The Data Dictionary And The System Catalog

- The data dictionary provides a detailed description of all tables found within the user/designer-created database. Thus, the data dictionary contains at least all of the attribute names and characteristics for each table in the system.
- In short, the data dictionary contains metadata—data about data.

TABLE 3.6 A Sample Data Dictionary

TABLE NAME	ATTRIBUTE NAME	CONTENTS	TYPE	FORMAT	RANGE	REQUIRED	PK OR FK	FK REFERENCED TABLE
CUSTOMER	CUS_CODE	Customer account code	CHAR(5)	99999	10000–99999	Y	PK	
	CUS_LNAME	Customer last name	VARCHAR(20)	Xxxxxxxx		Y		
	CUS_FNAME	Customer first name	VARCHAR(20)	Xxxxxxxx		Y		
	CUS_INITIAL	Customer initial	CHAR(1)	X				
	CUS_RENEW_DATE	Customer insurance renewal date	DATE	dd-mmm-yyyy				
	AGENT_CODE	Agent code	CHAR(3)	999			FK	AGENT_CODE
AGENT	AGENT_CODE	Agent code	CHAR(3)	999		Y	PK	
	AGENT_AREACODE	Agent area code	CHAR(3)	999		Y		
	AGENT_PHONE	Agent telephone number	CHAR(8)	999-9999		Y		
	AGENT_LNAME	Agent last name	VARCHAR(20)	Xxxxxxxx		Y		
	AGENT_YTD_SLS	Agent year-to-date sales	NUMBER(9,2)	9,999,999.99		Y		

FK	=	Foreign key
PK	=	Primary key
CHAR	=	Fixed character length data (1–255 characters)
VARCHAR	=	Variable character length data (1–2,000 characters)
NUMBER	=	Numeric data (NUMBER(9,2)) is used to specify numbers with two decimal places and up to nine digits, including the decimal places. Some RDBMSs permit the use of a MONEY or CURRENCY data type.

Note: Telephone area codes are always composed of digits 0–9. Because area codes are not used arithmetically, they are most efficiently stored as character data. Also, the area codes are always composed of three digits. Therefore, the area code data type is defined as CHAR(3). On the other hand, names do not conform to some standard length. Therefore, the customer first names are defined as VARCHAR(20), thus indicating that up to 20 characters may be used to store the names. Character data are shown as left-justified.

The Data Dictionary And The System Catalog

- The data dictionary in Table 3.6 is an example of the human view of the entities, attributes, and relationships. The purpose of this data dictionary is to ensure that all members of database design and implementation teams use the same table and attribute names and characteristics.
- The DBMS's internally stored data dictionary contains additional information about relationship types, entity and referential integrity checks and enforcement, and index types and components. This additional information is generated during the database implementation stage.
- The data dictionary is sometimes described as "the database designer's database" because it records the design decisions about tables and their structures.

The Data Dictionary And The System Catalog

- Like the data dictionary, the system catalog contains metadata. The system catalog can be described as a detailed system data dictionary that describes all objects within the database, including data about table names, the table's creator and creation date, the number of columns in each table, the data type corresponding to each column, index filenames, index creators, authorized users, and access privileges.

The Data Dictionary And The System Catalog

- The system catalog contains all required data dictionary information, the terms system catalog and data dictionary are often used interchangeably. In fact, current relational database software generally provides only a system catalog, from which the designer's data dictionary information may be derived. The system catalog is actually a system-created database whose tables store the user/designer-created database characteristics and contents. Therefore, the system catalog tables can be queried just like any user/designer-created table.

