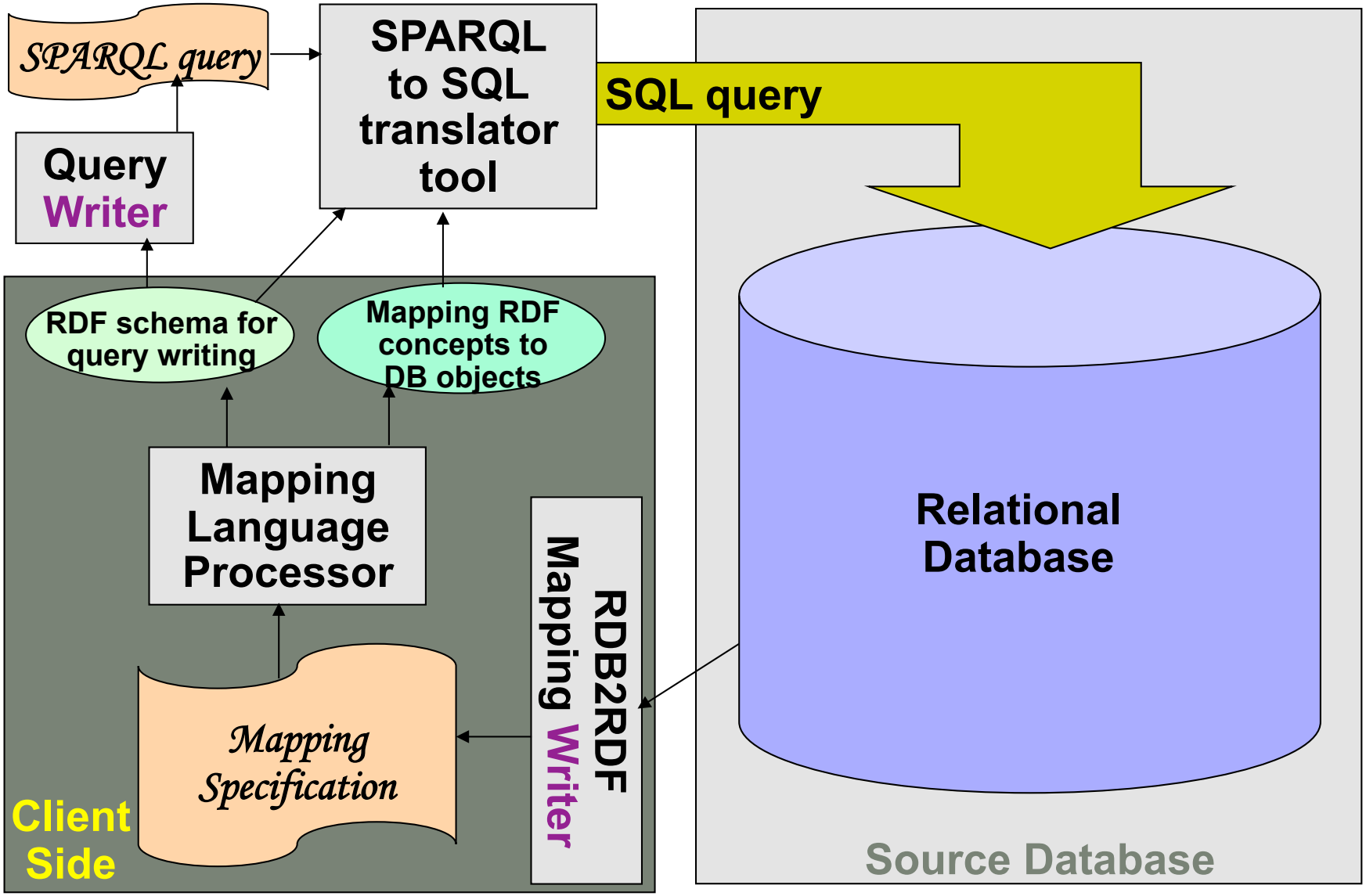
A SQL-based Approach for Mapping Relational Data to RDF

Souripriya Das
Database Semantic Technologies Group, Oracle

THE FOLLOWING IS INTENDED TO OUTLINE OUR GENERAL PRODUCT DIRECTION. IT IS INTENDED FOR INFORMATION PURPOSES ONLY, AND MAY NOT BE INCORPORATED INTO ANY CONTRACT. IT IS NOT A COMMITMENT TO DELIVER ANY MATERIAL, CODE, OR FUNCTIONALITY, AND SHOULD NOT BE RELIED UPON IN MAKING PURCHASING DECISION. THE DEVELOPMENT, RELEASE, AND TIMING OF ANY FEATURES OR FUNCTIONALITY DESCRIBED FOR ORACLE'S PRODUCTS REMAINS AT THE SOLE DISCRETION OF ORACLE.

