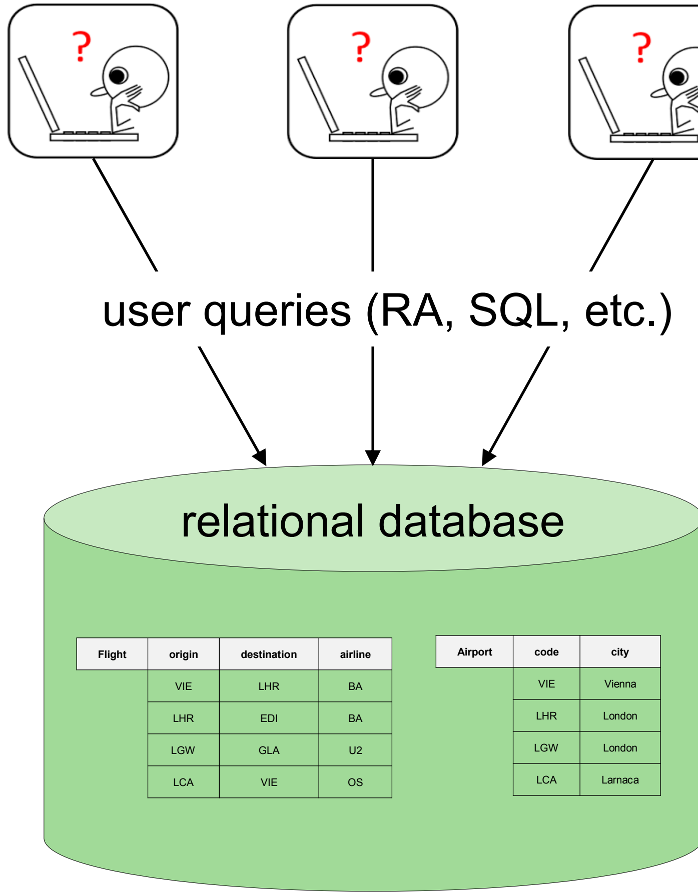
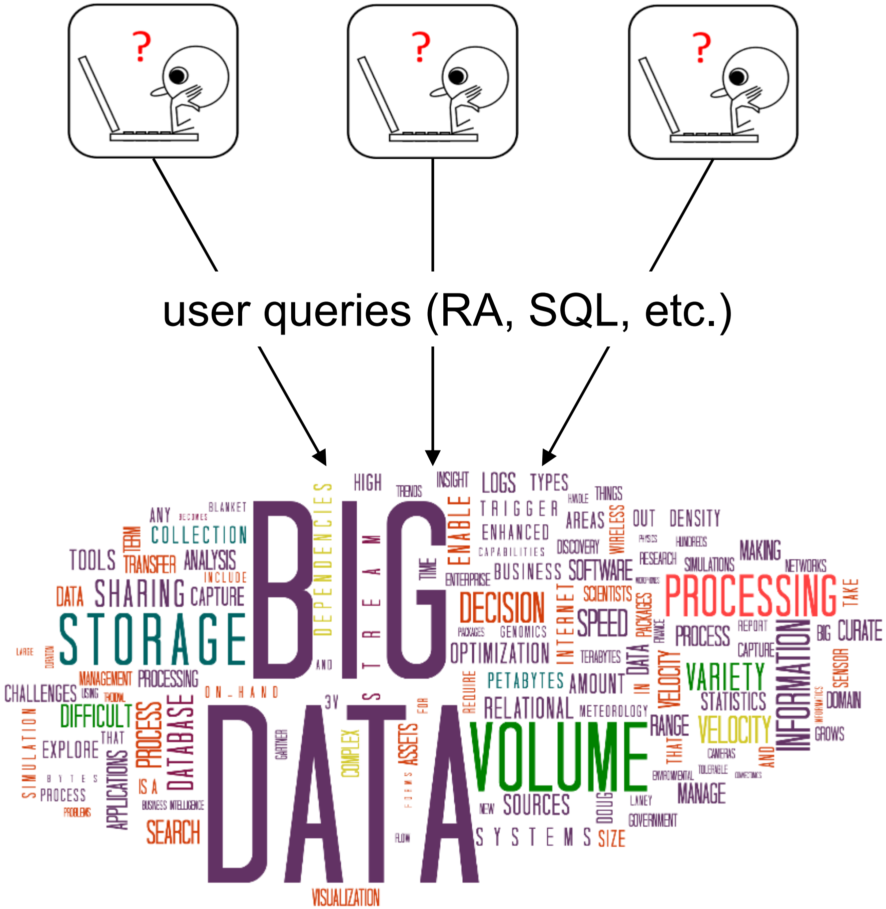


Flight	origin	destination	airline
	VIE	LHR	BA
	LHR	EDI	BA
	LGW	GLA	U2
	LCA	VIE	OS

Airport	code	city
	VIE	Vienna
	LHR	London
	LGW	London
	LCA	Larnaca

a standard database system



…but, we live in the era of big data

Volume
size does mattes
(thousands of TBs of data)

Veracity
data is often
incomplete/inconsistent

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

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

the rest of this course

Volume
size does mattes
(thousands of TBs of data)

