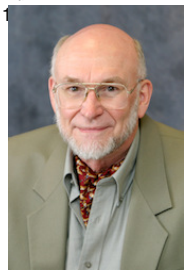
Overcoming the Limitations of ERel Modeling

DAMA- Indianapolis, 2013 October

GETITLE

1



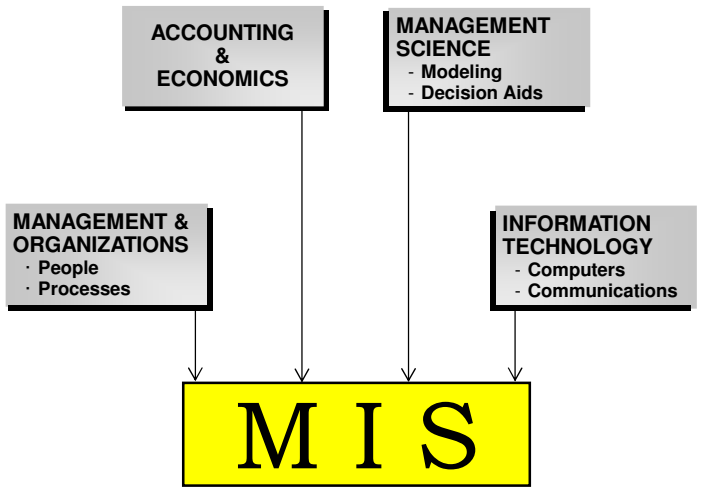
Overcoming the Limitations of ER/Relational Data Modeling

©Gordon C. Everest
Professor Emeritus of MIS and Database
Carlson School of Management
University of Minnesota
geverest@umn.edu www.tc.umn.edu/~geverest

My Background – Gordon C. Everest

GEINTRO

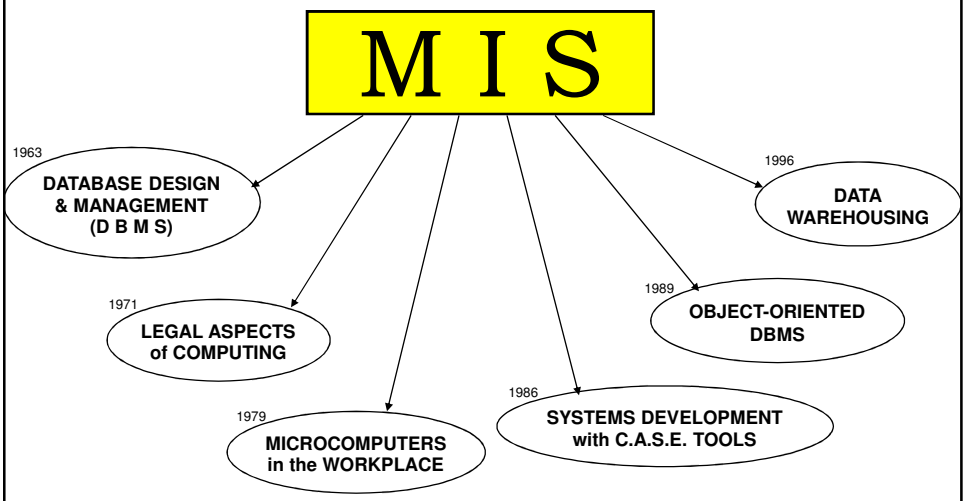
2



My Hot Buttons - Gordon C. Everest

GEINTRO

3



The Advanced Database Design Course

431INTRO

4

DESCRIPTION:

- Teaches a new way of thinking - ORM
  - ✓Avoids "table think" and all of its consequent problems
    The need for normalization; diagrams difficult to understand
  - ✓Learning ORM is the treatment for "Tableitis"
- Mix of students & working professionals
- Professionals often have more trouble unlearning what they know, have learned, and have practiced
- Prior knowledge and experience
  - – Used a DBMS to setup and query/manipulate tables
- Now offered online, piggy back on a face-to-face class
- This workshop arises out of my observations of problems stemming from faulty thinking – "*Table Think*"

Problems and Solutions

ORMvER

5

OBJECTIVES FOR THIS PRESENTATION:

- What is ER/Relational data modeling?
- Show several PROBLEMS with ER modeling schemes, (actually, *any* "record-based" modeling scheme).
- Identify the ROOT CAUSE of the problems

To stop there would be irresponsible, so…

- Show you a better way of thinking – a SOLUTION
  - Avoid "Table Think"; Defer building Tables.

- *What are we about in Data Modeling*
- *What are we trying to do?*

Logical Database Design
Objective, Principles, Benefits

B

DMOD

6

OBJECTIVE of LOGICAL DATABASE DESIGN: The *WHAT*

TO ACCURATELY AND COMPLETELY MODEL SELECTED PORTIONS OF THE REAL WORLD OF INTEREST TO A COMMUNITY OF USERS.

- USERS (COLLECTIVELY) WILL ALWAYS KNOW *MORE* ABOUT A DATA STRUCTURE THAN THE SYSTEM KNOWS, OR THAN COULD BE DEFINED TO THE SYSTEM.
- WHAT IS NOT FORMALLY DEFINED TO THE SYSTEM, THE SYSTEM CANNOT MANAGE . . . THE USERS MUST!
- THEREFORE, NEED TO CAPTURE RICH SEMANTICS WITH COMPREHENSIVE DATA MODELING and DEFINITION, INCLUDING INTEGRITY CONSTRAINTS AND OPERATIONS.

Let the 'system' do it! *Implications for a Tool!*

N

Purpose of Modeling – the *Why*

DMOD
7

To Facilitate *Human* Communication, Understanding, Validation

- Capture semantics – all relevant, important details
- Document – record and remember
- Understand – learn, raise questions, record answers, refine
- Communicate – shared with all interested parties
  - Users, stakeholders, management, developers
- Validate – a complete and accurate representation
  - Internal validation – consistent with the modeling rules
  - External validation – *Who can do this?*

SECONDARY:

- Blueprint to Build (a Database)

