R2RML: RDB to RDF Mapping Language

Werner Nutt

Acknowledgment

These slides are based on a slide set by Mariano Rodriguez

Reading Material/Sources

- R2RML specification by W3C
  http://www.w3.org/TR/r2rml/

- R2RML specification byW3C
  http://www.w3.org/2001/sw/rdb2rdf/test-cases/

Standards and Tools

Mapping languages

- Standards by RDB2RDF working group (W3C)
  - Direct Mapping
  - R2RML
- Proprietary

Tools

Free: D2R, Virtuoso, Morph, r2rml4net, db2triples, ultrawrap, Quest

- Commercial: Virtuoso, ultrawrap, Oracle SW

- Overview and Examples
- Detailed Specification

- Overview and Examples
- Detailed Specification

R2RML Overview

R2RML is a language for specifying mappings from relational to RDF data.

A mapping takes as input a logical table, i.e.,

- a database table
- a database view, or
- an SQL query

(called an “R2RML view” because it is like an SQL view but does not modify the database)

A logical table is mapped to a set of triples by a rule called

- triples map.

Triples Maps

A triples map has two parts:

- a subject map
- several predicate-object maps (combining predicate and object maps).

Input of a map:

- a row of the logical table

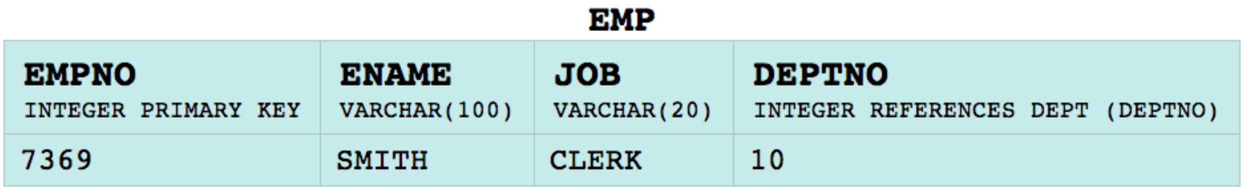
Output of a map: for each row,

- a subject resource (IRI or blank node), often generated from primary key values
- several triples with the same subject, but varying predicates and objects, generated from the attributes of the row

Triples Maps (cntd)

Idea: triples are produced by:

- subject maps
- predicate maps
- object maps.



EMP

EMPNO	ENAME	JOB	DEPTNO
INTEGER PRIMARY KEY	VARCHAR(100)	VARCHAR(20)	INTEGER REFERENCES DEPT (DEPTNO)
7369	SMITH	CLERK	10

Example

- The subject IRI is generated from the empno column by the template

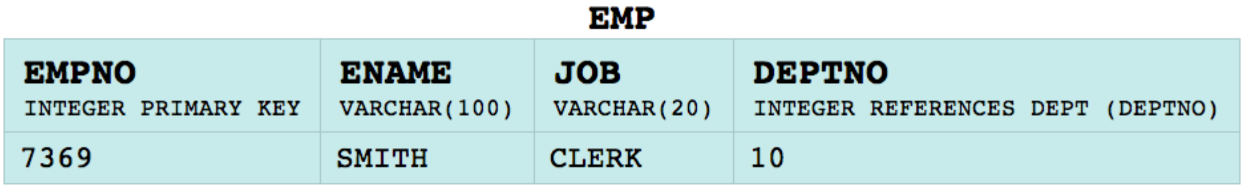
  http://data.example.com/employee/{empno}
- The predicate IRI is the constant ex:name
- The object is the literal "SMITH”, that is copied from the ENAME column

Output Graph

- By default, all RDF triples are in the default graph of the output dataset.

- A triples map can contain graph maps that place some or all of the triples into named graphs instead.

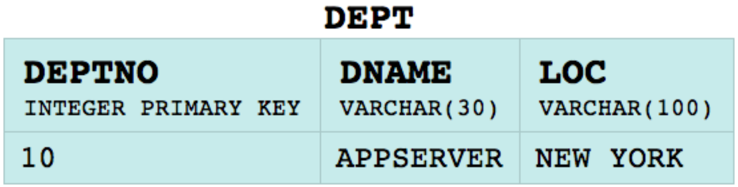
Example



EMP

EMPNO INTEGER PRIMARY KEY	**ENAME** VARCHAR(100)	**JOB** VARCHAR(20)	**DEPTNO** INTEGER REFERENCES DEPT (DEPTNO)
7369	SMITH	CLERK	10

Relational tables



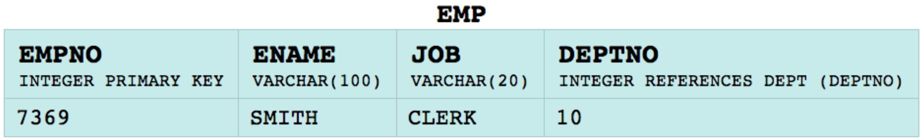
DEPT

DEPTNO INTEGER PRIMARY KEY	**DNAME** VARCHAR(30)	**LOC** VARCHAR(100)
10	APPSERVER	NEW YORK

Set of RDF triples

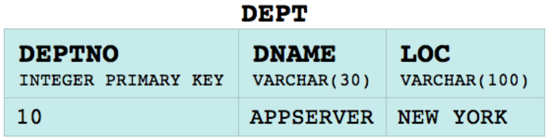
```
<http://data.example.com/employee/7369> rdf:type ex:Employee.
<http://data.example.com/employee/7369> ex:name "SMITH".
<http://data.example.com/employee/7369> ex:department <http://data.example.com/department/10>.

<http://data.example.com/department/10> rdf:type ex:Department.
<http://data.example.com/department/10> ex:name "APPSERVER".
<http://data.example.com/department/10> ex:location "NEW YORK".
<http://data.example.com/department/10> ex:staff 1.
```

Features of the Example



EMP

EMPNO	ENAME	JOB	DEPTNO
INTEGER PRIMARY KEY	VARCHAR(100)	VARCHAR(20)	INTEGER REFERENCES DEPT (DEPTNO)
7369	SMITH	CLERK	10



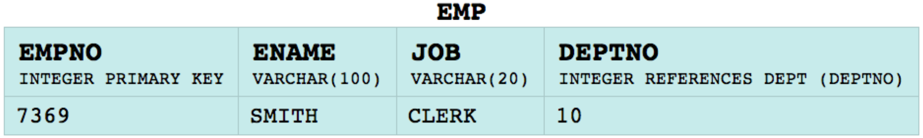
DEPT

DEPTNO	DNAME	LOC
INTEGER PRIMARY KEY	VARCHAR(30)	VARCHAR(100)
10	APPSERVER	NEW YORK

```
<http://data.example.com/employee/7369> rdf:type ex:Employee.
<http://data.example.com/employee/7369> ex:name "SMITH".
<http://data.example.com/employee/7369> ex:department <http://
data.example.com/department/10>.

<http://data.example.com/department/10> rdf:type ex:Department.
<http://data.example.com/department/10> ex:name "APPSERVER".
<http://data.example.com/department/10> ex:location "NEW YORK".
<http://data.example.com/department/10> ex:staff 1.
```

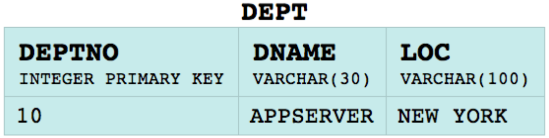
- Subjects are instances of classes from a general vocabulary
- Properties are from the same general vocabulary
- IRIs contain neither table nor column names
- The foreign key from EMP to DEPT is translated into a single property (no duplication into value and reference)
- The department resource has an additional property ex:staff, which contains the number of employees of the department

