Chapter 8
Database

8.1 What is a Database

The term "database" can refer to either a collection of data or a database management system.

When referring to a collection of data, it implies a set of items that share common attributes. Examples of such collections include student name lists, customer records, and the vast amounts of data collected by the Internet of Things (IoT). Additionally, the storage media used for housing data, such as magnetic tapes, hard disks, CDs, and USB drives, can also be referred to as databases.

When referring to a database management system, it is a software component for file management in the operating system. In earlier times, such systems were often referred to as "data banks", but today they are commonly known as "databases".

In the early days of computing, a computer capable of handling 100 tasks was priced at \$10,000, whereas a new computer capable of handling 1000 tasks was often priced at \$50,000; hence, many companies frequently replaced their old computers. However, frequent replacements brought about more problems than just cost-performance ratio. Managers began analyzing the data stored and managed on each computer and discovered significant duplication, leading to poor memory efficiency. As a result, they decided to consolidate the data managed by each computer and created a shared database, marking the start of database management systems [1].

Database management systems will become more important in the future because they play a vital role in supporting decision-making in our daily life and work.

W. Weng, *A Beginner's Guide to Informatics and Artificial Intelligence*,
https://doi.org/10.1007/978-981-97-1477-3_8

8.2 Relational Database

In routine office work, commands can be used in a database management system called the relational database [2]. Relational databases have several key characteristics, including clarity, data independence, non-procedural data manipulation, and compatibility with distributed databases. Additionally, the efficiency of data reading and writing, which was once considered a drawback, has seen significant improvements.

Table 8.1 illustrates the basic structure of a relational database. In a relational database, a table is often referred to as a "relation", and the relation name is usually positioned in the upper-left corner of the table. Each relation consists of multiple attributes, which are represented as individual columns.

Users can search for attribute values both vertically and horizontally in relational databases as well as perform various operations, similar to commands. Dr. E. F. Codd, the inventor of the relational database, introduced several fundamental operations to the database, and the following are five commonly used ones.

(1) Union
(2) Difference
(3) Intersection
(4) Restriction
(5) Projection

Union, difference, and intersection are applicable to compatible relations, whereas restriction and projection are applicable to a single relation.

Compatible relations are defined by two criteria: (1) The relations must have the same number of attributes, i.e. the same number of columns. (2) They must have the same content for each attribute, regardless of the attribute names. For example, Table 8.2 "IEEE (Institute of Electrical and Electronics Engineers) members" and Table 8.3 "ACIS (Association for Computing and Information Sciences) members" are two compatible relations. They both have three columns, satisfying the first condition. In both relations, the first column contains names, the second column contains age, and the third column contains affiliation, meeting the second condition. Despite having different attribute names in the first row, these relations are compatible.

It should be noted that data in this chapter is fictional with no association with any real-world cases and is used only for illustrative purposes. In the following, we detail union, difference, and intersection, which are applicable to compatible relations.

Table 8.1 Structure of a relation

Relation name

Attribute 1	Attribute 2	...	Attribute *m*

Table 8.2 IEEE members

Name	Age	Affiliation
John Smith	59	Duke University
Anna Miller	42	IBM
Taro Ito	47	Sony
Yi Zhang	54	Wuhan University

Table 8.3 ACIS members

Designation	Years old	Employed by
Yi Zhang	54	Wuhan University
Mike Taylor	32	University of London
Cihon Lee	49	Samsung

8.2.1 Union

The union operation is commonly used to form the parent set. It creates the set A ∪ B, which is the sum of two sets, A and B, as illustrated in Fig. 8.1.

Table 8.4 shows the result of applying a union operation to the two relations, Tables 8.2 and 8.3. It combines all the rows from both relations, excluding any duplicates.

The resulting relation should be named by combining the names of the two original relations with the " ∪ " symbol. The resulting attribute names should be the same as those of the first relation, in this case, relation A. The data in the result should

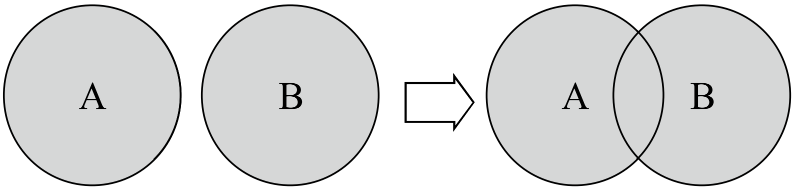

Fig. 8.1 Union of sets A and B

Table 8.4 IEEE members ∪ ACIS members

Name	Age	Affiliation
John Smith	59	Duke University
Anna Miller	42	IBM
Taro Ito	47	Sony
Yi Zhang	54	Wuhan University
Mike Taylor	32	University of London
Cihon Lee	49	Samsung

be a combination of the data from both original relations, representing a member of either IEEE or ACIS.

