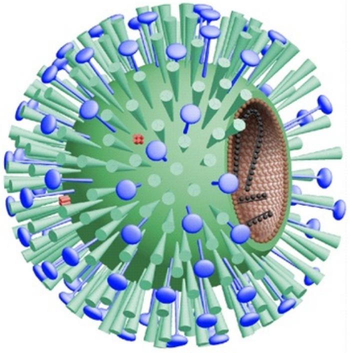
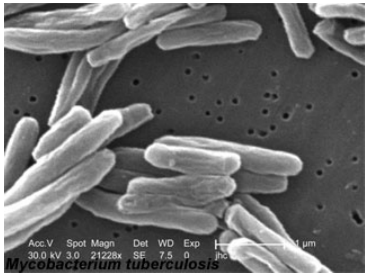
ViP: a User-centric View-Based Annotation Framework for Scientific Data

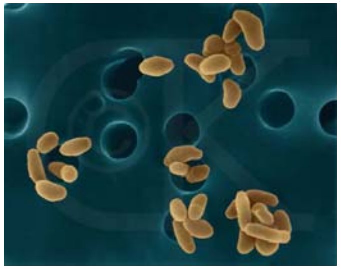
Qinglan Li, Alexandros Labrinidis, Panos K. Chrysanthis

Advanced Data Management Technologies Laboratory
University of Pittsburgh

Center for Modeling Pulmonary Immunity (CMPI)

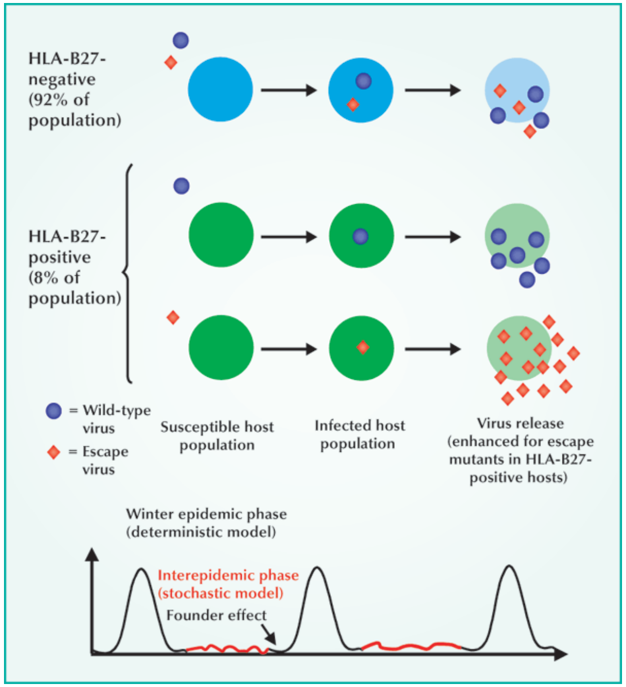



Acc.V Spot Magn Det WD Exp 1 µm
30.0 kV 3.0 21228x SE 7.5 0 jhc
Mycobacterium tuberculosis

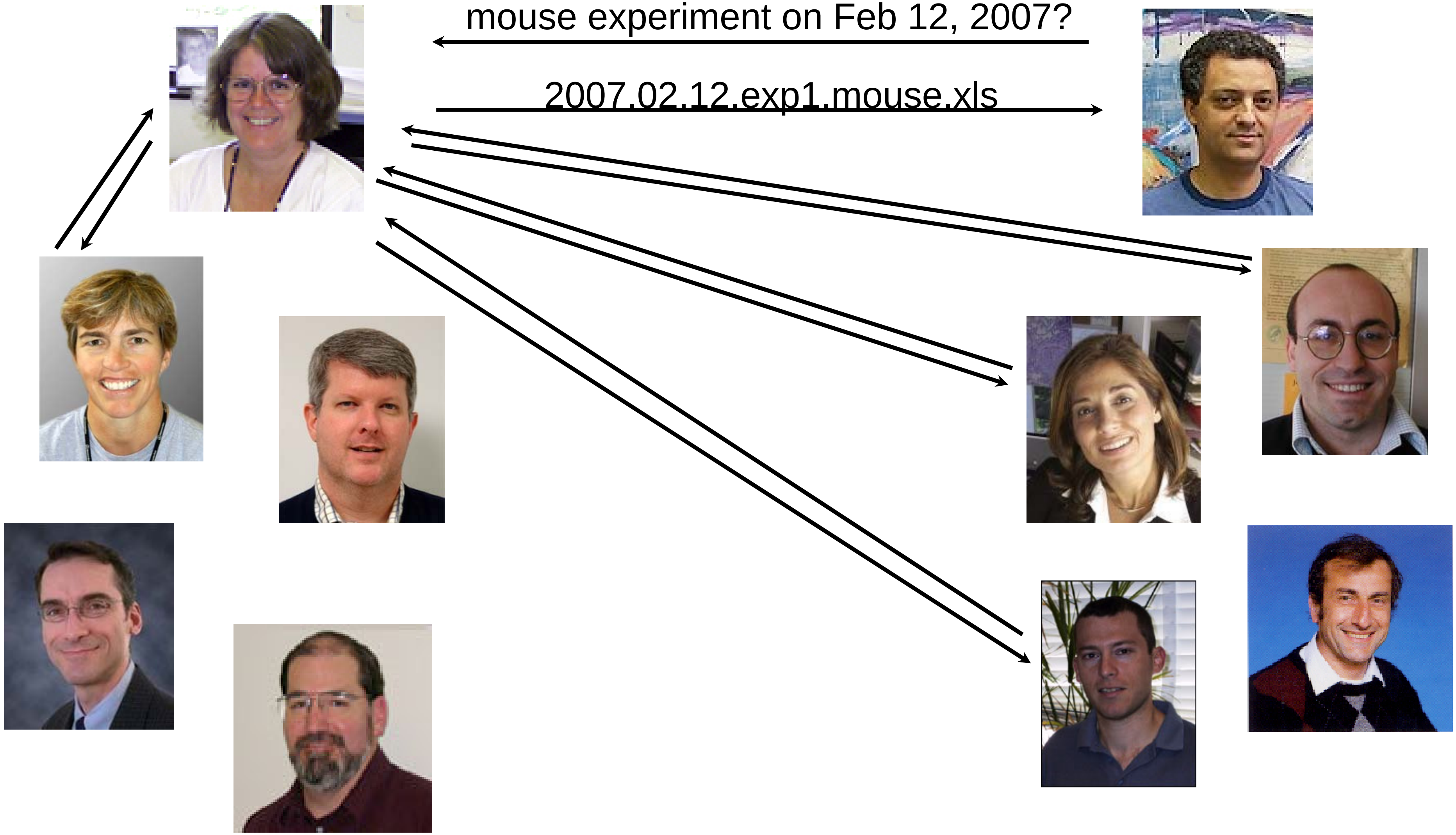


+

=


HLA-B27-negative (92% of population)
HLA-B27-positive (8% of population)
= Wild-type virus
= Escape virus
Susceptible host population
Infected host population
Virus release (enhanced for escape mutants in HLA-B27-positive hosts)
Winter epidemic phase (deterministic model)
Interepidemic phase (stochastic model)
Founder effect

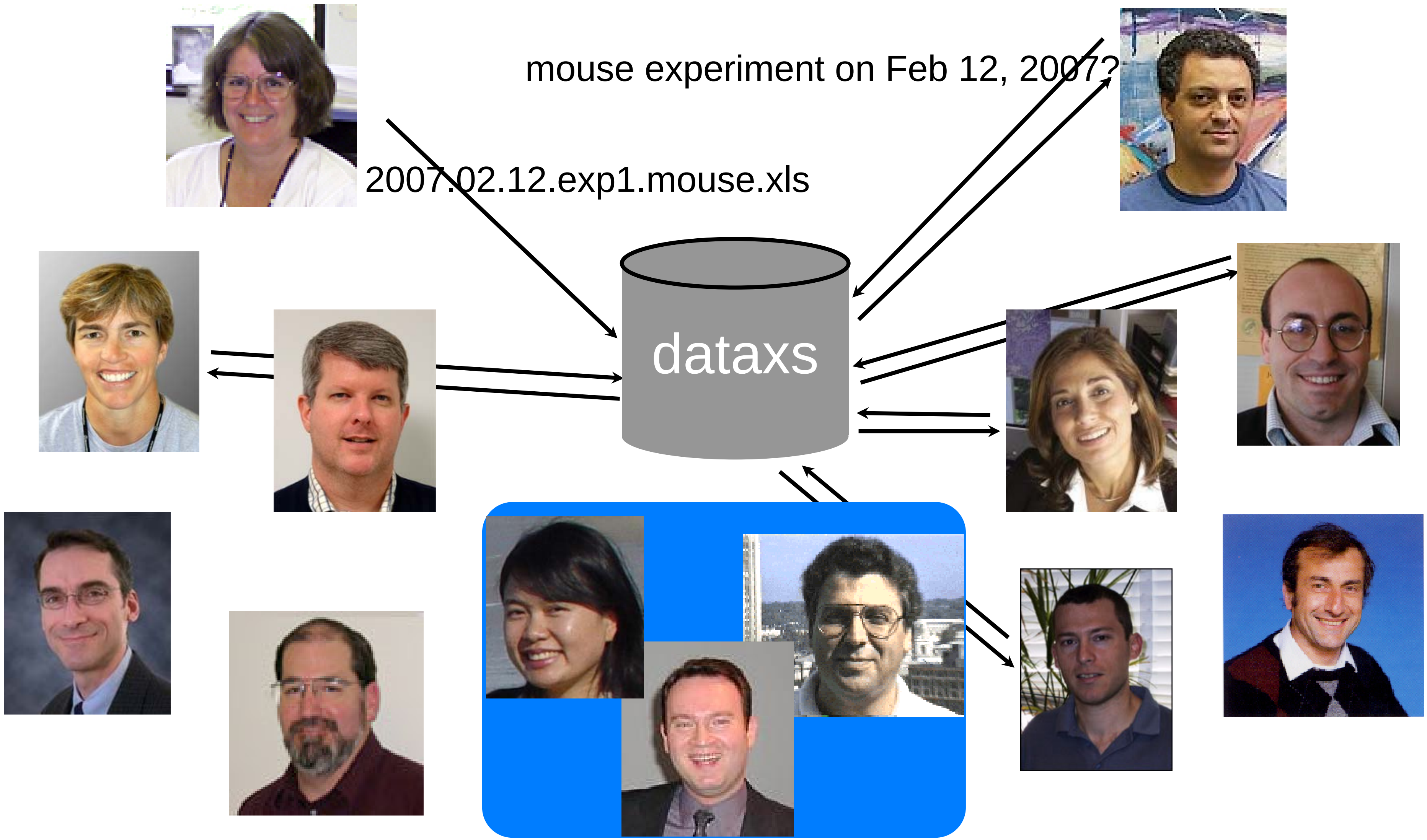
Before



mouse experiment on Feb 12, 2007?

