T. J. Watson Research Center

Unifying Data and Domain Knowledge Using Virtual Views

Lipyeow Lim, Haixun Wang & Min Wang

Background

- **DBMS originally designed for transaction data**
- **Many extensions for richer queries attempted**
  - OO DBMS and ORDBMS
  - OLAP (1990s)
  - Data Cube (ICDE 1996)
  - Data Mining (CACM 1996)
- **An unending quest**
  - Database or Knowledge-base?
  - New applications: the Semantic web, etc.
- **Move from simple transactional or analytical processing to semantics understanding and knowledge inferencing**

A motivating example

- **RDBMS allows us to query wines through attributes** ID, Type, Origin, Maker, Price**.**
- **Expressive power: relational complete (quite limited).**
- **Human intelligence operates in a quite different way.**

ID	Type	Origin	Maker	Price
1	Burgundy	CotesDOr	ClosDeVougeot	30
2	Riesling	NewZealand	Corbans	20
3	Zinfandel	EdnaValley	Elyse	15

A base table : Wine

Query 1:

- **Which wine originates from the US?**
  - Answer: Zinfandel
  - Zinfandel's **Origin** EdnaValley is located in California.
  - Domain knowledge used: EdnaValley is in California, and California is in the US.

- **We could issue:**

```
SELECT ID
FROM Wine
WHERE Origin = 'US';
```

Nice! Except nothing will be returned

ID	Type	Origin	Maker	Price
1	**Burgundy**	**CotesDOr**	**ClosDeVougeot**	**30**
2	**Riesling**	**NewZealand**	**Corbans**	**20**
3	**Zinfandel**	**EdnaValley**	**Elyse**	**15**

Query 2:

- **Which wine is a red wine?**
  - Answer: Zinfandel & Burgundy
  - Domain knowledge used:
    - Zinfandel is red;
    - Burgundy can be either red or white, but Burgundy from CotesDor is always red

- **We could issue:**

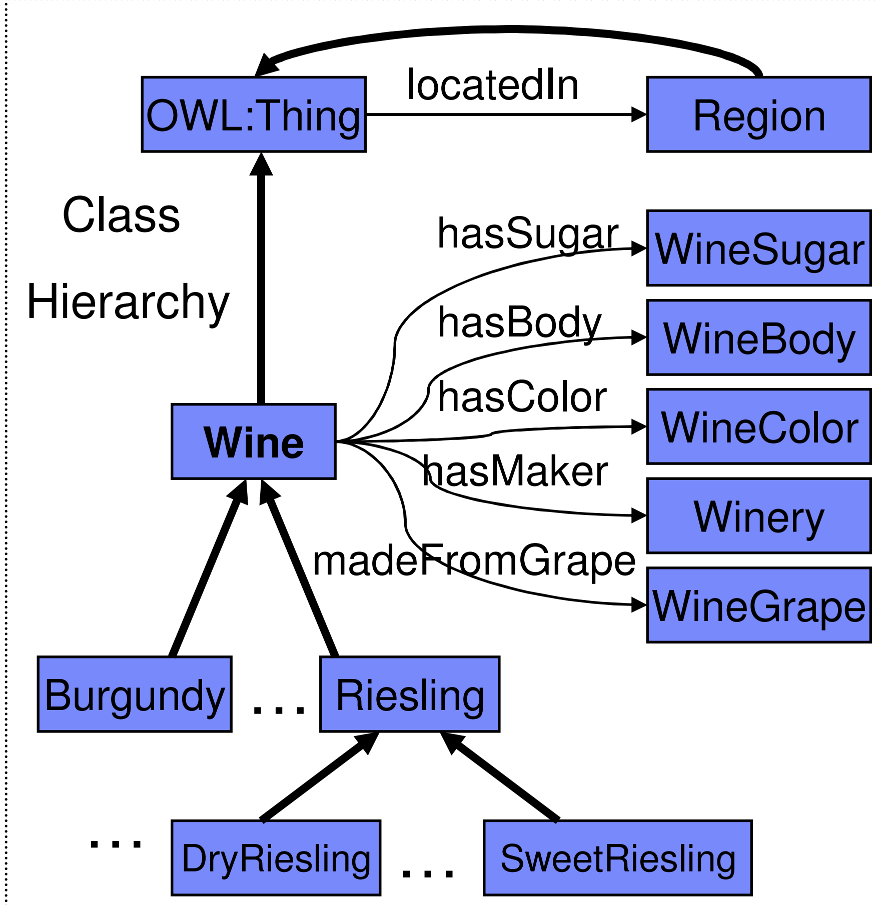
```
SELECT ID
FROM Wine
WHERE hasColor = 'red';
```

