Reference Scheme Modelling

Terry Halpin

e-mail: t.halpin@live.com
web: www.orm.net

Contents

- Introduction
- Simple Reference Schemes
- Compound Reference Schemes
- Disjunctive Reference Schemes
- Context-Dependent Reference Schemes
- Conclusion

Introduction

Natural ways of referring to objects (individual things)

- Ostension (pointing at the object of interest)

- Linguistic expressions
  - Proper names

    e.g. "Barack Obama"

  - Definite descriptions

    e.g. "The 44th president of the USA"

For computer systems, artificial object ids (visible or hidden) may be used, but for human communication, linguistic reference schemes should be used. However, there are major differences in the way that popular data modelling and ontological modelling languages support such reference schemes.

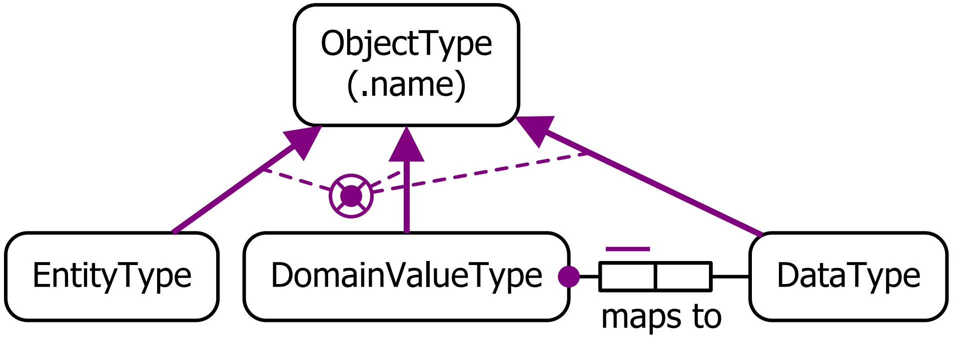
We now review how reference schemes are supported in:

ORM	(Object-Role Modelling)
UML	(Unified Modelling Language)
Barker ER	(Barker version of Entity Relationship modelling)
RDB	(Relational Database)
OWL	(Web Ontology Language)
LogiQL	(an extended version of Datalog)

- Understanding the differences in how these languages support reference schemes is important for:
  - Modelling identification schemes within these languages
  - Transforming models from one language to another

Simple Reference Schemes

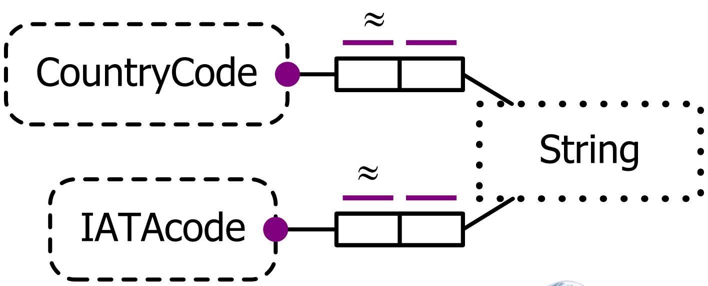
Object Types in ORM



Country
Person
Employee

CountryCode	String
FamilyName	String
EmployeeNr	Unsigned Integer

CountryCode 'CH'	'CH'



IATAcode 'CH'	'CH'

Bemidji Aviation

In ORM, an object is any individual thing of interest (other than null).

An object may be:

- an entity (e.g. a specific country)
- a domain value (e.g. a specific country code)
- a data value (e.g. the character string 'CH')

"≈" means "is represented by"

IATA = International
Air Transport Association

Value Reference

ORM allows any kind of object (including a domain value)
to play the role of the subject in a fact reading,
e.g.

The CountryCode ‘CH’ is based on the Language named ‘Latin’.

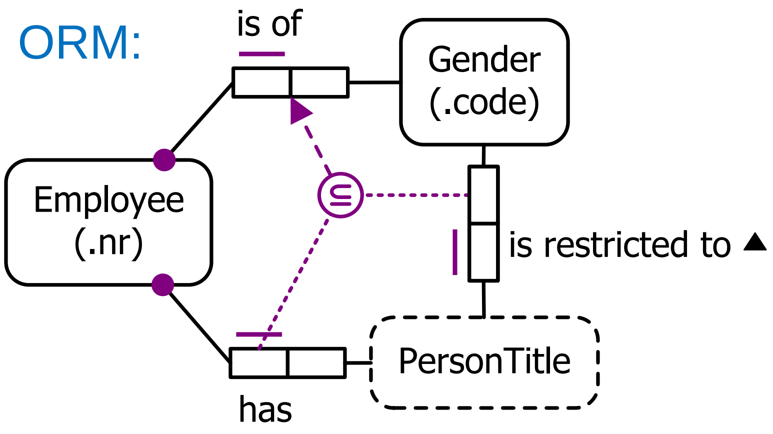
The PersonTitle ‘Mr’ is restricted to the Gender named ‘Male’.

The EnglishWord ‘gorse’ is a post-synonym of the EnglishWord ‘furze’.

This can also be modelled directly in RDBs and LogiQL.

UML, ER, and OWL do not allow this directly,
so require domain values that are subjects to be artificially remodelled as entities
(e.g. see next slide).

ORM:



is of

Gender
(.code)

Employee
(.nr)

⊆

is restricted to ▲

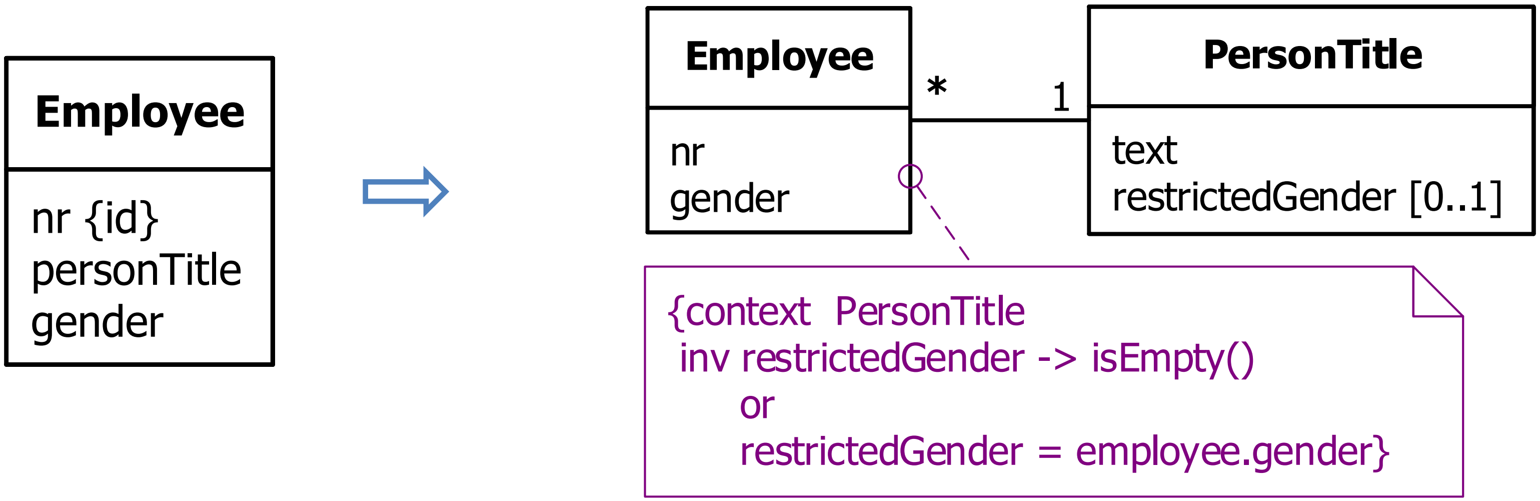
PersonTitle

has

Verbalization of join subset constraint:

If some Employee has **some** PersonTitle
that is restricted to **some** Gender
then that Employee is of **that** Gender.

