Fundamentals/ICY: Databases 2013/14

WEEK 7 –***Friday***

John Barnden
Professor of Artificial Intelligence

School of Computer Science
University of Birmingham, UK

Reminder of Monday

(necessarily **non**-symmetric) **1:M** recursive: "EMPLOYEE Manages EMPLOYEE"

FIGURE 4.23 IMPLEMENTATION OF THE 1:M "EMPLOYEE MANAGES EMPLOYEE" RECURSIVE RELATIONSHIP

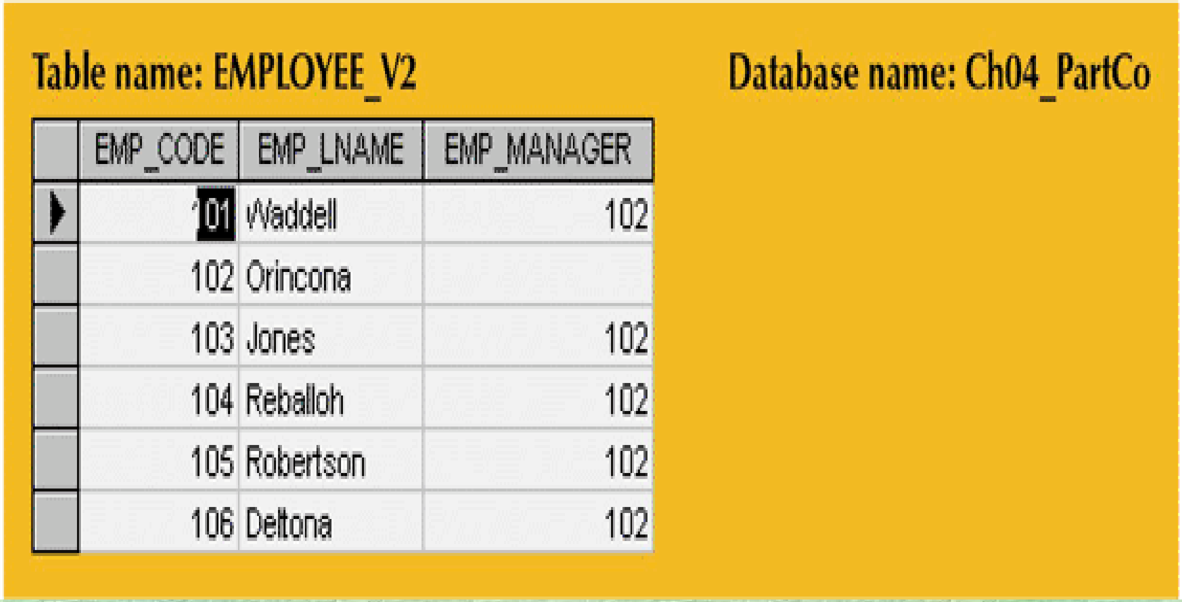

Table name: EMPLOYEE_V2

Database name: Ch04_PartCo

EMP_CODE	EMP_LNAME	EMP_MANAGER
101	Waddell	102
102	Orincona	
103	Jones	102
104	Reballoh	102
105	Robertson	102
106	Deltona	102

Just a standard 1:M implementation except linking a table to itself.

<u>No redundancy problem</u>.

non-symmetric **M:N** recursive: “PART Contains PART”