2007.02.12.exp1.mouse.xls

After



mouse experiment on Feb 12, 2007?

2007.02.12.exp1.mouse.xls

dataxs

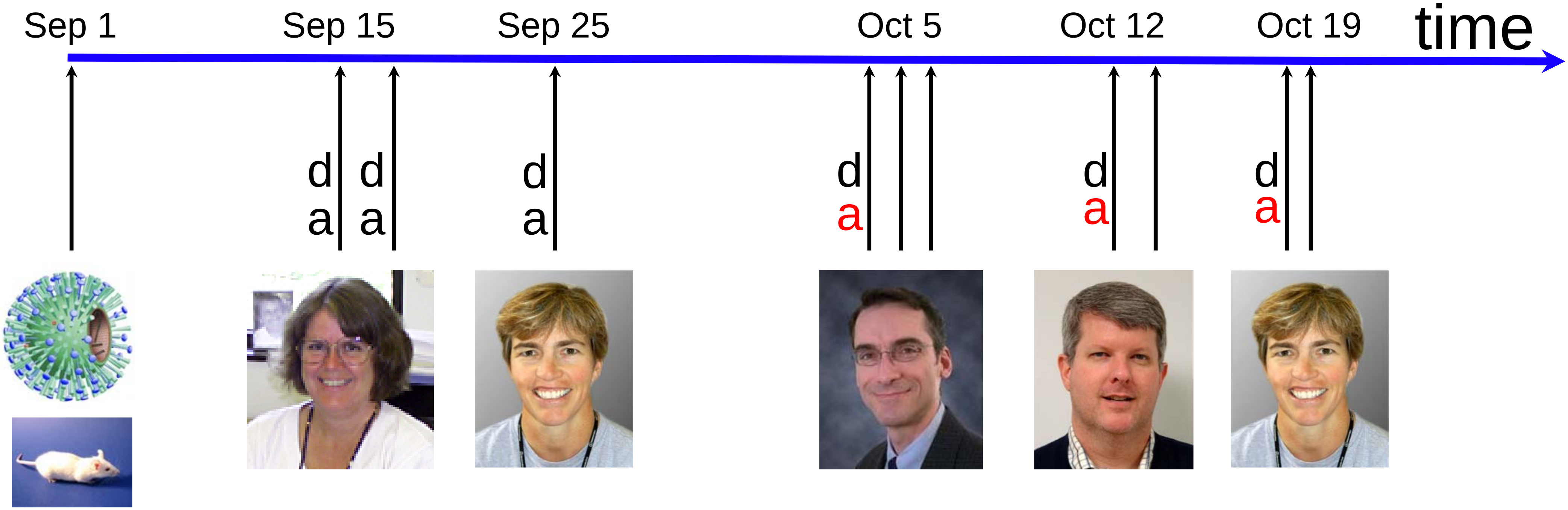
What About Annotations?

Question:
Are there any non-standard forms of annotations and of annotation propagation?

Outline

- Motivation
- Usage Patterns
  - Time Semantics
  - Network Semantics
- User-centric Access Control
- Putting It All Together
- Experimental Evaluation
- Summary

Usage Pattern #1



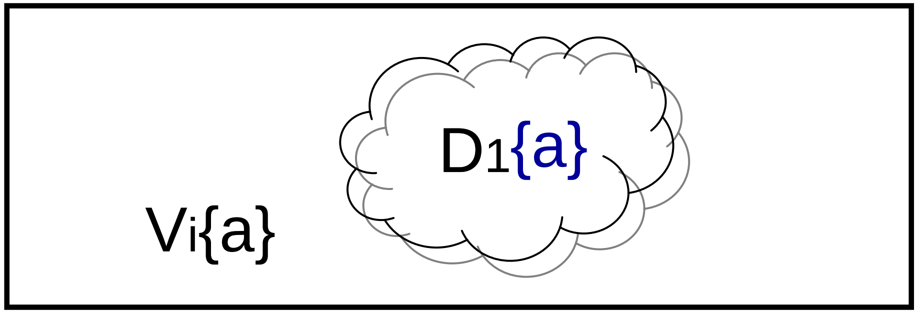
Sep 1 Sep 15 Sep 25 Oct 5 Oct 12 Oct 19 time

d a d a d a d a d a d a

Data is entered asynchronously

=> View-based annotation

User-centric Time Semantics

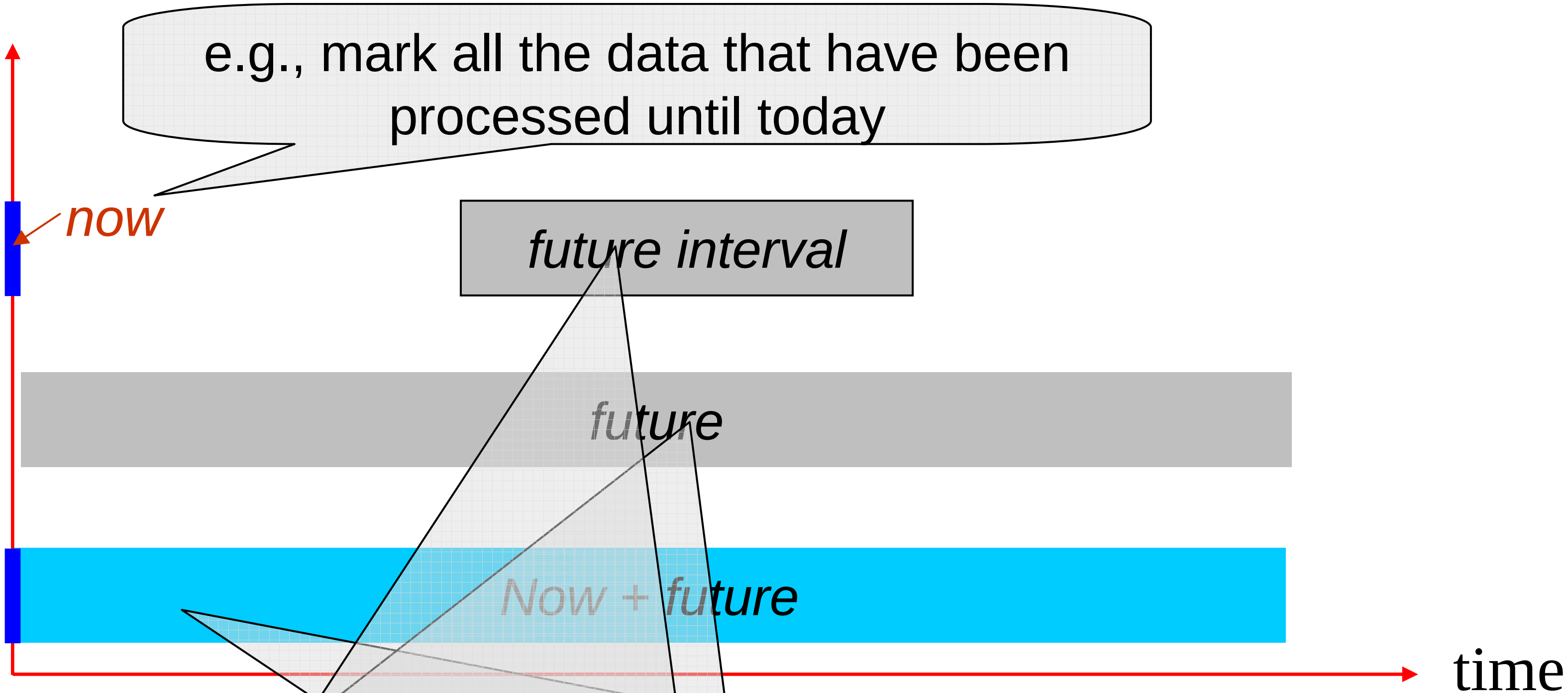


- INSERT (data) into VIEW
  - **D_1 becomes a member of view V_i**
  - **It will be associated with annotation a when it is queried**

- DELETE (data) from VIEW
  - **D_1 is no longer a member of view V_i**
  - **It will not be associated with annotation a**

- DELETE (view)
  - **if V_i is deleted, all the data items that were members of V_i and were associated with a will no longer be associated with it**

User-centric Time Semantics - Annotation View
Valid Time



e.g., mark all the data that have been processed until today

e.g., all the files until today have been fixed, but all files submitted in the future should be marked accordingly

e.g., assume that we have a microarray scanner which was miscalibrated on a certain day; when this is first discovered, we want to be able to annotate all experimental data in the system accordingly, but also do this for all data that would fall in this category, but are entered in the system later

e.g., for one week after the Daylight Saving Time show an annotation that reminds scientists to make sure they have accounted for Daylight Saving Time in experiment settings