The Data Dictionary And The System Catalog

- In effect, the system catalog automatically produces database documentation. As new tables are added to the database, that documentation also allows the RDBMS to check for and eliminate homonyms and synonyms. In general terms, homonyms are similar-sounding words with different meanings, or identically spelled words with different meanings, such as fair (meaning "just") and fair (meaning "festival"). In a database context, the word homonym indicates the use of the same attribute name to label different attributes. For example, using C_NAME to label a customer name attribute in a CUSTOMER table and also use C_NAME to label a consultant name attribute in a CONSULTANT table. To lessen confusion, you should avoid database homonyms; the data dictionary is very useful in this regard. In a database context, a synonym is the opposite of a homonym and indicates the use of different names to describe the same attribute. For example, car and auto refer to the same object. Synonyms must be avoided.

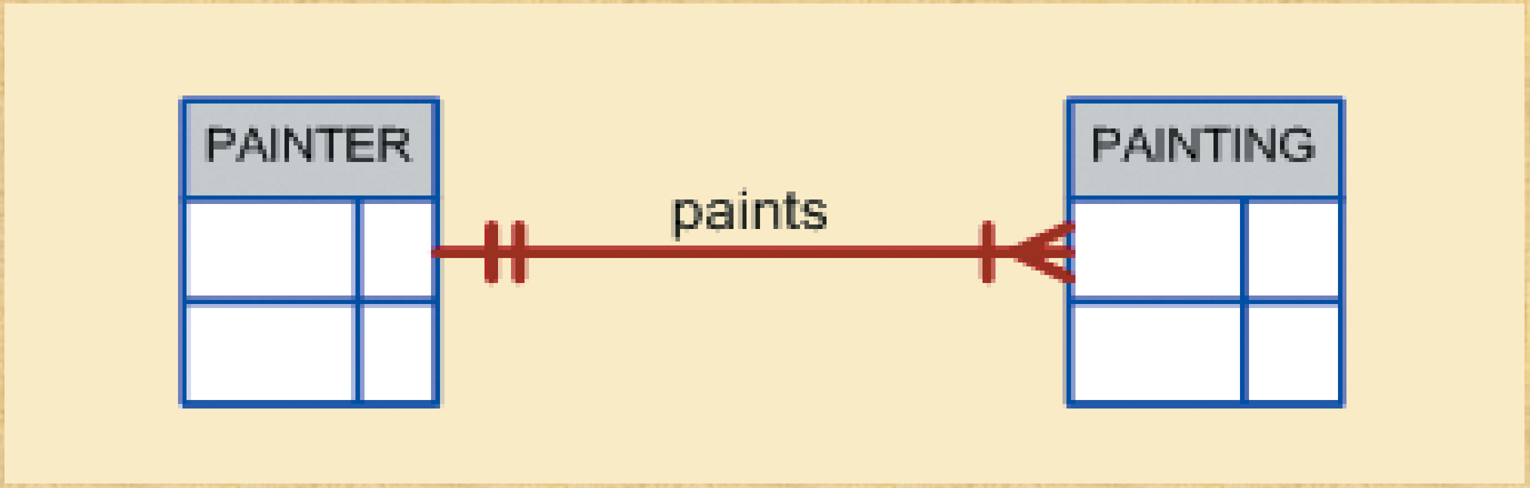
Relationships Within The Relational Database

- Exploring those relationships further to help to apply them properly when developer start developing database designs, focusing on the following points:
- The 1:M relationship is the relational modeling ideal. Therefore, this relationship type should be the norm in any relational database design.
- The 1:1 relationship should be rare in any relational database design.
- M:N relationships cannot be implemented as such in the relational model. It will be seen how any M:N relationships can be changed into two 1:M relationships.

