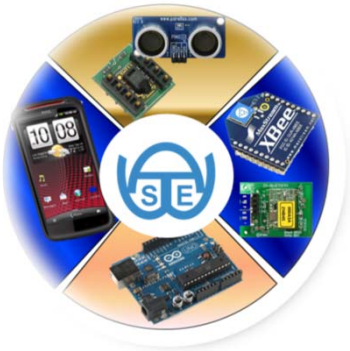
David M. Kroenke and David J. Auer
Database Processing:
Fundamentals, Design, and Implementation



Chapter Three:

The Relational Model and Normalization

Chapter Objectives

- To understand basic relational terminology
- To understand the characteristics of relations
- To understand alternative terminology used in describing the relational model
- To be able to identify functional dependencies, determinants, and dependent attributes
- To identify primary, candidate, and composite keys

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?

ORDER_ITEM

	OrderNumber	SKU	Quantity	Price	ExtendedPrice
1	1000	201000	1	300.00	300.00
2	1000	202000	1	130.00	130.00
3	2000	101100	4	50.00	200.00
4	2000	101200	2	50.00	100.00
5	3000	100200	1	300.00	300.00
6	3000	101100	2	50.00	100.00
7	3000	101200	1	50.00	50.00

SKU_DATA

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

SKU_ITEM

	OrderNumber	SKU	Quantity	Price	SKU_Description	Department	Buyer
1	1000	201000	1	300.00	Half-dome Tent	Camping	Cindy Lo
2	1000	202000	1	130.00	Half-dome Tent Vestibule	Camping	Cindy Lo
3	2000	101100	4	50.00	Dive Mask, Small Clear	Water Sports	Nancy Meyers
4	2000	101200	2	50.00	Dive Mask, Med Clear	Water Sports	Nancy Meyers
5	3000	100200	1	300.00	Std. Scuba Tank, Magenta	Water Sports	Pete Hansen
6	3000	101100	2	50.00	Dive Mask, Small Clear	Water Sports	Nancy Meyers
7	3000	101200	1	50.00	Dive Mask, Med Clear	Water Sports	Nancy Meyers

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

A Very Strange Table!

PRODUCT_BUYER

	BuyerName	SKU_Managed	CollegeMajor
1	Pete Hansen	100100	Business Administration
2	Pete Hansen	100200	Business Administration
3	Nancy Meyers	101100	Art
4	Nancy Meyers	101100	Info Systems
5	Nancy Meyers	101200	Art
6	Nancy Meyers	101200	Info Systems
7	Cindy Lo	201000	History
8	Cindy Lo	202000	History
9	Jenny Martin	301000	Business Administration
10	Jenny Martin	301000	English Literature
11	Jenny Martin	302000	Business Administration
12	Jenny Martin	302000	English Literature

To understand why this is a very strange table, consider how you would add the fact that **Nancy Meyers** is now managing **SKU 101300**!

But First—

- We need to understand:
  - The **relational model**
  - **Relational model terminology**

The Relational Model

- Introduced in a paper published in 1970.
- Created by E.F. Codd
  - He was an IBM engineer
  - The model used mathematics known as "relational algebra"
- Now the standard model for commercial DBMS products.

Important Relational Model Terms

Important Relational Terms
Relation
Functional dependency
Determinant
Candidate key
Composite key
Primary key
Surrogate key
Foreign key
Referential integrity constraint
Normal form
Multivalued dependency

Entity

- An **entity** is some identifiable thing that users want to track:
  - Customers
  - Computers
  - Sales

Relation

- **Relational DBMS products** store data about entities in relations, which are a special type of table.
- A **relation** is a two-dimensional table that has the following characteristics:

Characteristics of Relations
Rows contain data about an entity.
Columns contain data about attributes of the entities.
All entries in a column are of the same kind.
Each column has a unique name.
Cells of the table hold a single value.
The order of the columns is unimportant.
The order of the rows is unimportant.
No two rows may be identical.