Veracity
data is often
incomplete/inconsistent

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

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

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

$\Rightarrow$ linear time, let alone polynomial time, is not good enough

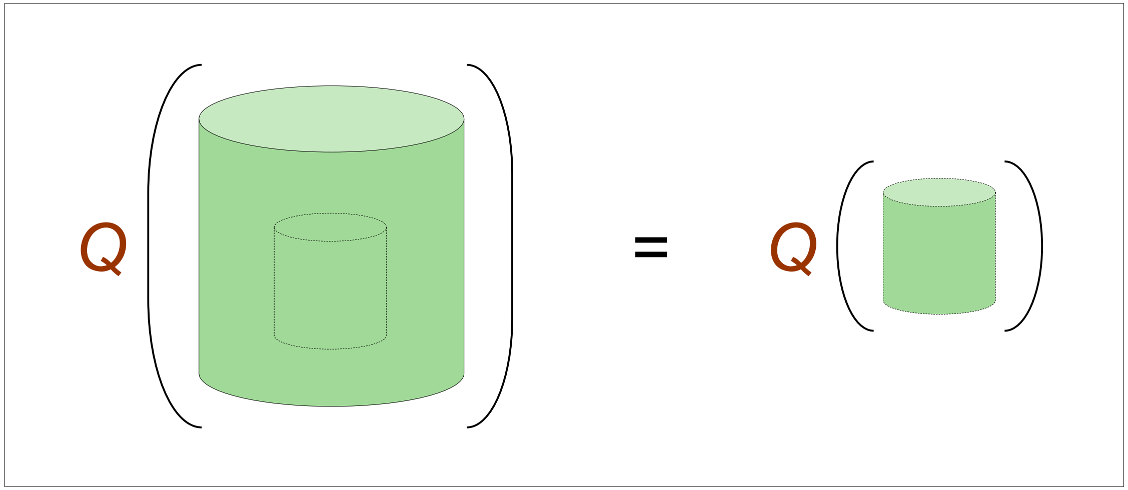
Possible Approaches

- Scale Independence – find queries than 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



- 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

– M. Armbrust, A. Fox, D. A. Patterson, N. Lanham, B. Trushkowsky, J. Trutna, and H. Oh. Scads: Scale-independent storage for social computing applications. In CIDR, 2009.

– M. Armbrust, K. Curtis, T. Kraska, A. Fox, M. J. Franklin, and D. A. Patterson. PIQL: Success-tolerant query processing in the cloud. In VLDB, 2011.

– M. Armbrust, E. Liang, T. Kraska, A. Fox, M. J. Franklin, and D. Patterson. Generalized scale independence through incremental precomputation. In SIGMOD, 2013.

Scale Independence: Facebook Example

Find all friends of a person who live in NYC

Person	<u>id</u>	name	city

FriendOf	id_1	id_2

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	<u>id</u>	name	city

FriendOf	id_1	id_2
	P_0	
	…	
	P_0	

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

- We are interested in a certain person P_0

Scale Independence: Facebook Example

Find all friends of a person who live in NYC

Person	id	name	city

FriendOf	id_1	id_2
	P_0	
	…	
	P_0	

≤ 5000

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

- We are interested in a certain person P_0
- 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

Person	<u>id</u>	name	city
	P		

≤ 1

FriendOf	id_1	id_2
	P_0	P
	…	
	P_0	

≤ 5000

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

- 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

$\Rightarrow$ 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

Person	<u>id</u>	name	city
	P		

FriendOf	id_1	id_2
	P_0	P
	…	
	P_0	

≤ 1

≤ 5000

$Q(P_0,n)$:- $FriendOf(P_0,id), Person(id,n,NYC)$

- 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 ≥ 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 ≥ 0 if Q is scale independent in D w.r.t. M, for every database D

*notice that **FO** = **RA** = **DRC** = **TRC**

Scale Independence: Algorithmic Problems

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 QDSI[*Q*](**L**) for a fixed query *Q* $\in$ **L**

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**)

L	**Non-Boolean**		**Boolean**	
	Combined	Data	Combined	Data
CQ	$\Sigma_{3,P}$-c	NP-c	$O(1)$-time	$O(1)$-time
FO	PSPACE-c	NP-c	PSPACE-c	NP-c

third level of the polynomial hierarchy

NP $\subseteq \Sigma_{3,P} \subseteq$ PSPACE

assuming that $|Q| \leq M$

Proof idea (upper bounds):

- Given Q, D, M and $D' \subseteq D$ such that $|D'| \leq M$, 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

Person	<u>id</u>	name	city
	P		

FriendOf	id_1	id_2
	P_0	P
	…	
	P_0	

≤ 1

≤ 5000

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

- 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

Facebook Example Revisited

Find all friends of a person who live in NYC

Person	<u>id</u>	name	city
	P		

≤ 1

FriendOf	id_1	id_2
	P_0	P
	…	
	P_0	

≤ 5000

$Q(P_0,n)$:- FriendOf(P_0,id), Person(id,n,NYC)

- We are interested in a certain person P...
- Cardinality constraint: F... limit of 5000 friends per user
- Key constrai... key attribute of Person

information about access to data

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

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

Facebook Example – Access Schemas

key attribute

Person	<u>id</u>	name	city

FriendOf	id_1	id_2
	P_0	
	…	
	P_0	

≤ 5000

- 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)

$$\mathbf{A} = \{(\text{Person}, \{\text{id}\}, 1, T_1), (\text{FriendOf}, \{\text{id}_1\}, 5000, T_1)\}$$

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

Person	<u>id</u>	name	city

FriendOf	id_1	id_2
	P_0	
	…	
	P_0	

≤ 5000

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

$$\mathbf{A} = \{(\text{Person}, \{id\}, 1, T_1), (\text{FriendOf}, \{id_1\}, 5000, T_1)\}$$

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 $Q(\mathbf{x},\mathbf{y})$, 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 $Q_{\mathbf{t}} = Q(\mathbf{t},\mathbf{y})$ over D can be computed in time that depends only on **A** and Q, but not on D

- For a fixed query $Q(\mathbf{x},\mathbf{y})$, 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 $Q_{\mathbf{t}} = Q(\mathbf{t},\mathbf{y})$ over D can be computed in polynomial time in **A**

Facebook Example – Access Schemas

Find all friends of a person who live in NYC

Person	id	name	city

FriendOf	id_1	id_2

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

A = {(Person, {id}, 1, T_1), (FriendOf, {id_1}, 5000, T_1)}

Q is efficiently {p}-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

IF (R,X,N,T) $\in$ **A**

THEN R(**y**) is **x**-controlled under **A**, where **x** is the subtuple of **y** corresponding to X

