ModelGen:
Model Independent Schema Translation

from work by
Paolo Atzeni(1), Paolo Cappellari(1), Phil Bernstein(2)

(1) Università Roma Tre
(2) Microsoft Research

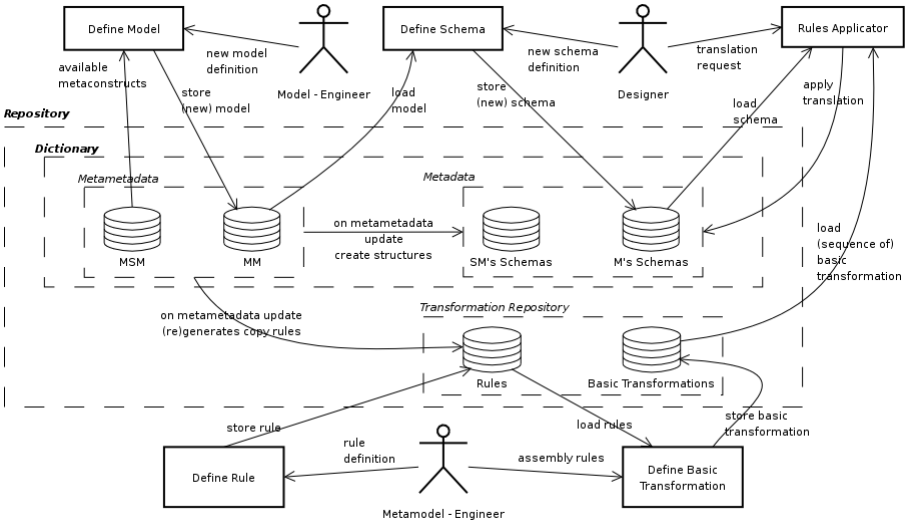
Background

- The MDM project [Atzeni & Torlone 96]
- The model management proposal [Bernstein 03]:
  - a set of model-independent operators to be used to handle metadata problems
  - one of the operators is ModelGen:
    - given a source data model M1, a target data model M2, and a source schema S1 expressed in M1, ModelGen generates a target schema S2 in M2
- We will come back to "Model management" in a few weeks

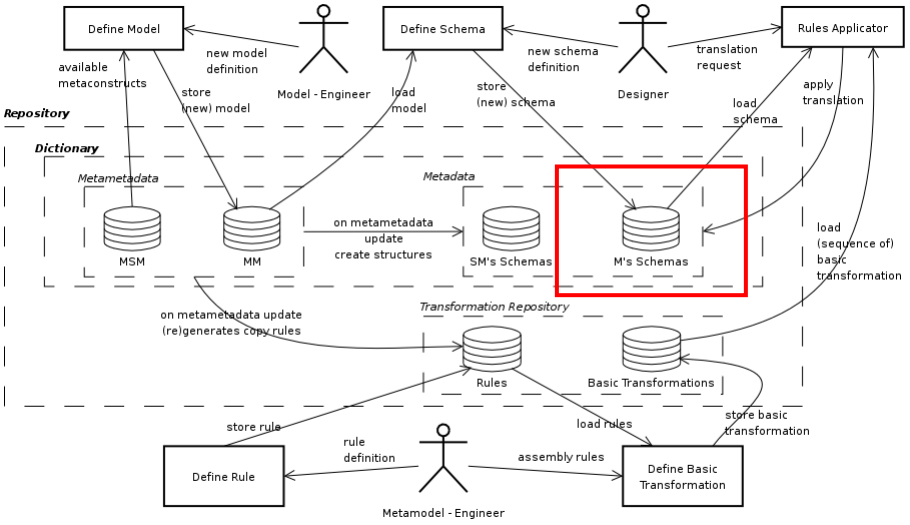
The main features

- a “white box” approach
  - it exposes both the dictionary and the translations,
  - thus allowing for rapid development (and maintenance) of models and translations,
  - and for reasoning on the correctness of translations

ModelGen: the architecture



The dictionary, bottom-up



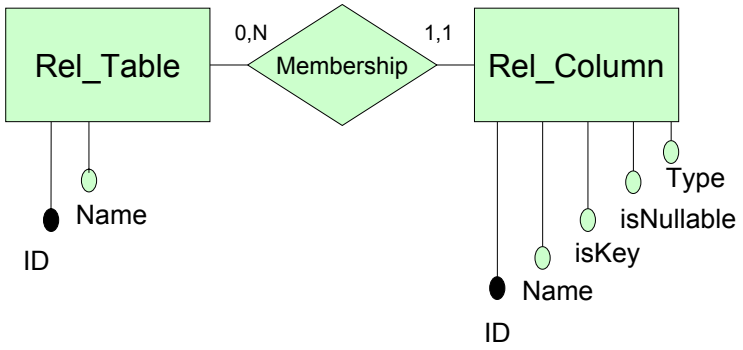
A relational database

- Contains a set tables each with a set of columns

Employees		
EmpNo	**Name**	**Dept**
22134	Smith	A
22201	Jones	B
22255	Black	A

Departments	
Name	**Address**
A	5, Pine St
B	10, Walnut St

The (conceptual) description of relational databases



The (logical) description of relational databases

Rel_Table	
OID	**Name**

Rel_Column					
OID	**Name**	**isKey**	**isNullable**	**Type**	**TableOID**

A relational database, with its dictionary

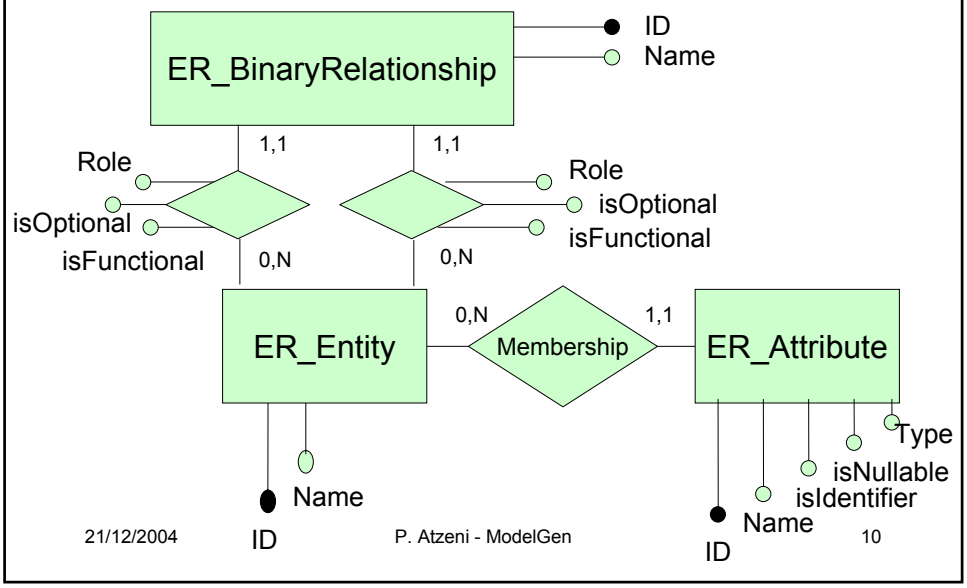
Employees		
EmpNo	**Name**	**Dept**
22134	Smith	A
22201	Jones	B
22255	Black	A

Departments	
Name	**Address**
A	5, Pine St
B	10, Walnut St

Rel_Table	
OID	**Name**
001	Employees
002	Departments

Rel_Column					
OID	**Name**	**isKey**	**isNullable**	**Type**	**TableOID**
101	EmpNo	T	F	Int	001
102	Name	F	F	Text	001
103	Dept	F	T	Char	001
104	Name	T	F	Char	002
105	Address	F	F	Text	002

The (conceptual) description of binary ER schemas



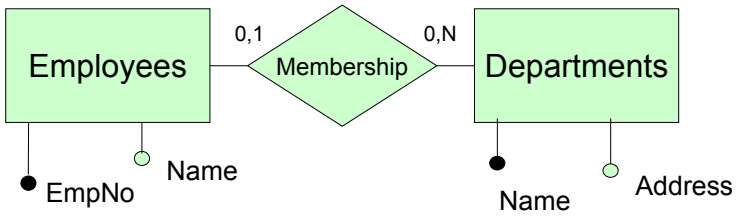
The (logical) description of binary ER schemas