FIGURE 4.21 IMPLEMENTATION OF THE M:N RECURSIVE “PART CONTAINS PART” RELATIONSHIP

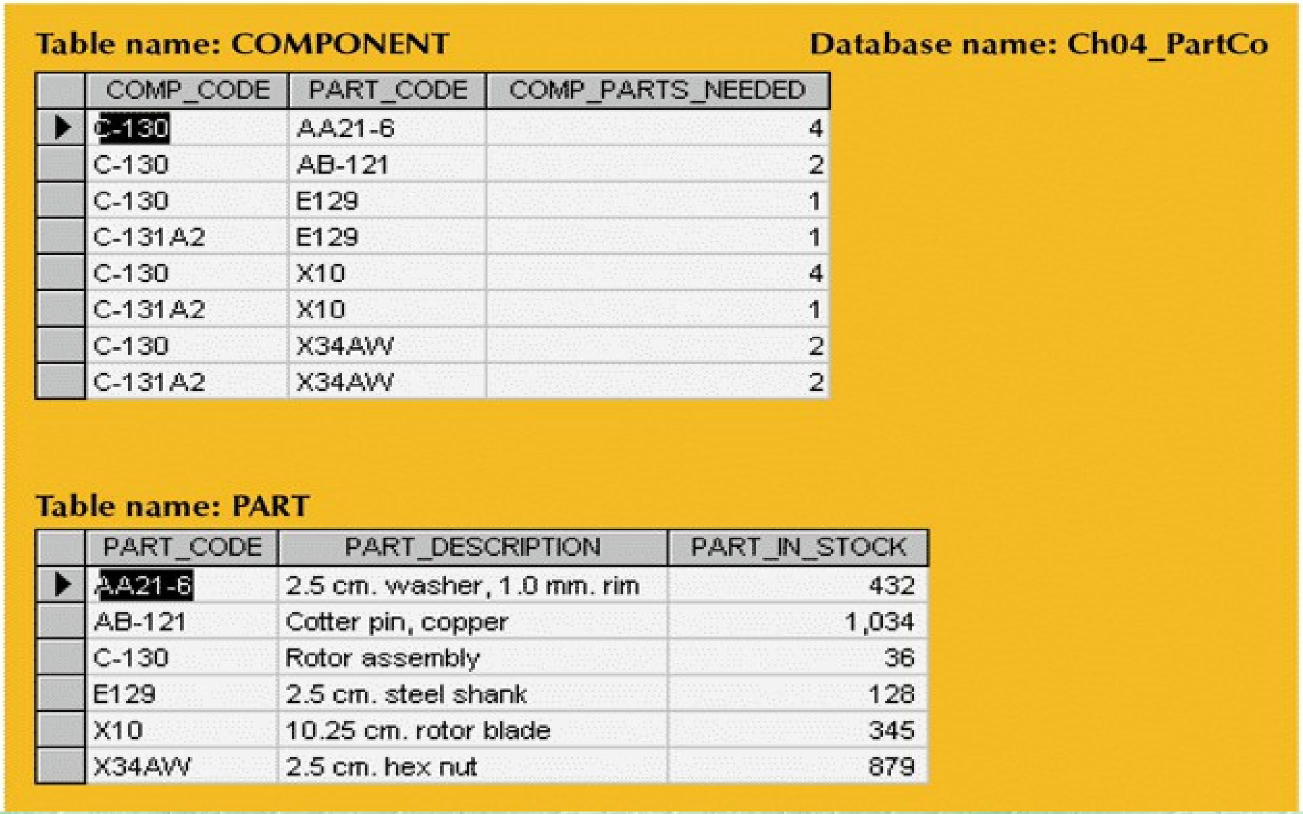

Table name: COMPONENT

Database name: Ch04_PartCo

COMP_CODE	PART_CODE	COMP_PARTS_NEEDED
C-130	AA21-6	4
C-130	AB-121	2
C-130	E129	1
C-131A2	E129	1
C-130	X10	4
C-131A2	X10	1
C-130	X34AW	2
C-131A2	X34AW	2

Table name: PART

PART_CODE	PART_DESCRIPTION	PART_IN_STOCK
AA21-6	2.5 cm. washer, 1.0 mm. rim	432
AB-121	Cotter pin, copper	1,034
C-130	Rotor assembly	36
E129	2.5 cm. steel shank	128
X10	10.25 cm. rotor blade	345
X34AW	2.5 cm. hex nut	879