Atomic Query	**Access Schema A**	**Controlling Variables**
FriendOf(p,id)	(FriendOf, {p}, 5000, T_1)	{p}
Visit(id,rid,yy,mm,dd)	(Visit, {id,yy,mm,dd}, 1, T_2)	{id, yy, mm, dd}
Person(id,pn,NYC)	(Person, {id}, 1, T_3)	{id}
Dates(yy,mm,dd)	(Dates, {yy}, 366, T_4)	{yy}
Restaurant(rid,rn,NYC,A)	(Restaurant, {rid}, 1, T_5)	{rid}

We underline the controlling variables: FriendOf($\underline{p}$,id), Visit($\underline{id}$,rid,$\underline{yy}$,$\underline{mm}$,$\underline{dd}$), etc.

x-controllability: Conjunction Rule

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

THEN $Q_1 \wedge Q_2$ is $(\mathbf{x_1} \cup (\mathbf{x_2} - \mathbf{y_1}))$-controlled and $(\mathbf{x_2} \cup (\mathbf{x_1} - \mathbf{y_2}))$-controlled under **A**

Consider the queries

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

and the query

Q(id,rid,yy,mm,dd) :- Visit(id,rid,yy,mm,dd), Dates(yy,mm,dd)

Controlling variables: {id,yy,mm,dd} ∪ ({yy} – {rid}) = {id,yy,mm,dd} or

{yy} ∪ ({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

IF $Q(\mathbf{y})$ is **x**-controlled under **A**, and **z** is a subtuple of **y** – **x**

THEN $\exists\mathbf{z}\ Q$ is **x**-controlled under **A**

Consider the query

Q(id,rid,yy,mm,dd) :- Visit(id,rid,yy,mm,dd), Dates(yy,mm,dd)

and recall that is {id,yy}-controlled under **A**

less distinguished variables

Then, the query

Q(id,yy) :- Visit(id,rid,yy,mm,dd), Dates(yy,mm,dd)

is also {id,yy}-controlled under **A**. Why?

x-controllability: Other Rules

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

$Q(yy,p,rn)$:- FriendOf(p,id), Visit(id,rid,yy,mm,dd), Person(id,pn,NYC), Dates(yy,mm,dd), Restaurant(rid,rn,NYC,A)

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

Access Schema A
(FriendOf, {p}, 5000, T_1)
(Visit, {id,yy,mm,dd}, 1, T_2)
(Person, {id}, 1, T_3)
(Dates, {yy}, 366, T_4)
(Restaurant, {rid}, 1, T_5)

x-controllability: Example

Step 1

$$Q_{\mathrm{FriendOf}}(\underline{p},id) \ \ \text{:-} \ \ \mathrm{FriendOf}(\underline{p},id)$$

$$Q_{\mathrm{Visit}}(\underline{id},rid,\underline{yy,mm,dd}) \ \ \text{:-} \ \ \mathrm{Visit}(\underline{id},rid,\underline{yy,mm,dd})$$

$$Q_{\mathrm{Person}}(\underline{id},pn) \ \ \text{:-} \ \ \mathrm{Person}(\underline{id},pn,NYC)$$

$$Q_{\mathrm{Dates}}(\underline{yy},mm,dd) \ \ \text{:-} \ \ \mathrm{Dates}(\underline{yy},mm,dd)$$

$$Q_{\mathrm{Restaurant}}(\underline{rid},rn) \ \ \text{:-} \ \ \mathrm{Restaurant}(\underline{rid},rn,NYC,A)$$

Step 2

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

Step 3

$$Q_2(\underline{id},rid,\underline{yy},mm,dd,pn) \ \ \text{:-} \ \ Q_1(\underline{id},rid,\underline{yy},mm,dd),\ Q_{\mathrm{Person}}(\underline{id},pn)$$

Step 4

$$Q_3(id,rid,\underline{yy},mm,dd,pn,\underline{p}) \ \ \text{:-} \ \ Q_2(\underline{id},rid,\underline{yy},mm,dd,pn),\ Q_{\mathrm{FriendOf}}(\underline{p},id)$$

Step 5

$$Q_4(id,rid,\underline{yy},mm,dd,pn,\underline{p},rn) \ \ \text{:-} \ \ Q_3(id,rid,\underline{yy},mm,dd,pn,\underline{p}),\ Q_{\mathrm{Restaurant}}(\underline{rid},rn)$$

Step 6

$$Q(\underline{yy,p},rn) \ \ \text{:-} \ \ Q_4(id,rid,\underline{yy},mm,dd,pn,\underline{p},rn)$$

✓

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 $Q(\mathbf{x},\mathbf{y})$, given a tuple of values **t** for **x**, how to retrieve a set $D_{Q,\mathbf{t}} \subseteq D$ such that $Q_{\mathbf{t}}(D_{Q,\mathbf{t}}) = Q_{\mathbf{t}}(D)$, where $Q_{\mathbf{t}} = Q(\mathbf{t},\mathbf{y})$, 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**, $Q_{\mathbf{t}}$ can be answered on any database that conforms to **A** in polynomial time in **A**
- An effective plan for identifying $D_{Q,\mathbf{t}} \subseteq D$ such that $Q_{\mathbf{t}}(D_{Q,\mathbf{t}}) = Q_{\mathbf{t}}(D)$ can be obtained