ER_Entity	
OID	**Name**

ER_Attribute					
OID	**Name**	**isIdent**	**isNullable**	**Type**	**TableOID**

ER_BinaryRelationship									
OID	**Name**	**eOID1**	**isF1**	**isO1**	**Role1**	**eOID2**	**isF2**	**isO2**	**Role2**

An ER schema and its translation



Employees		
EmpNo	**Name**	**Dept**

Departments	
Name	**Address**

The dictionary for an E-R schema

ER_Entity	
OID	**Name**
301	Employees
302	Departments

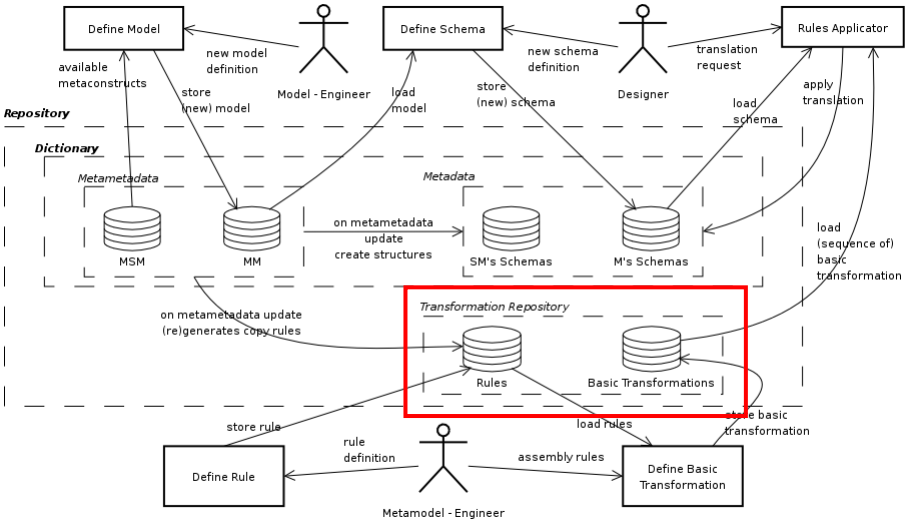
ER_Attribute					
OID	**Name**	**isIdent**	**isNullable**	**Type**	**EntityOID**
401	EmpNo	T	F	Int	301
402	Name	F	F	Text	301
404	Name	T	F	Char	302
405	Address	F	F	Text	302

ER_BinaryRelationship									
OID	**Name**	**eOID1**	**isF1**	**isO1**	**Role1**	**eOID2**	**isF2**	**isO2**	**Role2**
501	Membership	301	T	T		302	F	T	

How do we specify translations?

- In steps:
  - sequences of basic translations;
  - for example, to translate from the N-ary ER model to the relational one, we could have two steps
    - from the N-ary ER model to the binary ER model
    - from the binary ER model to the relational model
- Each basic translation is specified by means of a set of rules written in "Datalog with OID invention"

The dictionary, bottom-up



A basic translation

- From the binary ER model (with no attributes for relationships and no external identification) to the relational model
  - a table for each entity
  - a column (in the table that represents the entity) for each attribute
  - for each M:N relationship
    - a table for the relationship
    - a column (in the table that represents the relationship) for each attribute that participates to the identifier of each of the involved entities
  - for each 1:N relationship (and similarly for each 1:1 relationship)
    - a column (in the table that represents the "1" entity) for each attribute of the identifier of the other entity

"A table for each entity"

Rel_Table(OID: #tableOID_1(eOID), Name: name)
← ER_Entity (OID: eOID, Name: name) ;

Datalog with OID invention

- Datalog (informally):
  - a logic programming language with no function symbols and predicates that correspond to relations in a database
- Datalog with OID invention:
  - an extension of Datalog that uses functions to generate new identifiers when needed
- Skolem functions:
  - injective functions that generate "new" values (value that do not appear anywhere else; so different Skolem functions have disjoint ranges); indeed, we use a pragmatic variation of them
- In Datalog people usually adopt a positional notation, we prefer a non-positional one

The first rule, comments

Rel_Table(OID: #tableOID_1(eOID), Name: n)
<- ER_Entity (OID: eOID, Name: n) ;

- it generates a tuple in **Rel_Table** for each tuple in **ER_Entity**
  - the value for the attribute **Name** is copied (by using variable **n**)
  - the value for **OID** is "invented": the system generates a new value for the function **#tableOID_1()** for each different value of the argument, so a different value for each value of **ER_Entity.OID**

"A column for each attribute"

```
Rel_Column(
        OID: #columnOID_1(aOID),
        Name: name,
        TableOID: #tableOID_1(eOID),
        IsNullable: isNullable,
        IsKey: isID,
        Type : type )
    <-    ER_Attribute(
                OID: aOID,
                Name: name,
                EntityOID: eOID,
                isIdent: isID,
                isNullable: isNullable ,
                Type : type ) ;
```

Skolem functions, further comments

Rel_Column(OID: #columnOID_1(aOID), …
TableOID: #tableOID_1(eOID), …)
<- ER_Attribute(OID: aOID, … EntityOID: eOID, …) ;

- intuitively, the first function "generates" a new value, whereas the second "reuses" the value generated by the first rule

Rel_Table(OID: #tableOID_1(eOID), …)
<- ER_Entity (OID: eOID, …) ;

- functions
  - are functions
  - are injective
  - have disjoint ranges

"A table for each M:N relationship"

Rel_Table(OID: #tableOID_2(rOID), Name: n)
<- ER_BinaryRelationship(
OID: rOID, Name: n, isFun1: "False", isFun2: "False") ;

- another Skolem function **#tableOID_2()**:
  - generates a table for a relationship
- the rule applies only to M:N relationships, due to the conditions on **isFun1** and **isFun2**

Columns for each key attributes of entities in M:N relationships (first entity)

```
Rel_Column( OID: #columnOID_2(rOID, role, cOID),
                    Name: eName + kName + rName+ role ,
                    TableOID: #tableOID_2(rOID),
                    IsNullable: "False",
                    IsKey: "True",
                    Type: type)
    <-      Rel_Table(OID: #tableOID_1(eOID), Name: eName),
            Rel_Column( OID: cOID,
                    TableOID: #tableOID_1(eOID),
                    name: kName,
                    isKey: "True",
                    Type: type) ,
            ER_BinaryRelationship(
                    OID: rOID ,
                    name: rName ,
                    isFunct1: "False",
                    isFunct2: "False",
                    entity1OID: eOID,
                    role1: role );
```


Further comments

- With the positional notation, we can omit arguments not needed
- We can refer to the target tables in the bodies of rules:
  - this is recursion here, even if it were not really needed, but it allows for easy generalization to the case with external identification of entities
- We need two rules for technical reasons, but in the tool we have a form of parameterization

Columns for each key attributes of entities in M:N relationships (second entity)

```
Rel_Column( OID: #columnOID_2(rOID, role, cOID),
                    Name: eName + kName + rName+ role ,
                    TableOID: #tableOID_2(rOID),
                    IsNullable: "False",
                    IsKey: "True",
                    Type: type)
    <-      Rel_Table(OID: #tableOID_1(eOID), Name: eName),
            Rel_Column( OID: cOID,
                    TableOID: #tableOID_1(eOID),
                    name: kName,
                    isKey: "True",
                     Type: type) ,
            ER_BinaryRelationship(
                    OID: rOID ,
                    name: rName ,
                    isFunct1: "False",
                    isFunct2: "False",
                    entity2OID: eOID,
                    role2: role );
```