The COMPONENT entity type is just a bridging type, linking PART to itself. NB: its first two columns both refer to PART’s PK but must be differently named. No redundancy problem.

symmetric (1:1) recursive relationship: "EMPLOYEE Married to EMPLOYEE"

Suppose you tried the following:

FIGURE 4.19 THE 1:1 RECURSIVE RELATIONSHIP "EMPLOYEE IS MARRIED TO EMPLOYEE"

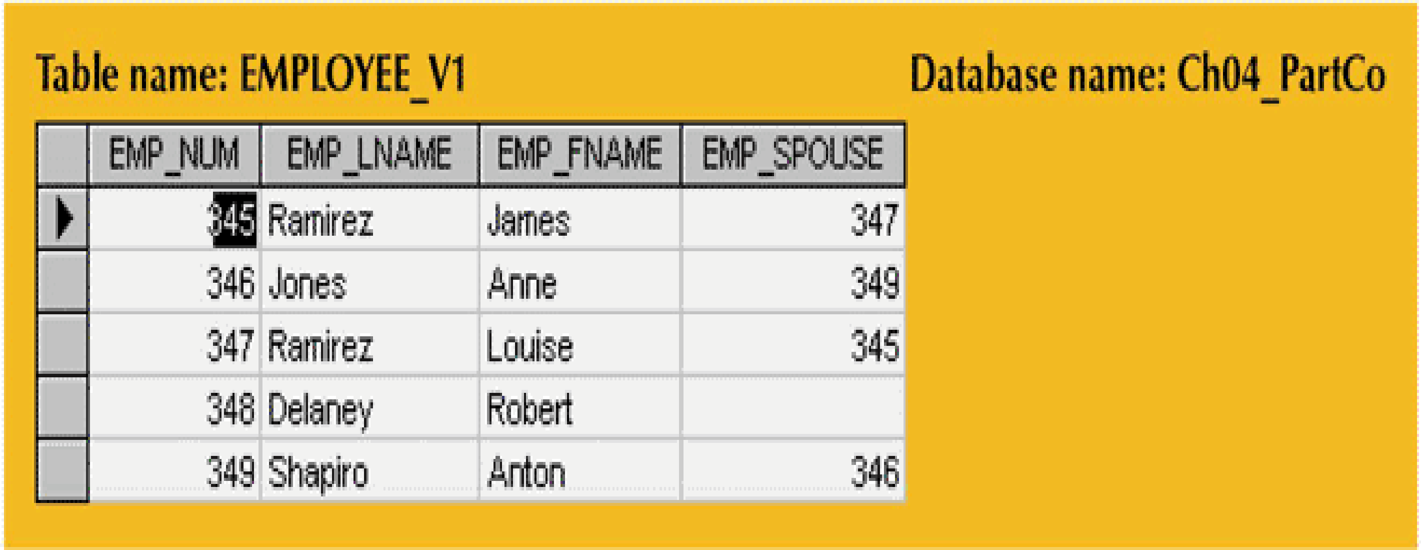

Table name: EMPLOYEE_V1

Database name: Ch04_PartCo

EMP_NUM	EMP_LNAME	EMP_FNAME	EMP_SPOUSE
345	Ramirez	James	347
346	Jones	Anne	349
347	Ramirez	Louise	345
348	Delaney	Robert	
349	Shapiro	Anton	346

Redundancy problem!!

New

Symmetry is the Problem

- ***A non-symmetric 1-1 relationship would not have the problem shown on previous slide.***