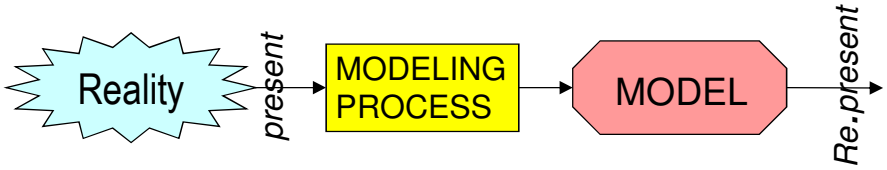
N

Modeling Process – the *HOW*

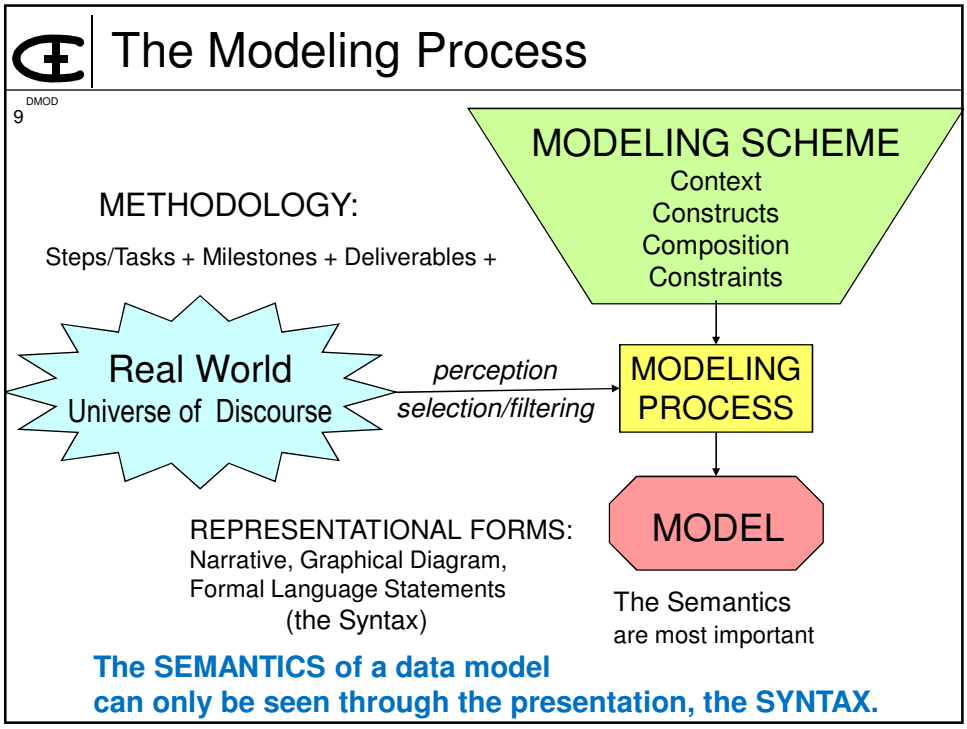
DMOD
8

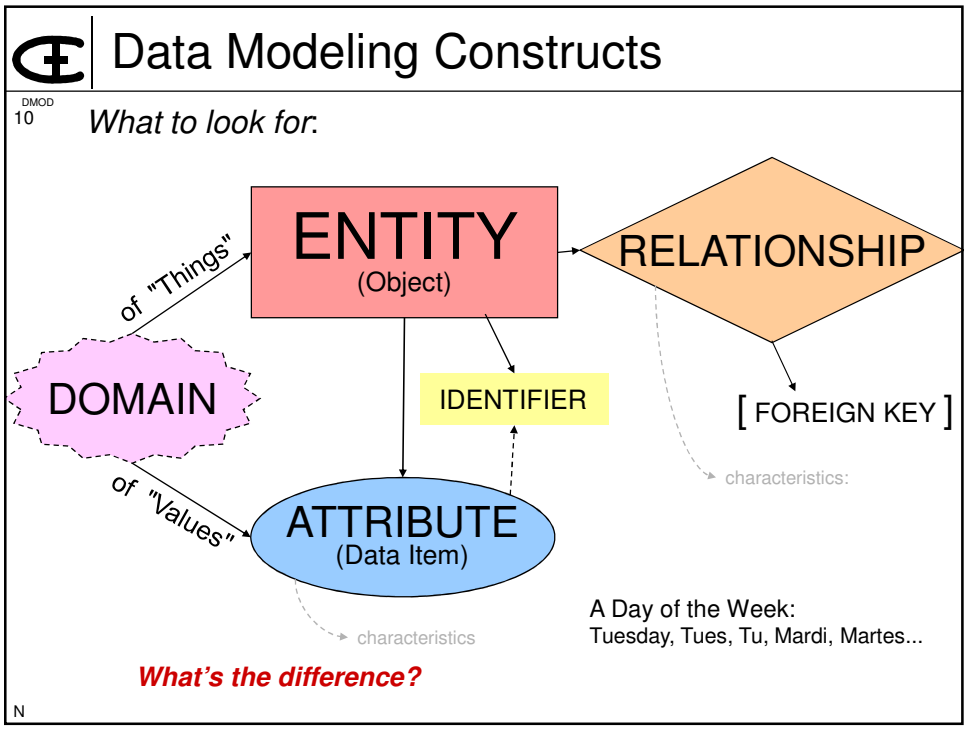
MODEL = Abstract (Re).present.(ation)

Knowledge in the world (infinitely complex) → Knowledge in the head (mental models) → Knowledge externalized, formalized, shared.



What drives or guides the process?





Data Modeling

B

431INTRO
11

- STARTS from some expression of the users world to be modeled… *in data*
  - Applications depend on a well-designed database
- TRADITIONAL APPROACH – think ER/Relational tables
- PROBLEM: some data items not in the right place
- SOLUTION: to find errors, apply the rules of normalization
- NORMALIZATION - the Achilles heel of data modeling
  - Even professional data modelers get it wrong
- REMEDY for violations => record decomposition

So wouldn't it be nice to have:

- Modeling scheme which avoids the need for normalization
  => ORM (Object Role Modeling, or Fact Oriented Modeling)
- Modeling tool for ORM => *NORMA* (also Visio in VS)
  - To automatically generate tables… in fifth normal form!

Record-Based Data Modeling

DMOD
12

- Commonly called Entity Relationship (ER) Modeling
- Attributes clustered into Entity Records (or Tables)
- Focus on Entities and Relationships (hence ER)
  suppressing attributes in ER Diagrams
  (hence no explicit representation of identifiers)
  leaving open the nature of the intra-record structure.
- Most general case *allows*:
  - "Nested" multivalued attributes or repeating groups
    Hence not in first normal form (1NF)
    (should still satisfy other normal forms – 2NF, 3NF, …)
  - Direct representation of M:N relationships between entities
  - Attributed relationships (i.e., with attributes)
  - Ternary relationships (and higher order)
- Restricting *all* of the above gives the Relational Model
  - Atomic (single-valued) attributes; binary (1:M) relationships (FKey)

=> Relational Table Diagrams are often called ERD*iagrams*

Record-based Design

G

ORMvER
13

WHAT SEMANTICS ARE PRESUMED
BY THE FOLLOWING RECORD STRUCTURE?

H1

X	A	B

ASSUME everything known about X, A, and B is shown.

1. What does it say about X ?
2. What does it say about A ?
3. What does it say about the relationship X-A ?
4. What does it say about the relationship A-B ?

Think in terms of:
Existence, Attributes, Roles, Relationships,
and relationship characteristics of Multiplicity, and Dependency.

State what is, NOT what is not, or what could be, or by construction, or processing.

N

Record-based Design

B

ORMvER
14

WHAT SEMANTICS ARE PRESUMED
BY THE FOLLOWING RECORD STRUCTURE?

X	A	B

Do we know if it is Normalized?
- to Third Normal Form (3NF)? How do we know?

Can a DBMS or data modeling tool help?

H

We must ASSUME it is Normalized,
Otherwise ambiguity in what [X | A | B] means.

Why Abstract Diagrams?

ORMvER
15

X	A	B

Rather than:

EMPLOYEE

Emp#	Name	...	SkillCode	Dept ID	…

- Examples can be dangerous, but…
- Examples can help confirm one's own understanding.
- If you can understand these concepts in the abstract you will be better able to do data modeling for real.

Record-based Design (1)