But "hasColor" is not a column in the table

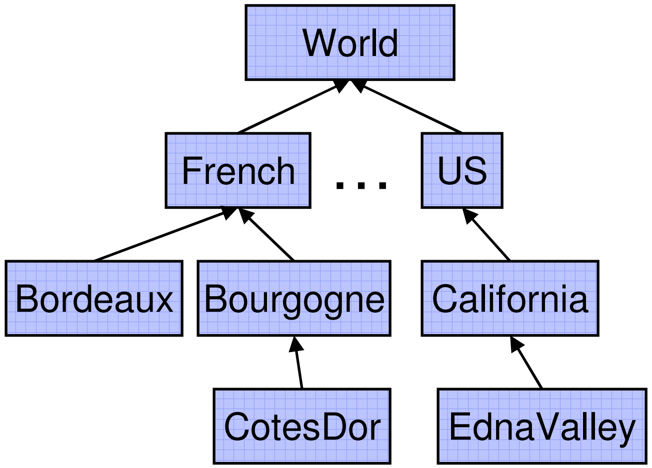
ID	Type	Origin	Maker	Price
1	**Burgundy**	**CotesDOr**	**ClosDeVougeot**	**30**
2	**Riesling**	**NewZealand**	**Corbans**	**20**
3	**Zinfandel**	**EdnaValley**	**Elyse**	**15**

Domain Knowledge from OWL Ontology

- **Eg. Wine Ontology from the web ontology language OWL (W3C)**
- **Extract class hierarchies, (transitive) properties, implications, etc from OWL**



Transitve Property *locatedIn*



Implications

(Type=CotesDor)↔(type=RedBurgundy) & (origin=CotesDorRegion)

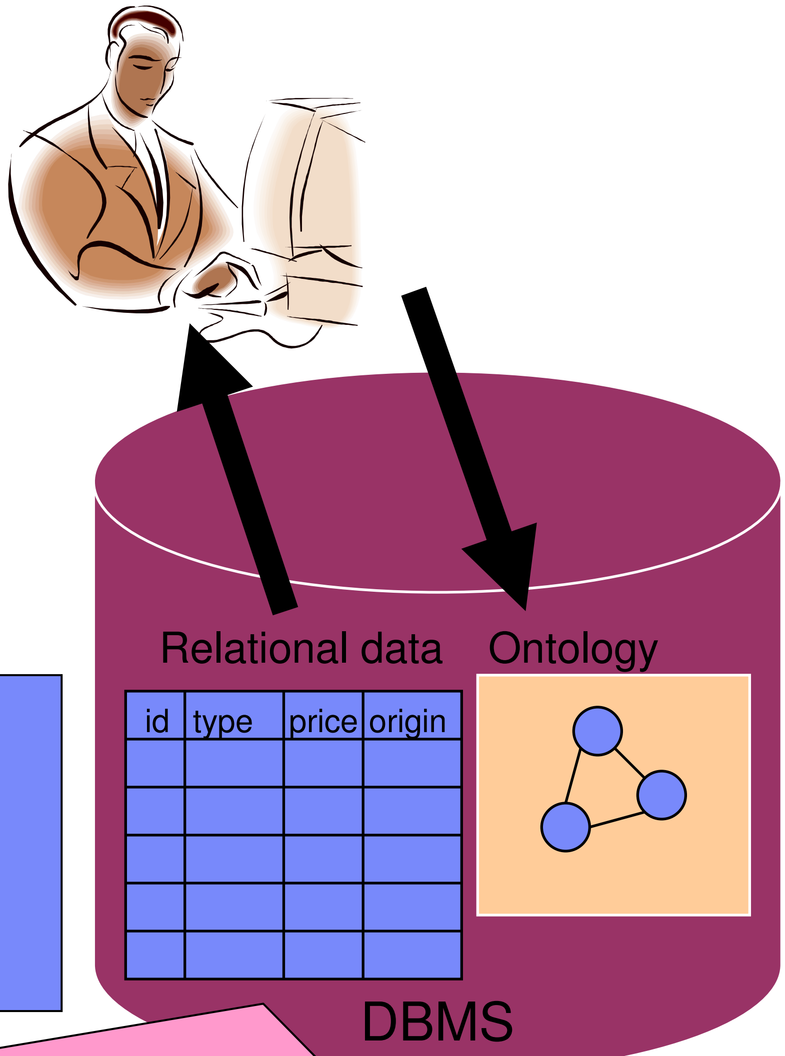
(Type=Zinfandel) → (hasColor=Red)

(Type=Zinfandel) → (hasSugar=dry)

(Type=RedWine) → (hasColor=Red)

Challenges

- **How to incorporate domain knowledge (ontology) into a RDBMS?**
- **How to integrate relational data with domain knowledge ?**
- **How to query relational data with meaning ?**
- **How to process such queries ?**

