Adem Woodroffe

DATABASE DESIGN AND CONCEPTS (notes)

CONTENTS

DATABASES PRELUDE

What is a Database, what is a Relational Database and what is Database Design?

Databases are specialized structures that allow computer-based systems to store, manage, and retrieve data very quickly. Virtually all modern business systems rely on databases.

A relational database is a type of database that organises data into rows and columns, which collectively form a table where the data points are related to each other. It organises data using relations, allowing us to insert, update and query the database. We should store only appropriate data in databases.

Determining entities, the attributes they are represented by and the relationships between them, is the heart and essence of Database Design.

Database Concepts

Database Concepts

"Conventional" Relational Database Design
Design Models
E-R Modelling (CSDE)
Structured Query Language – SQL
Security
Transactions and Concurrency
“New” Database Concepts

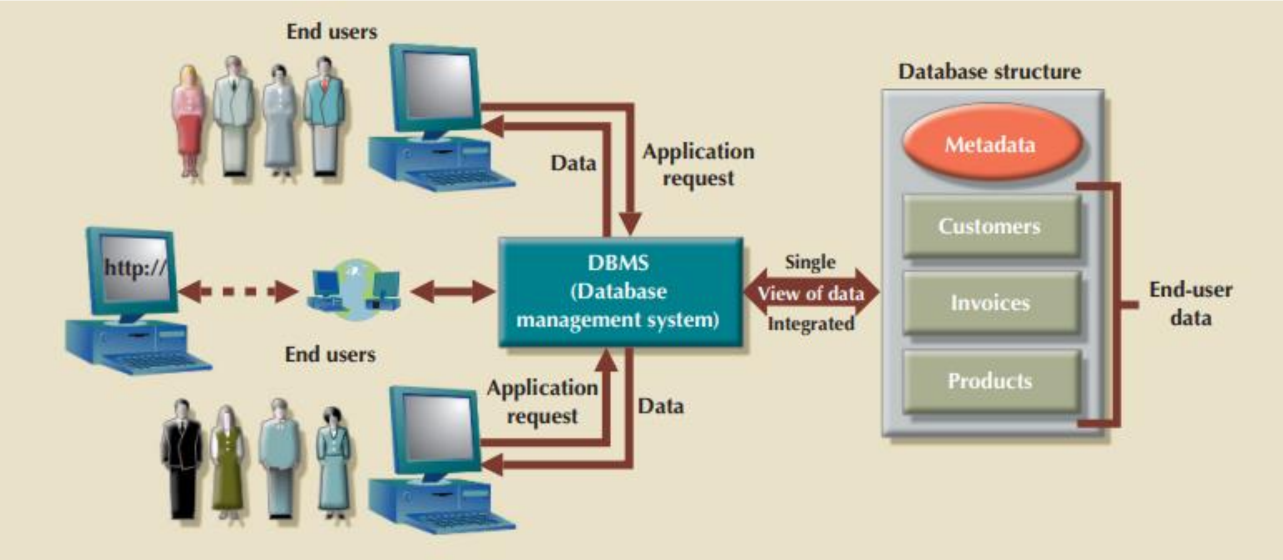
Data and Information

Data is the foundation of information, which is the bedrock of knowledge.

• Data constitutes the building blocks of information.
• Information is produced by processing data.
• Information is used to reveal the meaning of data.
• Accurate, relevant, and timely information is the key to good decision making.
• Good decision making is the key to organizational survival in a global environment

Database Management Systems (DBMS)

A database management system (DBMS) is a collection of programs that manages the database structure and controls access to the data stored in the database. In a sense, a database resembles a very well-organized electronic filing cabinet in which powerful software (the DBMS) helps manage the cabinet's contents. The DBMS serves as the intermediary between the user and the database.



Advantages of using a DBMS:

• Improved data sharing.

• Improved data security.

• Better data integration. Wider access to well-managed data promotes an integrated view of the organization's operations and a clearer view of the big picture. It becomes much easier to see how actions in one segment of the company affect other segments.

• Minimized data inconsistency. Data inconsistency exists when different versions of the same data appears in different places. For example, data inconsistency exists when a company's sales department stores a sales representative's name as Bill Brown and the company's personnel department stores that same person's name as William G. Brown

• Improved data access. The DBMS makes it possible to produce quick answers to ad hoc queries. From a database perspective, a query is a specific request issued to the DBMS for data manipulation—for example, to read or update the data

• Improved decision making.

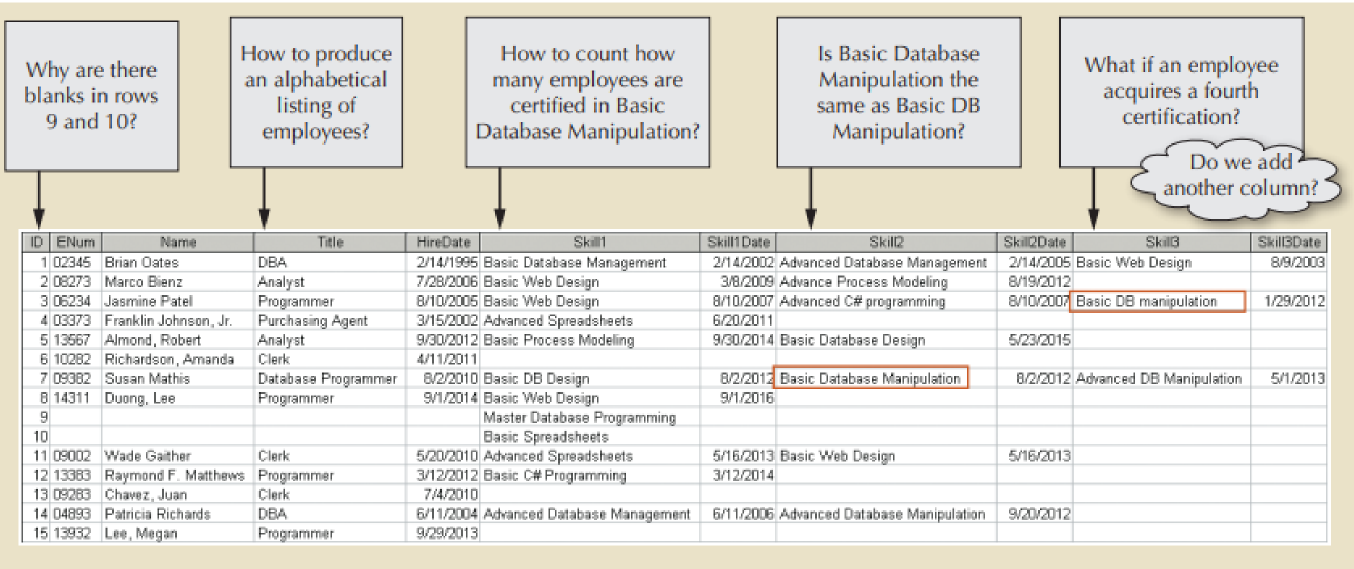
• Increased end-user productivity. The availability of data, combined with the tools that transform data into usable information, empowers end users to make quick, informed decisions that can make the difference between success and failure in the global economy.

Types of Databases

A single-user database supports only one user at a time. In other words, if user A is using the database, users B and C must wait until user A is done. A single-user database that runs on a personal computer is called a desktop database. In contrast, a multi-user database supports multiple users at the same time. When the multi-user database supports a relatively small number of users (usually fewer than 50) or a specific department within an organization, it is called a workgroup database. When the database is used by the entire organization and supports many users (more than 50, usually hundreds) across many departments, the database is known as an enterprise database. Location might also be used to classify the database. For example, a database that supports data located at a single site is called a centralized database. A database that supports data distributed across several different sites is called a distributed database.

Poor Database Design vs Good Database Design

A poor Database Design:



ID	ENum	Name	Title	HireDate	Skill1	Skill1Date	Skill2	Skill2Date	Skill3	Skill3Date
1	02345	Brian Oates	DBA	2/14/1995	Basic Database Management	2/14/2002	Advanced Database Management	2/14/2005	Basic Web Design	8/9/2003
2	08273	Marco Bienz	Analyst	7/28/2006	Basic Web Design	3/8/2009	Advance Process Modeling	8/19/2012		
3	06234	Jasmine Patel	Programmer	8/10/2005	Basic Web Design	8/10/2007	Advanced C# programming	8/10/2007	Basic DB manipulation	1/29/2012
4	03373	Franklin Johnson, Jr.	Purchasing Agent	3/15/2002	Advanced Spreadsheets	6/20/2011				
5	13567	Almond, Robert	Analyst	9/30/2012	Basic Process Modeling	9/30/2014	Basic Database Design	5/23/2015		
6	10282	Richardson, Amanda	Clerk	4/11/2011						
7	09382	Susan Mathis	Database Programmer	8/2/2010	Basic DB Design	8/2/2012	Basic Database Manipulation	8/2/2012	Advanced DB Manipulation	5/1/2013
8	14311	Duong, Lee	Programmer	9/1/2014	Basic Web Design	9/1/2016				
9					Master Database Programming					
10					Basic Spreadsheets					
11	09002	Wade Gaither	Clerk	5/20/2010	Advanced Spreadsheets	5/16/2013	Basic Web Design	5/16/2013		
12	13383	Raymond F. Matthews	Programmer	3/12/2012	Basic C# Programming	3/12/2014				
13	09283	Chavez, Juan	Clerk	7/4/2010						
14	04893	Patricia Richards	DBA	6/11/2004	Advanced Database Management	6/11/2006	Advanced Database Manipulation	9/20/2012		
15	13932	Lee, Megan	Programmer	9/29/2013						

• It would be difficult, if not impossible, to produce an alphabetical listing of employees based on their last names.

• To determine how many employees are certified in Basic Database Manipulation, you would need a program that counts the number of those certifications recorded in Skill1 and places it in a variable. Then the count of those certifications in Skill2 could be calculated and added to the variable. Finally, the count of those certifications in Skill3 could be calculated and added to the variable to produce the total.

• If you redundantly store the name of a skill with each employee who is certified in that skill, you run the risk of spelling the name differently for different employees. For example, the skill Basic Database Manipulation is also entered as Basic DB Manipulation for at least one employee, which makes it difficult to get an accurate count of employees who have the certification.