Usage Pattern #2

Assays (e.g., luminex) are expensive

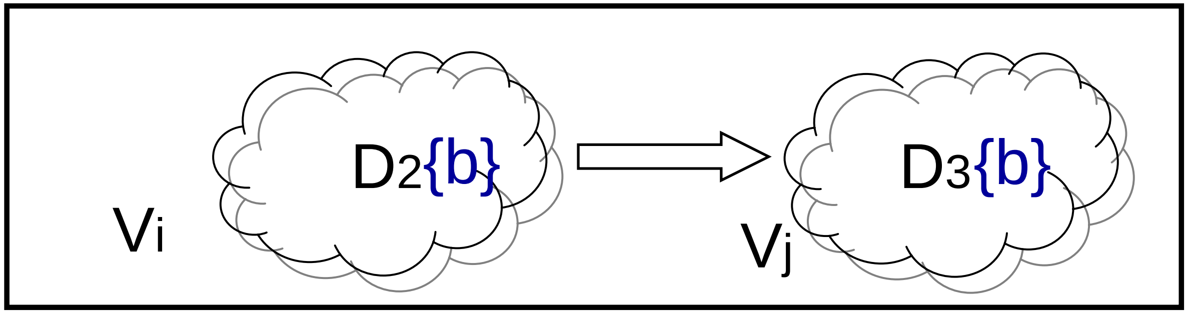
Makes fiscal sense to fully utilize plates

Usage Pattern #2 - Sample Luminex Plate

it appears that: unrelated experimental data are grouped together

therefore: must allow explicit annotation propagation

User-centric Network Semantics

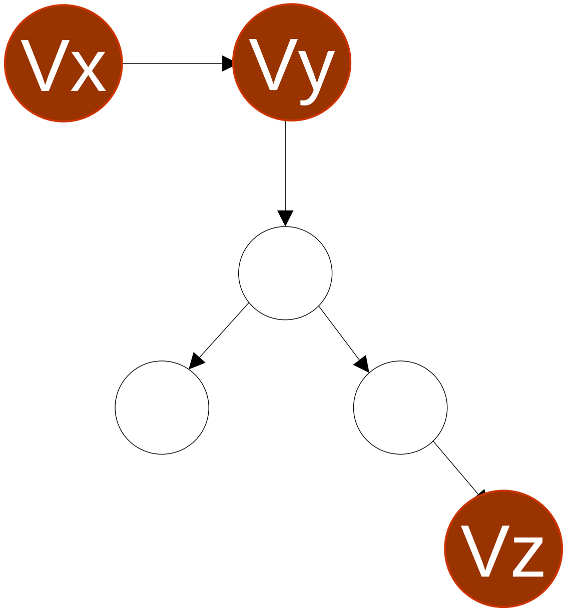


- Given a source view, Vi
- Given a destination view, Vj
- An explicit annotation propagation path Vi → Vj, any annotation that is added in a member of Vi must be propagated to all members of Vj

Annotations are propagated over not only existing implicit annotation propagation paths between source data and derived data (i.e., driven by the database schema and data transformations), but also over explicit paths

Example of User-centric Network Semantics

- Transitivity property forms networks out of explicit annotation propagation paths
- **Question**: should we allow unlimited propagation of annotations?
- **Answer**: let the user decide
  - Inspired by the TTL value of queries in unstructured peer-to-peer networks
  - Inspired by personalization work
  - Annotations will be visible differently for different users



User-centric Network Semantics - HAP

- **HAP on insert**: users can specify a variable, **HAP-i**, or *Hops Allowed to Propagate* at insertion, to indicate how far the newly-inserted annotation can be propagated
- **HAP on query**: users have the option to specify a *maximum number of hops an annotation is allowed to propagate* at query time, or **HAP-q**
- **Maximum HAP**: a system variable, **MAX-HAP**, *maximum number of hops allowed to propagate*, which puts a system-wide upper bound over how many hops any annotation is allowed to propagate
- The *maximum number of hops followed* is
- MIN(MAX-HAP, HAP-i, HAP-q)

Outline

- Motivation
- Usage Patterns
  - Time Semantics
  - Network Semantics
- User-centric Access Control
- Putting It All Together
- Experimental Evaluation
- Summary

User-centric Access Control - Motivation

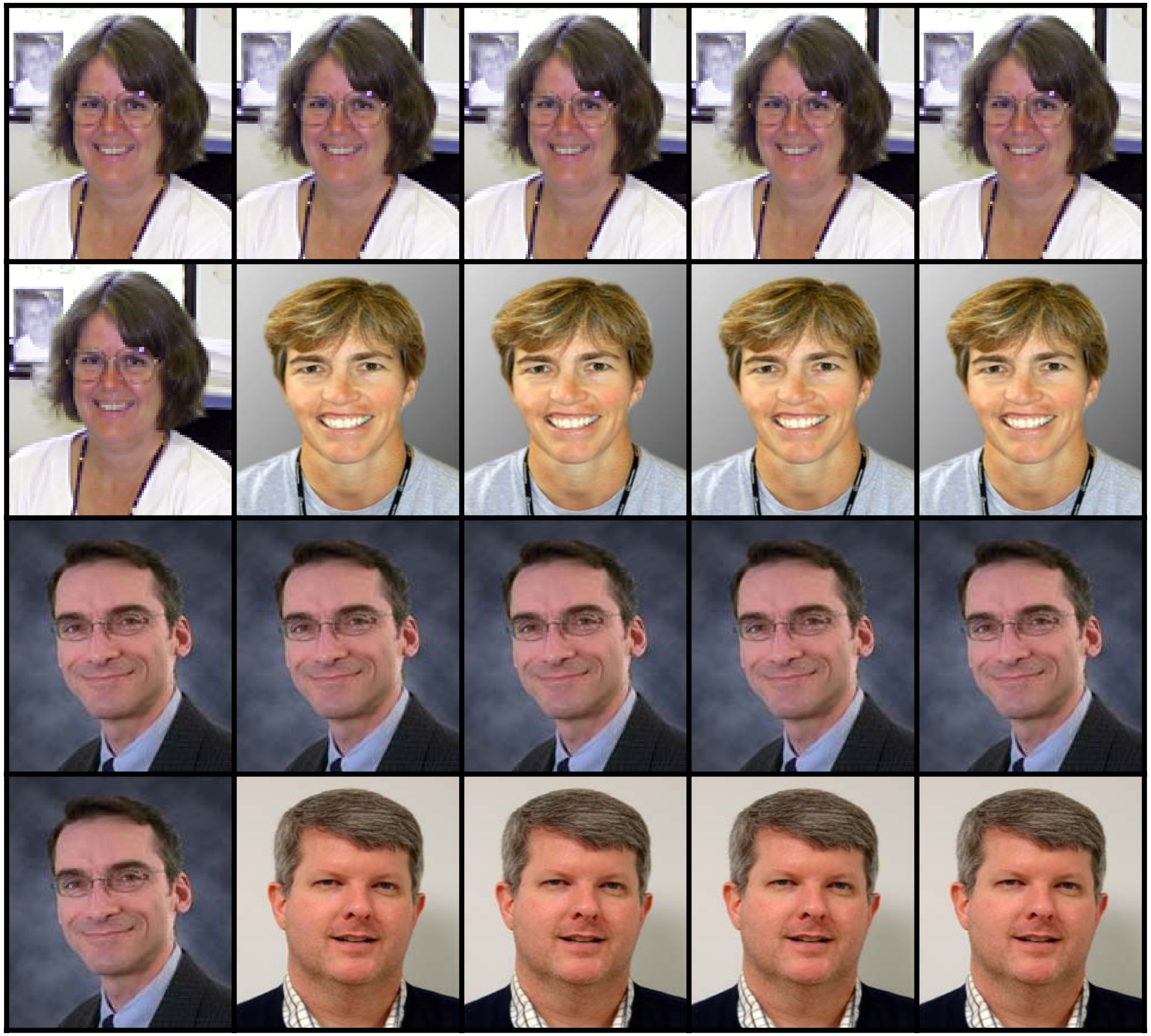
Private

vs

Public

data and

annotations



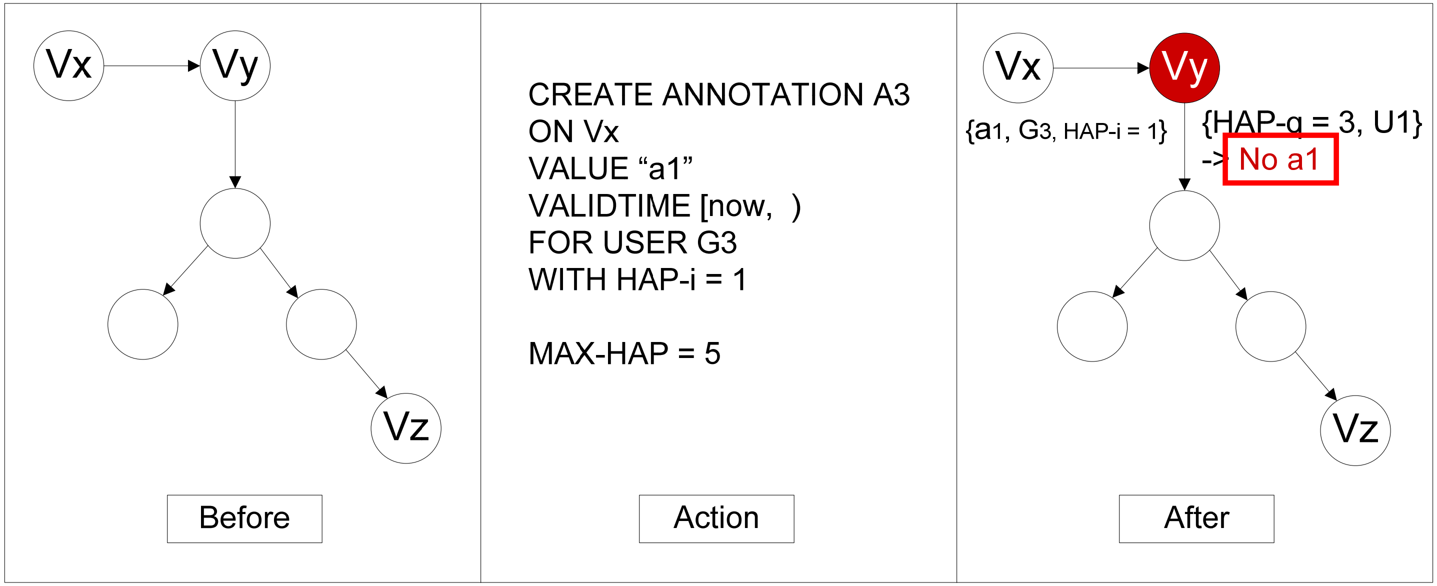
User-centric Access Control

- On the annotation level
  - Implement access control at the level of individual annotations
  - When an individual data item receives an annotation from a user, the user can specify who can access the annotation
  - Support arbitrary user hierarchies
- On the annotation view and path levels
  - Expect the majority of annotations to happen through views
  - Access controls are also implemented
- Different than traditional access control
  - Essentially means who is allowed to “execute” the annotation propagation mechanism and not who is allowed to see the data