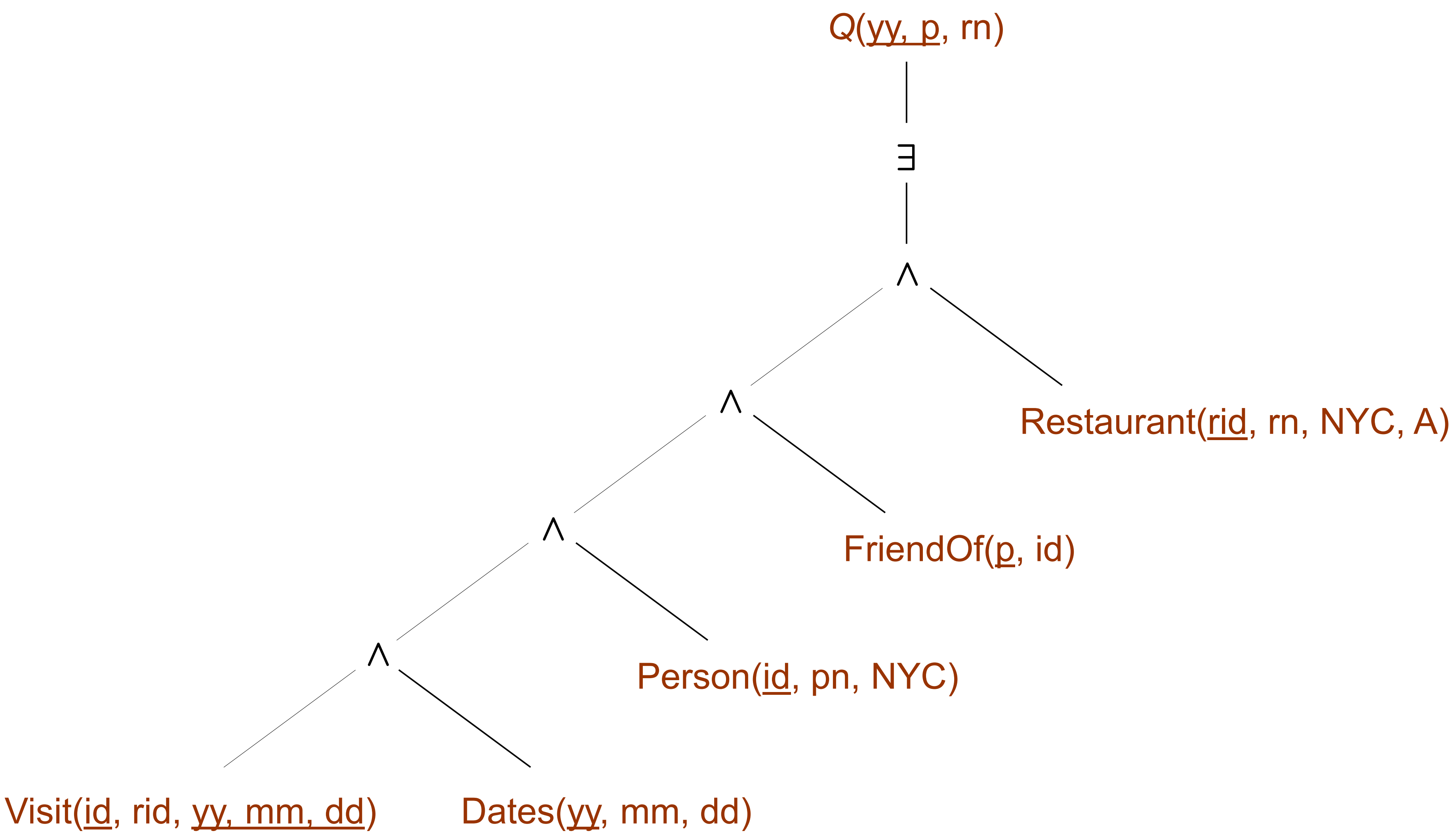
Effective Plan: Example

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

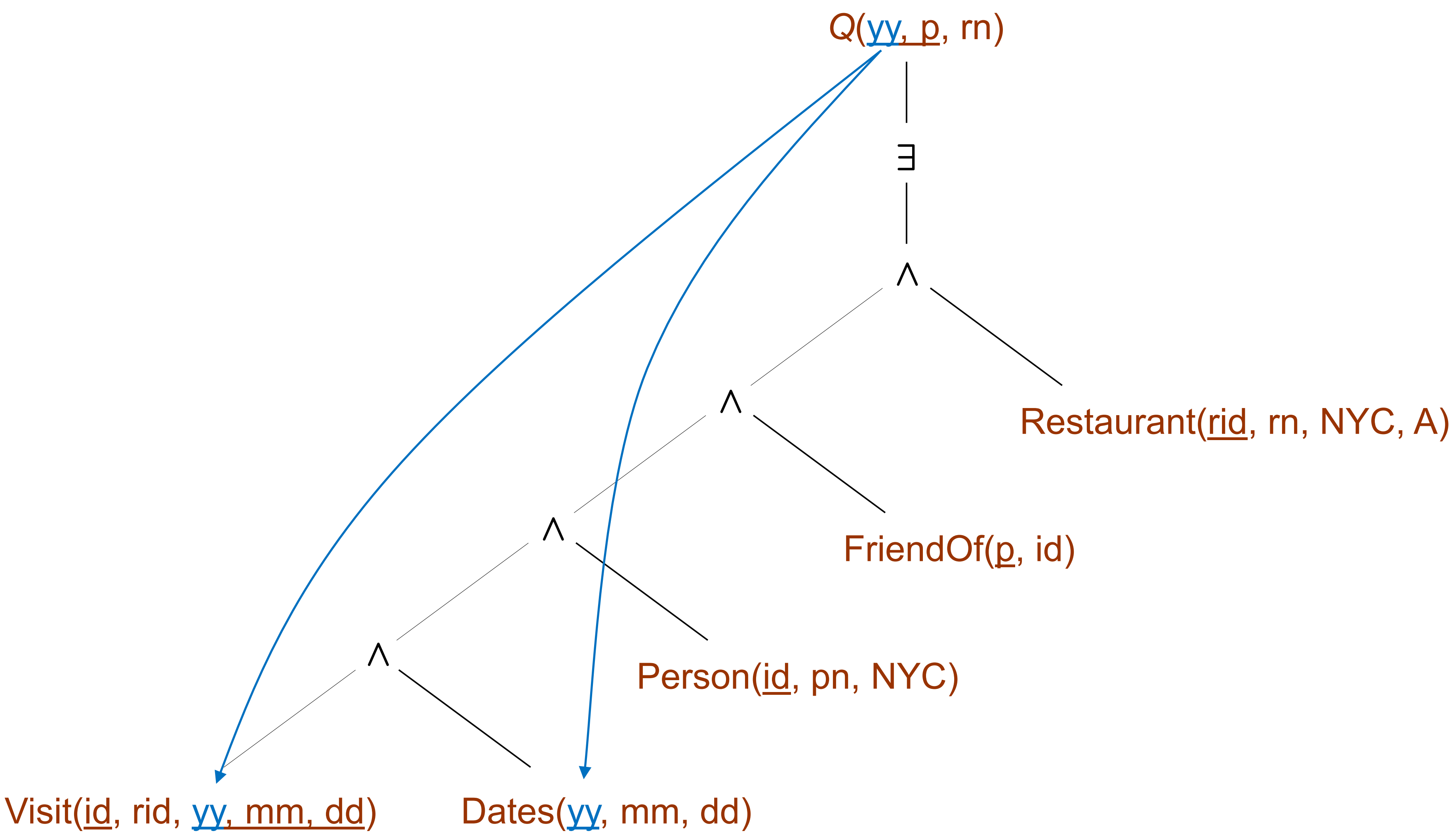
Q {yy,p}-controllable under **A**

Access Schema A
(FriendOf, {p}, 5000, T_1)
(Visit, {id,yy,mm,dd}, 1, T_2)
(Person, {id}, 1, T_3)
(Dates, {yy}, 366, T_4)
(Restaurant, {rid}, 1, T_5)