Improved Database Design:

Database name: Ch01_Text

Table name: EMPLOYEE

Employee_ID	Employee_FName	Employee_LName	Employee_HireDate	Employee_Title
02345	Johnny	Jones	2/14/1995	DBA
03373	Franklin	Johnson	3/15/2002	Purchasing Agent
04893	Patricia	Richards	6/11/2004	DBA
06234	Jasmine	Patel	8/10/2005	Programmer
08273	Marco	Bienz	7/28/2006	Analyst
09002	Ben	Joiner	5/20/2010	Clerk
09283	Juan	Chavez	7/4/2010	Clerk
09382	Jessica	Johnson	8/2/2010	Database Programmer
10282	Amanda	Richardson	4/11/2011	Clerk
13383	Raymond	Matthews	3/12/2012	Programmer
13567	Robert	Almond	9/30/2012	Analyst
13932	Megan	Lee	9/29/2013	Programmer
14311	Lee	Duong	9/1/2014	Programmer

Table name: CERTIFIED

Employee_ID	Skill_ID	Certified_Date
02345	100	2/14/2002
02345	110	8/9/2003
02345	180	2/14/2005
03373	120	6/20/2011
04893	180	6/11/2006
04893	220	9/20/2012
06234	110	8/10/2007
06234	200	8/10/2007
06234	210	1/29/2012
08273	110	3/8/2009
08273	190	8/19/2012
09002	110	5/16/2013
09002	120	5/16/2013
09382	140	8/2/2012
09382	210	8/2/2012
09382	220	5/1/2013
13383	170	3/12/2014
13567	130	9/30/2014
13567	140	5/23/2015
14311	110	9/1/2016

Table name: SKILL

Skill_ID	Skill_Name	Skill_Description
100	Basic Database Management	Create and manage database user accounts.
110	Basic Web Design	Create and maintain HTML and CSS documents.
120	Advanced Spreadsheets	Use of advanced functions, user-defined functions, and macroing.
130	Basic Process Modeling	Create core business process models using standard libraries.
140	Basic Database Design	Create simple data models.
150	Master Database Programming	Create integrated trigger and procedure packages for a distributed environment.
160	Basic Spreadsheets	Create single tab worksheets with basic formulas
170	Basic C# Programming	Create single-tier data aware modules.
180	Advanced Database Management	Manage Database Server Clusters.
190	Advance Process Modeling	Evaluate and Redesign cross-functional internal and external business processes.
200	Advanced C# Programming	Create multi-tier applications using multi-threading
210	Basic Database Manipulation	Create simple data retrieval and manipulation statements in SQL.
220	Advanced Database Manipulation	Use of advanced data manipulation methods for multi-table inserts, set operations, and correlated subqueries.

Design has been improved by decomposing the data into three related tables. These tables contain all of the same data from before, but the tables are structured so that you can easily manipulate the data to view it in different ways and answer simple questions. We can now:

• Produce an alphabetical listing of employees by last name: SELECT * FROM EMPLOYEE ORDER BY EMPLOYEE_LNAME;

• Determine how many employees are certified in Basic Database Manipulation: SELECT Count(*) FROM SKILL JOIN CERTIFIED ON SKILL.SKILL_ID = CERTIFIED.SKILL_ID WHERE SKILL_NAME = 'Basic Database Manipulation';

Note that because each skill name is stored only once, the names cannot be spelled or abbreviated differently for different employees. Also, the additional certification of an employee with a fourth or fifth skill does not require changes to the structure of the tables.

DATA MODELS AND RELATIONSHIPS

The basic building blocks of all data models are entities, attributes, relationships, and constraints. An entity is a person, place, thing, or event about which data will be collected and stored.

- An entity represents a particular type of object in the real world, which means an entity is "distinguishable"—that is, each entity occurrence is unique and distinct. For example, a CUSTOMER entity would have many distinguishable customer occurrences, such as John Smith, Pedro Dinamita, and Tom Strickland. Entities may be physical objects, such as customers or products, but entities may also be abstractions, such as flight routes or musical concerts.

- An attribute is a characteristic of an entity. For example, a CUSTOMER entity would be described by attributes such as customer last name, customer first name, customer phone number, customer address, and customer credit limit. Attributes are the equivalent of fields in file systems.

- A relationship describes an association among entities. For example, a relationship exists between customers and agents that can be described as follows: an agent can serve many customers, and each customer may be served by one agent. Data models use three types of relationships: one-to-many, many-to-many, and one-to-one. Database designers usually use the shorthand notations 1:M or 1..*, M:N or *..*, and 1:1 or 1..1, respectively. (Although the M:N notation is a standard label for the many-to-many relationship, the label M:M may also be used.) The following examples illustrate the distinctions among the three relationships.

Relationships

- One-to-many (1:M or 1..*) relationship. A painter creates many different paintings, but each is painted by only one painter. Thus, the painter (the "one") is related to the paintings (the "many"). Therefore, database designers label the relationship "PAINTER paints PAINTING" as 1:M. Note that entity names are often capitalized as a convention, so they are easily identified. Similarly, a customer (the "one") may generate many invoices, but each invoice (the "many") is generated by only a single customer. The "CUSTOMER generates INVOICE" relationship would also be labelled 1:M.

- Many-to-many (M:N or *..*) relationship. An employee may learn many job skills, and each job skill may be learned by many employees. Database designers label the relationship "EMPLOYEE learns SKILL" as M:N. Similarly, a student can take many classes and each class can be taken by many students, thus yielding the M:N label for the relationship expressed by "STUDENT takes CLASS."

- One-to-one (1:1 or 1..1) relationship. A retail company's management structure may require that each of its stores be managed by a single employee. In turn, each store manager, who is anemployee, manages only a single store. Therefore, the relationship "EMPLOYEE manages STORE" is labelled 1:1.

Linking Relational Tables

Tables for Agents and their customers

Table name: AGENT (first six attributes) **Database name: Ch02_InsureCo**

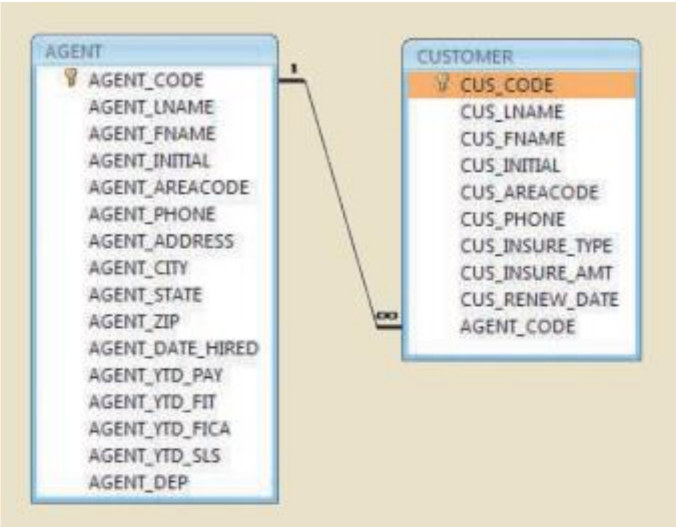
AGENT_CODE	AGENT_LNAME	AGENT_FNAME	AGENT_INITIAL	AGENT_AREACODE	AGENT_PHONE
501	Alby	Alex	B	713	228-1249
502	Hahn	Leah	F	615	882-1244
503	Okon	John	T	615	123-5589

Link through AGENT_CODE

Table name: CUSTOMER

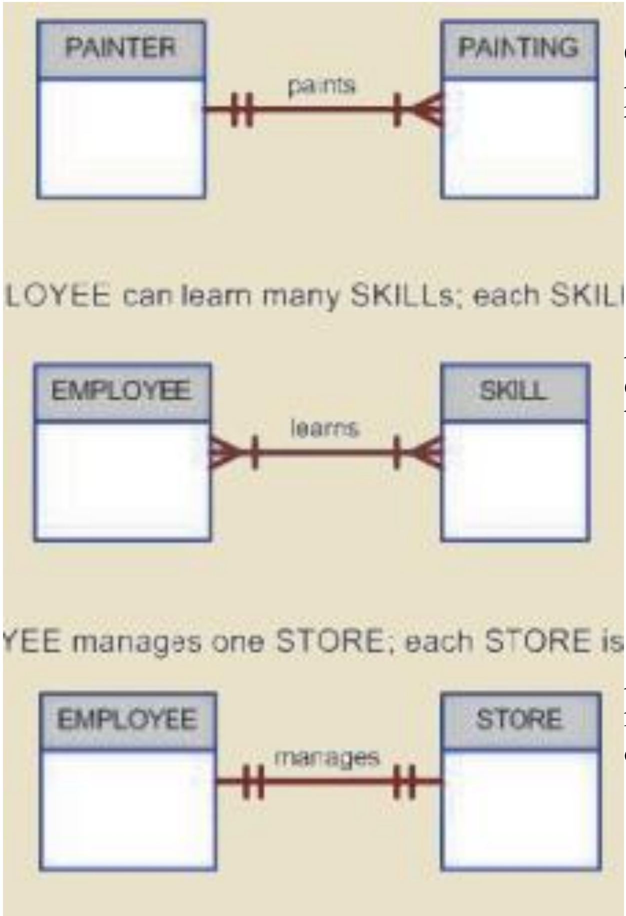
CUS_CODE	CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_AREACODE	CUS_PHONE	CUS_INSURE_TYPE	CUS_INSURE_AMT	CUS_RENEW_DATE	AGENT_CODE
10010	Ramas	Alfred	A	615	844-2573	T1	100.00	05-Apr-2016	502
10011	Dunne	Leona	K	713	894-1238	T1	250.00	16-Jun-2016	501
10012	Smith	Kathy	W	615	894-2285	S2	150.00	29-Jan-2017	502
10013	Olowski	Paul	F	615	894-2180	S1	300.00	14-Oct-2016	502
10014	Orlando	Myron		615	222-1672	T1	100.00	28-Dec-2017	501
10015	O'Brian	Amy	B	713	442-3381	T2	850.00	22-Sep-2016	503
10016	Brown	James	G	615	297-1228	S1	120.00	25-Mar-2017	502
10017	Williams	George		615	290-2556	S1	250.00	17-Jul-2016	503
10018	Farriss	Anne	G	713	382-7185	T2	100.00	03-Dec-2016	501
10019	Smith	Olette	K	615	297-3809	S2	500.00	14-Mar-2017	503

Relational Diagram



The relational diagram shows the connecting fields (in this case, AGENT_CODE) and the relationship type (1:M). employs the infinity symbol (∞) to indicate the "many" side. In this example, the CUSTOMER represents the "many" side because an AGENT can have many CUSTOMERs. The AGENT represents the "1" side because each CUSTOMER has only one AGENT.

