SMARTICS Journal, Vol.9, No.1, 2023, pp.42-48
ISSN online: 2476-9754, ISSN print: 2623-0429
DOI : https://doi.org/10.21067/smartics.v9i1.8390

Comparison of Relational Database Modeling Performance based on Number of Normalized Entities

Dika Rizky Yunianto[a,*], Yudistira Eka Putra[b], Cahya Rahmad[c]
[a,b]Politeknik Negeri Malang, Jl. Sukarno Hatta no.9, Malang, Indonesia
[b]Formulatrix Indonesia, Jl. Soekarno Hatta No.121,Salatiga, Indonesia
[*]correspondence email : dika.rizky@polinema.ac.id

Abstract— The database is the most important part of the development of a software. Management of data storage in the database is very important because the amount of data stored is increasing and varied. Thus requiring increased storage capacity and increased processing time. One form of database used in software is a relational database. Relational database modeling to reduce storage capacity can be circumvented by normalizing the database. From the experimental results, database normalization has an effect on the number of entities. So that the storage capacity and processing time in data processing are also affected by the normalization of the database. The more the number of entities will cause the more time needed in data processing.

Index Terms—Arduino; Blynk; Prototypes; Smart Class.

I. INRODUCTION

The database is a collection of data that is processed systematically and stored electronically. This data can be in words, numbers, images, videos, and files [1]. Utilization of the database is being used in various kinds of software applications. Application development is swift. In 2020 there was a surge in the number of existing applications in the world by 19% from the previous year [2].

The development of these applications also influences and is directly proportional to the existing development of data. For example, YouTube is a software application or platform for playing or saving videos. Until 2018, YouTube has used 45 Terabytes in its data transaction process, and this will increase later in the upcoming years [3]. This data growth causes the need for ample data storage over time. Because of that, the management and optimization of data storage in databases become exciting research area to develop.And also data modeling is another important thing in database management.

Data Modeling is a logical management of objects in the real world. These objects can be referred to as entities [4]. Entities themselves are things, people, places, units, or objects that represent data stored in databases [5]. The data model itself is defined as a logic of data structures, data operators and so on that interact with users [6].

One form of data modeling is the relational model. Relational and non-relational databases are models that have differences in performance, operation and format variations. In a relational database, data is stored in tabular form where a table represents an entity. These tables are interconnected to form a relationship [7].

The basic concept of data modeling in relational databases is the schema. The schema is the database itself. Within the schema, there are several entities and each entity has its own attributes. The attribute itself is a characteristic of an entity. For example, a database or schema “enterprise” has an entity “employee”. The “employee” entity has the attributes “employee_id”, “employee_name”, “address” and other attributes [5].

In forming a relational database, Edgar Codd as a relational database inventor introduced the concept of normalization. Normalization is the process of managing database models to reduce redundancies or data repetition. Usually, the normalization process will divide an entity or table into two or more and define the relationships between these tables [8].

The normalization process carried out to form a relational database design is expected to have an impact on storage capacity and processing time. Therefore, this journal examines the effect of database normalization results on storage capacity and processing time. The design form of the database is expected to be able to optimize the required storage capacity given the development of an increasing amount of data.

II. METODOLOGY

The data used for the example of processing the relational database design is fast food restaurant data. This data can be retrieved on data.world. [9]. The data has a total of 10,000 rows with 16 columns.

The normalization process is carried out to avoid data redundancy. This process breaks a table or entity to become stand-alone and have relationships with each other [10]. The 1st normalization (N1) is the process of solving data contents that contain more than one object. The data is split into new rows. An example of N1 modeling can be seen in Table 1 and Table 2.

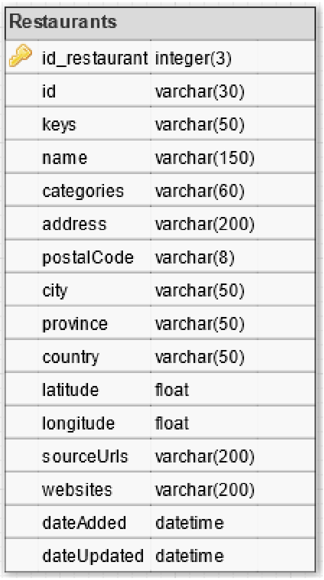
TABLE 1. DATA BEFORE NORMALIZED

Id	Name	Category
1	Rosewood Fast	Fast Food Restaurant, Burger Stand
2	Burgarry Dine	Fast Food Restaurant, Breakfast Restaurant
3	King Resto	Fast Food Restaurant

TABLE 2. THE TABLE AFTER 1ST NORMALIZATION (N1)

Id	Name	Category
1	Rosewood Fast	Fast Food Restaurant
2	Rosewood Fast	Burger Stand
3	Burgarry Dine	Fast Food Restaurant
4	Burgarry Dine	Breakfast Restaurant
5	King Resto	Fast Food Restaurant

The data used after 1st normalization (N1) becomes a database design which is described in the form of Physical Data Modeling (PDM) as shown in Figure 1.



