# Application of Set and Tree for Relational Database Management System

Garmastewira 13514068
*Program Studi Teknik Informatika*
*Sekolah Teknik Elektro dan Informatika*
*Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia*
*13514068@itb.ac.id*

***Abstract*—Database is a crucial part in this technological century. Any activity that is related to using a computer, whether it is a mobile or immobile device, must be using a database. One of the most used model of database management system is relational model. Relational DBMS depends strongly on the concept of set and tree in Discrete Mathematics. One of the oldest and most used RDBMS today is MySQL.**

***Keywords*—database, MYSQL, relation, set, schema, SQL, tree.**

## I. Introduction

"Whether you know it or not, you're using a database every hour, or even every minute." Nowadays, social media is a critical part of every single modern human. People check their smartphones for their instant messaging and email applications to communicate and update news. All of these features cannot be separated from something called as database.

Speaking formally, a database is simply an organized collection of data. The data types vary from tables, reports, schemas, queries, and other objects. To simplify, let's take Facebook as an example. It has a database of its users, e.g. the users' names, the users' connections/friends, where they work, to whom they are married to, etc. Facebook also saves each user's posting and comment updates to an unknown, perhaps unlimited, period of time.

It is extremely magical to think how Facebook could handle its database. All of those updates could happen a thousand times per second, and Facebook could stand still for a long period of time. It is the service of the database management system, or abbreviated as DBMS. DBMS is a computer software application that interacts with the user, other applications, and the database itself to capture and analyze data. To handle a site as gigantic as Facebook, according to [1], one must create a DBMS which is efficient, reliable, convenient, safe, and can handle multi-user storage and massive amounts of persistent data.

There are a lot of proposed models for DBMS, i.e. the flat file model, hierarchical model, object-oriented model, network model, and relational model. The most used and perhaps most comprehensible model is the relational model [5]. In fact, a huge site like Facebook relies on MySQL for handling its database, an open-source and the second-most popular relational DBMS in the world.

The relational model of DBMS is based on relational algebra. In fact, relational algebra applies the concepts of Discrete Mathematics, particularly set and tree. This paper will analyze relational algebra which is pretty much an expansion of the set in mathematics, the usage of tree for clearer depiction of relational algebra operations, and relational algebra's application on relational DBMS.

## II. Principal Theory

### *A. Set*

There are many definitions of set, but generally, a set is a collection of *distinct* objects [2]. Note that the *distinct* word is italicized to emphasize that a set cannot contain two same objects, otherwise it would be known as a *bag* or a *multiset*. A set can be served into a lot of representations. One of the most basic representation is by using enumeration. For example, we can represent the set A which has the Arabic numerals as the following,

$$A = \{0, 1, 2, \dots, 9\}$$

Another useful representation would be using Venn diagram. For example, let A be the previous set and B be a set which contains only odd numbers from zero to nine. The Venn diagram is as the following,

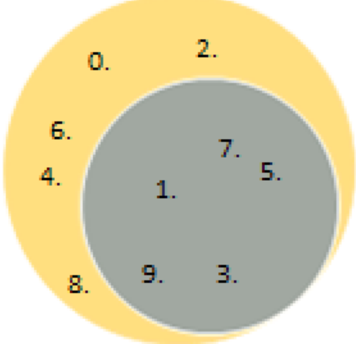



Figure 1 An example of Venn diagram depiction

It is possible to do operations on two or more sets, resulting in another set. There are some general operations, yet these are a few operations that will be often used in relational algebra [3]:

1. Intersection

Intersection of set A and set B is a set in which each of its element is a member of both set A and set B.

Notation: $A \cap B = \{x \mid x \in A \; and \; x \in B\}$

2. Union

Union of set A and set B is a set in which each of its element is a member of set A or set B.

Notation: $A \cup B = \{x \mid x \in A \; or \; x \in B\}$

3. Complement

Complement of set A relative to the universal set U results in a set in which each of its element is an element of U and is not an element of A.

Notation: $A^c = \{x \mid x \in U \; and \; x \notin A\}$

4. Difference

The difference of set A and B is a set in which each of its element is an element of A but not an element of B.

Notation: $A - B = \{x \mid x \in B \; and \; x \notin B\}$

Note that from the difference operator does not add an expressive power as $A - B = A \cap B^c$.

