THE RELATIONAL DATABASE MODEL

- Introduction to relational DB
- Basic Objects of relational model
- Properties of relation
- Representation of ER model to relation
- Keys
- Relational Integrity Rules
- Functional Dependencies
- Transitive Dependency

In this chapter, you will learn:

- That the relational database model takes a logical view of data
- The relational model's basic components are relations implemented through tables in a relational DBMS
- How relations are organized in tables composed of rows (tuples) and columns (attributes)

In this chapter, you will learn (continued):

- About relational database operators, the data dictionary, and the system catalog
- How data redundancy is handled in the relational database model
- Why indexing is important

Introduction to relational DB

- Relational model
  - Enables programmer to view data logically rather than physically
- Table
  - Has advantages of structural and data independence
  - Resembles a file from conceptual point of view
  - Easier to understand than its hierarchical and network database predecessors

Tables and Their Characteristics

- Table: two-dimensional structure composed of rows and columns
- Contains group of related entities = an entity set
  - Terms entity set and table are often used interchangeably

Tables and Their Characteristics (continued)

- Table also called a relation because the relational model's creator, Codd, used the term relation as a synonym for table
- Think of a table as a persistent relation:
  - A relation whose contents can be permanently saved for future use

Tables and Their Characteristics (continued)

TABLE 3.1 Characteristics of a Relational Table

1	A table is perceived as a two-dimensional structure composed of rows and columns.
2	Each table row (**tuple**) represents a single entity occurrence within the entity set.
3	Each table column represents an attribute, and each column has a distinct name.
4	Each row/column intersection represents a single data value.
5	All values in a column must conform to the same data format. For example, if the attribute is assigned an integer data format, all values in the column representing that attribute must be integers.
6	Each column has a specific range of values known as the **attribute domain**.
7	The order of the rows and columns is immaterial to the DBMS.
8	Each table must have an attribute or a combination of attributes that uniquely identifies each row.

Tables and Their Characteristics (continued)

FIGURE 3.1 STUDENT table attribute values

Database name: Ch03_TinyCollege
Table name: STUDENT

STU_NUM	STU_LNAME	STU_FNAME	STU_INIT	STU_DOB	STU_HRS	STU_CLASS
321452	Bowser	William	C	12-Feb-1975	42	So
324257	Smithson	Anne	K	15-Nov-1981	81	Jr
324258	Brewer	Juliette		23-Aug-1969	36	So
324269	Oblonski	Walter	H	16-Sep-1976	66	Jr
324273	Smith	John	D	30-Dec-1958	102	Sr
324274	Katinga	Raphael	P	21-Oct-1979	114	Sr
324291	Robertson	Gerald	T	08-Apr-1973	120	Sr
324299	Smith	John	B	30-Nov-1986	15	Fr

STUDENT table, continued

STU_GPA	STU_TRANSFER	DEPT_CODE	STU_PHONE	PROF_NUM
2.84	No	BIOL	2134	205
3.27	Yes	CIS	2256	222
2.26	Yes	ACCT	2256	228
3.09	No	CIS	2114	222
2.11	Yes	ENGL	2231	199
3.15	No	ACCT	2267	228
3.87	No	EDU	2267	311
2.92	No	ACCT	2315	230

STU_HRS = Credit hours earned
STU_CLASS = Student classification
STU_DOB = Student date of birth
STU_GPA = Grade point average
STU_PHONE = 4-digit campus phone extension
PROF_NUM = Number of the professor who is the student's advisor

Keys

- Consists of one or more attributes that determine other attributes
- Primary key (PK) is an attribute (or a combination of attributes) that uniquely identifies any given entity (row)
- Key's role is based on determination
  - If you know the value of attribute A, you can look up (determine) the value of attribute B

Keys (continued)

TABLE 3.2 Student Classification

HOURS COMPLETED	CLASSIFICATION
Less than 30	Fr
30–59	So
60–89	Jr
90 or more	Sr

Keys (continued)

- Composite key
  - Composed of more than one attribute
- Key attribute
  - Any attribute that is part of a key
- Superkey
  - Any key that uniquely identifies each row
- Candidate key
  - A superkey without redundancies

Keys (continued)

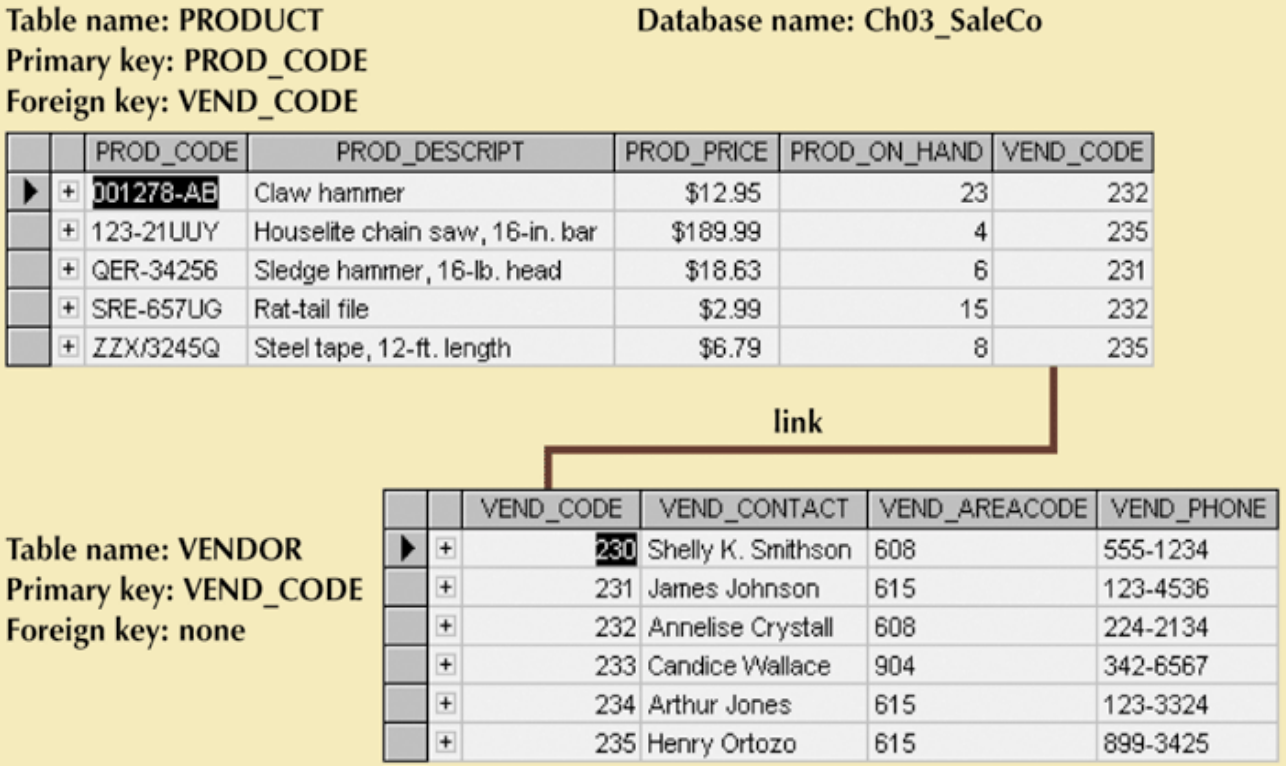
- Nulls:
  - No data entry
  - Not permitted in primary key
  - Should be avoided in other attributes
  - Can represent
    - An unknown attribute value
    - A known, but missing, attribute value
    - A “not applicable” condition
  - Can create problems when functions such as COUNT, AVERAGE, and SUM are used
  - Can create logical problems when relational tables are linked

Keys (continued)