ENTITY RELATIONSHIP MODELLING (ERM)



One to Many(1:M) Relationship: A Painter can paint many Paintings, each Painting is painted by one Painter.

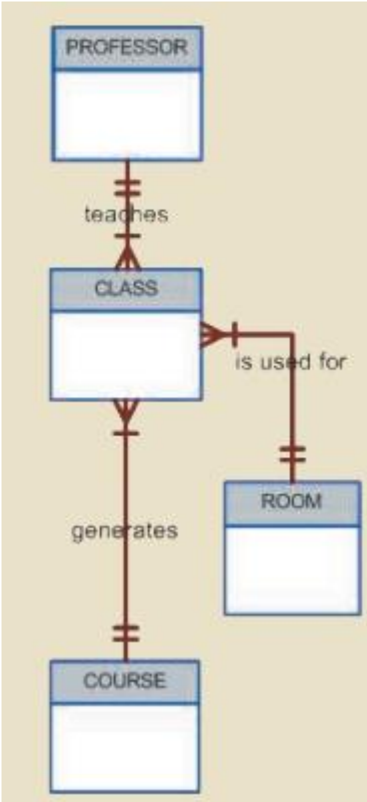
A Many to Many (M:N) relationship: An employee can learn many skills, each skill can be learned by many employees.

A one to one relationship: An employee manages one store, each store is managed by one employee.

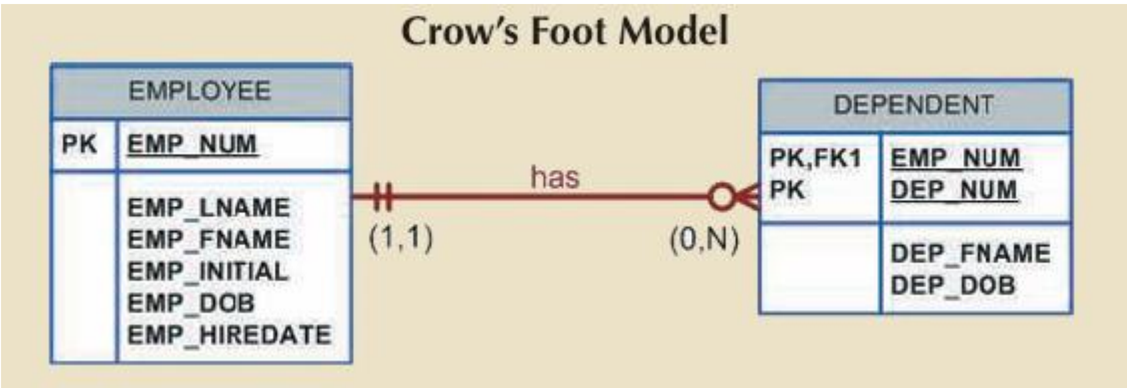
A professor teaches many classes.

Each room is used by many classes.

Each course generates many classes.



WEAK ENTITY RELATIONSHIP



The above is an example of a weak entity relationship.

1. The entity is existence-dependent; it cannot exist without the entity with which it has a relationship.

2. The entity has a primary key that is partially or totally derived from the parent entity in the relationship.

For example, a company insurance policy insures an employee and any dependents. For the purpose of describing an insurance policy, an EMPLOYEE might or might not have a DEPENDENT, but the DEPENDENT must be associated with an EMPLOYEE. Moreover, the DEPENDENT cannot exist without the EMPLOYEE; that is, a person cannot get insurance coverage as a dependent unless the person is a dependent of an employee. DEPENDENT is the weak entity in the relationship "EMPLOYEE has DEPENDENT."

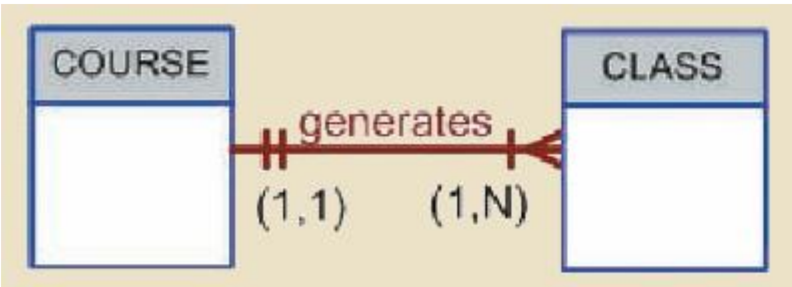
Table name: EMPLOYEE **Database nan**

EMP_NUM	EMP_LNAME	EMP_FNAME	EMP_INITIAL	EMP_DOB	EMP_HIREDATE
1001	Callifante	Jeanine	J	12-Mar-64	25-May-97
1002	Smithson	William	K	23-Nov-70	28-May-97
1003	Washington	Herman	H	15-Aug-68	28-May-97
1004	Chen	Lydia	B	23-Mar-74	15-Oct-98
1005	Johnson	Melanie		28-Sep-66	20-Dec-98
1006	Ortega	Jorge	G	12-Jul-79	05-Jan-02
1007	O'Donnell	Peter	D	10-Jun-71	23-Jun-02
1008	Brzenski	Barbara	A	12-Feb-70	01-Nov-03

Table name: DEPENDENT

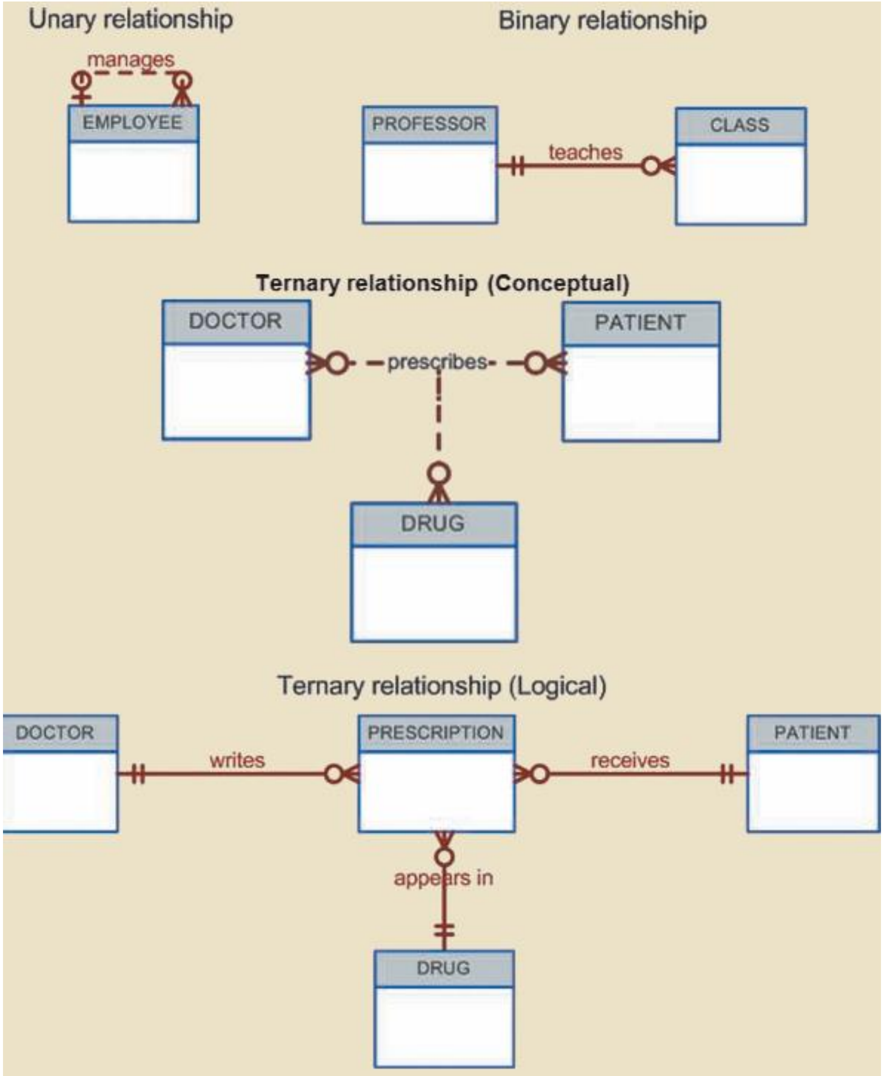
EMP_NUM	DEP_NUM	DEP_FNAME	DEP_DOB
1001	1	Annelise	05-Dec-97
1001	2	Jorge	30-Sep-02
1003	1	Suzanne	25-Jan-04
1006	1	Carlos	25-May-01
1008	1	Michael	19-Feb-95
1008	2	George	27-Jun-98
1008	3	Katherine	18-Aug-03

STRONG / MANDATORY ENTITY RELATIONSHIP

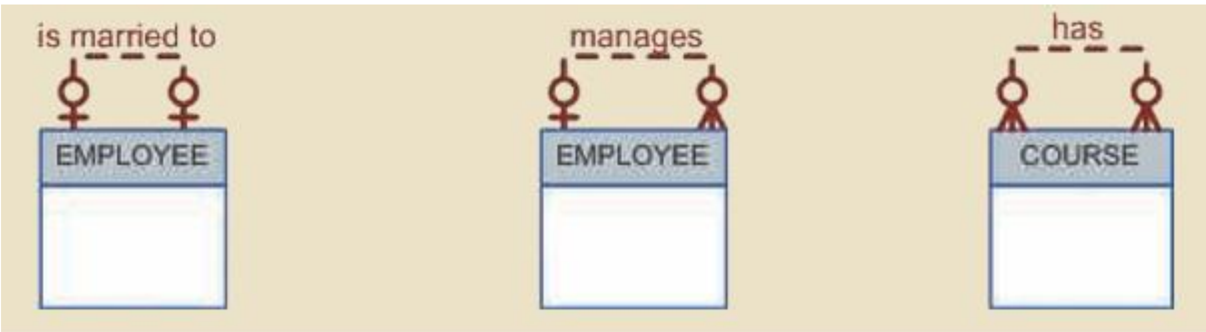


the “COURSE generates CLASS” relationship, it is easy to see that a CLASS cannot exist without a COURSE. Therefore, you can conclude that the COURSE entity is mandatory in the relationship.