- ***A symmetric M:N relationship would have a redundancy problem, whether implemented as in the 1-1 case or by a bridging table.***
  - ***E.g.: being-sibling-of.***

symmetric (1:1) recursive relationship: redundant & non-redundant implemntns

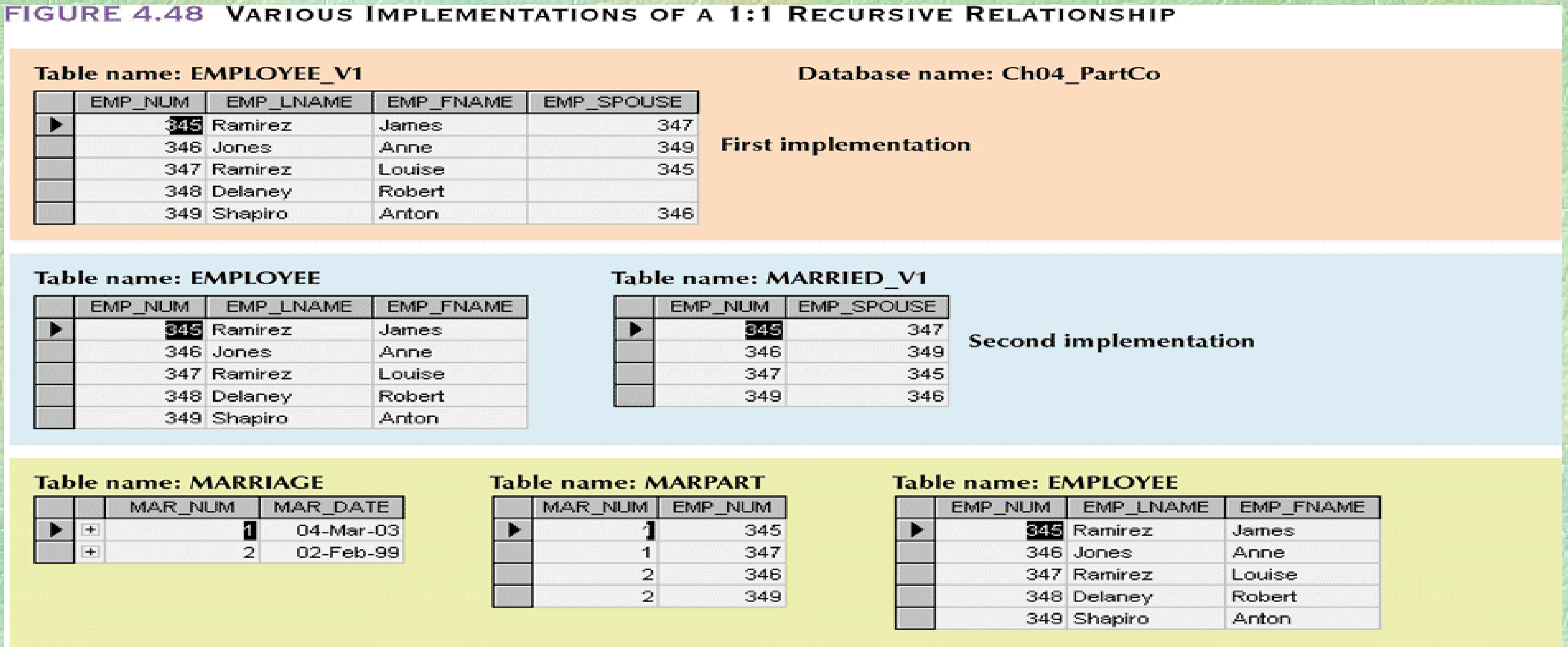

FIGURE 4.48 Various Implementations of a 1:1 Recursive Relationship

Table name: EMPLOYEE_V1

Database name: Ch04_PartCo

	EMP_NUM	EMP_LNAME	EMP_FNAME	EMP_SPOUSE
▶	345	Ramirez	James	347
	346	Jones	Anne	349
	347	Ramirez	Louise	345
	348	Delaney	Robert	
	349	Shapiro	Anton	346

First implementation

Table name: EMPLOYEE

	EMP_NUM	EMP_LNAME	EMP_FNAME
▶	345	Ramirez	James
	346	Jones	Anne
	347	Ramirez	Louise
	348	Delaney	Robert
	349	Shapiro	Anton