Restaurants	
id_restaurant	integer(3)
id	varchar(30)
keys	varchar(50)
name	varchar(150)
categories	varchar(60)
address	varchar(200)
postalCode	varchar(8)
city	varchar(50)
province	varchar(50)
country	varchar(50)
latitude	float
longitude	float
sourceUrls	varchar(200)
websites	varchar(200)
dateAdded	datetime
dateUpdated	datetime

Figure 1. PDM first normalization (N1).

The database model design that has been normalized for the first time (N1) is reprocessed for the 2nd normalization (N2). Examples of 2nd normalization (N2) can be seen in Table 4 and Table 5 where the 2nd normalization (N2) is carried out after 1st normalization (N1. The 2nd normalization (N2) results make the table split into two, where the two tables have a relation connected by the id_categories column.

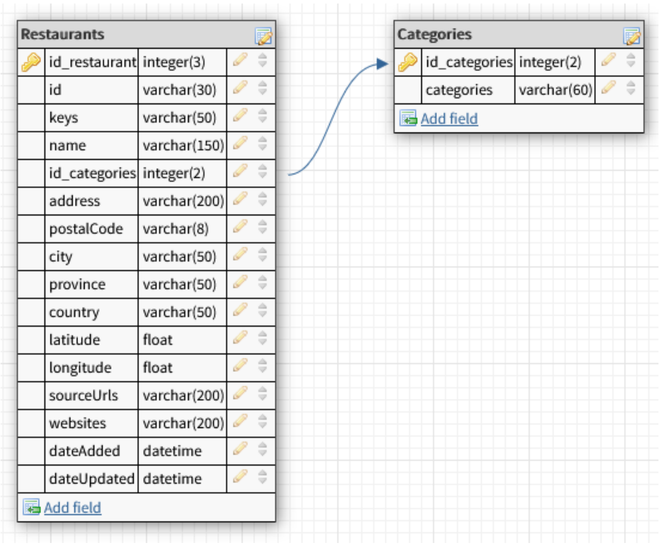
Table 4. The restaurant table, result of 2nd normalization (N2)

Id	Name	Id_Categories
1	Rosewood Fast	1
2	Rosewood Fast	2
3	Burgarry Dine	1
4	Burgarry Dine	3
5	King Resto	1

Table 5. The categories table, result of 2nd normalization (N2)

Id_Categories	Categories
1	Fast Food Restaurant
2	Burger Stand
3	Breakfast Restaurant

The PDM results from the 2nd normalization (N2) on the data can be seen in Figure 2.



Restaurants	
id_restaurant	integer(3)
id	varchar(30)
keys	varchar(50)
name	varchar(150)
id_categories	integer(2)
address	varchar(200)
postalCode	varchar(8)
city	varchar(50)
province	varchar(50)
country	varchar(50)
latitude	float
longitude	float
sourceUrls	varchar(200)
websites	varchar(200)
dateAdded	datetime
dateUpdated	datetime

Categories	
id_categories	integer(2)
categories	varchar(60)

Figure 2. PDM of 2nd normalization (N2).

The next step is to carry out the efficiency of the database design model with the 3rd normalization (N3). Examples of the third normalization (N3) can be seen in Table 6, Table 7, Table 8 and Table 9.

Table 6. The restaurant table, result of 3rd normalization (N3)

Id	Name	Id_postal_code
1	Rosewood Fast	1
2	Burgarry Dine	1
3	King Resto	2

Table 7. The restaurant_categories table, result of 3rd normalization (N3)

Id	Id_restaurant	Id_Categories
1	1	1
2	1	2
3	2	1
4	2	3
5	3	1

Table 8. The postal_code table, result of 3rd normalization (N3)

Id_postal_code	Postal_code	Id_city
1	77163	1
2	77144	1

Table 9. The city table, result of 3rd normalization (N3)

Id_city	city	Id_district
1	New York	1
2	Miami	2

The PDM results from the 3rd normalization (N3) on the data can be seen in Figure 3.