Mapping a Table



EMP

EMPNO INTEGER PRIMARY KEY	ENAME VARCHAR(100)	JOB VARCHAR(20)	DEPTNO INTEGER REFERENCES DEPT (DEPTNO)
7369	SMITH	CLERK	10



DEPT

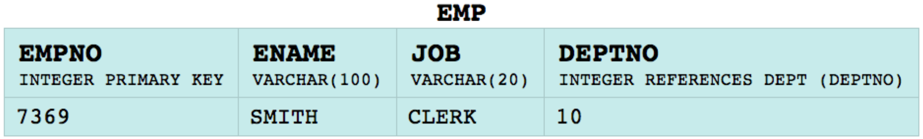
DEPTNO INTEGER PRIMARY KEY	DNAME VARCHAR(30)	LOC VARCHAR(100)
10	APPSERVER	NEW YORK

Result

```
<http://data.example.com/employee/7369> rdf:type ex:Employee.
<http://data.example.com/employee/7369> ex:name "SMITH".
<http://data.example.com/employee/7369> ex:department
                                            <http://data.example.com/department/10>.

<http://data.example.com/department/10> rdf:type ex:Department.
<http://data.example.com/department/10> ex:name "APPSERVER".
<http://data.example.com/department/10> ex:location "NEW YORK".
<http://data.example.com/department/10> ex:staff 1.
```

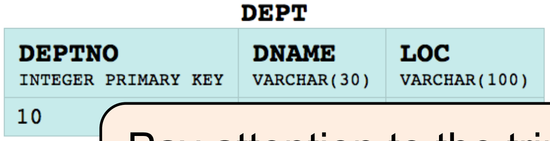
```
@prefix rr: <http://www.w3.org/ns/r2rml#>.
@prefix ex: <http://example.com/ns#>.

<#TriplesMap1>
    rr:logicalTable [ rr:tableName "EMP" ];
    rr:subjectMap [
        rr:template "http://data.example.com/employee/{EMPNO}";
        rr:class ex:Employee;
    ];
    rr:predicateObjectMap [
        rr:predicate ex:name;
        rr:objectMap [ rr:column "ENAME" ];
    ].
```

R2RML Views



EMP

EMPNO INTEGER PRIMARY KEY	ENAME VARCHAR(100)	JOB VARCHAR(20)	DEPTNO INTEGER REFERENCES DEPT (DEPTNO)
7369	SMITH	CLERK	10



DEPT

DEPTNO INTEGER PRIMARY KEY	DNAME VARCHAR(30)	LOC VARCHAR(100)
10		

Pay attention to the triple quotes:

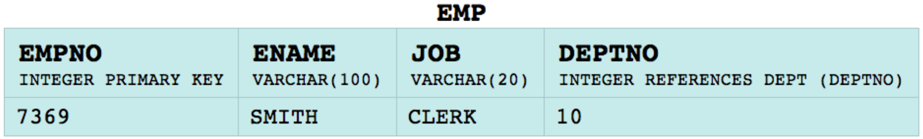
- needed for literals with linebreaks

```
<http://data.example.com/employee/7369> rdf:type ex:Employee.
<http://data.example.com/employee/7369> ex:name "SMITH".
com/employee/7369> ex:department <http://
partment/10>.

example.com/department/10> rdf:type ex:Department.
a.example.com/department/10> ex:name "APPSERVER".
data.example.com/department/10> ex:location "NEW YORK".
<    p://data.example.com/department/10> ex:staff 1.
```

```
<#DeptTableView> rr:sqlQuery """
SELECT DEPTNO,
       DNAME,
       LOC,
       (SELECT COUNT(*) FROM EMP WHERE EMP.DEPTNO=DEPT.DEPTNO)
       AS STAFF
FROM DEPT;
""".
```

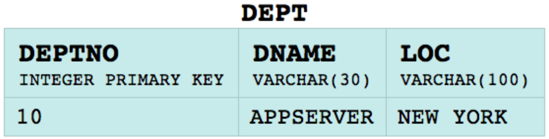
View definition

Views



EMP

EMPNO INTEGER PRIMARY KEY	ENAME VARCHAR(100)	JOB VARCHAR(20)	DEPTNO INTEGER REFERENCES DEPT (DEPTNO)
7369	SMITH	CLERK	10



DEPT

DEPTNO INTEGER PRIMARY KEY	DNAME VARCHAR(30)	LOC VARCHAR(100)
10	APPSERVER	NEW YORK

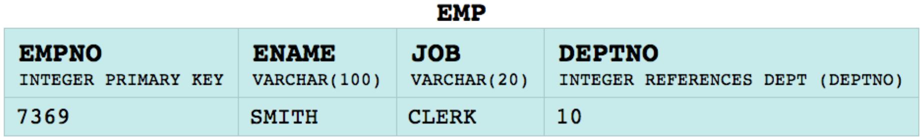
Result

```
<http://data.example.com/employee/7369> rdf:type ex:Employee.
<http://data.example.com/employee/7369> ex:name "SMITH".
<http://data.example.com/employee/7369> ex:department <http://data.example.com/
department/10>.

<http://data.example.com/department/10> rdf:type ex:Department.
<http://data.example.com/department/10> ex:name "APPSERVER".
<http://data.example.com/department/10> ex:location "NEW YORK".
<http://data.example.com/department/10> ex:staff 1.
```

```
<#TriplesMap2>
   rr:logicalTable <#DeptTableView>;
   rr:subjectMap [
      rr:template "http://data.example.com/department/{DEPTNO}";
      rr:class ex:Department;
   ];
   rr:predicateObjectMap [
      rr:predicate ex:name;
      rr:objectMap [ rr:column "DNAME" ];
   ];
   rr:predicateObjectMap [
      rr:predicate ex:location;
      rr:objectMap [ rr:column "LOC" ];
   ];
   rr:predicateObjectMap [
      rr:predicate ex:staff;
      rr:objectMap [ rr:column "STAFF" ];
   ].
```

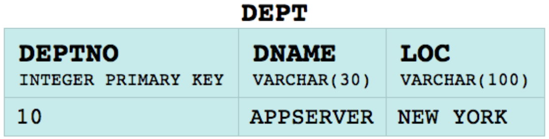
Mapping to a View Definition

Linking Two Logical Tables



EMP

EMPNO INTEGER PRIMARY KEY	**ENAME** VARCHAR(100)	**JOB** VARCHAR(20)	**DEPTNO** INTEGER REFERENCES DEPT (DEPTNO)
7369	SMITH	CLERK	10



DEPT

DEPTNO INTEGER PRIMARY KEY	**DNAME** VARCHAR(30)	**LOC** VARCHAR(100)
10	APPSERVER	NEW YORK

Result

```
<http://data.example.com/employee/7369> rdf:type ex:Employee.
<http://data.example.com/employee/7369> ex:name "SMITH".
<http://data.example.com/employee/7369> ex:department <http://data.example.com/
department/10>.

<http://data.example.com/department/10> rdf:type ex:Department.
<http://data.example.com/department/10> ex:name "APPSERVER".
<http://data.example.com/department/10> ex:location "NEW YORK".
<http://data.example.com/department/10> ex:staff 1.
```

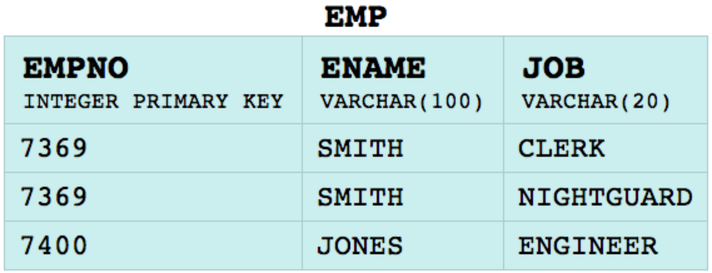
```
<#TriplesMap1>
    rr:predicateObjectMap [
        rr:predicate ex:department;
        rr:objectMap [
            rr:parentTriplesMap <#TriplesMap2>;
            rr:joinCondition [
                rr:child "DEPTNO";
                rr:parent "DEPTNO";
            ];
        ];
    ].
```

Linking Two Logical Tables: Features

- Additonal predicate object map for <#TriplesMap1>
- Object map retrieves subject from parent triples map by joining along a foreign key relationship
- It joins
  - the current row of the logical table
  - with the row of the logical table of <#TriplesMap1> that satisfies the join condition
- Note:
  - child = referencing map
  - parent = referenced map

