David M. Kroenke and David J. Auer
Database Processing:
Fundamentals, Design, and Implementation

Chapter Four:

Database Design Using Normalization

Chapter Objectives

- To design updatable databases to store data received from another source
- To use SQL to access table structure
- To understand the advantages and disadvantages of normalization
- To understand denormalization
- To design read-only databases to store data from updateable databases

Chapter Objectives

- To recognize and be able to correct common design problems:
  - The multivalue, multicolumn problem
  - The inconsistent values problem
  - The missing values problem
  - The general-purpose remarks column problem

Chapter Premise

- We have received one or more tables of existing data.
- The data is to be stored in a new database.
- QUESTION: Should the data be stored as received, or should it be transformed for storage?

How Many Tables?

SKU_DATA (SKU, SKU_Description, *Buyer*)
BUYER (Buyer, Department)

Where SKU_DATA.Buyer must exist in BUYER.Buyer

Should we store these two tables as they are, or should we combine them into one table in our new database?

SKU_DATA

	SKU	SKU_Description	Buyer
1	100100	Std. Scuba Tank, Yellow	Pete Hansen
2	100200	Std. Scuba Tank, Magenta	Pete Hansen
3	101100	Dive Mask, Small Clear	Nancy Meyers
4	101200	Dive Mask, Med Clear	Nancy Meyers
5	201000	Half-dome Tent	Cindy Lo
6	202000	Half-dome Tent Vestibule	Cindy Lo
7	301000	Light Fly Climbing Harness	Jerry Martin
8	302000	Locking carabiner, Oval	Jerry Martin

BUYER

	Buyer	Department
1	Cindy Lo	Camping
2	Jerry Martin	Climbing
3	Nancy Meyers	Water Sports
4	Pete Hansen	Water Sports

Assessing Table Structure

Guidelines for Assessing Table Structure
• Count rows and examine columns
• Examine data values and interview users to determine:
Multivalued dependencies
Functional dependencies
Candidate keys
Primary keys
Foreign keys
• Assess validity of assumed referential integrity constraints

Counting Rows in a Table

- To count the number of rows in a table use the **SQL COUNT(*) built-in aggregate function** :

```
SELECT   COUNT(*) AS SKU_DATA_NumRows
FROM     SKU_DATA;
```

	SKU_DATA_NumRows
1	8

Examining the Columns

- To determine the number and type of columns in a table, use an SQL SELECT statement.
  - To limit the number of rows retrieved, use the **SQL TOP {NumberOfRows} function**:

```
SELECT   TOP 5 *
FROM     SKU_DATA;
```

	SKU	SKU_Description	Buyer
1	100100	Std. Scuba Tank, Yellow	Pete Hansen
2	100200	Std. Scuba Tank, Magenta	Pete Hansen
3	101100	Dive Mask, Small Clear	Nancy Meyers
4	101200	Dive Mask, Med Clear	Nancy Meyers
5	201000	Half-dome Tent	Cindy Lo

Checking Validity of Assumed Referential Integrity Constraints II

- To find any foreign key values that violate the foreign key constraint

  SKU_DATA_2 (SKU, SKU_Description, *Buyer*)
  BUYER (Buyer, Department)
  Where SKU_DATA_2.Buyer must exist in BUYER.Buyer

  – An **empty set** for the query result indicates that *no* foreign key values violate the foreign key constraint

```
SELECT    Buyer
FROM      SKU_DATA
WHERE     Buyer NOT IN
( SELECT Buyer
  FROM   BUYER ) ;
```

Buyer

Type of Database

- Updateable database, or read-only database?
- If updateable database, we normally want tables in BCNF.
- If read-only database, we may not use BCNF tables.

Designing Updatable Databases

- **Updatable databases** are typically the operational databases of a company, such as the **online transaction processing (OLTP) system** discussed for Cape Codd Outdoor Sports at the beginning of Chapter 2.
- If you are constructing an updatable database, then you need to be concerned about modification anomalies and inconsistent data.
- Consequently, you must carefully consider normalization principles.

Normalization: Advantages and Disadvantages

Advantages and Disadvantages of Normalization
• Advantages
Eliminate modification anomalies
Reduce duplicated data
• Eliminate data integrity problems
• Save file space
Single table queries will run faster
• Disadvantages
More complicated SQL required for multitable subqueries and joins
Extra work for DBMS can mean slower applications

Non-Normalized Table: EQUIPMENT_REPAIR

EQUIPMENT_REPAIR

	ItemNumber	EquipmentType	AcquisitionCost	RepairNumber	RepairDate	RepairCost
1	100	Drill Press	3500.00	2000	2015-05-05	375.00
2	200	Lathe	4750.00	2100	2015-05-07	255.00
3	100	Drill Press	3500.00	2200	2015-06-19	178.00
4	300	Mill	27300.00	2300	2015-06-19	1875.00
5	100	Drill Press	3500.00	2400	2015-07-05	0.00
6	100	Drill Press	3500.00	2500	2015-08-17	275.00