Relationships Within The Relational Database

- **The 1:M Relationship**

The 1:M relationship is the relational database norm. To see how such a relationship is modeled and implemented, consider the PAINTER paints PAINTING



Relationships Within The Relational Database

- **Note: The one-to-many (1:M) relationship is easily implemented in the relational model by putting the primary key of the "1" side in the table of the "many" side as a foreign key.**
- **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

Relationships Within The Relational Database

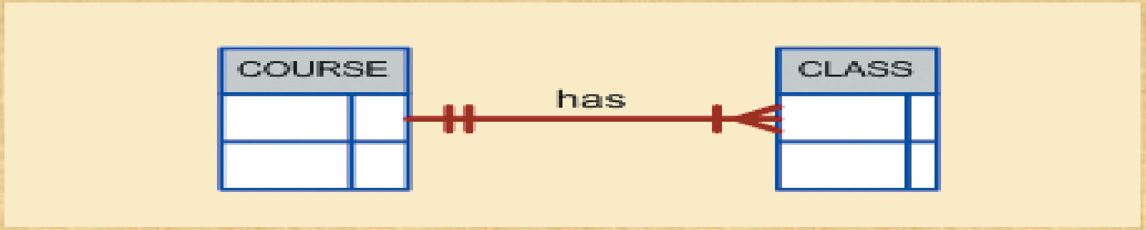
The 1:M Relationship

- The PAINTER and PAINTING table contents in Figure, note the following features:
- Each painting is painted by one and only one painter, but each painter could have painted many paintings. Note that painter 123 (Georgette P. Ross) has three paintings 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.

Relationships Within The Relational Database

The 1:M Relationship

- The 1:M relationship is found in any database environment. Students in a typical college or university will discover that each COURSE can generate many CLASSes but that each CLASS refers to only one COURSE. For example, an Accounting II course might yield two classes: one offered on Monday, Wednesday, and Friday (MWF) from 10:00 a.m. to 10:50 a.m. and one offered on Thursday (Th) from 6:00 p.m. to 8:40 p.m.



Relationships Within The Relational Database

The 1:M Relationship

- Therefore, the 1:M relationship between COURSE and CLASS might be described this way:
- Each COURSE can have many CLASSes, but each CLASS references only one COURSE.
- There will be only one row in the COURSE table for any given row in the CLASS table, but there can be many rows in the CLASS table for any given row in the COURSE table.

Relationships Within The Relational Database

ERM for the 1:M relationship between COURSE and CLASS

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

Relationships Within The Relational Database

The 1:M Relationship

- Review some important terminology. Note that CLASS_CODE in the CLASS table uniquely identifies each row. Therefore, CLASS_CODE has been chosen to be the primary key. However, the combination CRS_CODE and CLASS_SECTION will also uniquely identify each row in the class table. In other words, the composite key composed of CRS_CODE and CLASS_SECTION is a candidate key. Any candidate key must have the not null and unique constraints enforced.
- PAINTER table's primary key, PAINTER_NUM, is included in the PAINTING table as a foreign key. Similarly, in the COURSE table's primary key, CRS_CODE, is included in the CLASS table as a foreign key.

End of Lecture:

Any Questions... ?

Relationships Within The Relational Database

The 1:1 Relationship

- 1:1 label implies, in this relationship, one entity can be related to only one other entity, and vice versa. For example, one department chair—a professor—can chair only one department and one department can have only one department chair. The entities PROFESSOR and DEPARTMENT thus exhibit a 1:1 relationship.

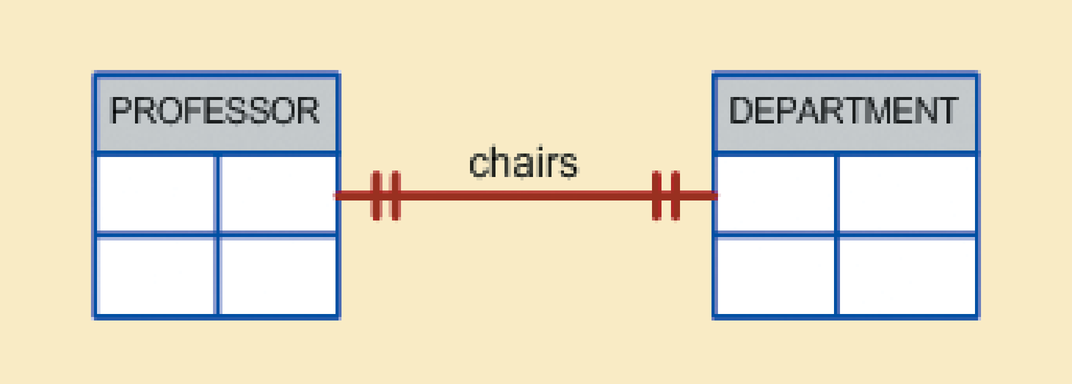


FIGURE 3.23 **The implemented 1:1 relationship between PROFESSOR and DEPARTMENT**