UML:



Employee

nr {id}
personTitle
gender

⇨

Employee

nr
gender

*

1

PersonTitle

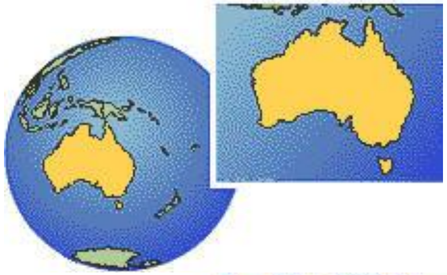
text
restrictedGender [0..1]

```
{context  PersonTitle
 inv restrictedGender -> isEmpty()
      or
      restrictedGender = employee.gender}
```

Referencing an entity by relating it to a single value

In this case, an entity is identified by one of the following

- an individual constant
- a single attribute
- a single relationship to a value

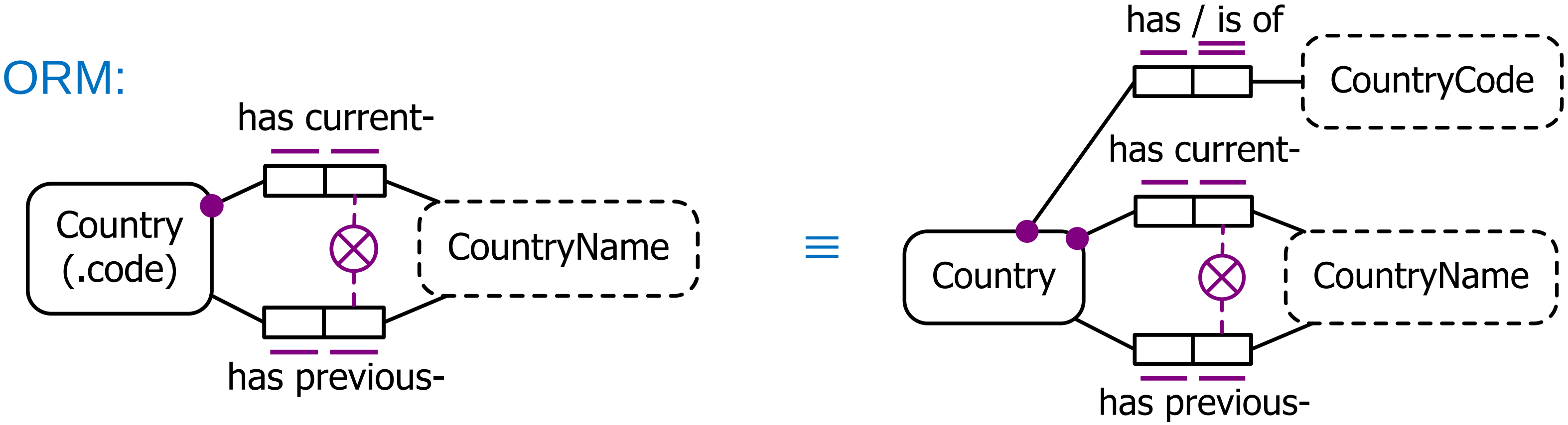


E.g. each country may be identified by its
ISO 3166 alpha-2 country code (e.g. 'AU' or 'CH')
or by its current name (e.g. 'Australia' or 'Switzerland').

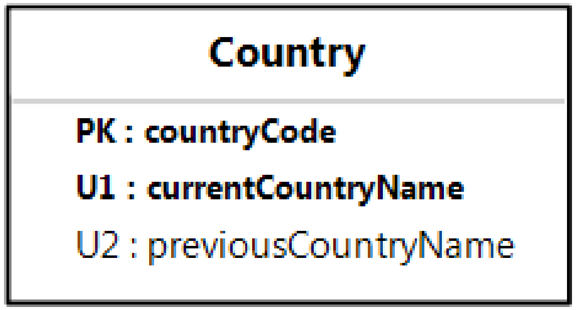
Those countries with a previous name can also be
referenced by that name (e.g. 'Ceylon' for Sri Lanka).

In the following models, country codes provide the preferred reference scheme.

ORM:



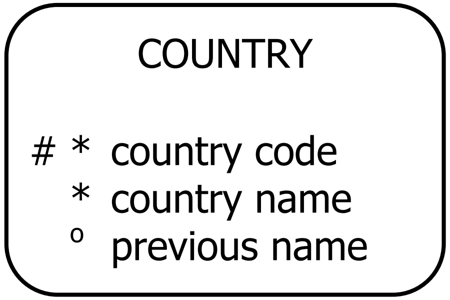
RDB:



Country

PK : countryCode
U1 : currentCountryName
U2 : previousCountryName

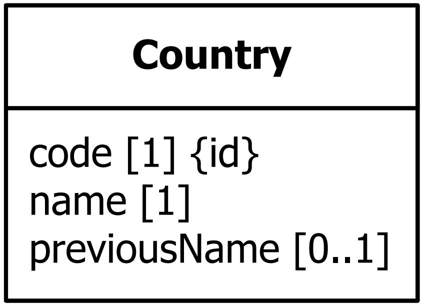
Barker ER:



COUNTRY

# * country code
* country name
o previous name

UML:



Country

code [1] {id}
name [1]
previousName [0..1]

Only ORM captures the exclusion constraint graphically.
Barker ER and UML also fail to graphically capture the uniqueness constraints on current and previous country names.

OWL has no standard graphic notation, but has
five textual languages that may be used to declare ontologies:

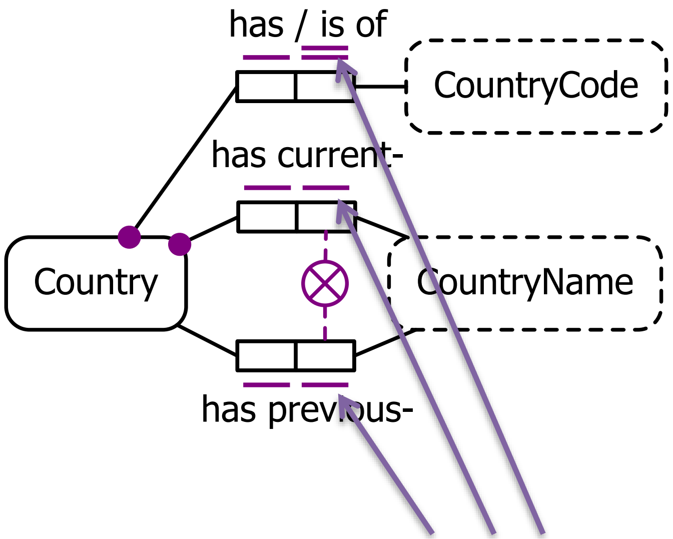
- RDF/XML
- OWL/XML
- Manchester Syntax
- Turtle
- Functional Syntax

Of these, Manchester syntax is by far the most readable, so we use that.

Named individuals are identified by IRIs (Internationalized Resource Indentifiers), e.g. www.eg.org#Czech_Republic.
These may be based on actual proper names (excluding spaces),
or be surrogate IRIs.
Human-readable labels may be added using rdfs:label annotation properties.

Assuming IRIs are provided, the Country model (ignoring the exclusion constraint) may be coded in Manchester syntax as shown on the next slide.

DataProperty: hasCountryCode
  Domain: Country
  Range: xsd:string
  Characteristics: Functional
DataProperty: hasCurrentCountryName
  Domain: Country
  Range: xsd:string
  Characteristics: Functional
DataProperty: hasPreviousCountryName
  Domain: Country
  Range: xsd:string
  Characteristics: Functional
Class: Country
  SubClassOf: hasCountryCode min 1
  HasKey: hasCountryCode
  SubClassOf: hasCurrentCountryName min 1
  HasKey: hasCurrentCountryName
  HasKey: hasPreviousCountryName



The HasKey declarations capture just the uniqueness constraints on the right-hand roles. HasKey declarations are needed to do this, because OWL forbids data properties (that relate entities to literals) to be declared inverse-functional.