- Controlled redundancy:
  - Makes the relational database work
  - Tables within the database share common attributes that enable the tables to be linked together
  - Multiple occurrences of values in a table are not redundant when they are required to make the relationship work
  - Redundancy exists only when there is unnecessary duplication of attribute values

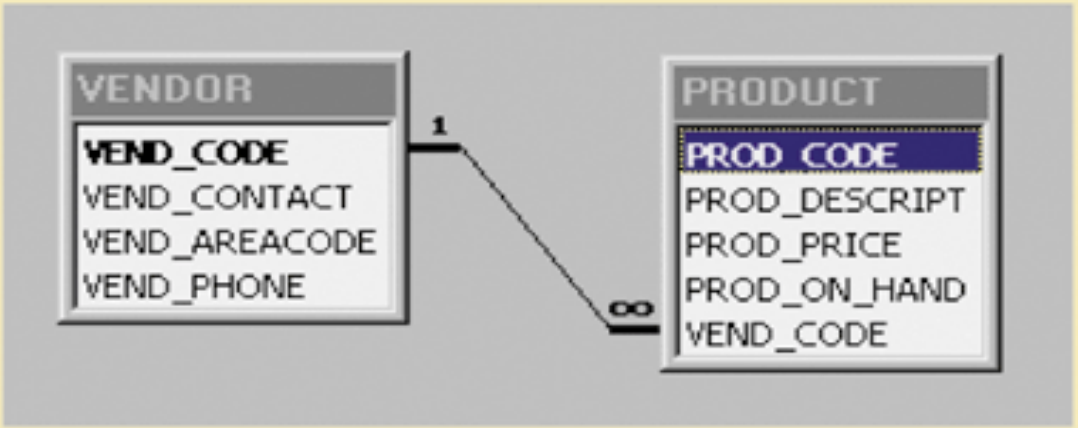
Keys (continued)

FIGURE 3.2 An example of a simple relational database



Keys (continued)

FIGURE 3.3 The relational diagram for the Ch03_SaleCo database



Keys (continued)

- Foreign key (FK)
  - An attribute whose values match primary key values in the related table
- Referential integrity
  - FK contains a value that refers to an existing valid tuple (row) in another relation
- Secondary key
  - Key used strictly for data retrieval purposes

Keys (continued)

TABLE 3.3 Relational Database Keys

KEY TYPE	DEFINITION
Superkey	An attribute (or combination of attributes) that uniquely identifies each row in a table.
Candidate key	A minimal superkey. A superkey that does not contain a subset of attributes that is itself a superkey.
Primary key	A candidate key selected to uniquely identify all other attribute values in any given row. Cannot contain null entries.
Secondary key	An attribute (or combination of attributes) used strictly for data retrieval purposes.
Foreign key	An attribute (or combination of attributes) in one table whose values must either match the primary key in another table or be null.

Integrity Rules

TABLE 3.4 Integrity Rules

ENTITY INTEGRITY	DESCRIPTION
Requirement	All primary key entries are unique, and no part of a primary key may be null.
Purpose	Each row will have a unique identity, and foreign key values can properly reference primary key values.
Example	No invoice can have a duplicate number, nor can it be null. In short, all invoices are uniquely identified by their invoice number.
REFERENTIAL INTEGRITY	**DESCRIPTION**
Requirement	A foreign key may have either a null entry—as long as it is not a part of its table's primary key—or an entry that matches the primary key value in a table to which it is related. (Every non-null foreign key value *must* reference an *existing* primary key value.)
Purpose	It is possible for an attribute NOT to have a corresponding value, but it will be impossible to have an invalid entry. The enforcement of the referential integrity rule makes it impossible to delete a row in one table whose primary key has mandatory matching foreign key values in another table.
Example	A customer might not yet have an assigned sales representative (number), but it will be impossible to have an invalid sales representative (number).

Integrity Rules (continued)

FIGURE 3.4 An illustration of integrity rules

Table name: CUSTOMER **Database name: Ch03_InsureCo**
Primary key: CUS_CODE
Foreign key: AGENT_CODE

CUS_CODE	CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_AREACODE	CUS_PHONE	CUS_RENEW_DATE	AGENT_CODE
10010	Ramas	Alfred	A	615	844-2573	12-Mar-06	502
10011	Dunne	Leona	K	713	894-1238	23-May-06	501
10012	Smith	Kathy	W	615	894-2285	05-Jan-06	502
10013	Olowski	Paul	F	615	894-2180	20-Sep-06	
10014	Orlando	Myron		615	222-1672	04-Dec-06	501
10015	O'Brian	Amy	B	713	442-3381	29-Aug-06	503
10016	Brown	James	G	615	297-1228	01-Mar-06	502
10017	Williams	George		615	290-2556	23-Jun-06	503
10018	Farriss	Anne	G	713	382-7185	09-Nov-06	501
10019	Smith	Olette	K	615	297-3809	18-Feb-06	503

Table name: AGENT
Primary key: AGENT_CODE
Foreign key: none

AGENT_CODE	AGENT_AREACODE	AGENT_PHONE	AGENT_LNAME	AGENT_YTD_SLS
501	713	228-1249	Alby	$1,735,453.75
502	615	882-1244	Hahn	$4,967,003.28
503	615	123-5589	Okon	$3,093,980.41

Integrity Rules (continued)

TABLE 3.5 A Dummy Variable Value Used as a Flag

AGENT_CODE	AGENT_AREACODE	AGENT_PHONE	AGENT_LNAME	AGENT_YTD_SALES
-99	000	000-0000	None	$0.00

Functional Dependencies

- attribute B is functionally dependent on attribute A if given a value of attribute A, there is only one possible corresponding value of attribute B
    - that is, any two rows with the same value of A must have the same value for B
- attribute A is the determinant of attribute B if attribute B is functionally dependent on attribute A

Example

STATE(StateAbbrev, StateName, UnionOrder, StateBird, StatePopulation)

CITY(StateAbbrev, CityName, CityPopulation)
StateAbbrev foreign key to STATE

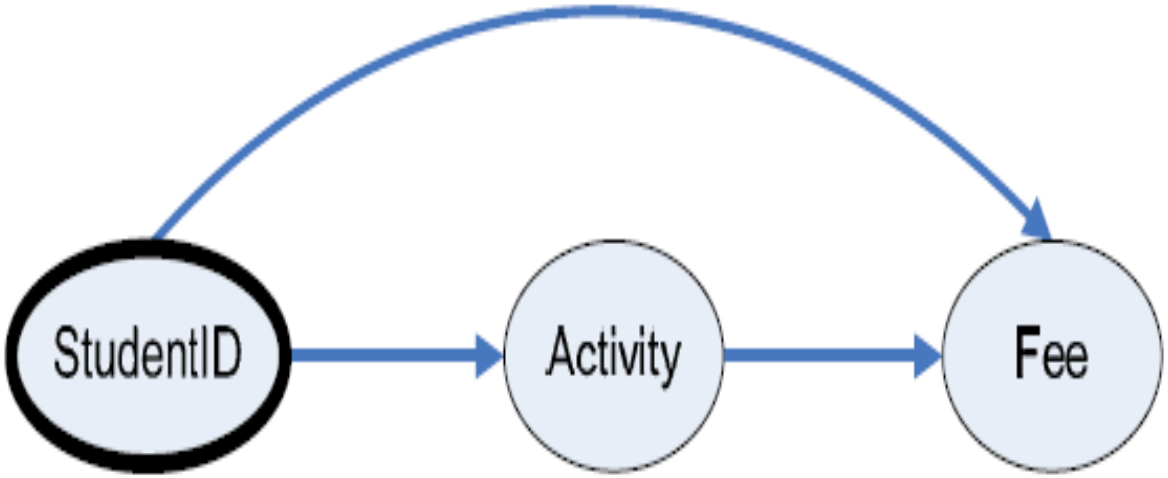
- in the STATE relation above, StateAbbrev is a determinant of all other attributes
- in the STATE relation, the attribute StateName is also a determinant of all other attributes
- so, StateAbbrev and StateName are both candidate keys for STATE
- in the CITY relation above, the attributes (StateAbbrev, CityName) together are a determinant of the attribute CityPopulation
- in the CITY relation, the attribute CityName is not a determinant of the attribute CityPopulation because multiple cities in the table may have the same name