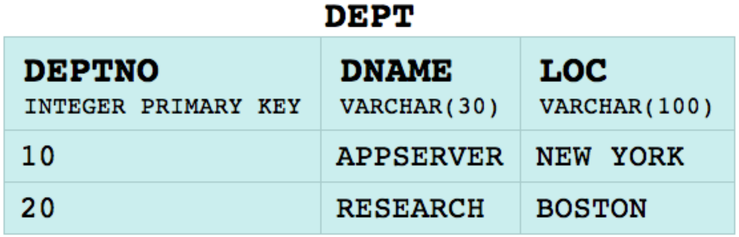
```
<#TriplesMap1>
    rr:predicateObjectMap [
        rr:predicate ex:department;
        rr:objectMap [
            rr:parentTriplesMap <#TriplesMap2>;
            rr:joinCondition [
                rr:child "DEPTNO";
                rr:parent "DEPTNO";
            ];
        ];
    ].
```

Many to Many Relationship: Approach 1



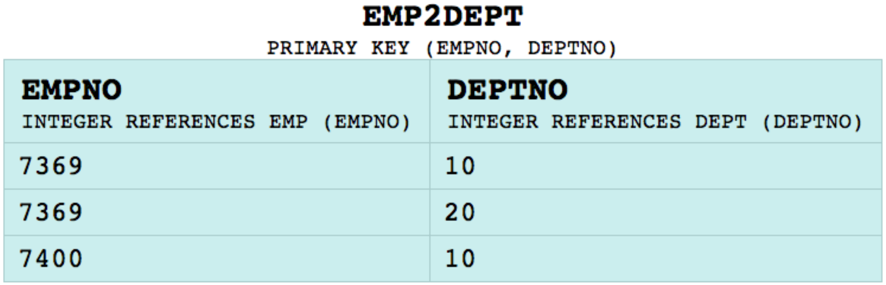
EMP

EMPNO INTEGER PRIMARY KEY	ENAME VARCHAR(100)	JOB VARCHAR(20)
7369	SMITH	CLERK
7369	SMITH	NIGHTGUARD
7400	JONES	ENGINEER



DEPT

DEPTNO INTEGER PRIMARY KEY	DNAME VARCHAR(30)	LOC VARCHAR(100)
10	APPSERVER	NEW YORK
20	RESEARCH	BOSTON



EMP2DEPT

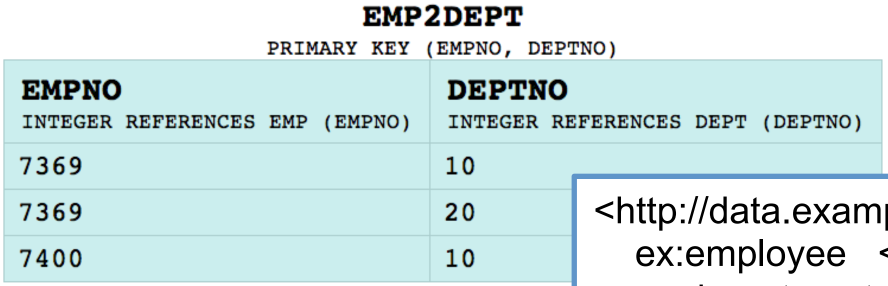
PRIMARY KEY (EMPNO, DEPTNO)

EMPNO INTEGER REFERENCES EMP (EMPNO)	DEPTNO INTEGER REFERENCES DEPT (DEPTNO)
7369	10
7369	20
7400	10

Direct mapping style output

```
<http://data.example.com/employee=7369/department=10>
   ex:employee   <http://data.example.com/employee/7369> ;
   ex:department <http://data.example.com/department/10> .

<http://data.example.com/employee=7369/department=20>
   ex:employee <http://data.example.com/employee/7369> ;
   ex:department <http://data.example.com/department/20> .

<http://data.example.com/employee=7400/department=10>
   ex:employee <http://data.example.com/employee/7400> ;
   ex:department <http://data.example.com/department/10> .
```

Many to Many Relationship: Approach 1



EMP2DEPT

PRIMARY KEY (EMPNO, DEPTNO)

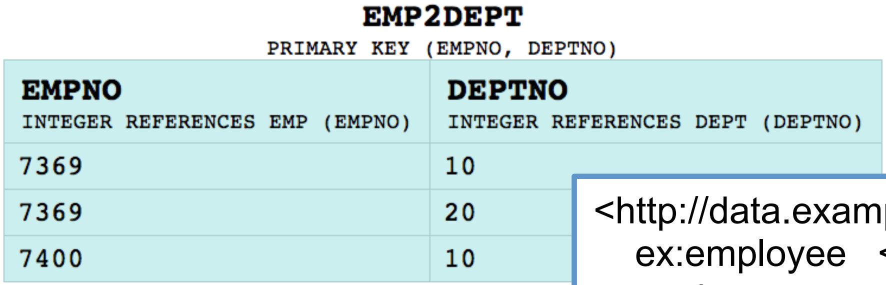
EMPNO INTEGER REFERENCES EMP (EMPNO)	DEPTNO INTEGER REFERENCES DEPT (DEPTNO)
7369	10
7369	20
7400	10

```
<http://data.example.com/employee=7369/department=10>
    ex:employee   <http://data.example.com/employee/7369> ;
    ex:department <http://data.example.com/department/10> .
```

```
<#TriplesMap3>
    rr:logicalTable [ rr:tableName "EMP2DEPT" ];
    rr:subjectMap [ rr:template "http://data.example.com/employee={EMPNO}/department={DEPTNO}" ];
    rr:predicateObjectMap [
        rr:predicate ex:employee;
        rr:objectMap [ rr:template "http://data.example.com/employee/{EMPNO}" ];
    ];
    rr:predicateObjectMap [
        rr:predicate ex:department;
        rr:objectMap [ rr:template "http://data.example.com/department/{DEPTNO}" ];
    ].
```

The mapping

Many to Many Relationship: Approach 1



EMP2DEPT

PRIMARY KEY (EMPNO, DEPTNO)

EMPNO INTEGER REFERENCES EMP (EMPNO)	**DEPTNO** INTEGER REFERENCES DEPT (DEPTNO)
7369	10
7369	20
7400	10

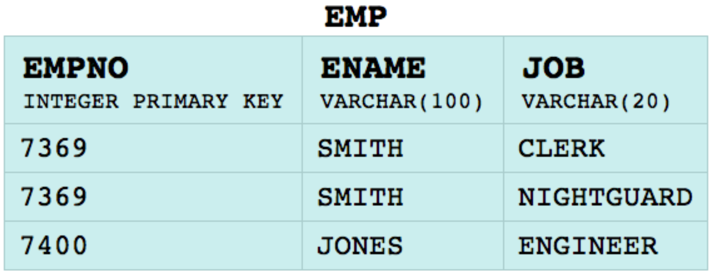
```
<http://data.example.com/employee=7369/department=10>
    ex:employee    <http://data.example.com/employee/7369> ;
    ex:department <http://data.example.com/department/10> .
```

```
<#TriplesMap3>
    rr:logicalTable [ rr:tableName "EMP2DEPT
    rr:subjectMap [ rr:template "http://data.exa                                    " ];
    rr:predicateObjectMap [
        rr:predicate ex:employee;
        rr:objectMap [ rr:template "http://data.example.com/employee/{EMPNO}" ];
    ];
    rr:predicateObjectMap [
        rr:predicate ex:department;
        rr:objectMap [ rr:template "http://data.example.com/department/{DEPTNO}" ];
    ].
```

Note: this models the case where the subject identifies each row (composite)

