

# Collection Type Constructors in Entity-Relationship Modelling

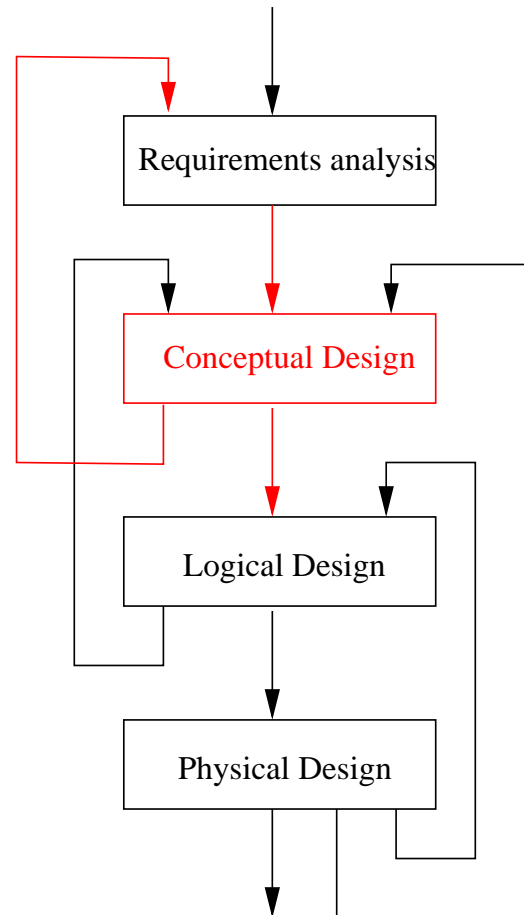
Sven Hartmann, Sebastian Link

*Information Science Research Centre, Massey University, New Zealand*

**This research is supported by the Marsden Fund Council from Government funding, administered by the Royal Society of New Zealand.**

## Conceptual Modelling in the Design Phase

- ▶ ER can provide well-defined and natural features
- ▶ ER can provide safe features leading to good database design
- ▶ IDNF



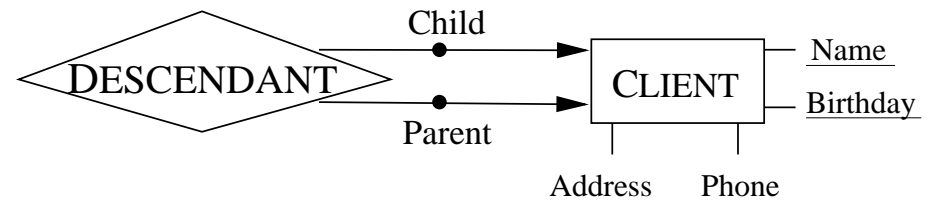
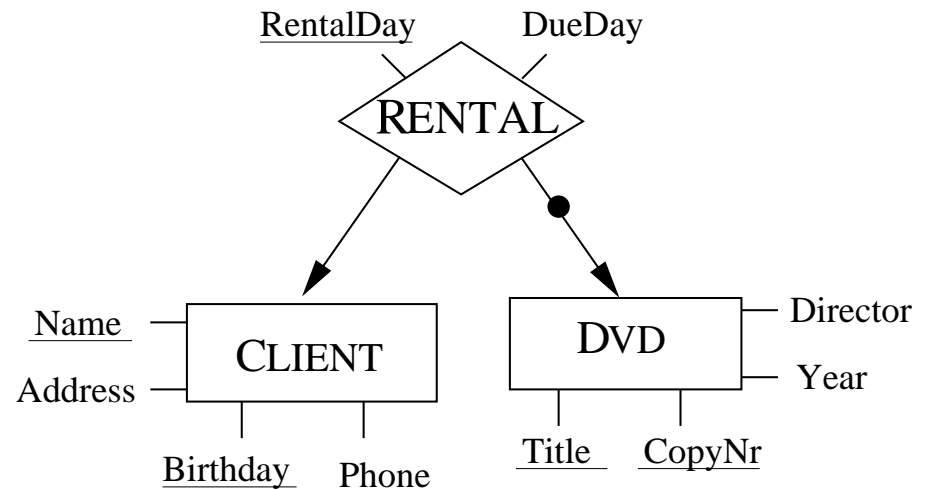
- ▶ conceptual models must provide means for communication between designer and user
- ▶ modelling as natural as possible
- ▶ how to model collections directly?
- ▶ how to map collections down?