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

Relationships Within The Relational Database

The 1:1 Relationship

- Each professor is a Tiny College employee. Therefore, the professor identification is through the EMP_NUM.
- 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 PROFESSOR table contains the DEPT_CODE foreign key to implement the 1:M
- DEPARTMENT employs PROFESSOR relationship. This is a good example of how two entities can participate in two (or even more) relationships simultaneously.

Relationships Within The Relational Database

The 1:1 Relationship

- The preceding "PROFESSOR chairs DEPARTMENT" example illustrates a proper 1:1 relationship. In fact, the use of a 1:1 relationship ensures that two entity sets are not placed in the same table when they should not be.
- However, the existence of a 1:1 relationship sometimes means that the entity components were not defined properly. It could indicate that the two entities actually belong in the same table!

Relationships Within The Relational Database

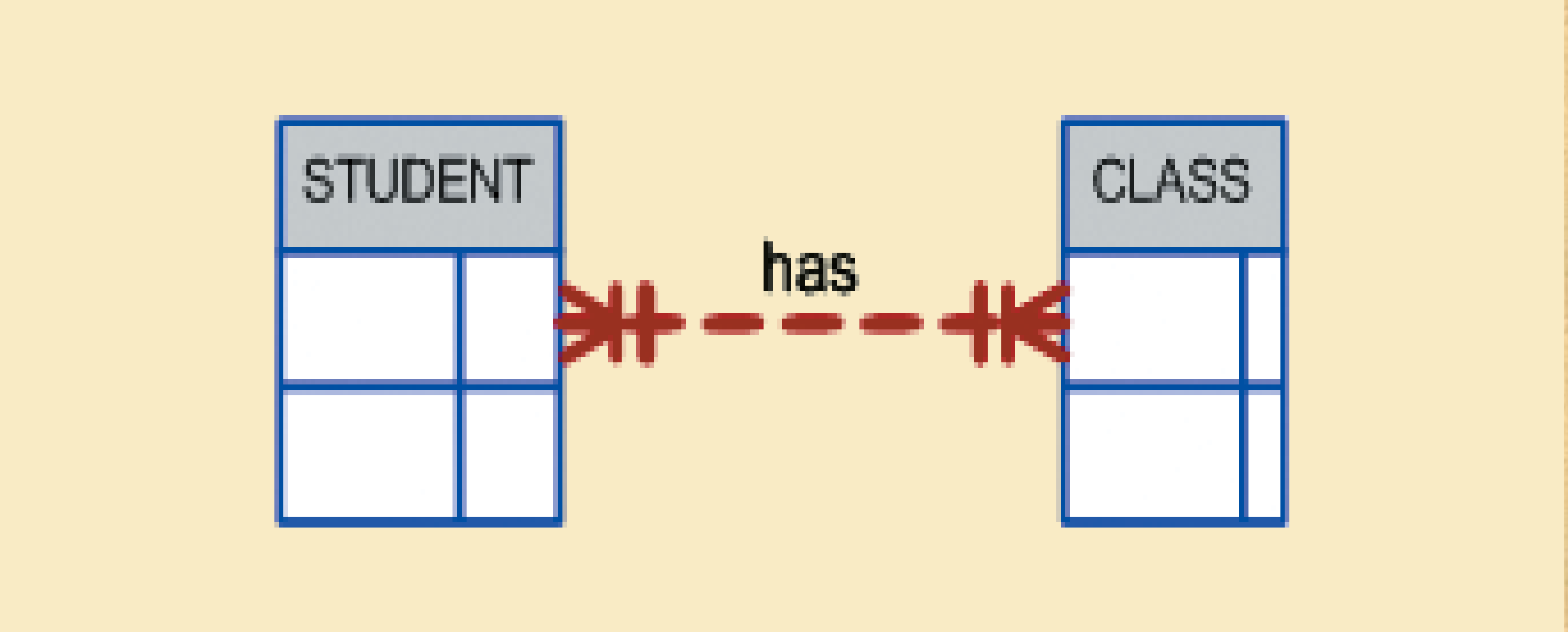
The 1:1 Relationship

- As rare as 1:1 relationships should be, certain conditions absolutely require their use. For example, suppose you manage the database for a company that employs pilots, accountants, mechanics, clerks, salespeople, service personnel, and more. Pilots have many attributes that the other employees don't have, such as licenses, medical certificates, flight experience records, dates of flight proficiency checks, and proof of required periodic medical checks. If you put all of the pilot-specific attributes in the EMPLOYEE table, you will have several nulls in that table for all employees who are not pilots.
- To avoid the proliferation of nulls, it is better to split the pilot attributes into a separate table (PILOT) that is linked to the EMPLOYEE table in a 1:1 relationship. Because pilots have many attributes that are shared by all employees—such as name, date of birth, and date of first employment—those attributes would be stored in the EMPLOYEE table

Relationships Within The Relational Database

The M:N Relationship

- A many-to-many (M:N) relationship is not supported directly in the relational environment. However, M:N relationships can be implemented by creating a new entity in 1:M relationships with the original entities.
- The ER model shows the M:N relationship.



Relationships Within The Relational Database