Outline

- Motivation
- Usage Patterns
  - Time Semantics
  - Network Semantics
- User-centric Access Control
- Putting It All Together
- Experimental Evaluation
- Summary

Putting It All Together – Query #1



CREATE ANNOTATION A3
ON Vx
VALUE “a1”
VALIDTIME [now,)
FOR USER G3
WITH HAP-i = 1

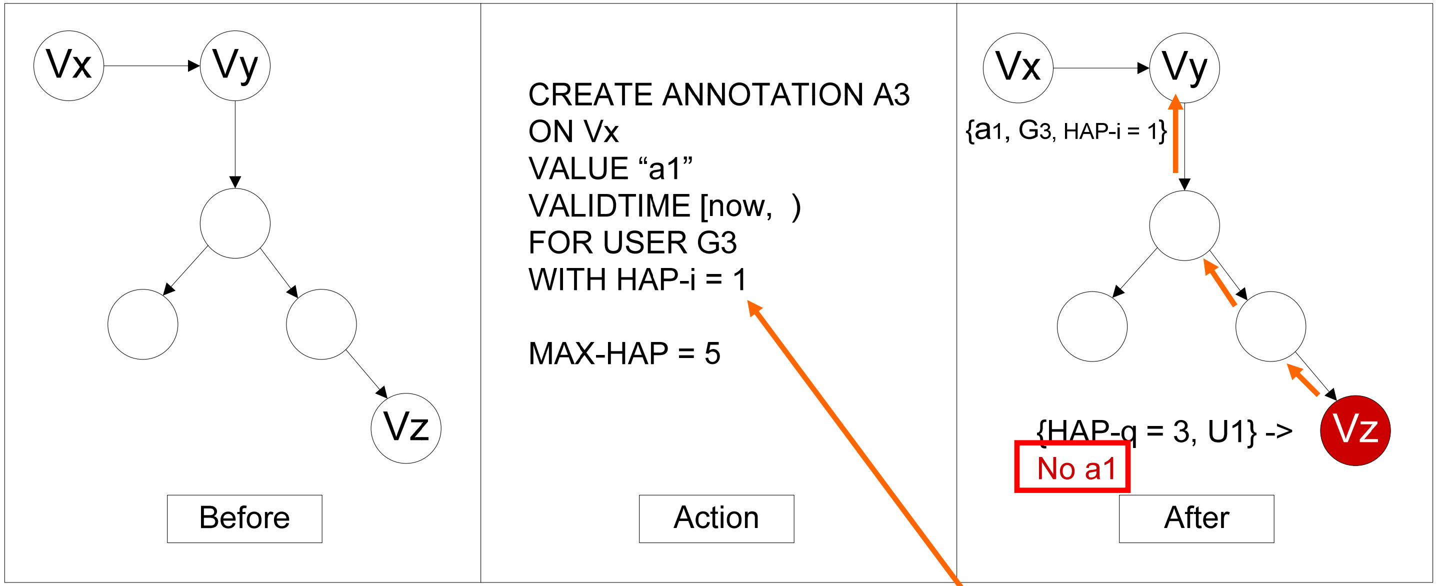
MAX-HAP = 5

Action

Query	Result	User	HAP-q	Annotation
#1	Vy	U1 ∉G3	3	No a1

U1 is not in the user group G3

Putting It All Together – Query #2



CREATE ANNOTATION A3
ON Vx
VALUE “a1”
VALIDTIME [now,)
FOR USER G3
WITH HAP-i = 1

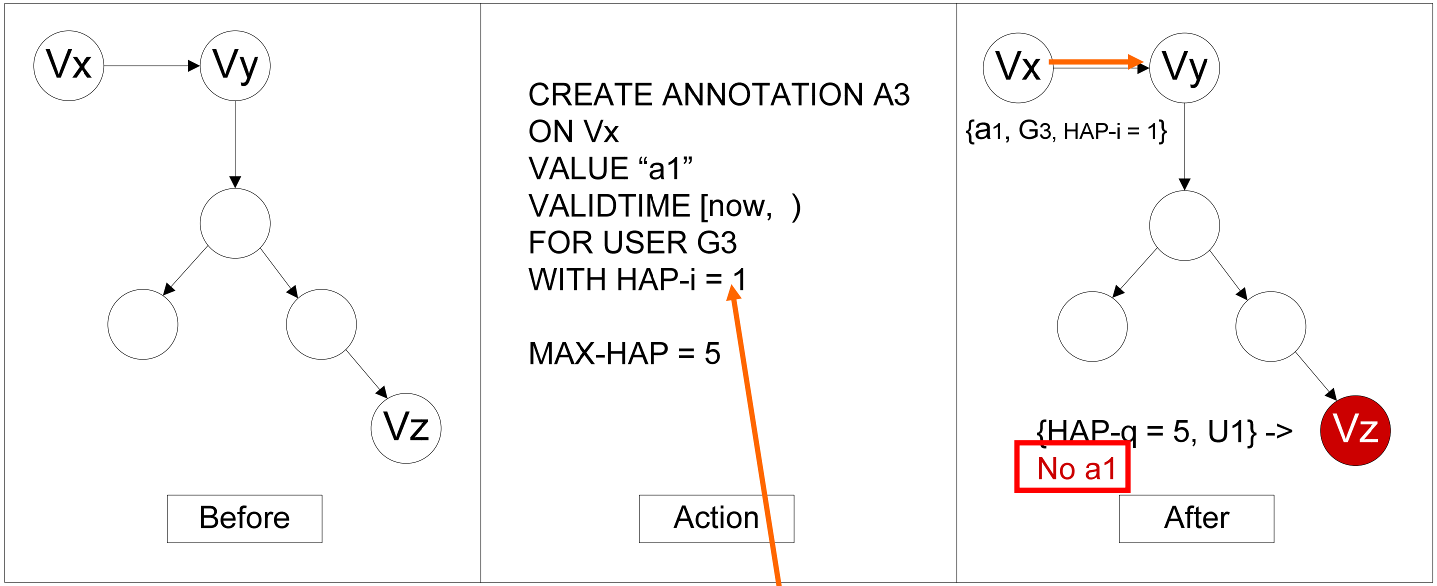
MAX-HAP = 5

Query	Result	User	HAP-q	Annotation
#2	Vz	U1 $\in$ G3	3	No a1

Same user group

HAP-q and HAP-i are not big enough

Putting It All Together – Query #3



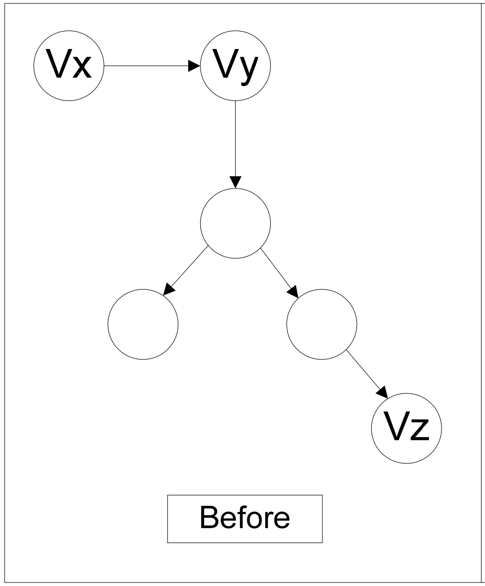
CREATE ANNOTATION A3
ON Vx
VALUE “a1”
VALIDTIME [now,)
FOR USER G3
WITH HAP-i = 1

MAX-HAP = 5

Query	Result	User	HAP-q	Annotation
#3	Vz	U1 ∈ G3	5	No a1

HAP-i is not long enough

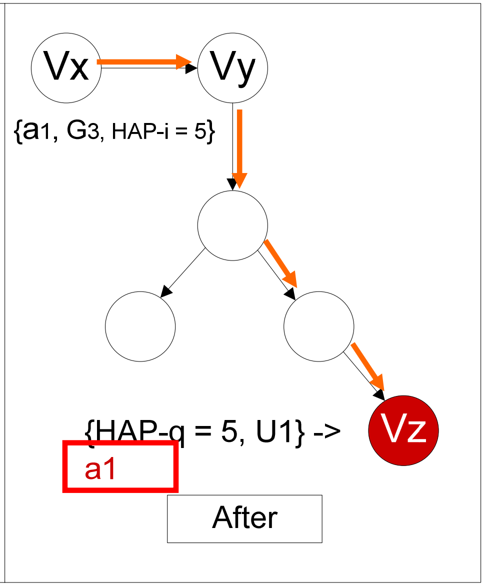
Putting It All Together – Query #4



```
CREATE ANNOTATION A3
ON Vx
VALUE "a1"
VALIDTIME [now,  )
FOR USER G3
WITH HAP-i = MAX-HAP

MAX-HAP = 5
```