NUMBERS OF RELATIONSHIPS



RECURSIVE RELATIONSHIP



THE RELATIONAL DATABASE MODEL

Table name: STUDENT **Database name: Ch03_TinyCollege**

STU_NUM	STU_LNAME	STU_FNAME	STU_INIT	STU_DOB	STU_HRS	STU_CLASS	STU_GPA	STU_TRANSFER	DEPT_CODE	STU_PHONE	PROF_NUM
321452	Bowser	William	C	12-Feb-1985	42	So	2.84	No	BIOL	2134	205
324257	Smithson	Anne	K	15-Nov-1991	81	Jr	3.27	Yes	CIS	2256	222
324258	Brewer	Juliette		23-Aug-1979	36	So	2.26	Yes	ACCT	2256	228
324269	Oblonski	Walter	H	16-Sep-1986	66	Jr	3.09	No	CIS	2114	222
324273	Smith	John	D	30-Dec-1968	102	Sr	2.11	Yes	ENGL	2231	199
324274	Katinga	Raphael	P	21-Oct-1989	114	Sr	3.15	No	ACCT	2267	228
324291	Robertson	Gerald	T	08-Apr-1983	120	Sr	3.87	No	EDU	2267	311
324299	Smith	John	B	30-Nov-1996	15	Fr	2.92	No	ACCT	2315	230

STU_NUM = Student number
STU_LNAME = Student last name
STU_FNAME = Student first name
STU_INIT = Student middle initial
STU_DOB = Student date of birth
STU_HRS = Credit hours earned
STU_CLASS = Student classification
STU_GPA = Grade point average
STU_TRANSFER = Student transferred from another institution
DEPT_CODE = Department code
STU_PHONE = 4-digit campus phone extension
PROF_NUM = Number of the professor who is the student's advisor

- The STUDENT table is perceived to be a two-dimensional structure composed of 8 rows (tuples) and 12 columns (attributes).

- Each row in the STUDENT table describes a single entity occurrence within the entity set. (The entity set is represented by the STUDENT table.) For example, row 4 describes a student named Walter H. Oblonski. Given the table contents, the STUDENT entity set includes eight distinct entities (rows), or students.

- Each column represents an attribute, and each column has a distinct name.

- All of the values in a column match the attribute's characteristics. For example, the grade point average (STU_GPA) column contains only STU_GPA entries for each of the table rows. Data must be classified according to its format and function. Although various DBMSs can support different data types, most support at least the following:

  a. Numeric. You can use numeric data to perform meaningful arithmetic procedures. For example, STU_HRS and STU_GPA are numeric attributes.
  b. Character. Character data, also known as text data or string data, can contain any character or symbol not intended for mathematical manipulation. STU_CLASS and STU_PHONE are examples of character attributes.
  c. Date. Date attributes contain calendar dates stored in a special format known as the Julian date format. STU_DOB is a date attribute.
  d. Logical. Logical data can only have true or false (yes or no) values. The STU_TRANSFER attribute uses a logical data format.

- The column's range of permissible values is known as its domain. Because the STU_GPA values are limited to the range 0–4, inclusive, the domain is [0,4].

- The order of rows and columns is immaterial to the user.

KEYS – PRIMARY KEY

In the relational model, keys are important because they are used to ensure that each row in a table is uniquely identifiable. They are also used to establish relationships among tables and to ensure the integrity of the data. A key consists of one or more attributes that determine other attributes. For example, an invoice number identifies all of the invoice attributes, such as the invoice date and the customer name.

The role of a key is based on the concept of determination. Determination is the state in which knowing the value of one attribute makes it possible to determine the value of another.

If you consider what the attributes of the STUDENT table actually represent, you will see a relationship among the attributes. If you are given a value for STU_NUM, then you can determine the value for STU_LNAME because one and only one value of STU_LNAME is associated with any given value of STU_NUM. A specific terminology and notation is used to describe relationships based on determination. The relationship is called functional dependence, which means that the value of one or more attributes determines the value of one or more other attributes. The standard notation for representing the relationship between STU_NUM and STU_LNAME is as follows:
STU_NUM → STU_LNAME

In this functional dependency, the attribute whose value determines another is called the determinant or the key. The attribute whose value is determined by the other attribute is called the dependent.

Using this terminology, it would be correct to say that STU_ NUM is the determinant and STU_LNAME is the dependent. STU_NUM functionally determines STU_LNAME, and STU_LNAME is functionally dependent on STU_NUM.

STU_NUM → (STU_LNAME, STU_FNAME, STU_GPA)

FOREIGN KEYS

Table name: PRODUCT **Database name:**
Primary key: PROD_CODE
Foreign key: VEND_CODE

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

link

Table name: VENDOR
Primary key: VEND_CODE
Foreign key: none

VEND_CODE	VEND_CONTACT	VEND_AREACODE	VEND_PHONE
230	Shelly K. Smithson	608	555-1234
231	James Johnson	615	123-4536
232	Annelise Crystall	608	224-2134
233	Candice Wallace	904	342-6567
234	Arthur Jones	615	123-3324
235	Henry Ortozo	615	899-3425

Just as the primary key has a role in ensuring the integrity of the database, so does the foreign key. Foreign keys are used to ensure referential integrity, the condition in which every reference to an entity instance by another entity instance is valid. In other words, every foreign key entry must either be null or a valid value in the primary key of the related table. Note that the PRODUCT table has referential integrity because every entry in VEND_CODE in the PRODUCT table is either null or a valid value in VEND_ CODE in the VENDOR table. Every vendor referred to by a row in the PRODUCT table is a valid vendor.

Entity integrity. The CUSTOMER table's primary key is CUS_CODE. The CUSTOMER primary key column has no null entries, and all entries are unique. Similarly, the AGENT table's prima

Referential integrity. The CUSTOMER table contains a foreign key, AGENT_CODE, that links entries in the CUSTOMER table to the AGENT table. The CUS_CODE row identified by the (primary key) number 10013 contains a null entry in its AGENT_ CODE foreign key because Paul F. Olowski does not yet have a sales representative assigned to him. The remaining AGENT_CODE entries in the CUSTOMER table all match the AGENT_CODE entries in the AGENT table.

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

CUS_CODE	CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_RENEW_DATE	AGENT_CODE
10010	Ramas	Alfred	A	05-Apr-2016	502
10011	Dunne	Leona	K	16-Jun-2016	501
10012	Smith	Kathy	W	29-Jan-2017	502
10013	Olowski	Paul	F	14-Oct-2016	
10014	Orlando	Myron		28-Dec-2016	501
10015	O'Brian	Amy	B	22-Sep-2016	503
10016	Brown	James	G	25-Mar-2017	502
10017	Williams	George		17-Jul-2016	503
10018	Farriss	Anne	G	03-Dec-2016	501
10019	Smith	Olette	K	14-Mar-2017	503

Table name: AGENT (only five selected fields are shown)
Primary key: AGENT_CODE
Foreign key: none

AGENT_CODE	AGENT_AREACODE	AGENT_PHONE	AGENT_LNAME	AGENT_YTD_SLS
501	713	228-1249	Alby	132735.75
502	615	882-1244	Hahn	138967.35
503	615	123-5589	Okon	127093.45

RELATIONAL ALGEBRA – Manipulating Table Content

SELECT – SELECT can be used to list all rows of a specific criterion.

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

SELECT ALL yields

New 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

SELECT only PRICE less than $2.00 yields

P_CODE	P_DESCRIPT	PRICE
213345	9v battery	1.92
254467	100W bulb	1.47

SELECT only P_CODE = 311452 yields

P_CODE	P_DESCRIPT	PRICE
311452	Powerdrill	34.99

PROJECT – PROJECT will return only the attributes requested, in the order in which they are requested. In other words, PROJECT yields a vertical subset of a table. PROJECT will not limit the rows returned so all rows of the specified attributes will be included in the result.

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

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

UNION – combines all rows from two tables, excluding duplicate rows. To be used in the UNION, the tables must have the same attribute characteristics; in other words, the columns and domains must be compatible. When two or more tables share the same number of col umns, and when their corresponding columns share the same or compatible domains, they are said to be union-compatible.

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
123458	Box Fan	10.99

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
345679	Dishwasher	500

INTERSECT – yields only the rows that appear in both tables. As with UNION, the tables must be union-compatible to yield valid results. For example, you cannot use INTERSECT if one of the attributes is numeric and one is character-based.

STU_FNAME	STU_LNAME
George	Jones
Jane	Smith
Peter	Robinson
Franklin	Johnson
Martin	Lopez

INTERSECT

EMP_FNAME	EMP_LNAME
Franklin	Lopez
William	Turner
Franklin	Johnson
Susan	Rogers

yields

STU_FNAME	STU_LNAME
Franklin	Johnson

DIFFERENCE – yields all rows in one table that are not found in the other table; that is, it subtracts one table from the other. As with UNION, the tables must be union-compatible to yield valid results.

STU_FNAME	STU_LNAME
George	Jones
Jane	Smith
Peter	Robinson
Franklin	Johnson
Martin	Lopez

DIFFERENCE

EMP_FNAME	EMP_LNAME
Franklin	Lopez
William	Turner
Franklin	Johnson
Susan	Rogers

yields

STU_FNAME	STU_LNAME
George	Jones
Jane	Smith
Peter	Robinson
Martin	Lopez

PRODUCT – yields all possible pairs of rows from two tables—also known as the Cartesian product. Therefore, if one table has 6 rows and the other table has 3 rows, the PRODUCT yields a list composed of 6 × 3 = 18rows.

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

JOIN – allows information to be intelligently combined from two or more tables. JOIN is the real power behind the relational database, allowing the use of independent tables linked by common attributes.

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

NATURAL JOIN – links tables by selecting only the rows with common values in their common attribute(s). SELECT is normally preferred over Natural Join.

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

LEFT OUTER JOIN – yields all of the rows in the CUSTOMER table, including those that do not have a matching value in the AGENT table.

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
1542311	Smithson	37134	421		

RIGHT OUTER JOIN – yields all of the rows in the AGENT table, including those that do not have matching values in the CUSTOMER table.

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
				333	9041234445

DIVIDE – operator is used to answer questions about one set of data being associated with all values of data in another set of data. The DIVIDE operation uses one 2-column table (Table 1) as the dividend and one single-column table (Table 2) as the divisor.

