CSI2441: Applications Development

Lecture 8

Using Relational Databases

Objectives

- Understand relational database fundamentals
- Create databases and table descriptions
- Identify primary keys
- Understand database structure notation
- Understand the principles of adding, deleting, updating, and sorting records within a table

Objectives (continued)

- Write queries
- Understand relationships between tables and functional dependence between columns
- Recognize poor table design
- Understand anomalies, normal forms, and the normalization process
- Understand the performance and security issues connected to database administration

Understanding Relational Database Fundamentals

- **Data hierarchy**: stores data from smallest usable unit of data to the largest
  - Characters
  - Fields
  - Records
  - Tables (aka Files)
- **Database**:
  - Has group of files needed to support an organization
  - Files in a database are called **tables**

Understanding Relational Database Fundamentals (continued)

- Data in tables can be arranged in rows and columns
  - Each row represents an entire record in the table

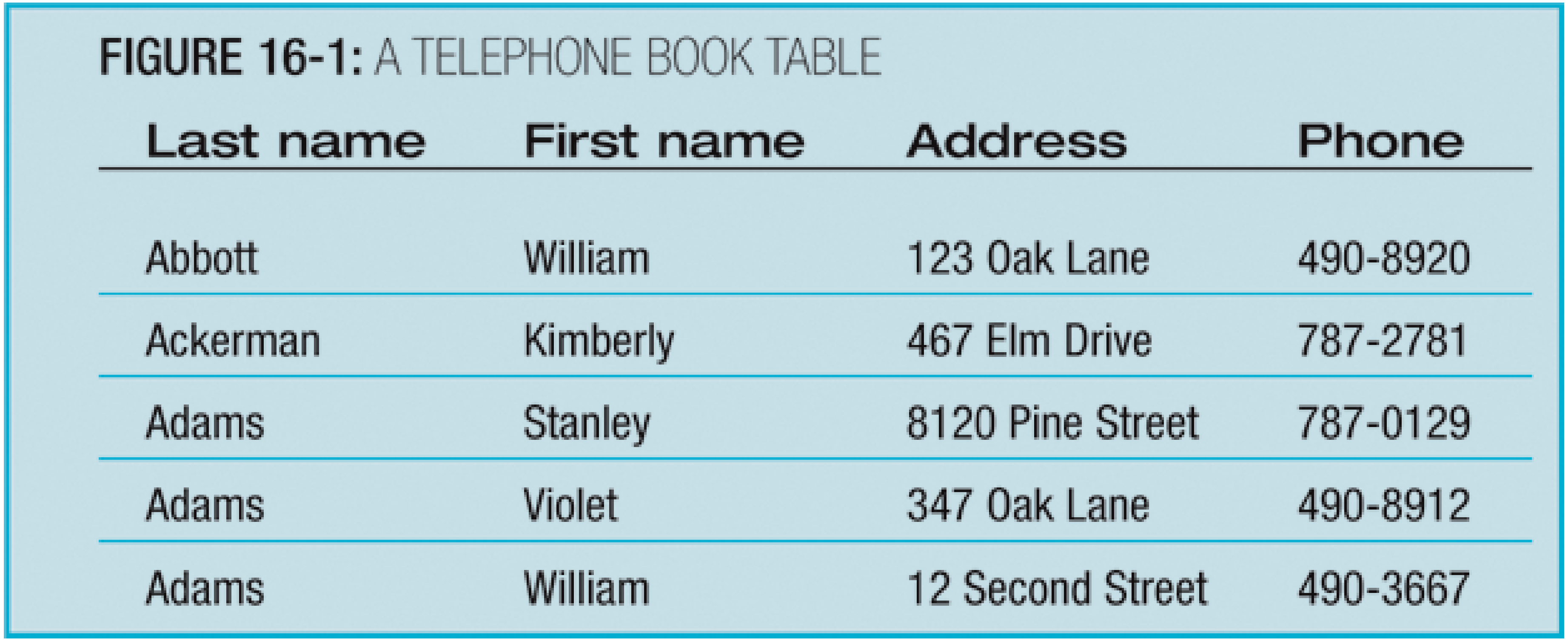


FIGURE 16-1: A TELEPHONE BOOK TABLE

Last name	First name	Address	Phone
Abbott	William	123 Oak Lane	490-8920
Ackerman	Kimberly	467 Elm Drive	787-2781
Adams	Stanley	8120 Pine Street	787-0129
Adams	Violet	347 Oak Lane	490-8912
Adams	William	12 Second Street	490-3667

Understanding Relational Database Fundamentals (continued)

- **Primary key** (or **key**):
  - Uniquely identifies a record
  - May be composed of one or multiple columns
  - Typically one
- **Compound key**: constructed from multiple columns

Understanding Relational Database Fundamentals (continued)

- **Database Management System** (**DBMS**) is software that allows you to:
  - Create table descriptions
  - Identify keys
  - Add, delete, and update records within a table
  - Sort records within a table by a specific field or fields
  - Write questions to select specific records for viewing
  - Write questions to combine information from multiple, related tables
  - Create reports
  - Secure the data
- On the three tier model, databases are typically 3rd tier or the Data Tier

Creating Databases and Table Descriptions

- Creating a database requires planning and analysis
  - What data to store
  - How to divide the data between tables
  - How the tables will interrelate
- Designing a database table:
  - Determine what columns are required and name them
  - Determine the type of data in each column

Creating Databases and Table Descriptions (continued)

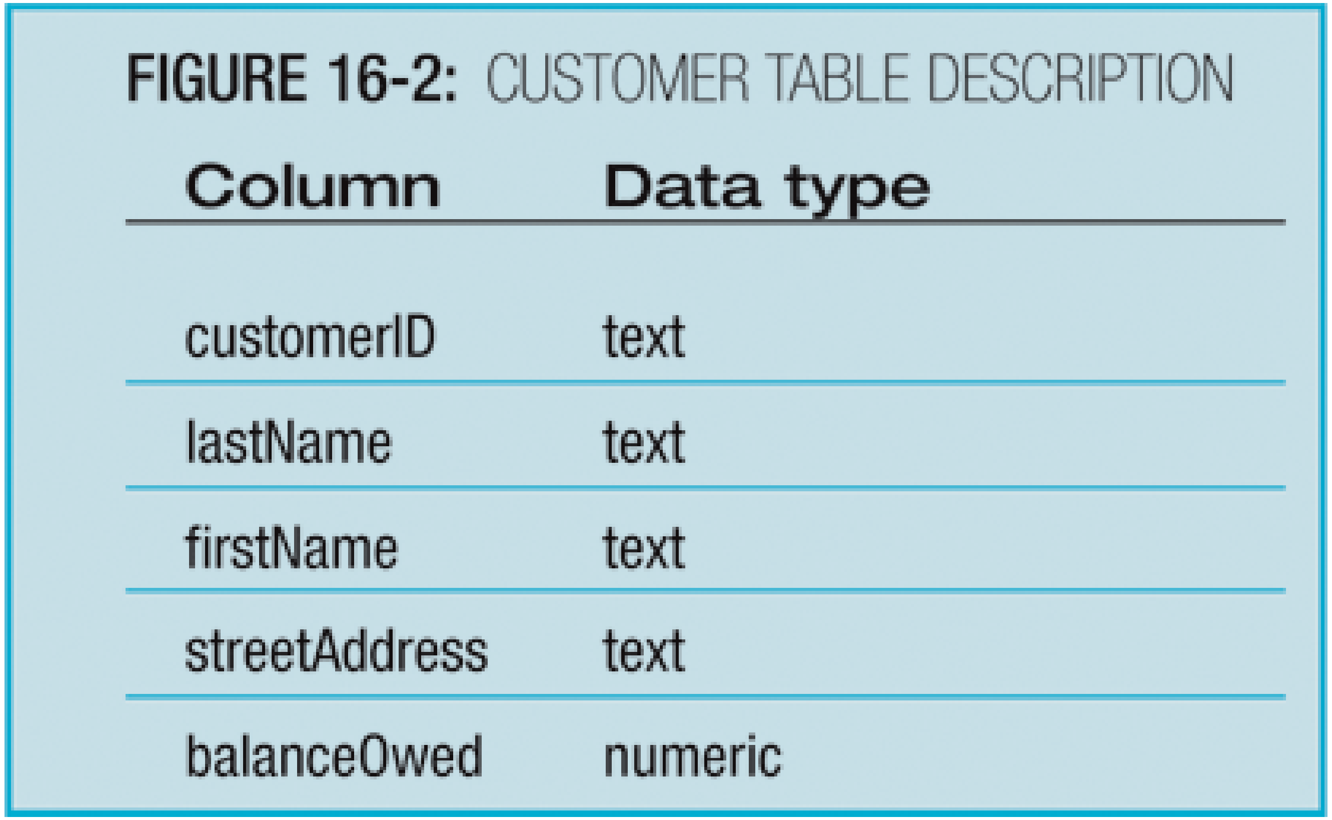


FIGURE 16-2: CUSTOMER TABLE DESCRIPTION

Column	Data type
customerID	text
lastName	text
firstName	text
streetAddress	text
balanceOwed	numeric