5. Cartesian Product

The Cartesian product of set A and B is another set in which each of its element is an ordered pair of two elements where the first element is A's element and the second is B's element.

Notation: $A \, X \, B = \{(a, b) \mid a \in A \; and \; b \in B\}$

The total elements of $A \, X \, B$ is equal to the number of elements in A and that of in B.

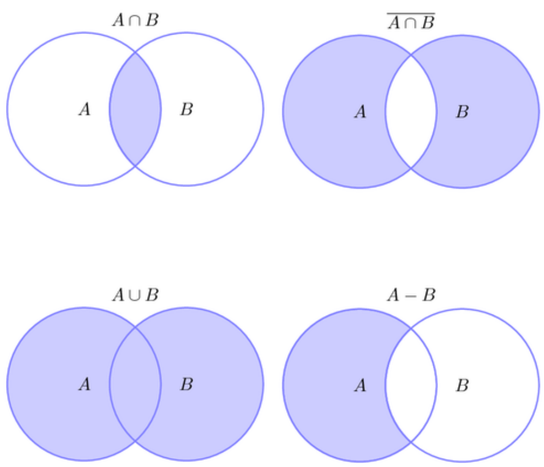



Figure 2 The depiction of basic set operations by using Venn diagram
Source: http://www.texample.net/tikz/examples/set-operations-illustrated-with-venn-diagrams

## B. Multiset

A multiset or primarily known as *bag* in English is a generalization of set in which it may contain multiple same elements. The majority of database management system prefer the usage of multiset to the normal set. One trivial example of multiset usage would be a database of students in a class as some sets of the database, e.g. name of the students, will have the same elements.

To distinguish set and multiset, the term multiplicity is used. Multiplicity of an element is the number of the elements in the multiset. For example, let multiset $A = \{0, 0, 0, 1, 1, 2\}$. The multiplicity of element 0 is 3 as it shows up three times in the multiset.

Multiset's operations are quite different from those of set. There are four fundamental multiset's operations:

1. Union

The union of multiset A and B results in a multiset in which its element's multiplicity equals to the maximum multiplicity of that element in set A and B.

Notation: $A \cup B$

2. Intersection

The intersection of multiset A and B results in a multiset in which its element's multiplicity equals to the minimum multiplicity of that element in set A and B

Notation: $A \cap B$

3. Difference

The difference of multiset A and B results in which its element's multiplicity equals to that element's multiplicity in set A minus that element's multiplicity in set B. Note that if the subtraction results is either zero or negative, then that element is not the element of the new multiset.

Notation: $A - B$

4. Sum / Disjoint Union / Bag Union

The sum of multiset A and B results in a multiset in which its element's multiplicity is the sum of that element's multiplicity in multiset A and B.

Notation: $A + B$ or $A \uplus B$

## C. Tree

A tree structure is a way of representing the hierarchical nature of a structure in a graphical form. A tree structure is merely conceptual. It can appears in a lot of forms depending on what specific field it is used. As this paper analyzes tree in discrete mathematics, we define tree in graph theory. A tree is an undirected graph in which it has no vertices.

A tree G has properties as the following:

1. Every pair of node is connected by a single path.

2. If a graph has n nodes, then it has n-1 edges, thereby having no circuit.

3. G is connected.

Usually, in a lot of tree applications, a certain node is acting as the root of the tree. Unlike a real tree, the root is put as the topmost node, and the nodes below the root that are connected to the root are called as the children. A node that does not have children is a leaf. Another important term is the n-ary tree. An n-ary tree is a form of tree in which each of its node can only have n children at maximum. The most used instance of n-ary tree is binary tree.

Rooted tree has extensive applications. Some beneficial rooted tree forms are decision tree, Huffman code, and expression tree. The latter form of applicative tree is a tree in which the leaves are considered as the operands and the other nodes are considered as the operators. Expression tree can represent many kind of mathematical expressions, one of them is arithmetic expression. By using binary expression tree, one is not required to put parentheses to show the operators' precedence and it is flexible to be read as a either postfix, infix, or prefix notation. An example of a binary expression tree is shown in Figure 3:

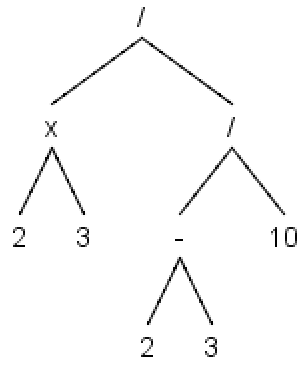



Figure 3 An example of binary expression tree

The tree can be translated into any of these following notations:

1. Prefix: / x 2 3 / - 2 3 10
2. Infix: (2 x 3) / ((a - b) / c)
3. Postfix: 2 3 x 2 3 - 10 / /

## III. APPLICATION OF SET AND TREE FOR RELATIONAL DATABASE CONCEPT

### *A. Database: The Relational Model*

In relational database management system, a database is a set of relations. It is often useful to depict a relation in a relational database simply as a table with rows and columns. Formally, row in a table is referred as **tuple** and column is referred as **attribute**. Each attribute has a type or a **schema**. For instance, let a database have two relations, one is the Student relation and the other is Acceptance relation, as depicted in both Table I and Table II below.

| Student ID | Name | National Exam |
|---|---|---|
| 111 | Gumi | 55.00 |
| 124 | Ali | 50.00 |
| 145 | Azka | 53.00 |
| 169 | Irfan | 59.80 |

Table I Student Relation

| Student ID | University | Acceptance Status |
|---|---|---|
| 111 | ITB | Yes |
| 111 | UI | No |
| 124 | UI | Yes |
| 145 | ITB | No |
| 145 | UI | No |
| 169 | ITB | Yes |
| 169 | UI | Yes |

Table II Acceptance Relation

For the Student relation, the attributes are Student ID, Name, and National Exam Score. (111, Gumi, 55.00) is an example of a tuple. The schema does not need to be shown explicitly, but from the relation, it can be inferred that Student ID's type is integer, and so on. Note that the type of attribute "Name" in Student relation and that of attribute "University" in Acceptance relation do not have the same type unless we define it explicitly.

It would be useful if it is possible to query the database to ask for certain information. For instance, a wise query would be "Name of students who are accepted to ITB". The key to answer the query is by using relational algebra, which depends strongly on set operations.

### *B. Relational Algebra Operations*

An operation to one or more relations results in another relation. These are the fundamental operations of relational algebra:

1. Selection

Notation: $\sigma_{c1,c2,\ldots,cn}(R)$, where c is a condition (as in "if" statement).

The selection operator is used to obtain a tuple of a relation that matches the condition that the operator provides. For example, we can select tuples in Student relation that has national exam score at least 55.00 by writing $\sigma_{National\ Exam\ >\ 55.0}(Student)$. The output is shown in Table III.

| Student ID | Name | National Exam |
|---|---|---|
| 111 | Gumi | 55.00 |
| 169 | Irfan | 59.80 |

Table III $\sigma_{National\ Exam\ >\ 55.0}(Student)$ results

2. Projection

Notation: $\pi_{A1,A2,\ldots,An}(R)$, where A is an attribute.

The projection operator selects a certain attribute/column. Note that this operator can be combined with the selection operator. For example, to answer the query "Student IDs of students who have national exam score at least 55.00", the operation would be $\pi_{Student\ ID}(\sigma_{National\ Exam\ >\ 55.0}(Student))$. The output is shown in table IV.

| Student ID |
|---|
| 111 |
| 169 |

Table IV $\pi_{Student\ ID}(\sigma_{National\ Exam\ >\ 55.0}(Student))$ results

3. Cross-Product

Notation: $R1\ X\ R2$

Similar to the Cartesian product of set, the cross product of two relations, e.g. relation R1 and R2, results in another relation in which each of its tuple is a tuple concatenation of R1's tuples and R2's tuples. Therefore, the number of the tuples is the number of R1's tuples times that of R2's tuples. Note that whenever there are attributes that have the same name, it must be clear from which relation each of the attribute is. For example, the output of relation $Student\ X\ Acceptance$ is shown in Table V.