P_CODE	CUS_CODE
123456	10400
123456	11501
123456	10030
123456	12550
234567	12350
234567	10040
234567	10600
234567	10030
234567	12550
345678	10400
345678	11630
345678	12550
456789	11630
567890	10600
567890	10030
567890	12550
678901	11500
678901	10400
678901	11630

DIVIDE

P_CODE
123456
234567
567890

yields

CUS_CODE
10030
12550

RELATIONSHIPS AND KEYS IN THE RELATIONAL DATABASE MODEL

1:M Relationship

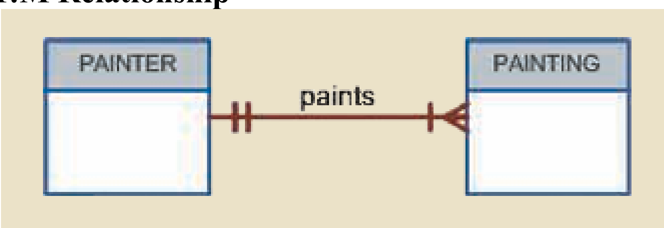


Table name: PAINTER
Primary key: PAINTER_NUM
Foreign key: none

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

- Each painting was created by one and only one painter, but each painter could have created many paintings. For Example Painter 123 (Georgette P. Ross) has three works stored in the PAINTING table.

- There is only one row in the PAINTER table for any given row in the PAINTING table, but there may be many rows in the PAINTING table for any given row in the PAINTER table

1:M RELATIONSHIP IN RELATIONAL DATABASES

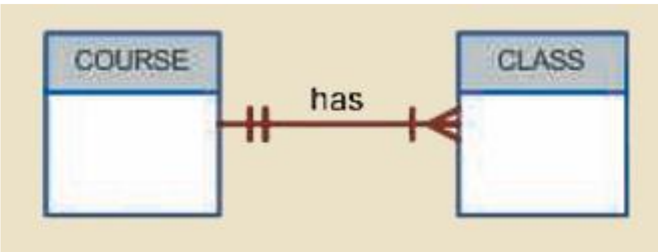


Table name: COURSE
Primary key: CRS_CODE
Foreign key: none

Database name: Ch03_TinyCollege

CRS_CODE	DEPT_CODE	CRS_DESCRIPTION	CRS_CREDIT
ACCT-211	ACCT	Accounting I	3
ACCT-212	ACCT	Accounting II	3
CIS-220	CIS	Intro. to Microcomputing	3
CIS-420	CIS	Database Design and Implementation	4
QM-261	CIS	Intro. to Statistics	3
QM-362	CIS	Statistical Applications	4

Table name: CLASS
Primary key: CLASS_CODE
Foreign key: CRS_CODE

CLASS_CODE	CRS_CODE	CLASS_SECTION	CLASS_TIME	CLASS_ROOM	PROF_NUM
10012	ACCT-211	1	MWF 8:00-8:50 a.m.	BUS311	105
10013	ACCT-211	2	MWF 9:00-9:50 a.m.	BUS200	105
10014	ACCT-211	3	TTh 2:30-3:45 p.m.	BUS252	342
10015	ACCT-212	1	MWF 10:00-10:50 a.m.	BUS311	301
10016	ACCT-212	2	Th 6:00-8:40 p.m.	BUS252	301
10017	CIS-220	1	MWF 9:00-9:50 a.m.	KLR209	228
10018	CIS-220	2	MWF 9:00-9:50 a.m.	KLR211	114
10019	CIS-220	3	MWF 10:00-10:50 a.m.	KLR209	228
10020	CIS-420	1	W 6:00-8:40 p.m.	KLR209	162
10021	QM-261	1	MWF 8:00-8:50 a.m.	KLR200	114
10022	QM-261	2	TTh 1:00-2:15 p.m.	KLR200	114
10023	QM-362	1	MWF 11:00-11:50 a.m.	KLR200	162
10024	QM-362	2	TTh 2:30-3:45 p.m.	KLR200	162

Note that the PAINTER table's primary key, PAINTER_ NUM, is included in the PAINTING table as a foreign key. Similarly, the COURSE table's primary key, CRS_CODE, is included in the CLASS table as a foreign key.

THE 1:1 RELATIONSHIP IN RELATIONAL DATABASES

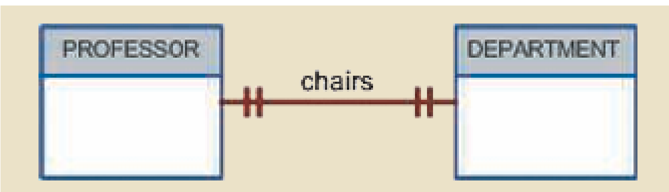


Table name: PROFESSOR
Primary key: EMP_NUM
Foreign key: DEPT_CODE

Database name: Ch03_TinyCollege

EMP_NUM	DEPT_CODE	PROF_OFFICE	PROF_EXTENSION	PROF_HIGH_DEGREE
103	HIST	DRE 156	6783	Ph.D.
104	ENG	DRE 102	5561	MA
105	ACCT	KLR 229D	8665	Ph.D.
106	MKT/MGT	KLR 126	3899	Ph.D.
110	BIOL	AAK 160	3412	Ph.D.
114	ACCT	KLR 211	4436	Ph.D.
155	MATH	AAK 201	4440	Ph.D.
160	ENG	DRE 102	2248	Ph.D.
162	CIS	KLR 203E	2359	Ph.D.
191	MKT/MGT	KLR 409B	4016	DBA
195	PSYCH	AAK 297	3550	Ph.D.
209	CIS	KLR 333	3421	Ph.D.
228	CIS	KLR 300	3000	Ph.D.
297	MATH	AAK 194	1145	Ph.D.
299	ECON/FIN	KLR 284	2851	Ph.D.
301	ACCT	KLR 244	4683	Ph.D.
335	ENG	DRE 208	2000	Ph.D.
342	SOC	BBG 208	5514	Ph.D.
387	BIOL	AAK 230	8665	Ph.D.
401	HIST	DRE 156	6783	MA
425	ECON/FIN	KLR 284	2851	MBA
435	ART	BBG 185	2278	Ph.D.

The 1:M DEPARTMENT employs PROFESSOR relationship is implemented through the placement of the DEPT_CODE foreign key in the PROFESSOR table.

The 1:1 PROFESSOR chairs DEPARTMENT relationship is implemented through the placement of the EMP_NUM foreign key in the DEPARTMENT table.

Table name: DEPARTMENT
Primary key: DEPT_CODE
Foreign key: EMP_NUM

DEPT_CODE	DEPT_NAME	SCHOOL_CODE	EMP_NUM	DEPT_ADDRESS	DEPT_EXTENSION
ACCT	Accounting	BUS	114	KLR 211, Box 52	3119
ART	Fine Arts	A&SCI	435	BBG 185, Box 128	2278
BIOL	Biology	A&SCI	387	AAK 230, Box 415	4117
CIS	Computer Info. Systems	BUS	209	KLR 333, Box 56	3245
ECON/FIN	Economics/Finance	BUS	299	KLR 284, Box 63	3126
ENG	English	A&SCI	160	DRE 102, Box 223	1004
HIST	History	A&SCI	103	DRE 156, Box 284	1867
MATH	Mathematics	A&SCI	297	AAK 194, Box 422	4234
MKT/MGT	Marketing/Management	BUS	106	KLR 126, Box 55	3342
PSYCH	Psychology	A&SCI	195	AAK 297, Box 438	4110
SOC	Sociology	A&SCI	342	BBG 208, Box 132	2008

- Each professor is a college employee. Therefore, the professor identification is through the EMP_NUM. (However, note that not all employees are professors— there's another optional relationship.)

- The 1:1 "PROFESSOR chairs DEPARTMENT" relationship is implemented by having the EMP_NUM foreign key in the DEPARTMENT table. Note that the 1:1 relationship is treated as a special case of the 1:M relationship in which the "many" side is restricted to a single occurrence. In this case, DEPARTMENT contains the EMP_NUM as a foreign key to indicate that it is the department that has a chair.

THE M:N RELATIONSHIP IN RELATIONAL DATABASES

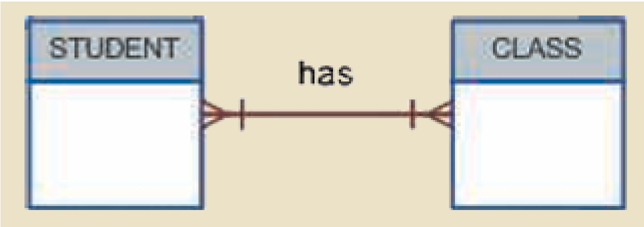


Table name: STUDENT
Primary key: STU_NUM
Foreign key: none

Database name: Ch03_Co

STU_NUM	STU_LNAME	CLASS_CODE
321452	Bowser	10014
321452	Bowser	10018
321452	Bowser	10021
324257	Smithson	10014
324257	Smithson	10018
324257	Smithson	10021

Table name: CLASS
Primary key: CLASS_CODE
Foreign key: STU_NUM

CLASS_CODE	STU_NUM	CRS_CODE	CLASS_SECTION	CLASS_TIME	CLASS_ROOM	PROF_NUM
10014	321452	ACCT-211	3	TTh 2:30-3:45 p.m.	BUS252	342
10014	324257	ACCT-211	3	TTh 2:30-3:45 p.m.	BUS252	342
10018	321452	CIS-220	2	MWF 9:00-9:50 a.m.	KLR211	114
10018	324257	CIS-220	2	MWF 9:00-9:50 a.m.	KLR211	114
10021	321452	QM-261	1	MWF 8:00-8:50 a.m.	KLR200	114
10021	324257	QM-261	1	MWF 8:00-8:50 a.m.	KLR200	114

- Each CLASS can have many STUDENTs, and each STUDENT can take many CLASSes.
- There can be many rows in the CLASS table for any given row in the STUDENT table, and there can be many rows in the STUDENT table for any given row in the CLASS table.
- HOWEVER, this example creates a lot of repetition(redundancies) and therefore inefficiency, complexity and potential errors. The tables create many redundancies. For example, note that the STU_NUM values occur many times in the STUDENT table. In a real-world situation, additional stu dent attributes such as address, classification, major, and home phone would also be contained in the STUDENT table, and each of those attribute values would be repeated in each of the records shown here. Similarly, the CLASS table contains much

CONVERTING A M:N RELATIONSHIP INTO TWO 1:M RELATIONSHIPS

Table name: STUDENT
Primary key: STU_NUM
Foreign key: none

Database name: Ch03

STU_NUM	STU_LNAME
321452	Bowser
324257	Smithson

Table name: ENROLL
Primary key: CLASS_CODE + STU_NUM
Foreign key: CLASS_CODE, STU_NUM