The M:N Relationship

- To explore the many-to-many (M:N) relationship, consider a rather typical college environment in which each STUDENT can take many CLASSes, and each CLASS can contain many STUDENTs.
- 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.

Relationships Within The Relational Database

The M:N Relationship

- To examine the M:N relationship more closely, imagine a small college with two students, each of whom takes three classes. Table 3.7 shows the enrollment data for the two students.

TABLE 3.7 Sample Student Enrollment Data

STUDENT'S LAST NAME	SELECTED CLASSES
Bowser	Accounting 1, ACCT-211, code 10014 Intro to Microcomputing, CIS-220, code 10018 Intro to Statistics, QM-261, code 10021
Smithson	Accounting 1, ACCT-211, code 10014 Intro to Microcomputing, CIS-220, code 10018 Intro to Statistics, QM-261, code 10021

FIGURE 3.25 The M:N relationship between STUDENT and CLASS

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

Database name: Ch03_CollegeTry

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

Relationships Within The Relational Database

The M:N Relationship

- Although the M:N relationship is logically reflected in Figure 3.24, it should not be implemented as shown in Figure 3.25 for two good reasons:
- 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 student 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 many duplications: each student taking the class generates a CLASS record. The problem would be even worse if the CLASS table included such attributes as credit hours and course description. Those redundancies lead to the anomalies.
- Given the structure and contents of the two tables, the relational operations become very complex and are likely to lead to system efficiency errors and output errors.

Relationships Within The Relational Database

The M:N Relationship

- Fortunately, the problems inherent in the many-to-many (M:N) relationship can easily be avoided by creating a composite entity (also referred to as a bridge entity or an associative entity). Because such a table is used to link the tables that originally were related in a M:N relationship, the composite entity structure includes—as foreign keys—at least the primary keys of the tables that are to be linked. The database designer has two main options when defining a composite table's primary key: use the combination of those foreign keys or create a new primary key.