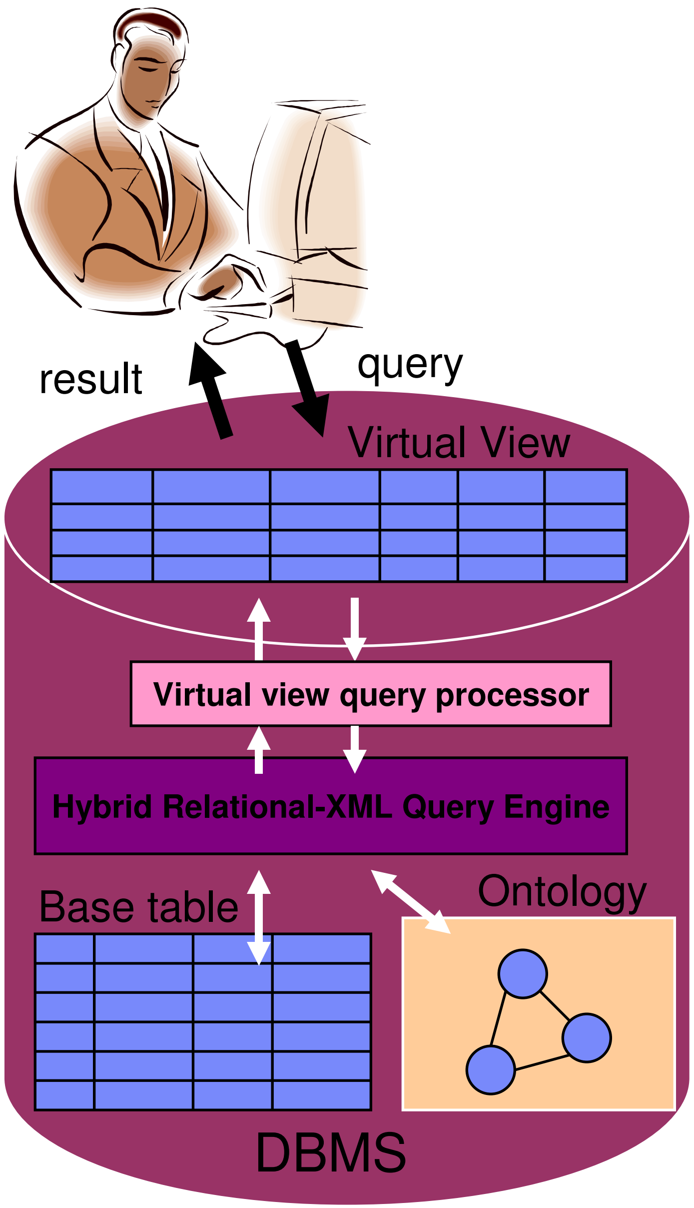


<u>Disclaimer</u>
- Not re-inventing
  - Expert Systems
  - Datalog Systems
  - OWL/RDF & SparQL Systems

Put a little semantics into relational SQL systems

Overview of our solution

- **Provide user with a unified view of the data and the domain knowledge.**
- **Through the virtual view, we offer a rich set of functionalities for knowledge inferencing out of the Spartan simplicity of SQL.**
- **Leverage hybrid relational-XML storage for managing domain knowledge**
- **Rewrite query on virtual view**
- **Leverage hybrid relational-XML query engine to process re-written query.**



Virtual View Unifies Data & Ontology

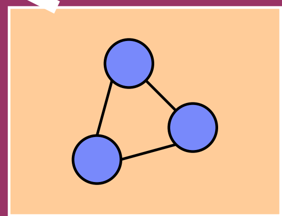
- **Users create virtual views over the relational data and the ontology**
- **Virtual columns/attributes not in original data**
- **Virtual columns not materialized -- inferred from the ontology**

DBMS

id	type	origin	maker	price	locatedIn(origin)	hasColor
1	Burgundy	CotesDOr	ClosDeVougeot	30	{Bourgogne, France}	red
2	Riesling	NewZealand	Corbans	20	{}	white
3	Zinfandel	EdnaValley	Elyse	15	{California, US}	red

id	type	origin	maker	price
1	Burgundy	CotesDOr	ClosDeVougeot	30
2	Riesling	NewZealand	Corbans	20
3	Zinfandel	EdnaValley	Elyse	15

Relational data



Ontology

The virtual view

ID	Type	Origin	Maker	Price	*LocatedIn*	*hasColor*
1	**Burgundy**	**CotesDOr**	**ClosDeVougeot**	**30**	**{Burgundy,France}**	**red**
2	**Riesling**	**NewZealand**	**Corbans**	**20**	**{}**	**white**
3	**Zinfandel**	**EdnaValley**	**Elyse**	**15**	**{California,US}**	**red**

- **Wine Burgundy is originated from CotesDOr, which is a sub-region of Burgundy, which in turn, is a sub-region of France.**

- **(type = Zinfandel) → (hasColor = red)**
- **(type = Riesling) → (hasColor = white)**