Identifying Primary Keys

- Identify a column or combination of columns to be the primary key
- Values of primary keys must be unique, such as:
  - Student ID number
  - Inventory part number
  - Social Security number
- In an environment where no such unique identifier exists, developers can use an incrementing primary key as managed by the database system

Identifying Primary Keys (continued)

- Primary key is used for:
  - Ensuring that multiple records with the same values cannot be added
  - Sorting the records in primary key order
  - Creating relationships between tables
  - Normalizing a database
- May need to use a multicolumn key to ensure unique values
- Or, as the previous slide indicated, you can use incrementing primary keys and then query the database BEFORE data insertion to check for existing data

Checking for Existing Data and Referential Integrity

- If you try to insert a value into a database which already exist (such as a Student ID in a Students table) the database will generate an error
- However, as developers it is often better not to rely on the database to do the checking (as the failure mode is often not very user friendly)
- For any insertion into the database, take the form data, then search the primary key or other main fields against the database to see if the record is already there
- Then a more specific error message can be thrown

Referential Integrity

- Most RDBMS systems can implement rules upon relationships
- An example of referential integrity might be that a foreign key value in a *Student_Enrolments* table (such as the Student_ID or UnitCode) cannot be inserted unless it exits as a primary key value in a related table (such as Student)
- This stops redundant data entering the system
- Not all database environments fully support referential integrity automatically

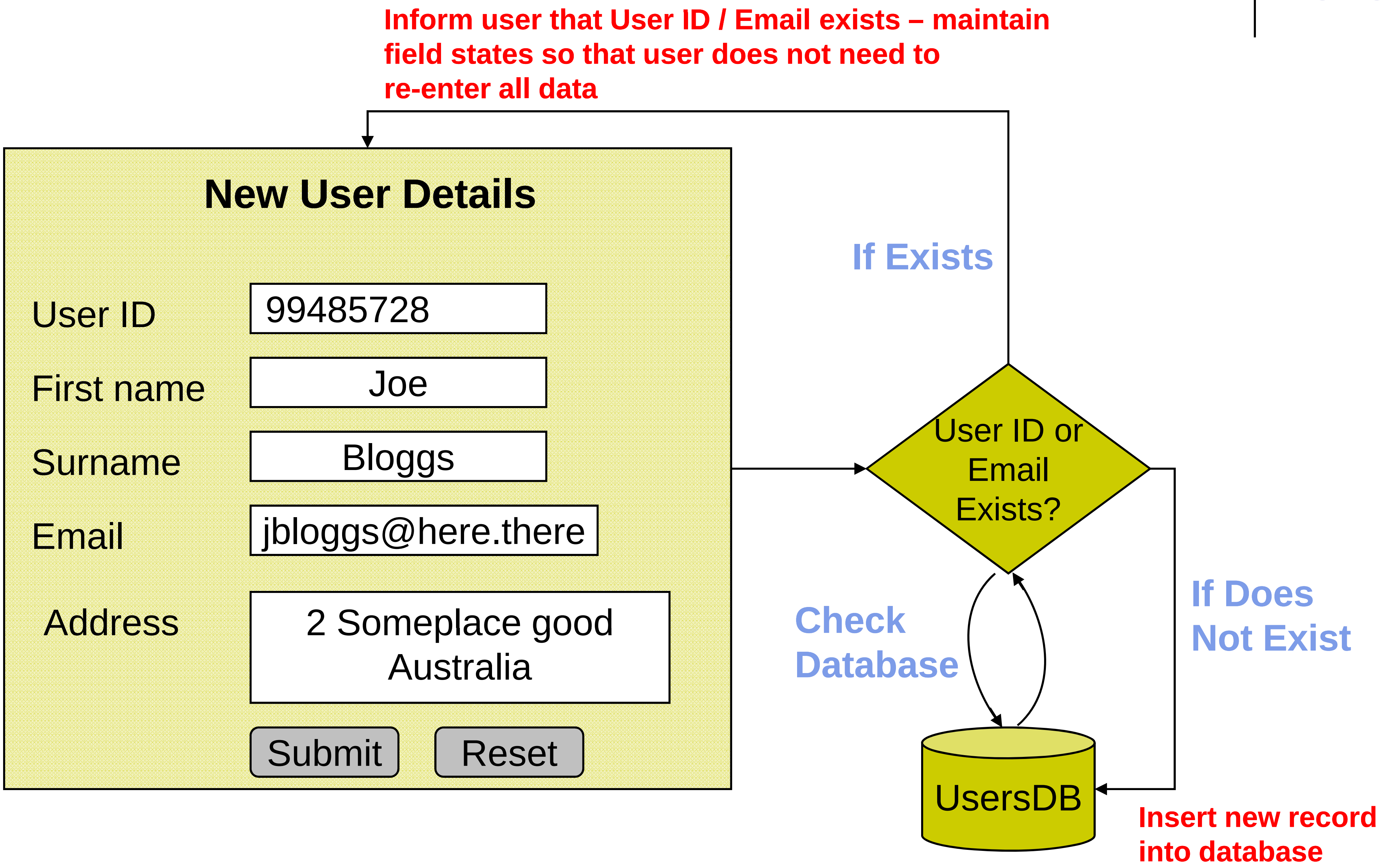
Programming Integrity

- Often it is better to encode referential integrity into your applications rather than rely on the database to do it
- This tends to increase the programming load
- Allows for application transportability – if you put more of the logic in the application and use less features of the database, the more likely it is to work with a different database

Key Management Issues

- As the previous slides have indicated, you can either define your own key or use the database to manage keys
- A classic mistake that budding developers use is to require key fields to be entered by the user – ie let users make them up
- In some instances, such as items tagged with a barcode, this is fine. However, if the end-user has to just make one up, problems will occur
- And in the same instance, if you are using automatically generated keys, in 95% of cases you do not need to show these to users

Key Management Issues



Inform user that User ID / Email exists – maintain field states so that user does not need to re-enter all data