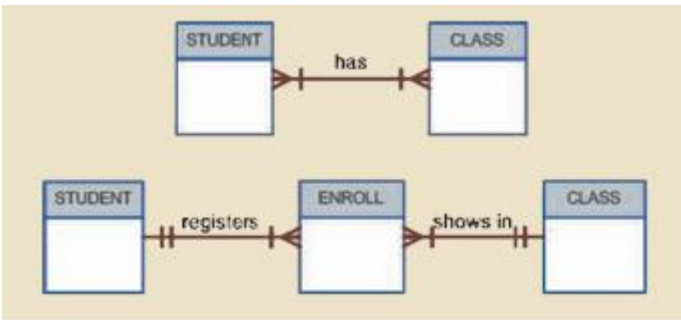
CLASS_CODE	STU_NUM	ENROLL_GRADE
10014	321452	C
10014	324257	B
10018	321452	A
10018	324257	B
10021	321452	C
10021	324257	C

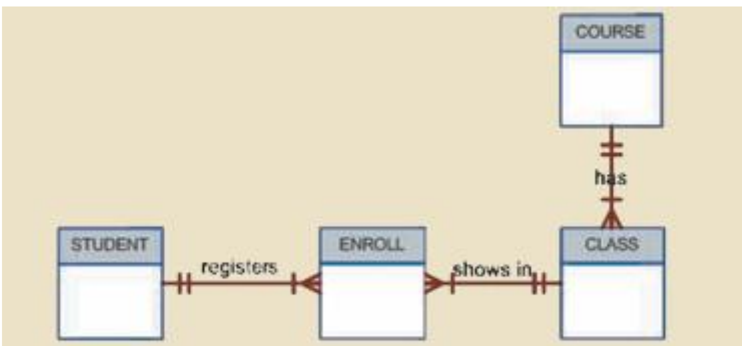
Table name: CLASS
Primary key: CLASS_CODE
Foreign key: CRS_CODE

CLASS_CODE	CRS_CODE	CLASS_SECTION	CLASS_TIME	CLASS_ROOM	PROF_NUM
10014	ACCT-211	3	TTh 2:30-3:45 p.m.	BUS252	342
10018	CIS-220	2	MWF 9:00-9:50 a.m.	KLR211	114
10021	QM-261	1	MWF 8:00-8:50 a.m.	KLR200	114

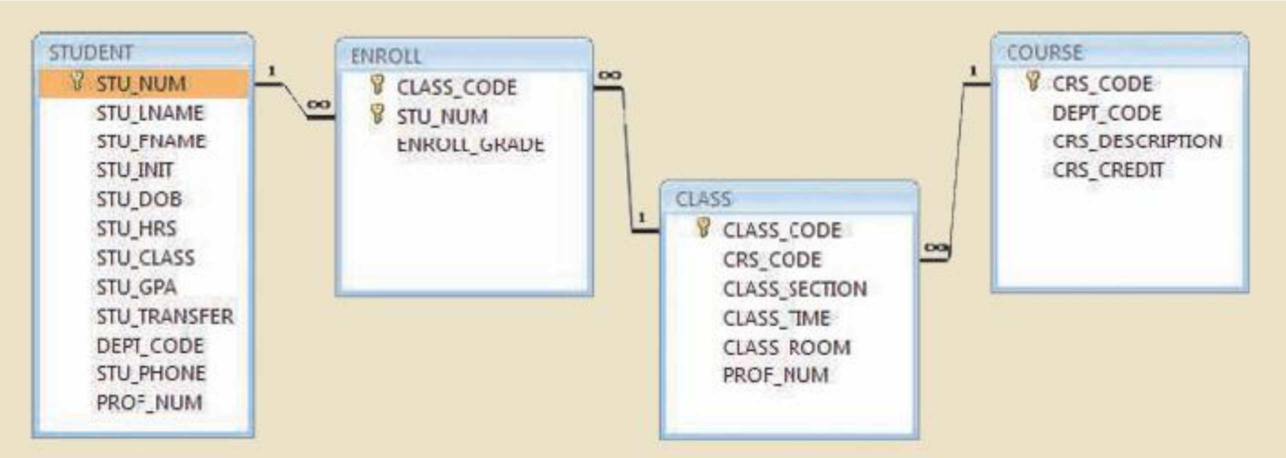
The ENROLL table links two tables, STUDENT and CLASS, it is also called a linking table. In other words, a linking table is the implementation of a composite entity.

ERMs



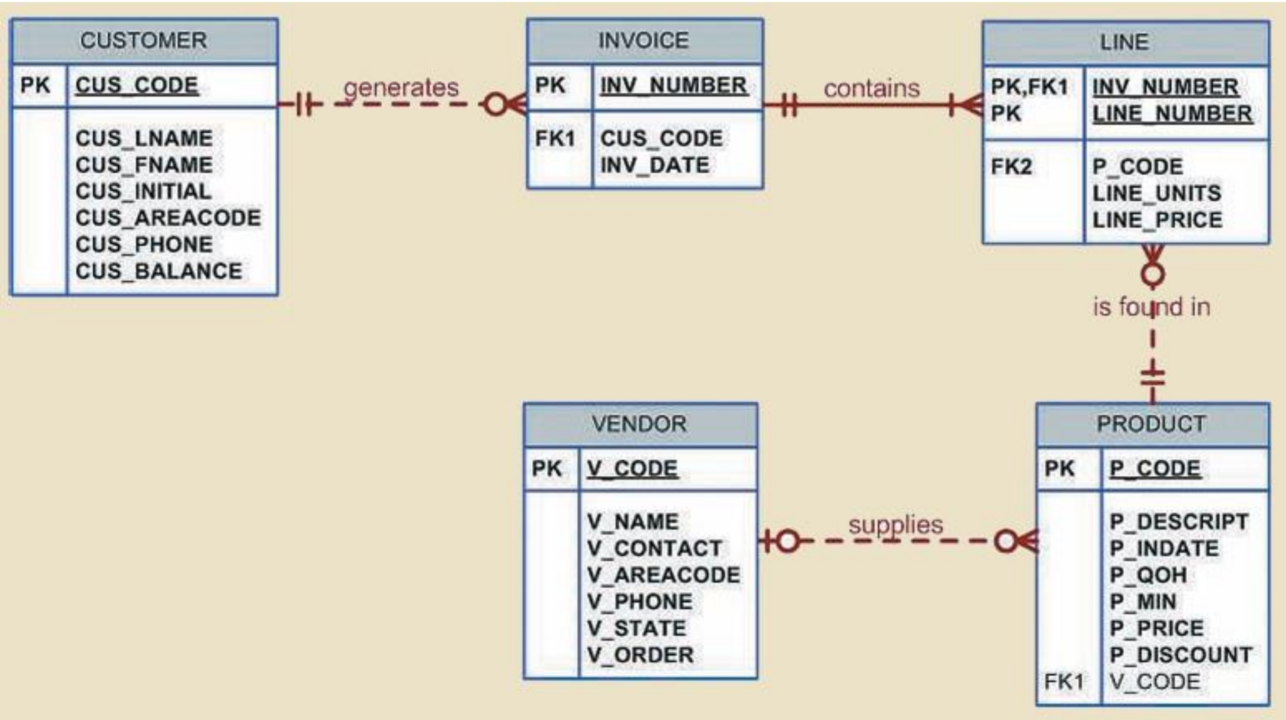


RELATIONAL DIAGRAM



STRUCTURED QUERY LANGUAGE (SQL)

Database / ERM example used in this section:



- A vendor may supply many products. Some vendors do not yet supply products. For example, a vendor list may include potential vendors.
- If a product is vendor-supplied, it is supplied by only a single vendor.

- Some products are not supplied by a vendor. For example, some products may be produced in-house or bought on the open market.

Table name: VENDOR **Databas**

V_CODE	V_NAME	V_CONTACT	V_AREACODE	V_PHONE	V_STATE	V_ORDER
21225	Bryson, Inc.	Smithson	615	223-3234	TN	Y
21226	SuperLoo, Inc.	Flushing	904	215-8995	FL	N
21231	D&E Supply	Singh	615	228-3245	TN	Y
21344	Gomez Bros.	Ortega	615	889-2546	KY	N
22567	Dome Supply	Smith	901	678-1419	GA	N
23119	Randsets Ltd.	Anderson	901	678-3998	GA	Y
24004	Brackman Bros.	Browning	615	228-1410	TN	N
24288	ORDVA, Inc.	Hakford	615	898-1234	TN	Y
25443	B&K, Inc.	Smith	904	227-0093	FL	N
25501	Damal Supplies	Smythe	615	890-3529	TN	N
25595	Rubicon Systems	Orton	904	456-0092	FL	Y

Database Table for this section:

Table name: PRODUCT

P_CODE	P_DESCRIPT	P_INDATE	P_QOH	P_MIN	P_PRICE	P_DISCOUNT	V_CODE
11QER/31	Power painter, 15 psi., 3-nozzle	03-Nov-15	8	5	109.99	0.00	25595
13-Q2/P2	7.25-in. pwr. saw blade	13-Dec-15	32	15	14.99	0.05	21344
14-Q1/L3	9.00-in. pwr. saw blade	13-Nov-15	18	12	17.49	0.00	21344
1546-QQ2	Hrd. cloth, 1/4-in., 2x50	15-Jan-16	15	8	39.95	0.00	23119
1558-QW1	Hrd. cloth, 1/2-in., 3x50	15-Jan-16	23	5	43.99	0.00	23119
2232/QTY	B&D jigsaw, 12-in. blade	30-Dec-15	8	5	109.92	0.05	24288
2232/QWE	B&D jigsaw, 8-in. blade	24-Dec-15	6	5	99.87	0.05	24288
2238/QPD	B&D cordless drill, 1/2-in.	20-Jan-16	12	5	38.95	0.05	25595
23109-HB	Claw hammer	20-Jan-16	23	10	9.95	0.10	21225
23114-AA	Sledge hammer, 12 lb.	02-Jan-16	8	5	14.40	0.05	
54778-2T	Rat-tail file, 1/8-in. fine	15-Dec-15	43	20	4.99	0.00	21344
89-WRE-Q	Hicut chain saw, 16 in.	07-Feb-16	11	5	256.99	0.05	24288
PVC23DRT	PVC pipe, 3.5-in., 8-ft	20-Feb-16	188	75	5.87	0.00	
SM-18277	1.25-in. metal screw, 25	01-Mar-16	172	75	6.99	0.00	21225
SW-23116	2.5-in. wd. screw, 50	24-Feb-16	237	100	8.45	0.00	21231
WR3/TT3	Steel matting, 4'x8'x1/6", .5" mesh	17-Jan-16	18	5	119.95	0.10	25595

SELECT – Which attributes to report.
FROM – which table/entity are attributes in.
WHERE – Which records do I want reported.