Dependency Diagrams

- a dependency diagram or bubble diagram is a pictorial representation of functional dependencies
  - an attribute is represented by an oval
  - you draw an arrow from A to B when attribute A is a determinant of attribute B

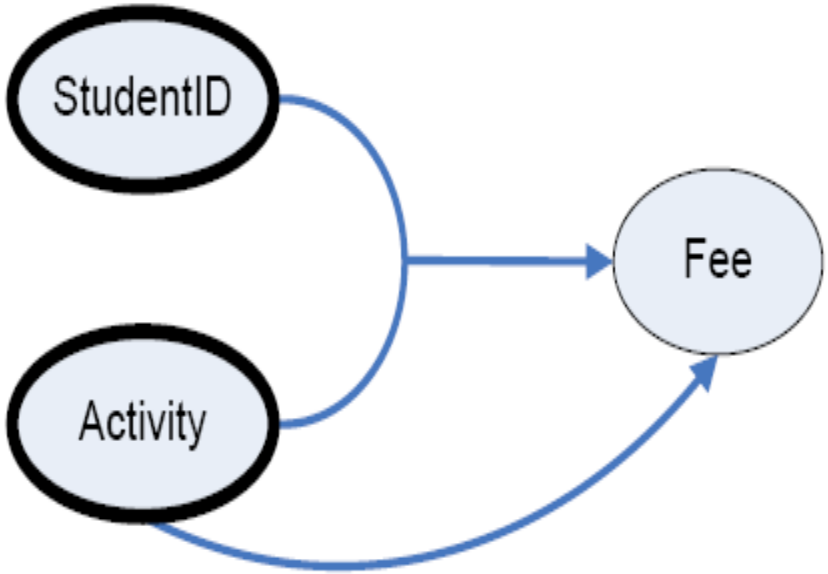
Dependency Diagrams

- example: when students were only allowed one sports activity, we have ACTIVITY(StudentID, Activity, Fee)



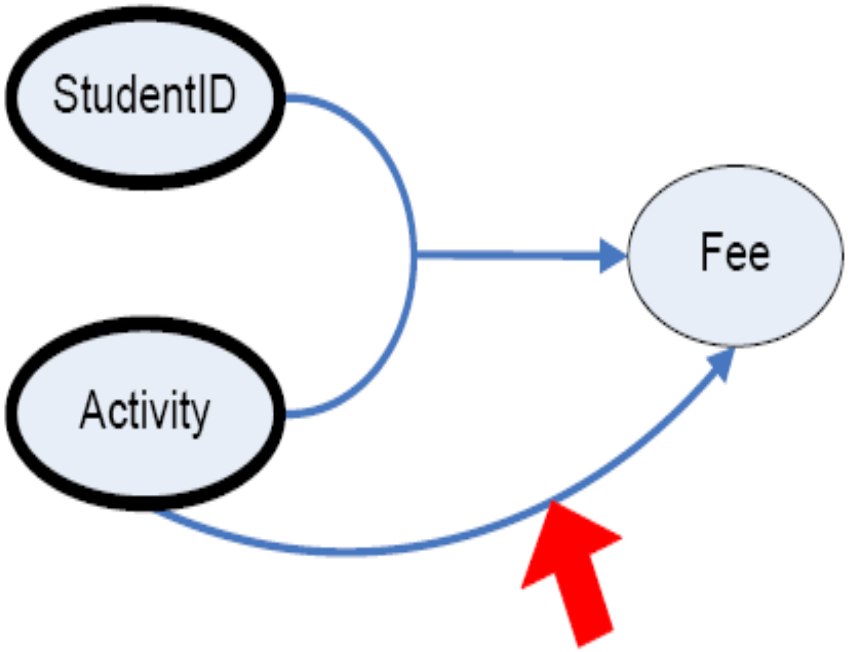
Dependency Diagrams

- example: when students can have multiple activities, we have ACTIVITY(StudentID, Activity, Fee)



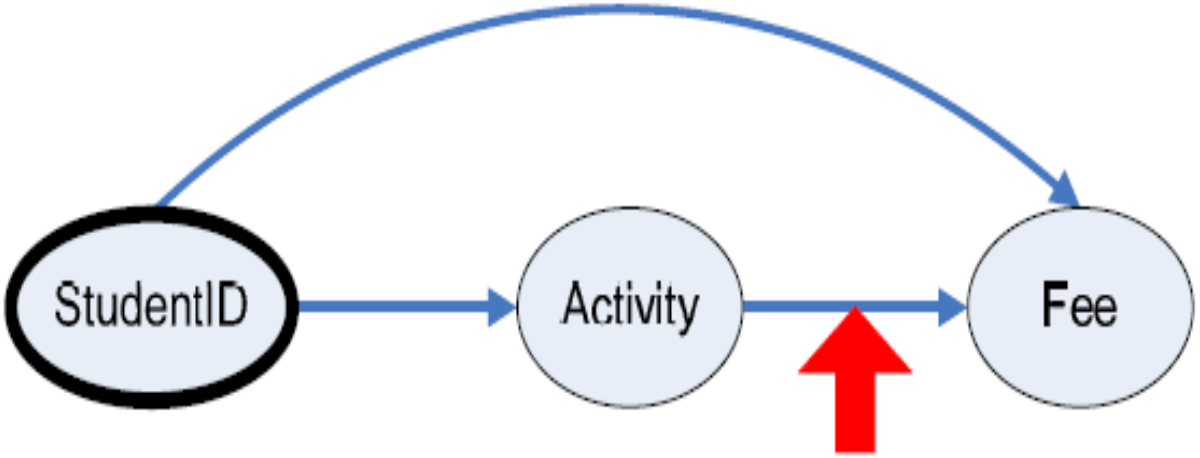
Partial Dependencies

- a partial dependency is a functional dependency whose determinant is part of the primary key (but not all of it)
- example: ACTIVITY(StudentID, Activity, Fee)



Transitive Dependencies

- a transitive dependency is a functional dependency whose determinant is not the primary key, part of the primary key, or a candidate key
- example: ACTIVITY(StudentID, Activity, Fee)



Relational Database Operators

- **Relational algebra**
  - Defines theoretical way of manipulating table contents using relational operators
  - Use of relational algebra operators on existing tables (relations) produces new relations

Relational Algebra Operators (continued)

- UNION
- INTERSECT
- DIFFERENCE
- PRODUCT
- SELECT
- PROJECT
- JOIN
- DIVIDE

Relational Algebra Operators (continued)

- Union:
  - Combines all rows from two tables, excluding duplicate rows
  - Tables must have the same attribute characteristics
- Intersect:
  - Yields only the rows that appear in both tables

Relational Algebra Operators (continued)

FIGURE 3.5 UNION

P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	$5.26
123457	Lamp	$25.15
123458	Box Fan	$10.99
213345	9v battery	$1.92
254467	100W bulb	$1.47
311452	Powerdrill	$34.99