Action



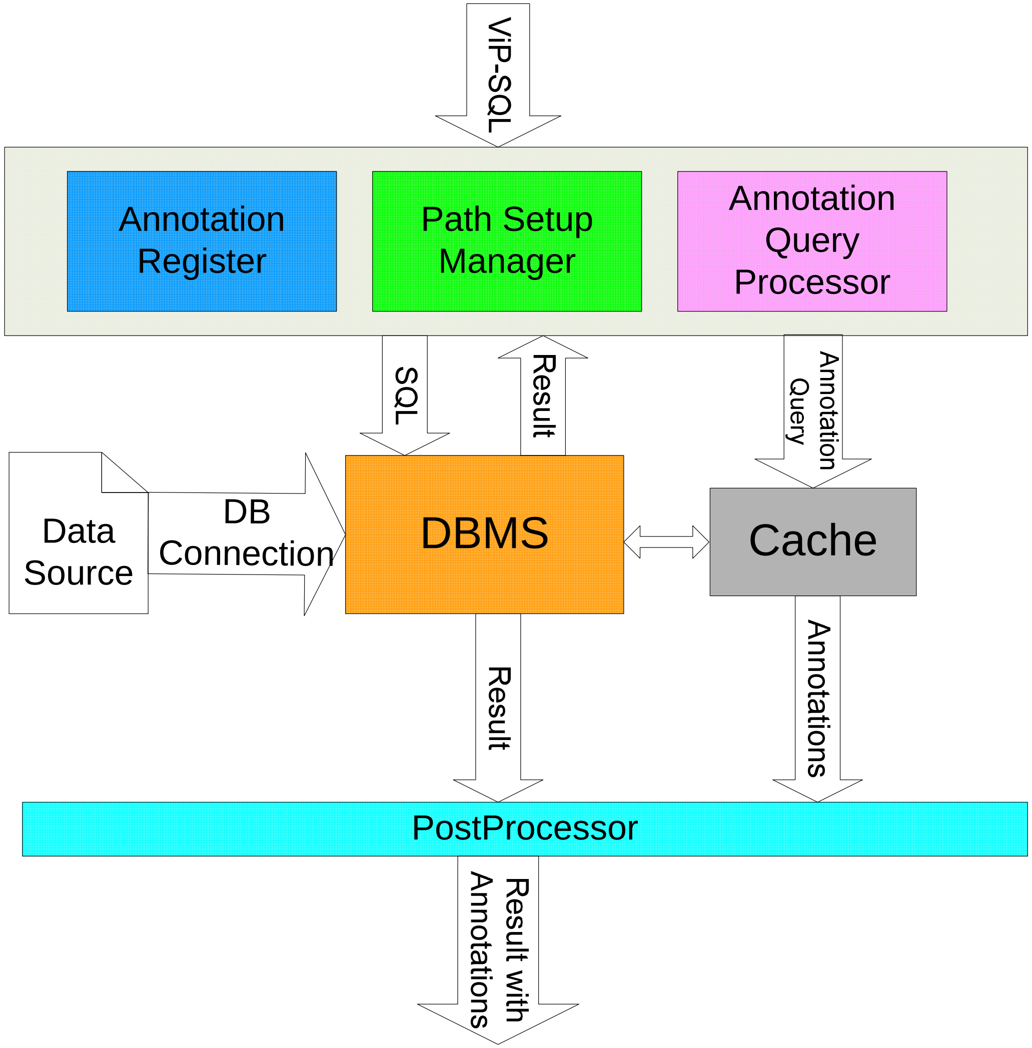
Query	Result	User	HAP-q	Annotation
#4	Vz	U1 ∈ G3	5	a1!

HAP-i is 5 now

Outline

- Motivation
- Usage Patterns
  - Time Semantics
  - Network Semantics
- User-centric Access Control
- Putting It All Together
- Experimental Evaluation
- Summary

System Architecture



- Annotation view - annotation register

- Annotation path - path setup manager updates the auxiliary table to record path source and target

- Caching is used to improve the query time

- Data items with its associated annotations are stored in the cache

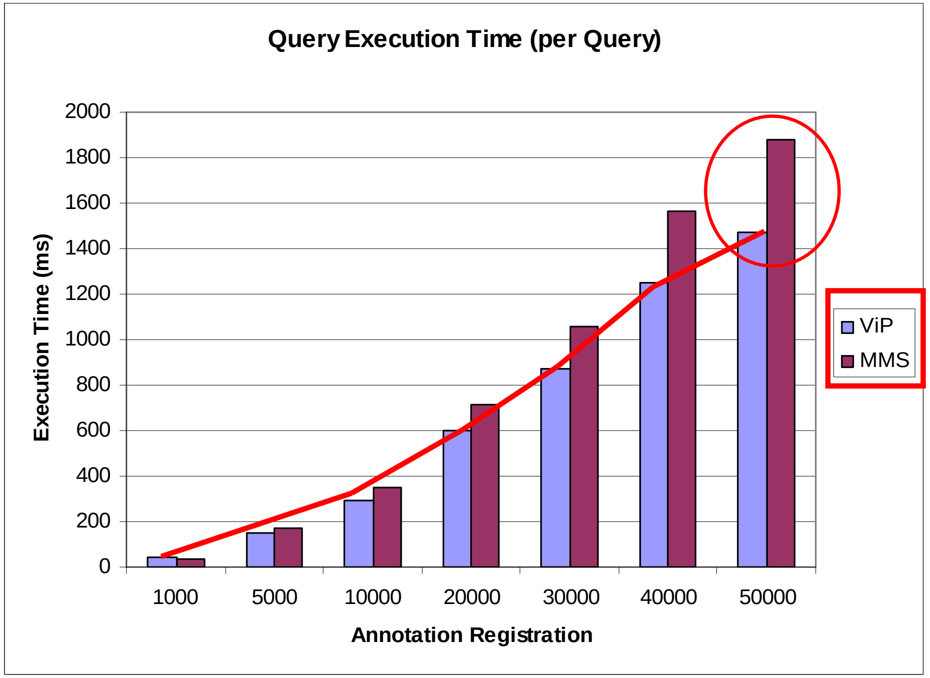
Experimental Evaluation

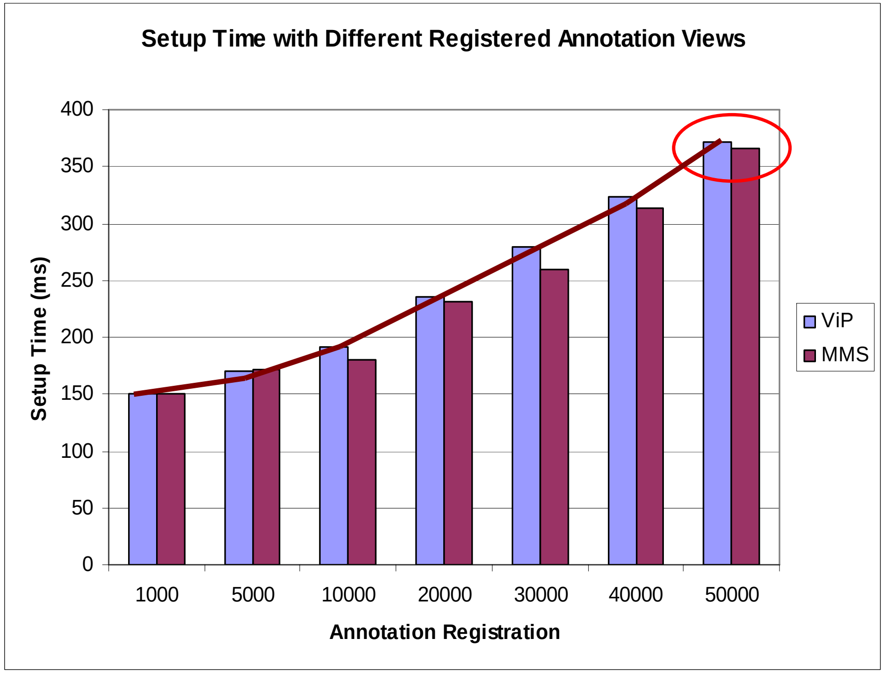
- Real system implementation – ViP Framework on DataXS
- Simulated workload, to stress-test the system
- Test all semantics
- Performance and features
- Compared with MMS system [D. Srivastava and Y. Velegrakis, SIGMOD’07]

Experiment Parameters

Parameters	Value Range
Number of Data tuples	300,000
Number of Annotation views	1 – 50,000
Number of Annotation paths	1 – 2,500
Number of Queries	1,000
Number of Users	1 – 100
Path Depth	1 – 10