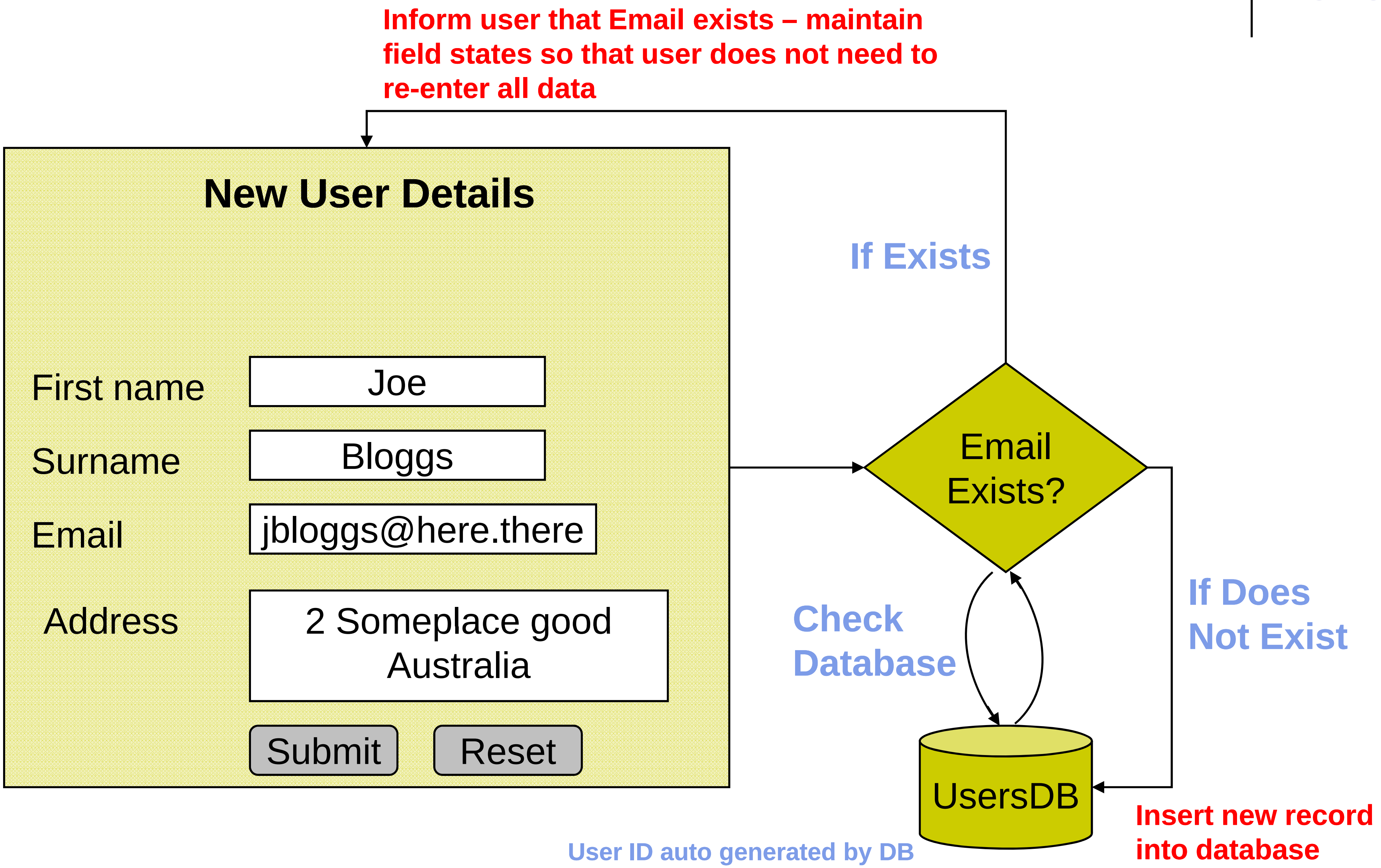
New User Details

User ID	99485728
First name	Joe
Surname	Bloggs
Email	jbloggs@here.there
Address	2 Someplace good Australia

Submit Reset

Insert new record into database

Key Management Issues



Inform user that Email exists – maintain field states so that user does not need to re-enter all data

New User Details

First name	Joe
Surname	Bloggs
Email	jbloggs@here.there
Address	2 Someplace good Australia

Submit Reset

Insert new record into database

User ID auto generated by DB

Adding, Deleting, and Updating Records Within Tables

- Adding data
  - Data types must match the column definitions
  - Database software may not permit blank values
- Records can be deleted from tables
- Fields within records can be modified
- Maintaining correct data at all times is extremely important
- The whole point of Relational Database Management Systems is to have a single instance of current information available from a centralised location

Sorting the Records in a Table

- Can sort a table based on any column
- After sorting:
  - Records can be grouped by specific values or ranges
  - Aggregate values can be calculated (counts, sums, averages, etc.)
- Data retrieved from tables can be formatted for display

Creating Queries

- **Query**: a question presented to the database which results in data being returned
- **Structured Query Language** (**SQL**): a common language used to query a database
- **SELECT-FROM-WHERE** is the basic form of a query:
  - Select which columns to use
  - Select the table from which to retrieve the data
  - Select records where one or more conditions are met
- Wildcard symbol can be used to specify “any” or “all”
- Can create compound conditions using AND or OR

Creating Queries (continued)

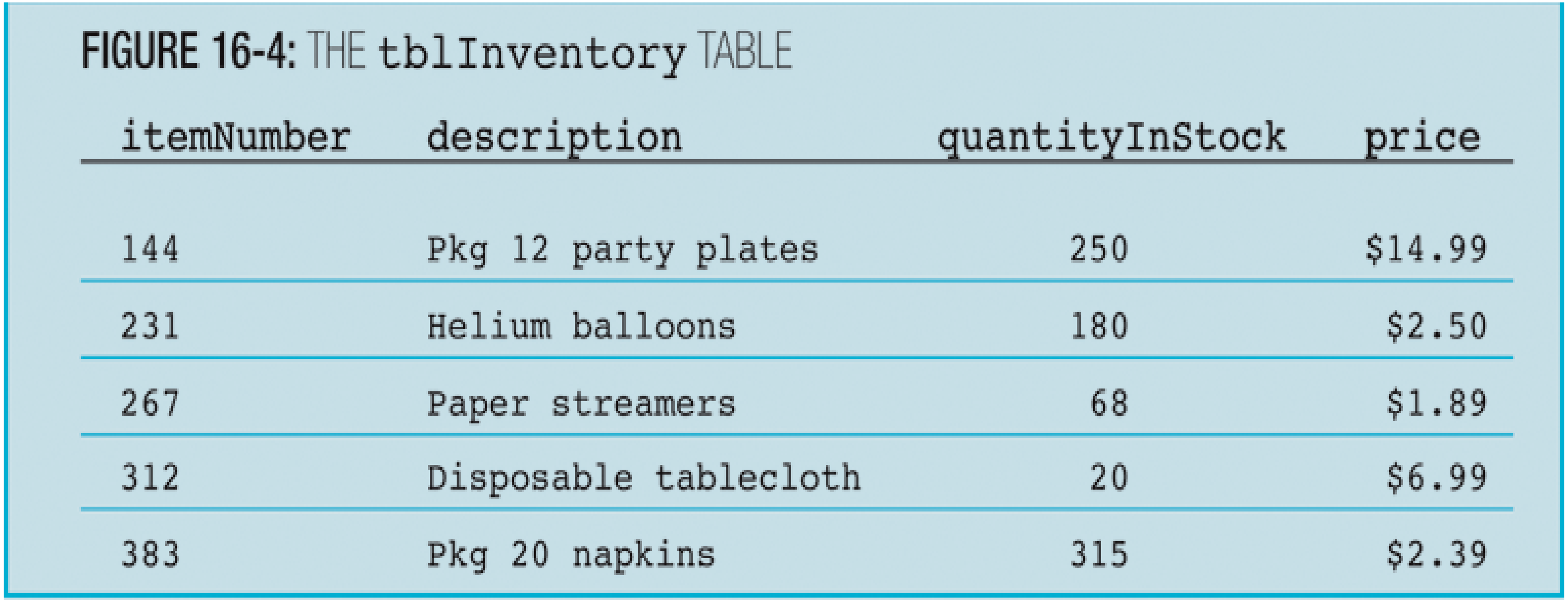


FIGURE 16-4: THE tblInventory TABLE

itemNumber	description	quantityInStock	price
144	Pkg 12 party plates	250	$14.99
231	Helium balloons	180	$2.50
267	Paper streamers	68	$1.89
312	Disposable tablecloth	20	$6.99
383	Pkg 20 napkins	315	$2.39

Creating Queries (continued)