UNION

P_CODE	P_DESCRIPT	PRICE
345678	Microwave	$160.00
345679	Dishwasher	$500.00

yields

P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	$5.26
123457	Lamp	$25.15
123458	Box Fan	$10.99
213345	9v battery	$1.92
254467	100W bulb	$1.47
311452	Powerdrill	$34.99
345678	Microwave	$160.00
345679	Dishwasher	$500.00

Relational Algebra Operators (continued)

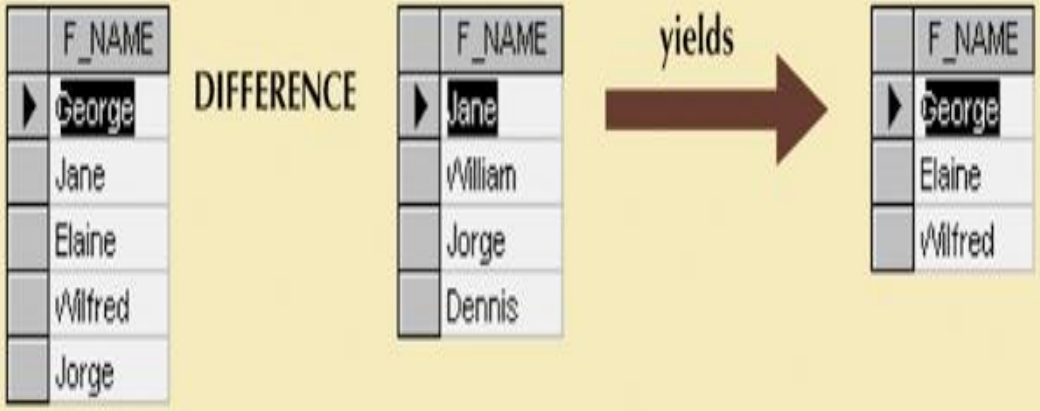
FIGURE 3.6 INTERSECT

Relational Algebra Operators (continued)

- Difference
  - Yields all rows in one table not found in the other table — that is, it subtracts one table from the other
- Product
  - Yields all possible pairs of rows from two tables
    - Also known as the Cartesian product

Relational Algebra Operators (continued)

FIGURE 3.7 DIFFERENCE



Relational Algebra Operators (continued)

FIGURE 3.8 PRODUCT

P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	$5.26
123457	Lamp	$25.15
123458	Box Fan	$10.99
213345	9v battery	$1.92
254467	100W bulb	$1.47
311452	Powerdrill	$34.99

PRODUCT

STORE	AISLE	SHELF
23	W	5
24	K	9
25	Z	6

yields

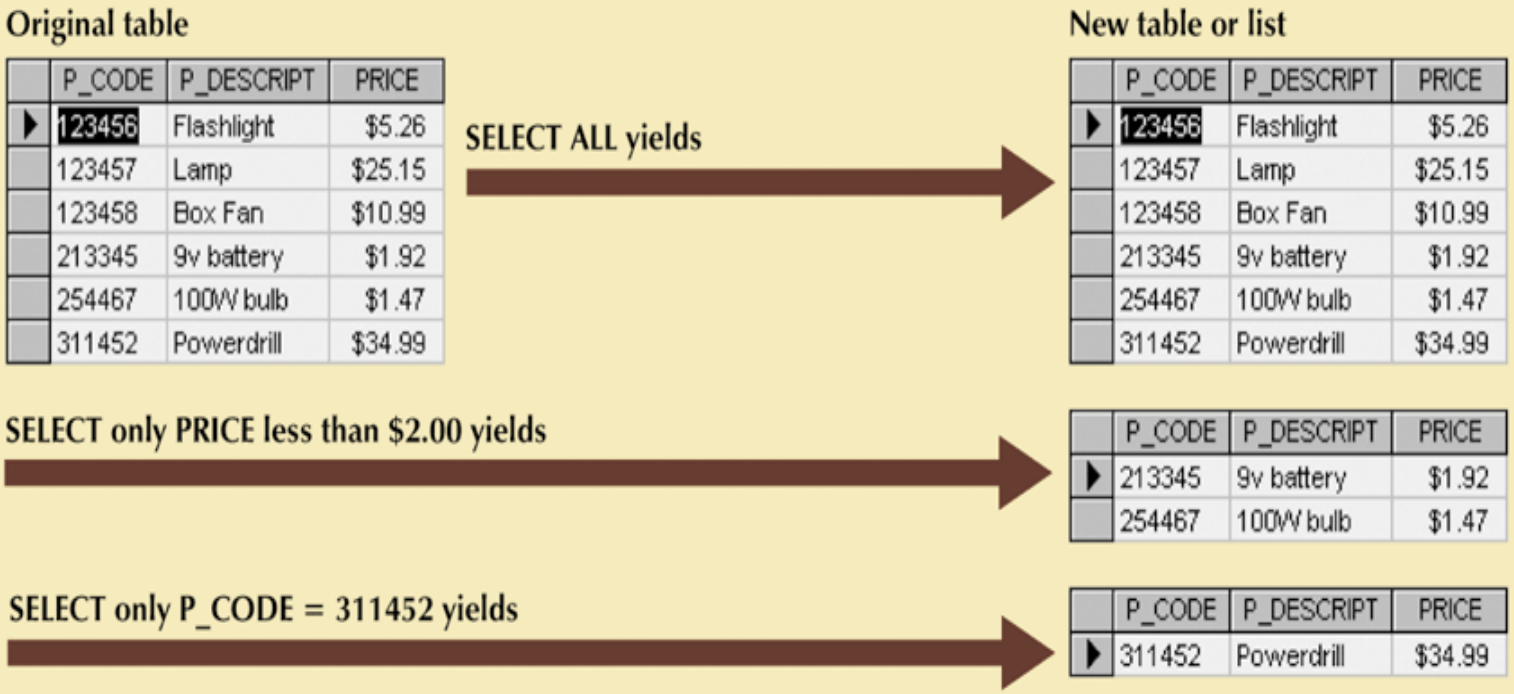
P_CODE	P_DESCRIPT	PRICE	STORE	AISLE	SHELF
123456	Flashlight	$5.26	23	W	5
123456	Flashlight	$5.26	24	K	9
123456	Flashlight	$5.26	25	Z	6
123457	Lamp	$25.15	23	W	5
123457	Lamp	$25.15	24	K	9
123457	Lamp	$25.15	25	Z	6
123458	Box Fan	$10.99	23	W	5
123458	Box Fan	$10.99	24	K	9
123458	Box Fan	$10.99	25	Z	6
213345	9v battery	$1.92	23	W	5
213345	9v battery	$1.92	24	K	9
213345	9v battery	$1.92	25	Z	6
311452	Powerdrill	$34.99	23	W	5
311452	Powerdrill	$34.99	24	K	9
311452	Powerdrill	$34.99	25	Z	6
254467	100W bulb	$1.47	23	W	5
254467	100W bulb	$1.47	24	K	9
254467	100W bulb	$1.47	25	Z	6

Relational Algebra Operators (continued)

- Select
  - Yields values for all rows found in a table
  - Can be used to list either all row values or it can yield only those row values that match a specified criterion
  - Yields a horizontal subset of a table
- Project
  - Yields all values for selected attributes
  - Yields a vertical subset of a table

Relational Algebra Operators (continued)

FIGURE 3.9 SELECT



Relational Algebra Operators (continued)

FIGURE 3.10 PROJECT

Original table

P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	$5.26
123457	Lamp	$25.15
123458	Box Fan	$10.99
213345	9v battery	$1.92
254467	100W bulb	$1.47
311452	Powerdrill	$34.99