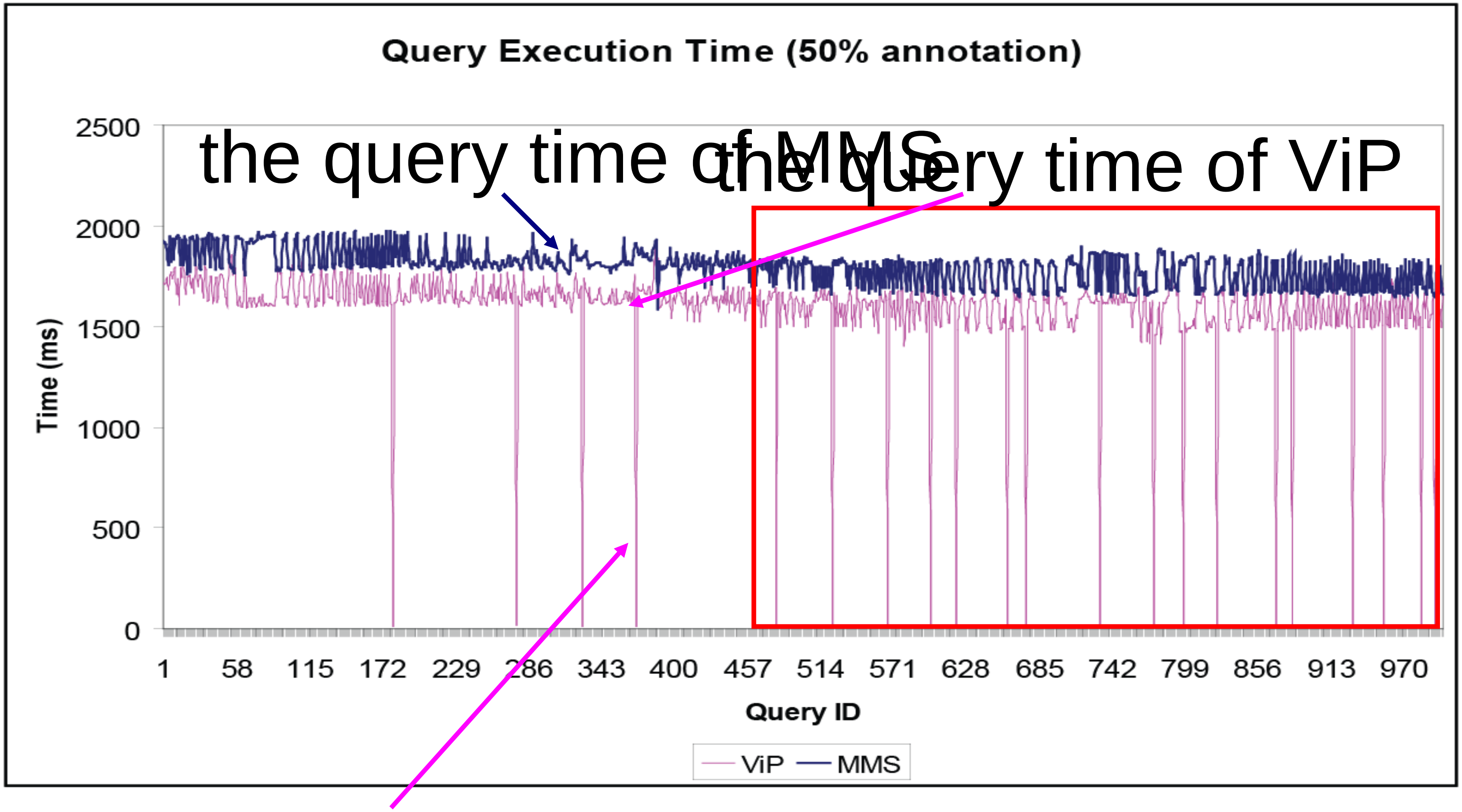
Query Execution Time





- ViP compares with MMS
- MMS outperforms other systems
- Use caching to optimize average annotation query time