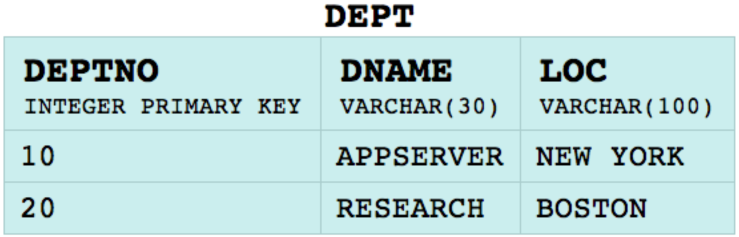
The mapping

Many to Many Relationship: Approach 2



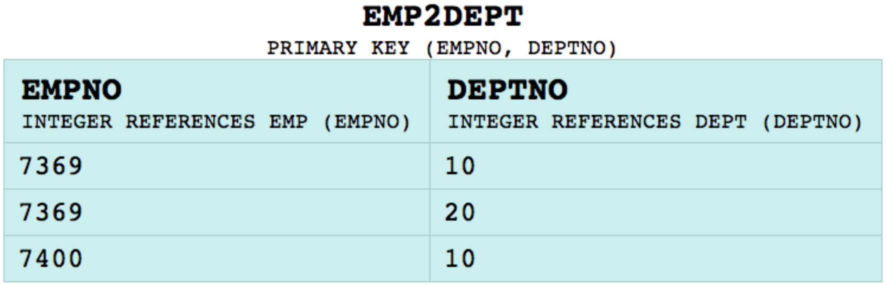
EMP

EMPNO INTEGER PRIMARY KEY	**ENAME** VARCHAR(100)	**JOB** VARCHAR(20)
7369	SMITH	CLERK
7369	SMITH	NIGHTGUARD
7400	JONES	ENGINEER



DEPT

DEPTNO INTEGER PRIMARY KEY	**DNAME** VARCHAR(30)	**LOC** VARCHAR(100)
10	APPSERVER	NEW YORK
20	RESEARCH	BOSTON



EMP2DEPT

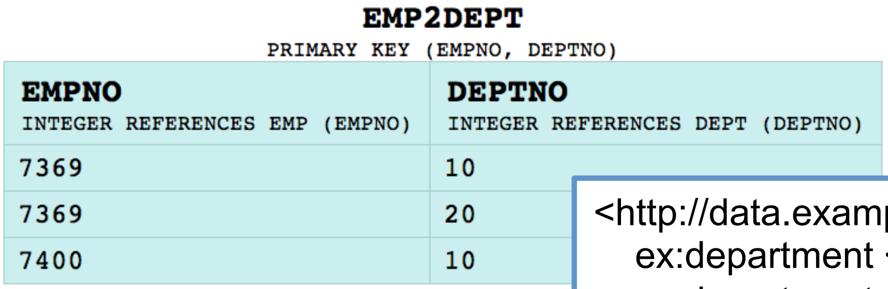
PRIMARY KEY (EMPNO, DEPTNO)

EMPNO INTEGER REFERENCES EMP (EMPNO)	**DEPTNO** INTEGER REFERENCES DEPT (DEPTNO)
7369	10
7369	20
7400	10

Expected output

```
<http://data.example.com/employee/7369>
    ex:department <http://data.example.com/department/10> ;
    ex:department <http://data.example.com/department/20> .

<http://data.example.com/employee/7400>
    ex:department <http://data.example.com/department/10>.
```

Many to Many Relationship: Approach 2



EMP2DEPT

PRIMARY KEY (EMPNO, DEPTNO)

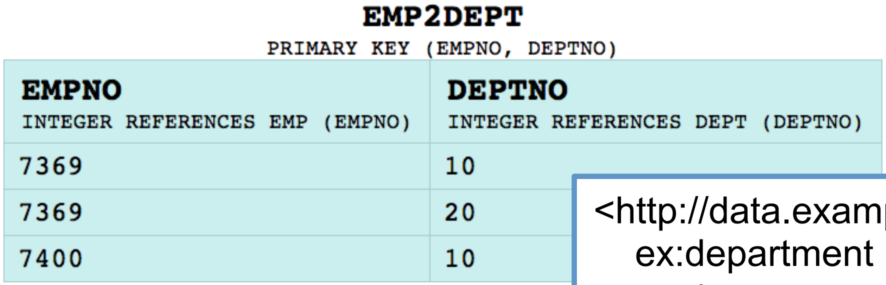
EMPNO INTEGER REFERENCES EMP (EMPNO)	**DEPTNO** INTEGER REFERENCES DEPT (DEPTNO)
7369	10
7369	20
7400	10

```
<http://data.example.com/employee/7369>
   ex:department <http://data.example.com/department/10> ;
   ex:department <http://data.example.com/department/20> .

<http://data.example.com/employee/7400>
   ex:department <http://data.example.com/department/10>.
```

```
<#TriplesMap3>
   rr:logicalTable [ rr:tableName "EMP2DEPT" ];
   rr:subjectMap [
       rr:template "http://data.example.com/employee/{EMPNO}";
   ];
   rr:predicateObjectMap [
     rr:predicate ex:department;
     rr:objectMap [ rr:template "http://data.example.com/department/{DEPTNO}" ];
   ].
```

The mapping

Many to Many Relationship: Approach 2



EMP2DEPT

PRIMARY KEY (EMPNO, DEPTNO)

EMPNO INTEGER REFERENCES EMP (EMPNO)	**DEPTNO** INTEGER REFERENCES DEPT (DEPTNO)
7369	10
7369	20
7400	10

```
<http://data.example.com/employee/7369>
   ex:department <http://data.example.com/department/10> ;
   ex:department <http://data.example.com/department/20> .

<http://data.example.com/employee/7400>
   ex:department <http://data.example.com/department/10>.
```

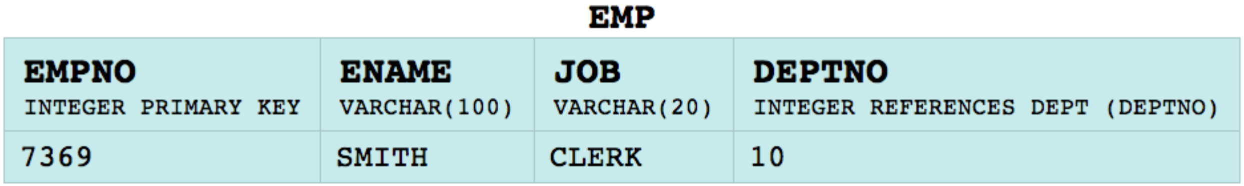
```
<#TriplesMap3>
   rr:logicalTable [ rr:tableName "EMP2DEPT" ];
   rr:subjectMap [
      rr:template "http://data.example.com/employee/{EMPNO}";
   ];
   rr:predicateObjectMap [
     rr:predicate ex:department;
     rr:objectMap [ rr:template "http://data.example.com/department/{DEPTNO}" ];
   ].
```

The mapping

Translating Job Codes to IRIs

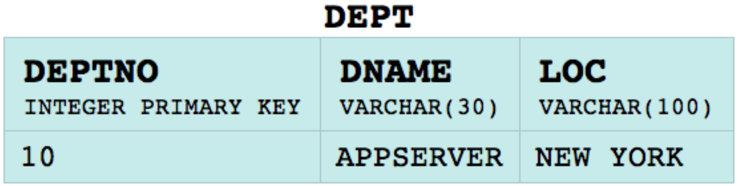
Assume the following correspondance:

CLERK	http://data.example.com/roles/general-office
NIGHTGUARD	http://data.example.com/roles/security
ENGINEER	http://data.example.com/roles/engineering



EMP

EMPNO	ENAME	JOB	DEPTNO
INTEGER PRIMARY KEY	VARCHAR(100)	VARCHAR(20)	INTEGER REFERENCES DEPT (DEPTNO)
7369	SMITH	CLERK	10

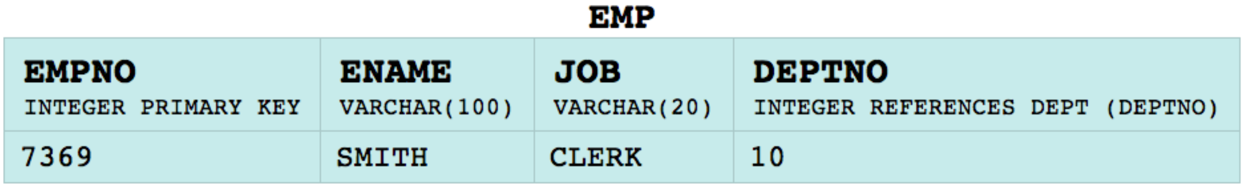


DEPT

DEPTNO	DNAME	LOC
INTEGER PRIMARY KEY	VARCHAR(30)	VARCHAR(100)
10	APPSERVER	NEW YORK

```
<http://data.example.com/employee/7369> ex:role
                    <http://data.example.com/roles/general-office>.
```