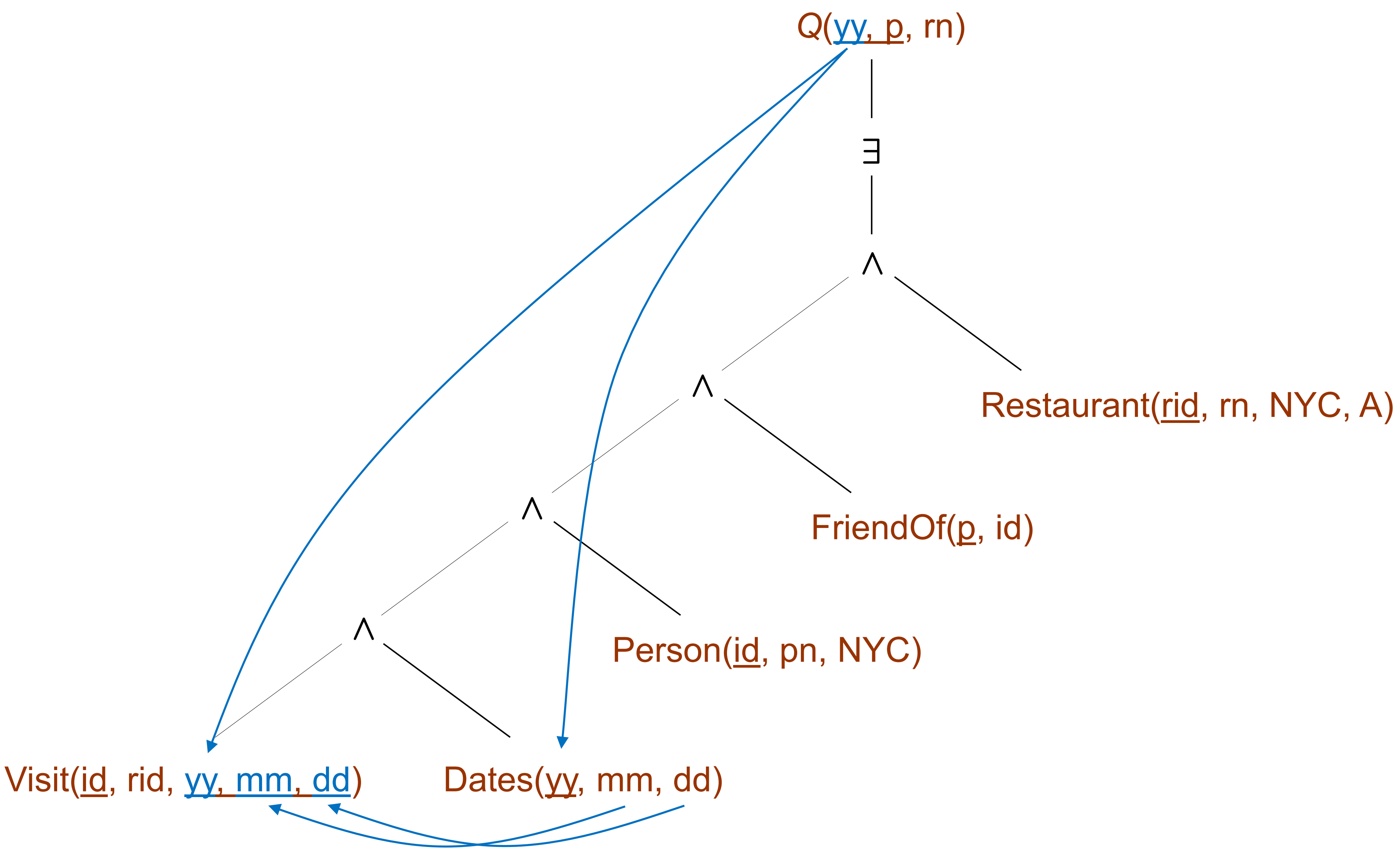
Effective Plan: Example



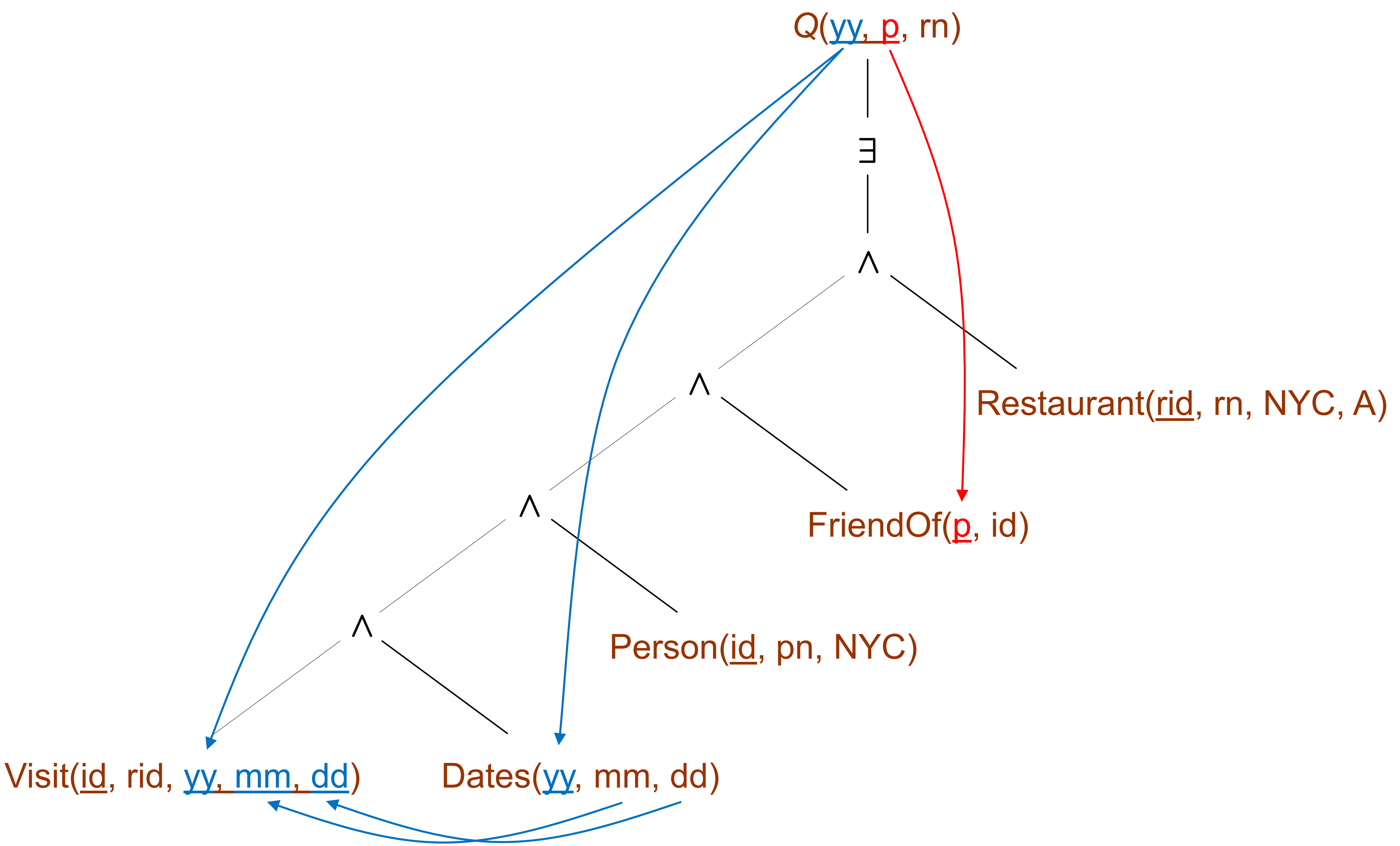
Effective Plan: Example



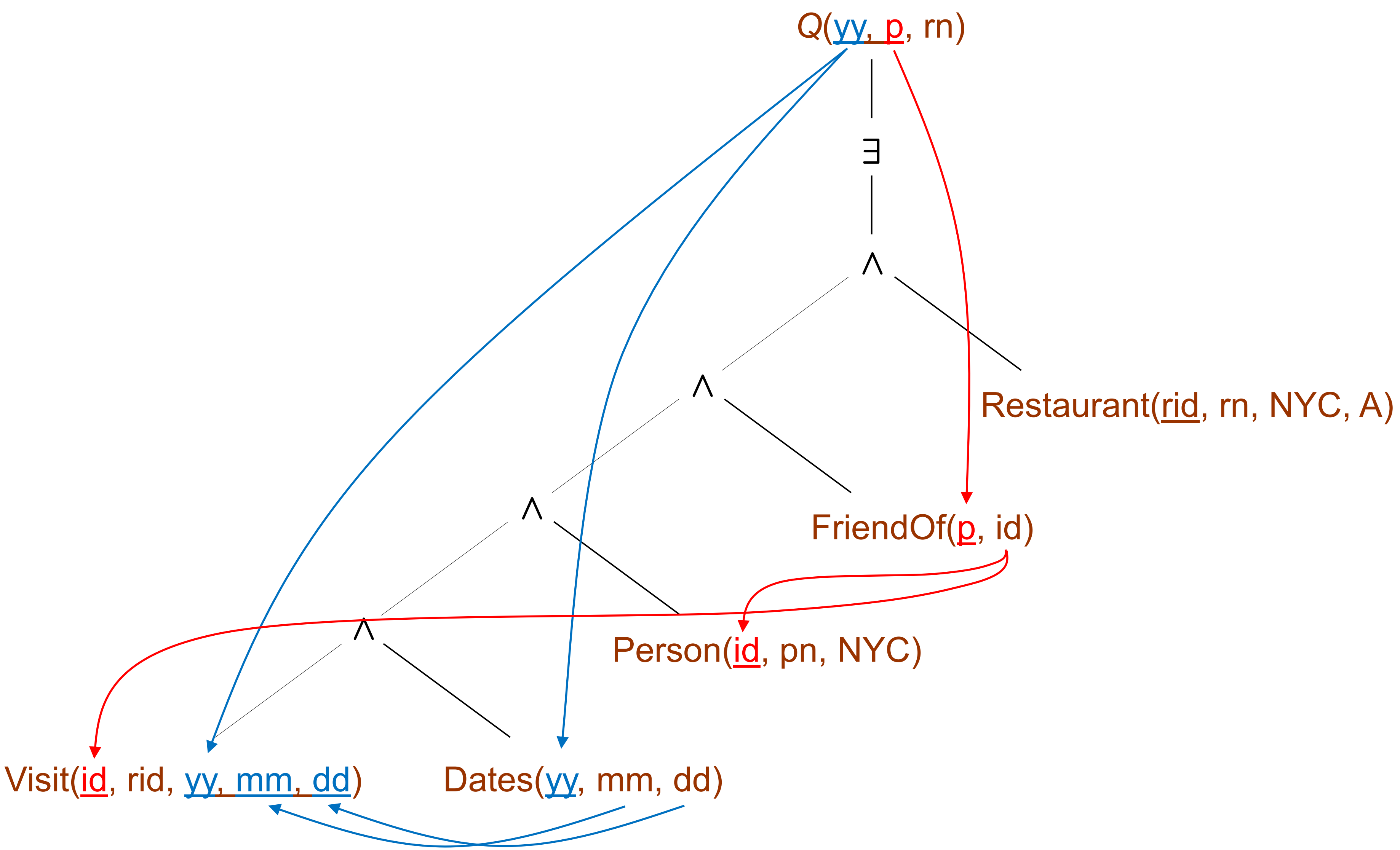
Effective Plan: Example



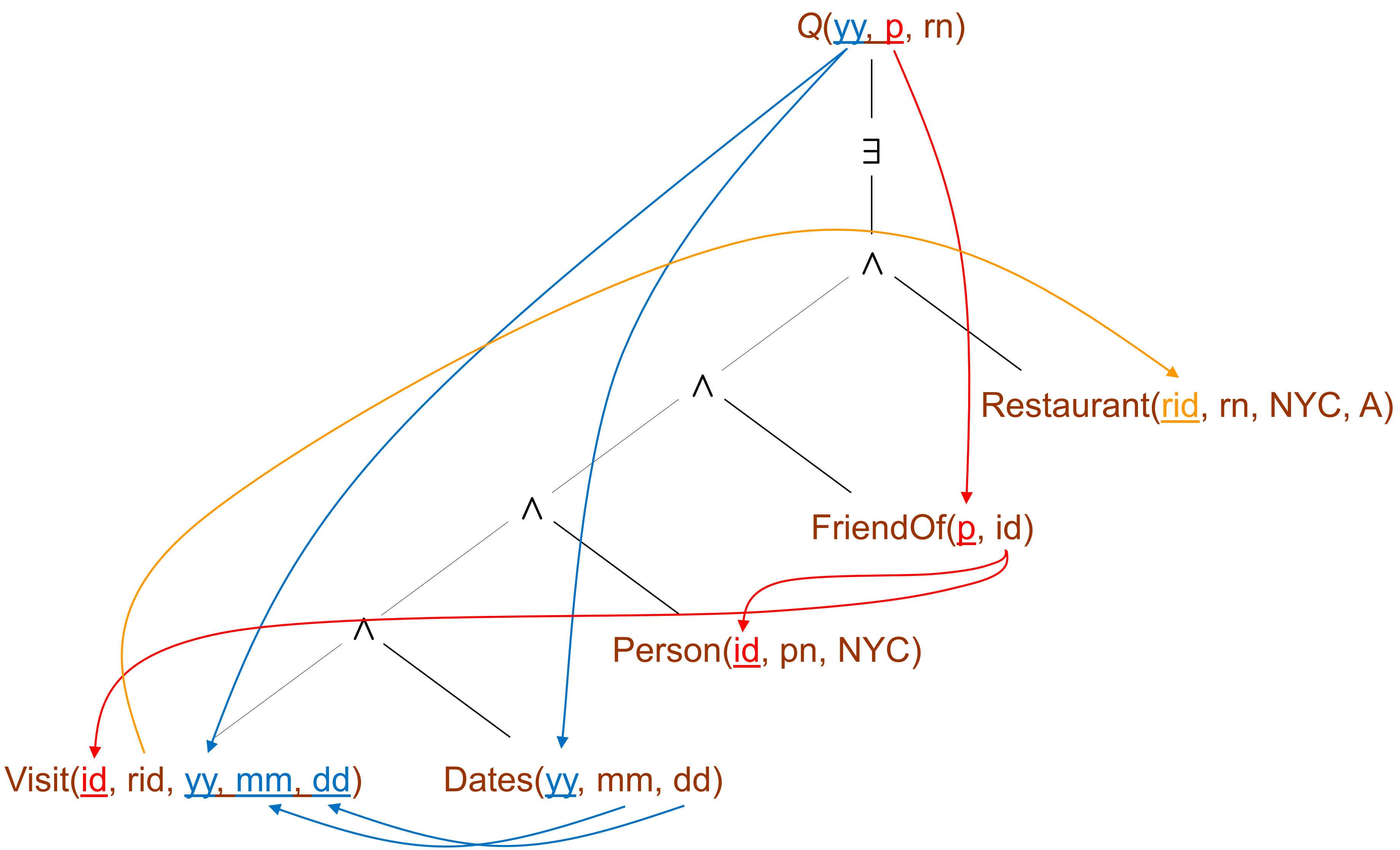
Effective Plan: Example