C

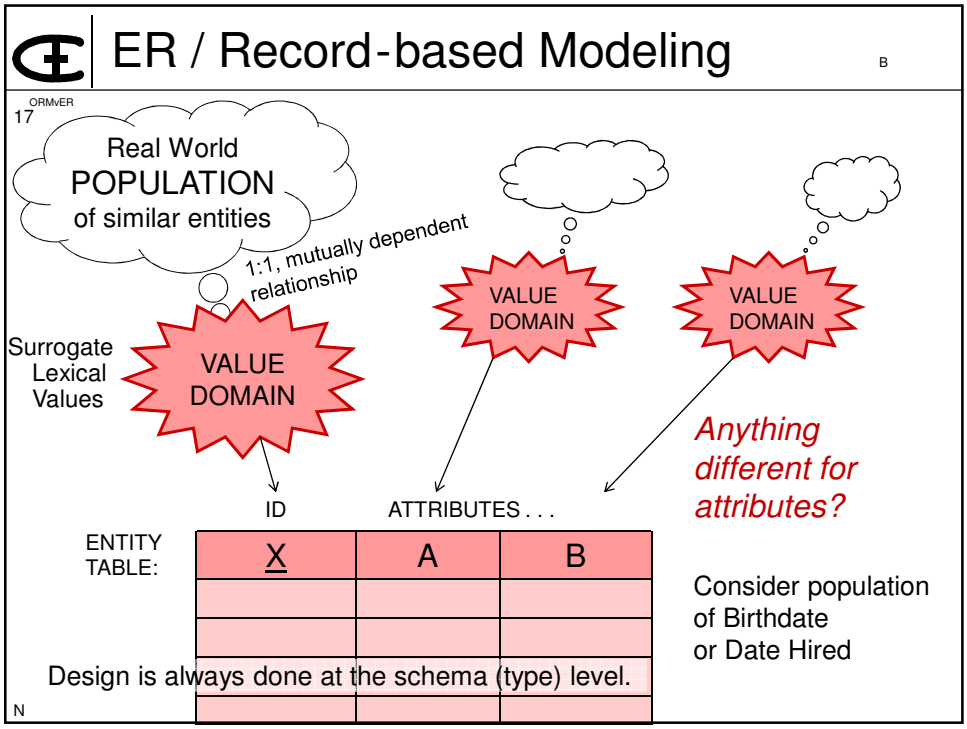
ORMvER
16

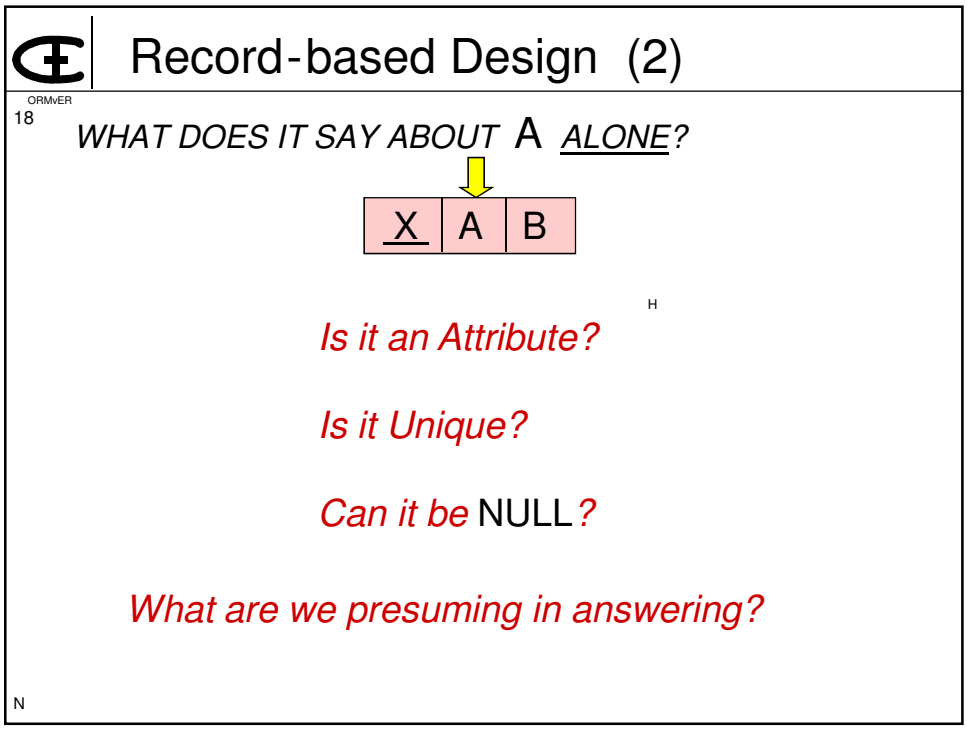
WHAT DOES IT SAY ABOUT X ?

X	A	B

Is **X** *an attribute?*

N





What is an Attribute?

ORMvER
19

An ATTRIBUTE

is an OBJECT

playing a ROLE

in a RELATIONSHIP

with some (other) OBJECT.

What comes first?

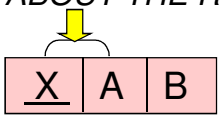
N

Record-based Design (3)

B

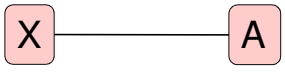
ORMvER
20

WHAT DOES IT SAY ABOUT THE RELATIONSHIP X-A *?*



H

What if I just drew:
What could you say?



How the table view frames our thinking!

N

Record-based Design

B

ORMvER

21

REPRESENTING THE RELATIONSHIP X-A *?*

X	A	B

What if A …

- *can be an orphan?* See H2.5
- *has additional attributes?* See H2.6
- *can be multi-valued?* See H2.7

To handle these situations
this diagram is wrong or insufficient
(as we will see in H2).

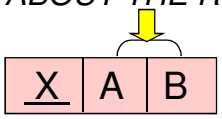
Record-based Design (4)

B

ORMvER

22

WHAT DOES IT SAY ABOUT THE RELATIONSHIP A-B *?*



- *Is there a relationship between* A *and* B *in* X *?*
- *Could there be a relationship between* A *and* B *?*
- *If so, what would it look like?*
- *Would the table of* X *still be normalized?*

H

- *Would there be any Foreign keys?*

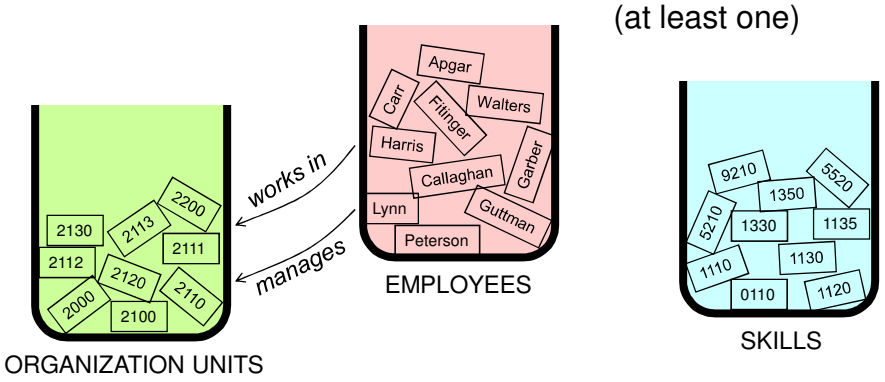
N

RELATIONSHIP

DMOD
23

between two Object Domains: (hence binary)

- Draw with or without replacement ----> EXCLUSIVITY (at most one)
- When done, has everyone been drawn at least once ----------> EXHAUSTIBILITY (at least one)

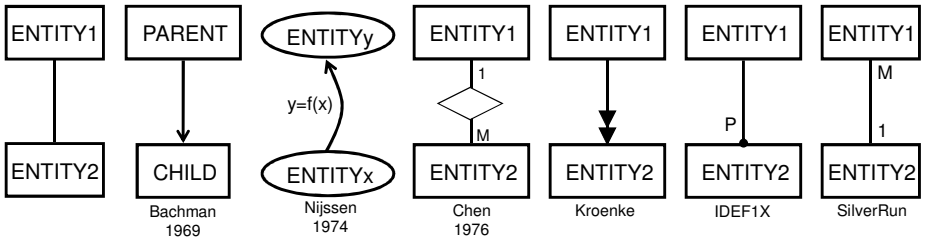


Choosing a Relationship Notation

DMOD
24

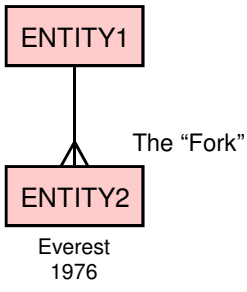
Everest-DM: p.224.

Candidate suggestions for Multiplicity ('one-to-many' or 1:M):



CRITERIA:

- NOT imply direction, access path, or physical representation
- Visually intuitive to aid human understanding



Also: Clive Finkelstein – Information Engineering (1979)

Relationship Characteristics

B1

DMOD 25

Each characteristic defined in *both directions*, on each entity:

- NAME
  - Singular VERB phrase

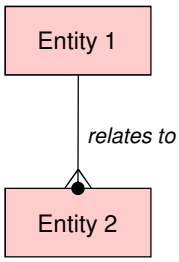
Everest-DM: p.223.