EmployeeNumber	FirstName	LastName
100	Jerry	Johnson
200	Mary	Abernathy
300	Liz	Smathers
400	Tom	Caruthers
500	Tom	Jackson
600	Eleanore	Caldera
700	Richard	Bandalone

A Relation
A Sample EMPLOYEE Relation

EmployeeNumber	FirstName	LastName	Department	Email	Phone
100	Jerry	Johnson	Accounting	JJ@somewhere.com	834-1101
200	Mary	Abernathy	Finance	MA@somewhere.com	834-2101
300	Liz	Smathers	Finance	LS@somewhere.com	834-2102
400	Tom	Caruthers	Accounting	TC@somewhere.com	834-1102
500	Tom	Jackson	Production	TJ@somewhere.com	834-4101
600	Eleanore	Caldera	Legal	EC@somewhere.com	834-3101
700	Richard	Bandalone	Legal	RB@somewhere.com	834-3102

Tables That Are Not Relations:
Multiple Entries per Cell

EmployeeNumber	FirstName	LastName	Department	Email	Phone
100	Jerry	Johnson	Accounting	JJ@somewhere.com	834-1101
200	Mary	Abernathy	Finance	MA@somewhere.com	834-2101
300	Liz	Smathers	Finance	LS@somewhere.com	834-2102
400	Tom	Caruthers	Accounting	TC@somewhere.com	834-1102, 834-1191, 834-1192
500	Tom	Jackson	Production	TJ@somewhere.com	834-4101
600	Eleanore	Caldera	Legal	EC@somewhere.com	834-3101
700	Richard	Bandalone	Legal	RB@somewhere.com	834-3102, 834-3191

Tables That Are Not Relations: Table with Required Row Order

EmployeeNumber	FirstName	LastName	Department	Email	Phone
100	Jerry	Johnson	Accounting	JJ@somewhere.com	834-1101
200	Mary	Abernathy	Finance	MA@somewhere.com	834-2101
300	Liz	Smathers	Finance	LS@somewhere.com	834-2102
400	Tom	Caruthers	Accounting	TC@somewhere.com	834-1102
				Fax:	834-9911
				Home:	723-8795
500	Tom	Jackson	Production	TJ@somewhere.com	834-4101
600	Eleanore	Caldera	Legal	EC@somewhere.com	834-3101
				Fax:	834-9912
				Home:	723-7654
700	Richard	Bandalone	Legal	RB@somewhere.com	834-3102

A Relation with Values of Varying Length

EmployeeNumber	FirstName	LastName	Department	Email	Phone	Comment
100	Jerry	Johnson	Accounting	JJ@somewhere.com	834-1101	Joined the Accounting Department in March after completing his MBA. Will take the CPA exam this fall.
200	Mary	Abernathy	Finance	MA@somewhere.com	834-2101	
300	Liz	Smathers	Finance	LS@somewhere.com	834-2102	
400	Tom	Caruthers	Accounting	TC@somewhere.com	834-1102	
500	Tom	Jackson	Production	TJ@somewhere.com	834-4101	
600	Eleanore	Caldera	Legal	EC@somewhere.com	834-3101	
700	Richard	Bandalone	Legal	RB@somewhere.com	834-3102	Is a full-time consultant to Legal on a retainer basis.

The Domain Integrity Constraint

- The requirement that all of the values in a column are of the same kind is know as the **domain integrity constraint**.
- The term domain means a grouping of data that meets a specific type definition.
  - **FirstName** could have a domain of names such as *Albert, Bruce, Cathy, David, Edith*, and so forth.
  - All values of **FirstName** *must* come from the names in that domain.
- Columns in different relations may have the same name.

Alternative Terminology

- Although not all tables are relations, the terms *table* and *relation* are normally used interchangeably.
- The following sets of terms are equivalent:

Table	Column	Row
Relation	Attribute	Tuple
File	Field	Record

To Key, or Not to Key That is the Question!

- In a relation as defined by Codd:
  - The rows of a relation must be *unique*
  - These is *no requirement* for a *designated primary key*
- The requirement for unique rows implies that a primary key can be designated.
- In the "real world", every relation has a primary key.
- When do we designate a primary key?
- **We need some more information!**

Functional Dependency

- A **functional dependency** occurs when the value of one (set of) attribute(s) determines the value of a second (set of) attribute(s):

  StudentID → StudentName

  StudentID → (DormName, DormRoom, Fee)
- The attribute on the left side of the functional dependency is called the **determinant**.
- Functional dependencies may be *based* on equations:

  ExtendedPrice = Quantity X UnitPrice

  (Quantity, UnitPrice) → ExtendedPrice
- Function dependencies are *not* equations!

