Chapter 3

The Relational Database Model

Answers to Review Questions

ONLINE CONTENT

The website (*www.cengagebrain.com*) includes MS Access databases and SQL script files (Oracle, SQL Server, and MySQL) for all of the data sets used throughout the book.

1. What is the difference between a database and a table?

A table, a logical structure that represents an entity set, is only one of the components of a database.

The database is a structure that houses one or more tables and metadata. The metadata are data about data. Metadata include the data (attribute) characteristics and the relationships between the entity sets.

2. What does it mean to say that a database displays both entity integrity and referential integrity?

Entity integrity describes a condition in which all tuples within a table are ***uniquely*** identified by their primary key. The unique value requirement prohibits a null primary key value, because nulls are not unique.

Referential integrity describes a condition in which a foreign key value has a match in the corresponding table or in which the foreign key value is null. The null foreign key "value" makes it possible ***not*** to have a corresponding value, but the matching requirement on values that are not null makes it impossible to have an invalid value.

3. Why are entity integrity and referential integrity important in a database?

Entity integrity and referential integrity are important because they are the basis for expressing and implementing relationships in the entity relationship model. Entity integrity ensures that each row is uniquely identified by the primary key. Therefore, entity integrity means that a proper search for an ***existing*** tuple (row) will always be successful. (And the failure to find a match on a row search will always mean that the row for which the search is conducted does not exist in that table.) Referential integrity means that, if the foreign key contains a value, that value refers to an existing valid tuple (row) in another relation. Therefore, referential integrity ensures that it will be impossible to assign a non-existing foreign key value to a table.

4. What are the requirements that two relations must satisfy in order to be considered union-compatible?

In order for two relations to be union-compatible, both must have the same number of attributes (columns) and corresponding attributes (columns) must have the same domain. The first requirement is easily identified be a cursory glance at the relations' structures. If the first relation has 3 attributes then the second relation must also have 3 attributes. If the first table has 10 attributes, then the second relation must also have 10 attributes. The second requirement is more difficult to assess and requires

understanding the meanings of the attributes in the business environment. Recall that an attribute's domain is the set of allowable values for that attribute. To satisfy the second requirement for union-compatibility, the first attribute of the first relation must have the same domain as the first attribute of the second relation. The second attribute of the first relation must have the same domain as the second attribute of the second relation. The third attribute of the first relation must have the same domain as the third attribute of the second relation, and so on.

5. Which relational algebra operators can be applied to a pair of tables that are not union-compatible?

The Product, Join, and Divide operators can be applied to a pair of tables that are not union-compatible. Divide does place specific requirements on the tables to be operated on; however, those requirements do not include union-compatibility. Select (or Restrict) and Project are performed on individual tables, not pairs of tables. (Note that if two tables are joined, then the result is a single table and the Select or Project operator is performed on that single table.)

6. Explain why the data dictionary is sometimes called "the database designer's database."

Just as the database stores data that is of interest to the users regarding the objects in their environment that are important to them, the data dictionary stores data that is of interest to the database designer about the important decisions that were made in regard to the database structure. The data dictionary contains the number of tables that were created, the names of all of those tables, the attributes in each table, the relationships between the tables, the data type of each attribute, the enforced domains of the attributes, etc. All of these data represent decisions that the database designer had to make and data that the database designer needs to record about the database.

7. A database user manual notes that, “The file contains two hundred records, each record containing nine fields.” Use appropriate relational database terminology to “translate” that statement.

Using the proper relational terminology, the statement may be translated to "the table -- or entity set -- contains two hundred rows -- or, if you like, two hundred tuples, or entities. Each of these rows contains nine attributes."

8. Using the STUDENT and PROFESSOR tables shown in Figure Q3.8 to illustrate the difference between a natural join, an equijoin, and an outer join.

FIGURE Q3.8 The Ch03_CollegeQue Database Tables

Table name: STUDENT Database name: Ch03_CollegeQue

STU_CODE	PROF_CODE
100278	
128569	2
512272	4
531235	2
531268	
553427	1

Table name: PROFESSOR

PROF_CODE	DEPT_CODE
1	2
2	6
3	6
4	4

The natural JOIN process begins with the PRODUCT of the two tables. Next, a SELECT (or RESTRICT) is performed on the PRODUCT generated in the first step to yield only the rows for which the PROF_CODE values in the STUDENT table are matched in the PROF table. Finally, a PROJECT is performed to produce the natural JOIN output by listing only a single copy of each attribute. The order in which the query output rows are shown is not relevant.