- MULTIPLICITY^{D} / EXCLUSIVITY
  - RELATE TO MULTIPLE or/ *AT MOST ONE*
  - 1:1 1:MANY MANY:MANY (M:N)

1 , M
0 , 1
("CARDINALITY")

- DEPENDENCY, MANDATORY, REQUIRED, EXHAUSTIVE / OPTIONAL, ORPHAN^{D}
  - MUST HAVE *AT LEAST ONE*, ELSE "ORPHANS' ALLOWED
  - Over time
    - DELAYED Dependency
    - FIXED to an Entity instance
- CRITERIA
  - BASIS IN MATCHING VALUES - "FOREIGN IDENTIFIERS"
  - ONE / MULTIPLE DATA ITEMS (composite)
  - Identifier or not => Multiplicity
- DEGREE
  - BINARY, TWO > TERNARY, OR MORE ENTITIES.

ALL REFLECTED IN A GRAPHIC DIAGRAM !



H1 - Typical Average Results

B

ORMvER 26

40% miss: There exists a population of things called A.

90% miss: There exists a relationship between X and A.

100% miss: X is a descriptor of A (even when asked!)

35% miss/wrong: dependency/optionality characteristic.

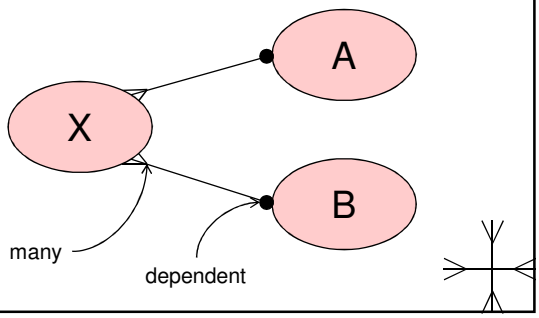
60% miss/wrong: multiplicity/exclusivity characteristic.

50% say or imply there is a relationship between A and B.

What might the results look like if presented with:

1. Object Domains
2. Relationships
3. Constraints

NO Tables, Identifiers or For. Keys



H2 – Data Model Diagrams

ORMvER
27

H2

Now that we know what [X | A | B] means,

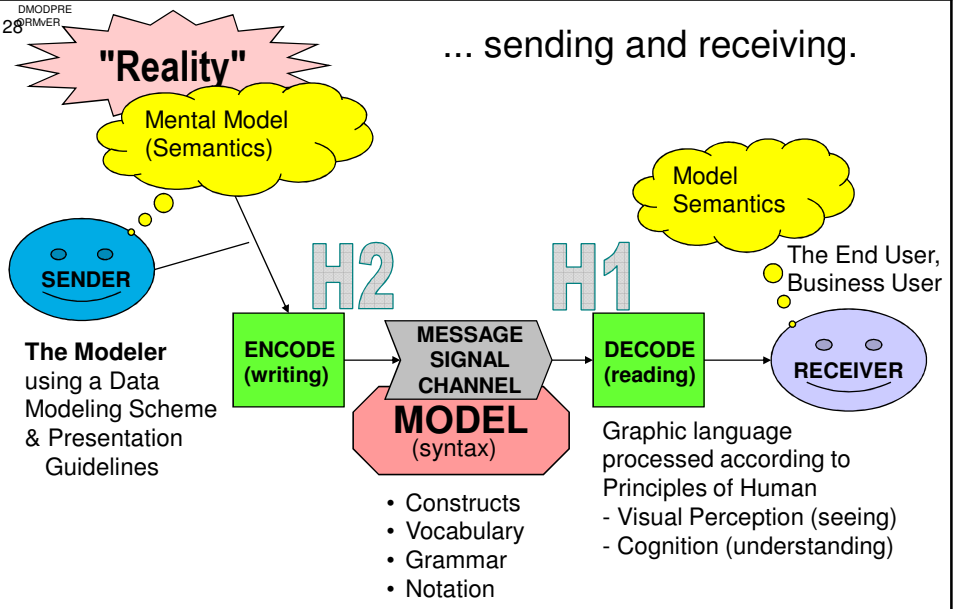
Encode Semantics to produce a Relational Table Diagram.

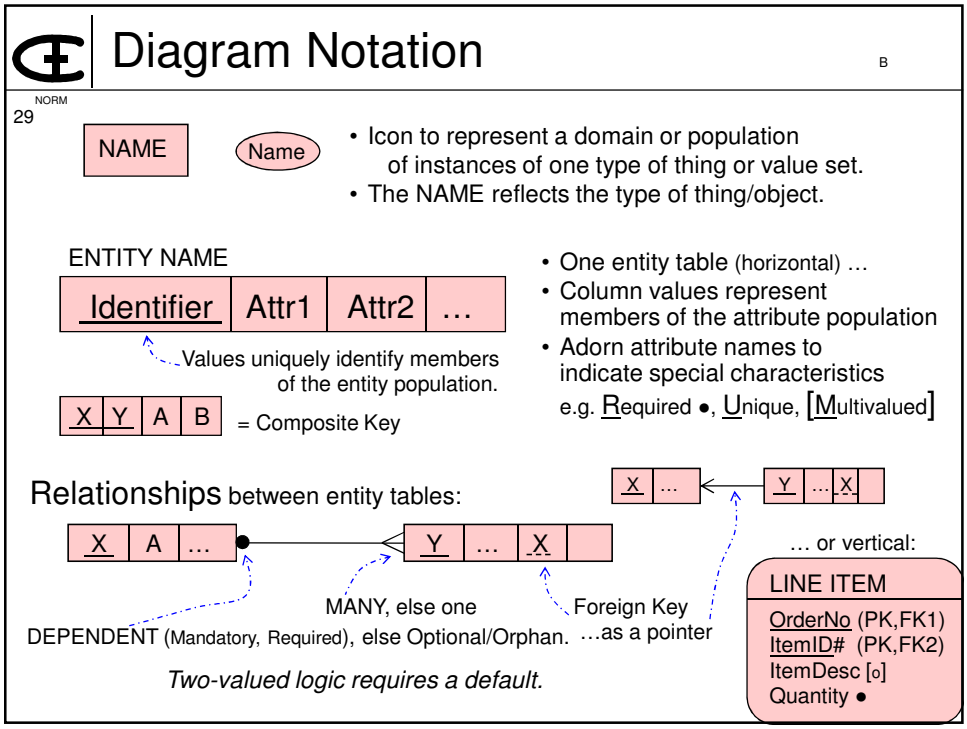
Variations on | X | A | B | commonly encountered.

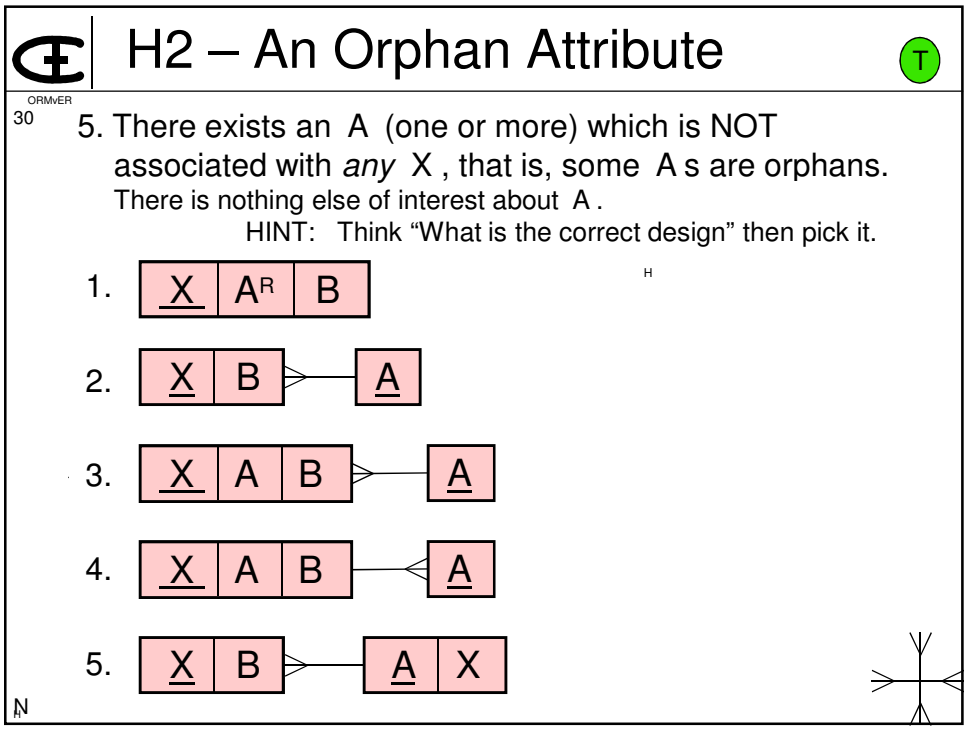
Communication

DMODPRE
ORMvER
28

... sending and receiving.