Referencing an entity by relating it to a single entity

OWL allows object properties (that relate entities to entities) to be declared inverse-functional.

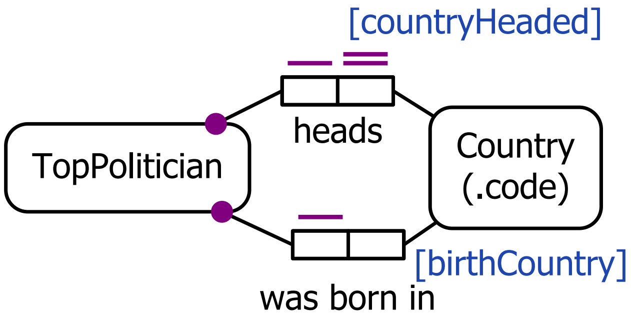
In OWL, entities may be:

- named individuals (identified by an IRI)
- unnamed individuals (represented by blank nodes).

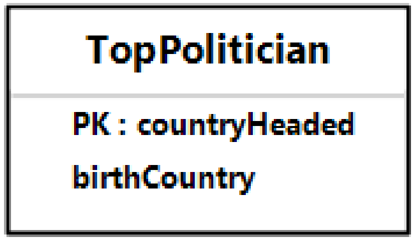
Hence OWL supports reference schemes that identify entities by relating them to other entities.

E.g. see the TopPolitican model on the next slide.
Here, the term “top politician” means the politician who is considered to be the head politician (e.g. a president, a prime minister) of a country. If a country has both a president and a prime minister, only one of these is considered the head politician.

ORM:



RDB:

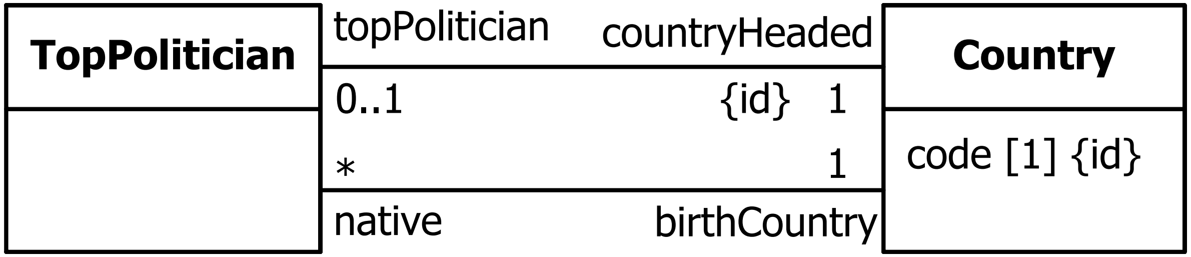


(sample data from 2013)

TopPolitician:

countryHeaded	birthCountry
AU	GB
GB	GB
US	US

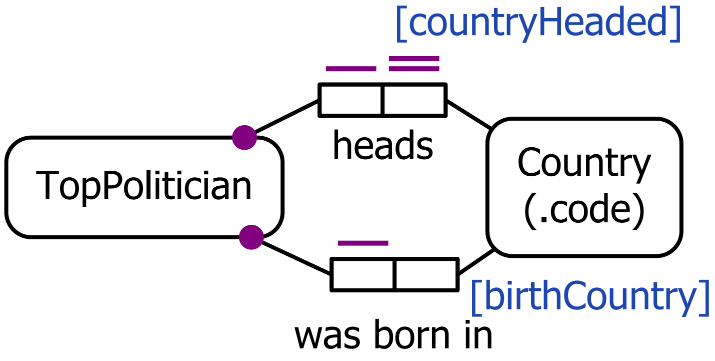
UML:



Barker ER does not support this kind of reference scheme
(although it allows relationships as components of a primary identifier,
it does not allow a single relationship to provide the whole identifier).

OWL code (in Manchester syntax) for this example is shown on the next slide.

```
DataProperty: hasCountryCode
… (see earlier code sample for details)
Class: Country
 SubClassOf:  hasCountryCode min 1
 HasKey:  hasCountryCode
ObjectProperty: headsCountry
 Domain: TopPolitician
 Range: Country
 Characteristics: Functional, InverseFunctional
ObjectProperty: wasBornInCountry
 Domain: TopPolitician
 Range: Country
 Characteristics: Functional
Class: TopPolitician
 SubClassOf:  headsCountry min 1
 SubClassOf:  wasBornInCountry min 1
```



TopPolitician:

countryHeaded	birthCountry
AU	GB
GB	GB
US	US

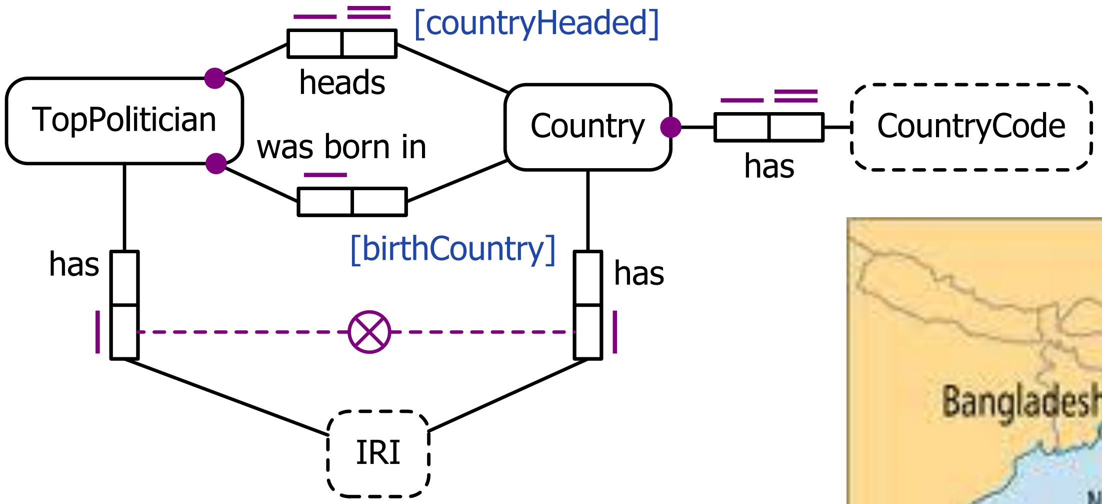
(sample data from 2013)

The first row of the RDB table records the fact that the top politician who heads Australia (country code = 'AU') was born in the United Kingdom (country code = 'GB').
We can record this without knowing the name of the politician (Julia Gillard).

In OWL, this fact may be coded using blank node ids for unnamed individuals.

```
Individual:  _:p1
  Facts:  headsCountry  _:c1,  wasBornInCountry  _:c2
Individual:  _:c1
  Facts:  hasCountryCode  "AU"
Individual:  _:c2
  Facts: hasCountryCode  "GB"
```

OWL individuals may be named (with one or more IRIs) or be unnamed.



Class: Country
  HasKey: hasCountryCode

Individual: Myanmar
   Facts: hasCountryCode "MM"
Individual: Burma
   Facts: hasCountryCode "MM"

⇨

Individual: Burma
  SameAs: Myanmar

Class: Country
  HasKey: hasCountryCode

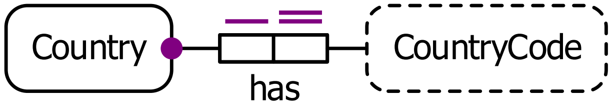
Individual: JuliaGillard
  Facts: wasBornInCountry _:c1
Individual: _:c1
  Facts: hasCountryCode "GB"

Individual: TheUK
  Facts: hasCountryCode "GB"

Will an OWL reasoner now infer the following?