Translating Job Codes to IRIs



EMP

EMPNO INTEGER PRIMARY KEY	ENAME VARCHAR(100)	JOB VARCHAR(20)	DEPTNO INTEGER REFERENCES DEPT (DEPTNO)
7369	SMITH	CLERK	10

```
<#TriplesMap1>
    rr:logicalTable [ rr:sqlQuery
        """
        SELECT EMP.*,
                       (CASE JOB
                             WHEN 'CLERK' THEN 'general-office'
                             WHEN 'NIGHTGUARD' THEN 'security'
                             WHEN 'ENGINEER' THEN 'engineering'
                         END) AS ROLE
        FROM EMP
        """ ];
    rr:subjectMap [
        rr:template "http://data.example.com/employee/{EMPNO}";
    ];
    rr:predicateObjectMap [
        rr:predicate ex:role;
        rr:objectMap [ rr:template "http://data.example.com/roles/{ROLE}" ];
    ].
```

- Overview and Examples
- Detailed Specification

R2RML Processors and Mapping Documents

- An R2RML mapping
  - defines a mapping from a relational database to RDF
  - consists of one or more triples maps.

  The input is called the input database.

- An R2RML processor,
  - given an R2RML mapping and an input database, provides access to the output dataset;
  - has access to an execution environment with:
    - an SQL connection to the input database,
    - a base IRI

- An R2RML processor may include an R2RML data validator

Data Errors

The RDF data produced by a mapping may be erroneous, due to the format and type of the data in the database.

Two cases:

- The map produces a term of type rr:IRI, but the term is not a valid IRI
- The map is intended to produce a literal, but the mapping specifies a datatype that overrides the natural RDF data type (there is a specific correspondence between SQL and RDF datatypes)

Data Errors

The RDF data produced by a mapping may be erroneous, due to the format and type of the data in the database.

Two cases:

- The map produces a term of type rr:IRI,
but the
- The ma
but the
natural
(there is
RDF dat

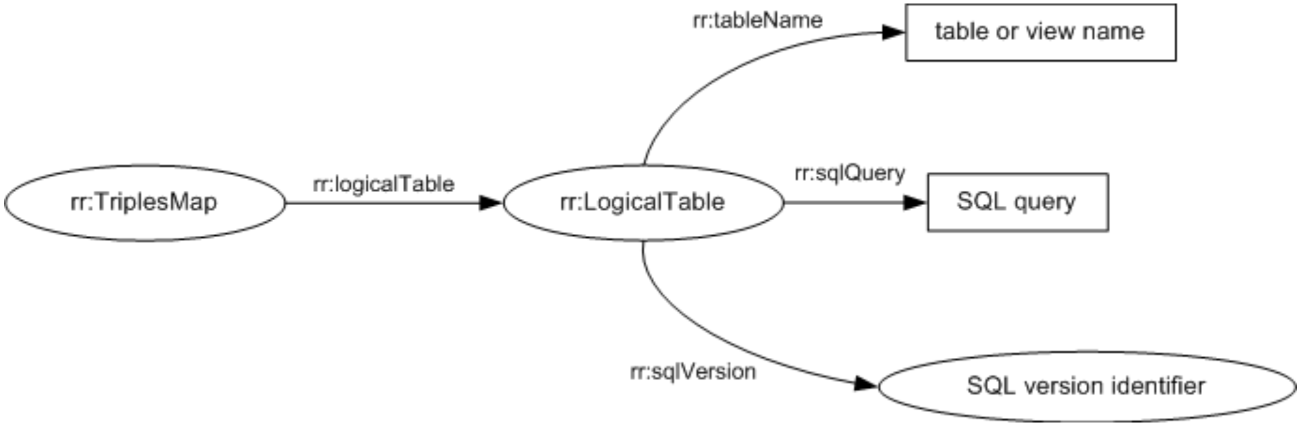
Data errors cannot generally be detected by analyzing the table schema of the database, but only by scanning the data in the tables. For large and rapidly changing databases, this can be impractical. Therefore, R2RML processors are allowed to answer queries that do not “touch” a data error, and the behavior of such operations is well-defined. For the same reason, the conformance of R2RML mappings is defined without regard for the presence of data errors.

Source: R2RML: RDB to RDF Mapping Language
W3C Recommendation

Direct Mapping as Default Mappings

- An R2RML processor may include an R2RML default mapping generator
  - Output: Direct Graph corresponding to the input database (Direct Mapping).
- No duplicate row preservation: For tables without a primary key, the Direct Graph requires that a fresh blank node is created for each row. This ensures that duplicate rows in such tables are preserved. This requirement is relaxed for R2RML default mappings: They may reuse the same blank node for multiple duplicate rows. This behavior does not preserve duplicate rows.

Logical Tables



- A logical table is a tabular SQL query result that is to be mapped to RDF triples. It is either
  - a SQL base table or view, or
  - an R2RML view.
- Every logical table has an effective SQL query
  - if executed over the SQL connection,
    it produces the contents of the logical table

Base Tables and SQL Views (rr:tableName)

- A SQL base table or view is a logical table containing SQL data from a base table or view in the input database. A SQL base table or view is represented by a resource that has exactly one rr:tableName property.

  The value of rr:tableName specifies the table or view name of the base table or view. Its value must be a valid schema-qualified name that names an existing base table or view in the input database.
- The effective SQL query of a SQL base table or view is:

```
SELECT * FROM {table}
```

Example of Mapping from a Base Table

```
@prefix rr: <http://www.w3.org/ns/r2rml#>.
@prefix ex: <http://example.com/ns#>.

<#TriplesMap1>
    rr:logicalTable [ rr:tableName "EMP" ];
    rr:subjectMap [
        rr:template "http://data.example.com/employee/{EMPNO}";
        rr:class ex:Employee;
    ];
    rr:predicateObjectMap [
        rr:predicate ex:name;
        rr:objectMap [ rr:column "ENAME" ];
    ].
```

R2RML Views (rr:sqlQuery, rr:sqlVersion)

An **R2RML view** is a **logical table** whose contents are the result of executing a SQL query against the input database.

It is represented by a **resource** that has exactly **one rr:sqlQuery property**, **whose value** is a literal with a lexical form that is a **valid SQL query**.

Data transformation

- R2RML mappings sometimes require **data transformation**, computation, or filtering before generating triples from the database.
- This can be achieved by defining a SQL view in the input database and referring to it with rr:tableName.
- However, this approach may sometimes not be practical for lack of database privileges or other reasons.
- R2RML views achieve the same effect without requiring changes to the input database.