Architecture Diagram for RDB2RDF processing

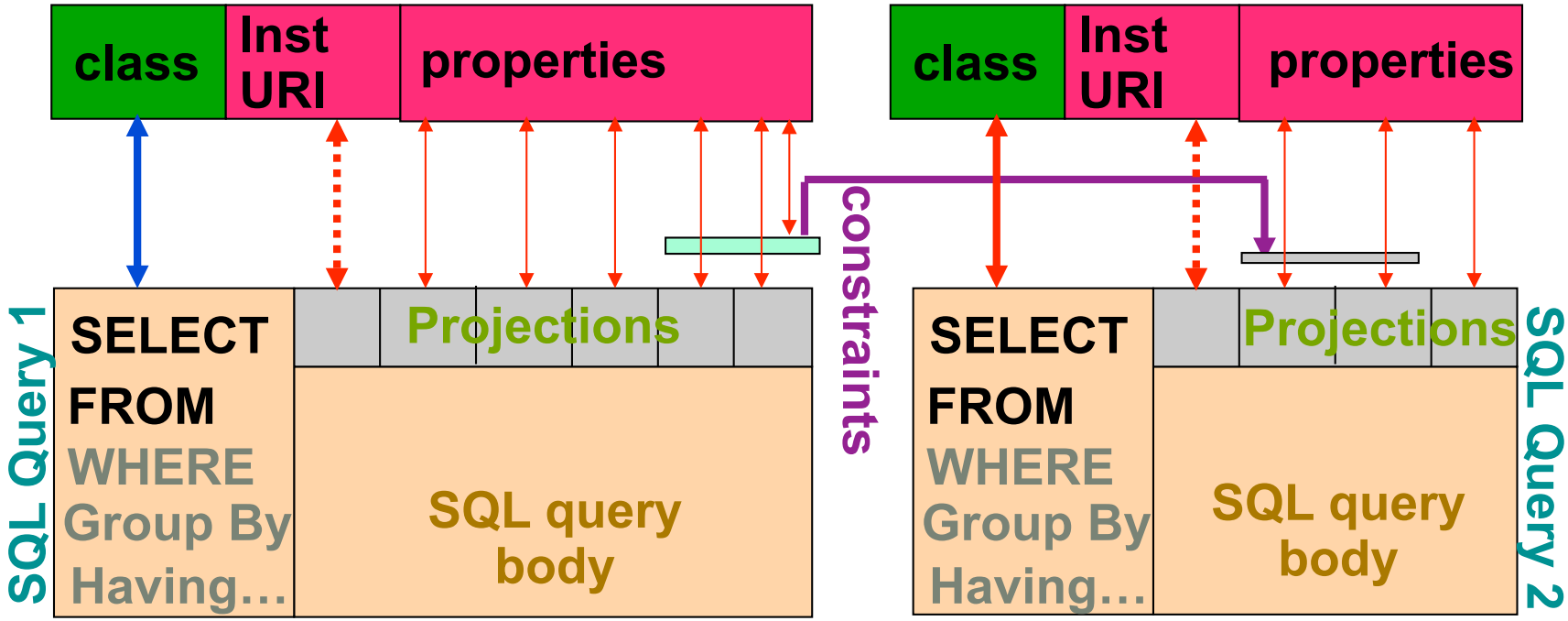


Overview

- Basis for defining an RDFS/OWL class and its properties based on relational data stored in a source database:
  - A SQL query specifying arbitrary transformation of source relational data
  - Related foreign and unique (or primary) key constraint definitions.