## Motivations for Directly Modelling Collections

- ▶ collections naturally appear as logical units in the real world
  - ↪ bags of purchases, bit sequences, rankings, lists of transactions
- ▶ the world is not flat (really!)
  - ↪ data becomes more and more complex
- ▶ other data models support collections
  - ↪ nested relational, object-relational, object-oriented, XML, RDFS
- ▶ collections appear in natural language
  - ↪ plurals
- ▶ collections have their representation in nested attributes
  - ↪ records, lists, sets, bags, unions
- ▶ but: collection constructors do not exist in ER-modelling
  - ↪ despite existence of constructors for records and disjoint unions

## EER Features: Entity and Relationship Types

- ▶  $E = (attr(E), id(E))$
- ▶  $R = (comp(R), attr(R), id(R))$
- ▶ *order* of object type  $O$  is
  - 0, if  $O$  entity type,
  - $k$ , if  $k - 1$  is max order of any component of  $O$



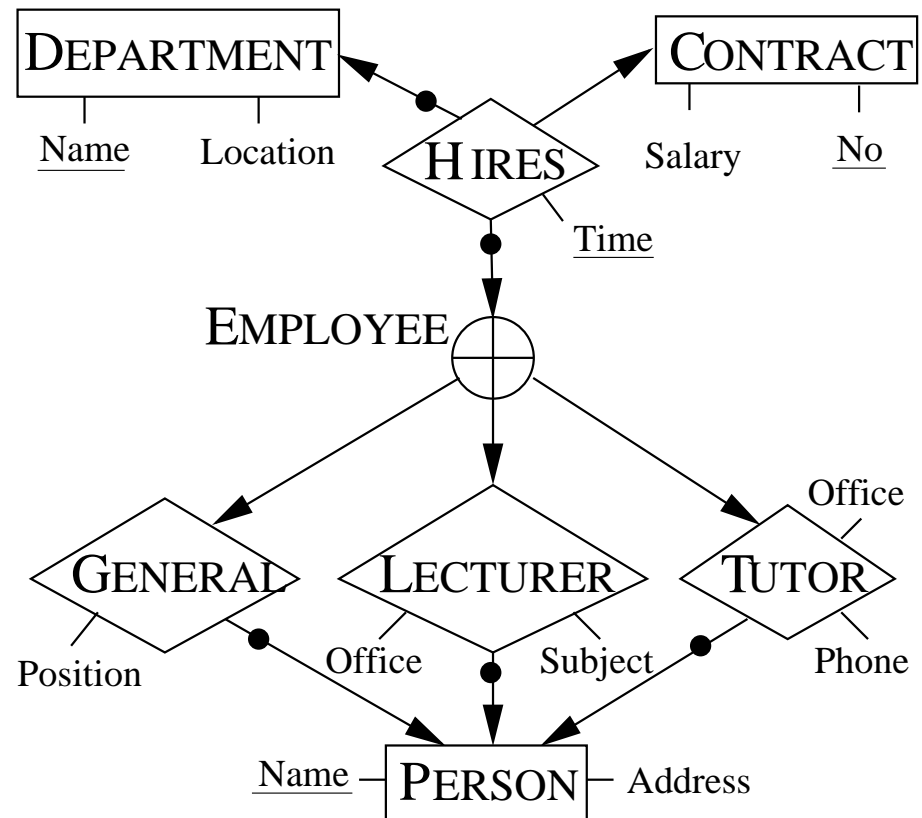
## EER Features: Specialisation and Generalisation