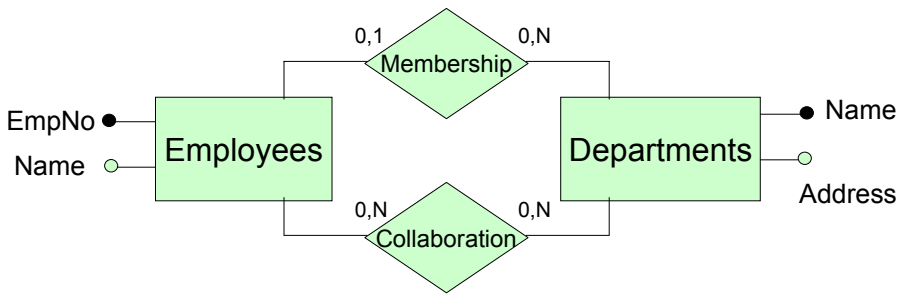

Columns for 1:N relationships

```
Rel_Column(
    OID: #columnOID_2(rOID, role1, cOID),
    Name: name+rName+role1,
    TableOID: #tableOID_1(e1OID),
    IsNullable: isOpt1,  IsKey: "False"
    )
    <-
    ER_BinaryRelationship(
          OID: rOID,
          Role1: role1,
          Entity1OID: e1OID, Entity2OID: e2OID,
          isOpt1: isOpt1,
          isFunct1: "True"
    ),
     Rel_Column( OID: cOID,
          Name: name, TableOID: #tableOID_1(e2OID), isKey: "True");
```

Comment

- We assume that 1:N relationships have
  - **"True"** for **isFunct1**
  - **"False"** for **isFunct2**
- For the names of the column (in the translated schemas), simple solutions are often feasible; in the examples we change names a little bit

A slightly more complex ER schema



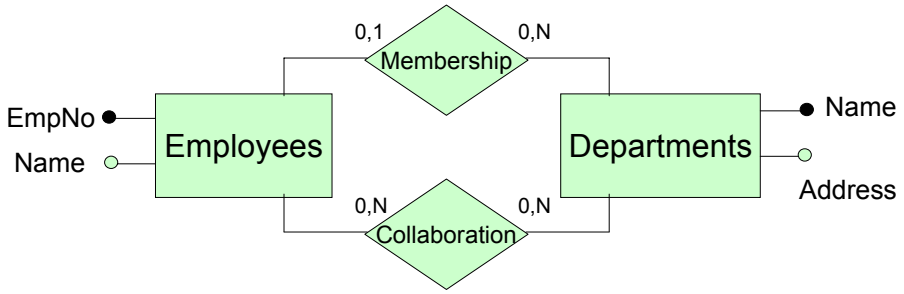
The dictionary for the second E-R schema

ER_Entity	
OID	**Name**
301	Employees
302	Departments

ER_Attribute					
OID	**Name**	**isIdent**	**isNullable**	**Type**	**EntityOID**
401	EmpNo	T	F	Int	301
402	Name	F	F	Text	301
404	Name	T	F	Char	302
405	Address	F	F	Text	302

ER_BinaryRelationship									
OID	**Name**	**eOID1**	**isF1**	**isO1**	**Role1**	**eOID2**	**isF2**	**isO2**	**Role2**
501	Membership	301	T	T		302	F	T	
502	Collaboration	301	F	T		302	F	T	

A translation for the second ER schema



Employees		
EmpNo	**Name**	**Dept**

Departments	
Name	**Address**

Collaboration	
EmpNo	**DeptName**

The translation for the second ER schema, with its dictionary

Employees		
EmpNo	**Name**	**Dept**

Collaboration	
EmpNo	**DeptName**

Departments	
Name	**Address**

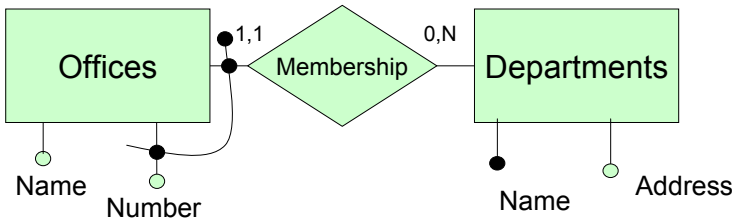
Rel_Table	
OID	**Name**
001	Employees
002	Departments
003	Collaboration

Rel_Column					
OID	**Name**	**isKey**	**isNullable**	**Type**	**TableOID**
101	EmpNo	T	F	Int	001
102	Name	F	F	Text	001
103	Dept	F	T	Char	001
104	Name	T	F	Char	002
105	Address	F	F	Text	002
106	EmpNo	T	F	Int	003
107	DeptName	T	F	Char	003

Exercise

- Extend the model and the rules to handle external identification

Another ER schema and its translation



Offices		
Number	**Dept**	**Name**

Departments	
Name	**Address**

Solution to the exercise

ER_BinaryRelationship									
OID	**Name**	**eOID1**	**isF1**	**isO1**	**Role1**	**IsIdentified**	**Role1**	**eOID2**	**…**

- The value of the new field is **True** if the first entity is identified through the relationship
- There is no need for a second field, as we assume that if there is an entity that is identified through the relationship, then it is the first one (in the same way as we assumed that 1:N relationships have **"True"** for **isFunct1 and "False"** for **isFunct2**)

The dictionary for the third E-R schema

ER_Entity	
OID	**Name**
301	Offices
302	Departments

ER_Attribute					
OID	**Name**	**isIdent**	**isNullable**	**Type**	**EntityOID**
401	Number	T	F	Int	301
402	Name	F	F	Text	301
404	Name	T	F	Char	302
405	Address	F	F	Text	302

ER_BinaryRelationship										
OID	**Name**	**eOID1**	**isF1**	**isO1**	**Role1**	**IsIdent**	**eOID2**	**isF2**	**isO2**	**Role2**
501	Membership	301	T	T		T	302	F	T	

The translation for the third ER schema, with its dictionary

Offices		
Number	**Dept**	**Name**

Departments	
Name	**Address**

Rel_Table	
OID	**Name**
001	Offices
002	Departments

Rel_Column					
OID	**Name**	**isKey**	**isNullable**	**Type**	**TableOID**
101	Number	T	F	Int	001
102	Dept	T	F	Char	001
103	Name	F	F	Text	001
104	Name	T	F	Char	002
105	Address	F	F	Text	002

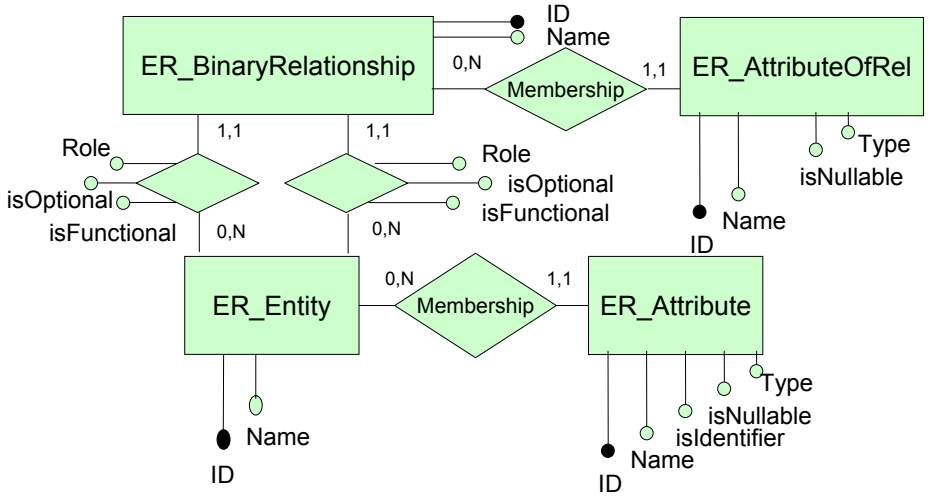
Columns for 1:N relationships: extension

```
Rel_Column(
    OID: #columnOID_2(rOID, role1, cOID),
    Name: name+rName+role1,
    TableOID: #tableOID_1(e1OID),
    IsNullable: isOpt1,  IsKey: isIdent )
    <-
    ER_BinaryRelationship(
          OID: rOID,
          Role1: role1,
          Entity1OID: e1OID, Entity2OID: e2OID,
          isOpt1: isOpt1,
          isFunct1: "True",
          isIdent: isIdent  ),
     Rel_Column( OID: cOID,
          Name: name, TableOID: #tableOID_1(e2OID), isKey: "True");
```

