Applying flower venn diagram for presenting database

Herlinda¹, Agi Candra Bramantia¹, Nunu Kustian¹, Khasanah² and Erlin Windia Ambarsari*

¹Informatics Department, Universitas Indraprasta PGRI, Jakarta, Indonesia
²Informatics Department, Universitas Siber Asia, Jakarta, Indonesia

*erlinunindra@gmail.com

Abstract. The Venn Diagram helps obtain information from several cases. Subsequently, we proposed a Flower Venn Diagram for tables organized and straightforward to present a Database. Flowers usually encountered anatomies such as petals, pistil (stigma, style, ovary), stamen (anther, filament), sepal, ovule, receptacle, and stalk. However, Flower Venn Diagram only takes two parts, namely petals and receptacle. The receptacle is a container centralized as tables link, tables itself referred to as petals. To construct a Flower Venn Diagram, it needs keys that have influenced for connecting petals and receptacle. However, when applied to SQL, we found that the Join clause was created sequentially for a Flower Venn Diagram establishment. It different when we used the Where clause; it constructs directly. When we add more another Flower Venn Diagram, we found that more complex and patterned. Therefore, All fields are conveniently searchable. Nevertheless, it needs to investigate further whether the data in the Diagram effectively

1. Introduction

Venn Diagram is a visualization of data with an oval shape to illustrate the data set, usually used to classify the data members. The Venn Diagram analysis helps obtain information from several cases using operators: subtraction, union, and intersection. For example, we found cases such as [1], that the information obtained from the Diagram using subtraction is sales order influenced by classification of work orders, reversed credit denials, late poor history waivers, and late product codlings. The influence has happened when the data has a relationship and combine as possible [2], [3]—several ovals combined in Diagram can establish a unique model. Some examples, the Diagram form finds in [4]–[6], one of which is a flower.

Therefore, we propose a Flower Venn Diagram for Database Design in Database Management System (DBMS), which applies the Venn Diagram to the database structure to describe related between tables. This study aims to map the dataset more organized and straightforward to present a Database than an Entity Relationship Diagram (ERD)[7]. The keys in the table had an essential role in making a relationship between the table. They are primary and foreign keys, which the point to build a Flower Venn Design.

2. Methodology

The concept of a Flower Venn Diagram obtains from flower anatomy[8], [9]. The general anatomy is as follows:
Flowers usually had anatomies such as petals, pistil (stigma, style, ovary), stamen (anther, filament), sepal, ovule, receptacle, and stalk. However, Flower Venn Diagram only takes two parts, namely petals and receptacle. The receptacle is a container centralized as tables link, tables itself referred to as petals. Construct Flower Venn Diagram; it needs keys that have influenced for connecting petals and receptacle.

Figure 2 makes a Flower Venn Diagram for developing a query, requiring an operator to relate all tables. It needs considering the operator as follows [10]:

a. Union
   it builds based on the integration of all data or fields in tables at once.

b. Intersection
   it builds based on a homogenous dataset that had a role as keys. Therefore, the data which had linkages with keys become connected.

When a Flower Venn Diagram had designed, then the query is formed based on the following clauses[11] in SQL:

```
SELECT table1.column1, table2.column2...
FROM table1
INNER JOIN table2
ON table1.field1 = table2.field2;
```

or

```
SELECT table1.column1, table2.column2...
FROM table1, table2
WHERE table1.field1 = table2.field2;
```

Where Relational Algebra: \[ \Pi \text{ (table1.column1, table2.column2) (} \sigma \text{ (table1.field1=table2.field2)(table1-table2)} \] 

For understanding to make a query based on a Flower Venn Diagram, we create tables as a sample for simulation as follows:
3. Result and Discussion

The result based on Figure 3, a Flower Venn Diagram is shown as follows:

**Figure 3. Sample Tables**

```
| ID_CASHIER | NAME_CASHIER | PHONE_NUMBER | ADDRESS |
|------------|--------------|--------------|---------|
| 21         | 57           | 0015555555   | IN      |
| 22         | AK           | 0026666666   | IT      |
| 23         | JW           | 0037777777   | JP      |
```

```
| ID_TENANT | NAME_TENANT | PHONE_NUMBER | IDENTITY_CARD |
|-----------|-------------|--------------|---------------|
| 1         | NI          | 0011111111   | 09101112      |
| 2         | NE          | 0022222222   | 09111233      |
| 3         | MA          | 0033333333   | 09121314      |
| 4         | RE          | 0044444444   | 09131415      |
```

```
| ID_Movies | GENRE | AGE_GROUP | PRICE |
|-----------|-------|-----------|-------|
| 1         | HORROR | 12+       | 35000 |
| 2         | ACTION | 21+       | 40000 |
| 3         | DRAMA  | 13+       | 40000 |
| 4         | ANIMATION | ALL GENDER | 30000 |
| 5         | SCIENCE FICTION | ALL GENDER | 45000 |
```

```
| ID_TRANSACTION | ID_Movies | ID_TENANT | ID_ROOM | ID_CASHIER | TICKETS_PURCHASED | PRICE_FINAL |
|----------------|-----------|-----------|---------|------------|-------------------|--------------|
| 6              | 4         | 4         | 2       | 21         | 1                 | 60000        |
| 7              | 1         | 2         | 1       | 22         | 2                 | 70000        |
| 8              | 3         | 3         | 2       | 22         | 1                 | 60000        |
| 9              | 5         | 4         | 4       | 21         | 2                 | 90000        |
```