Creating the Virtual View

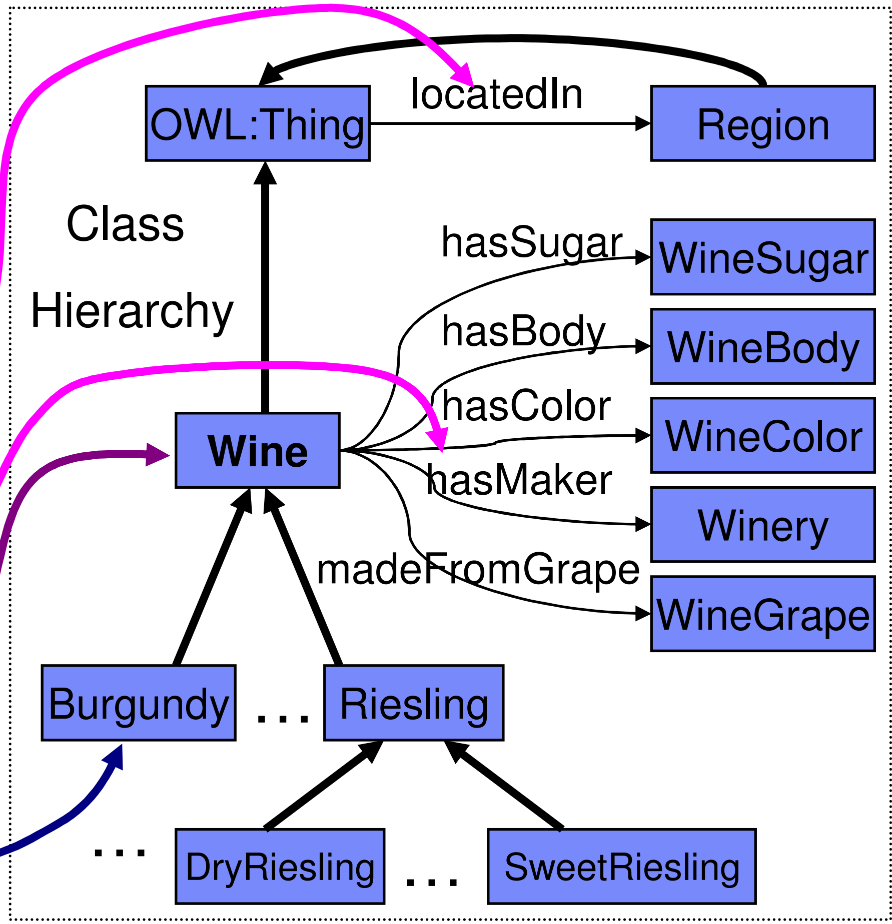
CREATE VIRTUAL VIEW WineView(Id, Type, Origin, Maker, Price, LocatedIn, HasColor) **AS**

SELECT W.*, O.locatedIn, O.hasColor,

FROM Wine **AS** W, WineOntology **AS** O

WHERE O.type = W.type **AND**
(O.type isA 'Wine') **AND**
O.locatedIn = W.origin **AND**
O.hasMaker = W.maker

Wine Ontology



Wine Table

id	type	origin	maker	price
1	Burgundy	CotesDOr	ClosDeVougeot	30
2	Riesling	NewZealand	Corbans	20
3	Zinfandel	EdnaValley	Elyse	15

Now we can write the semantic queries

- **Which wine originates from the US?**

```
SELECT Id
FROM WineView
WHERE 'US' IN LocatedIn;
```

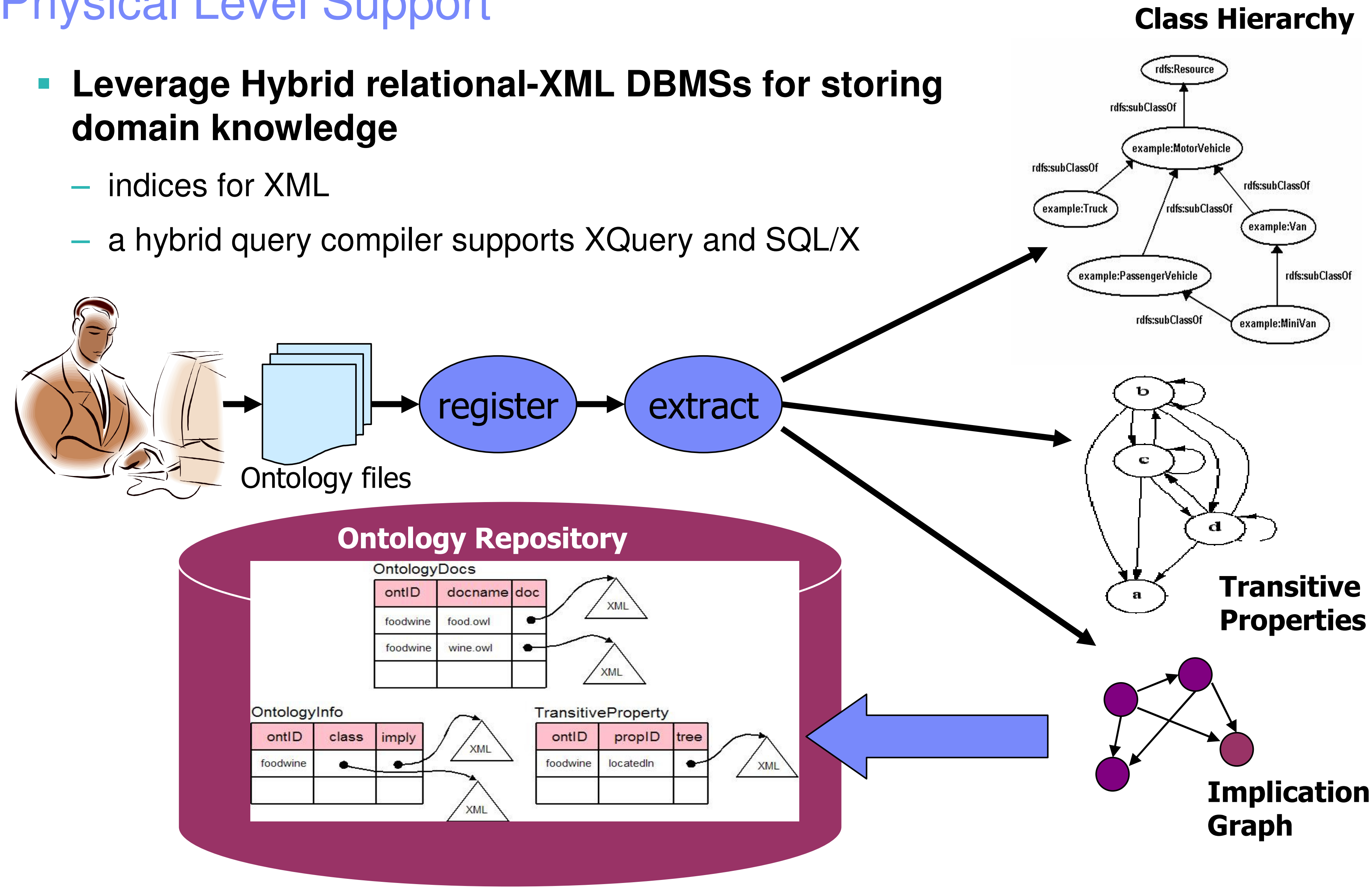
- **Which wine is a red wine?**

```
SELECT Id
FROM WineView
WHERE hasColor = 'red';
```

