

INF1343, Winter 2011

Data Modeling and Database Design

Yuri Takhteyev
University of Toronto



This presentation is licensed under Creative Commons Attribution License, v. 3.0. To view a copy of this license, visit <http://creativecommons.org/licenses/by/3.0/>. This presentation incorporates images from the Crystal Clear icon collection by Everaldo Coelho, available under LGPL from <http://everaldo.com/crystal/>.

Week 9

Databases and Objects

(after reviewing old material)

Schedule

Intro	Design	Interfacing	Security
SQL	Design	XML	Storage
SQL	Design	Objects	Scale

April 4: Final exam

Final Exam

Skills:

weeks 1-6

(SQL, ER, translation,
normalization)

Concepts:

weeks 1-12

E.g.: “Why does XML have
to do with databases?”

ER

RDF

Relational

OO

XML

Relational vs ER

Relational: relations (tables), records (rows), fields, PKs, FKs.

ER: entities, attributes, relationships.

Relational vs ER

ER: Conceptual + a visual language

Relational: Physical / conceptual

RM is more flexible.

(Not necessarily a good thing!)

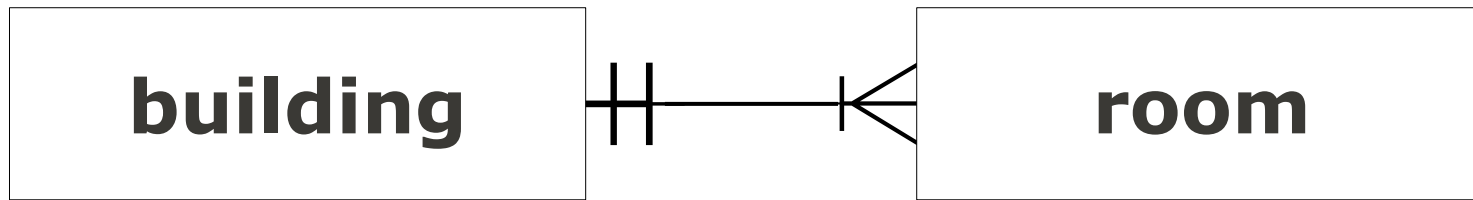
Can translate from ER into RM.

(Not always the other way around!)

ER → RM Translation

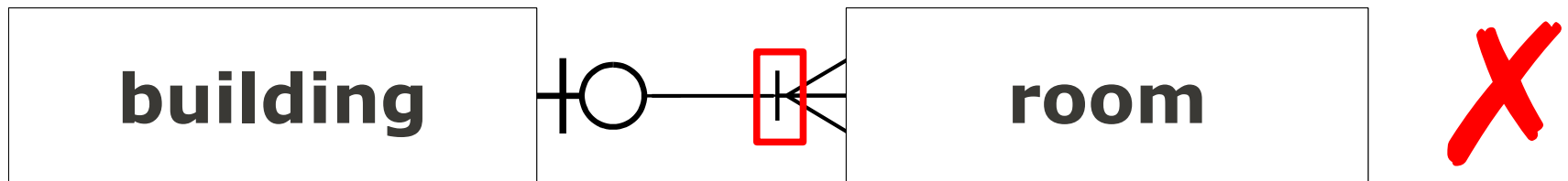
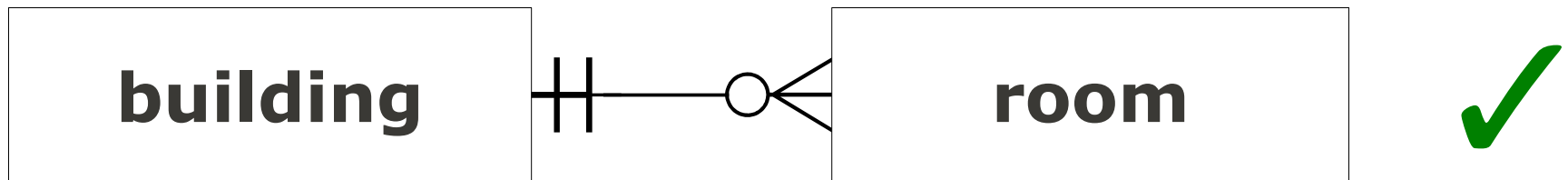
entities	→	relations (tables)
instances	→	records (rows)
attributes	→	fields
relationships	→	PK/FK constraints

Limitations



Dually mandatory relationships become problematic in RM.