**Figure 4. Flower Venn Diagram Sample**

Figure 4 shows that cinemaTransaction centralizes all connected tables, and the keys built to relate to other tables are ID_CASHIER, ID_Movies, ID_ROOM, and ID_TENANT. The notation for the relationship keys as follows: cashier = {ID_CASHIER}, movies = {ID_Movies}, room = {ID_ROOM}, tenant = {ID_TENANT}, and cinemaTransaction = {ID_CASHIER, ID_Movies, ID_ROOM, ID_TENANT}; which means the table which had the same of the name and data type with all tables, it is connected to all tables. The table becomes a receptacle, and the other become petals. Therefore, cinemaTransaction is as a receptacle; cashier, movies, room, and tenant are petals.

When Figure 4 applied to SQL command, several stages had carried out as follows:
| No | SQL Command | Flower Venn Diagrams |
|----|-------------|---------------------|
| 1  | Select t.ID_TRANSACTION, c.ID_CASHIER, c.NAME_CASHIER from cinemaTransaction t INNER JOIN cashier c on c.ID_CASHIER=t.ID_CASHIER; | ![Flower Venn Diagram](image1) |
| 2  | Select t.ID_TRANSACTION, c.ID_CASHIER, c.NAME_CASHIER, m.GENRE, m.AGE_GROUP, m.PRICE from cinemaTransaction t INNER JOIN cashier c on c.ID_CASHIER=t.ID_CASHIER inner join movies m on m.ID_MOVIES=t.ID_MOVIES; | ![Flower Venn Diagram](image2) |
| 3  | Select t.ID_TRANSACTION, c.ID_CASHIER, c.NAME_CASHIER, m.GENRE, m.AGE_GROUP, m.PRICE, r.NAME_ROOM from cinemaTransaction t INNER JOIN cashier c on c.ID_CASHIER=t.ID_CASHIER inner join movies m on m.ID_MOVIES=t.ID_MOVIES INNER JOIN room r on r.ID_ROOM=t.ID_ROOM; | ![Flower Venn Diagram](image3) |
| 4  | Select t.ID_TRANSACTION, c.ID_CASHIER, c.NAME_CASHIER, m.GENRE, m.AGE_GROUP, m.PRICE, r.NAME_ROOM, tn.NAME_TENANT, tn.PHONE_NUMBER from cinemaTransaction t INNER JOIN cashier c on c.ID_CASHIER=t.ID_CASHIER inner join movies m on m.ID_MOVIES=t.ID_MOVIES INNER JOIN room r on r.ID_ROOM=t.ID_ROOM INNER join tenant tn on tn.ID_TENANT=t.ID_TENANT; | ![Flower Venn Diagram](image4) |

The query result based on number 4 in Table 1, as follows:

```
MariaDB [rektcinema]> select *
from cinemaTransaction t
INNER JOIN cashier c on c.ID_CASHIER=t.ID_CASHIER
INNER JOIN movies m on m.ID_MOVIES=t.ID_MOVIES
INNER JOIN room r on r.ID_ROOM=t.ID_ROOM
INNER JOIN tenant tn on tn.ID_TENANT=t.ID_TENANT;
```

| No  | ID_TRANSACTION | ID_CASHIER | NAME_CASHIER | GENRE | AGE_GROUP | PRICE | NAME_ROOM | PHONE_NUMBER |
|-----|----------------|------------|--------------|-------|-----------|-------|-----------|--------------|
| 1   | 6              | 21         | 53           | AVANT | ALL GENDER | 30000 | STUDIO 2   | RK           |
| 2   | 9              | 21         | 53           | SCIENCE | ALL GENDER | 45000 | STUDIO 1   | RK           |
| 3   | 7              | 22         | 10000        |       | 13x       | 13x   | 13x       | 13x          |
| 4   | 8              | 22         | 10000        |       | 13x       | 13x   | 13x       | 13x          |

4 rows in set (0.001 sec)

**Figure 5. The Query Result For Flower Venn Diagram**

Inner Join able to construct Flower Venn Diagram. However, the command must work sequentially for the formation of a Venn Diagram. In other words, Inner Join created a relationship table one by one. It was different when we used the Where clause, in which we create all petals such as cashier,
movies, room, and tenant. Afterward, a receptacle cinema transaction made last. The Where clause in the SQL command for constructing Flower Venn Diagram is: `SELECT transaction.ID_TRANSACTION, cashier.ID_CASHIER, cashier.NAME_CASHIER, movies.GENRE, movies.AGE_GROUP, movies.PRICE, room.NAME_ROOM from cashier, movies, room, tenant, transaction where cashier.ID_CASHIER=transaction.ID_CASHIER and movies.ID_MOVIES=transaction.ID_MOVIES and room.ID_ROOM=transaction.ID_ROOM and tenant.ID_TENANT=transaction.ID_TENANT`.