| Student. Student ID | Name | National Exam | Acceptance. Student ID | University | Acceptance Status |
|---|---|---|---|---|---|
| 111 | Gumi | 55.00 | 111 | ITB | Yes |
| 124 | Ali | 50.00 | 111 | ITB | Yes |
| 145 | Azka | 53.00 | 111 | ITB | Yes |
| 169 | Irfan | 59.80 | 111 | ITB | Yes |
| 111 | Gumi | 55.00 | 111 | UI | No |
| 124 | Ali | 50.00 | 111 | UI | No |
| 145 | Azka | 53.00 | 111 | UI | No |
| 169 | Irfan | 59.80 | 111 | UI | No |
| 111 | Gumi | 55.00 | 124 | UI | Yes |
| 124 | Ali | 50.00 | 124 | UI | Yes |
| 145 | Azka | 53.00 | 124 | UI | Yes |
| 169 | Irfan | 59.80 | 124 | UI | Yes |
| 111 | Gumi | 55.00 | 145 | ITB | No |
| 124 | Ali | 50.00 | 145 | ITB | No |
| 145 | Azka | 53.00 | 145 | ITB | No |
| 169 | Irfan | 59.80 | 145 | ITB | No |
| 111 | Gumi | 55.00 | 145 | UI | No |
| 124 | Ali | 50.00 | 145 | UI | No |
| 145 | Azka | 53.00 | 145 | UI | No |
| 169 | Irfan | 59.80 | 145 | UI | No |
| 111 | Gumi | 55.00 | 169 | ITB | Yes |
| 124 | Ali | 50.00 | 169 | ITB | Yes |
| 145 | Azka | 53.00 | 169 | ITB | Yes |
| 169 | Irfan | 59.80 | 169 | ITB | Yes |
| 111 | Gumi | 55.00 | 169 | UI | Yes |
| 124 | Ali | 50.00 | 169 | UI | Yes |
| 145 | Azka | 53.00 | 169 | UI | Yes |
| 169 | Irfan | 59.80 | 169 | UI | Yes |

Table V $Student\ X\ Acceptance$ results

4. Natural Join

Notation: $R1 \bowtie R2$

The natural join operator creates a new relation by connecting R1 and R2 with equating the same attributes and then projecting one copy of each pair of equated attributes. It would be best to illustrate the results of $Student \bowtie Acceptance$ in Table VI. In this case, the equated attribute is the Student ID attribute.

| Student ID | Name | National Exam | University | Acceptance Status |
|---|---|---|---|---|
| 111 | Gumi | 55.00 | ITB | Yes |
| 111 | Gumi | 55.00 | UI | No |
| 124 | Ali | 50.00 | UI | Yes |
| 145 | Azka | 53.00 | ITB | No |
| 145 | Azka | 53.00 | UI | No |
| 169 | Irfan | 59.80 | ITB | Yes |
| 169 | Irfan | 59.80 | UI | Yes |

Table VI $Student \bowtie Acceptance$ results

5. Rename

Notation: $\rho_{R(A1,A2,\ldots,An)}(R1)$, where R is the new relation's name and A is the new attribute's name

Rename operator is used to change the schema of attribute(s) of a relation. It is handy when operations like union, intersection, and difference is performed on the relation.

6. Union

Notation: $R1 \cup R2$

Union operator acts as the tuples unifier of relation R1 and R2 if and only if the attributes of R1 and R2 have exactly the same types. For example, if we want to create a "student and university names" relation, yet we know that University and Student Name have different schemas, we could use the rename operator to handle it. Assume that the new relation's name is R. The operation to create that relation is the following:

$$\rho_{R(Name)}\left(\pi_{Name}(Student)\right) \cup \rho_{R(Name)}\left(\pi_{University}(Acceptance)\right)$$

The output is shown in Table VII.

| Name |
|---|
| Gumi |
| Ali |
| Azka |
| Irfan |
| ITB |
| UI |

Table VII "Student and university names" relation

7. Intersection and Difference

Notation: $R1 \cap R2\ and\ R1 - R2$

Similar to the concept of set's intersection operation, the intersection of R1 and R2 results in a relation in which its tuples are the tuples of both R1 and R2. The difference of R1 and R2 results in a relation in which its tuples are the tuples of R1 that are not the tuples of R2. As previously has been said, $R1 - R2$ is similar to $R1 \cap R2^c$. Note that just like the union operator, the attributes' schemas of both R1 and R2 must be the same prior to applying the operator.