Relationships Within The Relational Database

The M:N Relationship

- Remember that each entity in the ERM is represented by a table. Therefore, you can create the composite ENROLL table shown in Figure 3.26 to link the tables CLASS and STUDENT. In this example, the ENROLL table's primary key is the combination of its foreign keys CLASS_CODE and STU_NUM. But the designer could have decided to create a single-attribute new primary key such as ENROLL_LINE, using a different line value to identify each ENROLL table row uniquely. (Microsoft Access users might use the Autonumber data type to generate such line values automatically.)

FIGURE 3.26 Converting the M:N relationship into two 1:M relationships

Database name: Ch03_CollegeTry2

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

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

Relationships Within The Relational Database

The M:N Relationship

- Because the ENROLL table in Figure 3.26 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.

Note

- In addition to the linking attributes, the composite ENROLL table can also contain such relevant attributes as the grade earned in the course. In fact, a composite table can contain any number of attributes that the designer wants to track. Keep in mind that the composite entity, although it is implemented as an actual table, is conceptually a logical entity that was created as a means to an end: to eliminate the potential for multiple redundancies in the original M:N relationship.

Relationships Within The Relational Database

The M:N Relationship

- The linking table (ENROLL) shown in Figure 3.26 yields the required M:N to 1:M conversion. Observe that the composite entity represented by the ENROLL table must contain at least the primary keys of the CLASS and STUDENT tables (CLASS_CODE and STU_NUM, respectively) for which it serves as a connector. Also note that the STUDENT and CLASS tables now contain only one row per entity. The linking ENROLL table contains multiple occurrences of the foreign key values, but those controlled redundancies are incapable of producing anomalies as long as referential integrity is enforced. Additional attributes may be assigned as needed.

Relationships Within The Relational Database

The M:N Relationship

- In this case, ENROLL_GRADE is selected to satisfy a reporting requirement. Also note that the ENROLL table's primary key consists of the two attributes CLASS_CODE and STU_NUM because both the class code and the student number are needed to define a particular student's grade. Naturally, the conversion is reflected in the ERM, too. The revised relationship is shown in Figure 3.27.
- As you examine Figure 3.27, note that the composite entity named ENROLL represents the linking table between STUDENT and CLASS.

Relationships Within The Relational Database

The M:N Relationship

- The 1:M relationship between COURSE and CLASS was first illustrated in Figure 3.20 and Figure 3.21. With the help of this relationship, you can increase the amount of available information even as you control the database's redundancies. Thus, Figure 3.27 can be expanded to include the 1:M relationship between COURSE and CLASS shown in Figure 3.28. Note that the model is able to handle multiple sections of a CLASS while controlling redundancies by making sure that all of the COURSE data common to each CLASS are kept in the COURSE table.