1. Deletion anomaly
2. Addition anomaly
3. Data integrity

Normalized Tables: ITEM and REPAIR

EQUIPMENT_ITEM

	ItemNumber	EquipmentType	AcquisitionCost
1	100	Drill Press	3500.00
2	200	Lathe	4750.00
3	300	Mill	27300.00

REPAIR

	RepairNumber	ItemNumber	RepairDate	RepairCost
1	2000	100	2015-05-05	375.00
2	2100	200	2015-05-07	255.00
3	2200	100	2015-06-19	178.00
4	2300	300	2015-06-19	1875.00
5	2400	100	2015-07-05	0.00
6	2500	100	2015-08-17	275.00

Copying Data to New Tables

- To copy data from one table to another, use the **SQL INSERT statement**:

```
/* *** SQL-IN                          */
INSERT INTO EQUIPMENT_ITEM
    SELECT      DISTINCT ItemNumber, EquipmentType, AcquisitionCost
    FROM        EQUIPMENT_REPAIR;

/* *** SQL-                            * */
INSERT INTO REPAIR
    SELECT      RepairNumber, ItemNumber, RepairDate, RepairCost
    FROM        EQUIPMENT_REPAIR;
```

Table Name

Table Name

資料來源

資料來源

Final Steps

- In Chapters 7 and 8, you will learn how to:
  - Remove unneeded tables after the data is copied, using the **SQL DROP TABLE statement**.
  - Create the referential integrity constraint, using the **SQL ALTER TABLE statement**.

Choosing Not To Use BCNF

- BCNF is used to control anomalies from functional dependencies.
- There are times when BCNF is not desirable.
- The classic example is ZIP codes:
  - ZIP codes almost never change.
  - Any anomalies are likely to be caught by normal business practices.
  - Not having to use SQL to join data in two tables will speed up application processing.

Multivalued Dependencies

- Anomalies from multivalued dependencies are very problematic.
- ***Always*** place the columns of a multivalued dependency into a **separate table** (**4NF**).

Designing Read-Only Databases

- The extracted sales data that we used for **Cape Codd Outdoor Sports** in Chapter 2 is a small, but typical example of a read-only database.
- Read-only databases are used in **business intelligence (BI) systems** for producing information for assessment, analysis, planning, and control, as we discussed for Cape Codd Outdoor Sports in Chapter 2.
- Read-only databases are commonly used in a **data warehouse**, which we also introduced in Chapter 2.

Read-Only Databases

- **Read-only databases** are nonoperational databases using data extracted from operational databases.
- They are used for querying, reporting, and data mining applications.
- They are never updated (in the operational database sense—they may have new data imported from time to time).

Denormalization

- For read-only databases, normalization is seldom an advantage.
  - Application processing speed is more important.
- **Denormalization** is the joining of the data in normalized tables prior to storing the data.
- The data is then stored in nonnormalized tables.

Normalized Tables

STUDENT

	StudentID	StudentName
1	100	Jones
2	200	Davis
3	300	Garrett
4	400	Jones

ACTIVITY

	Activity	ActivityFee
1	Golf	65.00
2	Skiing	200.00
3	Swimming	50.00

PAYMENT

	StudentID	Activity	ActivityFee
1	100	Golf	65.00
2	100	Skiing	200.00
3	200	Skiing	200.00
4	200	Swimming	50.00
5	300	Skiing	200.00
6	300	Swimming	50.00
7	400	Golf	65.00
8	400	Swimming	50.00

Denormalizing the Data

```
/* *** SQL-INSERT-CH04-03 *** */
INSERT INTO STUDENT_ACTIVITY_PAYMENT_DATA
     SELECT     STUDENT.StudentID, StudentName,
                ACTIVITY.Activity, ActivityFee,
                AmountPaid
     FROM       STUDENT, PAYMENT, ACTIVITY
     WHERE      STUDENT.StudentID = PAYMENT.StudentID
         AND    PAYMENT.Activity = ACTIVITY.Activity;
```

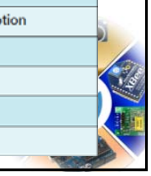
STUDENT_ACTIVITY_PAYMENT_DATA

	StudentID	StudentName	Activity	ActivityFee	AmountPaid
1	100	Jones	Golf	65.00	65.00
2	100	Jones	Skiing	200.00	0.00
3	200	Davis	Skiing	200.00	0.00
4	200	Davis	Swimming	50.00	50.00
5	300	Garrett	Skiing	200.00	100.00
6	300	Garrett	Swimming	50.00	50.00
7	400	Jones	Golf	65.00	65.00
8	400	Jones	Swimming	50.00	50.00

Customized Tables I

- Read-only databases are often designed with many copies of the same data, but with each copy customized for a specific application.
- Consider the PRODUCT table:

Product
• SKU (Primary Key)
• PartNumber (Candidate key)
• SKU_Description (Candidate key)
• VendorNumber
• VendorName
• VendorContact_1
• VendorContact_2
• VendorStreet
• VendorCity
• VendorState
• VendorZip
• QuantitySoldPastYear
• QuantitySoldPastQuarter
• QuantitySoldPastMonth
• DetailPicture
• ThumbNailPicture
• MarketingShortDescription
• MarketingLongDescription
• PartColor
• UnitsCode
• BinNumber
• ProductionKeyCode



Customized Tables II

PRODUCT_PURCHASING (SKU, SKU_Description, VendorNumber, VendorName, VendorContact_1, VendorContact_2, VendorStreet, VendorCity, VendorState, VendorZIP)

PRODUCT_USAGE (SKU, SKU_Description, QuantitySoldPastYear, QuantitySoldPastQuarter, QuantitySoldPastMonth)

PRODUCT_WEB (SKU, DetailPicture, ThumbnailPicture, MarketingShortDescription, MarketingLongDescription, PartColor)

PRODUCT_INVENTORY (SKU, PartNumber, SKU_Description, UnitsCode, BinNumber, ProductionKeyCode)

Common Design Problems

Practical Problems in Designing Databases from Existing Data
The multivalue, multicolumn problem
Inconsistent values
Missing values
General-purpose remarks column

The Multivalue, Multicolumn Problem

- The **multivalue, multicolumn problem** occurs when multiple values of an attribute are stored in more than one column:

EMPLOYEE (EmployeeNumber, EmployeeLastName, EmployeeFirstName, Email, Auto1_LicenseNumber, Auto2_LicenseNumber, Auto3_LicenseNumber)

- This is another form of a **multivalued dependency**.
- Solution = like the **4NF** solution for multivalued dependencies, use a separate table to store the multiple values.

Inconsistent Values I

- **Inco**erent user
sligh ata
valu
  - Di
    - SKU_Description = 'Large Can Corn'
  - Different spellings:
    - Coffee, Cofee, Coffeee

Practical Problems in Designing Databases from Existing Data

- The multivalue, multicolumn problem
- Inconsistent values
- Missing values
- General-purpose remarks column

Inconsistent Values II

- Particularly problematic are primary or foreign key values.
- To detect:
  - Use referential integrity check already discussed for checking keys.
  - Use the **SQL GROUP BY clause** on suspected columns.

Group by check

```
/* *** SQL-Query-CH04-05 *** */
SELECT     SKU_Description, COUNT(*) as SKU_Description_Count
FROM       SKU_DATA
GROUP BY   SKU_Description;
```

此為檢查?
加SKU呢?

	SKU_Description	SKU_Description_Count
1	Dive Mask, Med Clear	1
2	Dive Mask, Small Clear	1
3	Half-dome Tent	1
4	Half-dome Tent Vestibule	1
5	Light Fly Climbing Harness	1
6	Locking Carabiner, Oval	1
7	Std. Scuba Tank, Magenta	1
8	Std. Scuba Tank, Yellow	1

Missing Values

- A m alue that
  - In s in up

Practical Problems in Designing Databases from Existing Data
The multivalue, multicolumn problem
Inconsistent values
Missing values
General-purpose remarks column

Null Values

- Null values are ambiguous:
  - May indicate that a value is inappropriate;
    - DateOfLastChildbirth is inappropriate for a male.
  - May indicate that a value is appropriate but unknown;
    - DateOfLastChildbirth is appropriate for a female, but may be unknown.
  - May indicate that a value is appropriate and known, but has never been entered;
    - DateOfLastChildbirth is appropriate for a female, and may be known but no one has recorded it in the database.

Checking for Null Values

- Use the **SQL IS NULL operator** to check for null values:

```
/* *** SQL-Query-CH04-06 *** */
SELECT      COUNT (*) as QuantityNullCount
FROM        ORDER_ITEM
WHERE       Quantity IS NULL;
```

	QuantityNullCount
1	0

The General-Purpose Remarks Column

- A ge
  colur
  - Re
  - Co
  - No
- It ofte n
  incor
  - A t
    interests.
- Such a column may:
  - Be used inconsistently
  - Hold multiple data items

Practical Problems in Designing Databases from Existing Data
The multivalue, multicolumn problem
Inconsistent values
Missing values
General-purpose remarks column

The General-Purpose Remarks Column: Hidden Foreign Key Data

CONTACT (ContactID, ContactLastName, ContactFirstName, Address, . . . {other data}, Remarks, *AirplaneModelID*)

AIRPLANE_MODEL (AirplaneModelID, AirplaneModelName, AirplaneModelDescription, . . . {other airplane model data})

- In a typical situation, the data for the foreign key may have been recorded in the Remarks column.
  - 'Wants to buy a Piper Seneca II'
  - 'Owner of a Piper Seneca II'
  - 'Possible buyer for a turbo Seneca'.

Identify the different purposes of the remarks: ex. Own, Wants, Possible

David Kroenke and David Auer

Database Processing

Fundamentals, Design, and Implementation

(14th Edition, Global Edition)

End of Presentation:

Chapter Four