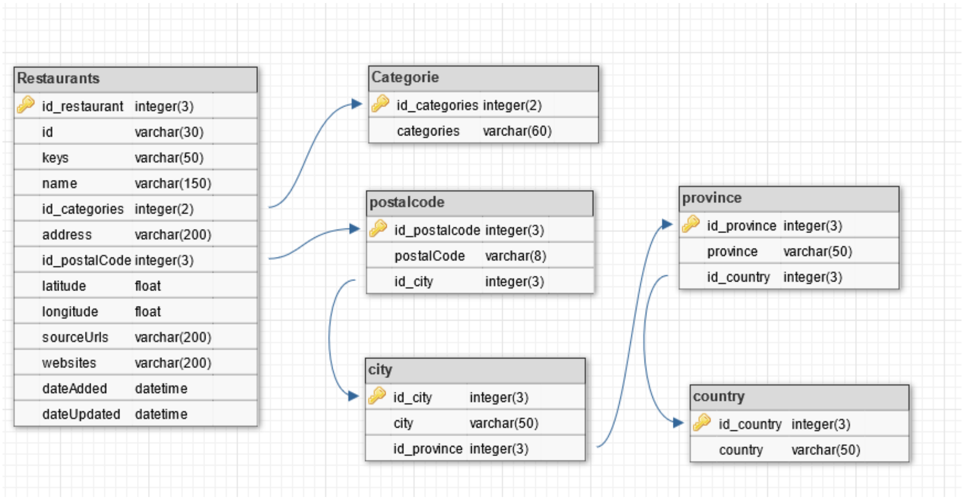


Figure 3. PDM of 3nd normalization (N3).

III. RESULT AND DISCUSSION

The normalized database design model will be analyzed at the memory capacity being used when filled with data and there is no data. The comparison of the memory capacity used can be seen in Table 10 and Figure 4.

Table 10.The Storage Memory Capacity when there is no data.

Database Model	Entities Count	Memory Capacity (Kb)
N1	1	1803
N2	2	2312
N3	6	4238

When the database model has been implemented using the Data Definition Language (DDL) in MySQL, the storage space used is different in each normalization model. The database model with the first normalization (N1) has a smaller storage space capacity compared to the storage space capacity of the database model with the second normalization (N2). Likewise, the storage space capacity of the second normalization (N2) is also smaller than the storage space capacity of the database model with the third normalization (N3).

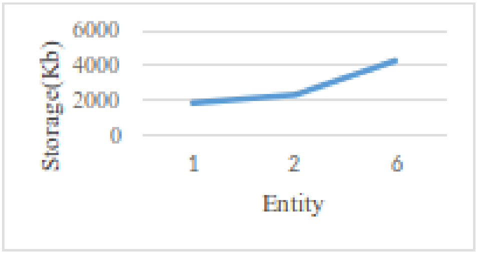


Figure 4. Storage Memory Graph when there is no data.

When viewed from the entities count perspective, the more the number of entities will have resulted in greater the required storage space capacity. However, this applies too when there is no data or the database is empty. The state of storage space capacity when the data is filled can be seen in Table 11 and Figure 5.

Table 11. The Storage Memory Capacity when filled with data.

Database Model	Entities Count	Memory Capacity (Kb)
N1	1	4092
N2	2	4009
N3	6	3950

As we can see from Table 10 and Table 11 there is an increase and decrease in storage space capacity due to different circumstances. The difference is when filled with data and there is no data.

When there is no data inside the database, the normalization process will increase the required storage space capacity. However, when the database is already filled, the normalization process helps to reduce storage space capacity.

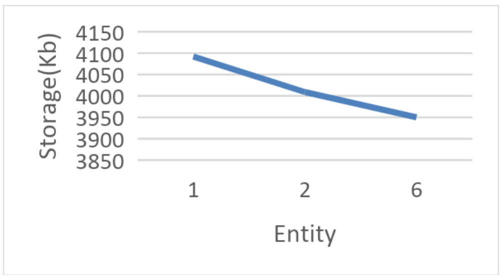


Figure 5. Storage Memory Graph when when filled with data.

The database with normalization modeling N1, N2, and N3 also has different processing times. Testing will be carried out by looking at the processing time of the entered SQL query. The first SQL query that is processed is the query that is used to display the same column. Differences in normalization design cause differences in the number of tables involved in the processing. The SQL query can be seen in Table 12.