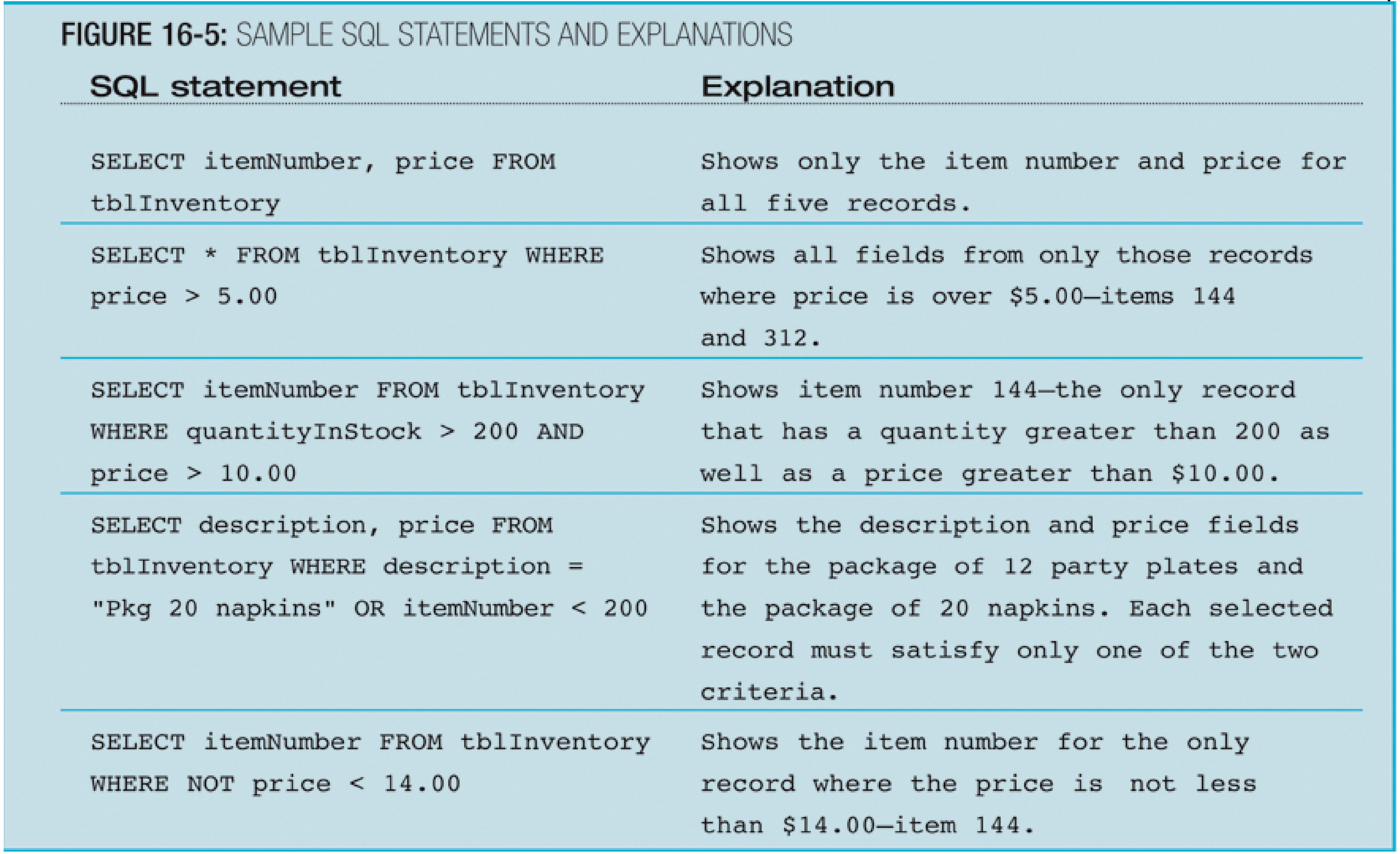


FIGURE 16-5: SAMPLE SQL STATEMENTS AND EXPLANATIONS

SQL statement	Explanation
`SELECT itemNumber, price FROM tblInventory`	Shows only the item number and price for all five records.
`SELECT * FROM tblInventory WHERE price > 5.00`	Shows all fields from only those records where price is over $5.00–items 144 and 312.
`SELECT itemNumber FROM tblInventory WHERE quantityInStock > 200 AND price > 10.00`	Shows item number 144–the only record that has a quantity greater than 200 as well as a price greater than $10.00.
`SELECT description, price FROM tblInventory WHERE description = "Pkg 20 napkins" OR itemNumber < 200`	Shows the description and price fields for the package of 12 party plates and the package of 20 napkins. Each selected record must satisfy only one of the two criteria.
`SELECT itemNumber FROM tblInventory WHERE NOT price < 14.00`	Shows the item number for the only record where the price is not less than $14.00–item 144.

Understanding Table Relationships

- **Relationship**: a connection between two tables
- **Relational database**: a database containing relationships
- **Join operation** (or **join**): connecting two tables based on values in a common column
- Query returns data taken from each joined table
- 3 types of relationships:
  - One-to-many
  - Many-to-many (try to avoid)
  - One-to-one

Understanding Table Relationships (continued)

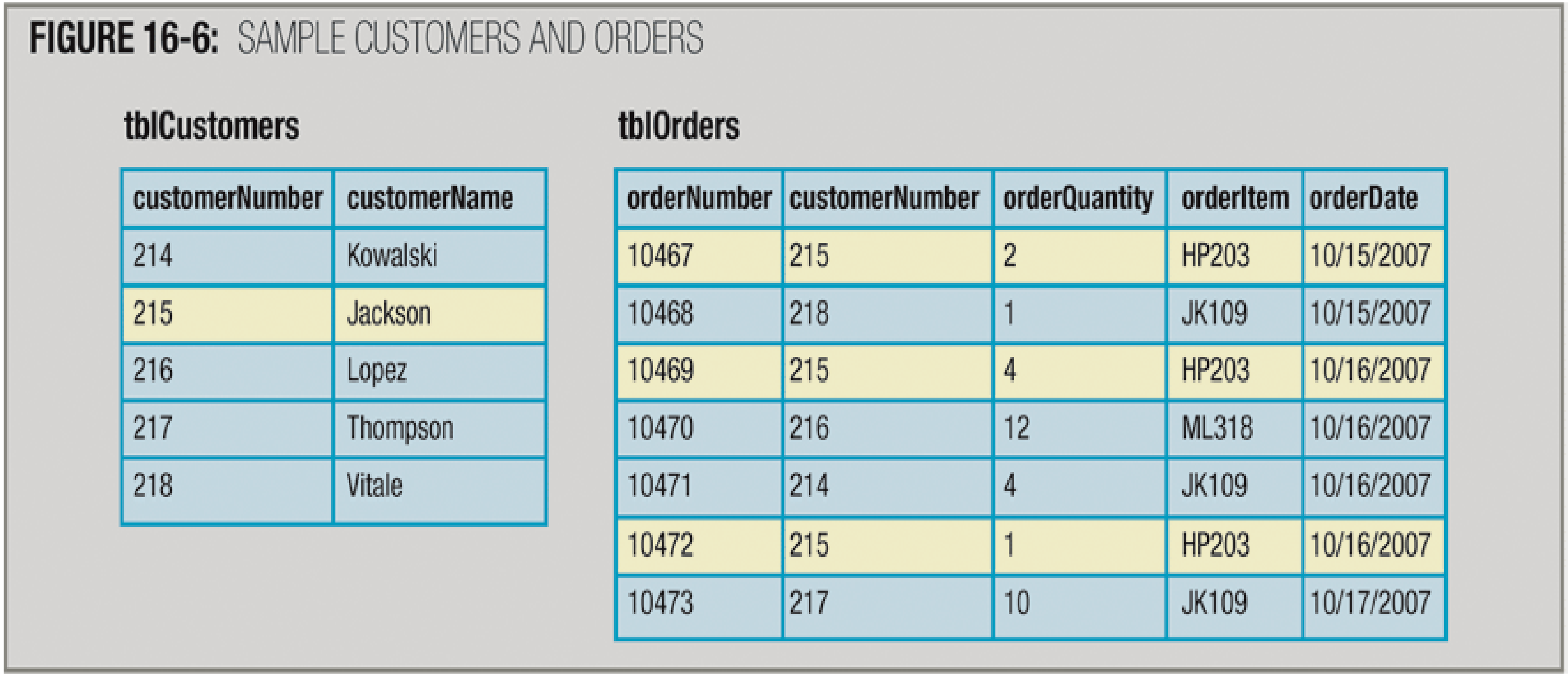


FIGURE 16-6: SAMPLE CUSTOMERS AND ORDERS

tblCustomers

customerNumber	customerName
214	Kowalski
215	Jackson
216	Lopez
217	Thompson
218	Vitale

tblOrders

orderNumber	customerNumber	orderQuantity	orderItem	orderDate
10467	215	2	HP203	10/15/2007
10468	218	1	JK109	10/15/2007
10469	215	4	HP203	10/16/2007
10470	216	12	ML318	10/16/2007
10471	214	4	JK109	10/16/2007
10472	215	1	HP203	10/16/2007
10473	217	10	JK109	10/17/2007

Understanding One-to-Many Relationships

- **One-to-many relationship**:
  - A row in one table is related to one or more rows in another table
  - Most common type of table relationship
- Relationship can be based on one or more columns
- On one side of the relationship, a table's primary key is used for the join
- On the other side, it may be a non-key column
- **Foreign key**: a field in a table which is also a primary key in another table

Understanding One-to-Many Relationships (continued)