H2 – Attributes of an Attribute

ORMvER 31

6. Suppose we now have an additional data attribute D which is of interest to us about B .

Many got this, but missed 5.

Is B *still dependent on* X?

1. X | A | B — B | D_{esc}
2. X | A | B — B | D_{esc}
3. X | A | B | D
4. X | A | B^R — B | D_{esc}
5. X | A — B | D_{esc}

Special case: a 'Decode Table' for B

H2 – Multi-valued Attribute

ORMvER 32

7. X is NOT exclusive on A , that is, there may be multiple values of A for a given X .

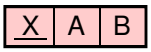
Many missed this.
First, understand the semantics

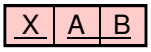
1. X | A^M | B
2. X | A | B — A
3. X | B — A
4. X | B — A
5. X | B — X | A
6. X | B — A | X
7. X | B — X | A
8. X | B — X | A — A

H2 – Attributes with M:N Relationship

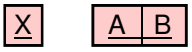
ORMvER
33

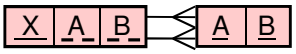
8. There exists a M:N relationship between A and B, in addition to the relationship each of them has with X.

1. 

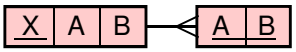
2. 

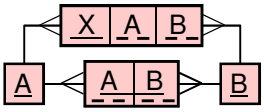
3. 

4. 

5. 

6. 

7. 

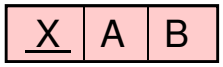
8. 

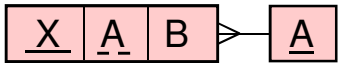
N

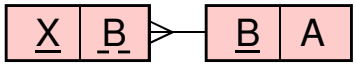
H2 – Attributes with 1:M Relationship

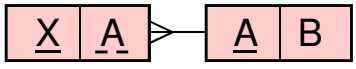
ORMvER
34

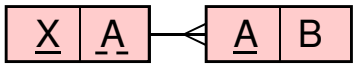
9. There is a many-to-one relationship between A and B since we observe that for every different or unique value of A, the values of B are all the *same* value.

1. 

2. 

3. 

4. 

5. 

N

H2 - Problems for the Students

ORMvER
35

- Much confusion with the Foreign Key
  - Inconsistent with the relationship arc
  - Must have an ID to point to
  - Can only represent at most a 1:Many relationship
- Putting an Attribute or Foreign Key in a table means the entity can have at most one of them.
- *Every* Relationship must be represented *somewhere* in the model with a pair of values (for a binary relationship)
- M:N Relationship means there must be a composite key *somewhere* in the model, and vv.
- Just because two attributes are together in a table does not mean there is a relationship.

The problem

ORMvER
36

"TABLE THINK"

… is the problem

Is the Relational Data Model for people or for the "machine/system"?

Why do we draw Data Model Diagrams?

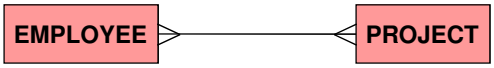
See description of "Tableitis" – a serious malady in our discipline.

Representing a M:N Relationship

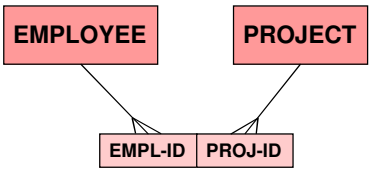
DMOD 37

WATSON5, Ch.5, p.115.

Another Pattern:

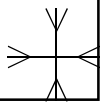


- If you cannot store multiple Projects (or Project IDs) in an Employee record, or multiple Employees (or Employee IDs) in a Project record (as is the case in a Relational Database in 1NF), then … you *must* introduce an "Intersection Entity" between them to represent the Many-to-Many Relationship.



What is the IDentifier?
What is the problem with this representation?
What is the underlying cause of the problem?

- The Intersection Entity also provides the place to store additional attributes of the relationship e.g., Hours Worked, Rate of Pay, …



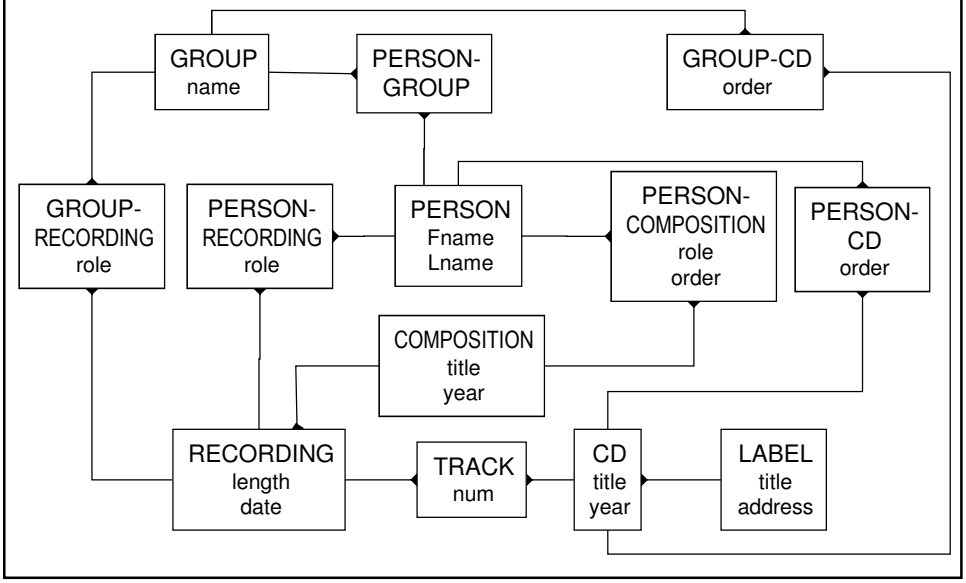
N

Resolving M:N Relationships

DMOD 38

WATSON5-6p153.

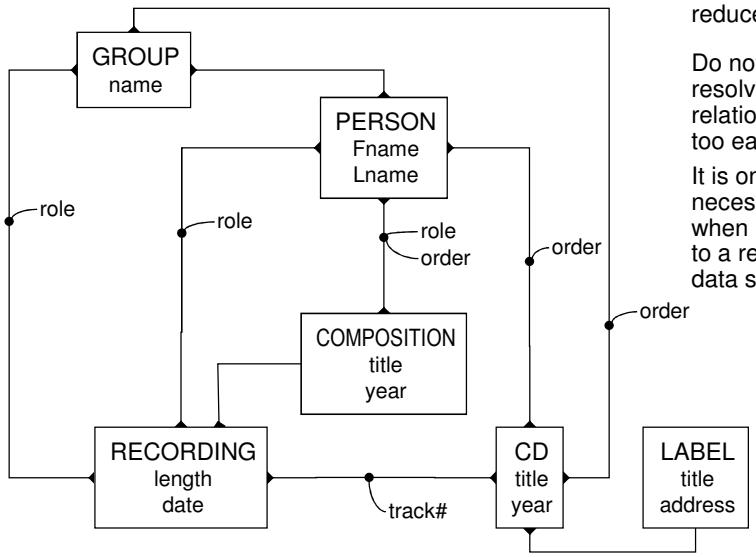
What is most important? (Omitting IDs and Foreign Keys)



Retaining M:N Relationships

DMOD 39

Which is more understandable to the users?