CREATE TABLE

```
CREATE TABLE VENDOR (
V_CODE          INTEGER         NOT NULL        UNIQUE,
V_NAME          VARCHAR(35)     NOT NULL,
V_CONTACT       VARCHAR(25)     NOT NULL,
V_AREACODE      CHAR(3)         NOT NULL,
V_PHONE         CHAR(8)         NOT NULL,
V_STATE         CHAR(2)         NOT NULL,
V_ORDER         CHAR(1)         NOT NULL,

PRIMARY KEY (V_CODE));
```

```
CREATE TABLE PRODUCT (
P_CODE          VARCHAR(10)     NOT NULL        UNIQUE,
P_DESCRIPT      VARCHAR(35)     NOT NULL,
P_INDATE        DATE            NOT NULL,
P_QOH           SMALLINT        NOT NULL,
P_MIN           SMALLINT        NOT NULL,
P_PRICE         NUMBER(8,2)     NOT NULL,
P_DISCOUNT      NUMBER(5,2)     NOT NULL,
V_CODE          INTEGER,

PRIMARY KEY (P_CODE),
FOREIGN KEY (V_CODE) REFERENCES VENDOR ON UPDATE CASCADE);
```

INSERT

INSERT INTO VENDOR VALUES (21225,'Bryson, Inc.','Smithson','615','223-3234','TN','Y');

INSERT INTO VENDOR VALUES (21226,'Superloo, Inc.','Flushing','904','215-8995','FL','N');

INSERT INTO PRODUCT VALUES ('11QER/31','Power painter, 15 psi., 3-nozzle','03-Nov-15',8,5,109.99,0.00,25595);

INSERT INTO PRODUCT VALUES ('13-Q2/P2','7.25-in. pwr. saw blade','13-Dec-15',32,15,14.99, 0.05, 21344);

SELECT AND SELECT QUERY

SELECT * FROM PRODUCT;

P_CODE	P_DESCRIPT	P_INDATE	P_QOH	P_MIN	P_PRICE	P_DISCOUNT	V_CODE
11QER/31	Power painter, 15 psi., 3-nozzle	03-Nov-15	8	5	109.99	0.00	25595
13-Q2/P2	7.25-in. pwr. saw blade	13-Dec-15	32	15	14.99	0.05	21344
14-Q1/L3	9.00-in. pwr. saw blade	13-Nov-15	18	12	17.49	0.00	21344
1546-QQ2	Hrd. cloth, 1/4-in., 2x50	15-Jan-16	15	8	39.95	0.00	23119
1558-QW1	Hrd. cloth, 1/2-in., 3x50	15-Jan-16	23	5	43.99	0.00	23119
2232/QTY	B&D jigsaw, 12-in. blade	30-Dec-15	8	5	109.92	0.05	24288
2232/QWE	B&D jigsaw, 8-in. blade	24-Dec-15	6	5	99.87	0.05	24288
2238/QPD	B&D cordless drill, 1/2-in.	20-Jan-16	12	5	38.95	0.05	25595
23109-HB	Claw hammer	20-Jan-16	23	10	9.95	0.10	21225
23114-AA	Sledge hammer, 12 lb.	02-Jan-16	8	5	14.40	0.05	
54778-2T	Rat-tail file, 1/8-in. fine	15-Dec-15	43	20	4.99	0.00	21344
89-WRE-Q	Hicut chain saw, 16 in.	07-Feb-16	11	5	256.99	0.05	24288
PVC23DRT	PVC pipe, 3.5-in., 8-ft	20-Feb-16	188	75	5.87	0.00	
SM-18277	1.25-in. metal screw, 25	01-Mar-16	172	75	6.99	0.00	21225
SW-23116	2.5-in. wd. screw, 50	24-Feb-16	237	100	8.45	0.00	21231
WR3/TT3	Steel matting, 4'x8'x1/6", .5" mesh	17-Jan-16	18	5	119.95	0.10	25595

example written out:

SELECT P_CODE, P_DESCRIPT, P_INDATE, P_QOH, P_MIN, P_PRICE, P_DISCOUNT, V_CODE FROM PRODUCT;

Select Query:

```
SELECT    P_DESCRIPT, P_INDATE, P_PRICE, V_CODE
FROM      PRODUCT
WHERE     V_CODE = 21344;
```

P_DESCRIPT	P_INDATE	P_PRICE	V_CODE
7.25-in. pwr. saw blade	13-Dec-15	14.99	21344
9.00-in. pwr. saw blade	13-Nov-15	17.49	21344
Rat-tail file, 1/8-in. fine	15-Dec-15	4.99	21344

Not Equal = <>

```
SELECT P_DESCRIPT, P_QOH, P_PRICE, V_CODE
FROM PRODUCT
WHERE V_CODE <> 21344;
```

P_DESCRIPT	P_QOH	P_PRICE	V_CODE
Power painter, 15 psi., 3-nozzle	8	109.99	25595
Hrd. cloth, 1/4-in., 2x50	15	39.95	23119
Hrd. cloth, 1/2-in., 3x50	23	43.99	23119
B&D jigsaw, 12-in. blade	8	109.92	24288
B&D jigsaw, 8-in. blade	6	99.87	24288
B&D cordless drill, 1/2-in.	12	38.95	25595
Claw hammer	23	9.95	21225
Hicut chain saw, 16 in.	11	256.99	24288
1.25-in. metal screw, 25	172	6.99	21225
2.5-in. wd. screw, 50	237	8.45	21231
Steel matting, 4'x8'x1/6", .5" mesh	18	119.95	25595

```
SELECT P_DESCRIPT, P_QOH, P_MIN, P_PRICE
FROM PRODUCT
WHERE P_PRICE <= 10;
```

P_DESCRIPT	P_QOH	P_MIN	P_PRICE
Claw hammer	23	10	9.95
Rat-tail file, 1/8-in. fine	43	20	4.99
PVC pipe, 3.5-in., 8-ft	188	75	5.87
1.25-in. metal screw, 25	172	75	6.99
2.5-in. wd. screw, 50	237	100	8.45

Computation:

```
SELECT      P_DESCRIPT, P_QOH, P_PRICE,         P_QOH * P_PRICE
FROM        PRODUCT;
```

P_DESCRIPT	P_QOH	P_PRICE	Expr1
Power painter, 15 psi., 3-nozzle	8	109.99	879.92
7.25-in. pwr. saw blade	32	14.99	479.68
9.00-in. pwr. saw blade	18	17.49	314.82
Hrd. cloth, 1/4-in., 2x50	15	39.95	599.25
Hrd. cloth, 1/2-in., 3x50	23	43.99	1011.77
B&D jigsaw, 12-in. blade	8	109.92	879.36
B&D jigsaw, 8-in. blade	6	99.87	599.22
B&D cordless drill, 1/2-in.	12	38.95	467.40
Claw hammer	23	9.95	228.85
Sledge hammer, 12 lb.	8	14.40	115.20
Rat-tail file, 1/8-in. fine	43	4.99	214.57
Hicut chain saw, 16 in.	11	256.99	2826.89
PVC pipe, 3.5-in., 8-ft	188	5.87	1103.56
1.25-in. metal screw, 25	172	6.99	1202.28
2.5-in. wd. screw, 50	237	8.45	2002.65
Steel matting, 4'x8'x1/6", .5" mesh	18	119.95	2159.10

SELECT AND, OR, NOT OPERATORS:

```
SELECT      P_DESCRIPT, P_INDATE, P_PRICE, V_CODE
FROM        PRODUCT
WHERE       V_CODE = 21344 OR V_CODE = 24288;
```

P_DESCRIPT	P_INDATE	P_PRICE	V_CODE
7.25-in. pwr. saw blade	13-Dec-15	14.99	21344
9.00-in. pwr. saw blade	13-Nov-15	17.49	21344
B&D jigsaw, 12-in. blade	30-Dec-15	109.92	24288
B&D jigsaw, 8-in. blade	24-Dec-15	99.87	24288
Rat-tail file, 1/8-in. fine	15-Dec-15	4.99	21344
Hicut chain saw, 16 in.	07-Feb-16	256.99	24288

```
SELECT      P_DESCRIPT, P_INDATE, P_PRICE, V_CODE
FROM        PRODUCT
WHERE       P_PRICE < 50
AND         P_INDATE > '15-Jan-2016';
```

P_DESCRIPT	P_INDATE	P_PRICE	V_CODE
B&D cordless drill, 1/2-in.	20-Jan-16	38.95	25595
Claw hammer	20-Jan-16	9.95	21225
PVC pipe, 3.5-in., 8-ft	20-Feb-16	5.87	
1.25-in. metal screw, 25	01-Mar-16	6.99	21225
2.5-in. wd. screw, 50	24-Feb-16	8.45	21231

SELECT P_DESCRIPT, P_INDATE, P_PRICE, V_CODE
FROM PRODUCT
WHERE (P_PRICE < 50 AND P_INDATE > '15-Jan-2016')
OR V_CODE = 24288;

P_DESCRIPT	P_INDATE	P_PRICE	V_CODE
B&D jigsaw, 12-in. blade	30-Dec-15	109.92	24288
B&D jigsaw, 8-in. blade	24-Dec-15	99.87	24288
B&D cordless drill, 1/2-in.	20-Jan-16	38.95	25595
Claw hammer	20-Jan-16	9.95	21225
Hicut chain saw, 16 in.	07-Feb-16	256.99	24288
PVC pipe, 3.5-in., 8-ft	20-Feb-16	5.87	
1.25-in. metal screw, 25	01-Mar-16	6.99	21225
2.5-in. wd. screw, 50	24-Feb-16	8.45	21231

SELECT *
FROM PRODUCT
WHERE NOT (V_CODE = 21344);

SPECIAL OPERATORS: BETWEEN, IS NULL, LIKE, IN, EXISTS

BETWEEN

SELECT *
FROM PRODUCT
WHERE P_PRICE BETWEEN 50.00 AND 100.00;
or:
WHERE P_PRICE => 50.00 AND P_PRICE <= 100.00;

IS NULL

SELECT P_CODE, P_DESCRIPT, V_CODE V_CODE
FROM PRODUCT
WHERE V_CODE IS NULL;

LIKE

The LIKE special operator is used in conjunction with wildcards to find patterns within string attributes. Standard SQL allows you to use the percent sign (%) and underscore (_) wildcard characters to make matches when the entire string is not known:

```
SELECT     V_NAME, V_CONTACT, V_AREACODE, V_PHONE
FROM       VENDOR
WHERE      V_CONTACT LIKE 'Smith%';
```

Will yield all names with Smith in the name like Smith and Smithson. It Is case sensitive.