However, Flower Venn Diagram built becomes complicated when it created more than one Flower Venn. Although a pattern of Database Relationship becomes visible, two variant patterns illustrate as follows:

![Figure 6. Database Relationship Pattern](image)

The SQL command in Figure 6 had shown as follows:

```
Select E.Id IdTblE, A.Tbla_Clm1, B.TblB_Clm1, C.TblC_Clm1, D.TblD_Clm1, F.Tblf_Clm1, J.Id IdTblJ, G.TblG_Clm1, H.TblH_Clm1, I.TblI_Clm1 From tblE E, TblA A, TblB B, TblC C, TblD D, TblF F, tblJ J, TblG G, TblH H, tblI I where A.Id=E.RefTblA and B.Id= E.RefTblB and C.Id= E.RefTblC and D.Id= E.RefTblD and J.Id= E.RefTblE and J.RefTblF=f.Id and G.Id=J.RefTblG and H.Id=J.RefTblH and I.Id=J.RefTblI and F.id=c.id=f.RefTblC
```

And

```
Select E.Id IdTblE, A.Tbla_Clm1, B.TblB_Clm1, C.TblC_Clm1, D.TblD_Clm1, k.TblK_Clm1, L.TblL_Clm1, M.TblM_Clm1, N.TblN_Clm1 From tblE E join TblA A on A.Id=E.RefTblA join TblB B on B.Id= E.RefTblB join TblC C on C.Id= E.RefTblC join TblD D on D.Id= E.RefTblD join TblK K on A.Id= K.RefTblA join TblL L on L.Id=K. RefTblL join TblM M on M.id=K. RefTblM join TblN N on N.id=K. RefTblN
```
Based on the result of applying the Flower Venn Diagram, which had a simulation. It was simple to understand how the tables had connected because it organized. Sometimes, we add more another Flower Venn Diagram; we found that more complex and patterned. Therefore, All fields are conveniently searchable. However, it needs to investigate further whether the data in the Diagram become effective or not.

4. Conclusion
A Flower Venn Diagram that we proposed aims to map the dataset more organized and straightforward to present a Database. Based on the study result with simulation, the Join clause was created sequentially for a Flower Venn Diagram establishment. It different when we used the Where clause; it constructs directly.

However, Flower Venn Diagram built become complicated when it created more than one Flower Venn. Although a pattern of Database Relationship become visible and all fields are conveniently searchable. However, it needs to investigate further whether the data in the Diagram for effectively.

References

[1] R. Hughes, “Artifacts for the Enterprise Requirements Value Chain,” in Agile Data Warehousing for the Enterprise, Elsevier, 2016, pp. 181–213.
[2] C. Wu, R. Buyya, and K. Ramamohanarao, “Big Data Analytics = Machine Learning + Cloud Computing,” in Big Data: Principles and Paradigms, Elsevier Inc., 2016, pp. 3–38.
[3] M. West, “Some Types and Uses of Data Models,” in Developing High Quality Data Models, Elsevier, 2011, pp. 23–36.
[4] M. H. Liu et al., “Transcriptome analysis of leaves, roots and flowers of Panax notoginseng identifies genes involved in ginsenoside and alkaloid biosynthesis,” BMC Genomics, vol. 16, no. 1, 2015.
[5] G. Lin et al., “VennPainter: A Tool for the Comparison and Identification of Candidate Genes Based on Venn Diagrams,” PLoS One, vol. 11, no. 4, p. e0154315, Apr. 2016.
[6] A. M. Moustafa et al., “Comparative Genomic Analysis of Asian Haemorrhagic Septicaemia-Associated Strains of Pasteurella multocida Identifies More than 90 Haemorrhagic Septicaemia-Specific Genes,” PLoS One, vol. 10, no. 7, p. e0130296, Jul. 2015.
[7] H. K. Al-Masree, “Extracting Entity Relationship Diagram (ERD) From Relational Database Schem,” Int. J. Database Theory Appl., vol. 8, no. 3, pp. 15–26, 2015.
[8] T. Ijiri, S. Owada, M. Okabe, and T. Igarashi, “Floral diagrams and inflorescences: Interactive flower modeling using botanical structural constraints,” in ACM Transactions on Graphics, 2005, vol. 24, no. 3, pp. 720–726.
[9] A. C. Conrad and M. F. Mathabatha, “Elucidating variable traits of flower pigments in clivian plants’ species,” Vegetos, vol. 30, no. 1, pp. 9–23, Mar. 2017.
[10] I. Young, Mathematics for Public and Occupational Health Professionals. Pressbook, 2019.
[11] F. Y. Alnawajha, “The Performance of Inner Join Types in SQL,” Int. J. Adv. Stud. Comput. Sci. Eng., vol. 5, no. 11, pp. 164–168, 2016.