id	type	origin	maker	price	locatedIn(origin)	hasColor
1	Burgundy	CotesDOr	ClosDeVougeot	30	{Bourgogne, French}	red
2	Riesling	NewZealand	Corbans	20	{}	white
3	Zinfandel	EdnaValley	Elyse	15	{California, US}	red

Physical Level Support

- **Leverage Hybrid relational-XML DBMSs for storing domain knowledge**
  – indices for XML
  – a hybrid query compiler supports XQuery and SQL/X



Hybrid Relational-XML DBMS

- CREATE TABLE **ClassHierarchy(**
  id INTEGER**, name** VARCHAR**(27), hierarchy** XML **);**
- INSERT INTO **ClassHierarchy** VALUES**(1, ’Wine’,**

```
XMLParse('<?xml version='1.0'>
<wine>
    <WhiteWine><WhiteBurgundy>...</WhiteBurgundy>...
  </WhiteWine>
  <DessertWine><SweetRiesling/>...
  </DessertWine>...
</wine>')
);
```

- Example: **find class ids and class names of all class hierarchies that contain the XPath /Wine/DessertWine/SweetRiesling:**

```
SELECT id, name
FROM ClassHierarchy AS C
WHERE XMLExists('$t/Wine/DessertWine/SweetRiesling'
                PASSING BY REF C.order AS "t")
```

Query Re-writing

- **Query expansion on virtual columns using implications.**

SELECT V.Id

FROM WineView AS V

WHERE V.hasColor=White;

- Since the following implications exists, we use them to expand the query predicate

(Type=WhiteWine) → (hasColor=white)

(Type=Riesling) → (hasColor=white)

SELECT V.Id

FROM Wine AS W

WHERE W.type=WhiteWine

OR W.type=Riesling;

- **Subsumption checking via XPath & XMLExists SQL/XML function**

SELECT V.Id

FROM WineView AS V

WHERE US ∈ V.locatedIn;

- Since locatedIn is a virtual column on the transitive closure of W.origin, we rewrite the query to

SELECT W.Id

FROM Wine AS W, TransitiveProperty AS T

WHERE T.ontID='wine'

AND T.propID='locatedIn'

AND XMLExists(T.tree//USRegion//W.origin);

But the expansion is not that simple

SELECT ID

FROM V

WHERE (A=v1)

