AI - Chapter 3.3

Set Theory

Instructor: Saptarshi Jana

Learning Objectives

After studying this chapter, students will be able to:[a]

- Comprehend fundamental principles of Set Theory and Relational Algebra
- Understand the importance of context in mathematical operations
- Master various set operations and their applications
- Explore database table joins and relational operations

[a] Unit 3: Mathematics in AI

January 2, 2026

1 Introduction to Set Theory

1.1 Fundamentals of Sets

Set theory provides the mathematical foundation for organizing and manipulating collections of objects, which is particularly valuable for database operations and artificial intelligence applications. This branch of mathematics helps us understand how different collections relate to one another and forms the basis for efficient data retrieval systems.

In mathematics, a **set** represents a well-defined collection of distinct objects. These objects, called *elements* or *members*, can be anything—numbers, letters, people, or even other sets. Sets are typically denoted using capital letters and their elements are enclosed in curly braces.

Set Notation

A set S containing elements a, b, and c is written as:

$$S = \{a, b, c\}$$

Membership notation:

- $a \in S$ means "a is an element of S"
- $d \notin S$ means "d is not an element of S"

1.1.1 Examples of Sets

1. **Finite Set:** $A = \{2, 4, 6, 8, 10\}$ represents the first five positive even numbers
2. **Infinite Set:** $B = \{..., -6, -3, 0, 3, 6, 9, ...\}$ represents all multiples of 3
3. **Empty Set:** $\emptyset = \{\}$ contains no elements
4. **Set with Mixed Types:** $C = \{\text{red}, \text{blue}, \text{green}\}$ represents primary colors

1.2 Set Theory Principles

Set theory encompasses the mathematical framework for understanding well-defined collections and their relationships. It serves as a foundational concept in computer science and AI, particularly for modeling data structures and implementing efficient algorithms for information retrieval.

Sets can be represented in two primary ways:

- **Roster Form:** Explicitly listing all elements, e.g., $D = \{5, 10, 15, 20\}$
- **Set-Builder Notation:** Describing properties of elements, e.g., $E = \{x \mid x \text{ is a prime number less than } 20\}$

2 Context in Set Theory

2.1 Understanding Context

The concept of **context** is crucial in set theory as it establishes the framework within which we define and work with sets. Context provides the necessary boundaries and assumptions that ensure our mathematical operations remain meaningful and consistent.

Without proper context definition, set operations can become ambiguous or lead to logical contradictions. Consider the statement "the set of large numbers"—without context, we cannot determine which numbers qualify as "large."

2.1.1 Universe of Discourse

The *universe of discourse* (denoted as U) represents the complete collection of all elements under consideration in a particular problem or discussion. This universal set contains every object that might be relevant to our analysis.

Key Properties of Universe of Discourse

- Every set we consider is assumed to be a subset of U
- The universe determines what elements are available for our operations
- Different problems may have different universes of discourse
- Complement operations depend entirely on the defined universe

Example: If we're analyzing student test scores in a class:

- Universe U = All students enrolled in the class
- Set A = Students who scored above 80%
- Set B = Students who submitted assignments on time
- The complement A' would contain all students NOT in A, but still within U

2.2 Implicit vs. Explicit Context

Context can be specified in two ways:

1. **Implicit Context:** The universe is understood from the problem domain
   - When discussing "even numbers," it's typically understood we mean integers
   - Natural language often relies on shared understanding
2. **Explicit Context:** The universe is clearly stated to avoid confusion
   - Mathematical proofs require explicit context definition
   - Set-builder notation: $\{x \in \mathbb{N} \mid x \text{ is even}\}$ explicitly states we're working within natural numbers

2.3 Context Dependency in Operations

The meaning and results of set operations vary based on context:

- **Set Definition:** "Numbers divisible by 2" means different things in $\mathbb{Z}$ (integers), $\mathbb{N}$ (natural numbers), or $\mathbb{R}$ (real numbers)
- **Set Operations:** Union, intersection, and complement produce different results depending on the universal set
- **Complement Sensitivity:** The complement of $A = \{1, 2, 3\}$ depends entirely on what universe we're working in:
  - If $U = \{1, 2, 3, 4, 5\}$, then $A' = \{4, 5\}$
  - If $U = \mathbb{N}$, then $A' = \{4, 5, 6, 7, ...\}$

3 Set Operations

3.1 Fundamental Operations

When multiple sets are combined using mathematical principles, these processes are called **set operations**. These operations form the backbone of data manipulation in databases and logical reasoning in AI systems.

3.1.1 Set Union

The *union* of sets A and B (written as $A \cup B$) contains all elements that appear in either set or in both.