STU_CODE	PROF_CODE	DEPT_CODE
128569	2	6
512272	4	4
531235	2	6
553427	1	2

The equiJOIN's results depend on the specified condition. At this stage of the students' understanding, it may be best to focus on equijoins that retrieve all matching values in the common attribute. In such a case, the output will be:

STU_CODE	STUDENT. PROF_CODE	PROFESSOR. PROF_CODE	DEPT_CODE
128569	2	2	6
512272	4	4	4
531235	2	2	6
553427	1	1	2

Notice that in equijoins, the common attribute appears from both tables. It is normal to prefix the attribute name with the table name when an attribute appears more than once in a table. This maintains the requirement that attribute names be unique within a relational table.

In the Outer JOIN, the unmatched pairs would be retained and the values that do not have a match in the other table would be left null. It should be made clear to the students that Outer Joins are not the

opposite of Inner Joins (like Natural Joins and Equijoins). Rather, they are "Inner Join Plus" – they include all of the matched records found by the Inner Join **plus** the unmatched records. Outer JOINs are normally performed as either a Left Outer Join or a Right Outer Join so that the operator specifies which table's unmatched rows should be included in the output. Full Outer Joins depict the matched records plus the unmatched records from both tables. Also, like Equijoins, Outer Joins do not drop a copy of the common attribute. Therefore, a Full Outer Join will yield these results:

STU_CODE	STUDENT. PROF_CODE	PROFESSOR. PROF_CODE	DEPT_CODE
128569	2	2	6
512272	4	4	4
531235	2	2	6
553427	1	1	2
100278			
531268			
		3	6

A Left Outer Join of STUDENT to PROFESSOR would include the matched rows plus the unmatched STUDENT rows:

STU_CODE	STUDENT. PROF_CODE	PROFESSOR. PROF_CODE	DEPT_CODE
128569	2	2	6
512272	4	4	4
531235	2	2	6
553427	1	1	2
100278			
531268			

A Right Outer Join of STUDENT to PROFESSOR would include the matched rows plus the unmatched PROFESSOR row.

STU_CODE	STUDENT. PROF_CODE	PROFESSOR. PROF_CODE	DEPT_CODE
128569	2	2	6
512272	4	4	4
531235	2	2	6
553427	1	1	2
		3	6

9. Create the table that would result from π_{stu_code}(student).

STU_CODE
128569
512272
531235
553427
100278
531268

10. Create the table that would result from $\pi_{stu\_code,dept\_code}$(student ⋈ professor).

STU_CODE	DEPT_CODE
128569	6
512272	4
531235	6
553427	2

11. Create the basic ERD for the database shown in Figure Q3.8.

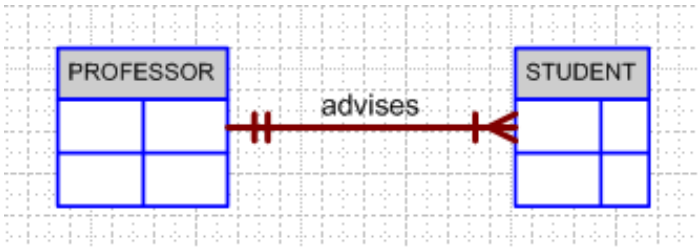
Both the Chen and Crow's Foot solutions are shown in Figure Q3.11. (We have used the PowerPoint template to produce the first of the two Crow's Foot ERDs and Visio Professional to produce the second of the two Crow's Foot ERDs.

Figure Q3.11 The Chen and Crow's Foot ERD Solutions for Question 11

Chen ERD (generated with PowerPoint)

Crow's Foot ERD (generated with PowerPoint)

Chen ERD (generated with Visio Professional)



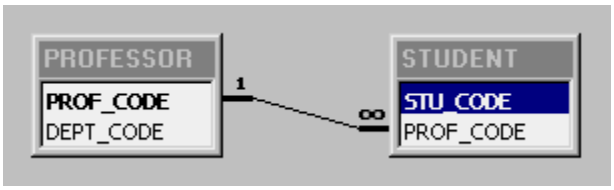
NOTE

From this point forward, we will show the ERDs in Visio Professional format unless the problem specifies a different format.