Individual: JuliaGillard
  Facts: wasBornInCountry TheUK

This HasKey UC applies only to named individuals



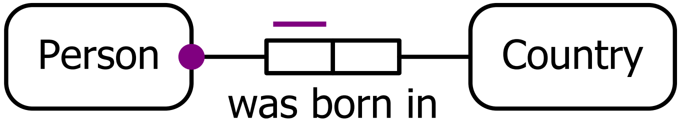
No! HasKey declarations apply only to named individuals
(unlike InverseFunctional declarations).
OWL allows that there could be many unnamed individuals
that have the country code “GB”, not just the named individual TheUK.

Typical databases adopt closed world semantics, and treat declarations such as "Each person was born in some country" as constraints, so an update attempt to record a person without his/her birth country will be rejected.

In contrast, OWL adopts open world semantics, and treats many declarations simply as propositions, not as constraints.

e.g.

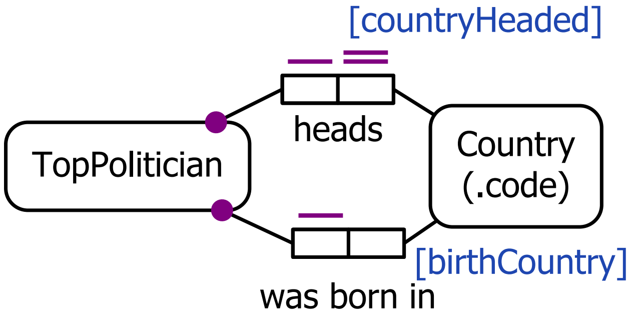
```
    ObjectProperty: wasBornInCountry
     Domain: Person
     Range: Country
     Characteristics: Functional
    Class: Person
     SubClassOf:  wasBornInCountry min 1
```



The OWL code declares that each recorded person was born in exactly one country, but it does not require that the system knows which country that is.

So care is required when mapping between data modelling approaches and OWL. Some recent proposals have made to extend OWL to cater properly for constraints (see references [7] and [22] in the cited Halpin (2019) paper).

LogiQL[1] is an extended version of Datalog developed by LogicBlox that provides good performance for large databases, and deep support for logical constraints and derivation rules (especially recursive rules).
It adopts the closed world assumption.



The ORM schema may be coded in LogiQL as follows.

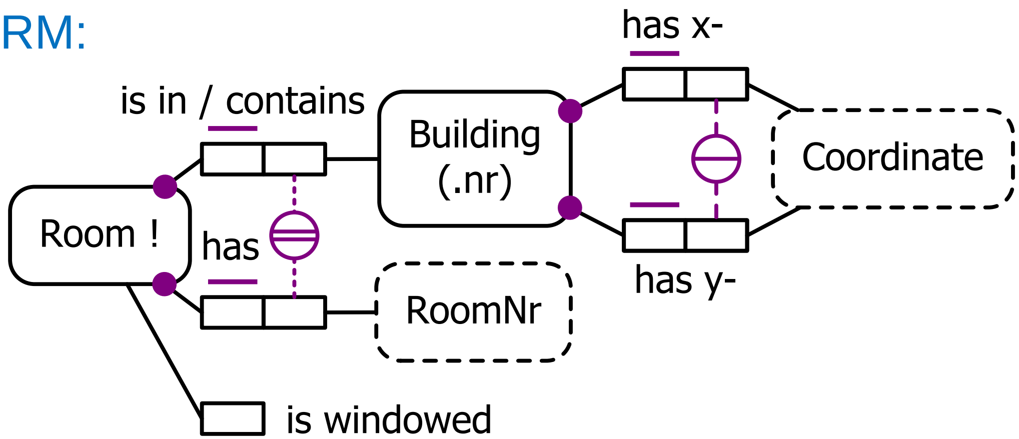
```
Country(c), hasCountryCode(c:cc) -> string(cc).
TopPolitician(p) -> .
countryHeadedBy[p] = c -> TopPolitician(p), Country(c).
birthCountryOf[p] = c -> TopPolitician(p), Country(c).
countryHeadedBy[p1] = c, countryHeadedBy[p2] = c -> p1 = p2.
TopPolitician(p) -> countryHeadedBy[p] = _, birthCountryOf[p] = _.
```

[1] https://developer.logicblox.com/technology/

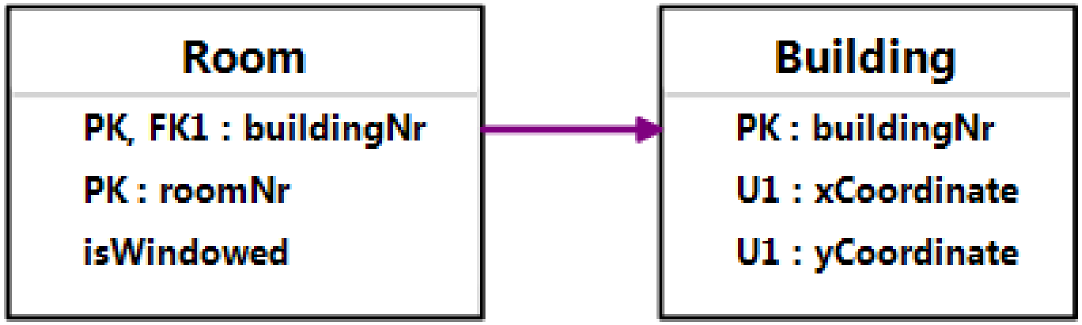
Compound Reference Schemes

A composite reference scheme for an entity identifies it using a combination of two or more attributes or relationships, e.g.

ORM:



RDB:



Room

PK, FK1 : buildingNr

PK : roomNr

isWindowed

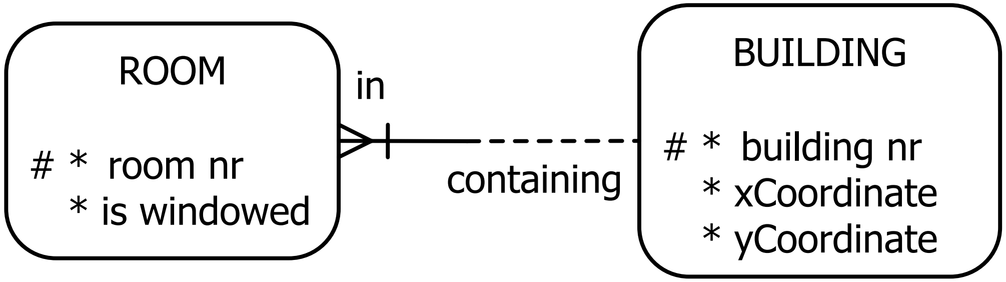
Building

PK : buildingNr

U1 : xCoordinate

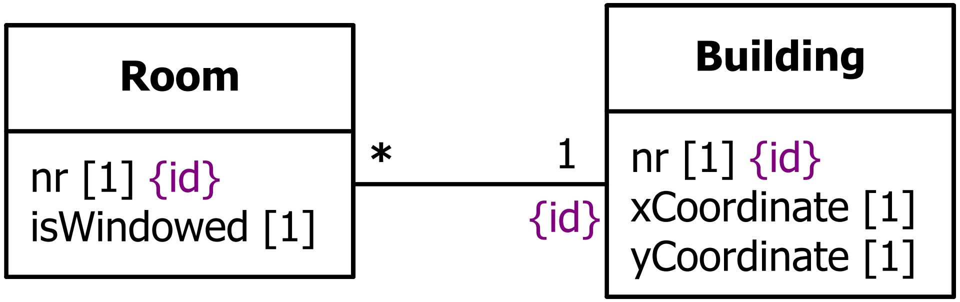
U1 : yCoordinate

Barker ER:

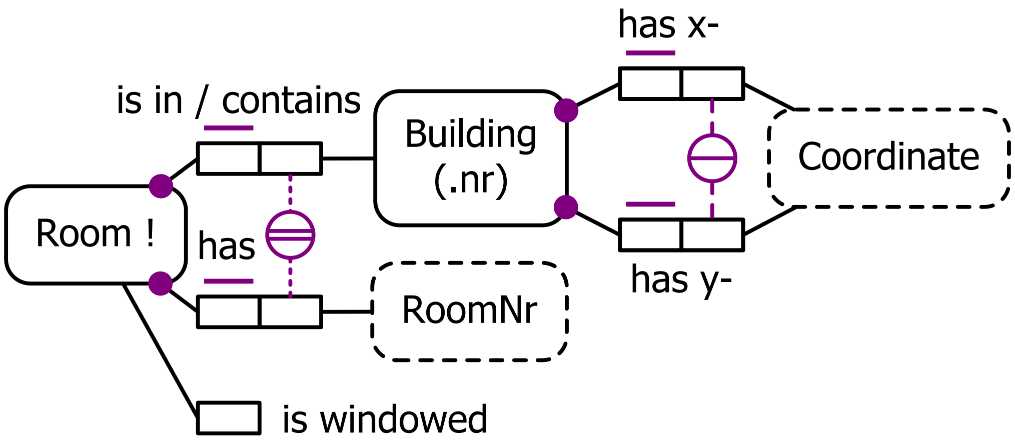


The composite uniqueness constraint on *x* and *y* coordinate pairs is lost.

UML:



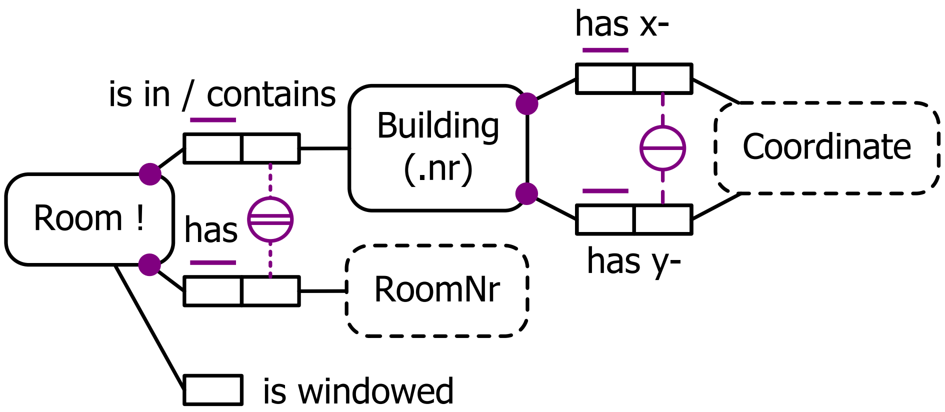
Again, the composite uniqueness constraint on *x* and *y* coordinate pairs is lost.



In OWL, the unary isWindowed predicate is replaced by a binary data property that maps Room to a Boolean data type.
The rest of the schema may be coded in a similar way to that discussed earlier.
The reference predicates are coded as HasKey properties (see below), but these are effective only if meaningful IRIs (hence named individuals) are supplied, e.g. "Room3-205" for Room 205 in Building3.

```
Class:  Building
  HasKey:  hasBuildingNr
  HasKey:  hasXcoordinate, hasYcoordinate
Class:  Room
  HasKey: isInBuilding, hasRoomNr
```

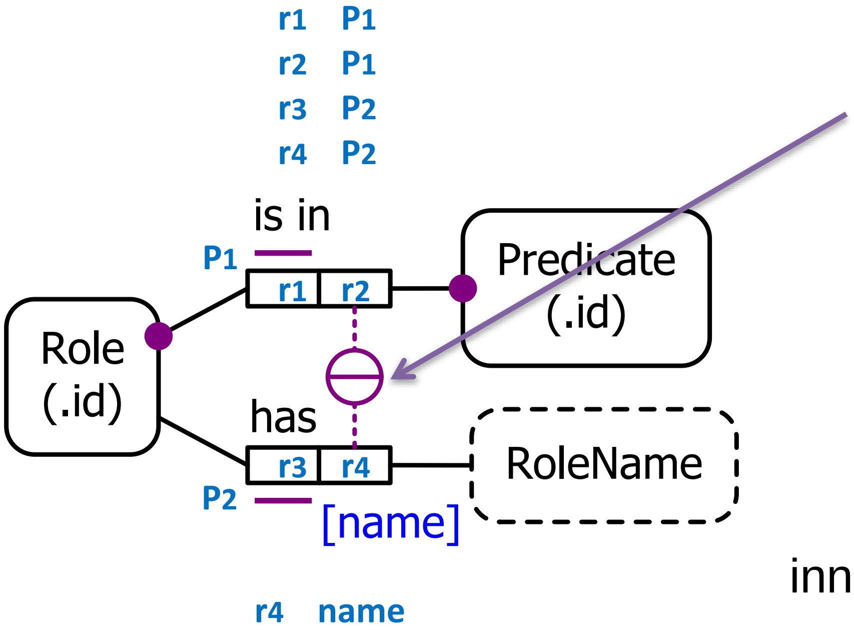
The ORM schema may be coded in LogiQL thus:



```
Room(r)  ->  .
Building(b), hasBuildingNr(b:bn)  ->  int(bn).
buildingContaining[r] = b  ->  Room(r), Building(b).
roomNrOf[r] = rn  ->  Room(r), string(rn).
// external uniqueness constraint for Room
buildingContaining[r1] = b, roomNrOf[r1] = rn,
  buildingContaining[r2] = b, roomNrOf[r2] = rn  ->  r1 = r2.
Room(r)  ->  buildingContaining[r] = _, roomNrOf[r] = _.
isWindowed(r)  ->  Room(r).
xCoordinateOf[b] = x  ->  Building(b), int(x).
yCoordinateOf[b] = y  ->  Building(b), int(y).
// external uniqueness constraint for building
xCoordinateOf[b1] = x,  yCoordinateOf[b1] = y,
  xCoordinateOf[b2] = x,  yCoordinateOf[b2] = y  ->  b1 = b2.
Building(b)  ->  xCoordinateOf[b] = _,  yCoordinateOf[b] = _.
```

Disjunctive Reference Schemes

Join Semantics for External Uniqueness Constraints

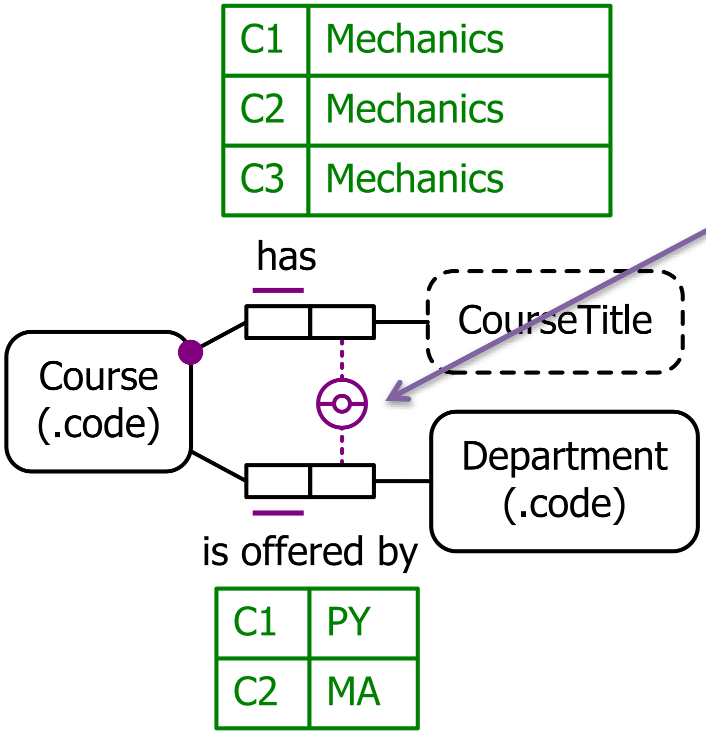


This external uniqueness constraint has inner join semantics

P_1 left outer join P_2

	Role (roleId,	predicateId,	[roleName])
	r1	P1	?
	r2	P1	?
	r3	P2	?
inner join:	r4	P2	name

All roles (named or unnamed) may be referenced by their roleId.
Role names are optional in ORM,
but within the same predicate, role names must be distinct.
Hence, named roles may also be referenced by the combination of their name and predicate.