Formal Definition:

$$A \cup B = \{x \mid x \in A \text{ OR } x \in B\}$$

Example:

$$\begin{aligned} \text{Let } P &= \{a, e, i, o, u\} \text{ (vowels)} \\ Q &= \{a, b, c, d, e\} \text{ (first five letters)} \\ P \cup Q &= \{a, b, c, d, e, i, o, u\} \end{aligned}$$

Properties of Union:

- Commutative: $A \cup B = B \cup A$
- Associative: $(A \cup B) \cup C = A \cup (B \cup C)$
- Identity: $A \cup \emptyset = A$
- Duplicate elements appear only once in the result

3.1.2 Set Intersection

The *intersection* of sets A and B (written as $A \cap B$) contains only elements that appear in both sets simultaneously.

Formal Definition:

$$A \cap B = \{x \mid x \in A \text{ AND } x \in B\}$$

Example:

$$\begin{aligned} \text{Let } M &= \{2, 4, 6, 8, 10\} \text{ (even numbers)} \\ N &= \{3, 6, 9, 12\} \text{ (multiples of 3)} \\ M \cap N &= \{6\} \text{ (numbers that are both even and multiples of 3)} \end{aligned}$$

Special Case: When $A \cap B = \emptyset$, the sets are called *disjoint sets*.

3.1.3 Set Difference

The *difference* of sets A and B (written as $A - B$ or $A \setminus B$) contains elements that belong to A but not to B.

Formal Definition:

$$A - B = \{x \mid x \in A \text{ AND } x \notin B\}$$

Example:

$$\begin{aligned} \text{Let } X &= \{1, 2, 3, 4, 5, 6\} \\ Y &= \{4, 5, 6, 7, 8\} \\ X - Y &= \{1, 2, 3\} \\ Y - X &= \{7, 8\} \end{aligned}$$

Important Note: Set difference is NOT commutative: $A - B \neq B - A$ in general.

3.1.4 Complement of a Set

The *complement* of set A (written as A' or $\bar{A}$ or A^c) contains all elements in the universe that are not in A.

Formal Definition:

$$A' = U - A = \{x \mid x \in U \text{ AND } x \notin A\}$$

Example:

$$\begin{aligned} \text{Universe } U &= \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\} \\ \text{Set } A &= \{2, 4, 6, 8, 10\} \text{ (even numbers)} \\ A' &= \{1, 3, 5, 7, 9\} \text{ (odd numbers)} \end{aligned}$$

Properties:

- $(A')' = A$ (complement of complement returns original set)
- $A \cup A' = U$
- $A \cap A' = \emptyset$
- $U' = \emptyset$ and $\emptyset' = U$

3.1.5 Cartesian Product

The *Cartesian product* of sets A and B (written as $A \times B$) produces all possible ordered pairs where the first element comes from A and the second from B.

Formal Definition:

$$A \times B = \{(a, b) \mid a \in A \text{ AND } b \in B\}$$

Example:

$$\begin{aligned} \text{Let } A &= \{1, 2\} \\ B &= \{x, y, z\} \\ A \times B &= \{(1, x), (1, y), (1, z), (2, x), (2, y), (2, z)\} \end{aligned}$$

Important Properties:

- NOT commutative: $A \times B \neq B \times A$
- If $|A| = m$ and $|B| = n$, then $|A \times B| = m \times n$
- Used extensively in database theory for creating relationships

Summary of Set Operations

Operation	**Notation**	**Description**
Union	$A \cup B$	Elements in A or B or both
Intersection	$A \cap B$	Elements in both A and B
Difference	$A - B$	Elements in A but not in B
Complement	A'	Elements not in A (within U)
Cartesian Product	$A \times B$	All ordered pairs from A and B

4 Databases and Relational Algebra

4.1 Introduction to Databases

A **database** is a structured repository of information organized to support efficient storage, retrieval, modification, and analysis of data. Modern databases serve as the foundation for organizational decision-making and data-driven applications.

4.1.1 Key Database Components

1. **Database:** A comprehensive collection of related tables storing information about entities
2. **Table (Relation):** A two-dimensional structure resembling a spreadsheet with rows and columns
3. **Column (Attribute):** A vertical section containing data of the same type and meaning

4. **Row (Tuple/Record):** A horizontal entry representing a complete set of related information

Example Database: Library Management System
BOOKS Table:

Book_ID	Title	Author	Year
101	Data Science Basics	Smith, J.	2020
102	AI Fundamentals	Johnson, M.	2021
103	Python Programming	Williams, R.	2019
104	Machine Learning	Brown, A.	2022

BORROWERS Table:

Borrower_ID	Name	Email
1	Alice Cooper	alice@email.com
2	Bob Wilson	bob@email.com
3	Carol Davis	carol@email.com

4.2 Relational Algebra

Relational algebra comprises a collection of operations that manipulate relations (tables) to extract desired information. It forms the theoretical foundation for database query languages like SQL.

4.2.1 Selection Operation (σ)

The selection operator filters rows from a table based on specified conditions, producing a horizontal subset.
Notation: $\sigma_{\text{condition}}(\text{Table})$
Example:

$$\sigma_{\text{Year}>2020}(\text{BOOKS}) = \{\text{rows where Year is greater than 2020}\}$$

This would return only the books published after 2020.