12. Create the relational diagram for the database shown in Figure Q3.8.

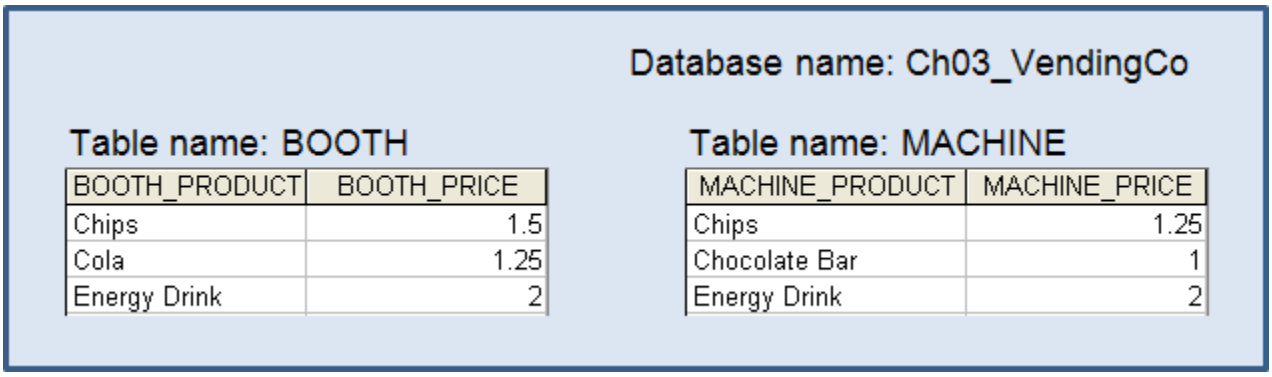
The relational diagram, generated in the Microsoft Access **Ch03_CollegeQue** database, is shown in Figure Q.3.11.

Figure Q3.11 The Relational Diagram



Use Figure Q3.13 to answer questions 13 – 17.

Figure Q3.13 The Ch03_VendingCo database tables



Database name: Ch03_VendingCo

Table name: BOOTH

BOOTH_PRODUCT	BOOTH_PRICE
Chips	1.5
Cola	1.25
Energy Drink	2

Table name: MACHINE

MACHINE_PRODUCT	MACHINE_PRICE
Chips	1.25
Chocolate Bar	1
Energy Drink	2

13. Write the relational algebra formula to apply a UNION relational operator to the tables shown in Figure Q3.13.

The question does not specify the order in which the table should be used in the operation. Therefore, both of the following are correct.

BOOTH ∪ MACHINE
MACHINE ∪ BOOTH

You can use this as an opportunity to emphasize that the order of the tables in a UNION command do not change the contents of the data returned.

14. Create the table that results from applying a UNION relational operator to the tables shown in Fig Q3.13

BOOTH_PRODUCT	BOOTH_PRICE
Chips	1.5
Cola	1.25
Energy Drink	2
Chips	1.25
Chocolate Bar	1

Note that when the attribute names are different, the result will take the attribute names from the first relation. In this case, the solution assumes the operation was BOOTH UNION MACHINE. If the operation had been MACHINE UNION BOOTH then the attribute names from the MACHINE table would have appears as the attribute names in the result. Also, notice that the "Chips" from both tables appears in the result, but the "Energy Drink" from both does not. A UNION operator will eliminate duplicate rows from the result; however, the entire row must match for two rows to be considered duplicates. In the case of "Chips", the product names were the same but the prices were different. In the case of "Energy Drink", both the product names and the prices matched so the second Energy Drink row was dropped from the result.

15. Write the relational algebra formula to apply an INTERSECT relational operator to the tables shown in Figure Q3.13.

The question does not specify the order in which the table should be used in the operation. Therefore, both of the following are correct.

BOOTH ⋂ MACHINE
MACHINE ⋂ BOOTH

16. Create the table that results from applying an INTERSECT relational operator to the tables shown in Fig Q3.13.

BOOTH_PRODUCT	BOOTH_PRICE
Energy Drink	2

Note that when the attribute names are different, the result will take the attribute names from the first relation. In this case, the solution assumes the operation was BOOTH INTERSECT MACHINE. If the operation had been MACHINE INTERSECT BOOTH then the attribute names from the MACHINE table would have appears as the attribute names in the result.