- ▶ specialisation:

$$S = (\{C\}, attr(R), \{C\})$$

- ▶ generalisation:

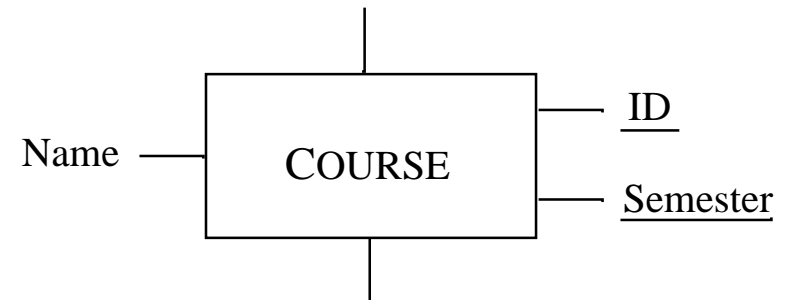
cluster type  $C = O_1 \oplus \dots \oplus O_k$   
 with  $comp(C) = \{O_1, \dots, O_k\}$



## EER Features: Nested Attributes

- ▶ flat attributes  $\mathcal{A}$ , labels  $\mathcal{L}$  and null attribute  $\lambda$
- ▶  $\mathcal{N}$  over  $\mathcal{A}$  and  $\mathcal{L}$ :
  - $\mathcal{A} \subseteq \mathcal{N}$ ,  $\lambda \in \mathcal{N}$
  - $N_1, \dots, N_k \in \mathcal{N}$ ,  $L \in \mathcal{L}$ :  
 $L(N_1, \dots, N_k) \in \mathcal{N}$ ,  
 $L(N_1 \oplus \dots \oplus N_k) \in \mathcal{N}$
  - $N \in \mathcal{N}$ ,  $L \in \mathcal{L}$ :  
 $L[N]$ ,  $L\{N\}$ ,  
 $L\{\{N\}\}$ ,  $L[N] \in \mathcal{N}$
- ▶ no concepts of  
 $\hookrightarrow$  LECTURER, ARTICLE

LECTURERS<sub>s</sub> { LECTURER (Name, E-Mail, Department) }



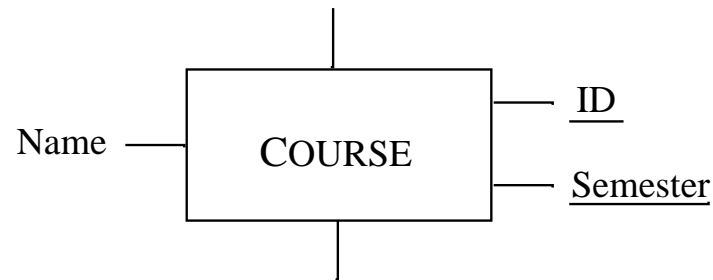
READINGS<sub>s</sub> [ R EADING ( ARTICLE (Title, Citation)  $\oplus$   
 BOOK (ISBN, Chapter, Title)) ]