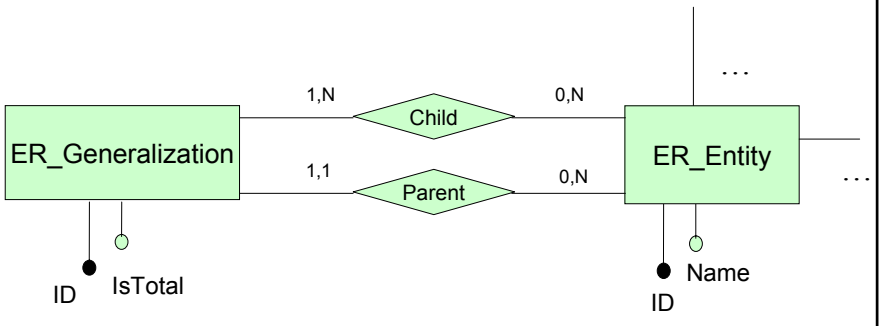

More models

- Relation with foreign keys
- Object relational
- Binary ER model with
  - attributes for relationships
  - generalizations
  - relationships also over relationships
  - multivalued attributes
  - nested attributes
- N-ary ER model
  - also with (as above)
- ADM (logical model for Web sites, with nested attributes)
- UML class diagram
- Java class structure

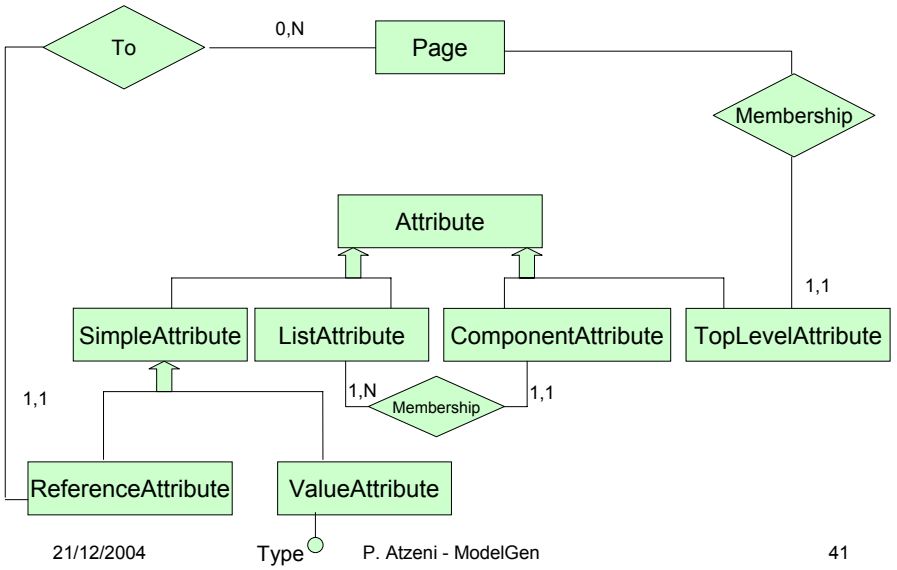
The ER model with attributes for relationships



The ER model with generalizations



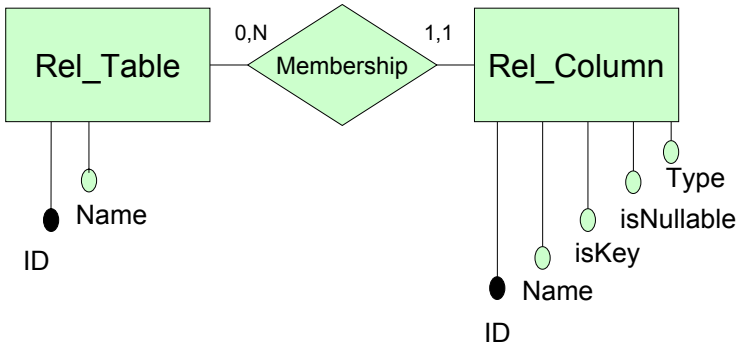
ADM



Towards a metamodel

- We would like to describe constructs
- Let us consider the various models we have seen so far

The (conceptual) description of relational databases

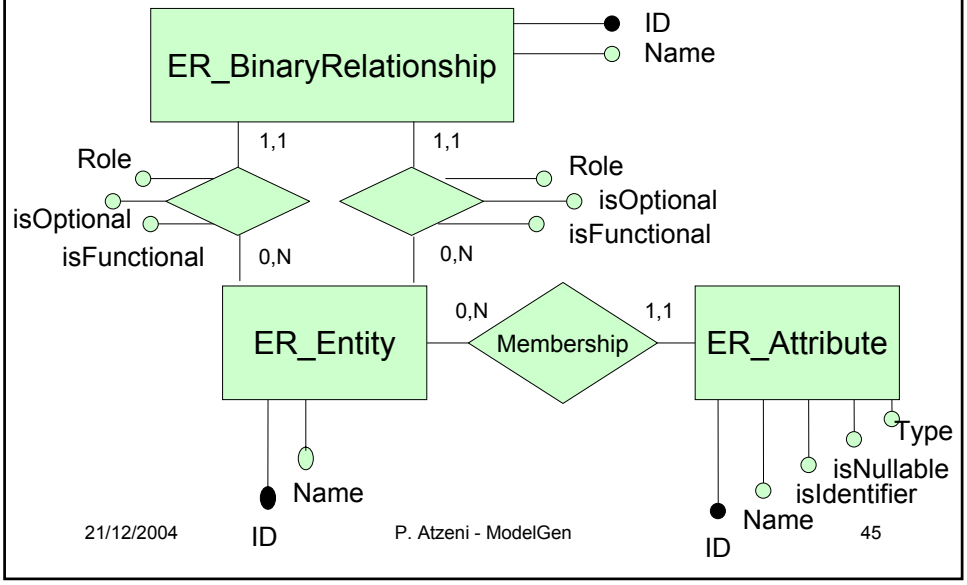


The (logical) description of relational databases

Rel_Table	
OID	**Name**

Rel_Column					
OID	**Name**	**isKey**	**isNullable**	**Type**	**TableOID**

The (conceptual) description of binary ER schemas



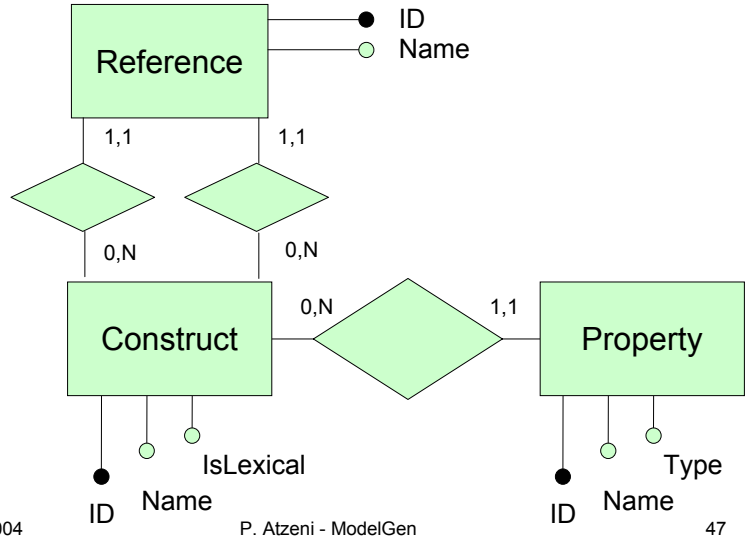
The (logical) description of binary ER schemas

ER_Entity	
OID	**Name**

ER_Attribute					
OID	**Name**	**isIdent**	**isNullable**	**Type**	**TableOID**

ER_BinaryRelationship									
OID	**Name**	**eOID1**	**isF1**	**isO1**	**Role1**	**eOID2**	**isF2**	**isO2**	**Role2**

The conceptual schema for a metadictionary



The logical schema for a metadictionary

Construct		
OID	Name	IsLexical

Property			
OID	Name	Construct	Type

Reference			
OID	Name	Construct	Target

The relational model in the metadictionary

Construct		
OID	**Name**	**IsLex**
1	Rel_Table	F
2	Rel_Column	T

Property			
OID	**Name**	**Construct**	**Type**
11	Name	1	String
12	Name	2	String
13	IsKey	2	Boolean
14	IsNullable	2	Boolean
15	Type	2	String