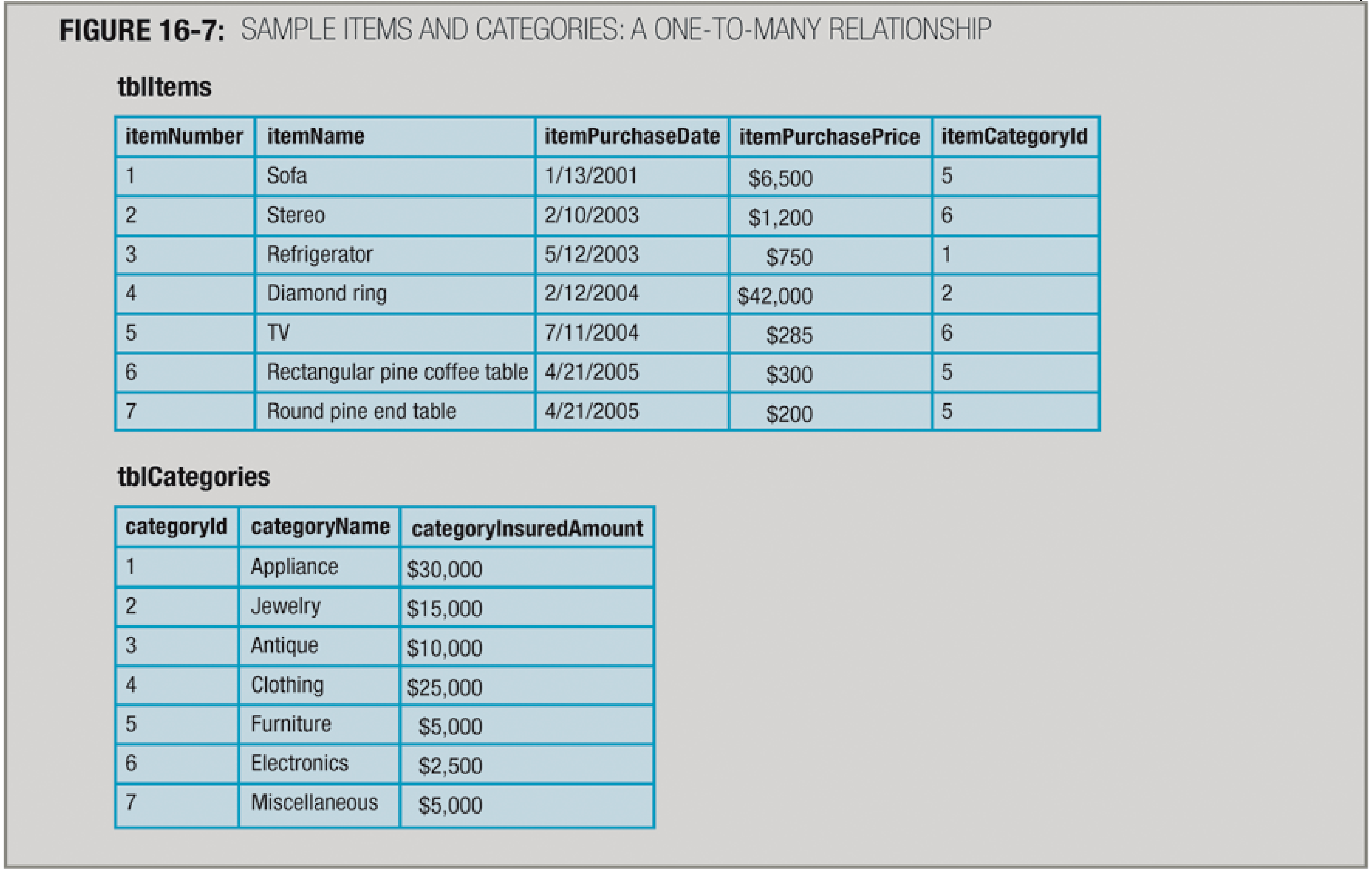


FIGURE 16-7: SAMPLE ITEMS AND CATEGORIES: A ONE-TO-MANY RELATIONSHIP

tblItems

itemNumber	itemName	itemPurchaseDate	itemPurchasePrice	itemCategoryId
1	Sofa	1/13/2001	$6,500	5
2	Stereo	2/10/2003	$1,200	6
3	Refrigerator	5/12/2003	$750	1
4	Diamond ring	2/12/2004	$42,000	2
5	TV	7/11/2004	$285	6
6	Rectangular pine coffee table	4/21/2005	$300	5
7	Round pine end table	4/21/2005	$200	5

tblCategories

categoryId	categoryName	categoryInsuredAmount
1	Appliance	$30,000
2	Jewelry	$15,000
3	Antique	$10,000
4	Clothing	$25,000
5	Furniture	$5,000
6	Electronics	$2,500
7	Miscellaneous	$5,000

Understanding Many-to-Many Relationships

- Many-to-many relationship:
  - Multiple rows in each table can correspond to multiple rows in the other table
- Use an additional table to contain the pairs of primary keys from each table
- These pairs form unique keys in the new table
- Sometimes called an intermediate table

Understanding Many-to-Many Relationships (continued)

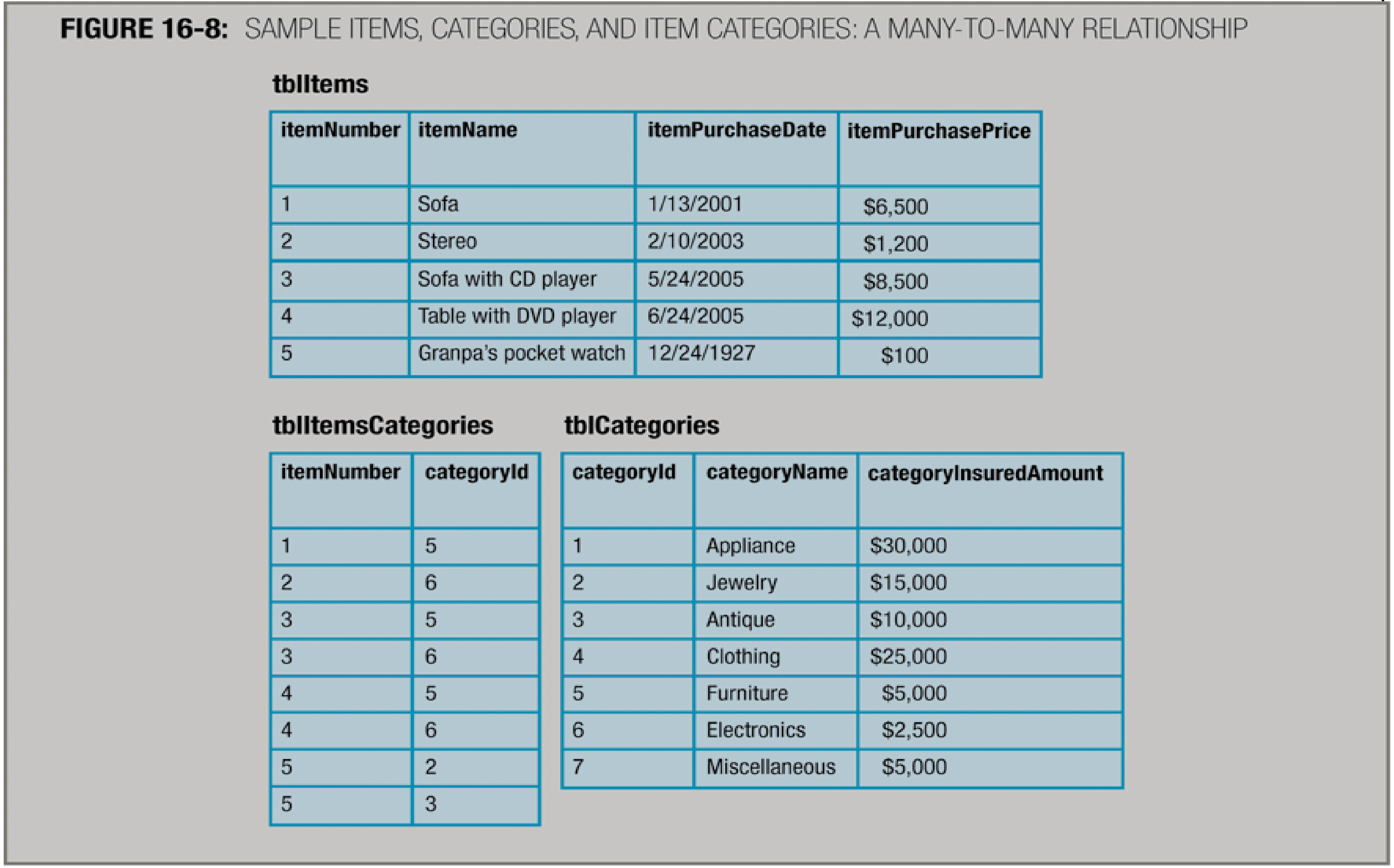


FIGURE 16-8: SAMPLE ITEMS, CATEGORIES, AND ITEM CATEGORIES: A MANY-TO-MANY RELATIONSHIP

tblItems

itemNumber	itemName	itemPurchaseDate	itemPurchasePrice
1	Sofa	1/13/2001	$6,500
2	Stereo	2/10/2003	$1,200
3	Sofa with CD player	5/24/2005	$8,500
4	Table with DVD player	6/24/2005	$12,000
5	Granpa's pocket watch	12/24/1927	$100