C1	Mechanics
C2	Mechanics
C3	Mechanics

C1	PY
C2	MA

This external uniqueness constraint
has outer join semantics
(with the added proviso that
nulls are treated as actual values)

Course (courseCode, courseTitle, [departmentCode])

C1	Mechanics	PY	
C2	Mechanics	MA	
C3	Mechanics	?	} violates constraint
C4	Mechanics	?	

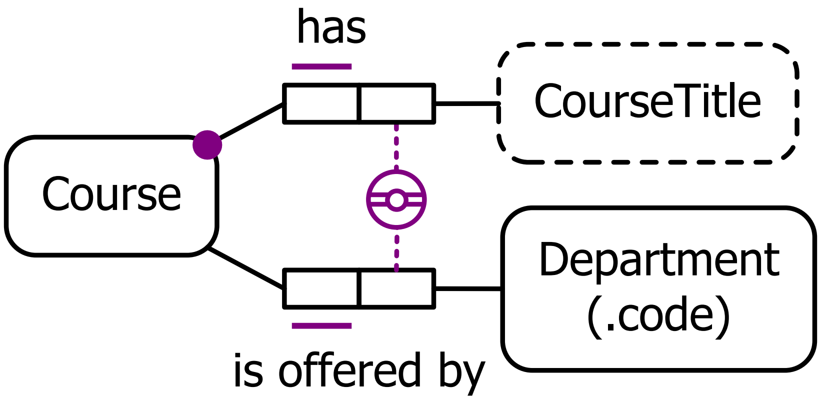
All courses may be referenced by their courseCode.
Some courses might not offered by a department (e.g. a course by a visitor),
but courses offered by the same department must have distinct titles.
Each course may also be referenced by exactly one of the following patterns:

courseTitle and its department

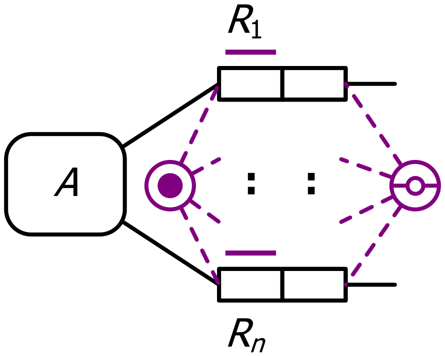
courseTitle where the course has no department

Reference schemes involving a disjunction of two or more patterns are known as **disjunctive reference schemes**.

External uniqueness constraints with outer join semantics may be used for the preferred reference scheme.
In this case, a double-bar is used.



If at least one referencing relationship is optional for its entity type,
an external uniqueness constraint with inner join semantics
cannot be used for the preferred reference scheme
since it can be used to reference only some instances of the entity type.



The general, weakest pattern allowed
for disjunctive reference (*n* > 1).
If used for preferred reference, use a double-bar.

$\forall y_1..y_n\ \exists^{0..1}x\ (xR_1y_1\ \&\ \ldots\ \&\ xR_ny_n)$ ← Inner join part.

$\&$

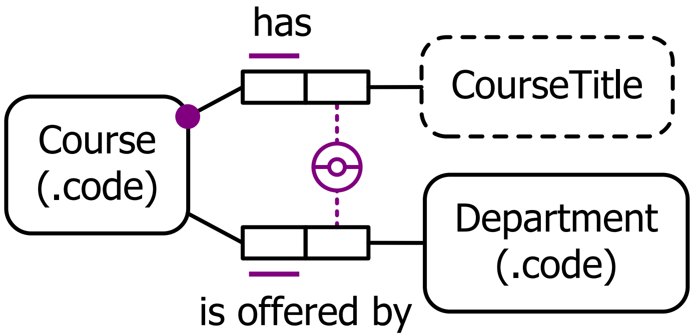
$\forall y\ \exists^{0..1}x\ [xR_1y\ \&\ {\sim}\exists z(xR_2z\ \ldots \vee xR_nz)]$

$\&\ \ldots$

$\&\ \forall y_1..y_{n\text{-}1}\ \exists^{0..1}x\ (xR_1y_1\ \&\ \ldots\ \&\ xR_{n\text{-}1}y_{n\text{-}1}\ \&\ {\sim}\exists z\ xR_nz\ )$

The outer join part covers all patterns where 1 or more components is absent.

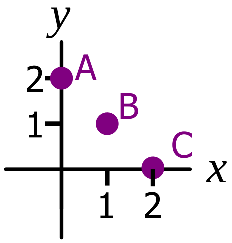
E.g.



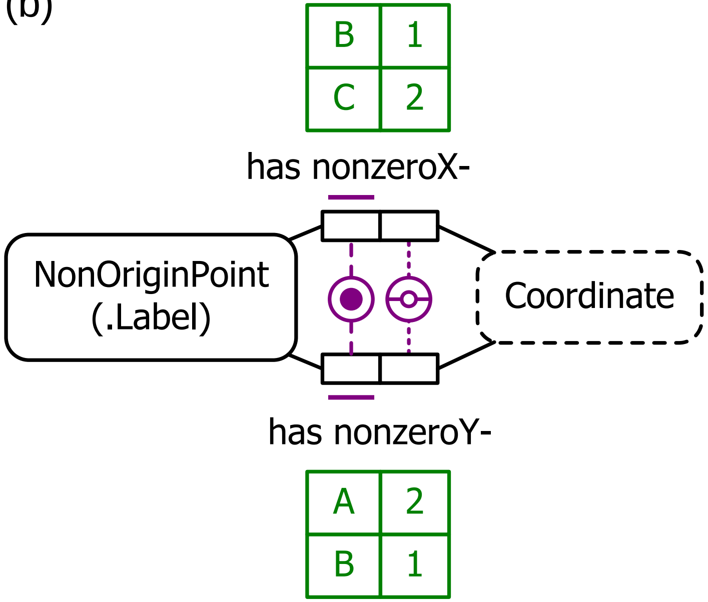
$\forall ct$:CourseTitle,d:Department $\exists^{0..1}c$:Course (c hasCourseTitle ct & c isOfferedBy d)
& $\forall ct$:CourseTitle $\exists^{0..1}c$:Course [c hasCourseTitle ct & ~$\exists d$:Department c isOfferedBy d]

An example of the weakest disjunctive reference pattern used for secondary reference of non-origin points on a Cartesian plane.