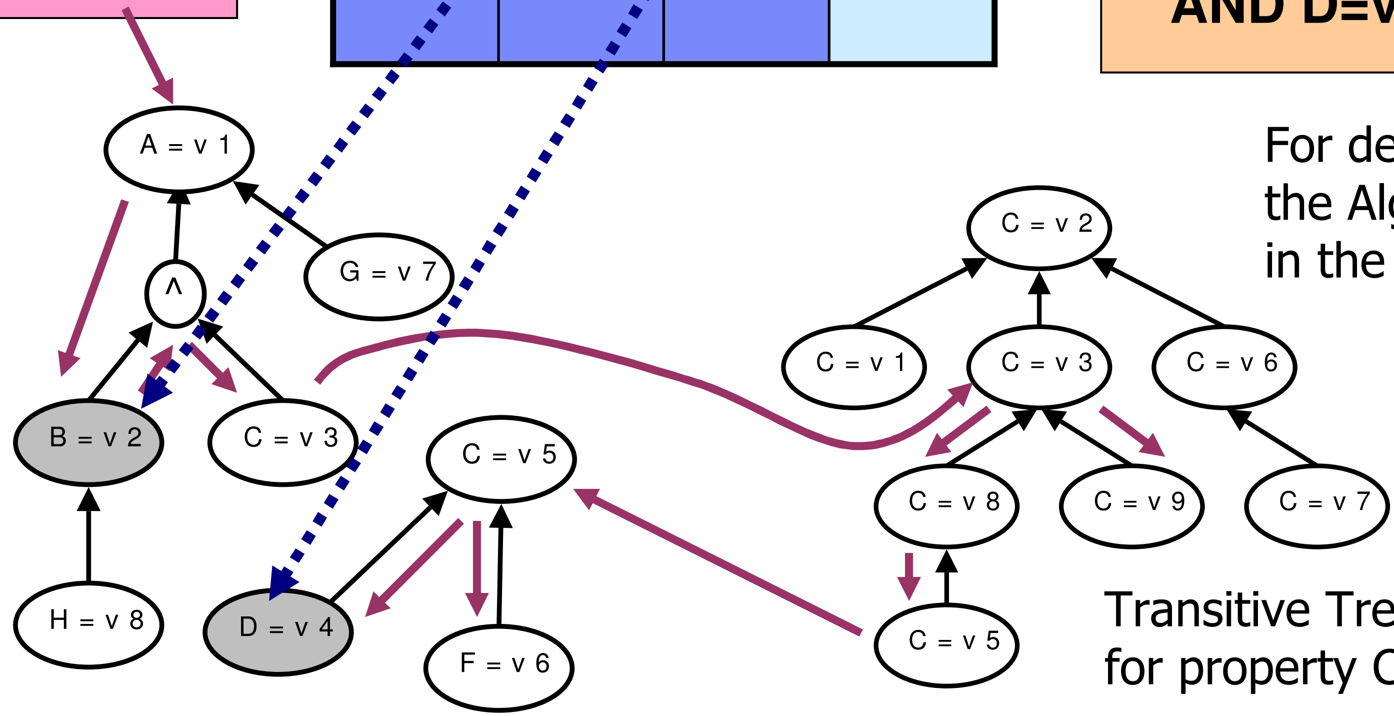
Virtual View **V**

ID	B	D	A

SELECT ID

FROM V

WHERE B=v2 AND D=v4



Implication Graph

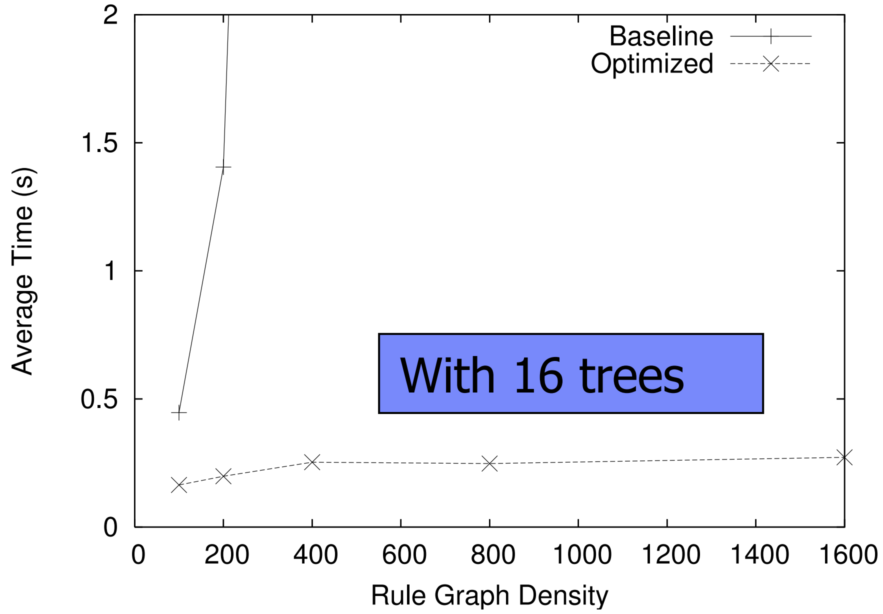
Transitive Tree for property C

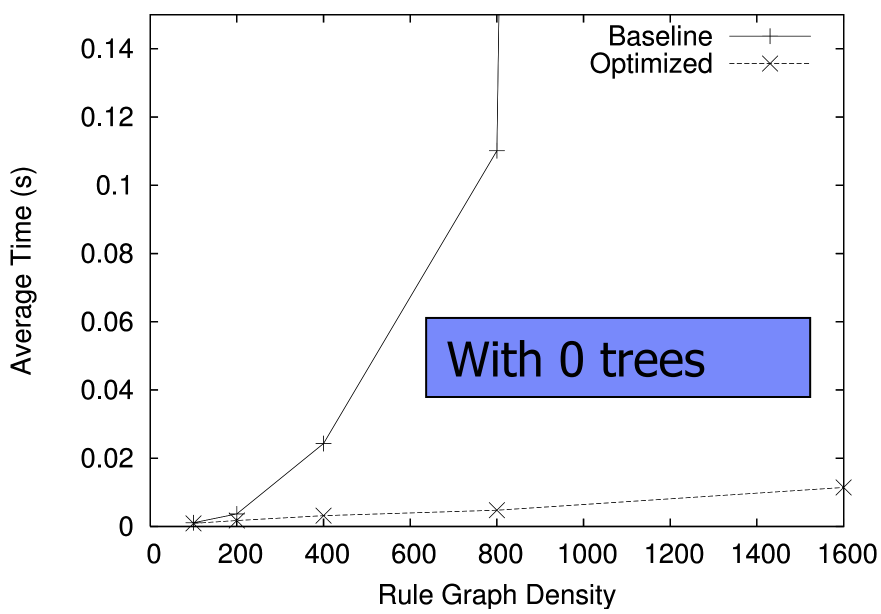
For details see the Algorithm in the paper