tblItemsCategories

itemNumber	categoryId
1	5
2	6
3	5
3	6
4	5
4	6
5	2
5	3

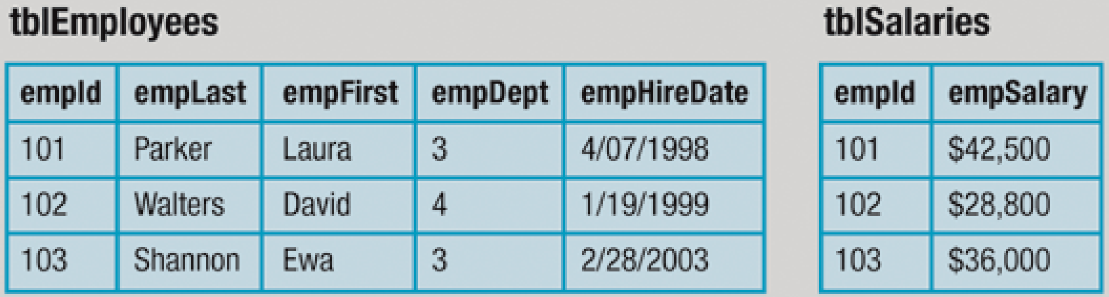
tblCategories

categoryId	categoryName	categoryInsuredAmount
1	Appliance	$30,000
2	Jewelry	$15,000
3	Antique	$10,000
4	Clothing	$25,000
5	Furniture	$5,000
6	Electronics	$2,500
7	Miscellaneous	$5,000

Understanding One-to-One Relationships

- **One-to-one relationship**:
  - A row in one table corresponds to exactly one row in another table
- One-to-one relationships indicate that the tables could be combined into a single table
- Often keep the tables separate for security purposes, such as salary below or password associated with a user account

FIGURE 16-9: EMPLOYEES AND SALARIES TABLES: A ONE-TO-ONE RELATIONSHIP



tblEmployees

empId	empLast	empFirst	empDept	empHireDate
101	Parker	Laura	3	4/07/1998
102	Walters	David	4	1/19/1999
103	Shannon	Ewa	3	2/28/2003

tblSalaries

empId	empSalary
101	$42,500
102	$28,800
103	$36,000

Recognizing Poor Table Design

- If tables are not designed correctly, the database may not support the needs of the application
- What are the shortcomings of this table design?

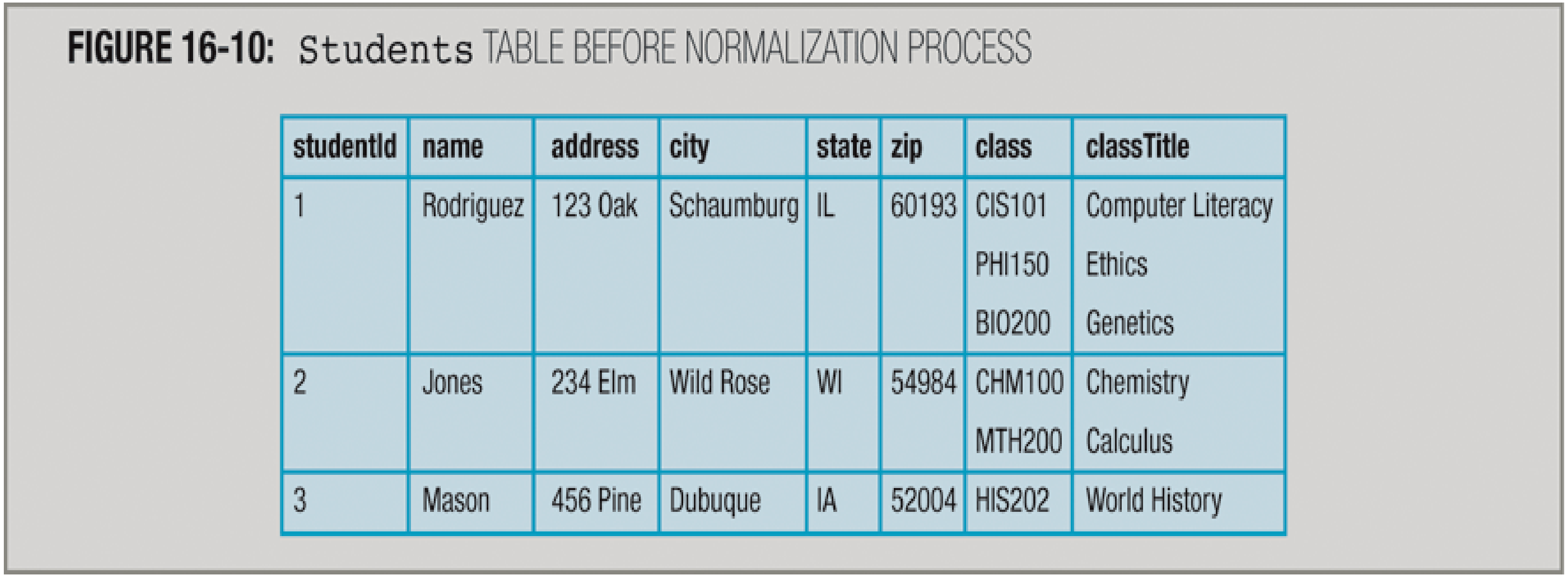


FIGURE 16-10: Students TABLE BEFORE NORMALIZATION PROCESS

studentId	name	address	city	state	zip	class	classTitle
1	Rodriguez	123 Oak	Schaumburg	IL	60193	CIS101 PHI150 BIO200	Computer Literacy Ethics Genetics
2	Jones	234 Elm	Wild Rose	WI	54984	CHM100 MTH200	Chemistry Calculus
3	Mason	456 Pine	Dubuque	IA	52004	HIS202	World History

Understanding Anomalies, Normal Forms, and the Normalization Process

- **Normalization**:
  - Process of designing and creating a database structure that satisfies needs
  - Helps reduce duplication of data
- **Data redundancy**: unnecessary duplication of data
  - Data appears in more than one place – example might be student name appearing in other tables aside from StudentDetails table
- **Anomaly**: irregularity in database design that causes problems
  - An example might be deleting data in a one-to-many relationship – if you delete the primary record but do not remove the foreign key records, those foreign keys could reference a primary key that no longer exist

Understanding Anomalies, Normal Forms, and the Normalization Process (continued)

- Three common types of anomalies:
  - Update anomalies
  - Delete anomalies
  - Insert anomalies
- **Update anomaly**: when updating data in one table, you must update the same data in another table
- **Delete anomaly**: deleting a record causes other problems, such as loss of unrelated information
- **Insert anomaly**: inability to add a new record due to lack of related data

Understanding Anomalies, Normal Forms, and the Normalization Process (continued)

- Normalization removes redundancies and anomalies
- Three normal forms:
  - **First normal form** (or **1NF**): eliminate repeating groups
  - **Second normal form** (or **2NF**): eliminate partial key dependencies
  - **Third normal form** (**3NF**): eliminate transitive dependencies

First Normal Form

- **Unnormalized**: a table that contains repeating groups
- **Repeating group**: a subset of rows in a table that all depend on the same key
- After eliminating repeating **`class`** and **`classTitle`**:

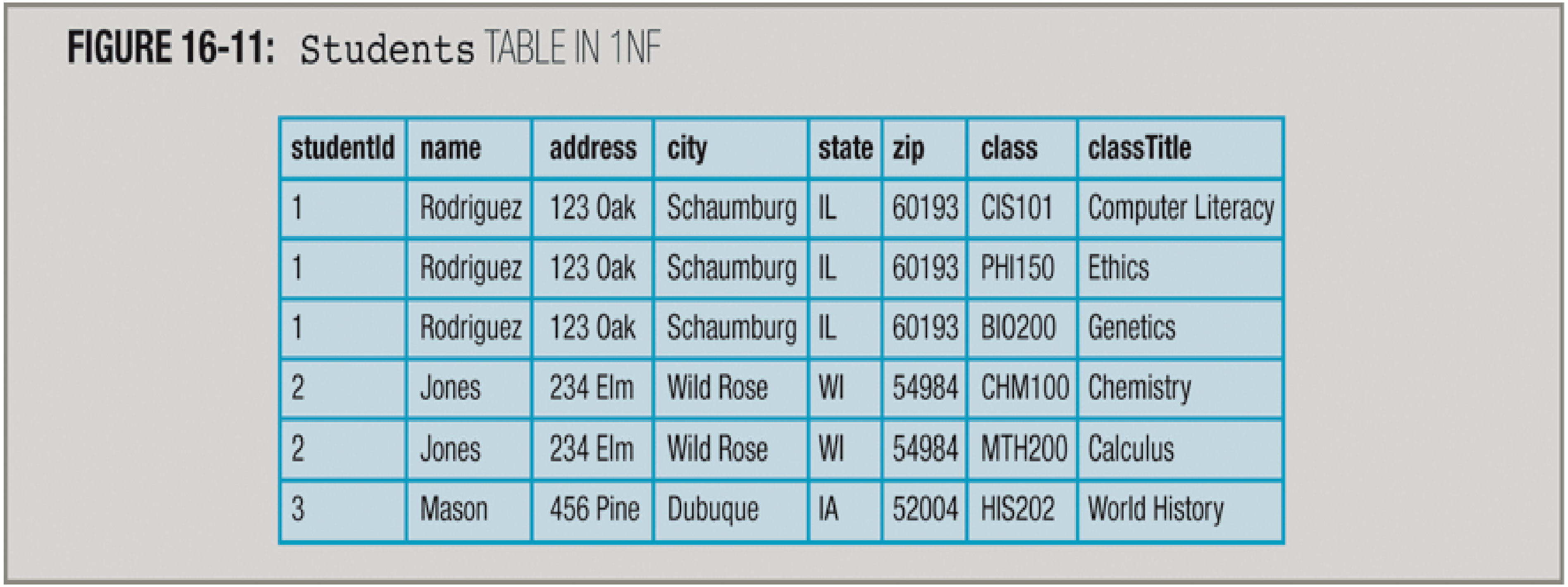


FIGURE 16-11: `Students` TABLE IN 1NF

studentId	name	address	city	state	zip	class	classTitle
1	Rodriguez	123 Oak	Schaumburg	IL	60193	CIS101	Computer Literacy
1	Rodriguez	123 Oak	Schaumburg	IL	60193	PHI150	Ethics
1	Rodriguez	123 Oak	Schaumburg	IL	60193	BIO200	Genetics
2	Jones	234 Elm	Wild Rose	WI	54984	CHM100	Chemistry
2	Jones	234 Elm	Wild Rose	WI	54984	MTH200	Calculus
3	Mason	456 Pine	Dubuque	IA	52004	HIS202	World History

First Normal Form (continued)

- When repeating groups are eliminated, you may have to change the key field if it is no longer unique
- Can use a compound key to solve this problem
- **Atomic attributes**: each attribute contains an undividable piece of data

Second Normal Form

- Partial key dependencies: when a column depends on only part of the key
- For 2NF:
  - Database must already be in 1NF
  - All non-key fields must be dependent on the entire primary key
- Eliminate partial key dependencies by creating multiple tables

Second Normal Form (continued)

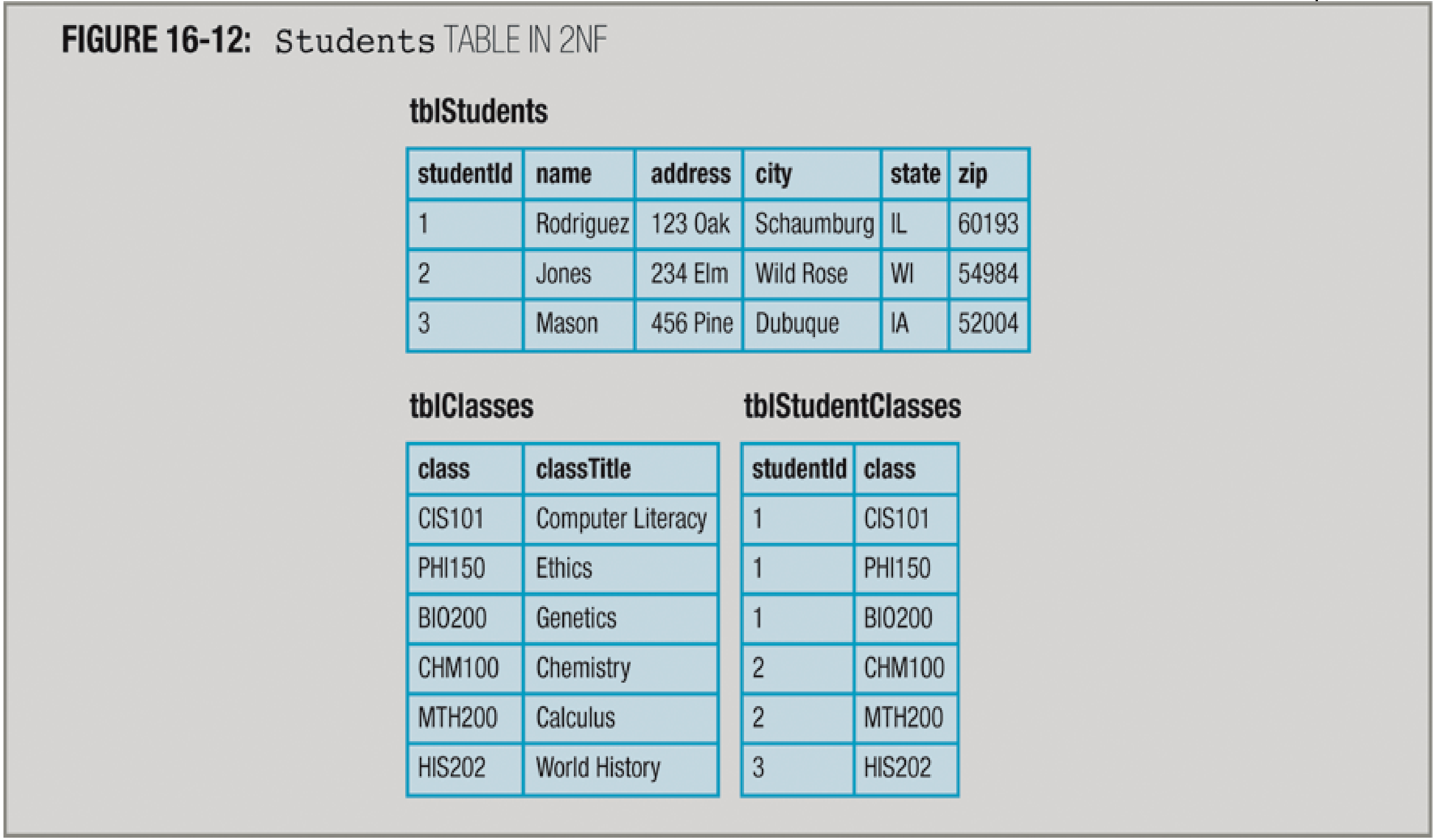


FIGURE 16-12: Students TABLE IN 2NF

tblStudents

studentId	name	address	city	state	zip
1	Rodriguez	123 Oak	Schaumburg	IL	60193
2	Jones	234 Elm	Wild Rose	WI	54984
3	Mason	456 Pine	Dubuque	IA	52004

tblClasses

class	classTitle
CIS101	Computer Literacy
PHI150	Ethics
BIO200	Genetics
CHM100	Chemistry
MTH200	Calculus
HIS202	World History

tblStudentClasses

studentId	class
1	CIS101
1	PHI150
1	BIO200
2	CHM100
2	MTH200
3	HIS202

Third Normal Form

- **Transitive dependency**: when the value of a non-key attribute determines or predicts the value of another non-key attribute
- For 3NF:
  - Database must already be in 2NF
  - No transitive dependencies
- Remove the attributes that are functionally dependent on the attribute that causes the transitive dependency