(a)



(b)



B	1
C	2

A	2
B	1

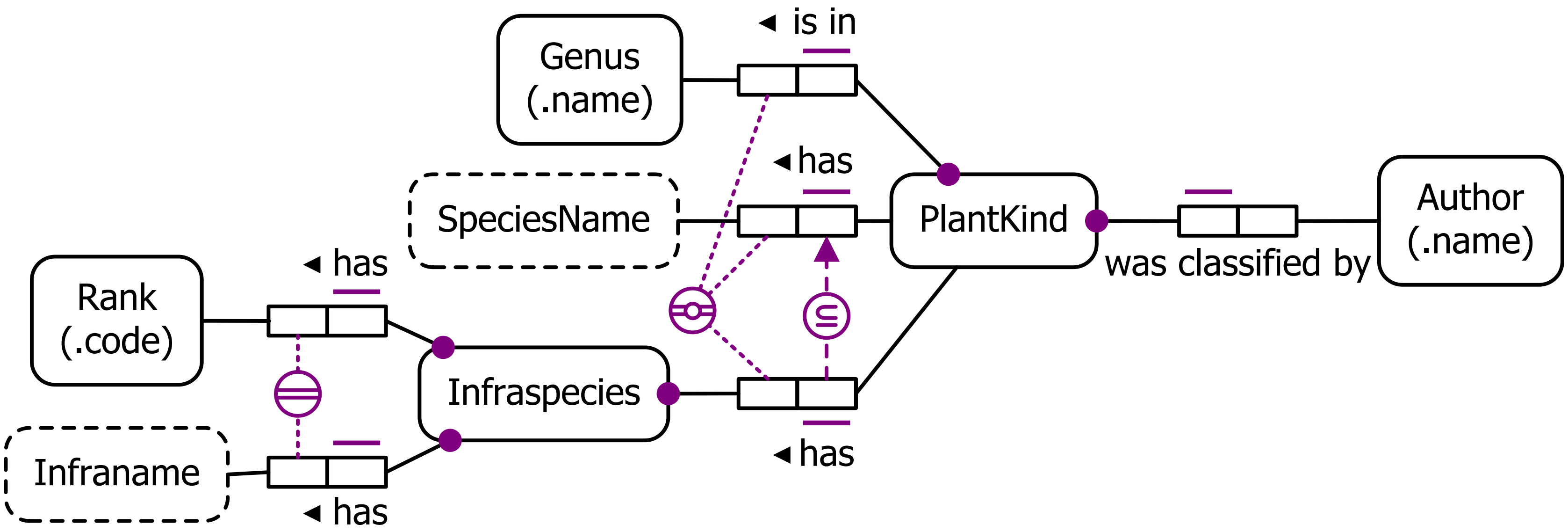
(c)

NonOriginPoint (label, [xCoord], [yCoord])[1]

A	null	2
B	1	1
C	2	null

[1] xCoord **is not null** **or** yCoord **is not null**

Simplified version of an industrial model for botanical naming.



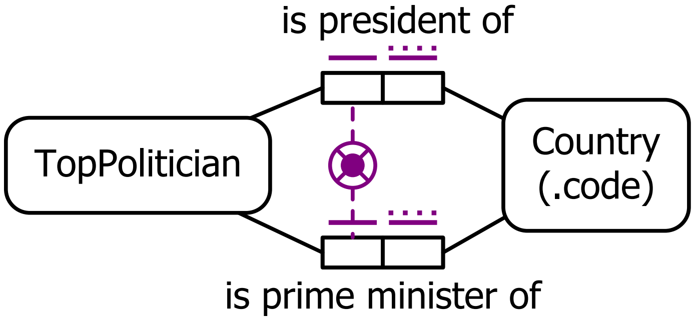
Some plant kinds are identified purely by their genus, e.g. *Agrostis*.

Some are identified by combining genus and species name, e.g. *Acacia interior*.

Others are identified by combining genus, species name and infraspecies (itself identified by combining rank and infraname), e.g. *Eucalyptus fibrosa ssp. nubila*.

A uniqueness constraint with a double-bar, one bar of which is solid and one dotted, may be used to reference just some instances of the relevant entity type.

A disjunctive reference scheme for the entity type may then be provided by two or more such partial, preferred reference relationships, e.g.

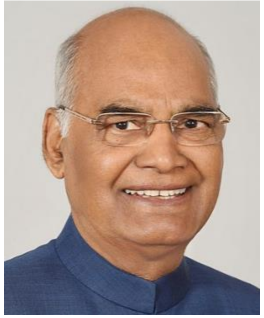


Unlike our earlier example, this allows a country to have two top politicians, e.g.

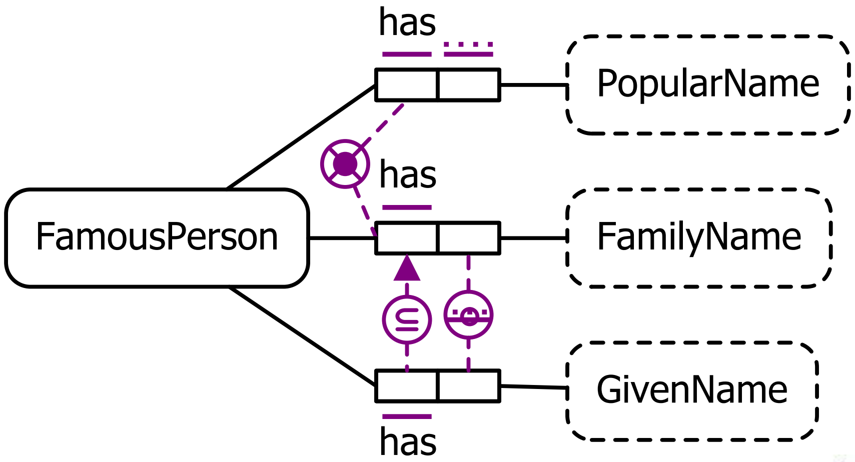
The TopPolitican who is prime minister of India

The TopPolitican who is president of India

Narendra Modi

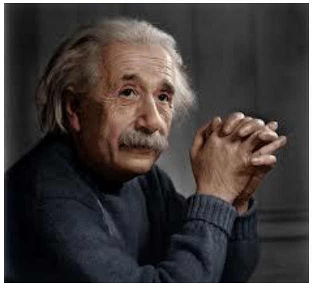


Ram Nath Kovind



In this example,
some famous persons may be identified by
just a popular name, e.g. ‘Confucius’
(instead of ‘Kong Qiu’ or K’ung fu tzu).

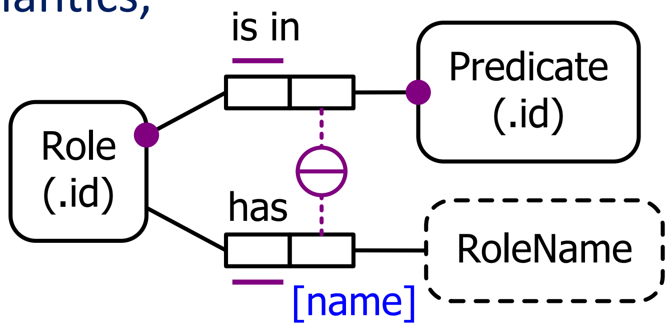
Some may be identified by just their family name, e.g. ‘Einstein’



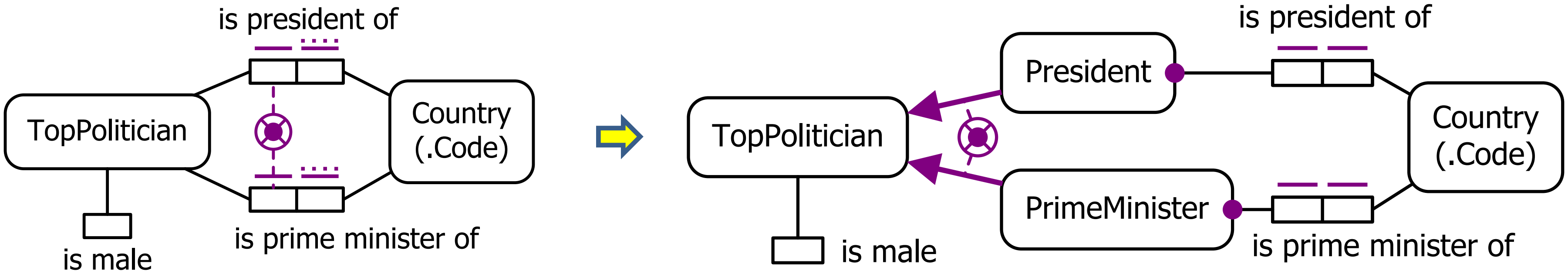
Others may be identified by
combining their family name with a given name,
e.g. ‘Marie Curie’,
‘Pierre Curie’

Disjunctive reference schemes can be mapped from ORM to RDB schemas, but are not supported in the graphical notation of Barker ER or UML.