Experiments

- **Investigate time to rewrite the queries on virtual views**
- **Data Generation**
  - trees for transitive properties parametrized by
    - Number of nodes
    - Maximum fanout
  - graphs for implications parametrized by
    - Number of relationships
    - Number of values
    - Number of levels in the graph
    - Density : number of rules between two consecutive levels
    - Fanout : number of atoms in a rule body
- **Measurement: rewriting time averaged over 5 randomly generated data sets.**
- **Performance for baseline rewriting algorithm and optimized rewriting algorithm (using memoization)**

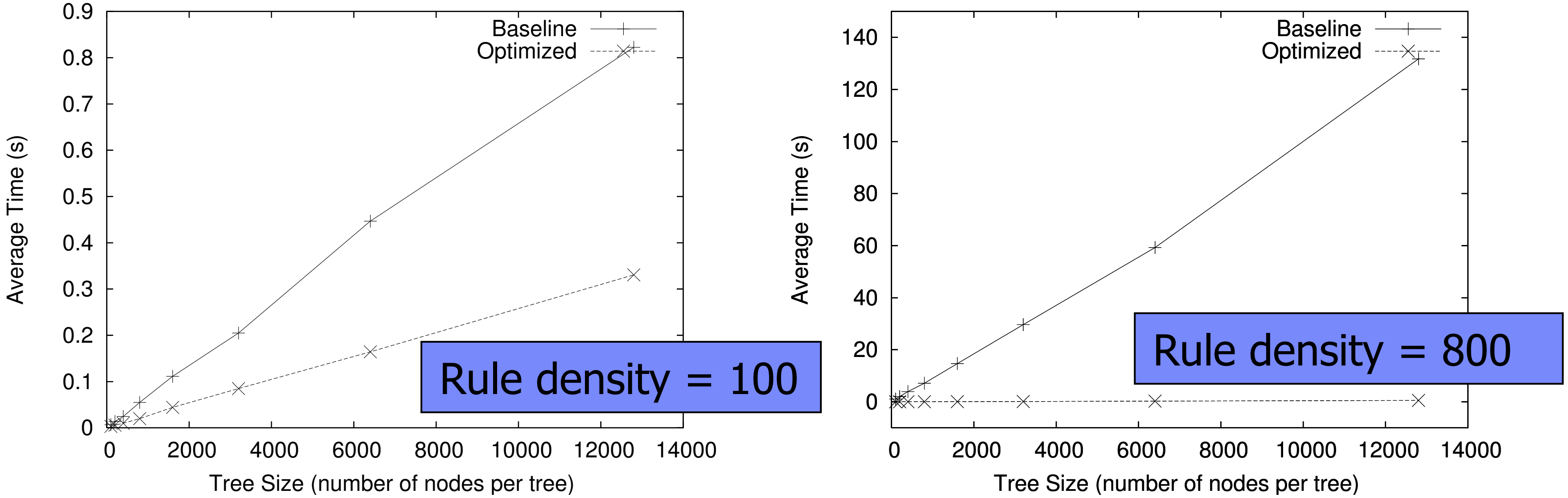
Implication Graph Density





- **Number of rules did not affect rewriting performance as much as density of the implication rule graph.**
- **Baseline algorithm is not scalable. Memoization is much better.**
- **In general, the rewriting time is reasonable (< 0.5 s)**