Limitations



No easy way to represent the mandatory "many" end.

Comments on ER

- Reality vs representation
- Duplication of information
- Determining an instance vs a set

Comments on RM

- Candidate keys

- Tables without PKs

- 1NF, 2NF, 3NF

1NF: letter vs spirit; repeated values vs free form

2NF: dependency on a part the PK

country_code → country_name

3NF: dependency on non-key attributes

series_code → series_name, target, goal

series_name → target, goal

target → goal

ER

RDF

Relational

OO

XML

RM vs XML

RM: simple records

advantage: ease of retrieval, consistency

XML: complex documents

advantage: exchange / serialization



JSON, YAML:

a similar data model to XML,
different serialization syntax

SQL Support in RDBMS

Direct XML export

```
mysql --xml -e "use starwars; select * from persona;"  
mysqldump --xml starwars
```

Direct XML import

LOAD XML command in MySQL 5.1 (~~to be installed soon~~).

Storing XML as a data type

Available in some systems, includes (some) XPATH support.
(Not in MySQL 5.1, but you can store XML in a BLOB.)

ER

RDF

Relational

OO

XML

The Web of Data

Modest:

Exchanging data via HTTP in XML

“APIs”, “Open data”, e.g.:

<http://www.toronto.ca/open/>

[http://webservices.nextbus.com/service/publicXMLFeed?
command=predictions&a=ttc&r=510&s=spadsuss_n](http://webservices.nextbus.com/service/publicXMLFeed?command=predictions&a=ttc&r=510&s=spadsuss_n)

Ambitious:

Same, but interlinked. “Semantic web”

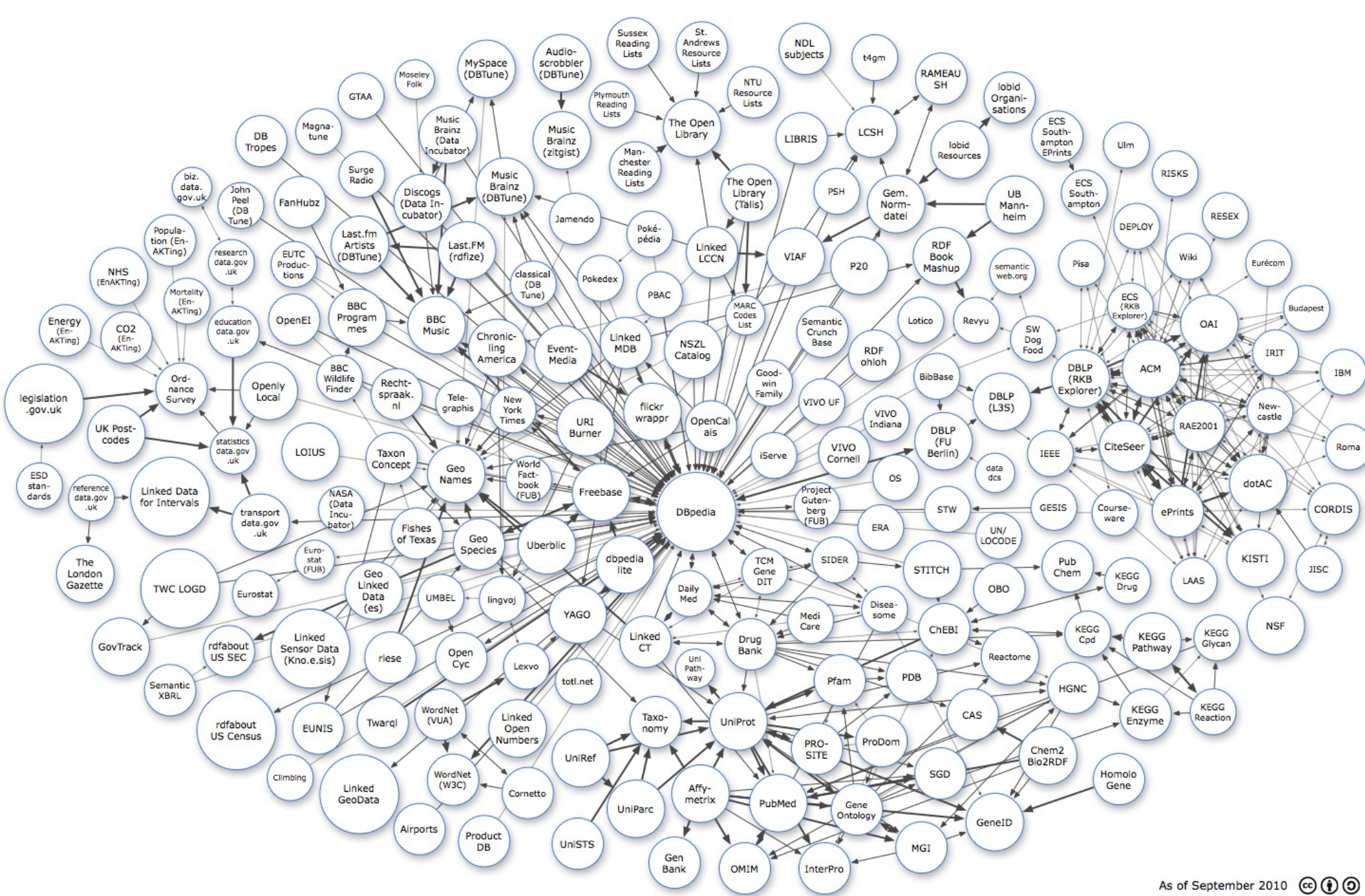
RDF: metadata in “triple” form:

@prefix foaf: <<http://xmlns.com/foaf/0.1/>>

ObiWan **a** foaf:Person

ObiWan **foaf:knows** Yoda

SPARQL – a bit like SQL adapted for RDF



As of September 2010

Linking Open Data cloud diagram, by Richard Cyganiak and Anja Jentzsch. <http://lod-cloud.net/>

ER

RDF

Relational

OO

XML

Objects

The “native” data format for many programming languages.

Classes vs instances (cf. ER)

Attributes vs Methods

`student.email_address`

`student.send_email(message)`

Encapsulation

DB the “Dumb” Way

```
cursor=db.cursor()
query = TEMPLATE % species
cursor.execute(query);
rows = cursor.fetchall()
print "<table>"
for row in rows:
    print "<tr>"
    for x in [0,1,3]:
        print "<td>", row[x], "</td>"
    print "</tr>"
print "</table>"
```

DB the "Dumb" Way

```
cursor=db.cursor()
query = TEMPLATE % species
cursor.execute(query);
rows = cursor.fetchall()
print "<table>"
for row in rows:
    print "<tr>"
    for x in [0,1,3]:
        print "<td>", row[x], "</td>"
    print "</tr>"
print "</table>"
```

DB the OO Way

```
results=db.find_species(species)
print "<table>"
for result in results:
    print "<tr>"
    print "<td>", result.name, "</td>"
    print "<td>", result.size, "</td>"
    print "<td>", result.gender, "</td>"
    print "</tr>"
print "</table>"
```


ER

RDF

Relational

OO

XML

Object-Relational Mapping

Option 1:

Map objects to relations
(code → database)

Option 2:

Map relations to objects
(database → code)

Option 2

Manual:

We setup a mapping

```
pet.name → getName($name)  
          setName($name)
```

Automatic:

The software figures it out
("introspection")

Examples

Rails / ActiveRecord (Ruby)

Django (Python)

Hibernate (Java)

Doctrine (PHP)

Django

Defining the model in Python:

```
/django/mysite/books/models.py
```

Django

Using the model:

```
cd /django/mysite  
python manage.py shell
```

In the Python console:

```
from books.models import Publisher  
p = Publisher(name="U of T Press",  
              city="Toronto", country="Canada")  
p.save()
```

Django

```
from books.models import Publisher
p = Publisher(name="Spadina Press",
              city="Toronto", country="Canada")
p.save()
```

```
results = Publisher.objects.filter(
    country="Canada")
for x in results :
    print "%s (%s)" % (x.name, x.city)
```

Questions?