17. Using the tables in Figure Q3.13, create the table that results from MACHINE DIFFERENCE BOOTH.

MACHINE_PRODUCT	MACHINE_PRICE
Chips	1.25
Chocolate Bar	1

Note that the order in which the relations are specified is significant in the results returned. The DIFFERENCE operator returns the rows from the first relation that are not duplicated in the second relation. Just as with the INTERSECT operator, the entire row must match an existing row to be considered a duplicate.

18. Suppose that you have the ERM shown in Figure Q3.18.

FIGURE Q3.14 The Crow's Foot ERD for Question 18

During some time interval, a DRIVER can drive many TRUCKs and any TRUCK can be driven by many DRIVERs

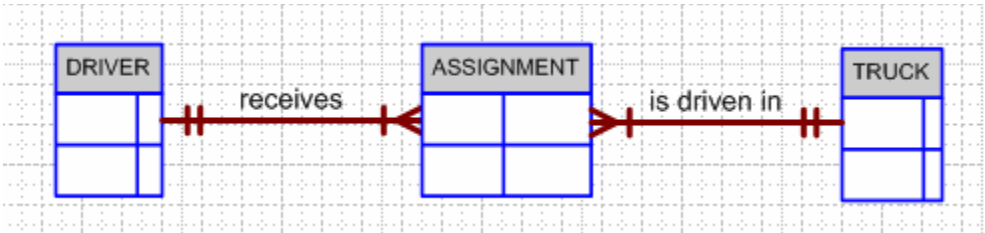
How would you convert this model into an ERM that displays only 1:M relationships? (Make sure that you draw the revised ERM.)

The Crow's Foot solution is shown in Figure Q3.18. Note that the original M:N relationship has been decomposed into two 1:M relationships based on these business rules:

- A driver may receive many (driving) assignments.
- Each (driving) assignment is made for a single driver.
- A truck may be driven in many (driving) assignments.
- Each (driving) assignment is made for a single truck.

Note that a driver can drive only one truck at a time, but during some period of time, a driver may be assigned to drive many trucks. The same argument holds true for trucks – a truck can only be driven during one trip (assignment) at a time, but during some period of time, a truck may be assigned to be driven in many trips. Also, remind students that they will be introduced to optional (and additional) relationships as they study Chapter 4, "Entity Relationship (ER) Modeling." Finally, remind your students that you always read the relationship from the "1" side to the "M" side. Therefore, you read "DRIVER receives ASSIGNMENT and "TRUCK is driven in ASSIGNMENT."

Figure Q3.18 The Crow's Foot ERM Solution for Question 18



19. What are homonyms and synonyms, and why should they be avoided in database design?

Homonyms appear when more than one attribute has the same name. Synonyms exist when the same attribute has more than one name. Avoid both to avoid inconsistencies. For example, suppose we check the database for a specific attribute such as NAME. If NAME refers to customer names as well as to sales rep names, a clear case of a homonym, we have created an ambiguity, because it is no longer clear which entity the NAME belongs to.

Synonyms make it difficult to keep track of foreign keys if they are named differently from the primary keys they point to. Using REP_NUM as the foreign key in the CUSTOMER table to reference the primary key REP_NUM in the SALESREP table is much clearer than naming the CUSTOMER table's foreign key SLSREP. The proliferation of different attribute names to describe the same attributes will also make the data dictionary more cumbersome to use.

Some data RDBMSes let the data dictionary check for homonyms and synonyms to alert the user to their existence, thus making their use less likely. For example, if a CUSTOMER table contains the (foreign) key REP_NUM, the entry of the attribute REP_NUM in the SALESREP table will either cause it to inherit all the characteristics of the original REP_NUM, or it will reject the use of this attribute name when different characteristics are declared by the user.

20. How would you implement a l:M relationship in a database composed of two tables? Give an example.

Let's suppose that an auto repair business wants to track its operations by customer. At the most basic level, it's reasonable to assume that any database design you produce will include at least a car entity and a customer entity. Further suppose that it is reasonable to assume that:

- A car is owned just by one customer.
- A customer can own more than one car.

The CAR and CUSTOMER entities and their relationships are represented by the Crow's Foot ERD shown in Figure Q3.20. (Discussion: Explain to your students that the ERDs are very basic at this point. Your students will learn how to incorporate much more detail into their ERDs in Chapter 4. For example, no thought has –yet – been given to optional relationships or to the strength of those relationships. At this stage of learning the business of database design, simple is good! To borrow an old Chinese proverb, a journey of a thousand miles begins with a single step.)