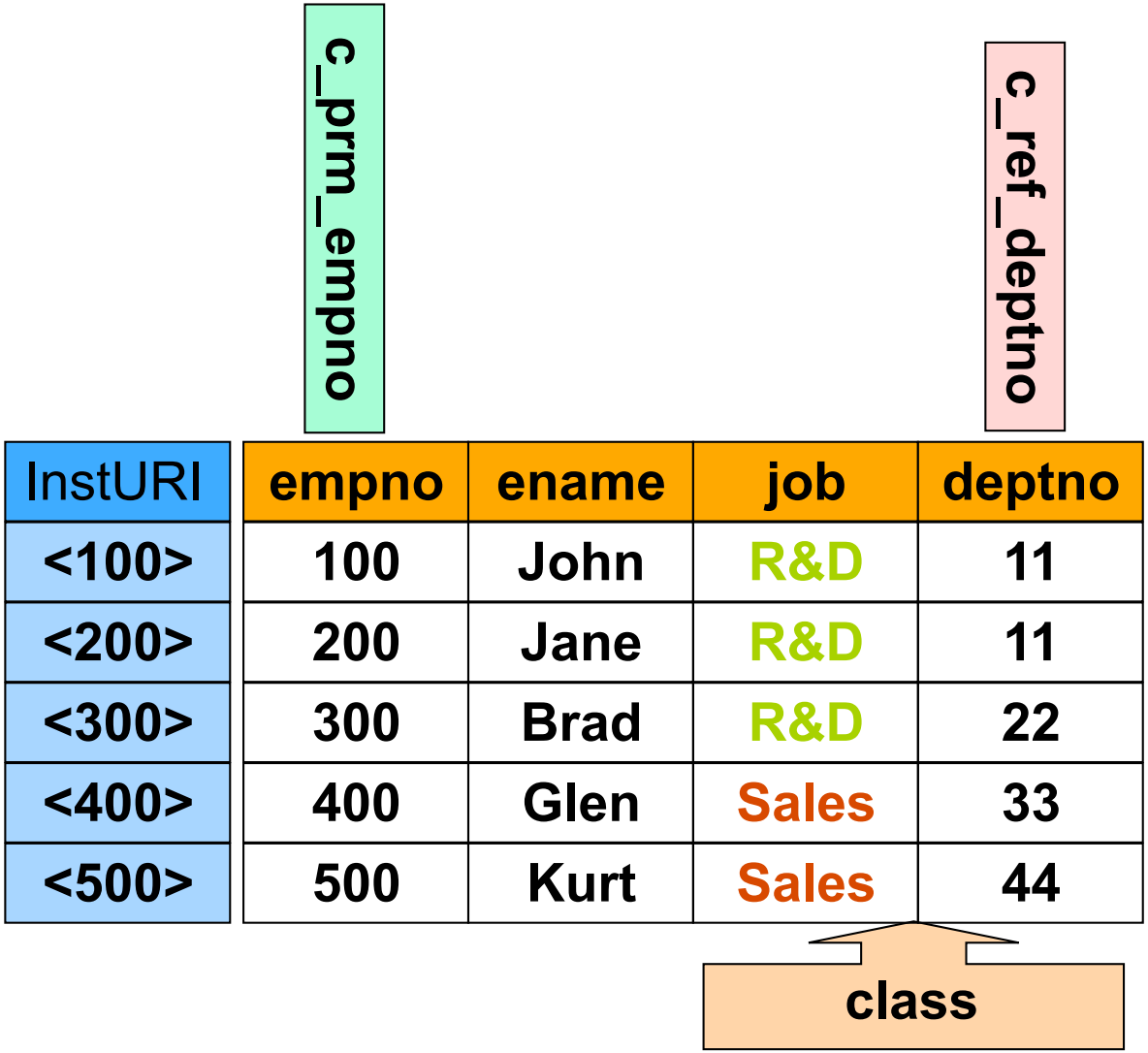
- Client side capabilities may vary
  - We will assume => No SQL support on client-side: Only connects to database using JDBC or ODBC.