### *C. Alternative Notation: Expression Tree*

By using the operators above, the relation "Name of students who are accepted to ITB" can be created, which is by using the following expression:

$$\pi_{Name}(\sigma_{University=\text{ITB},\ status=\text{Yes}}(Student \bowtie Acceptance))$$

Note that such notation might be convoluted when the expression is sufficiently long. Instead of the general notation, we can apply the expression tree to relational algebra. The leaves of the tree are the relations, and the other nodes are the operators. For instance, the above expression will look like Fig. 4 if it is expressed as a tree.

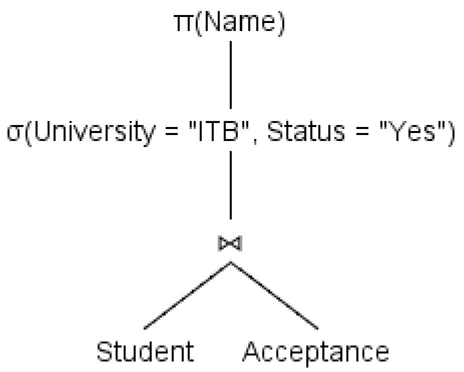



Figure 4 The expression tree of query "Name of students who are accepted to ITB"

## IV. Relational DBMS Implementation: MySQL Introduction

According to [4], SQL (Standard Query Language) is a special-purpose programming language designed for managing data held in a relational database management system (RDBMS). SQL is created in 1974 by Raymond F. Boyce and Donald D. Chamberlin. SQL is such a powerful language that it is considered to be one of very-high level language.

One of the most used relational database management system that implements SQL is MySQL. Although MySQL was created in 1995, which is 20 years ago, and does not provide a GUI for any platform, it is still the second-most used RDBMS in the world.

Querying using MySQL is similar to what we do by using relational algebra notation. Here in MySQL, the querying notation is as the following:

```
SELECT A1, A2, …
FROM R1, R2, ….
WHERE (Condition)
```

This notation is equivalent to $\pi_{A1,A2,\ldots}(\sigma_{Cond}(R1,\ldots))$.

Other useful MySQL commands are the following:

```
- CREATE Database <database name>
- CREATE Table <table name>,
<attribute 1> <type>,
<attribute 2> <type>,
…
<attribute n> <type>;
```

One important note is that when we talk about SQL, then the relation is not a set. Instead, each relation of the SQL database is a bag/multiset. For example, if we create "student and university names" relation in SQL, we should not expect the results as shown in Table VII. The repetition of both ITB and UI will not be deleted. The results are as shown in Table VIII.

| Name |
|---|
| Gumi |
| Ali |
| Azka |
| Irfan |
| ITB |
| UI |
| UI |
| ITB |
| UI |
| ITB |
| UI |

Table VIII "Student and university names" relation (multiset)

## V. Conclusion

Database is an important part of nowadays' technology and today, the most known model of database management system is the relational model. It is extremely important to understand the concept of both set and tree in discrete mathematics as both of these concepts are fundamental to the relational model of database.

## VI. Acknowledgment

The author would like to thank Prof. Jennifer Widom of Stanford University for being an awe-inspiring teacher in her *Introduction to Database* massive online open course. The author would like to express his gratitude to Dr. Rinaldi Munir as well for giving the author a strong comprehension on Discrete Mathematics so that the author could learn myriads of new Computer Science courses effortlessly.

## References


[1] Widom, Ullman, *A First Course in Database System 3rd Edition*, Pearson, 2007.

[2] Liu, C.L, *Element of Discrete Mathematics*, McGraw-Hill, Inc, 1985.

[3] Munir, Rinaldi. *Matematika Diskrit Edisi 3*, Penerbit Informatika, 2010.

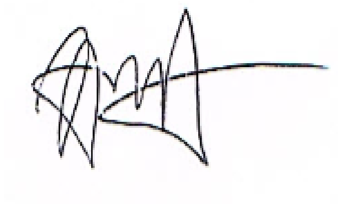
[4] IBM Knowledge Center, "SQL: Tables", <htttp://www-01.ibm.com> [retrieved 8th December 22.28]

[5] Geographic Information Technology Training Alliance, "Database Models", <http://www.gitta.info> [retrieved 9th December 2015]


## DECLARATION

I hereby certify that this paper is a copyright on my own, neither a copy nor a translation of any other paper, and not an act of plagiarism.

Bandung, 9th December 2015



Garmastewira 13514068