PROJECT PRICE yields

New table or list

PRICE
$5.26
$25.15
$10.99
$1.92
$1.47
$34.99

PROJECT P_DESCRIPT and PRICE yields

P_DESCRIPT	PRICE
Flashlight	$5.26
Lamp	$25.15
Box Fan	$10.99
9v battery	$1.92
100W bulb	$1.47
Powerdrill	$34.99

PROJECT P_CODE and PRICE yields

P_CODE	PRICE
123456	$5.26
123457	$25.15
123458	$10.99
213345	$1.92
254467	$1.47
311452	$34.99

Relational Algebra Operators (continued)

- Join
  - Allows information to be combined from two or more tables
  - Real power behind the relational database, allowing the use of independent tables linked by common attributes

Relational Algebra Operators (continued)

FIGURE 3.11 Two tables that will be used in join illustrations

Table name: CUSTOMER

CUS_CODE	CUS_LNAME	CUS_ZIP	AGENT_CODE
1132445	Walker	32145	231
1217782	Adares	32145	125
1312243	Rakowski	34129	167
1321242	Rodriguez	37134	125
1542311	Smithson	37134	421
1657399	Vanloo	32145	231

Table name: AGENT

AGENT_CODE	AGENT_PHONE
125	6152439887
167	6153426778
231	6152431124
333	9041234445

Relational Algebra Operators (continued)

- Natural Join
  - Links tables by selecting only rows with common values in their common attribute(s)
  - Result of a three-stage process:
    - PRODUCT of the tables is created
    - SELECT is performed on Step 1 output to yield only the rows for which the AGENT_CODE values are equal
      - Common column(s) are called join column(s)
    - PROJECT is performed on Step 2 results to yield a single copy of each attribute, thereby eliminating duplicate columns

Relational Algebra Operators (continued)

FIGURE 3.12 Natural join, Step 1: PRODUCT

CUS_CODE	CUS_LNAME	CUS_ZIP	CUSTOMER.AGENT_CODE	AGENT.AGENT_CODE	AGENT_PHONE
1132445	Walker	32145	231	125	6152439887
1132445	Walker	32145	231	167	6153426778
1132445	Walker	32145	231	231	6152431124
1132445	Walker	32145	231	333	9041234445
1217782	Adares	32145	125	125	6152439887
1217782	Adares	32145	125	167	6153426778
1217782	Adares	32145	125	231	6152431124
1217782	Adares	32145	125	333	9041234445
1312243	Rakowski	34129	167	125	6152439887
1312243	Rakowski	34129	167	167	6153426778
1312243	Rakowski	34129	167	231	6152431124
1312243	Rakowski	34129	167	333	9041234445
1321242	Rodriguez	37134	125	125	6152439887
1321242	Rodriguez	37134	125	167	6153426778
1321242	Rodriguez	37134	125	231	6152431124
1321242	Rodriguez	37134	125	333	9041234445
1542311	Smithson	37134	421	125	6152439887
1542311	Smithson	37134	421	167	6153426778
1542311	Smithson	37134	421	231	6152431124
1542311	Smithson	37134	421	333	9041234445
1657399	Vanloo	32145	231	125	6152439887
1657399	Vanloo	32145	231	167	6153426778
1657399	Vanloo	32145	231	231	6152431124
1657399	Vanloo	32145	231	333	9041234445

Relational Algebra Operators (continued)

FIGURE 3.13 Natural join, Step 2: SELECT

CUS_CODE	CUS_LNAME	CUS_ZIP	CUSTOMER.AGENT_CODE	AGENT.AGENT_CODE	AGENT_PHONE
1217782	Adares	32145	125	125	6152439887
1321242	Rodriguez	37134	125	125	6152439887
1312243	Rakowski	34129	167	167	6153426778
1132445	Walker	32145	231	231	6152431124
1657399	Vanloo	32145	231	231	6152431124

Relational Algebra Operators (continued)

FIGURE 3.14 Natural join, Step 3: PROJECT

	CUS_CODE	CUS_LNAME	CUS_ZIP	AGENT_CODE	AGENT_PHONE
▶	217782	Adares	32145	125	6152439887
	1321242	Rodriguez	37134	125	6152439887
	1312243	Rakowvski	34129	167	6153426778
	1132445	Walker	32145	231	6152431124
	1657399	Vanloo	32145	231	6152431124

Relational Algebra Operators (continued)

- Natural Join:
  - Final outcome yields table that
    - Does not include unmatched pairs
    - Provides only copies of matches
  - If no match is made between the table rows
    - the new table does not include the unmatched row

Relational Algebra Operators (continued)

- Natural Join (continued):
  - The column on which the join was made - that is, AGENT_CODE - occurs only once in the new table
  - If the same AGENT_CODE were to occur several times in the AGENT table,
    - a customer would be listed for each match

Relational Algebra Operators (continued)

- Equijoin
  - Links tables on the basis of an equality condition that compares specified columns of each table
  - Outcome does not eliminate duplicate columns
  - Condition or criterion to join tables must be explicitly defined
  - Takes its name from the equality comparison operator (=) used in the condition
- Theta join
  - If any other comparison operator is used

Relational Algebra Operators (continued)

- Outer join:
  - Matched pairs are retained and any unmatched values in other table are left null
  - In outer join for tables CUSTOMER and AGENT, two scenarios are possible:
    - Left outer join
      - Yields all rows in CUSTOMER table, including those that do not have a matching value in the AGENT table
    - Right outer join
      - Yields all rows in AGENT table, including those that do not have matching values in the CUSTOMER table

Relational Algebra Operators (continued)

FIGURE 3.15 Left outer join

	CUS_CODE	CUS_LNAME	CUS_ZIP	AGENT_CODE	AGENT_PHONE
▶	1217782	Adares	32145	125	6152439887
	1321242	Rodriguez	37134	125	6152439887
	1312243	Rakowski	34129	167	6153426778
	1132445	Walker	32145	231	6152431124
	1657399	Vanloo	32145	231	6152431124
	1542311	Smithson	37134	421	

Relational Algebra Operators (continued)

FIGURE 3.16 Right outer join

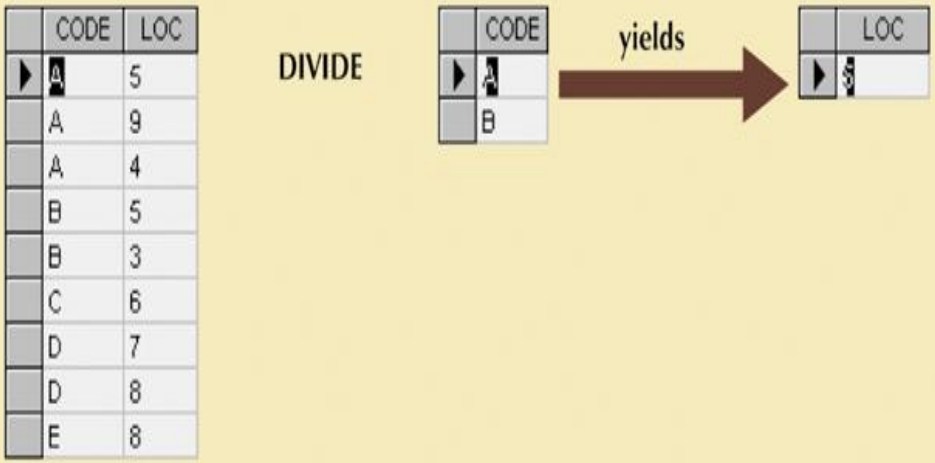
CUS_CODE	CUS_LNAME	CUS_ZIP	AGENT_CODE	AGENT_PHONE
1217782	Adares	32145	125	6152439887
1321242	Rodriguez	37134	125	6152439887
1312243	Rakowski	34129	167	6153426778
1132445	Walker	32145	231	6152431124
1657399	Vanloo	32145	231	6152431124
			333	9041234445