Table name: MARRIED_V1

	EMP_NUM	EMP_SPOUSE
▶	345	347
	346	349
	347	345
	349	346

Second implementation

Table name: MARRIAGE

		MAR_NUM	MAR_DATE
▶	+	1	04-Mar-03
	+	2	02-Feb-99

Table name: MARPART

	MAR_NUM	EMP_NUM
▶	1	345
	1	347
	2	346
	2	349

Table name: EMPLOYEE

	EMP_NUM	EMP_LNAME	EMP_FNAME
▶	345	Ramirez	James
	346	Jones	Anne
	347	Ramirez	Louise
	348	Delaney	Robert
	349	Shapiro	Anton

1) *As previously—redundant .*

2) MARRIED_V1 *is just a bridging entity type: still redundant.*

3) MARRIAGE *together with* MARPART *act as a sort of bridge. Non-redundant.*

Symmetric M:N, etc.

- Method 3 on previous slide can straightforwardly be generalized to:
  - symmetric recursive ***M:N*** relationships
  - (((partially-)symmetric&recursive, etc. relationships that link more entities together, whatever the connectivity --- M:N:P, 1:1:1, M:1:P, …))

Summary: Creating ERMs/ERDs

- Designing an ER model for a database is an ***iterative process***, because, e.g.:
  - As you proceed, you think of **new ways of conceiving** what's going on (much as in ordinary programming)
  - **Multivalued attributes** need to be re-represented eventually
  - **Supertype/subtype** notation needs eventually to be converted into more standard diagram notation
  - **M:N relationships** can be included as such at an early stage, but usually need to be replaced by means of bridging entity types later
  - **1:1 relationships** raise a **red flag**: may indicate poor design, especially when mandatory both ways.
    - ***But standard in supertype/subtype representation, and natural in recursive relationships like "married-to"***.
  - **Symmetric** recursive relationships usually need **special handling.**
  - Conversion of ERM portions to a **"Normal Form" – LATER.**

Next major databases topic

is

Normalization

but for now …

MATHEMATICAL BACKGROUND TO TABLES

Mathematical “sets”: Basics

- A “**set**” is an **unordered** collection of items of any sorts (people, numbers, numerals, shoes, atoms, strings of characters, databases, sets, blades of grass, …) **without any duplication of items**.

  The items are called “elements” or “members”.

- **S = {34, JAB, 59, UoB},** where “JAB” is a name for me and “UoB” is a name for this university,

  means that

  S is the set consisting of (exactly) the following four items:

  the abstract number 34, me, the abstract number 59, this university.

Basics, contd

- **{34, JAB, 59, UoB} = {UoB, 59, 34, JAB, JAB, 34}**
  - **Order of writing the members doesn’t matter; duplication in the writing doesn’t duplicate the member.**
- A set can be **infinite** (e.g., the set of all whole numbers).
- A set can contain just **one member** (e.g. the set whose only element is your favourite pencil). **Singleton set.**
  - It’s **different** from the member itself..
- There’s a set **with no members at all**: the “**empty set**”, usually notated as **∅**, but can also be written **{ }**.
  - Somewhat analogous to zero, or a new committee which has no members yet.
  - There is only one empty set (rather than an empty set of numbers, an empty set of pencils, etc.)

Another Notation

- **{n | n is an integer, n > 301}** =
  "The set of n such that n is an integer and n > 301."
  (Actually, this notation is a slight simplification.)

  The set is the ***same*** as that denoted by, for instance,
  {n | n is an integer, n ≥ 302}.

Some More Examples

- **{JAB, “JAB”}** has 2 members: me, and a 3-char string.
- **{3, {4,5}, 4, 6}**has 4 members, one of which is a set.
- **{3, {5,4}, 4, 6}** is that same set.
- **{ {4,5} }** has 1 member, which is a set.

  {4,5} has 2 members, both numbers.
- {∅} is a singleton set. Its only member is the empty set.
- {{∅}} is a different singleton set.