## Our Proposal: Collection Type Constructors

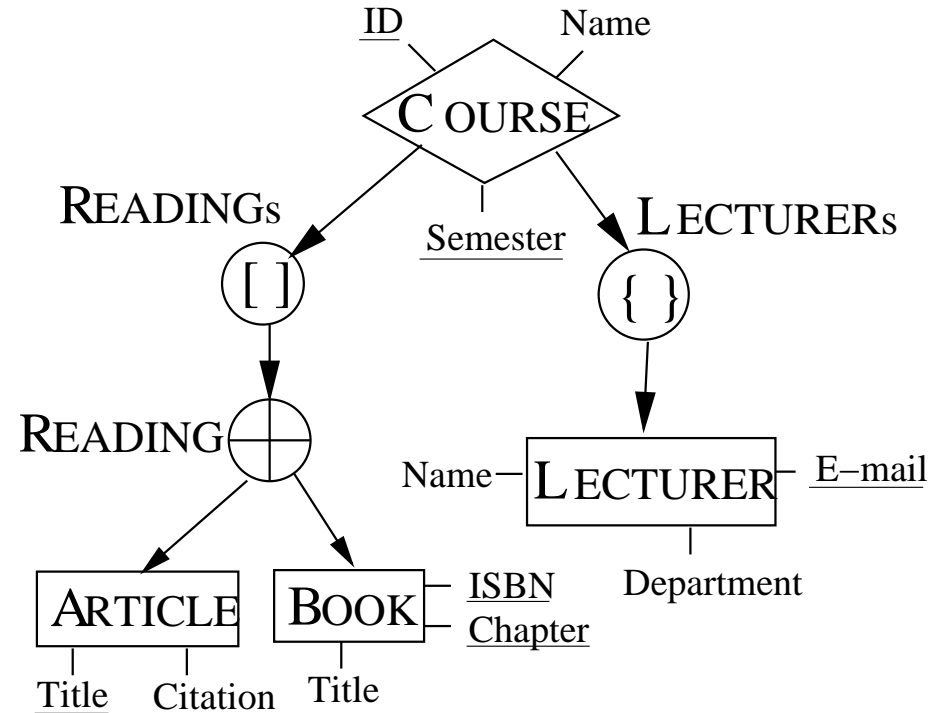
- ▶ collections:
  - ↪ lists (duplicates, order),
  - ↪ sets (no duplicates, no order),
  - ↪ bags (duplicates, no order),
  - ↪ rankings (no duplicates, order)
  
- ▶ notation:
  - ↪ braces: order does not matter, brackets: order does matter
  - ↪ single parenthesis: no duplicates, double parenthesis: duplicates
  
- ▶ *list-, set-, bag-, ranking-*type  $U$  with  $comp(U) = \{C\}$ 
  - ↪ list type  $U[[C]]$  and ranking type  $U[C]$
  - ↪ bag type  $U\{\{C\}\}$  and set type  $U\{C\}$

## Collection Types vs. Nested Attributes

LECTURERS<sub>s</sub> { LECTURER (Name,E-Mail,Department) }



READINGS<sub>s</sub> [ READING ( ARTICLE (Title,Citation) ⊕  
BOOK (ISBN,Chapter,Title)) ]



- ▶ new concepts: LECTURER, ARTICLE, BOOK, READING etc.
- ▶ naturally reflect interdependence of natural language sentences
- ▶ easy to plug-in

## Semantics of Collection Types

- ▶ use  $U(C)$  to denote collection type with component  $C$
- ▶ object set  $\mathcal{I}(U)$  associated with  $U(C)$  contains
  - ↪ finite lists (sets, bags, rankings, resp.) of objects in  $\mathcal{I}(C)$
- ▶ *key* of  $U(C)$  is  $U(C)$  itself
- ▶ *object type* may refer to entity, relationship, cluster or collection type
- ▶ *object* may refer to entity, relationship or collection
- ▶ *instance*  $\mathcal{I}$  of ER schema  $\mathcal{S}$  requires now additionally:

$$\forall U(C) \in \mathcal{S}. \forall O \in \mathcal{I}(U). (o \in O \Rightarrow o \in \mathcal{I}(C))$$