Relational Algebra Operators (continued)

- DIVIDE requires the use of one single-column table and one two-column table

Relational Algebra Operators (continued)

FIGURE 3.17 DIVIDE



Relationships within the Relational Database

- 1:M relationship
  - Relational modeling ideal
  - Should be the norm in any relational database design
- 1:1 relationship
  - Should be rare in any relational database design
- M:N relationships
  - Cannot be implemented as such in the relational model
  - M:N relationships can be changed into two 1:M relationships

The 1:M Relationship (continued)

FIGURE 3.19 The implemented 1:M relationship between PAINTER and PAINTING

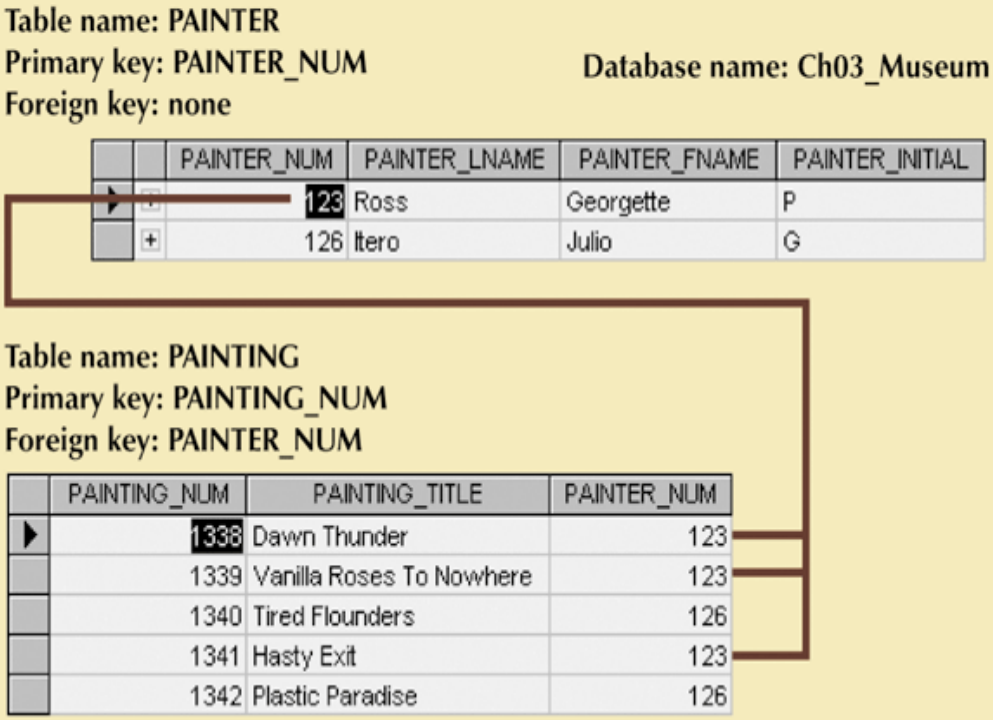


Table name: PAINTER
Primary key: PAINTER_NUM
Foreign key: none

Database name: Ch03_Museum

PAINTER_NUM	PAINTER_LNAME	PAINTER_FNAME	PAINTER_INITIAL
123	Ross	Georgette	P
126	Itero	Julio	G

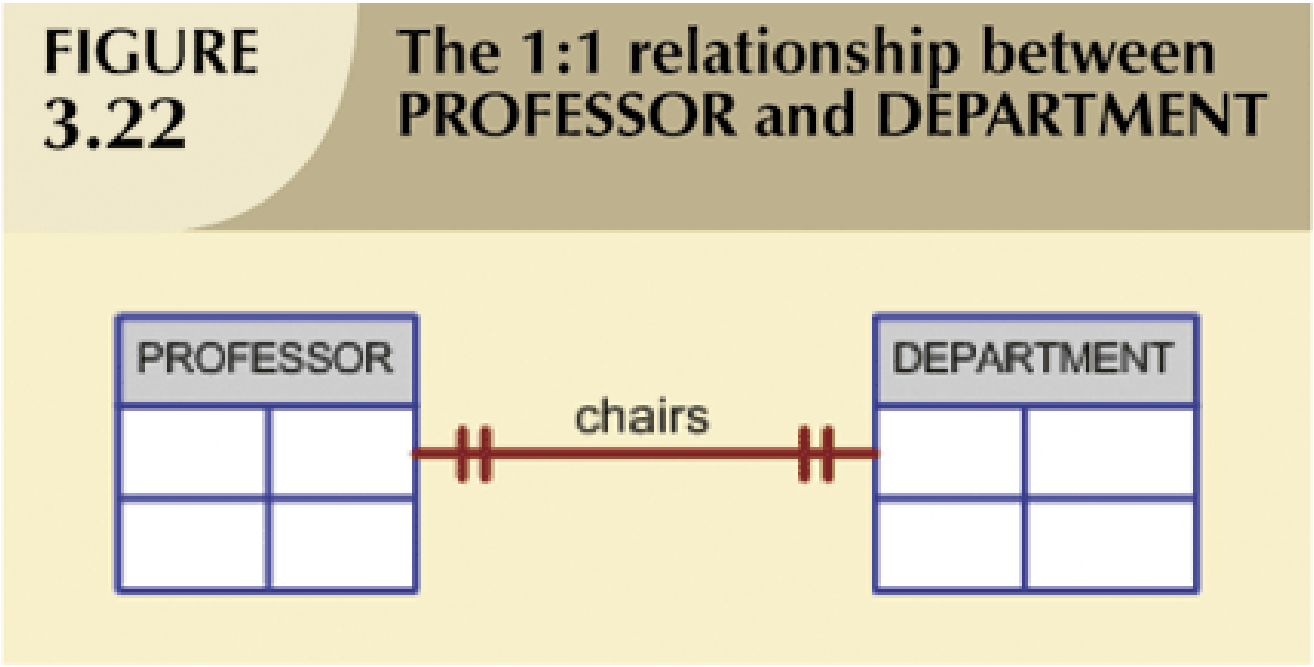
Table name: PAINTING
Primary key: PAINTING_NUM
Foreign key: PAINTER_NUM

PAINTING_NUM	PAINTING_TITLE	PAINTER_NUM
1338	Dawn Thunder	123
1339	Vanilla Roses To Nowhere	123
1340	Tired Flounders	126
1341	Hasty Exit	123
1342	Plastic Paradise	126

The 1:1 Relationship

- One entity can be related to only one other entity, and vice versa
- Sometimes means that entity components were not defined properly
- Could indicate that two entities actually belong in the same table
- As rare as 1:1 relationships should be, certain conditions absolutely require their use

The 1:1 Relationship (continued)