Table 12. SQL Query for First Testing Scenario

Model	SQL Query
N1	select * from restaurants;
N2	select * from restaurants, categories where restaurants,id_categories=categories,id_categories;
N3	select * from restaurants, categories,city,country,postalcode,province where restaurants,id_categories=categories,id_categories and restaurants,id_postalCode=postalcode,id_postalco de and postalcode,id_city=city,id_city and city,id_province=province,id_province and province,id_country=country,id_country;

From the First Testing Scenario SQL Queries, different processing times are obtained to get the same result. The results of the query processing time can be seen in table 13 and Figure 6.

Table 13. Processing Time of First Testing Scenario

Database Model	Join Table Count	Processing Time
N1	1	$0{,}37 \times 10^{-3}$
N2	2	$0{,}7 \times 10^{-3}$
N3	6	$0{,}8 \times 10^{-3}$

The more JOIN tables that are performed in query processing, the longer it takes to process SQL Queries.

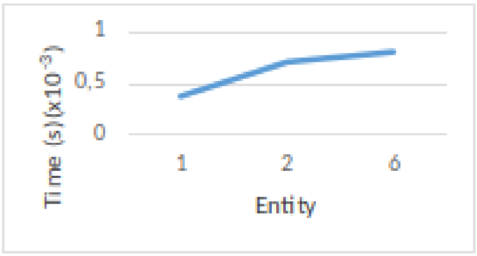


Figure 6. The First Testing Scenario Processing Time Graph

The next scenario is to use an Aggregate Function of an SQL Query such as the count() function. The SQL used can be seen in table 14.

Table 14. SQL Query for Second Testing Scenario

Database Model	SQL Query
N1	select count(*) from restaurants;
N2	select count(*) from restaurants;
N3	select count(*) from restaurants;

From the Second Testing Scenario SQL Queries, different processing times are obtained to get the same result. The results of the query processing time can be seen in table 15 and Figure 7.

Table 15. Processing Time of Second Testing Scenario

Database Mode	Join Table Count	Processing Time
N-1	1	$3,75 \times 10^{-2}$
N-2	1	$0,5 \times 10^{-2}$
N-3	1	$0,2 \times 10^{-2}$

The processing time to run the aggregate function greatly affects by the size of the data, both the amount of data and the number of columns in the table. The bigger the size of data in a table, the longer it takes to run the aggregate function process from the SQL Query.

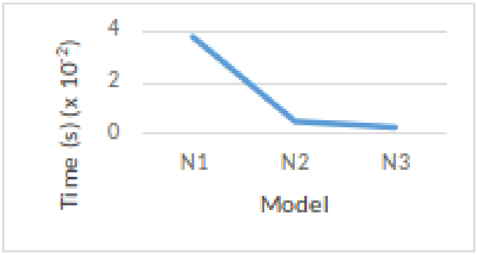


Figure 7. The First Testing Scenario Processing Time Graph.

IV. CONCLUSION

In creating a relational database, it is necessary to design and model the database. Database modeling that applies the normalization process will increase the number of entities or tables that are formed. When the database is empty, more tables will make more required storage capacity. However, when the database is filled with data, more tables will make less required storage capacity.

The number of tables that increase due to the normalization process also affects the processing time. The more tables needed in an SQL Query processing, the longer it takes to carry out the process. From this statement, it can be used as a guide in designing a relational database when it requires fast processing time or more efficient storage capacity.

REFERENCES

[1] A. Amazon, "What Is A Database?," 2023. [Online]. Available: https://aws.amazon.com/what-is/database/?nc1=h_ls.
[2] J. Koetsier, "There Are Now 8.9 Million Mobile Apps, And China Is 40% Of Mobile App Spending," Forbes, 28 February 2020.
[3] C. B. Products, "Top 10 Largest Databases in the World," Compare Business Products, 2018.
[4] W. Kim, "Object-Oriented Databases: Definition and Research Directions," IEEE Transactions on Knowledge and Data Engineering, Vol 2, No 3, pp. 327-340, 1990.
[5] R. Elmasri dan S. Bavthe, Fundamentals of Database Systems: Seventh Edition, 2021.
[6] C. J. Date, Database Design and Relational Theory: Normal Forms and All That Jazz, 2019.
[7] N. Thakur dan N. Gupta, "Relational and Non Relational Databases: A Review," Journal of University of Shanghai for Science and Technology, Vol 23, Issue 8, 2021.
[8] D. Masri, Developing Data Migrations and Integrations with Salesforce: Patterns and Best Practices, New York: Apress, 2019.
[9] Datafiniti, "Fast Food Restaurant Across America," 2018.
[10] J. A. Bachman, B. M. Gyori dan P. K. Sorger, "Automated assembly of molecular mechanisms atscale from text mining and curated databases," Molecular Systems Biology, 2023.