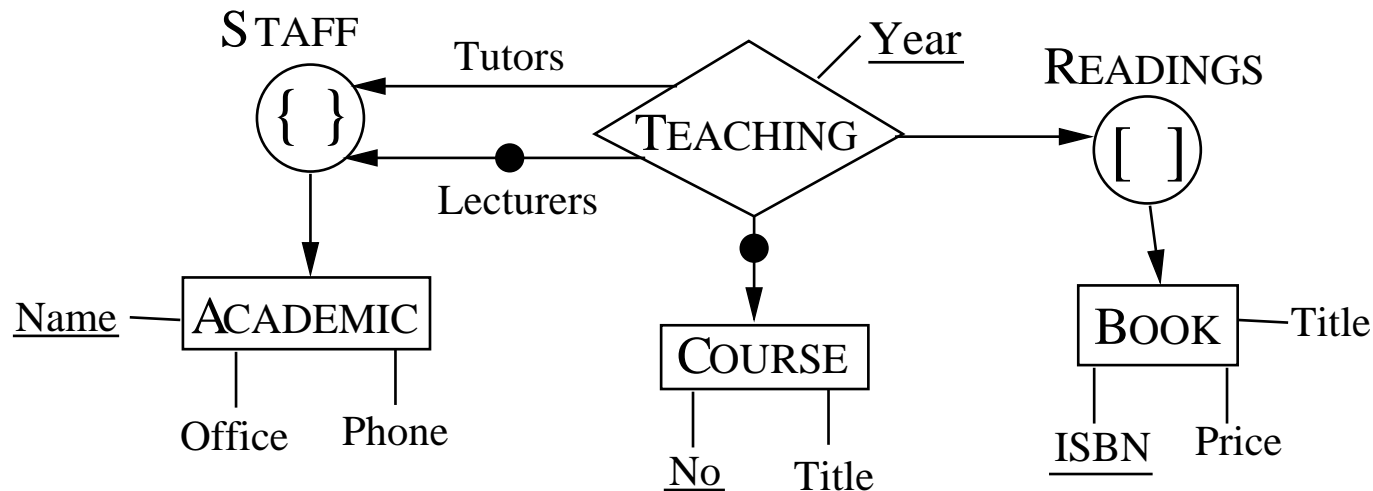
## Natural Fit for Natural Language Correspondences

English sentence concept	EER feature
transitive verb	relationship type
common noun	component of relationship type
adjective	attribute of component
adverb	attribute of relationship type
numerical expression	attribute of object type
preposition	role name of component
gerund	relationship type that is component of another relationship type
clause	relationship type with components
complex sentence	relationship type of order higher than 1
alternative phrase	cluster type
<b>plural</b>	<b>collection type</b> /nested attribute
“is a” sentence	specialisation

- ▶ Peter P.-S. Chen: *English Sentence Structure and ER Diagrams*, Inf. Sci. 29(2-3): 127-149, 1983
- ▶ S. Hartmann, S. Link: *English Sentence Structure and EER Modeling*, CRPIT 69: 27-35, 2007.

## An Example ER Diagram with Collection Types

- ↪  $\text{ACADEMIC} = (\{\text{Name, Phone, Office}\}, \{\text{Name}\})$ ,
- ↪  $\text{BOOK} = (\{\text{ISBN, Price, Title}\}, \{\text{ISBN}\})$ ,
- ↪  $\text{COURSE} = (\{\text{No, Title}\}, \{\text{No}\})$ ,
- ↪  $\text{STAFF} \{ \text{ACADEMIC} \}, \text{READINGS} [ \text{BOOK} ]$
- ↪  $\text{TEACHING} = (\{\text{COURSE, Lecturers:STAFF, Tutors:STAFF, READINGS}\}, \{\text{Year}\}, \{\text{COURSE, Lecturers:STAFF, Year}\})$



## An Example ER Instance with Collections

COURSE	
No	Title
157266	Data Modeling
157357	IS Security

BOOK		
Title	ISBN	Price
ER Modeling	3540654704	138.-
DB Design	0201565234	90.-
Cryptography	0130914290	87.-
Viruses	0471007684	123.-

ACADEMIC		
Name	Office	Phone
Sven	2.09	7308
Sebastian	2.10	2717
Ernie	2.33	0077
Bert	2.33	0077

READINGS
[3540654704, 0201565234]
[0130914290, 0471007684]

STAFF
{Sven, Sebastian}
{Sebastian}
{Ernie, Bert}
∅

TEACHING				
COURSE	Year	READINGS	Lecturers:STAFF	Tutors:STAFF
157266	2007	[3540654704, 0201565234]	{Sven, Sebastian}	{Ernie, Bert}
157357	2007	[0130914290, 0471007684]	{Sebastian}	∅