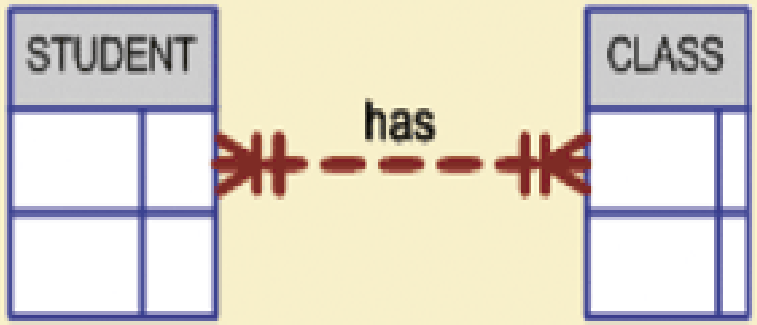


The M:N Relationship

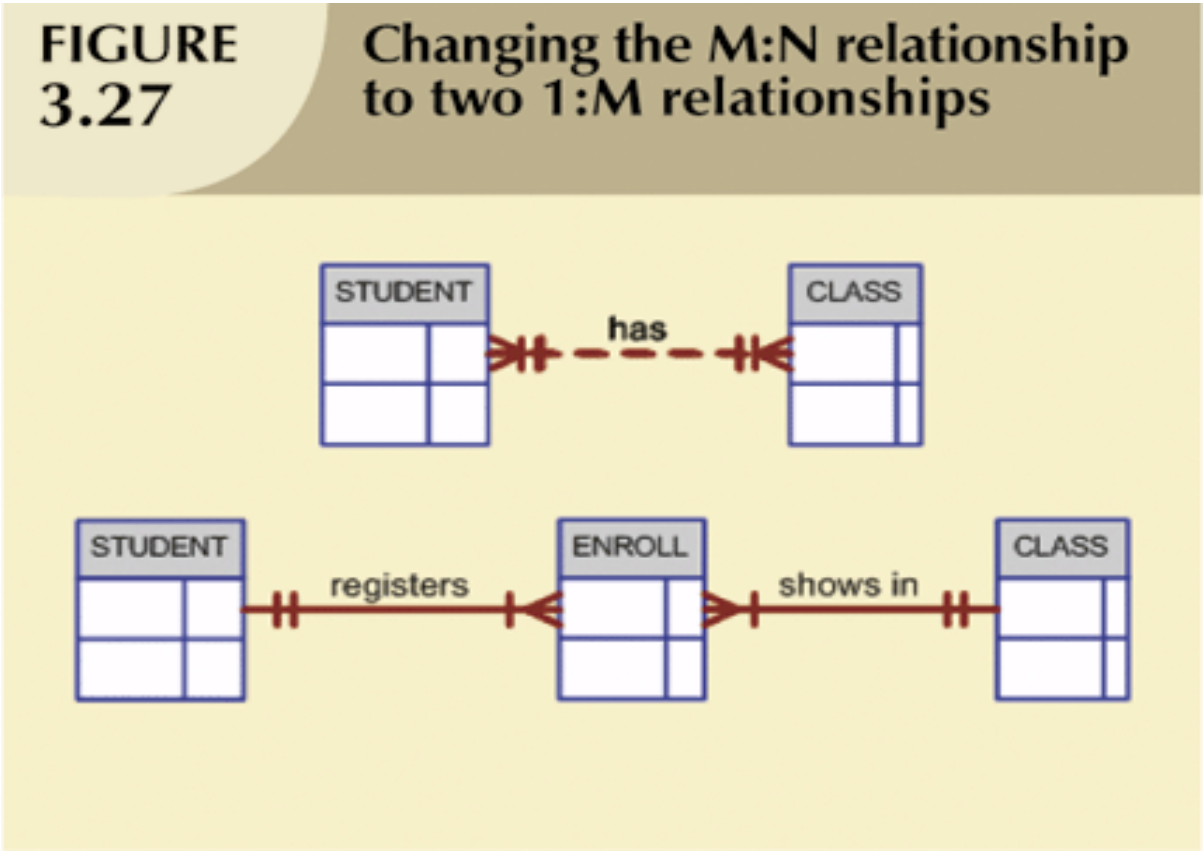
- Can be implemented by breaking it up to produce a set of 1:M relationships
- Can avoid problems inherent to M:N relationship by creating a composite entity or bridge entity

The M:N Relationship (continued)

FIGURE 3.24 The ERD's M:N relationship between STUDENT and CLASS

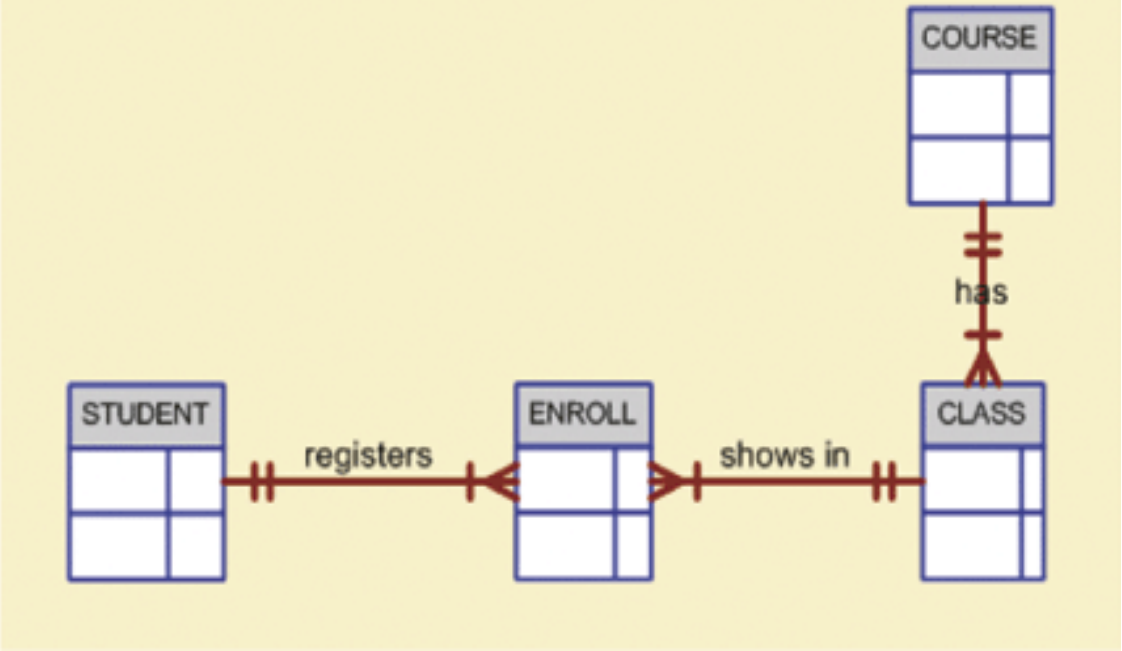


The M:N Relationship (continued)



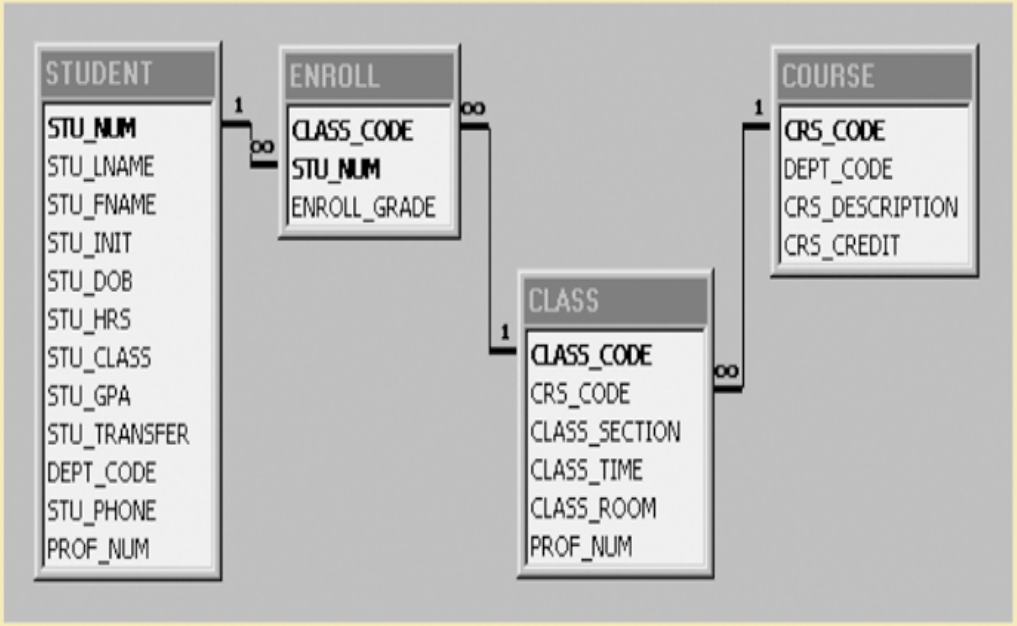
The M:N Relationship (continued)