Redundancy is not allowed in union or other operations applicable to compatible relations. Redundancy occurs when the same data is duplicated within the result. The data “Yi Zhang, 54, Wuhan University” is present in both relations A and B, having it appear twice in the result would cause redundancy. Therefore, the process of ensuring that “Yi Zhang, 54, Wuhan University” appears only once in the result is known as removing redundancy.

8.2.2 Difference

The difference operation is often used as a constraint for data in the parent set. It creates the set A − B, which is the difference between two sets, A and B, as illustrated in Fig. 8.2.

Table 8.5 displays the result of performing a difference operation between Tables 8.2 and 8.3. This operation subtracts the second relation from the first, retaining only the attributes that differ between the two relations.

The resulting relation should be named by connecting the names of the two original relations by a minus symbol “−”. The resulting attribute names should be the same as those of the first relation, i.e. relation A. The data in the result should consist of the data from relation A that are not found in relation B, i.e. members of IEEE who are not members of ACIS.

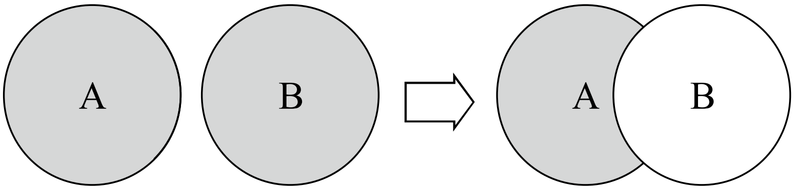

Fig. 8.2 Difference between sets A and B

Table 8.5 IEEE members − ACIS members

Name	Age	Affiliation
John Smith	59	Duke University
Anna Miller	42	IBM
Taro Ito	47	Sony

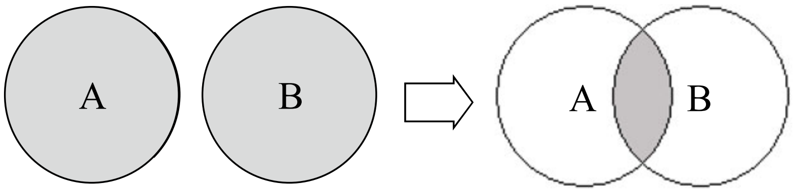

Fig. 8.3 Intersection of sets A and B

Table 8.6 IEEE members ∩ ACIS members

Name	Age	Affiliation
Yi Zhang	54	Wuhan University

8.2.3 Intersection

The intersection operation is often used as a constraint for data within the parent set. It extracts the common elements of two sets, A and B, as illustrated in Fig. 8.3.

Table 8.6 displays the result of applying the intersection operation to the two relations Tables 8.2 and 8.3. This operation extracts the rows that appear in both relations.

The resulting relation should be named by connecting the names of the two original relations by the " ∩ " symbol. The resulting attribute names should be the same as those of the first relation, i.e. relation A. The data in the result should be those that are present in both relation A and relation B, i.e. members who belong to both IEEE and ACIS.

8.2.4 Restriction

The restriction operation retrieves data that satisfy a constraint formula $A_i \theta A_j$. In this formula, A_i and A_j represent attributes, and θ is one of the operators $\{>, <, =, \neq, \geq, \leq\}$. The resulting relation, $R[A_i \theta A_j]$, contains data for which the formula $A_i \theta A_j$ holds.

To perform the restriction operation, relation R must be θ-comparable, which means that any value of attribute A_i can be compared with the value of attribute A_j using the operator θ.

Table 8.8 shows an example of getting the list of products requiring replenishment from the data in Table 8.7. The constraint is "stock $\leq$ safety stock". Here, "stock" refers to the current inventory level, and "safety stock" represents the expected quantity to be sold during the time between placing an order and its delivery.

