text{Person}}(id, pn) \]

---

Step 4

\[ Q_3(id, rid, yy, mm, dd, pn, p) : Q_2(id, rid, yy, mm, dd, pn), Q_{\text{FriendOf}}(p, id) \]

---

Step 5

\[ Q_4(id, rid, yy, mm, dd, pn, p, rn) : Q_3(id, rid, yy, mm, dd, pn, p), Q_{\text{Restaurant}}(rid, rn) \]

---

Step 6

\[ Q(yy, p, rn) : Q_4(id, rid, yy, mm, dd, pn, p, rn) \]
Main Result on \( x \)-controllability

**Theorem:** Consider a first-order query \( Q \), and an access schema \( A \).
If \( Q \) is \( x \)-controlled under \( A \), then \( Q \) is efficiently \( x \)-scale independent under \( A \).

**Proof hint:** Show by induction on the structure of \( Q(x,y) \), given a tuple of values \( t \) for \( x \), how to retrieve a set \( D_{Q,t} \subseteq D \) such that \( Q_t(D_{Q,t}) = Q_t(D) \), where \( Q_t = Q(t,y) \), and establish polynomial bounds for its size and query evaluation time.

- The above result states that by filling the variables \( x \) in \( Q \) by \( t \), \( Q_t \) can be answered on any database that conforms to \( A \) in polynomial time in \( A \).

- An effective plan for identifying \( D_{Q,t} \subseteq D \) such that \( Q_t(D_{Q,t}) = Q_t(D) \) can be obtained...
Q\{(yy,p,rn)\} := \text{FriendOf}(p, id), \text{Visit}(id, rid, yy, mm, dd), \text{Person}(id, pn, NYC),
\text{Dates}(yy, mm, dd), \text{Restaurant}(rid, rn, NYC, A)

Q \{yy,p\}-controllable under A

<table>
<thead>
<tr>
<th>Access Schema A</th>
</tr>
</thead>
<tbody>
<tr>
<td>(FriendOf, {p}, 5000, T_1)</td>
</tr>
<tr>
<td>(Visit, {id,yy,mm,dd}, 1, T_2)</td>
</tr>
<tr>
<td>(Person, {id}, 1, T_3)</td>
</tr>
<tr>
<td>(Dates, {yy}, 366, T_4)</td>
</tr>
<tr>
<td>(Restaurant, {rid}, 1, T_5)</td>
</tr>
</tbody>
</table>
Effective Plan: Example

Visit(id, rid, yy, mm, dd)

Dates(yy, mm, dd)

Person(id, pn, NYC)

FriendOf(p, id)

Restaurant(rid, rn, NYC, A)

Q(yy, p, rn)
Visit(id, rid, yy, mm, dd) \\
\forall Visit(id, rid, yy, mm, dd) \\
Q(yy, p, rn) \\
Q(yy, p, rn) \\
Restaurant(rid, rn, NYC, A) \\
Restaurant(rid, rn, NYC, A) \\
\forall FriendOf(p, id) \\
\forall Person(id, p, id) \\
\forall Person(id, pn, NYC) \\
\forall Person(id, pn, NYC) \\
\exists Dates(yy, mm, dd) \\
\exists Dates(yy, mm, dd)
Effective Plan: Example

\[
Q(\text{yy}, \text{p}, \text{rn}) \\
\exists \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
\land \\
Effective Plan: Example

\[ Q(yy, p, rn) \]

\[ \exists \]

\[ \land \]

\[ \land \]

Restaurant(rid, rn, NYC, A)

\[ \land \]

FriendOf(p, id)

\[ \land \]

Person(id, pn, NYC)

\[ \land \]

Visit(id, rid, yy, mm, dd)

\[ \land \]

Dates(yy, mm, dd)
Effective Plan: Example

Visit(id, rid, yy, mm, dd)

Dates(yy, mm, dd)

Person(id, pn, NYC)

Restaurant(rid, rn, NYC, A)

FriendOf(p, id)

Q(yy, p, rn)
Effective Plan: Example
Effective Plan: Example

Visit(id, rid, yy, mm, dd) 

Dates(yy, mm, dd) 

FriendOf(p, id) 

Restaurant(rid, rn, NYC, A) 

Person(id, pn, NYC) 

Q(yy, p, rn)
Wrap-Up

- A fixed first-order query $Q(x,y)$ that is $x$-controlled under an access schema $A$ is efficiently $x$-scale independent under $A$, i.e., with $Q_t = Q(t,y)$:

  $$Q_t(\{D\}) = Q_t(D_{Q,t})$$

  database that conforms to $A$

  of polynomial size in $A$
  (identified via an effective plan)

- Then, exploit existing database technology to answer $Q_t$ on $D_{Q,t}$ (and thus on $D$)
Associated Papers


  Two early systems paper on scalability; what we saw in class was a formalization of their approach

- Michael Armbrust, Eric Liang, Tim Kraska, Armando Fox, Michael J. Franklin, David A. Patterson: Generalized scale independence through incremental precomputation. SIGMOD 2013:625-636

  Scalability under updates to the underlying data
Associated Papers

• Wenfei Fan, Floris Geerts, Frank Neven: Making Queries Tractable on Big Data with Preprocessing. PVLDB 6(9): 685-696 (2013)

New notions of complexity for handling large volumes of data

• Wenfei Fan, Floris Geerts, Leonid Libkin: On scale independence for querying big data. PODS 2014:51-62

We saw the notion of controllability here. Eligible topics for an essay are incremental computation and using views

• Yang Cao, Wenfei Fan, Tianyu Wo, Wenyuan Yu: Bounded Conjunctive Queries. PVLDB 7(12): 1231-1242 (2014)

Specialized algorithms for handling select-project-join queries over big data