4.2.2 Projection Operation (π)

The projection operator selects specific columns from a table, producing a vertical subset.
Notation: $\pi_{\text{attributes}}(\text{Table})$
Example:

$$\pi_{\text{Title, Author}}(\text{BOOKS}) = \{\text{only Title and Author columns}\}$$

4.2.3 Union Operation ($\cup$)

Combines rows from two compatible tables (same structure) into a single result, eliminating duplicates.
Requirement: Tables must have identical attributes (union-compatible).
Example: If we have two tables of students from different semesters with the same structure, their union would give us all unique students.

4.2.4 Intersection Operation ($\cap$)

Returns only the rows that appear in both tables simultaneously.

Example: Finding students who took courses in both Fall and Spring semesters.

4.2.5 Difference Operation ($-$)

Returns rows that exist in the first table but not in the second.

Example: Students who registered for Fall semester but not for Spring semester.

4.2.6 Cartesian Product ($\times$)

Combines every row from the first table with every row from the second table, creating all possible combinations.

Example: If BOOKS has 4 rows and BORROWERS has 3 rows, BOOKS $\times$ BORROWERS produces 12 rows.

4.2.7 Rename Operation (ρ)

Allows changing the name of a relation or its attributes for clarity or to resolve naming conflicts.

4.2.8 Division Operation ($\div$)

Used for "for all" queries—finding tuples in one relation associated with all tuples in another relation.

Example: Finding students who have taken ALL required courses.

5 SQL Joins

5.1 Understanding Joins

In relational databases, information is distributed across multiple tables to minimize redundancy and maintain data integrity. **SQL joins** enable us to combine related data from different tables based on common attributes, reconstructing meaningful information.

A join operation requires:

1. Two or more tables
2. A join condition (typically matching columns)

5.2 Types of Joins

5.2.1 INNER JOIN

An *INNER JOIN* returns only the rows where matching values exist in both tables.

Syntax:

```
SELECT columns
FROM table1
INNER JOIN table2
ON table1.common_column = table2.common_column;
```

Example: Using our library database:

```
SELECT BORROWERS.Name, LOANS.Book_ID, LOANS.Loan_Date
FROM BORROWERS
INNER JOIN LOANS
ON BORROWERS.Borrower_ID = LOANS.Borrower_ID;
```

This query returns only borrowers who have active loans, excluding borrowers with no loans and loans with invalid borrower references.

5.2.2 LEFT OUTER JOIN (LEFT JOIN)

A *LEFT JOIN* returns all rows from the left table and matching rows from the right table. When no match exists, NULL values fill the right table's columns.

Syntax:

```
SELECT columns
FROM table1
LEFT JOIN table2
ON table1.common_column = table2.common_column;
```

Example:

```
SELECT BORROWERS.Name, LOANS.Book_ID
FROM BORROWERS
LEFT JOIN LOANS
ON BORROWERS.Borrower_ID = LOANS.Borrower_ID;
```

This returns all borrowers, including those who haven't borrowed any books (with NULL for Book_ID).

5.2.3 RIGHT OUTER JOIN (RIGHT JOIN)

A *RIGHT JOIN* returns all rows from the right table and matching rows from the left table. Non-matching rows from the left table appear as NULL.

Example:

```
SELECT BORROWERS.Name, LOANS.Book_ID
FROM BORROWERS
RIGHT JOIN LOANS
ON BORROWERS.Borrower_ID = LOANS.Borrower_ID;
```

This returns all loans, even those with invalid borrower references (orphaned records).

5.2.4 FULL OUTER JOIN

A *FULL OUTER JOIN* returns all rows from both tables, with NULL values where no match exists.

Example: Combining all borrowers and all loans, showing all possible records regardless of matches.

Join Comparison

- **INNER JOIN:** Only matching records
- **LEFT JOIN:** All left table + matching right table
- **RIGHT JOIN:** All right table + matching left table
- **FULL JOIN:** All records from both tables

Key Concepts Summary

Chapter Recap

Set Theory:

- Sets are well-defined collections of distinct objects
- Context (universe of discourse) is essential for unambiguous operations
- Set operations (union, intersection, difference, complement, Cartesian product) combine sets mathematically
- Set theory provides foundations for database operations and logical reasoning

Databases and Relational Algebra:

- Databases organize data in tables (relations) with rows (tuples) and columns (attributes)
- Relational algebra operations (selection, projection, joins) manipulate tables
- SQL joins (INNER, LEFT, RIGHT, FULL) combine related data from multiple tables
- Understanding joins is crucial for effective database querying

6 Practice Exercises

1. Define set theory and explain its significance in database management systems.
2. Explain why context is crucial when performing set operations. Provide an example.
3. Differentiate between INNER JOIN and LEFT OUTER JOIN with examples.
4. Given sets $A = \{2, 4, 6, 8\}$ and $B = \{6, 8, 10, 12\}$, find:
   (a) $A \cup B$ (b) $A \cap B$ (c) $A - B$ (d) $B - A$
5. Describe the Cartesian product operation and provide a practical database example.