Reference			
OID	**Name**	**Construct**	**Target**
21		2	1

The ER model in the metadictionary

Construct		
OID	**Name**	**IsLex**
1	ER_Entity	F
2	ER_Attribute	T
3	ER_Relationship	F

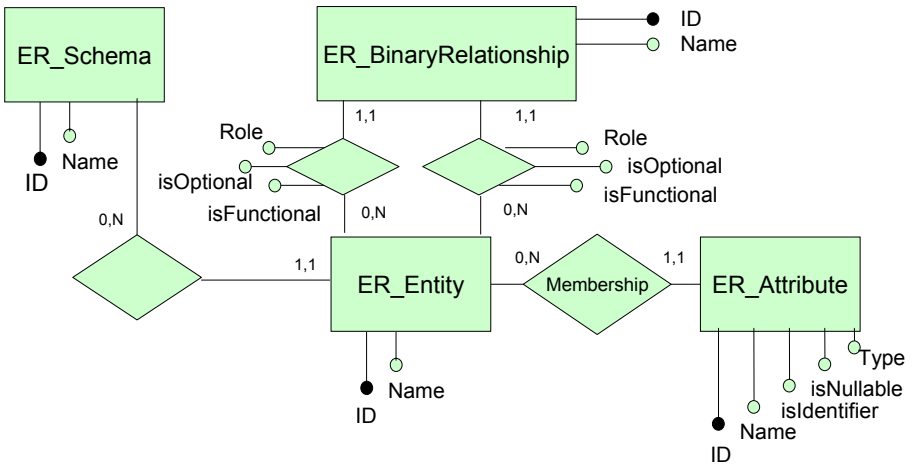
Property			
OID	**Name**	**Construct**	**Type**
11	Name	1	String
12	Name	2	String
13	IsIdentifier	2	Boolean
14	IsNullable	2	Boolean
15	Type	2	String
16	IsFunct1	3	Boolean
17	IsOptional1	3	Boolean
18	Role1	3	String
19	IsFunct2	3	Boolean
20	IsOptional2	3	Boolean
21	Role2	3	String

Reference			
OID	**Name**	**Construct**	**Target**
31	Entity	2	1
32	Entity1	3	1
33	Entity2	3	1

More models and more schemas

- In the dictionary and in the meta-dictionary seen so far we handle one schema or one model at a time
- However we can handle more if we add
  - mention of schemas in the dictionary
  - mention of models in the metadictionary

A dictionary for handling various ER schemas (conceptual level)



Comment

- In general, **ER_Schema** should be related to all entities, but we have not indicated it in the diagram, as all constructs "depend" on **ER_Entity**
- However, for convenience, we repeat it for each table in the relational implementation

A dictionary for handling various ER schemas (logical level)

ER_Entity

OID	Schema	Name
301	1	Employees
302	1	Departments
303	2	Employees
304	2	Departments

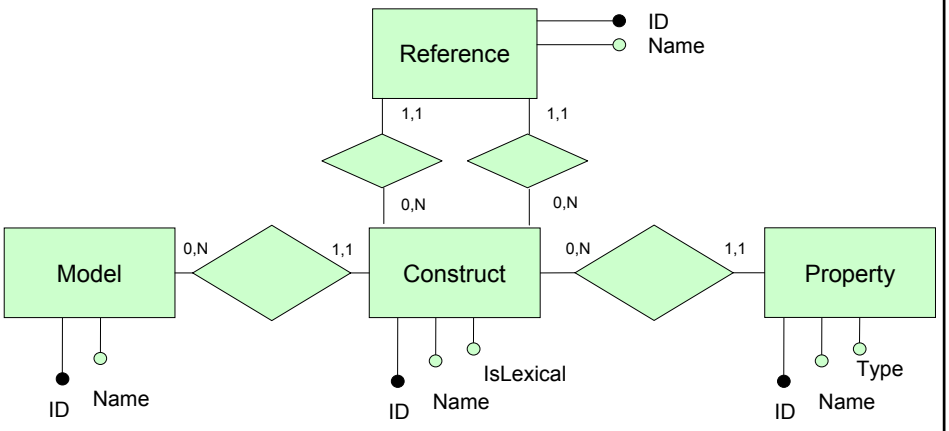
ER_Attribute

OID	Schema	Name	isIdent	isNullable	Type	EntityOID
401	1	EmpNo	T	F	Int	301
402	1	Name	F	F	Text	301
404	1	Name	T	F	Char	302
405	1	Address	F	F	Text	302
406	2	EmpNo	T	F	Int	301
407	2	Name	F	F	Text	301
408	2	Name	T	F	Char	302
409	2	Address	F	F	Text	302

ER_BinaryRelationship

OID	Schema	Name	eOID1	isF1	isO1	Role1	eOID2	isF2	isO2	Role2
501	1	Membership	301	T	T		302	F	T	
502		Membership	303	T	T		304	F	T	
503	2	Collaboration	303	F	T		304	F	T	

A metadictionary handling various models (conceptual level)



A metadictionary handling various models (logical level)

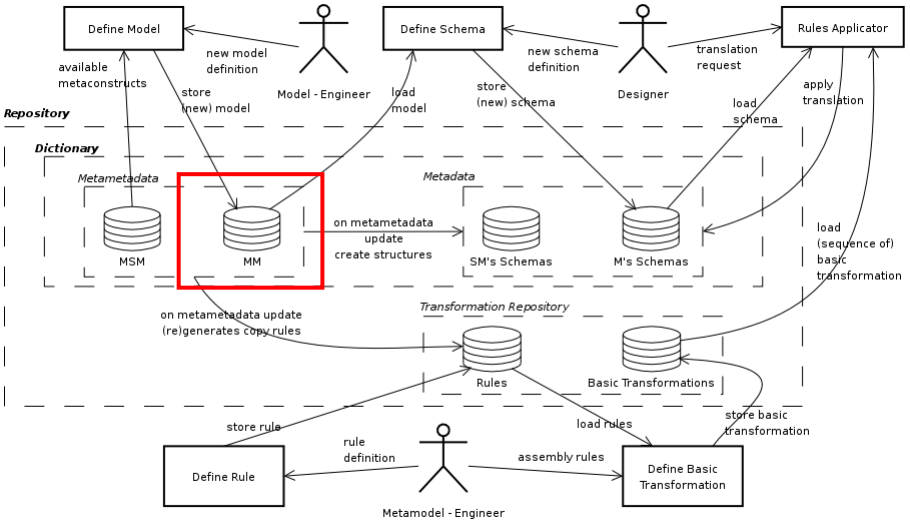
Model	
OID	**Name**
1	Relational
2	Entity-Relationship

Construct			
OID	**Model**	**Name**	**IsLex**
1	1	Rel_Table	F
2	1	Rel_Column	T
3	2	ER_Entity	F
4	2	ER_Attribute	T
5	2	ER_Relationship	F

Reference			
OID	**Name**	**Construct**	**Target**
30		2	1
31	Entity	4	3
32	Entity1	5	3
33	Entity2	5	3

Property			
OID	**Name**	**Construct**	**Type**
11	Name	1	String
12	Name	2	String
13	IsKey	2	Boolean
14	IsNullable	2	Boolean
15	Type	2	String
16	Name	3	String
17	Name	4	String
18	IsIdentifier	4	Boolean
19	IsNullable	4	Boolean
20	Type	4	String
21	IsFunct1	5	Boolean
22	IsOptional1	5	Boolean
23	Role1	5	String
24	IsFunct2	5	Boolean
25	IsOptional2	5	Boolean
26	Role2	5	String

We have seen another part of the dictionary



The Supermodel

- A model that includes all the constructs (in their most general forms)
  - aggregations of lexicals (e.g., tables)
  - components of aggregations of lexicals (e.g., columns)
  - abstracts (e.g., entities or classes)
  - attributes of abstracts
  - binary aggregations of abstracts
- Since the supermodel is a model, we can describe it in a meta-dictionary (with one model only)