FIGURE 3.27 Changing the M:N relationship to two 1:M relationships

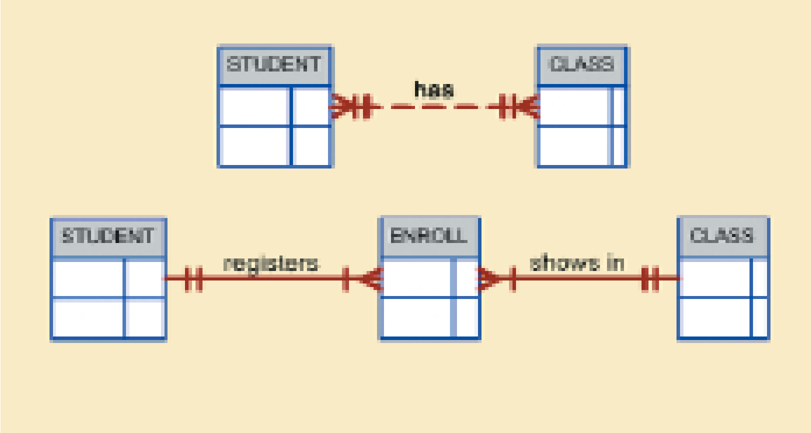


FIGURE 3.28 The expanded entity relationship model

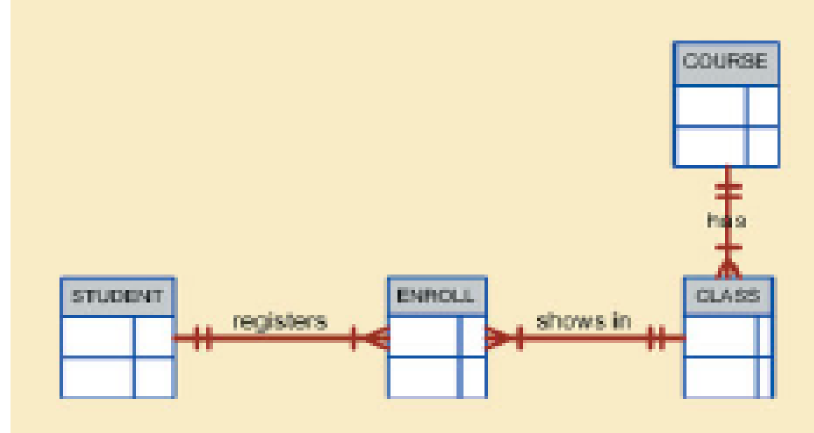
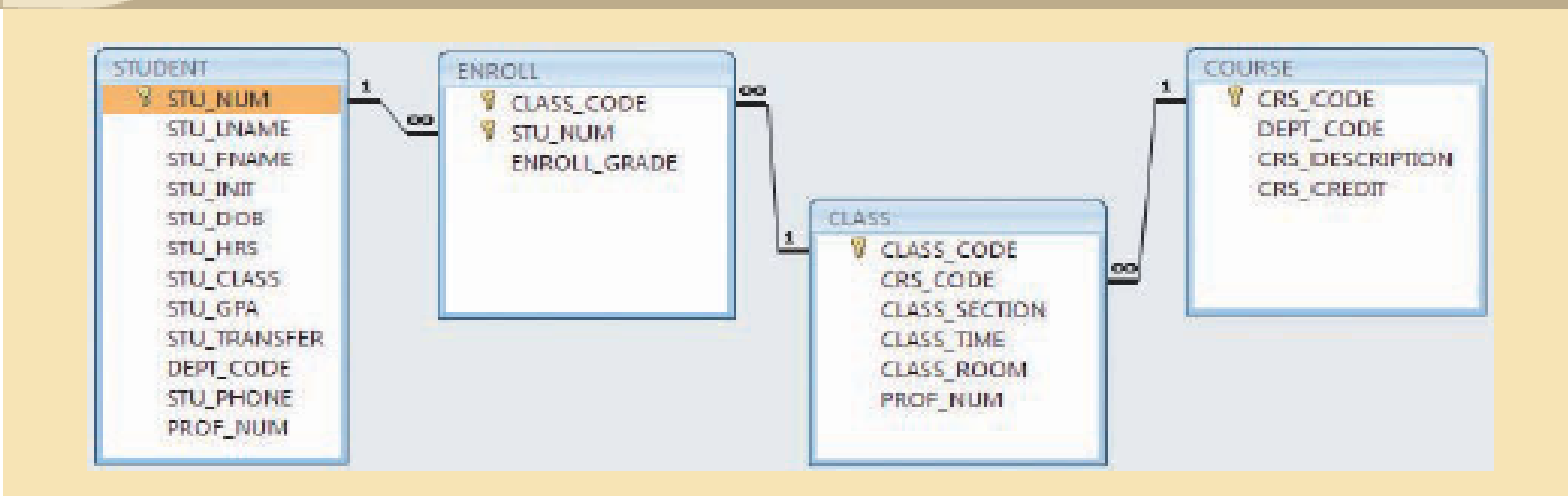


FIGURE 3.29 The relational diagram for the Ch03_TinyCollege database



End of Lecture:

Any Questions... ?