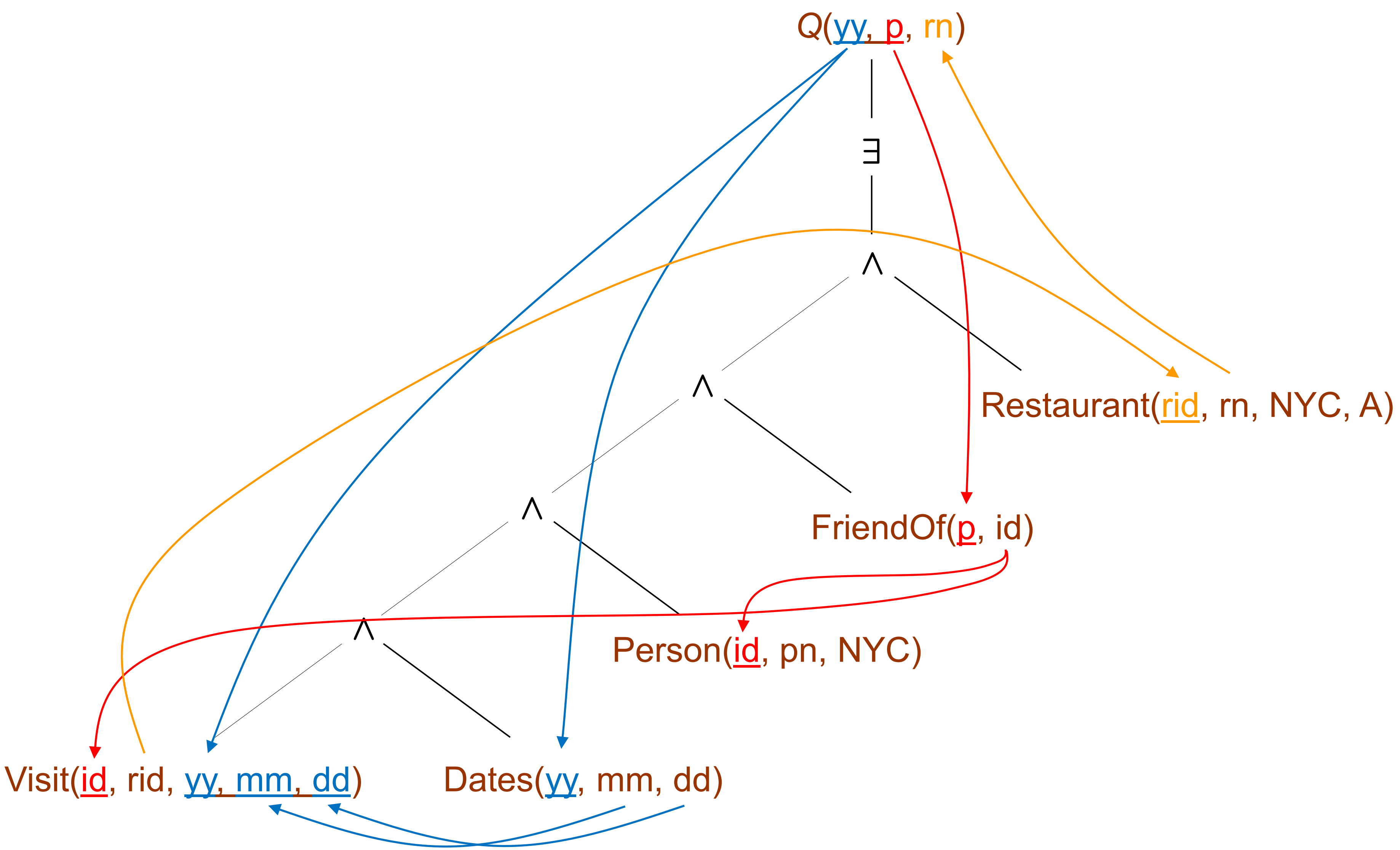
Effective Plan: Example



Effective Plan: Example

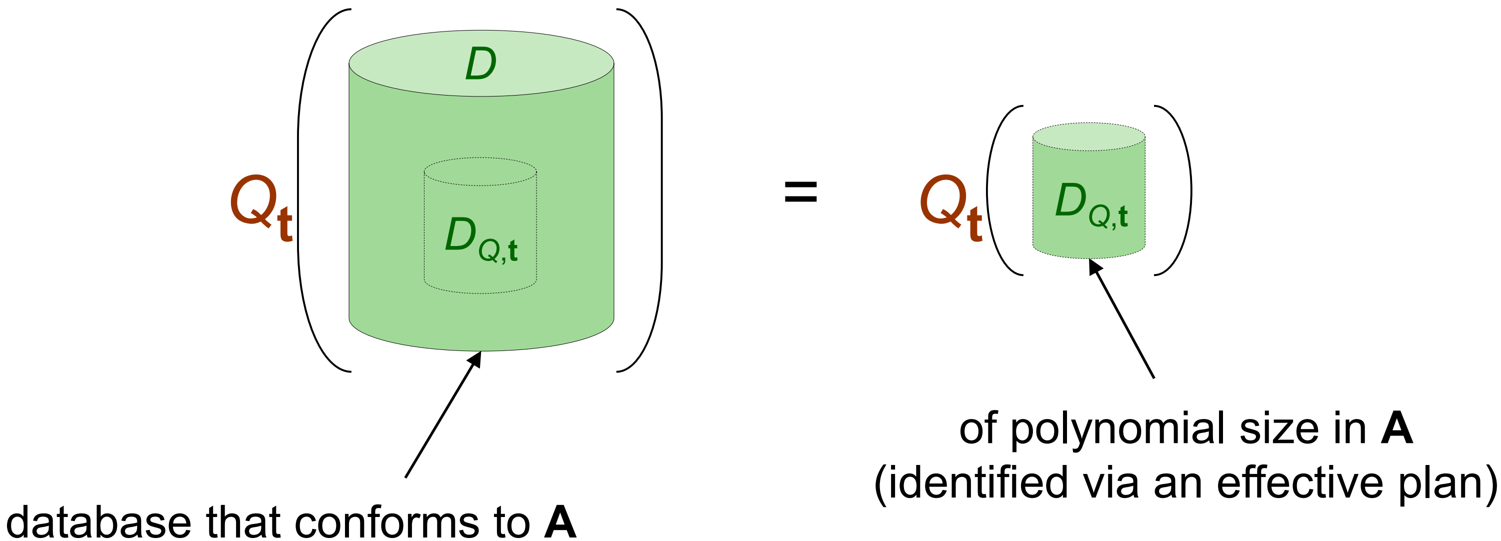


Effective Plan: Example



Wrap-Up

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



- Then, exploit existing database technology to answer $Q_{\mathbf{t}}$ on $D_{Q,\mathbf{t}}$ (and thus on D)

Associated Papers

- Michael Armbrust, Kristal Curtis, Tim Kraska, Armando Fox, Michael J. Franklin, David A. Patterson: PIQL: Success-Tolerant Query Processing in the Cloud. PVLDB 5(3):181-192 (2011)

- Michael Armbrust, Armando Fox, David A. Patterson, Nick Lanham, Beth Trushkowsky, Jesse Trutna, Haruki Oh: SCADS: Scale-Independent Storage for Social Computing Applications. CIDR 2009

  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