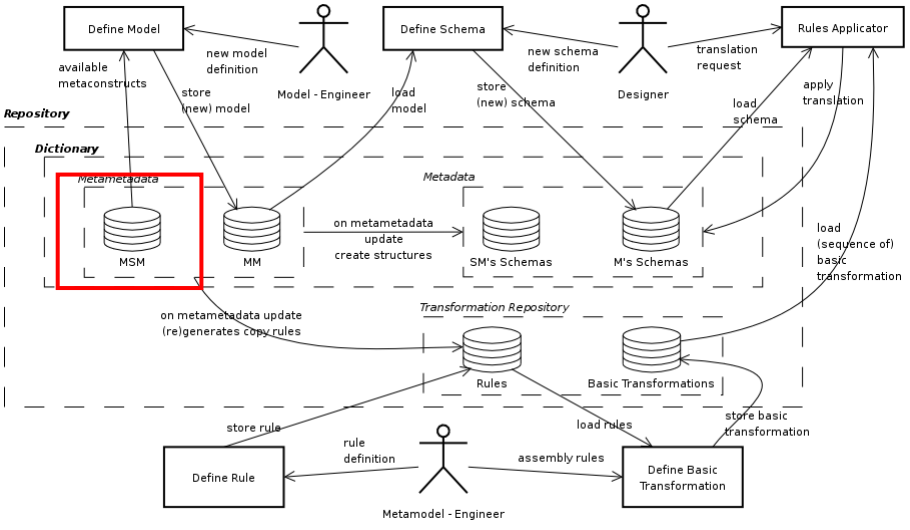
The metadictionary for the supermodel

Construct		
OID	**Name**	**IsLex**
1	AggregationOfLexicals	F
2	ComponentOfAggrOfLex	T
3	Abstract	F
4	AttributeOfAttribute	T
5	BinaryAggregationOfAbstracts	F

Reference			
OID	**Name**	**Construct**	**Target**
30		2	1
31	Abstract	4	3
32	Abstract1	5	3
33	Abstract2	5	3

Property			
OID	**Name**	**Construct**	**Type**
11	Name	1	String
12	Name	2	String
13	IsKey	2	Boolean
14	IsNullable	2	Boolean
15	Type	2	String
16	Name	3	String
17	Name	4	String
18	IsIdentifier	4	Boolean
19	IsNullable	4	Boolean
20	Type	4	String
21	IsFunct1	5	Boolean
22	IsOptional1	5	Boolean
23	Role1	5	String
24	IsFunct2	5	Boolean
25	IsOptional2	5	Boolean
26	Role2	5	String

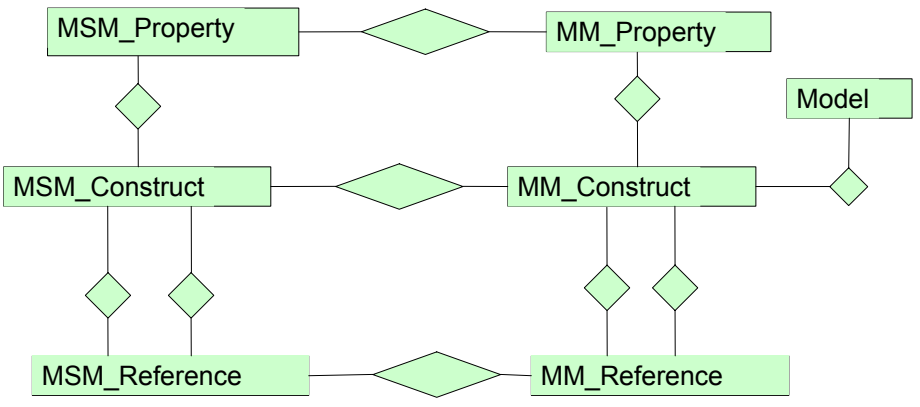
In the tool … the metadictionary for SM



In the tool: MM and MSM

- MSM describes the allowed constructs
- MM describes models in terms of a set of MSM constructs, by specifying which are included and how they are renamed

MM and MSM



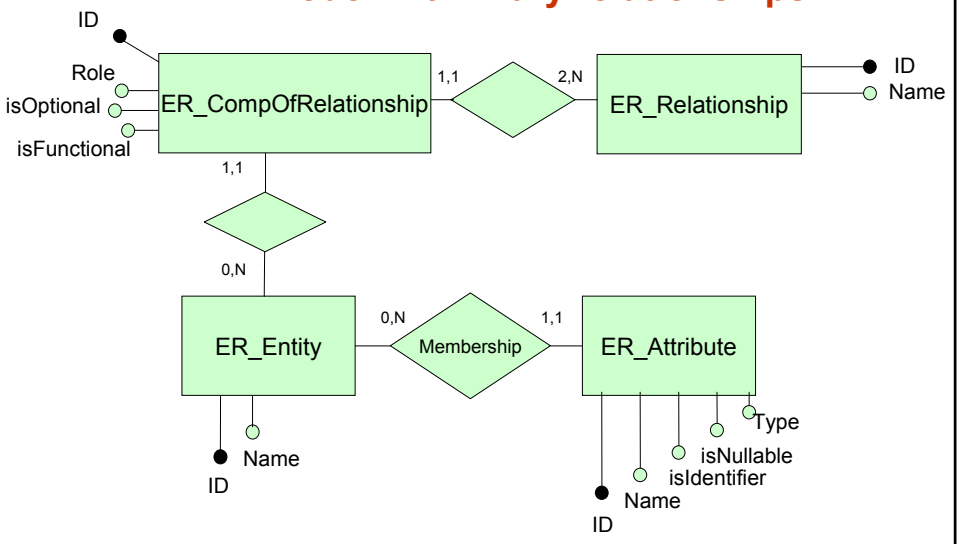
Homework

1. ER model with N-ary relationships
2. ER model with relationships over relationships
3. Object-relational model, with both value based and ID-based tables (non-nested attributes only, but references to ID-based tables in both)
4. Simple object database model, with classes and value based and reference attributes (non-nested only)
5. Relational model with referential integrity

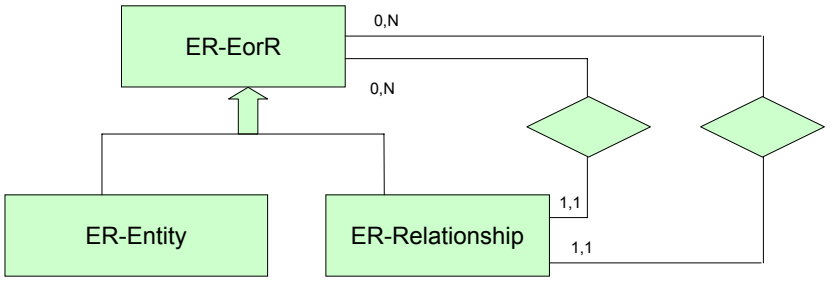
Homework, comments

- In some cases, there are various possible solutions, and we just show one
- The tool is flexible and therefore one can later change