## Transformation to the Relational Model of Data

- ▶ How to preserve structural properties of collections in flat tables?
- ▶ surrogate flat attribute  $U'_{ID}$  identifies  $\mathcal{I}(U)$ -objects
  - ↪ bag type  $U\{\{C\}\}$  requires info on multiplicity: surrogate  $U'_{Mul}$
  - ↪ ranking/list types  $U(C)$  need info on position: surrogate  $U'_{Pos}$
- ▶ each collection type  $U(C)$  results in following relation schema  $U'$ 
  - ↪ key attributes of corresponding component schema  $C'$  and  $U'_{ID}$
  - ↪ if component  $C$  is collection type, then its key attribute is  $C'_{ID}$
  - ↪ for ordered collection types additionally  $U'_{Pos}$
  - ↪ for bag type additionally  $U'_{Mul}$

## Example Translation - Schema Level

### ► recall ER Schema

- ↪  $\text{ACADEMIC} = (\{\text{Name, Phone, Office}\}, \{\text{Name}\})$ ,
- ↪  $\text{BOOK} = (\{\text{ISBN, Price, Title}\}, \{\text{ISBN}\})$ ,
- ↪  $\text{COURSE} = (\{\text{No, Title}\}, \{\text{No}\})$ ,
- ↪  $\text{STAFF}\{\text{ACADEMIC}\}, \text{READINGS}[\text{BOOK}]$
- ↪  $\text{TEACHING} = (\{\text{COURSE, Lecturers:STAFF, Tutors:STAFF, READINGS}\}, \{\text{Year}\},$   
 $\{\text{COURSE, Lecturers:STAFF, Year}\})$

### ► corresponding relational database schema

- ↪  $\text{ACADEMIC}' = \{\text{Name, Phone, Office}\}$  with minimal key  $\{\text{Name}\}$ ,
- ↪  $\text{BOOK}' = \{\text{ISBN, Price, Title}\}$  with minimal key  $\{\text{ISBN}\}$ ,
- ↪  $\text{COURSE}' = \{\text{No, Title}\}$  with minimal key  $\{\text{No}\}$ .
- ↪  $\text{STAFF}' = \{\text{ACADEMIC.Name, Staff}'_{ID}\}$
- ↪  $\text{READINGS}' = \{\text{BOOK.ISBN, Readings}'_{ID}, \text{Readings}'_{Pos}\}$
- ↪  $\text{TEACHING}' = \{\text{COURSE.No, Lecturers:Staff}'_{ID}, \text{Tutors:Staff}'_{ID}, \text{Readings}'_{ID}, \text{Year}\}$

## Example Translation - Instance Level

READINGS'		
ISBN	Readings' <sub>ID</sub>	Readings' <sub>Pos</sub>
3540654704	1	1
0201565234	1	2
0130914290	2	1
0471007684	2	2

STAFF'	
Name	Staff' <sub>ID</sub>
Sven	2
Sebastian	2
Sebastian	1
Ernie	3
Bert	3

TEACHING'				
CourseNo	Year	Readings' <sub>ID</sub>	Lecturers:Staff' <sub>ID</sub>	Tutors:Staff' <sub>ID</sub>
157266	2007	1	2	3
157357	2007	2	1	0

## Minimal Keys for Collection Types

- ▶ For  $U(C)$  which minimal keys do we specify on relation schema  $U'$ ?
- ▶ ordered collection types  $U(C)$  define  $\{U'_{ID}, U'_{Pos}\}$ 
  - ↪  $U'$ -tuple identified by collection and position in this collection
- ▶ ranking type  $U[C]$  also results in minimal key  $\{U'_{ID}\} \cup k_{attr}(C)$ 
  - ↪ element in ranking uniquely identifies its position in the ranking
- ▶ bag type  $U\{\{C\}\}$  results in minimal key  $\{U'_{ID}\} \cup k_{attr}(C)$ 
  - ↪ element in bag identified by bag and whatever identifies element
- ▶ set type  $U\{C\}$  results in minimal key  $\{U'_{ID}\} \cup k_{attr}(C)$ 
  - ↪ element in set identified by set and whatever identifies element