Third Normal Form (continued)

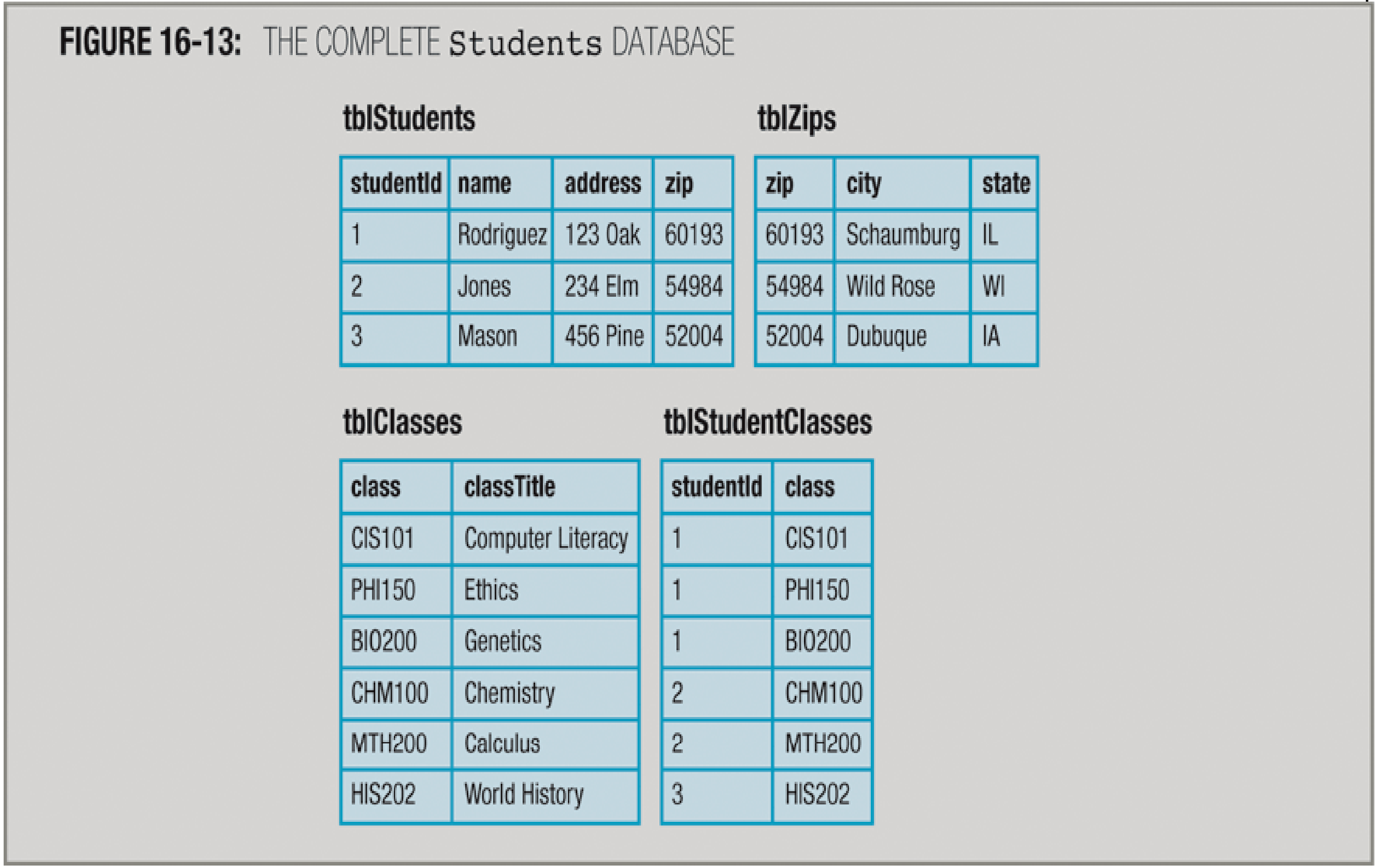


FIGURE 16-13: THE COMPLETE `Students` DATABASE

tblStudents

studentId	name	address	zip
1	Rodriguez	123 Oak	60193
2	Jones	234 Elm	54984
3	Mason	456 Pine	52004

tblZips

zip	city	state
60193	Schaumburg	IL
54984	Wild Rose	WI
52004	Dubuque	IA

tblClasses

class	classTitle
CIS101	Computer Literacy
PHI150	Ethics
BIO200	Genetics
CHM100	Chemistry
MTH200	Calculus
HIS202	World History

tblStudentClasses

studentId	class
1	CIS101
1	PHI150
1	BIO200
2	CHM100
2	MTH200
3	HIS202

Third Normal Form (continued)

- All redundancies and anomalies have now been removed
- Determinant is allowed in 3NF if it is a candidate key
- Normalization summary:
  - 1NF: no repeating groups
  - 2NF: 1NF plus no partial key dependencies
  - 3NF: 2NF plus no transitive dependencies

Database Performance and Security Issues

- A company’s data must be protected
- Data security includes:
  - Providing data integrity
  - Recovering lost data
  - Providing rollback features
  - Avoiding concurrent update problems
  - Providing authentication and permissions
  - Providing encryption
  - Providing an audit trail as to who accessed and altered what and when

Providing Data Integrity

- **Data integrity**:
  - Data is accurate and consistent
- Database software must enforce referential integrity
- Database enforces data type and data presence rules, such as if what type of data a field will accept, in what format and whether it can be left blank or not

Recovering Lost Data

- Data loss can be caused by:
  - User mistakes
  - Hackers or other malicious users
  - Hardware problems
  - Fire, flood, or other natural disasters
- **Recovery**: returning the database to a correct form that existed before the problem occurred
- Can use a backup copy of the database with a record of all transactions to recover a database
- **Transaction**: a change made to data in the database
  - Most modern RDBMS environments provide rollback facilities for one or more operations (difficult when applied across large amounts of ‘related’ records)

Avoiding Concurrent Update Problems

- **Concurrent update problem**:
  - When two users both need to make changes to the same record
  - If each user changes the data and saves the record, whose update will not be in the database?
- **Lock**: a mechanism to prevent changes to a database record for some period of time
  - This is a real problem in file-based databases as most operating systems lock a file open to only one user at a time
- Solving concurrent update problem:
  - Use record-level locking
  - Make transactions off-line, and process as a batch

Providing Authentication and Permissions

- Database software must determine that a user is legitimate and is authorized to use the database
- Authentication techniques include:
  - Storing and verifying passwords
  - Using biometric data to identify users
- **Permissions**: settings that determine what actions a user is allowed to perform
- Authentication determines what permissions a user has
- Web applications typically deal with security at two levels
  - The access the application has to the database and the permissions on the connection
  - The access the application provides to its user based on its business rules and user account stored in the database
  - The two are ‘usually’ separate

Providing Encryption

- **Encryption**: coding data into a format that humans cannot read
- Prevents use of the data by unauthorized users even if they gain access to the database itself
- However, when in ‘development’ phase it is often not a good idea to turn this on as it means you cannot edit the database directly if you need to read and verify a record manually
- Also, if the database is lost and is recovered, some environments need the original encryption key(s) to recover the data – if the keys are lost, recovered data may be forever unreadable
  - http://searchsecurity.techtarget.co.uk/news/article/0,289142,sid180_gci1372414,00.html

Summary

- Database: collection of tables containing an organization's data
- Primary key: value that uniquely identifies a record
- Database management software allows you to add, delete, and update records in the database
- Query: question that selects data from database
- Database creation requires planning and analysis

Summary (continued)

- Primary key can consist of one or multiple columns
- Most data is in a constant state of change
- Can sort a table based on any column
- Can do aggregate calculations on data
- Normalization: designing a database to meet stated needs yet avoiding redundancies and anomalies
- Three forms of normalization are commonly used

Summary (continued)

- Database may be one of a company’s most important assets, so it must be secured
- Security issues: data integrity, recovery, avoiding concurrent update problems, authentication and permissions, and providing encryption

Readings

- Farrell, J. (2006). Programming Logic and Design Comprehensive, 4th Ed. Thomson : Boston. Chapter 16