Typically, safety stock is determined using historical data averages. For example, in the case of the first item, "champagne", the safety stock is 10, because it typically

Table 8.7 Inventory

Code	Name	Stock	Safety stock
P0001	Champagne	8	10
P0002	Red wine	11	10
P0003	Brandy	7	9
P0004	White wine	17	11
P0005	Beer	16	12

Table 8.8 Inventory [stock ≤ safety stock]

Code	Name	Stock	Safety stock
P0001	Champagne	8	10
P0003	Brandy	7	9

takes one week from order placement to delivery and during this period, approximately 10 bottles are sold on average. As a result, if the current stock falls below 10, there is a risk of running out of inventory before the next scheduled delivery unless an order is placed immediately. Similarly, the stock level for the third product also falls below its safety stock. Therefore, it is necessary to place immediate orders for both products, as shown in the result.

The resulting relation name should follow the format $R[A_i\theta A_j]$, where R is the original relation name and $[A_i\theta A_j]$ is the constraint. The resulting attribute names should be the same as the original attribute names.

In the restriction operation, attributes can undergo various mathematical operations such as multiplication, division, addition, and subtraction with constants. Furthermore, attributes can be compared to constants as well. For example, if you aim to reduce the risk of stockouts by proactively placing orders, you might consider increasing the safety stock by 3. The resulting restriction operation is given in Table 8.9.

Table 8.9 Inventory [stock ≤ safety stock + 3]

Code	Name	Stock	Safety stock
P0001	Champagne	8	10
P0002	Red wine	11	10
P0003	Brandy	7	9

8.2.5 *Projection*

The projection operation is typically used when dealing with a high number of attributes that cannot fit on a single screen or when there is a need to focus on only a few attributes. It extracts specific columns from a relation. Table 8.10 shows an example of a projection operation on the “name” and “stock” attributes in Table 8.7. The result is a vertical view comprising the two projected columns.

Table 8.10 Inventory

Name	Stock
Champagne	8
Red wine	11
Brandy	7
White wine	17
Beer	16

The resulting relation name should be the same as the original relation name. The resulting attribute names should be the same as those of the projected attributes. The data should be the same as those in the projected columns.

Exercises

1. Using the data provided in Tables 8.11 and 8.12, create a projection on the “name” attribute of the products belonging to the set (alcoholic drinks $\cup$ soft drinks) with a constraint of [stock $\leq$ safety stock $+ 2$].

Table 8.11 Alcoholic drinks

Code	Name	Stock	Safety stock
P0001	Champagne	8	10
P0002	Red wine	11	10
P0003	Brandy	7	9
P0004	White wine	17	11
P0005	Beer	16	12

Table 8.12 Soft drinks

Number	Product	Stock	Safety stock
P00010	Orange juice	18	10
P00011	Apple juice	14	12
P00012	Red tea	9	9

• **Guidance**

Steps to solve this problem.

1. Create the union of the two given sets “alcoholic drinks” and “soft drinks”.
2. Apply the restriction operation to the relation created in Step 1, filtering for [stock quantity $\leq$ safety stock + 2].
3. Apply the projection operation to the attribute “name” in the relation obtained in Step 2.

The final result after completing all the three steps is necessary, whereas the intermediate results of Steps 1 and 2 are not required. Please pay attention to the relation name: (1) Ensure that you create the relation name; (2) The name should be the outcome of performing the above three operations.

*2. Using the data provided in Table 8.13, which lists foods recommended for providing weekly nutrition, along with foods eaten this week (Table 8.14) and favorite foods (Table 8.15), perform necessary operations to identify the name(s) of the favorite foods that are recommended weekly but have not been eaten this week. Present the result.

Table 8.13 Weekly recommended foods

Code	Food
1	Apple
2	Tomato
3	Carrot
4	Yogurt
5	Fish

Table 8.14 Foods eaten this week

Number	Item
1	Apple
2	Beef
3	Ice cream
4	Milk
5	Fish

Table 8.15 Favorite foods

Number	Item
1	Apple
2	Peach
3	Beef
4	Yogurt
5	Fish

References

1. R Elmasri, Navathe S B (2015) Fundamentals of database systems, 7th edn. Pearson, Hoboken, NJ. https://amirsmvt.github.io/Database/Static_files/Fundamental_of_Database_Systems.pdf
2. Silberschatz A, Korth HF, Sudarshan S (2010) Database system concepts, 6th edn. McGraw-Hill Science Engineering https://www.octawian.ro/fisiere/situri/asor/build/html/_downloads/1fcab53a6d916e39c715fc20a9a9c2a8/Silberschatz_A_databases_6th_ed.pdf