## Example Translation - Minimal Keys for Collection Types

### ► recall ER Schema

- ↪  $\text{ACADEMIC} = (\{\text{Name, Phone, Office}\}, \{\text{Name}\})$ ,
- ↪  $\text{BOOK} = (\{\text{ISBN, Price, Title}\}, \{\text{ISBN}\})$ ,  $\text{COURSE} = (\{\text{No, Title}\}, \{\text{No}\})$ ,
- ↪  $\text{STAFF}\{\text{ACADEMIC}\}$ ,  $\text{READINGS}[\text{BOOK}]$
- ↪  $\text{TEACHING} = (\{\text{COURSE, Lecturers:STAFF, Tutors:STAFF, READINGS}\}, \{\text{Year}\}, \{\text{COURSE, Lecturers:STAFF, Year}\})$

### ► corresponding relational database schema

- ↪  $\text{STAFF}' = \{\text{ACADEMIC.Name, Staff}'_{ID}\}$  with minimal key  $\text{STAFF}'$
- ↪  $\text{READINGS}' = \{\text{BOOK.ISBN, Readings}'_{ID}, \text{Readings}'_{Pos}\}$  with minimal keys  
 $\{\text{Readings}'_{ID}, \text{Readings}'_{Pos}\}$  and  $\{\text{BOOK.ISBN, Readings}'_{ID}\}$
- ↪  $\text{TEACHING}' = \{\text{COURSE.No, Lecturers:Staff}'_{ID}, \text{Tutors:Staff}'_{ID}, \text{Readings}'_{ID}, \text{Year}\}$   
with minimal key  $\{\text{COURSE.No, Lecturers:Staff}'_{ID}, \text{Year}\}$

## Foreign Keys and Inclusion Dependencies

- ▶ For  $U(C)$  which foreign keys do we specify on relation schema  $U'$ ?
- ▶ if  $C$  not collection type: define  $[C.A_1, \dots, C.A_n] \subseteq C'[A_1, \dots, A_n]$
- ▶ if  $C = V(D)$  collection type (and thus  $k\_attr(C) = \{V'_{ID}\}$ ), then
  - ↳ same  $V'_{ID}$ -value may be associated with different  $D$ -objects
  - ↳ namely those belonging to collection identified by  $V'_{ID}$ -value
- ▶ do not specify inclusion dependency  $U'[V'_{ID}] \subseteq V'[V'_{ID}]$ 
  - ↳ empty  $V(D)$ -collection cannot be modelled in  $V'$ -relation
  - ↳  $V'_{ID}$ -value may represent empty  $V(D)$ -collection in  $U'$ -relation
  - ↳ this surrogate  $V'_{ID}$ -value cannot occur in  $V'$ -relation
- ▶ use null values *not exists* for attributes in  $k\_attr(C)$  instead?
  - ↳ none of the attributes in  $k\_attr(C)$  can be used as key attributes
  - ↳ leaving the safe boundary of complete information

## Example Translation - Foreign Keys for Collection Types

### ▶ recall ER Schema

- ↪  $\text{ACADEMIC} = (\{\text{Name, Phone, Office}\}, \{\text{Name}\})$ ,
- ↪  $\text{BOOK} = (\{\text{ISBN, Price, Title}\}, \{\text{ISBN}\})$ ,  $\text{COURSE} = (\{\text{No, Title}\}, \{\text{No}\})$ ,
- ↪  $\text{STAFF}\{\text{ACADEMIC}\}$ ,  $\text{READINGS}[\text{BOOK}]$
- ↪  $\text{TEACHING} = (\{\text{COURSE, Lecturers:STAFF, Tutors:STAFF, READINGS}\}, \{\text{Year}\}, \{\text{COURSE, Lecturers:STAFF, Year}\})$