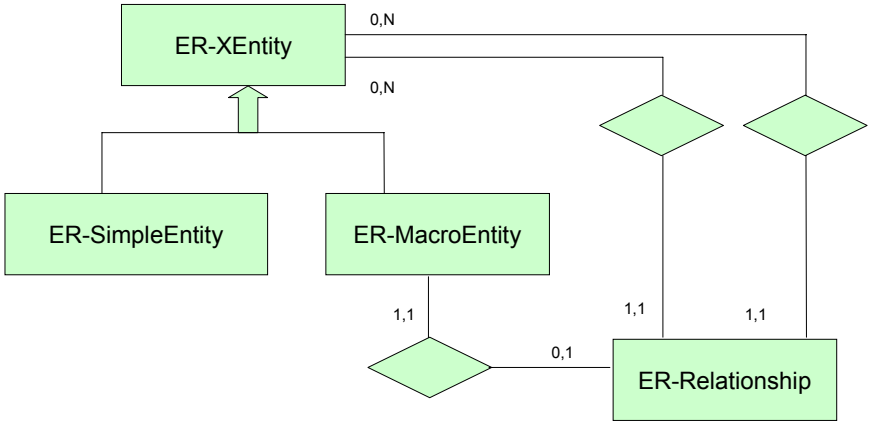
1. ER model with N-ary relationships



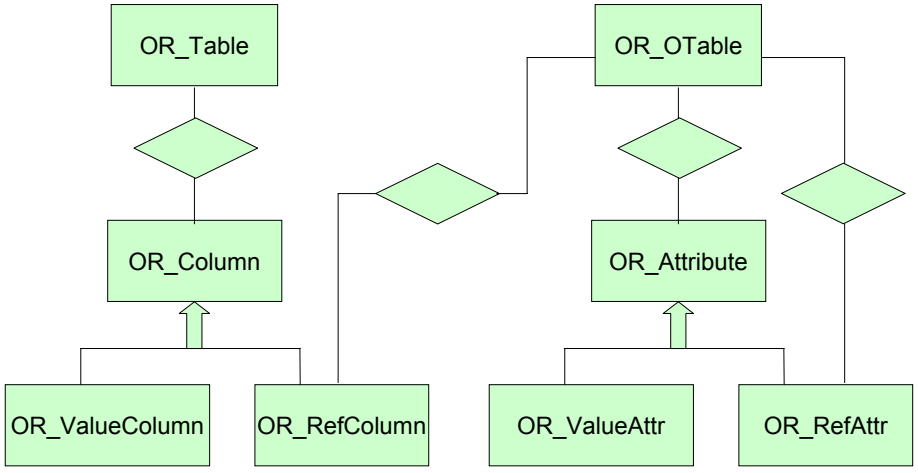
2. ER w/ relationships over relationships (a)



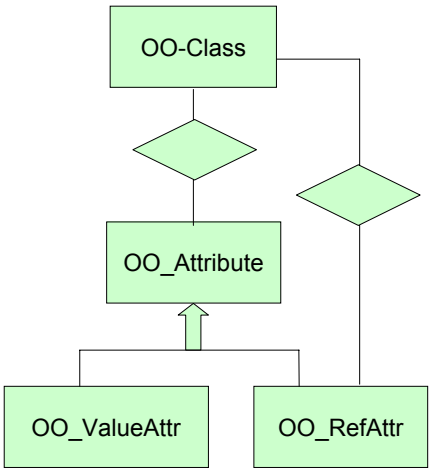
2. ER w/ relationships over relationships (b)



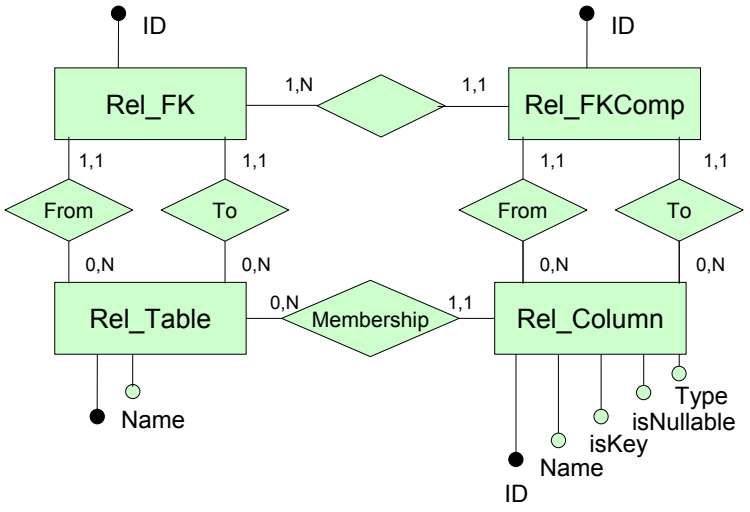
3. Object-relational model, with …



4. Simple object database model



5. Relational model with referential integrity



The Supermodel

- A model that includes all the constructs (in their most general forms)
  - aggregations of lexicals (e.g., tables)
  - components of aggregations of lexicals (e.g., columns)
  - abstracts (e.g., entities or classes)
  - attributes of abstracts
  - binary aggregations of abstracts
- Since the supermodel is a model, we can describe it in a meta-dictionary (with one model only)

A small supermodel (for simple ER & OO)

- Abstract
- Lexical Attribute Of Abstract
- Binary Aggregation Of Abstracts
- Abstract Attribute Of Abstracts

Simple ER in the simple supermodel

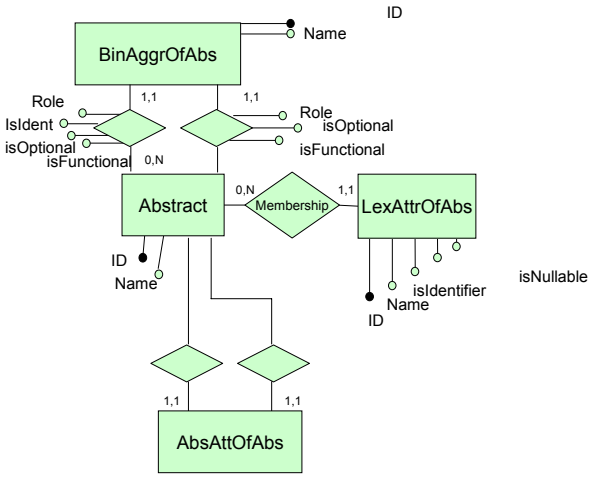
- Entity: Abstract
- Attribute of Entity: Lexical Attribute Of Abstract
- Binary Relationship: Binary Aggregation Of Abstracts

Simple OO in the simple supermodel

- Class: Abstract
- Attribute of Class: Lexical Attribute Of Abstract

- Reference to Class: Abstract Attribute Of Abstracts

A small supermodel (for simple ER & OO)



A small supermodel (for simple ER & OO)

Abstract		
OID	**Schema**	**Name**
1	1	Emp
2	1	Dept
3	2	Emp
4	2	Dept

LexicalAttributeOfAbstract						
OID	**Schema**	**Name**	**isIdent**	**isNullable**	**Type**	**Abstract**
11	1	EmpNo	T	F	Int	1
12	1	Name	F	F	Text	1
…						
16	2	EmpNo	T	F	Int	3
…						

AbstractAttributeOfAbstract						
OID	**Schema**	**Name**	**isIdent**	**isOptional**	**AbstractTo**	**Abstract**
51	2	Memb	F	T	4	3

BinaryAggregationOfAbstracts											
OID	**Schema**	**Name**	**Ent1**	**isF1**	**IsIdent**	**isO1**	**Role1**	**Ent2**	**isF2**	**isO2**	**Role2**
71	1	Memb	1	T	F	T		2	F	T	

The supermodel: two approaches

- Fewer constructs, with variations
  - example:
    - aggregation of abstracts, both for binary and N-ary models
- More constructs, classifying special cases separately
  - example:
    - binary aggregation of abstracts
    - N-ary aggregation of abstracts

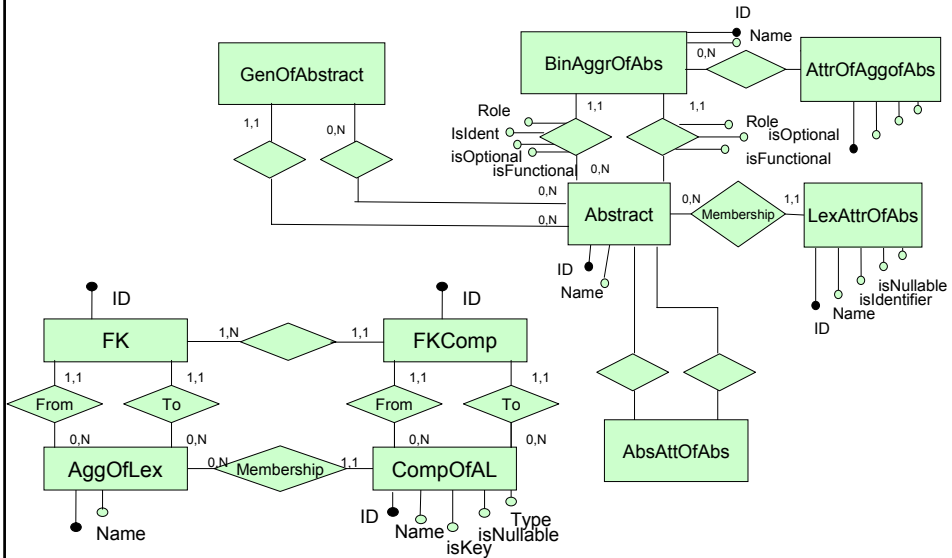
The supermodel for homework 2

- Abs Abstract
- LeA Lexical Attribute Of Abstract
- AbA Abstract Attribute Of Abstract
- BAg Binary Aggregation Of Abstracts
- AAg Attribute Of Aggregation Of Abstracts
- GAb Generalization Of Abstracts
- CGA Component of Generalization of Abstracts
- AgL Aggregation Of Lexicals
- CAL Component Of Aggregation Of Lexicals
- FKA Foreign Key For Aggregation Of Lexicals
- CFK Component of Foreign Key For Aggregation Of Lexicals