IN – SPECIAL OPERATOR

The IN operator is especially valuable when it is used in conjunction with subqueries. For example, suppose that you want to list the V_CODE and V_NAME of only those vendors who provide products. In that case, you could use a subquery within the IN operator to automatically generate the value list. The query would be:

```
SELECT     V_CODE, V_NAME
FROM       VENDOR
WHERE      V_CODE IN (SELECT V_CODE FROM PRODUCT);
```

EXISTS

The EXISTS special operator can be used whenever there is a requirement to execute a command based on the result of another query. That is, if a subquery returns any rows, run the main query; otherwise, do not. For example, the following query will list all vendors, but only if there are products to order:

```
SELECT     *
FROM       VENDOR
WHERE      EXISTS (SELECT * FROM PRODUCT WHERE P_QOH <= P_MIN);
```

UPDATE

Change P_INDATE from December 13, 2015, to January 18, 2016, in the second row of the PRODUCT table, use the primary key PRODUCT_CODE (P_CODE) (13-Q2/P2) to locate the correct row.

```
UPDATE     PRODUCT
SET        P_INDATE = '18-JAN-2016'
WHERE      P_CODE = '13-Q2/P2';
```

Updating more than one attribute in a row, update P_INDATE and P_PRICE:

```
UPDATE      PRODUCT
SET         P_INDATE = '18-JAN-2016', P_PRICE = 17.99, P_MIN = 10
WHERE       P_CODE = '13-Q2/P2';
```

DELETE

```
DELETE FROM     PRODUCT
WHERE           P_MIN = 5;
```

SQL JOIN

Old style join: Join a table with another table through their common Primary key and foreign key:

```
SELECT      P_CODE, P_DESCRIPT, P_PRICE, V_NAME
FROM        PRODUCT, VENDOR
WHERE       PRODUCT.V_CODE = VENDOR.V_CODE;
```

CROSS JOIN: performs a relational product (also known as the Cartesian product) of two tables. The cross join syntax is:

SELECT column-list FROM table1 CROSS JOIN table2

SELECT * FROM INVOICE CROSS JOIN LINE;

performs a cross join of the INVOICE and LINE tables that generates 144 rows. (There are 8 invoice rows and 18 line rows, yielding 8 × 18 = 144 rows.) You can also perform a cross join that yields only specified attributes. For example, you can specify:

SELECT INVOICE.INV_NUMBER, CUS_CODE, INV_DATE, P_CODE
FROM INVOICE CROSS JOIN LINE;

NATURAL JOIN

SELECT column-list FROM table1 NATURAL JOIN table2

The natural join will perform the following tasks:

- Determine the common attribute(s) by looking for attributes with identical names and compatible data types.
- Select only the rows with common values in the common attribute(s).

- If there are no common attributes, return the relational product of the two tables. The following example performs a natural join of the CUSTOMER and INVOICE tables and returns only selected attributes:

```
SELECT     CUS_CODE, CUS_LNAME, INV_NUMBER, INV_DATE
FROM       CUSTOMER NATURAL JOIN INVOICE;
```

JOIN ON

Another way to express a join when the tables have no common attribute names is to use the JOIN ON operand. The query will return only the rows that meet the indicated join condition. The join condition will typically include an equality comparison expression of two columns. Syntax:

SELECT column-list FROM table1 JOIN table2 ON join-condition

SELECT INVOICE.INV_NUMBER, PRODUCT.P_CODE, P_DESCRIPT, LINE_UNITS, LINE_PRICE

FROM INVOICE JOIN LINE ON INVOICE.INV_NUMBER = LINE. INV_NUMBER JOIN PRODUCT ON LINE.P_CODE = PRODUCT.P_CODE;

INV_NUMBER	P_CODE	P_DESCRIPT	LINE_UNITS	LINE_PRICE
1001	13-Q2/P2	7.25-in. pwr. saw blade	1	14.99
1001	23109-HB	Claw hammer	1	9.95
1002	54778-2T	Rat-tail file, 1/8-in. fine	2	4.99
1003	2238/QPD	B&D cordless drill, 1/2-in.	1	38.95
1003	1546-QQ2	Hrd. cloth, 1/4-in., 2x50	1	39.95
1003	13-Q2/P2	7.25-in. pwr. saw blade	5	14.99
1004	54778-2T	Rat-tail file, 1/8-in. fine	3	4.99
1004	23109-HB	Claw hammer	2	9.95
1005	PVC23DRT	PVC pipe, 3.5-in., 8-ft	12	5.87
1006	SM-18277	1.25-in. metal screw, 25	3	6.99
1006	2232/QTY	B&D jigsaw, 12-in. blade	1	109.92
1006	23109-HB	Claw hammer	1	9.95
1006	89-WRE-Q	Hicut chain saw, 16 in.	1	256.99
1007	13-Q2/P2	7.25-in. pwr. saw blade	2	14.99
1007	54778-2T	Rat-tail file, 1/8-in. fine	1	4.99
1008	PVC23DRT	PVC pipe, 3.5-in., 8-ft	5	5.87
1008	WR3/TT3	Steel matting, 4'x8'x1/6", .5" mesh	3	119.95
1008	23109-HB	Claw hammer	1	9.95

18 rows selected.

OUTER JOINS

An outer join returns not only the rows matching the join condition (that is, rows with matching values in the common columns), it returns the rows with unmatched values. The ANSI standard defines three types of outer joins: left, right, and full. The left and right designations reflect the order in which the tables are processed by the DBMS. Remember that join operations take place two tables at a time. The first table named in the FROM clause will be the left side, and the second table named will be the right side. If three or more tables are being joined, the result of joining the first two tables becomes the left side, and the third table becomes the right side.

The **LEFT OUTER JOIN** returns not only the rows matching the join condition (that is, rows with matching values in the common column), it returns the rows in the left table with unmatched values in the right table. The syntax is:

SELECT column-list
FROM table1 LEFT [OUTER] JOIN table2 ON join-condition

For example, the following query lists the product code, vendor code, and vendor name for all products and includes those vendors with no matching products:

SELECT P_CODE, VENDOR.V_CODE, V_NAME

FROM VENDOR LEFT JOIN PRODUCT ON VENDOR. V_CODE = PRODUCT.V_CODE;

```
P_CODE         V_CODE V_NAME
---------- ---------- -----------------
11QER/31        25595 Rubicon Systems
13-Q2/P2        21344 Gomez Bros.
14-Q1/L3        21344 Gomez Bros.
1546-QQ2        23119 Randsets Ltd.
1558-QW1        23119 Randsets Ltd.
2232/QTY        24288 ORDVA, Inc.
2232/QWE        24288 ORDVA, Inc.
2238/QPD        25595 Rubicon Systems
23109-HB        21225 Bryson, Inc.
54778-2T        21344 Gomez Bros.
89-WRE-Q        24288 ORDVA, Inc.
SM-18277        21225 Bryson, Inc.
SW-23116        21231 D&E Supply
WR3/TT3         25595 Rubicon Systems
                22567 Dome Supply
                21226 SuperLoo, Inc.
                24004 Brackman Bros.
                25501 Damal Supplies
                25443 B&K, Inc.

19 rows selected.
```

The **RIGHT OUTER JOIN** returns not only the rows matching the join condition (that is, rows with matching values in the common column), it returns the rows in the right table with unmatched values in the left table. The syntax is:

SELECT column-list
FROM table1 RIGHT [OUTER] JOIN table2 ON join-condition

For example, the following query lists the product code, vendor code, and vendor name for all products and includes products that do not have a matching vendor code:

SELECT P_CODE, VENDOR.V_CODE, V_NAME

FROM VENDOR RIGHT JOIN PRODUCT ON VENDOR. V_CODE = PRODUCT.V_CODE;

```
P_CODE         V_CODE V_NAME
---------- ---------- -----------------
SM-18277        21225 Bryson, Inc.
23109-HB        21225 Bryson, Inc.
SW-23116        21231 D&E Supply
54778-2T        21344 Gomez Bros.
14-Q1/L3        21344 Gomez Bros.
13-Q2/P2        21344 Gomez Bros.
1558-QW1        23119 Randsets Ltd.
1546-QQ2        23119 Randsets Ltd.
89-WRE-Q        24288 ORDVA, Inc.
2232/QWE        24288 ORDVA, Inc.
2232/QTY        24288 ORDVA, Inc.
WR3/TT3         25595 Rubicon Systems
2238/QPD        25595 Rubicon Systems
11QER/31        25595 Rubicon Systems
PVC23DRT
23114-AA

16 rows selected.
```

The **FULL OUTER JOIN** returns not only the rows matching the join condition (that is, rows with matching values in the common column), it returns all of the rows with unmatched values in the table on either side. The syntax is:

SELECT FROM column-list table1 FULL [OUTER] JOIN table2 ON join-condition

For example, the following query lists the product code, vendor code, and vendor name for all products and includes all product rows (products without matching vendors) as well as all vendor rows (vendors without matching products):

SELECT P_CODE, VENDOR.V_CODE, V_NAME

FROM VENDOR FULL JOIN PRODUCT ON VENDOR. V_CODE = PRODUCT.V_CODE;

```
P_CODE         V_CODE V_NAME
---------- ---------- -----------------
11QER/31        25595 Rubicon Systems
13-Q2/P2        21344 Gomez Bros.
14-Q1/L3        21344 Gomez Bros.
1546-QQ2        23119 Randsets Ltd.
1558-QW1        23119 Randsets Ltd.
2232/QTY        24288 ORDVA, Inc.
2232/QWE        24288 ORDVA, Inc.
2238/QPD        25595 Rubicon Systems
23109-HB        21225 Bryson, Inc.
23114-AA
54778-2T        21344 Gomez Bros.
89-WRE-Q        24288 ORDVA, Inc.
PVC23DRT
SM-18277        21225 Bryson, Inc.
SW-23116        21231 D&E Supply
WR3/TT3         25595 Rubicon Systems
                22567 Dome Supply
                21226 SuperLoo, Inc.
                24004 Brackman Bros.
                25501 Damal Supplies
                25443 B&K, Inc.

21 rows selected.
```