R2RML Views (rr:sqlQuery)

- No duplicated columns allowed:

  ```
  SELECT EMP.DEPTNO, 1 AS DEPTNO FROM EMP;
  ```

- Unnamed columns are not recommended

  ```
  SELECT DEPTNO,
                 COUNT(EMPNO)
  FROM EMP
  GROUP BY DEPTNO;
  ```

Example of Mapping from View

```
<#DeptTableView> rr:sqlQuery """
SELECT DEPTNO,
       DNAME,
       LOC,
       (SELECT COUNT(*) FROM EMP WHERE EMP.DEPTNO=DEPT.DEPTNO) AS
STAFF
FROM DEPT;
""".
```

```
<#TriplesMap2>
   rr:logicalTable <#DeptTableView>;
   rr:subjectMap [
      rr:template "http://data.example.com/department/{DEPTNO}";
      rr:class ex:Department;
   ];
   rr:predicateObjectMap [
      rr:predicate ex:name;
      rr:objectMap [ rr:column "DNAME" ];
   ];
rr:predicateObjectMap [
      rr:predicate ex:staff;
      rr:objectMap [ rr:column "STAFF" ];
   ].
```

Version Identifiers (rr:sqlVersion)

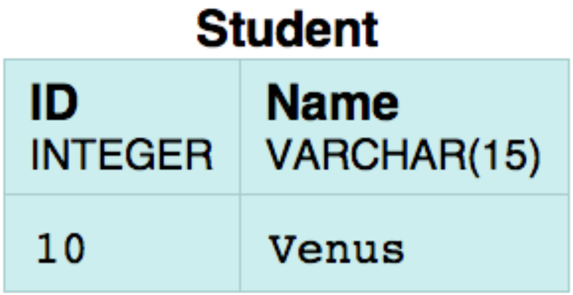
- An R2RML view may have one or more SQL version identifiers. They must be valid IRIs and are represented as values of the rr:sqlVersion property. The following SQL version identifier indicates that the SQL query conforms to Core SQL 2008: http://www.w3.org/ns/r2rml#SQL2008
- The absence of a SQL version identifier indicates that no claim to Core SQL 2008 conformance is made.
- Additional identifiers, not normative, an be found at: http://www.w3.org/2001/sw/wiki/RDB2RDF/SQL_Version_IRIs

Example

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix ex: <http://example.com/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#
@base <http://example.com/base/> .

<TriplesMap1>
     a rr:TriplesMap;

    rr:logicalTable [
                    rr:sqlQuery """
                      SELECT "ID”, "Name” FROM "Student" """;
                  rr:sqlVersion rr:SQL2008
                  ];
    rr:subjectMap [ rr:template "http://example.com/{\"ID\"}/{\"Name\"}";  ];
    rr:predicateObjectMap
    [
      rr:predicate           foaf:name ;
      rr:objectMap           [ rr:column "\"Name\"" ]
    ]
    .
```

Pay attention to SQL identifiers in double quotes:

- “delimited” identifiers

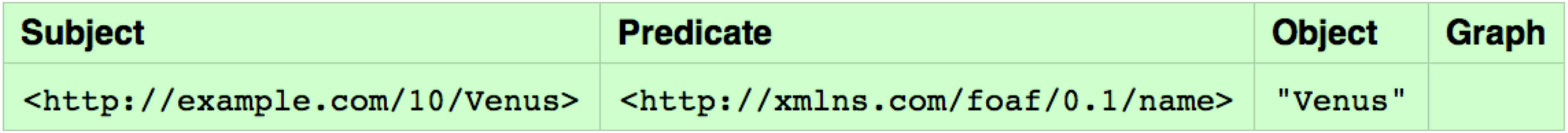


Student

ID INTEGER	Name VARCHAR(15)
10	Venus

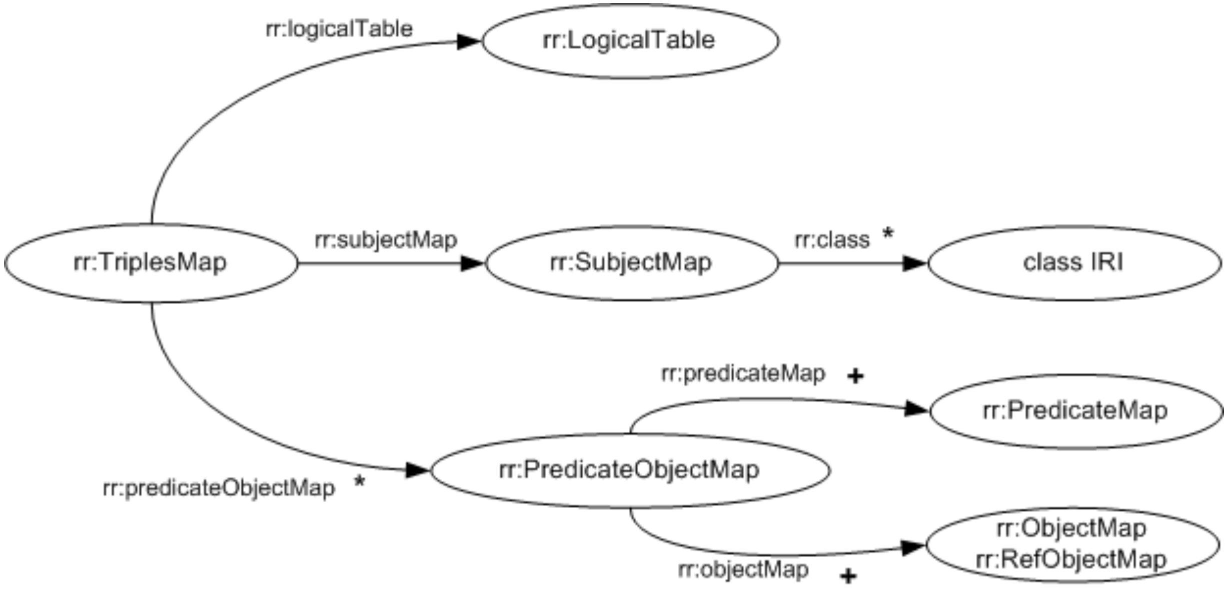
Pay attention to the backslash quotes:

- escape characters in “flat” literals



Subject	Predicate	Object	Graph
`<http://example.com/10/Venus>`	`<http://xmlns.com/foaf/0.1/name>`	`"Venus"`	

Mapping Logical Tables to RDF with Triples Maps



A **triples map** specifies a rule for translating each row of a logical table to zero or more RDF triples.

Mapping Logical Tables to RDF with Triples Maps

The **RDF triples** generated from **one row** in the logical table **all share the same subject**.

A triples map is represented by a resource that references the following other resources:

- It must **have exactly one rr:logicalTable** property. Its value is a logical table that specifies a SQL query result to be mapped to triples.
- It must have **exactly one subject map** that specifies how to generate a subject for each row of the logical table. It may be specified in two ways:
  - using the rr:subjectMap property, whose value must be the subject map, or
  - using the constant shortcut property rr:subject.
- It may have **zero or more rr:predicateObjectMap properties**, whose values must be predicate-object maps. They specify **pairs of predicate maps and object maps** that, together with the subjects generated by the subject map, may form one or more RDF triples for each row.

Mapping Logical Tables to RDF with Triples Maps

```
[]
   rr:logicalTable [ rr:tableName "DEPT" ];
   rr:subjectMap [ rr:template "http://data.example.com/department/{DEPTNO}" ];
   rr:predicateObjectMap [
      rr:predicate ex:name;
      rr:objectMap [ rr:column "DNAME" ];
   ];
   rr:predicateObjectMap [
      rr:predicate ex:location;
      rr:objectMap [ rr:column "LOC" ];
   ].
```

Mapping Logical Tables to RDF with Triples Maps

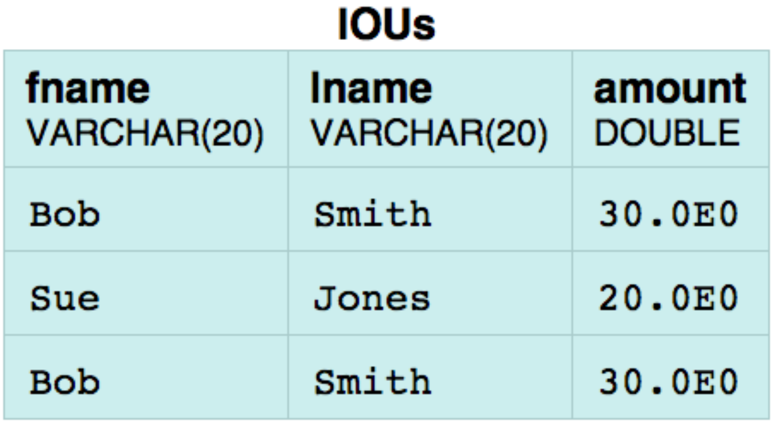
```
@prefix rr: <http://www.w3.org/ns/r2rml#>.
@prefix ex: <http://example.com/ns#>.