13 Entities reduced to 6!

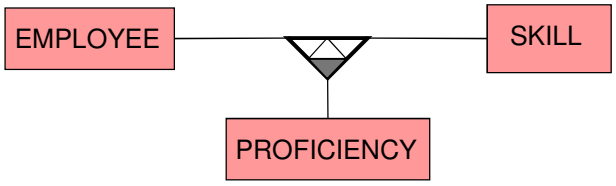
Do not resolve M:N relationships too early.

It is only necessary when mapping to a relational data structure.

Representing a Ternary Relationship

DMOD 40

While we can develop a consistent notation for binary relationships, ternary relationships are a problem. Sample EER (Teorey) notation:



How to represent and store the relationship and its characteristics?

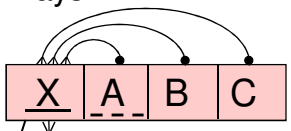
- *If one of the entities is single valued, is it really ternary? Or "attributed" binary?*
- *What lends uniqueness to each instance of the relationship?*
- *How to verbalize the relationship? Which order?*
- *How to represent Multiplicity / Exclusivity ?*
- *How to represent Dependency? Must have all 3?*

Representing Relationships

ORMvER
41

Multiple *different* ways:

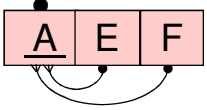
(1) Intra-record =>
Entity with Attributes



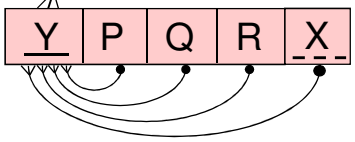
(3) Between/Among Attributes?

Spurious Associations?

(2) Inter-record =>
between Entities,
Entity with
another Entity



What if:
- *some* A's *can be orphans ?*
- *Is* A *still dependent on* X *?*
- A *has some other attributes?*



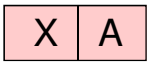
Where are the Foreign Keys?

N *How are they different?*

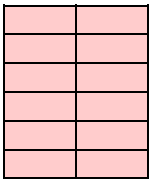
Representing a Relationship

ORMvER
42

- The schema design level:

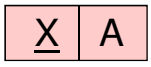


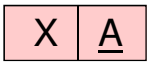
- The instance (data) level:

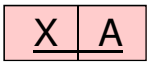


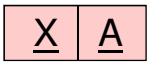
- What is the Identifier?
  Don't know until…?

All valid X-A pairs
(in the R/W)









Must know the multiplicity/exclusivity characteristics of X-A
before you can put X and A in a Table Diagram.
That requires the designation of an identifier;
a foreign key requires the prior designation of an identifier.

"TABLE THINK"

ORMvER
43

The process: focus on entity tables

- Find Entities
  - Top down from a macro view to find major entities, or
  - Bottom up from a list of data items (say from a DFD)
- Design a Table for each entity
  clustering or adding attributes with each entity
- Columns as attributes of the entity – Name, Type
- Designate/define the Identifier
- Represent Entity Relationships with Foreign Keys
- [Normalize – not required or enforced by the system.]

To jump in and think of putting data into tables means we must have already decided on:

- Entities, Attributes, Entity Identifiers, Foreign Keys (relationships), and relationship characteristics

Attributes in a Table

ORMvER
44

- When putting Attributes in a Table, we generally don't ask:
  - What is the object population it represents?
  - Could there be members of that population that would not be related to any member of this entity population but for which we would still be interested in recording?
  - Any other entity have/share the same attribute?
  - Anything else of interest about this entity?
  - Is it really an entity in its own right?
    if so, does it need its own table?

We make assumptions implicitly about these questions when we go directly to thinking/designing tables.

Attribute Characteristics

ORMvER
45

An attribute of what (entity or object)?

Nothing is an Attribute by itself.

X	A

- All about the relationship with some entity
  - Optional or Required, conditional existence, derived
  - Single valued (or multi-valued which must be resolved)
  - Foreign key (relationship of this entity with another entity)

A

- All about identifying and describing instances

  Lexical surrogate for members of the object population
  - Name(s), description
  - Data type, format
  - Size, precision
  - Value set (domain of values) - range/s, list, encoded

Attribute Migration

ORMvER
46

Given:

Attribute... or Entity... or what?

EMPLOYEE

Emp#	Name	...	SkillCode

What is SkillCode?

Now add a Description, etc. for SkillCode:

Now what is SkillCode?

What is the Relationship?

What is SkillCode in Employee table?

Suppose an Employee has multiple Skills:

Now what is the Entity?

(In SQL, every table must have a name)

What are the Relationships?

Is SkillCode still an attribute of Employee?

H

Functional Dependency in Relationships

NORM 47

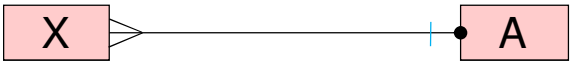
Basis for Database/Table *Normalization*.

A ← f (X)

◄ is functionally dependent on

determines ►

or X → A



A is *dependent* on X, and the Relationship is *exclusive* on A, *multiple* on X.

Clustered into a record/table for entity of X:

X	A	…

RULE: *Store A with its Determinant as the Key.*

There can only be *one* A for each X.
There can be multiple Xs for a given A.

Encompasses all rules of Normalization!

Normalization – Testing your Understanding

NORM 48

Assuming that A is single valued with respect to X (i.e. 1NF).
(enforced by construction)

GIVEN:

Could you have a violation of: (if not, why not?)

X	A

2NF? 3NF? 4NF or 5NF?

X	A	B

X	A	B

Clustering attributes into tables
the modeler may make a mistake.
Normalization is the test,
Record decomposition is the remedy.

If you don't cluster attributes into tables,
you cannot violate the rules of normalization.
Normalization becomes unnecessary and irrelevant.

H

"TABLE THINK" – Problems-1

ORMvER
49

- Anchors you on the entity of the identifier, which frames your thinking in one direction.
- Inter-entity relationships represented redundantly with a foreign key and an arc; users see it twice in the model.
- Difficult to represent the entire population of an object, whether entity or attribute domain.
- Enforcing single-valued (atomic) attributes (i.e., 1NF) makes it impossible to directly represent M:N relationships. It requires an intersection "entity."
- When speaking of attribute characteristics, most of the time it is characteristics of the relationship with the entity. (required or optional, single- or multi-valued, unique)
- Putting an attribute in an entity table presumes a particular relationship with the entity without being explicitly defined, consequently we can do it wrong, requiring normalization.

At the Root,

ORMvER
50

What's wrong with

ERel Modeling?

N

Problems with ERel Modeling - Summary

ORMvER
51

All a consequence of clustering !

- Cannot capture the "conceptual" view directly,
  Must *mentally* map to the "logical" (record-based) view
  by clustering Attributes into Entity records/tables.
  - Modeler must *a priori* choose whether Entity or Attribute
  - Too much clustering; attributes in the wrong place
  - Ignores (or presumes normalized) intra-record structure
    (that is, relationships between/among Attributes)
    - creates (implies) spurious inter-attribute relationships
- Human modeler is responsible for normalization
  remedy is *always* record decomposition
- Must choose unique names
  - for attributes in a record; for spurious new "entities"
  - column names = domain + role; lose object domains
- Modeling / Processing dilemma:
  - Complete representation of an entity object - more clustering
  - Full normalization (1NF) – decomposition, more fragmentation
- Indirect representation of M:N relationships with intersection “entity”
- Difficulty representing Ternary relationships
- Stability of the query language (SQL)

All are solved in ORM !