Figure Q3.20 The CUSTOMER owns CAR ERM

21. Identify and describe the components of the table shown in Figure Q3.21, using correct terminology. Use your knowledge of naming conventions to identify the table's probable foreign key(s).

FIGURE Q3.21 The Ch03_NoComp Database EMPLOYEE Table

EMP_NUM	EMP_LNAME	EMP_INITIAL	EMP_FNAME	DEPT_CODE	JOB_CODE
11234	Friedman	K	Robert	MKTG	12
11238	Olanski	D	Delbert	MKTG	12
11241	Fontein		Juliette	INFS	5
11242	Cruazona	J	Maria	ENG	9
11245	Smithson	B	Bernard	INFS	6
11248	Washington	G	Oleta	ENGR	8
11256	McBride		Randall	ENGR	8
11257	Kachinn	D	Melanie	MKTG	14
11258	Smith	W	William	MKTG	14
11260	Ratula	A	Katrina	INFS	5

Figure Q3.21's database table contains:

- **One entity set:** EMPLOYEE.
- **Five attributes:** EMP_NUM, EMP_LNAME, EMP_INIT, EMP_FNAME, DEPT_CODE and JOB_CODE.
- **Ten entities:** the ten workers shown in rows 1-10.
- **One primary key:** the attribute EMP_NUM because it identifies each row uniquely.
- **Two foreign keys:** the attribute DEPT_CODE, which probably references a department to which the employee is assigned and the attribute JOB_CODE which probably references another table in which you would find the description of the job and perhaps additional information pertaining to the job.

Use the database composed of the two tables shown in Figure Q3.22 to answer Questions 22-27.

FIGURE Q3.18 The Ch03_Theater Database Tables

Table name: DIRECTOR Database name: Ch03_Theater

DIR_NUM	DIR_LNAME	DIR_DOB
100	Broadway	12-Jan-65
101	Hollywoody	18-Nov-53
102	Goofy	21-Jun-62

Table name: PLAY

PLAY_CODE	PLAY_NAME	DIR_NUM
1001	Cat On a Cold, Bare Roof	102
1002	Hold the Mayo, Pass the Bread	101
1003	I Never Promised You Coffee	102
1004	Silly Putty Goes To Washington	100
1005	See No Sound, Hear No Sight	101
1006	Starstruck in Biloxi	102
1007	Stranger In Parrot Ice	101

22. Identify the primary keys.

DIR_NUM is the DIRECTOR table's primary key.
PLAY_CODE is the PLAY table's primary key.

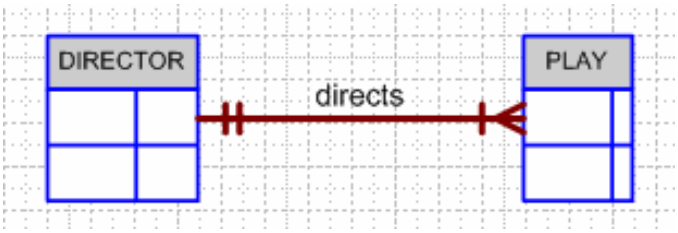
23. Identify the foreign keys.

The foreign key is DIR_NUM, located in the PLAY table. Note that the foreign key is located on the "many" side of the relationship between director and play. (Each director can direct many plays ... but each play is directed by only one director.)

24. Create the ERM.

The entity relationship model is shown in Figure Q3.24.

Figure Q3.24 The Theater Database ERD



25. Create the relational diagram to show the relationship between DIRECTOR and PLAY.

The relational diagram, shown in Figure 3.21, was generated with the help of Microsoft Access. (Check the **Ch03_Theater** database.)

Figure Q3.25 The Relational Diagram

26. Suppose you wanted quick lookup capability to get a listing of all plays directed by a given director. Which table would be the basis for the INDEX table, and what would be the index key?

The PLAY table would be the basis for the appropriate index table. The index key would be the attribute DIR_NUM.

27. What would be the conceptual view of the INDEX table that is described in question 26? Depict the contents of the conceptual INDEX table.

The conceptual index table is shown in Table Q3.27.

Figure Q3.27 The Conceptual Index Table

Index Key	Pointers to the PLAY table
100	4
101	2, 5, 7
102	1, 3, 6