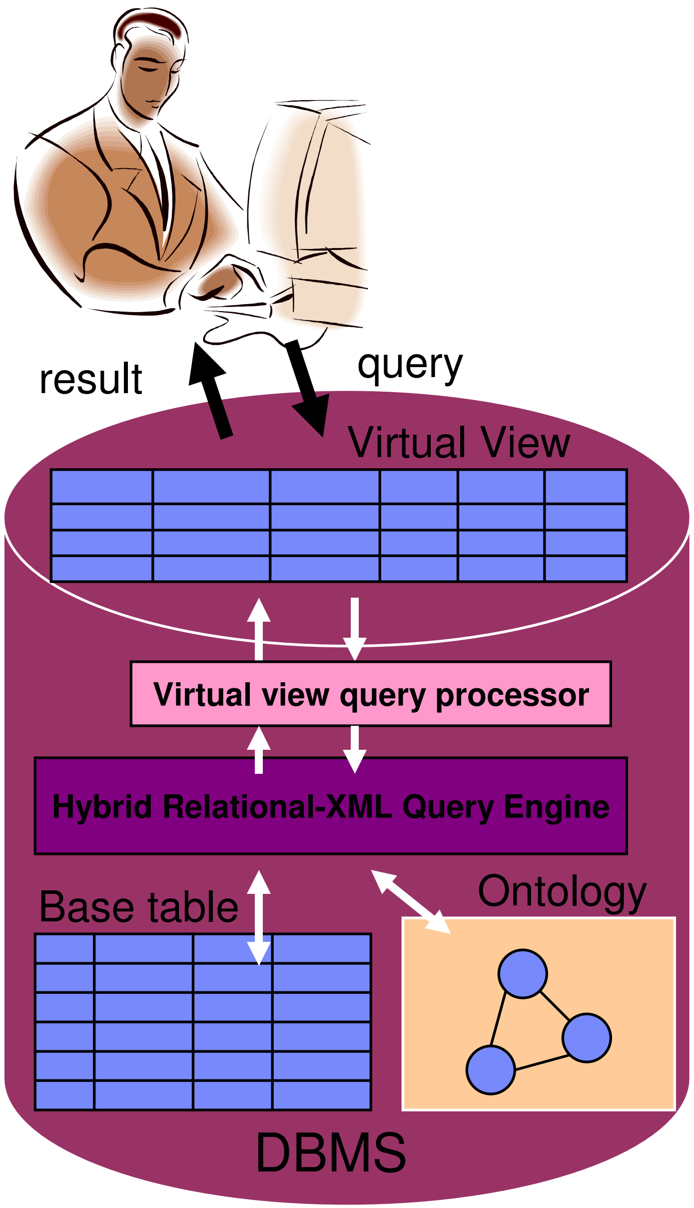
Size of transitive property trees



- **Rewriting time scales linearly with size of trees.**

Conclusion

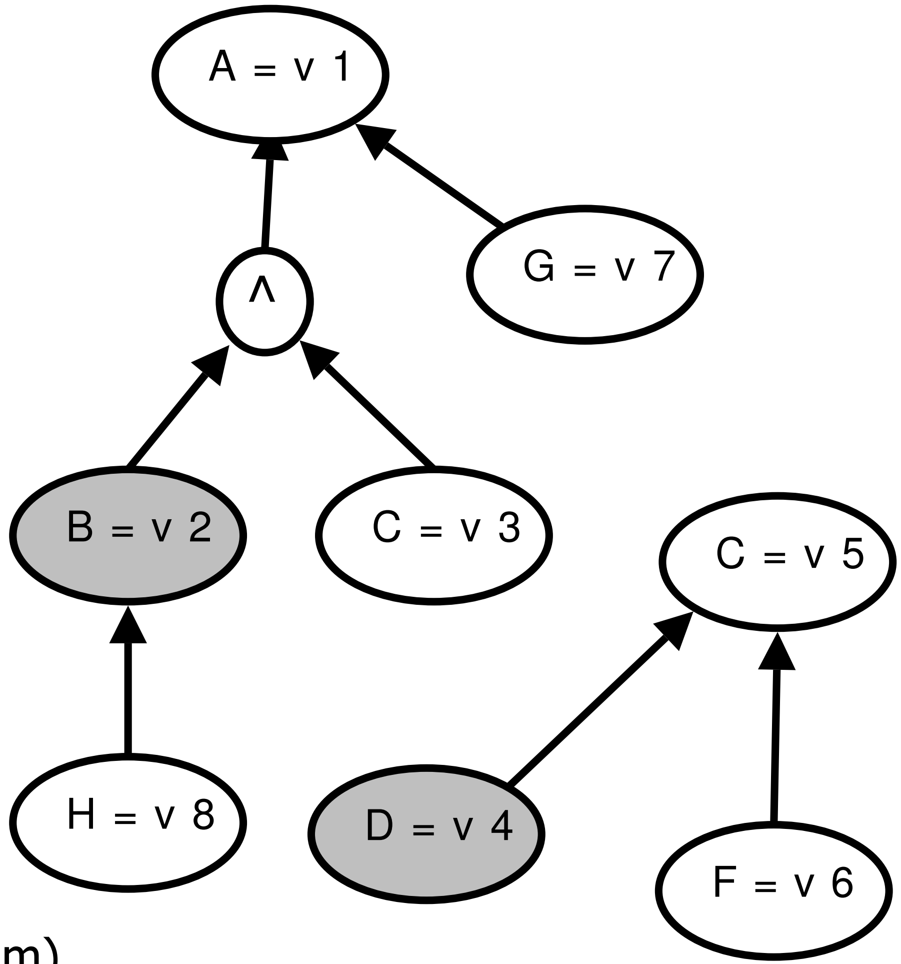
- **Framework for putting a little semantics into relational SQL systems.**
- **Users register ontologies in DBMS and links them with relational data by creating virtual views**
- **Virtual columns in the virtual views are not materialized**
- **Queries on the virtual columns are rewritten to predicates on base table columns.**
- **Future work: performance issues**



Questions

Implication Graph

- **A=v1 ← G=v7**
- **A=v1 ← B=v2 ^ C=v3**
- **B=v2 ← H=v8**
- **C=v5 ← D=v4**
- **C=v5 ← F=v6**



A=v1 : Clause (e.g., x.hasBody=Medium)

^ : Operator (e.g., AND)