“TABLE THINK” – My Epiphany

ORMvER
52

- When we design a relational database we are actually modeling all and only the relationships among things in the users’ world
  - Relationships both within and between entity tables
  - Clustering (“throwing in”) attributes into tables without explicitly defining the intra-entity relationships.
  - Designers make mistakes – thinking of tables first
- We can only speak of attributes in the context of a relationship: _ _ _ _ is an attribute of _ _ _ _?
- We are not modeling *all* object domains first.

Defining an Object Population

ORMvER
53

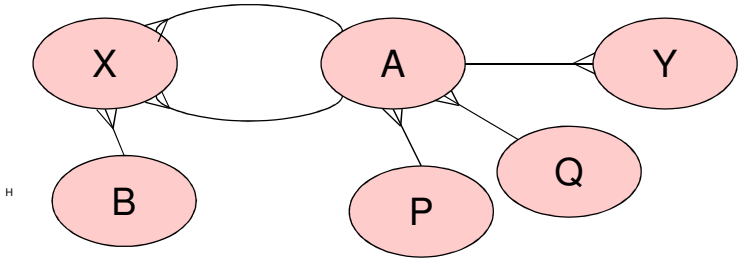
- What is the population of A?
- How can you get the population of A?

1. X | A — *Any orphan A's?*
2. X | A Y | A — *Is A still dependent on X?*
3. X | A Y | A A | P | Q — *Any FKeys?*
4. X | A | B | A_2 Y | A A | P | Q

How many times do you define a population of A's? Redundant?

A Better Data Model Diagram

ORMvER
54



H

1. Objects (domains) shown only once each
   Not distinguish Entities and Attributes
2. All Relationships shown, and the same way (an arc).
3. Add multiplicity constraints.
4. Where are the tables? Do we care? Who does care?

Which is easier to design? To understand?

X | A | B | A_2 Y | A A | P | Q

N

Domains

DMOD
55

… of objects, whether entity or attribute

DOMAIN
of R/W objects
(Person)

Cannot be stored in a database

1:1 mutually dependent

DOMAIN
of "values" to
represent objects
(Surrogate 'names')

Lexical representation

DATA TYPE: Alpha
VALUES: Mary
Mary Everest

stored in a database

EmpID	Name	B'Date	…

Record-based Design

B

ORMvER
56

WHAT DOES THIS "RECORD" REPRESENT?

ENTITY

X	A

X	B

X	C

Design minimal "records"
with *at most one* non-key domain.
Remedy for Normal form violations is Decomposition.
This is the ultimate end of Record Decomposition.

Now what do these "records" represent?
Perhaps Codd was right in naming it a _____!

Avoids spurious associations, e.g., A-B …
Could there be any violations of normal forms?

What about representing the entity X ? or any domain?

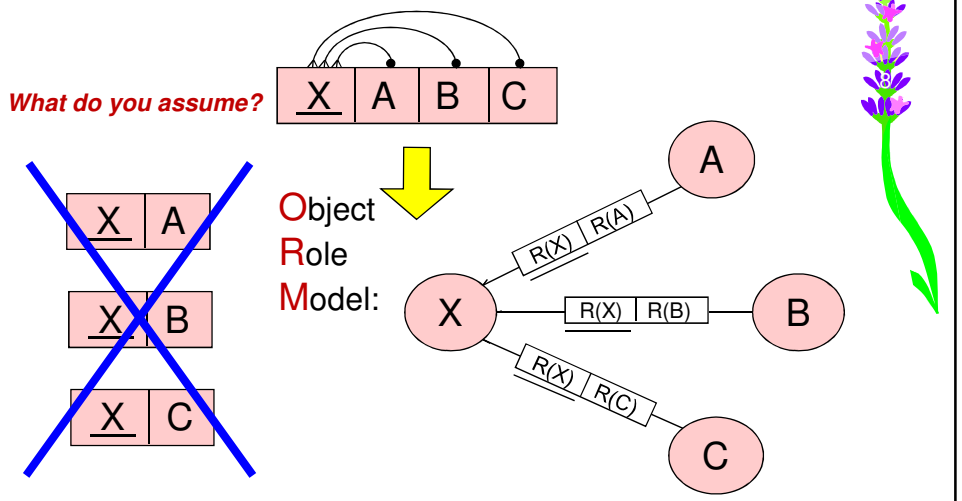
What if A is related to (or attribute of) other "entities"?

Transform Record-based (ER) Design

ORMvER

57

TO REALLY REPRESENT THE ENTITY DOMAINS



OBJECTS (ENTITIES) *have* "ATTRIBUTES" *(DESCRIPTORS) by playing* ROLES *in* RELATIONSHIPS *with other* OBJECTS.

N

Record-Based Modeling Ex.1

ORMvER

58

GIVEN TWO FACTS (conceptually):

- one about the CITY a PERSON *lives in*
- another about the CITY a PERSON *works in*

ASSUME:

- every person has to live and work in a city
- each person can live and work in only one city (at a time)
- not interested in anything more about persons or cities

EXAMPLE (two elementary fact instances):

Gordon Everest lives in Roseville and * works in Minneapolis

DIAGRAM A CONCEPTUAL DATA MODEL

– to represent this information (a database to contain these facts)

Record-Based Data Model Ex.2

ORMvER
59

For: PERSON *lives in / works in* CITY

- What is the entity and what is the attribute?
- Would it make any sense to say (to a novice layperson - a user):
  - CITY was an "attribute" of PERSON not an entity (no table)?
- Doing more than is necessary at the conceptual level

PERSON

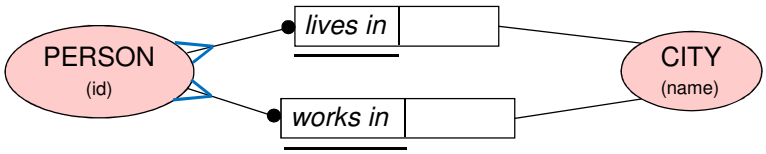
PersonID [key]
LiveCity
WorkCity

- cannot have CITY and CITY as attributes of PERSON
- column/attribute name reflects " entity + role "
- CITY as an entity/object is lost (not its own table)
- what if there is a CITY where no one lives or works
- some add concept of a DOMAIN in SQL
  (but in a Relational database it is not in our table diagram!)

Object-Role Model Ex.3

ORMvER
60

for: PERSON *lives in / works in* CITY



FORML language statements (verbalization):

- PERSON *lives in* CITY
- Every PERSON *lives in* some CITY
- Each PERSON *lives in* at most one CITY

... Similarly for *works in*

FACT

● Required / Mandatory

Unique (Identifier)

Now add: PERSON *makes sales calls* in multiple CITIES

G

Verbalize User Descriptions

noun
verb
constraint

ORMODLG
61

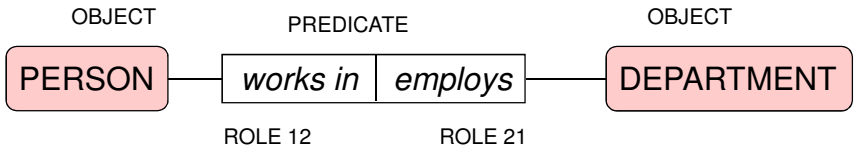
GIVEN A DESCRIPTION FROM THE USER(S):

Famous Foods, a small, specialty food wholesaler, fills orders for restaurants. Customers have names, addresses, etc. An order can include several products. Products have unique SKU numbers, descriptions, manufacturer, etc. The company has one big warehouse with many rooms on several floors. Each product is stored in only one bin location in the warehouse, but it can change frequently. Multiple products may be stored in the same bin. Bin numbers are only unique within a room, hence the same number can be used in different rooms. Since the bin locations can be hard to find in a room (could be on a shelf, on the floor, in a cabinet or cooler, hanging from the ceiling, etc.), and the rooms can be hard to find in the warehouse (with many hallways, doors, tunnels, split levels, mezzanines, etc.), explicit location directions must be recorded for each room and for each bin in the room. Location information is a textual narrative and is used by the pickers who run around gathering the items to fill an order. Each product has its own standard price but it may be modified by applying a discount (a fraction) on any individual order. The discount can be different for each of the products on an order, and for the same product on different orders. The quantity of each product on an order is recorded (it is not the quantity on hand or in inventory). Terms indicates the number of days during which a standard discount can be taken on the payment. The terms can vary from one customer to the next, and from one order to the next for the same customer.