### ▶ corresponding relational database schema

- ↪  $\text{STAFF}' = \{\text{ACADEMIC.Name, Staff}'_{ID}\}$  with minimal key  $\text{STAFF}'$  and foreign key  $[\text{ACADEMIC.Name}] \subseteq \text{ACADEMIC}'[\text{Name}]$
- ↪  $\text{READINGS}' = \{\text{BOOK.ISBN, Readings}'_{ID}, \text{Readings}'_{Pos}\}$  with minimal keys  $\{\text{Readings}'_{ID}, \text{Readings}'_{Pos}\}$  and  $\{\text{BOOK.ISBN, Readings}'_{ID}\}$  and foreign key  $[\text{BOOK.ISBN}] \subseteq \text{BOOK}'[\text{ISBN}]$
- ↪  $\text{TEACHING}' = \{\text{COURSE.No, Lecturers:Staff}'_{ID}, \text{Tutors:Staff}'_{ID}, \text{Readings}'_{ID}, \text{Year}\}$  with minimal key  $\{\text{COURSE.No, Lecturers:Staff}'_{ID}, \text{Year}\}$  and foreign key  $[\text{COURSE.No}] \subseteq \text{COURSE}'[\text{No}]$

## Referential Integrity and Extensionality

- ▶  $U'_{ID}$ -value uniquely identifies collections in relational database  
 $\hookrightarrow$  if two different  $dom(U'_{ID})$ -values occur in any relation, then there is  $C$ -object  $t$  separating the two collections **in  $U'$ -relation**
- ▶  $adom_{\mathcal{S}'}(U'_{ID})$  collects  $U'_{ID}$ -values of all relations in  $\mathcal{S}'$

$$\forall id_1, id_2 \in adom_{\mathcal{S}'}(U'_{ID}). (id_1 \neq id_2 \Rightarrow \exists t \in \prod_{A \in k\_attr(C)} dom(A). \\ ((id_1, t) \in U'[U_{ID}, k\_attr(C)] \wedge (id_2, t) \notin U'[U_{ID}, k\_attr(C)]) \vee \\ ((id_1, t) \notin U'[U_{ID}, k\_attr(C)] \wedge (id_2, t) \in U'[U_{ID}, k\_attr(C)])).$$

- ▶ each collection can be identified by its surrogate from  $dom(U'_{ID})$
- ▶  $adom_{\mathcal{S}'}(U'_{ID})$ -value either in  $U'$ -relation or denotes empty collection
- ▶ in university example we get on active domain of  $Staff'_{ID}$ :

$$\forall id_1, id_2 \in adom(Staff'_{ID}). (id_1 \neq id_2 \Rightarrow \exists t \in dom(ACADEMIC.Name). \\ ((id_1, t) \in STAFF' \wedge (id_2, t) \notin STAFF') \vee ((id_1, t) \notin STAFF' \wedge (id_2, t) \in STAFF')).$$

## Example Translation - Instance Level

READINGS'		
ISBN	Readings' <sub>ID</sub>	Readings' <sub>Pos</sub>
3540654704	1	1
0201565234	1	2
0130914290	2	1
0471007684	2	2

STAFF'	
Name	Staff' <sub>ID</sub>
Sven	2
Sebastian	2
Sebastian	1
Ernie	3
Bert	3

TEACHING'				
CourseNo	Year	Readings' <sub>ID</sub>	Lecturers:Staff' <sub>ID</sub>	Tutors:Staff' <sub>ID</sub>
157266	2007	1	2	3
157357	2007	2	1	0

- ▶  $adom(\text{Staff}'_{ID}) = \{0, 1, 2, 3\}$  with 0 denoting the empty set
- ▶ violated is  $\text{TEACHING}'[\text{Tutors:Staff}'_{ID}] \subseteq \text{STAFF}'[\text{Staff}'_{ID}]$

## Conclusion

- ▶ collection type constructors provide natural support for modelling
- ▶ direct support in ER model possible
- ▶ extends ER conceptual support to variety of data models
- ▶ transformation into relational dbs preserves features of collections
- ▶ study different mappings into different data models
- ▶ study query/constraint languages for ER model with collection types