<#TriplesMap1>
    rr:logicalTable [ rr:tableName "EMP" ];
    rr:subjectMap [
        rr:template "http://data.example.com/employee/{EMPNO}";
        rr:class ex:Employee;
    ];
    rr:predicateObjectMap [
        rr:predicate ex:name;
        rr:objectMap [ rr:column "ENAME" ];
    ].
```

Creating Resources with Subject Maps

- A subject map is a **term map**. It specifies a rule for generating the subjects of the RDF triples generated by a triples map.
- **Term maps** are used to generate the subjects, predicates and objects of the RDF triples that are generated by a triples map. Consequently, there are several kinds of term maps, depending on where in the mapping they occur: subject maps, predicate maps, object maps and graph maps.
- A **term map** must be exactly one of the following:
  - a constant-valued term map,
  - a column-valued term map,
  - a template-valued term map.

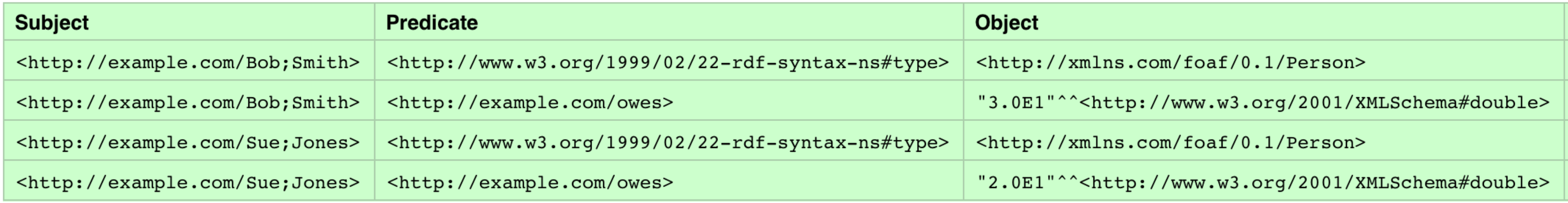
Example with Template

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix ex: <http://example.com/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@base <http://example.com/base/> .

<TriplesMap1>
   a rr:TriplesMap;

   rr:logicalTable [ rr:tableName "\"IOUs\"" ];

   rr:subjectMap [ rr:template "http://example.com/{\"fname\"};{\"lname\"}";
                rr:class foaf:Person ];

   rr:predicateObjectMap
   [
     rr:predicate                ex:owes ;
     rr:objectMap                [ rr:column "\"amount\""; ]
   ];

   .
```



IOUs

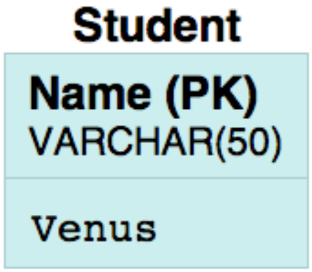
fname VARCHAR(20)	**lname** VARCHAR(20)	**amount** DOUBLE
Bob	Smith	30.0E0
Sue	Jones	20.0E0
Bob	Smith	30.0E0



Subject	Predicate	Object
<http://example.com/Bob;Smith>	<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>	<http://xmlns.com/foaf/0.1/Person>
<http://example.com/Bob;Smith>	<http://example.com/owes>	"3.0E1"^^<http://www.w3.org/2001/XMLSchema#double>
<http://example.com/Sue;Jones>	<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>	<http://xmlns.com/foaf/0.1/Person>
<http://example.com/Sue;Jones>	<http://example.com/owes>	"2.0E1"^^<http://www.w3.org/2001/XMLSchema#double>

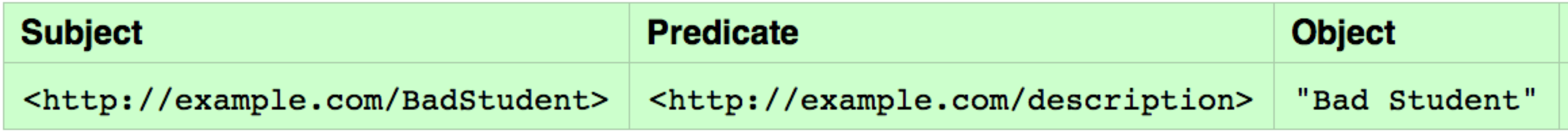
Example with constants

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix ex: <http://example.com/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@base <http://example.com/base/> .

<TriplesMap1>
    a rr:TriplesMap;

    rr:logicalTable [ rr:tableName "\"Student\"" ];
    rr:subjectMap [ rr:constant ex:BadStudent ] ;

    rr:predicateObjectMap
    [
      rr:predicateMap [ rr:constant ex:description ];
      rr:objectMap    [ rr:constant "Bad Student"; ]
    ]
    .
```



Student

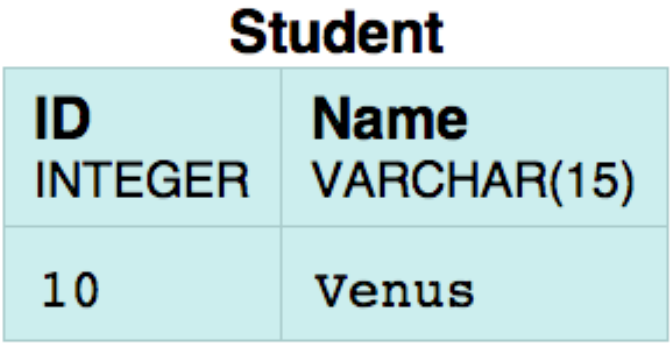
Name (PK) VARCHAR(50)
Venus



Subject	Predicate	Object
`<http://example.com/BadStudent>`	`<http://example.com/description>`	`"Bad Student"`

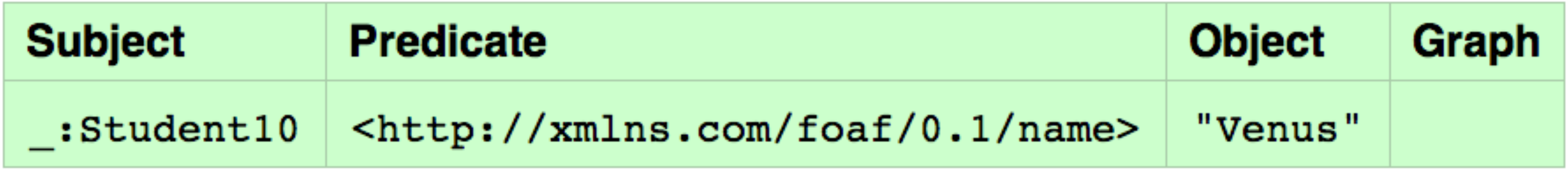
Creating Resources with Subject Maps

```
<TriplesMap1>
    a rr:TriplesMap;

    rr:logicalTable [
              rr:sqlQuery ""”
                    Select ('Student' || "ID" ) AS StudentId , "ID”, "Name”
                    From "Student"
                                ""”  ];
    rr:subjectMap [ rr:column "StudentId"; rr:termType rr:BlankNode; ];
    rr:predicateObjectMap
    [
      rr:predicate                    foaf:name ;
      rr:objectMap                    [ rr:column "\"Name\"" ]
    ]
    .
```



Student

ID INTEGER	Name VARCHAR(15)
10	Venus



Subject	Predicate	Object	Graph
_:Student10	<http://xmlns.com/foaf/0.1/name>	"Venus"	

Creating Properties and Values with Predicate-Object Maps

- A **predicate-object map** is a function that creates one or more predicate-object pairs for each logical table row of a logical table.
- It is used in conjunction with a subject map to generate RDF triples in a triples map.

Creating Properties and Values with Predicate-Object Maps

A predicate-object map is represented by a resource that references the following other resources:

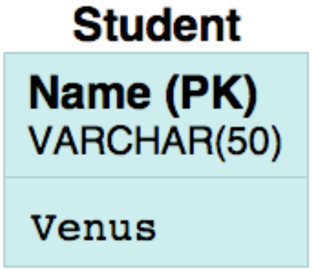
- **One or more predicate maps**. Each of them may be specified in one of two ways:
  - using the **rr:predicateMap** property, whose value must be a **predicate map**, or
  - using the **constant shortcut property** rr:predicate.
- **One or more object maps** or **referencing object maps**. Each of them may be specified in one of two ways:
  - using the **rr:objectMap** property, whose value must be either an **object map**, or a **referencing object map**.
  - using the **constant shortcut property** rr:object.

A **predicate map** is a term map.

An **object map** is a term map.

Example with Constants

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix ex: <http://example.com/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@base <http://example.com/base/> .

<TriplesMap1>
    a rr:TriplesMap;

    rr:logicalTable [ rr:tableName "\"Student\"" ];
    rr:subjectMap [ rr:constant ex:BadStudent ] ;

    rr:predicateObjectMap
    [
      rr:predicateMap [ rr:constant ex:description ];
      rr:objectMap    [ rr:constant "Bad Student"; ]
    ]
    .
```