Symbolize: ORM Constructs

ORMODLG
62

Elementary Binary Fact Sentence (schema):

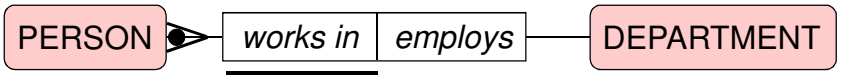


Verbalization:

"PERSON works in DEPARTMENT"
"DEPARTMENT employs PERSON"

Adding ORM Constraints

ORMCONSTR
63

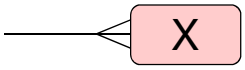


- Verbalization: "PERSON *works in* DEPARTMENT"
  "DEPARTMENT *employs* PERSON"

- DEPENDENCY (REQUIRED or MANDATORY):
  "PERSON *must work[s] in some* DEPARTMENT"

EXCLUSIVITY (UNIQUENESS):
"PERSON *works in at most one* DEPARTMENT"

Prefer the "fork" for Multiplicity.



Overcoming the Limitations of ERel

ORMvER
64

… requires a different way of thinking

1. First, think and model object populations
   - Name with a singular noun reflecting R/W object
   - Describe the population
   - Show only once in the model diagram
   - [Optional: Designate a lexical surrogate, i.e., identifier]
   - Note: *all* object populations are mutually exclusive
2. Then find *all* relevant relationships
   - Define them explicitly and in the same way
   - Name the roles objects play in the relationships
3. Define integrity constraints or "business rules"
4. Present it to the user domain experts
   for human understanding and validation

Only then are you ready to put into tables

"New" Modeling Approach - Example

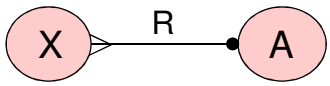
ORMvER 65

- First **Object populations** (nouns)
  - NOTE: all object populations are *mutually exclusive*!
  - Each has its own "table" with one column – the identifier

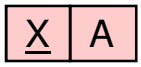
- Then **Relationships** (verbs) and **Roles** w.r.t. each other
  - Not presuming the nature of the relationship

What is the identifier?

- Then add **Constraints**
  - Dependency/optionality, and multiplicity/exclusivity



Now we can build tables:

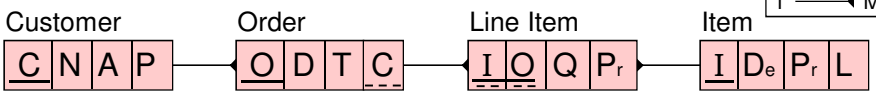


NOTE: This is the relationship!

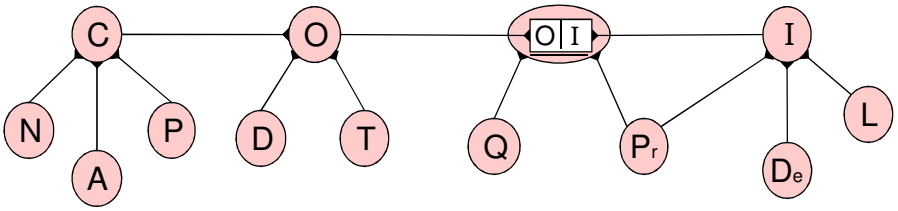
Transforming ERel to ORM

ORMvER 66

LEGEND:
Primary Key
Foreign Key
1 —◂ M



ADD Foreign Keys (redundant). NOTE: Duplicate fields - C P_r I O .
Break out all Entities and Attributes into separate Objects:
ADD Relationships (presuming above to be fully normalized):



NOTE: All Object Domains are shown only once!

NOTE: All Functional Dependencies are now explicitly shown!

Data Modeling in ORM

ORMvER
67

Try it yourself, starting with =>

Modify / extend the diagram with these semantics:

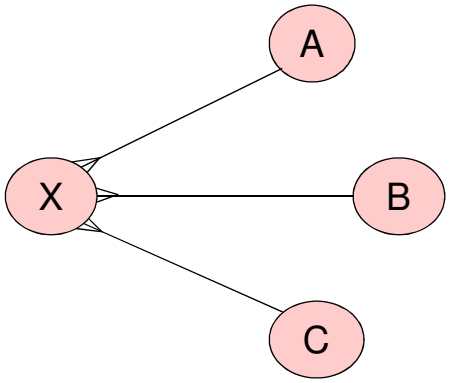
For each X:

1. A is REQUIRED (●)
2. A is UNIQUE for all X
3. A is MULTIVALUED(─<)
4. A is INDEPENDENT (!)
   i.e., can be ORPHAN

Also:

5. B is FUNCTIONALLY DEPENDENT on A
6. B and C are RELATED
7. D is an ATTRIBUTE of A

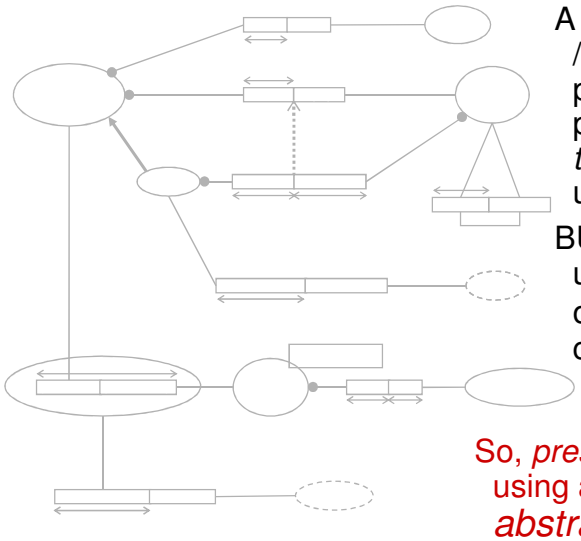
N



NOTE:
- No "Attributes"
- No 'TABLE THINK'
- No Foreign Keys
- No Normalization
- Focus on Object Domains
- Think *all* Relationships

ORM Data Model - Presentation

DMODPRE
68



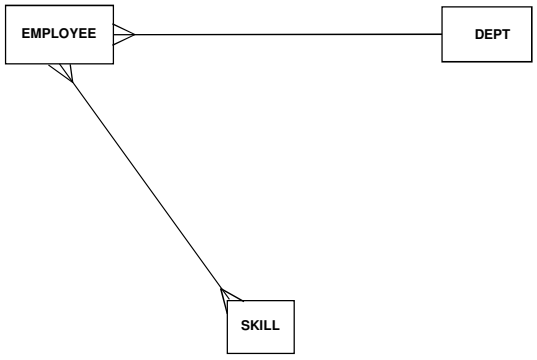
A major criticism of NIAM / ORM, both by protagonists and proponents, is that it is *too detailed*, a bottom-up design,

BUT… ER Diagrams usually *hide* the details of attributes and most constraints.

So, *present* the ORM model using a series of top-down *abstractions*.

Abstractions of ORM Data Model

DMODPRE 60



1. Hide "Terminal" (M:1) Objects (=> Attributes)
2. Hide Reference Modes
3. Hide Constraints
4. Hide Less Important Objects & Predicates
   - Subtypes
   - Objectified Predicates
   - Reflexive Relationships
5. Hide all Predicates

   Leaving BASE Entities!
6. Add back Multiplicity char. on relationships

Is this the same data model we started with?

=> A High-level Abstract "Conceptual" Data Model...

an ER Diagram ?!!!

Stages of Data Modeling

DMOD 70

Start at the highest Conceptual Level!