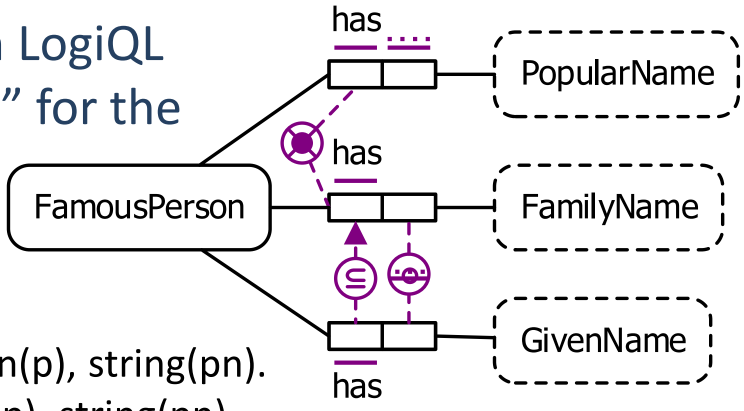
HasKey properties in OWL have inner join semantics, so cases like this can be coded in OWL, along with the usual limitations discussed for HasKey properties discussed earlier.



Disjunctive reference with outer join semantics can be implemented in OWL but some remodelling is typically required, e.g. to create a partition of relevant subclasses, e.g.



This ORM schema may be specified in LogiQL as follows. LogiQL uses a semicolon “;” for the inclusive-or operator.

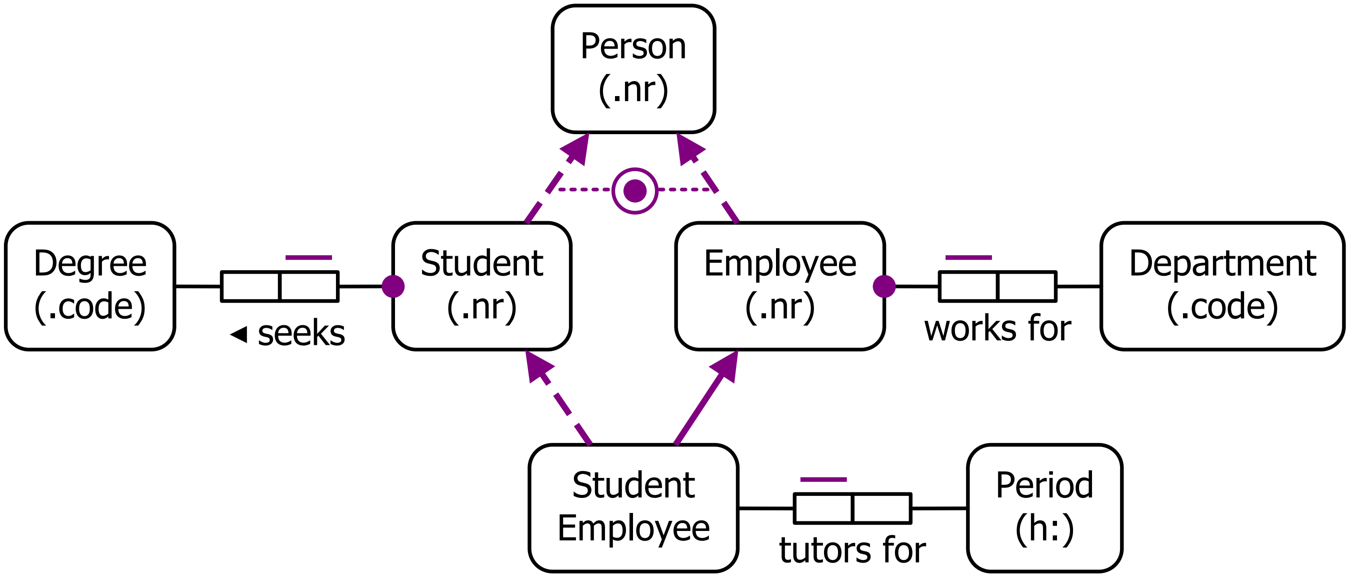


```
FamousPerson(p)  -> .
isMale(p)  ->  FamousPerson(p).
popularNameOf[p] = pn  ->  FamousPerson(p), string(pn).
familyNameOf[p] = pn  ->  FamousPerson(p), string(pn).
givenNameOf[p] = pn  ->  FamousPerson(p), string(pn).
popularNameOf[p1] = pn, popularNameOf[p2] = pn  ->  p1 = p2.
// inner join aspect of external uniqueness constraint
familyNameOf[p1] = pn, givenNameOf[p1] = gn,
 familyNameOf[p2] = pn, givenNameOf[p2] = gn  ->  p1 = p2.
// outer join aspect of external uniqueness constraint
familyNameOf[p1] = pn, !givenNameOf[p1] = _,
 familyNameOf[p2] = pn, !givenNameOf[p2] = _  ->  p1 = p2.
givenNameOf[p1] = pn, !familyNameOf[p1] = _,
 givenNameOf[p2] = pn, !familyNameOf[p2] = _  ->  p1 = p2.
// inclusive or constraint
FamousPerson(p)  ->  popularNameOf[p] = _; familyNameOf[p] = _.
//exclusion constraint
popularNameOf[p] = _  ->  ! familyNameOf[p] = _.
// subset constraint
givenNameOf[p] = _  ->  familyNameOf[p] = _.
```

Context-Dependent Reference Schemes

In a context-dependent reference scheme,
the preferred identifier for an entity varies according to its context.

ORM supports this by allowing subtypes to introduce new preferred reference schemes used within the scope of their immediate fact types (displayed by a dashed subtyping link), e.g.



Mapping of context-dependent reference schemes from ORM to RDBs is discussed in Halpin & Morgan (2008), pp, 519-521.

Barker ER and UML have no direct support for this notion.
However, UML's implicit use of oids for class instances provides support for global identifiers.

OWL allows multiple IRIs for the same entity, and use of the owl:sameAs predicate to equate individuals.
This can be used to provide basic support for context-dependent reference.

LogiQL can model most aspects of context-dependent reference, but does not yet fully support multiple inheritance.

Conclusion

Reference Scheme Support	ORM	RDB	Barker ER	UML	OWL	LogiQL
simple, primary	Yes	Yes	Mostly	Yes	Mostly	Yes
simple, secondary	Yes	Yes	No	No	Mostly	Yes
compound, primary	Yes	Yes	Yes	Yes	Mostly	Yes
compound, secondary	Yes	Yes	No	No	Mostly	Yes
disjunctive	Yes	Yes	No	No	Mostly	Yes
context-dependent	Yes	Yes	No	Partly	Partly	Partly

Future research plans include extending the NORMA tool with full support for new disjunctive reference cases (including automated verbalization) and automated mapping between ORM, RDB, ER, UML, OWL and LogiQL.

Selected References and Websites:

Halpin, T. 2019, ‘Reference Scheme Modeling’, *New Perspectives on Information Systems Modeling and Design*, IGI Global, Hershey, pp. 227-254.

Halpin, T. & Morgan T. 2008, *Information Modeling and Relational Databases, 2nd edition*, Morgan Kaufmann.

Halpin, T. 2015, *Object-Role Modeling Fundamentals*, Technics Publications, NJ.

Halpin, T. 2016, *Object-Role Modeling Workbook*. Technics Publications. NJ.

www.BRcommunity.com	-- *Business Rules Journal* (my series on *ontological modelling & logical modelling*)
www.orm.net	-- my website
www.ORMFoundation.org	-- ORM Foundation,
www.ORMsolutions.com	-- Browser-based model viewer, …
www.factbasedmodeling.org	-- Fact based modelling website
www.omg.org/spec/UML/	-- UML specification (current version 2.5.1)
www.w3.org/TR/owl2-direct-semantics	-- OWL 2 direct semantics