Query Time with 50% Annotation Density

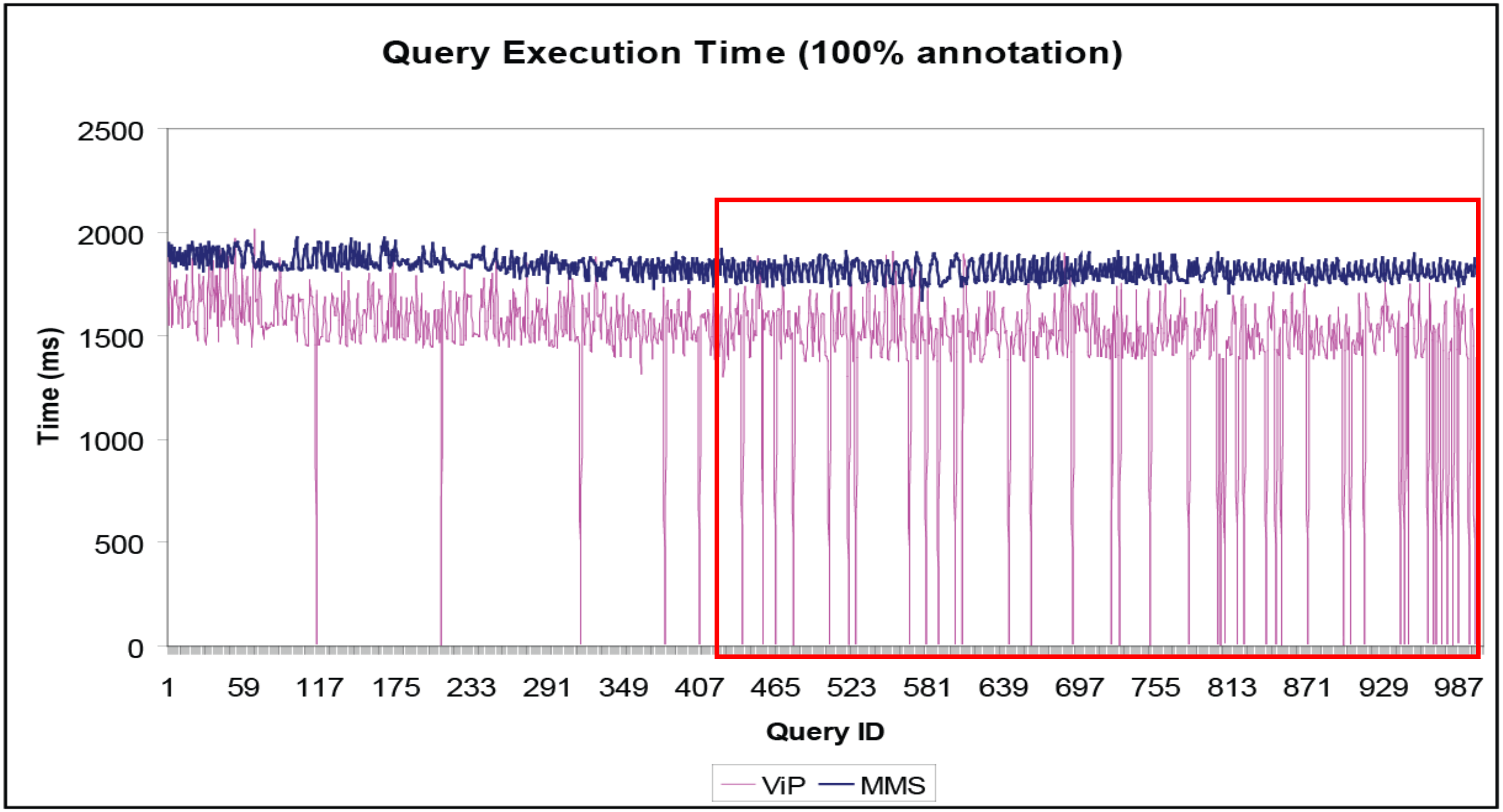


the query time of MMS

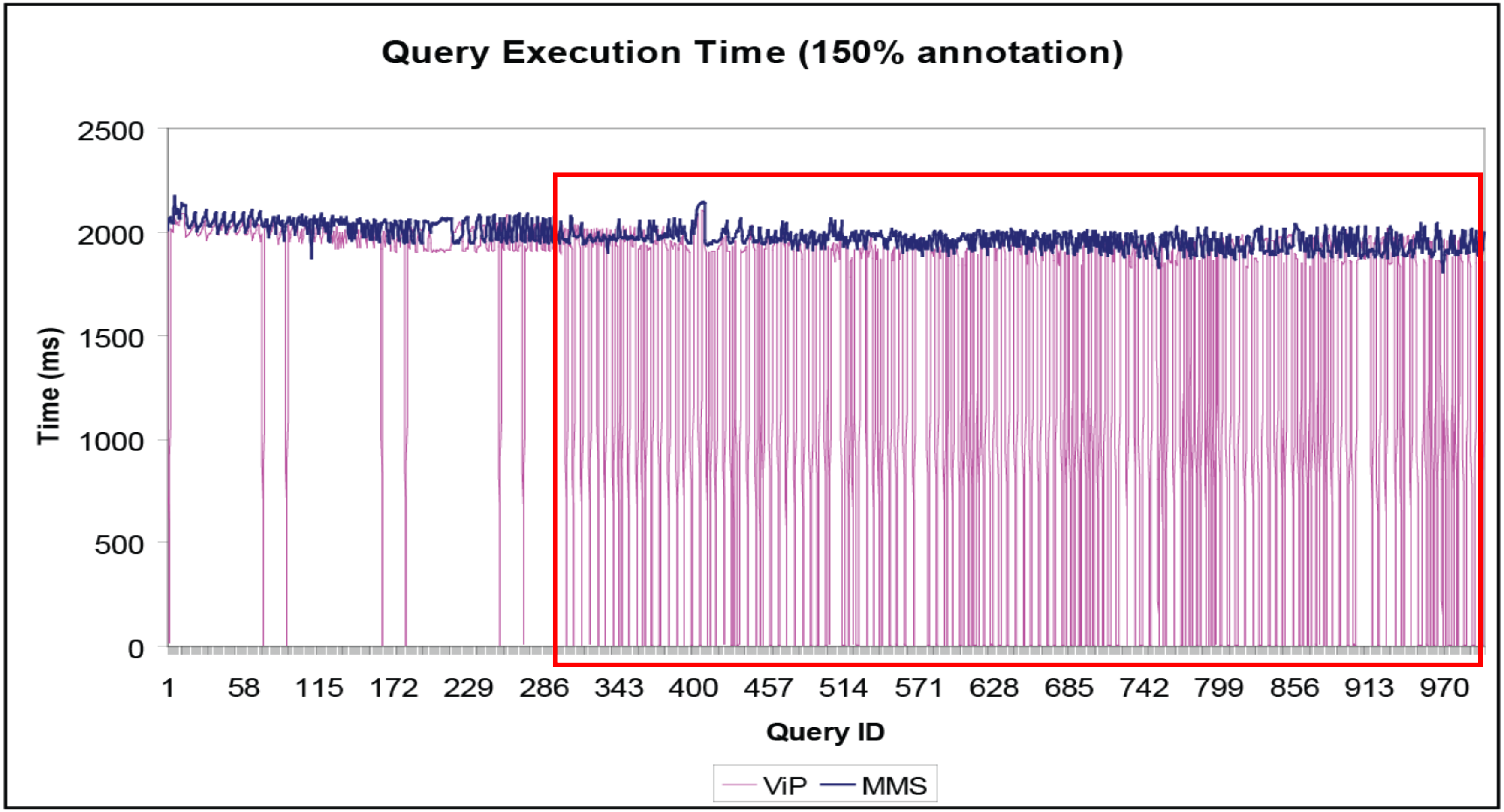
the query time of ViP

Each vertical line is a cache hit

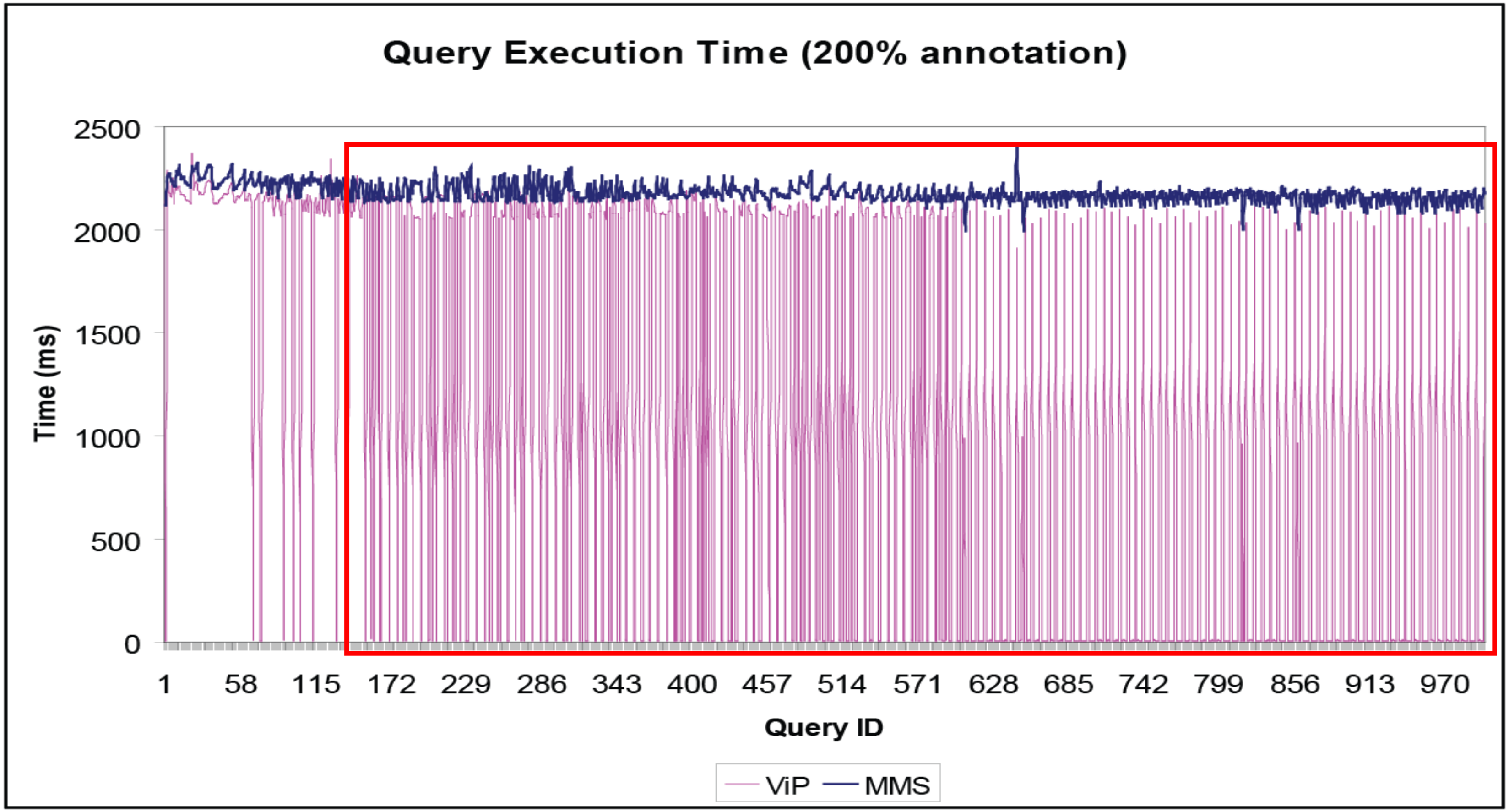
Query Time with 100% Annotation Density



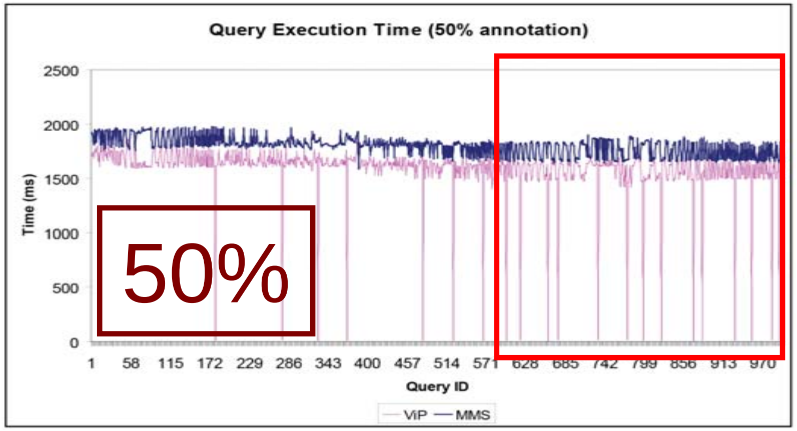
Query Time with 150% Annotation Density

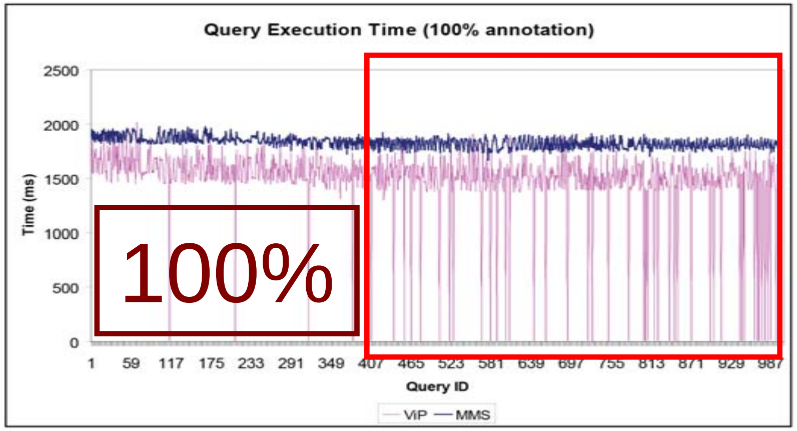


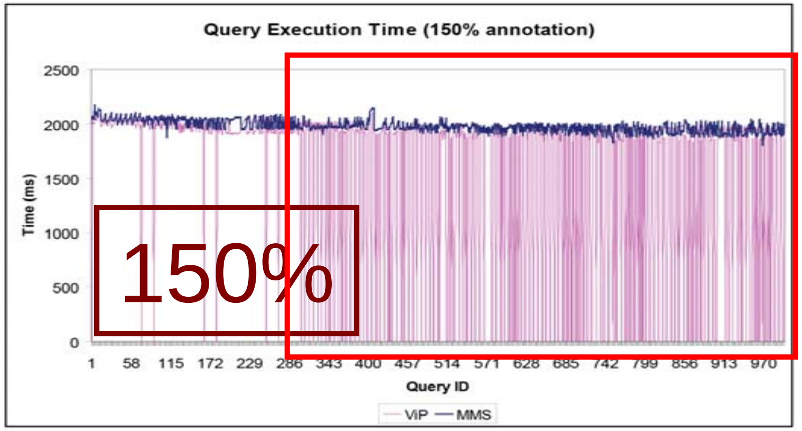
Query Time with 200% Annotation Density

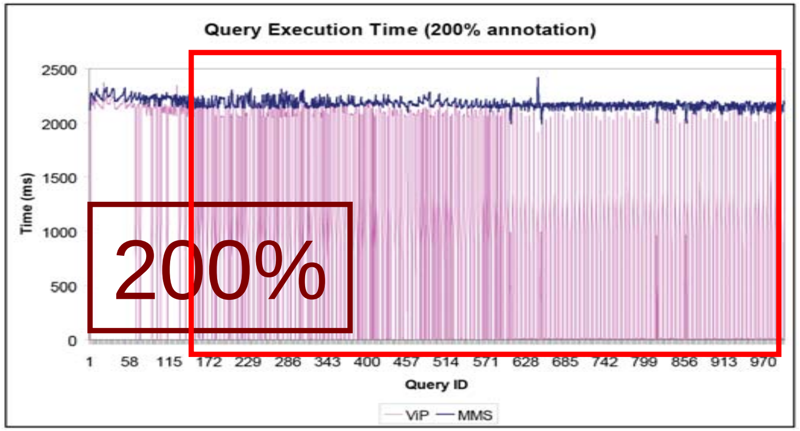


Query Time with Different Annotation Densities



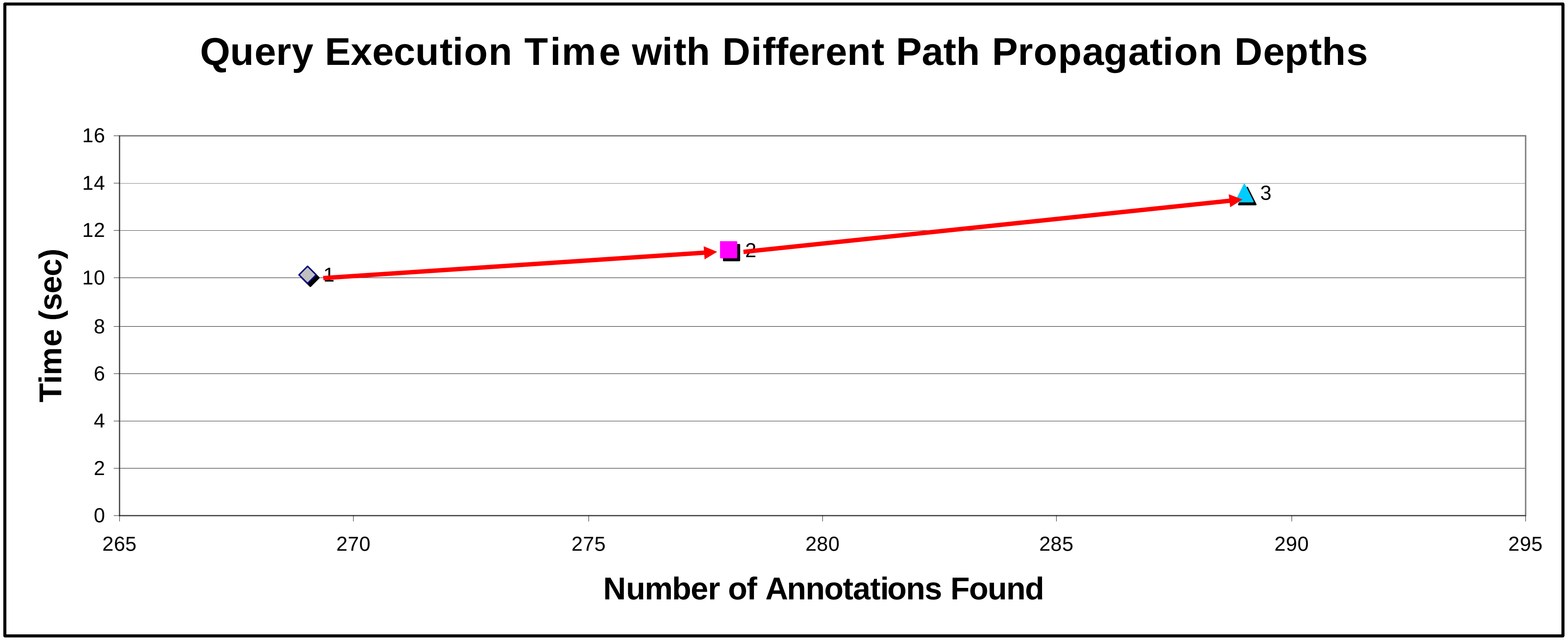






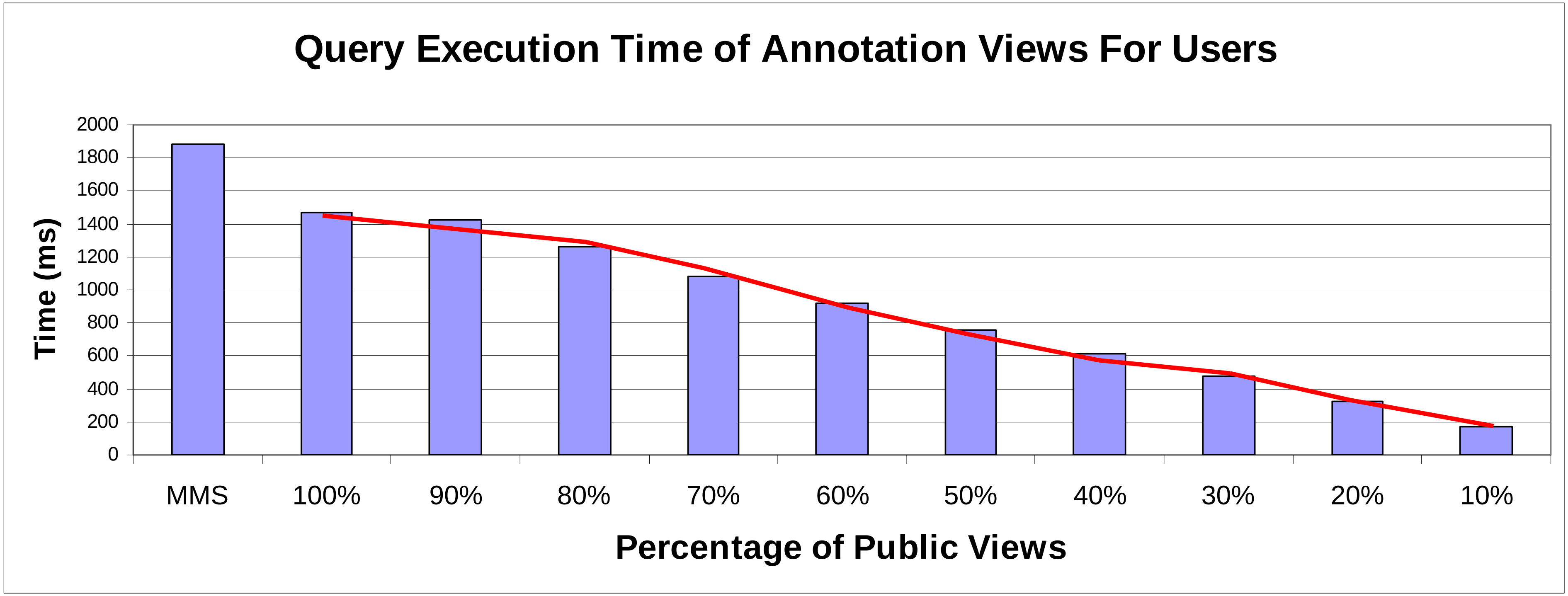
- Higher density with more cache hits, less query time

Path Depths



- Annotations can be propagated via paths
- Topological order gives the inheritance information

User-centric Annotation Views



- Less public views, less query time
- Expect such user-centric features to have a compound effect if used together

Outline

- Motivation
- Usage Patterns
  - Time Semantics
  - Network Semantics
- User-centric Access Control
- Putting It All Together
- Experimental Evaluation
- Summary

Related Work – Standard Features

Features	**DBNotes**	**Mondrian**	**ULDB**	**bdbms**	**MMS**	**ViP**
Annotation	Yes	Yes	Confidence	Yes	Yes	Yes
Provenance	Yes	Yes	Lineage	Yes	Yes	Yes
Time Semantics:						
•**Implicitly-defined**	No	No	No	No	**Yes**	**Yes**
•**Explicitly-defined**	No	No	No	No	**No**	**Yes**
Network Semantics:						
•**Implicitly-defined**	Limited	Limited	Limited	Limited	**Yes**	**Yes**
•**Explicitly-defined**	No	No	No	No	**No**	**Yes**
Propagate Type	Eager	On-demand	On-demand	Eager	On-demand	Hybrid
Annotation Storage	Naive	Naive	X-relations	Annotation table	Q-type	A-table
Scalability	Small	Medium	Medium	Medium	Large	Large
Query	pSQL	Color algebra	TriSQL	A-SQL	Predicate	ViP-SQL

Related Work – User-centric Features

Features	DBNotes	Mondrian	ULDB	bdbms	MMS	ViP
Time Semantics:						
Valid Time	No	No	No	No	No	Yes
Network Semantics:						
Propagation Method	Yes	No	No	Limited	No	Yes
Access Control:						
Annotation	No	No	No	Limited	No	Yes
Annotation Views	No	No	No	No	No	Yes
Annotation Paths	No	No	No	No	No	Yes