Overview



- Proposed mapping language: (Leverage SQL on source database and) provide the simplest way to specify the following mapping
  - Classes ←→ (SQL queries on) relational tables
  - Datatype properties ←→ (projected) columns (from SQL queries)
  - Object properties ←→ foreign key constraints (like a virtual column)
- The language must also support constraint specification (no expressions)
  - ordered group of projected columns: ConsName, type, <col-seq.>

Table (or SQL query) ➔ RDF Class and Properties



Tuples for mapping Classes

Class (ClassName, SQLdefString)
Class (“xyz:dept”, <sql-def-for-dept>)
Class (“xyz:emp”, <sql-def-for-emp>)

Tuples for mapping Properties[1]

Property (PropertyName, ViewName, ColPos)
Property (“dept:InstURI”, "xyz:dept”, 1)
Property (“dept:deptno”, "xyz:dept”, 2)
Property (“dept:Name”, “xyz:dept”, 3)
Property (“dept:location”, “xyz:dept”, 4)
Property (“emp:InstURI”, “xyz:emp”, 1)
Property ("emp:empno”, “xyz:emp”, 2)
Property (“emp:Name”, “xyz:emp”, 3)
Property (“emp:rdf:type”, “xyz:emp”, 4)
Property (“emp:job”, “xyz:emp”, 5)
Property (“emp:deptNum”, “xyz:emp”, 6)
Property (“emp:c_ref_deptno”, “xyz:emp”, 0)

[1] **Inverse function specification for properties not shown, but can be added easily.**

- **Range information can be derived from type info obtained from SQL engine at source database.**

Tuples for mapping Constraints

Constraint (ConsName, ConsType, ClassName, RefConsName)
Constraint (“dept:c_unq_deptno”, Unique, “xyz:dept”, NULL)
Constraint (“emp:c_prm_empno”, Primary, “xyz:emp”, NULL)
Constraint (“emp:c_ref_deptno”, Reference, “xyz:emp”, “dept:c_unq_deptno”)

ConstraintColumn (ConsName, ClassName, ColName, ColPosInKey)
ConstraintColumn (“dept:c_unq_deptno”, “xyz:dept”, “dept:deptno”, 1)
ConstraintColumn (“emp:c_prm_empno”, “xyz:emp”, “emp:empno”, 1)
ConstraintColumn (“emp:c_ref_deptno”, “xyz:emp”, emp:deptNum”, 1)

RDF Schema generated by Mapping specification

xyz:dept	rdf:type	rdfs:Class .			
xyz:emp	rdf:type	rdfs:Class .			
dept:deptno	rdfs:domain	xyz:dept ;	rdfs:range	xsd:integer .	
dept:Name	rdfs:domain	xyz:dept ;	rdfs:range	xsd:string .	
dept:location	rdfs:domain	xyz:dept ;	rdfs:range	xsd:string .	
emp:empno	rdfs:domain	xyz:emp ;	rdfs:range	xsd:integer .	
emp:Name	rdfs:domain	xyz:emp ;	rdfs:range	xsd:string .	
emp:rdf:type	*rdfs:domain*	*xyz:emp ;*	*rdfs:range*	*rdfs:Class .*	
emp:job	rdfs:domain	xyz:emp ;	rdfs:range	xyz:string .	
emp:deptNum	rdfs:domain	xyz:emp ;	rdfs:range	xsd:integer .	
emp:c_ref_deptno	rdfs:domain	xyz:emp ;	rdfs:range	xyz:dept .	

SPARQL Query: Avoiding join

```
SELECT    ?x ?ename
WHERE {   ?x  xyz:ename       ?ename .
          ?x  xyz:job <xyz.com/emp/job/SALES> }
```

```
SELECT    e.instURI x
          , foo(e.ename) ename
FROM      emp e
WHERE     e.job = 'SALES'
```