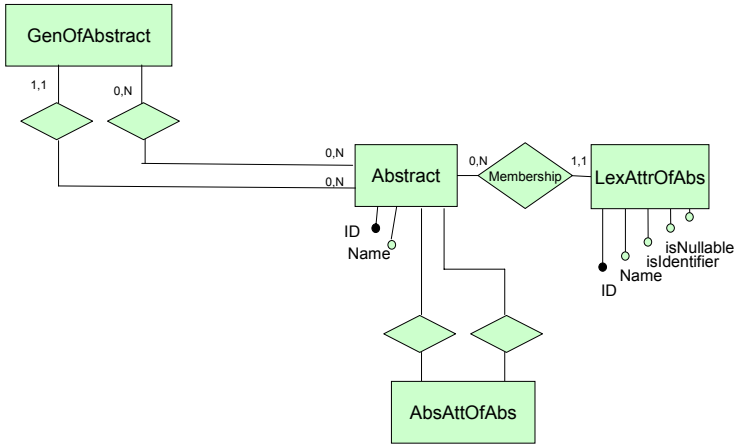
Models of interest for homework 2

- OO with generalizations
  - Abs LeA AbA GAb CGA
- OO without generalizations
  - Abs LeA AbA
- Full ER
  - Abs LeA BAg AAg GAb CGA
- ER without generalizations
  - Abs LeA BAg AAg
- ER without attributes on (binary) relationships
  - Abs LeA BAg GAb CGA
- ER without M:N relationships
  - Abs LeA BAg (IsFun1=True) AAg GAb CGA
- Relational
  - AgL CAL FKA CFK

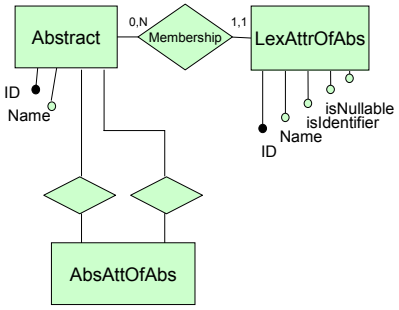
The supermodel (some attributes missing)



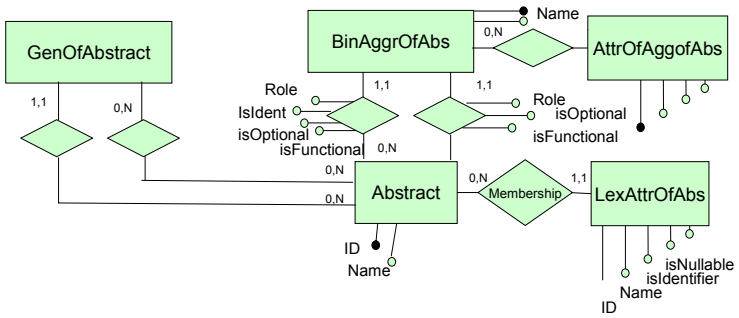
OO with generalizations



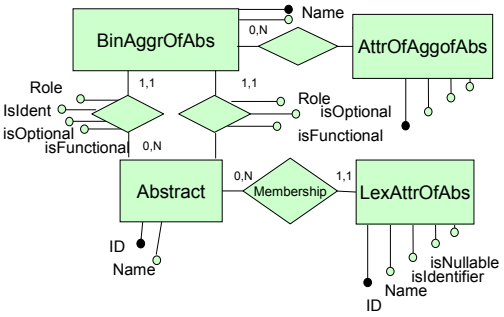
OO without generalizations



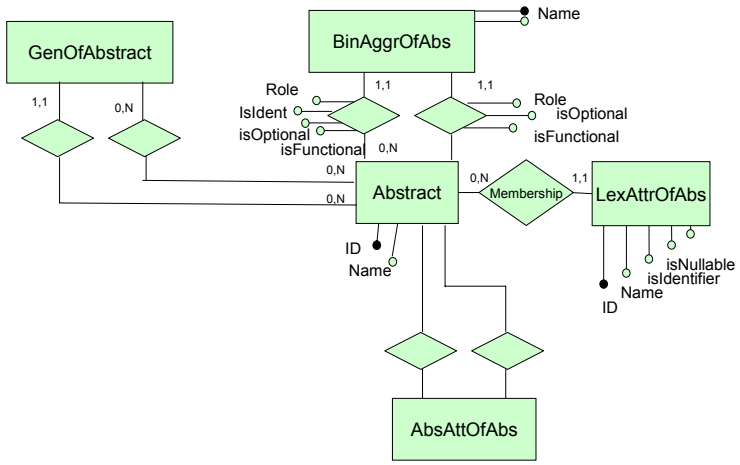
Full ER



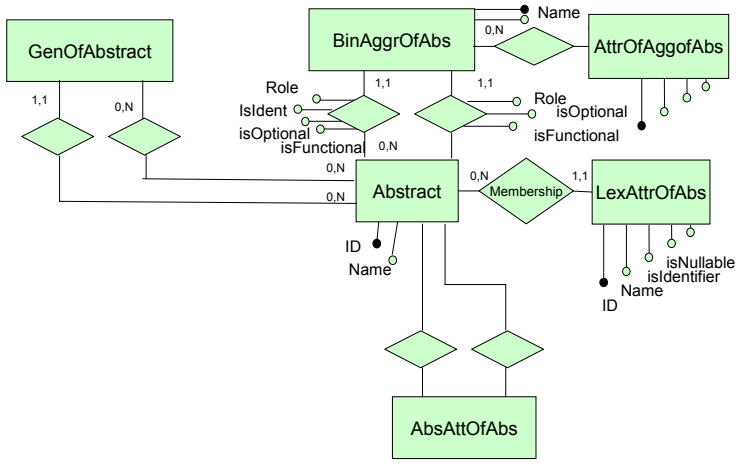
ER without generalizations



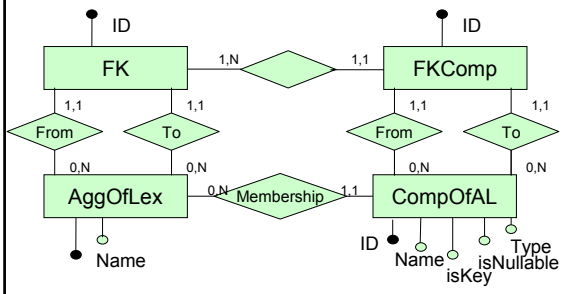
ER without attributes on (binary) relationships



ER without M:N relationships (IsFun1=True)



Relational



Basic translations for homework 2

1. OO with gen → OO without gen (parent only)
2. OO with gen → OO without gen (children only)
3. OO with gen → OO without gen (parent and children)
4. 5. 6. ER with gen → ER without gen (three alternatives)
7. OO → (Full) ER
8. Full ER → ER without attributes for relationships
9. ER w/o attributes for relationships → ER w/o M:N relationships
10. ER w/o M:N relationships and w/o attributes for rels → OO
11. OO without gen → relational
12. relational → OO
13. ER without gen → relational
14. relational → ER

Comments, 1-6

- Elimination of hierarchies as usual in early logical design
- Hierarchies can have multiple levels, and all levels have to be translated in the same way.
- If you encounter difficulties, assume that there is no multiple inheritance and so hierarchies are trees

Comments, 7-10

7 & 10 ER and OO differ because ER has aggregations whereas OO has attributes that are references to abstracts

8 & 9 Elimination of attributes in relationships and of M:N relationships is done by promoting relationships (aggregations) to entities (abstracts)

Comments, 11-14

- Value based references in the relational model (described by foreign keys) have to be translated into aggregations (ER model) or abstract attributes (OO model) and viceversa