Contributions

- Introduce new annotation propagation methods, suitable for scientific data
- Propose user-centric features that enable users to personalize annotation propagation
- Propose to use views as the formal mechanism to implement the new annotation propagation features and also as a user-interface
- Utilize caching to significantly improve the performance over the state of the art
- Experimentally evaluate the proposed ViP framework using a real system implementation and simulated workloads

http://db.cs.pitt.edu

http://cmpi.cs.pitt.edu

Questions and comments?

Caching

- If a data tuple is not found in the cache, execute the annotation query and save its annotation query result set into the cache
- If a data tuple is found in the cache, verify if it is still “fresh”
- No action if a data tuple is inserted, deleted, or updated
- An annotation registration is updated/inserted, reset the cache appropriately
- If an annotation is removed, remove its related entries from the cache

User-centric Network Semantics – HAP (Cont.)

- Given these three parameters (some of which are optional):
  - By setting HAP-i or MAX-HAP to 0, effectively disable explicit annotation direct paths
  - By setting MAX-HAP to 1, effectively disable cascading annotation propagation

User-centric Access Control - Motivation

- Scientific annotation must have a strong user-centric component
  - Appropriate access controls need to be in place for the raw data and the annotations on them
  - The annotations are often private, since they reflect additional analysis that is not ready to be made available to all
  - The way that raw data are associated to private information that should not be made public

User-centric Access Control on Annotation Paths

- Users would want to control who can take advantage of the explicit annotation propagation paths that they introduce
  - For confidentiality of paths, i.e., not willing to make relationships between data public
  - For scalability of paths from an information absorption point of view, i.e., not everybody is interested in everybody else’s beliefs on which data is related
  - Certain paths will not be visible to some users