Student

Name (PK) VARCHAR(50)
Venus

Example with Shortcuts

```
<TriplesMap1>
     a rr:TriplesMap;

     rr:logicalTable [ rr:tableName "\"Student\""; ];
     rr:subjectMap [ rr:template "http://example.com/Student/{\"ID\"}/{\"Name\"}";
                                                            rr:graph ex:PersonGraph;
              ];

    rr:predicateObjectMap
    [
              rr:predicate          rdf:type;
              rr:object             foaf:Person;
    ];

    rr:predicateObjectMap
    [
               rr:predicate         foaf:name;
              rr:objectMap          [ rr:column "\"Name\"" ]
    ]
```

Creating Properties and Values with Predicate-Object Maps

A **referencing object map** allows using the subjects of another triples map as the objects generated by a predicate-object map. Since both triples maps may be based on different logical tables, this may require a join between the logical tables. This is not restricted to 1:1 joins.

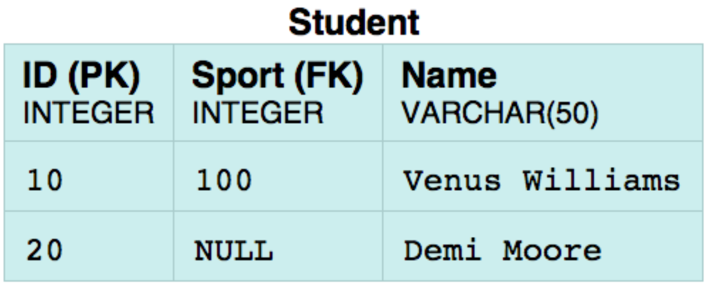
A **referencing object map** is represented by a resource that:

- has exactly one **rr:parentTriplesMap** property, whose value must be a triples map, known as the referencing object map's parent triples map.
- may have one or more **rr:joinCondition** properties, whose values must be join conditions.

A **join condition** is represented by a resource that has exactly one value for each of the following two properties:

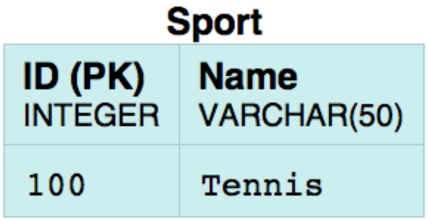
- **rr:child**, whose value is known as the join condition's child column and must be a column name that exists in the logical table of the triples map that contains the referencing object map
- **rr:parent**, whose value is known as the join condition's parent column and must be a column name that exists in the logical table of the referencing object map's parent triples map.

```
<TriplesMap1> a rr:TriplesMap;
     rr:logicalTable [ rr:tableName  "\"Student\"" ];
     rr:subjectMap [ rr:template "http://example.com/resource/student_{\"ID\"}"; ];
     rr:predicateObjectMap
    [
          rr:predicate        foaf:name ;   rr:objectMap  [ rr:column "\"Name\""; ];
    ];

    rr:predicateObjectMap
    [
      rr:predicate   <http://example.com/ontology/practises> ;
      rr:objectMap          [
          a rr:RefObjectMap ;   rr:parentTriplesMap <TriplesMap2>;
          rr:joinCondition [ rr:child "\"Sport\"" ;       rr:parent "\"ID\"" ;   ]
      ];
    ];
    .

<TriplesMap2> a rr:TriplesMap;
     rr:logicalTable [ rr:tableName  "\"Sport\"" ];
     rr:subjectMap [ rr:template "http://example.com/resource/sport_{\"ID\"}"; ];
     rr:predicateObjectMap
    [
                 rr:predicate rdfs:label ;  rr:objectMap [ rr:column "\"Name\""; ];
    ];
                 .
```



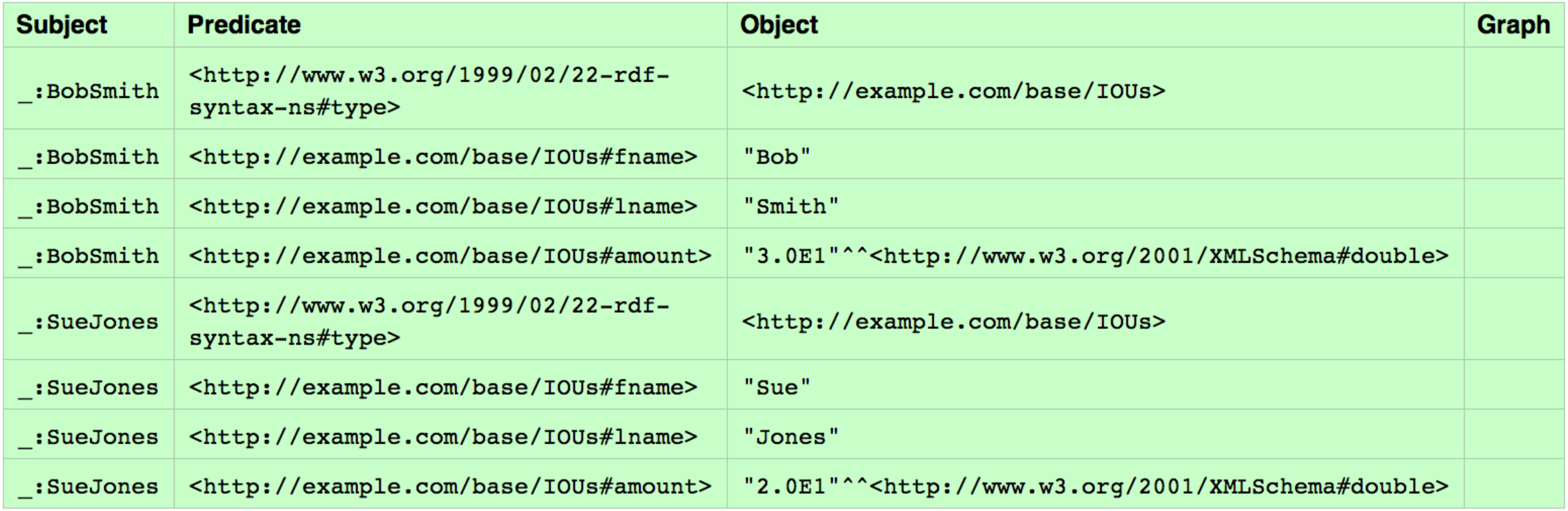
Student

ID (PK) INTEGER	Sport (FK) INTEGER	Name VARCHAR(50)
10	100	Venus Williams
20	NULL	Demi Moore



Sport

ID (PK) INTEGER	Name VARCHAR(50)
100	Tennis



Subject	Predicate	Object	Graph
_:BobSmith	<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>	<http://example.com/base/IOUs>	
_:BobSmith	<http://example.com/base/IOUs#fname>	"Bob"	
_:BobSmith	<http://example.com/base/IOUs#lname>	"Smith"	
_:BobSmith	<http://example.com/base/IOUs#amount>	"3.0E1"^^<http://www.w3.org/2001/XMLSchema#double>	
_:SueJones	<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>	<http://example.com/base/IOUs>	
_:SueJones	<http://example.com/base/IOUs#fname>	"Sue"	
_:SueJones	<http://example.com/base/IOUs#lname>	"Jones"	
_:SueJones	<http://example.com/base/IOUs#amount>	"2.0E1"^^<http://www.w3.org/2001/XMLSchema#double>	