Functional Dependencies Are Not Equations

Object Color	Weight	Shape
Red	5	Ball
Blue	5	Cube
Yellow	7	Cube

ObjectColor → Weight
ObjectColor → Shape
ObjectColor → (Weight, Shape)

Composite Determinants

- **Composite determinant** = a determinant of a functional dependency that consists of more than one attribute

  (StudentName, ClassName) → (Grade)

Functional Dependency Rules

- If A → (B, C), then A → B and A→ C.
  - This is the **decomposition rule**.
- If A → B and A→ C, then A → (B, C).
  - This is the **union rule**.
- However, if (A,B) → C, then *neither A nor B determines C by itself*.

Functional Dependencies in the SKU_DATA Table

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

SKU → (SKU_Description, Department, Buyer)
SKU_Description → (SKU, Department, Buyer)
Buyer → Department

Functional Dependencies in the ORDER_ITEM Table

	OrderNumber	SKU	Quantity	Price	ExtendedPrice
1	1000	201000	1	300.00	300.00
2	1000	202000	1	130.00	130.00
3	2000	101100	4	50.00	200.00
4	2000	101200	2	50.00	100.00
5	3000	100200	1	300.00	300.00
6	3000	101100	2	50.00	100.00
7	3000	101200	1	50.00	50.00

(OrderNumber, SKU) → (Quantity, Price, ExtendedPrice)

(Quantity, Price) → (ExtendedPrice)

What Makes Determinant Values Unique?

- A determinant is unique in a relation if and only if, it determines every other column in the relation.
- You cannot find the determinants of all functional dependencies simply by looking for unique values in one column:
  - Data set limitations
  - Must be logically a determinant

Keys

- A **key** is a combination of one or more columns that is used to identify rows in a relation.
- A **composite key** is a key that consists of two or more columns.

Candidate and Primary Keys

- A **candidate key** is a key that determines all of the other columns in a relation.
- A **primary key** is a candidate key selected as the primary means of identifying rows in a relation.
  - There is only one primary key per relation.
  - The primary key may be a composite key.
  - The ideal primary key is short, numeric, and never changes.

The Entity Integrity Constraint

- The requirement that, in order to function properly, the primary key must have unique data values for every row in the table is know as the **entity integrity constraint**.
- The phrase *unique data values* implies that this column is NOT NULL, and does not allow a NULL value in any row.

Surrogate Keys

- A **surrogate key** is an artificial column added to a relation to serve as a primary key.
  - DBMS supplied
  - Short, numeric, and never changes—an ideal primary key
  - Has artificial values that are meaningless to users
  - Normally hidden in forms and reports

Surrogate Keys

NOTE: The primary key of the relation is underlined below:

- RENTAL_PROPERTY without surrogate key:

  RENTAL_PROPERTY (Street, City, State/Province, Zip/PostalCode, Country, Rental_Rate)

- RENTAL_PROPERTY with surrogate key:

  RENTAL_PROPERTY (PropertyID, Street, City, State/Province, Zip/PostalCode, Country, Rental_Rate)

Foreign Keys

- A **foreign key** is the primary key of one relation that is placed in another relation to form a link between the relations.
  - A foreign key can be a single column or a composite key.
  - The term refers to the fact that key values are *foreign* to the relation in which they appear as foreign key values.

Foreign Keys

NOTE: The primary keys of the relations are underlined and any foreign keys are in *italics* in the relations below:

DEPARTMENT (DepartmentName, BudgetCode, ManagerName)
EMPLOYEE (EmployeeNumber, EmployeeLastName, EmployeeFirstName, *DepartmentName*)

The Referential Integrity Constraint

- A **referential integrity constraint** is a statement that limits the values of the foreign key to those already existing as primary key values in the corresponding relation:

  SKU in ORDER_ITEM must exist in SKU in SKU_DATA

Foreign Key with a Referential Integrity Constraint

NOTE: The primary key of the relation is underlined and any foreign keys are in *italics* in the relations below:

SKU_DATA (SKU, SKU_Description, Department, Buyer)
ORDER_ITEM (OrderNumber, *SKU*, Quantity, Price, ExtendedPrice)

Where ORDER_ITEM.SKU must exist in SKU_DATA.SKU

Database Integrity

- We have defined three constraints so far in our discussion:
  - The **domain integrity constraint**
  - The **entity integrity constraint**
  - The **referential integrity constraint**
- The purpose of these three constraints, taken as a whole, is to create **database integrity**, which means that the data in our database will be useful, meaningful data.