Advanced topic: Subclasses and rdf:type properties

c_prm_empno

c_ref_deptno

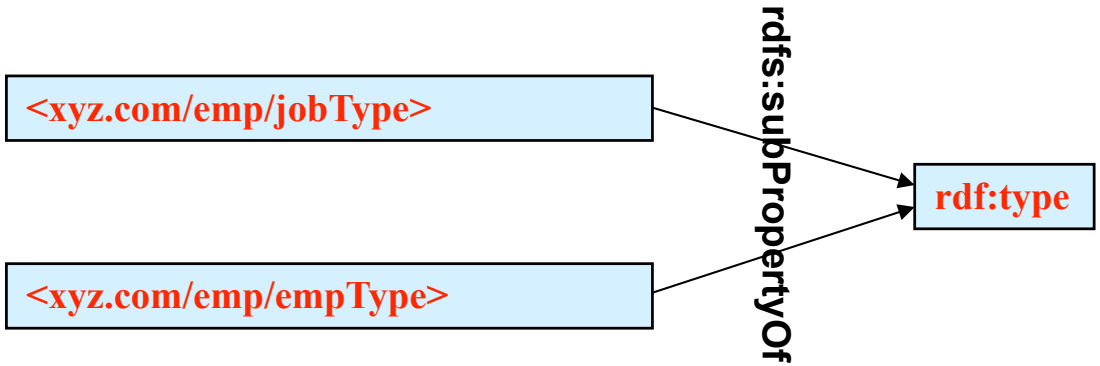
job- based subclasses

etype- based subclasses

InstURI	empno	ename	job	deptno	rdf:type	etype	rdf:type
<100>	100	John	R&D	11		part	<part>
<200>	200	Jane	R&D	11		full	<full>
<300>	300	Brad	R&D	22		intrn	<intrn>
<400>	400	Glen	Sales	33		full	<full>
<500>	500	Kurt	Sales	44		full	<full>

class

Subproperties of rdf:type



Advanced topic: destination graph(s)

c_prm_empno

c_ref_deptno

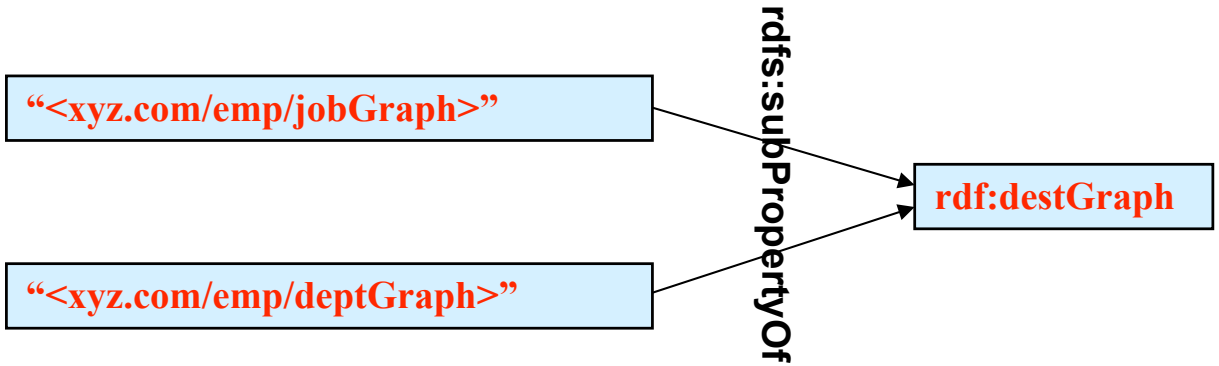
job- based destination graph

deptno- based destination graph

InstURI	empno	ename	job	deptno	rdf:graph	etype	rdf:graph
<100>	100	John	R&D	11		part	<d11>
<200>	200	Jane	R&D	11		full	<d11>
<300>	300	Brad	R&D	22		intrn	<d22>
<400>	400	Glen	Sales	33		full	<d33>
<500>	500	Kurt	Sales	44		full	<d44>

class

Subproperties of a special property: rdf:destGraph



SPARQL Query: Involving generic rdf:type

```
SELECT    ?x ?ename
WHERE {   ?x <xyz.com/emp/Name> ?ename .
          ?x rdf:type <xyz.com/emp/job/SALES> }
```

```
SELECT    e.instURI x
          , e.“<xyz.com/emp/Name>” ename
FROM      “<xyz.com/emp>” e
WHERE     e.“<xyz.com/emp/jobtype>” = “<xyz.com/emp/job/SALES>”
or        e.“<xyz.com/emp/emptype>” = “<xyz.com/emp/job/SALES>”
```


ORACLE IS THE INFORMATION COMPANY