FIGURE 3.28 The expanded entity relationship model



The M:N Relationship (continued)

FIGURE 3.29 The relational diagram for the Ch03_TinyCollege database



Data Redundancy Revisited

- Data redundancy leads to data anomalies
  - Such anomalies can destroy the effectiveness of the database
- Foreign keys
  - Control data redundancies by using common attributes shared by tables
  - Crucial to exercising data redundancy control
- Sometimes, data redundancy is necessary

Data Redundancy Revisited (continued)

FIGURE 3.30 A small invoicing system

Table name: CUSTOMER
Primary key: CUS_CODE
Foreign key: none
Database name: Ch03_SaleCo

CUS_CODE	CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_AREACODE	CUS_PHONE
10010	Ramas	Alfred	A	615	844-2573
10011	Dunne	Leona	K	713	894-1238
10012	Smith	Kathy	W	615	894-2285
10013	Olowski	Paul	F	615	894-2180
10014	Orlando	Myron		615	222-1672
10015	O'Brian	Amy	B	713	442-3381
10016	Brown	James	G	615	297-1228
10017	Williams	George		615	290-2556
10018	Farriss	Anne	G	713	382-7185
10019	Smith	Olette	K	615	297-3809

Table name: INVOICE
Primary key: INV_NUMBER
Foreign key: CUS_CODE

INV_NUMBER	CUS_CODE	INV_DATE
1001	10014	08-Mar-06
1002	10011	08-Mar-06
1003	10012	08-Mar-06
1004	10011	09-Mar-06

Table name: LINE
Primary key: INV_NUMBER + LINE_NUMBER
Foreign keys: INV_NUMBER, PROD_CODE

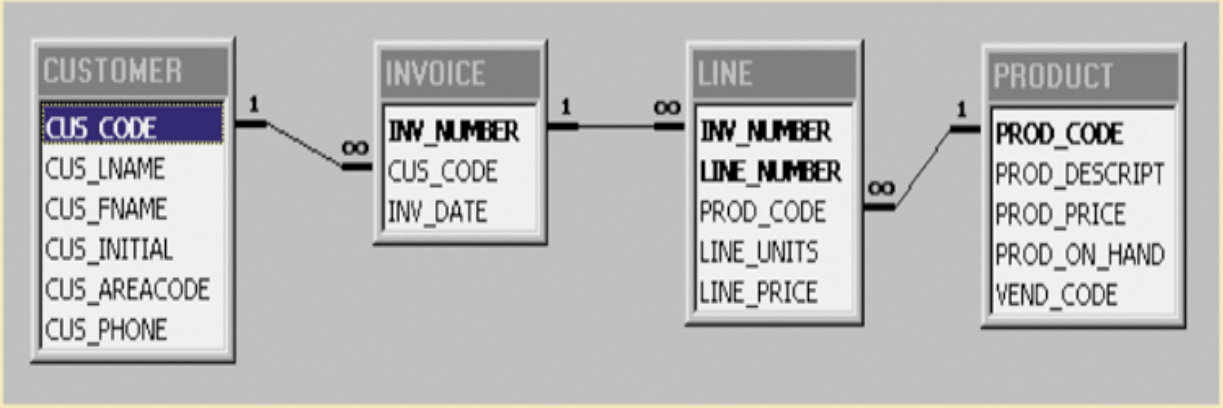
INV_NUMBER	LINE_NUMBER	PROD_CODE	LINE_UNITS	LINE_PRICE
1001	1	123-21UUY	1	$189.99
1001	2	SRE-657UG	3	$2.99
1002	1	QER-34256	2	$18.63
1003	1	ZZX/3245Q	1	$6.79
1003	2	SRE-657UG	1	$2.99
1003	3	001278-AB	1	$12.95
1004	1	001278-AB	1	$12.95
1004	2	SRE-657UG	2	$2.99

Table name: PRODUCT
Primary key: PROD_CODE
Foreign key: none

PROD_CODE	PROD_DESCRIPT	PROD_PRICE	PROD_ON_HAND	VEND_CODE
001278-AB	Claw hammer	$12.95	23	232
123-21UUY	Houselite chain saw, 16-in. bar	$189.99	4	235
QER-34256	Sledge hammer, 16-lb. head	$18.63	6	231
SRE-657UG	Rat-tail file	$2.99	15	232
ZZX/3245Q	Steel tape, 12-ft. length	$6.79	8	235

Data Redundancy Revisited (continued)

FIGURE 3.31 The relational diagram for the invoicing system



Codd's Relational Database Rules

- In 1985, Codd published a list of 12 rules to define a relational database system
- The reason was the concern that many vendors were marketing products as "relational" even though those products did not meet minimum relational standards

Codd's Relational Database Rules (Continued)

TABLE 3.8 Dr. Codd's 12 Relational Database Rules

RULE	RULE NAME	DESCRIPTION
1	**Information**	All information in a relational database must be logically represented as column values in rows within tables.
2	**Guaranteed Access**	Every value in a table is guaranteed to be accessible through a combination of table name, primary key value, and column name.
3	**Systematic Treatment of Nulls**	Nulls must be represented and treated in a systematic way, independent of data type.
4	**Dynamic On-Line Catalog Based on the Relational Model**	The metadata must be stored and managed as ordinary data, that is, in tables within the database. Such data must be available to authorized users using the standard database relational language.
5	**Comprehensive Data Sublanguage**	The relational database may support many languages. However it must support one well- defined declarative language with support for data definition, view definition, data manipulation (interactive and by program), integrity constraints, authorization, and transaction management (begin, commit, and rollback).
6	**View Updating**	Any view that is theoretically updatable must be updatable through the system.
7	**High-Level Insert, Update and Delete**	The database must support set-level inserts, updates, and deletes.
8	**Physical Data Independence**	Application programs and ad hoc facilities are logically unaffected when physical access methods or storage structures are changed.
9	**Logical Data Independence**	Application programs and ad hoc facilities are logically unaffected when changes are made to the table structures that preserve the original table values (changing order of column or inserting columns).
10	**Integrity Independence**	All relational integrity constraints must be definable in the relational language and stored in the system catalog, not at the application level.
11	**Distribution Independence**	The end users and application programs are unaware and unaffected by the data location (distributed vs. local databases).
12	**Nonsubversion**	If the system supports low-level access to the data, there must not be a way to bypass the integrity rules of the database.
	Rule Zero	All preceding rules are based on the notion that in order for a database to be considered relational, it must use its relational facilities exclusively to manage the database.

Summary

- Tables are basic building blocks of a relational database
- Keys are central to the use of relational tables
- Keys define functional dependencies
  - Superkey
  - Candidate key
  - Primary key
  - Secondary key
  - Foreign key

Summary (continued)

- Each table row must have a primary key which uniquely identifies all attributes
- Tables can be linked by common attributes. Thus, the primary key of one table can appear as the foreign key in another table to which it is linked
- The relational model supports relational algebra functions: SELECT, PROJECT, JOIN, INTERSECT, UNION, DIFFERENCE, PRODUCT, and DIVIDE.
- Good design begins by identifying appropriate entities and attributes and the relationships among the entities. Those relationships (1:1, 1:M, and M:N) can be represented using ERDs.