International Journal of Engineering & Technology, 8 (1.1) (2019) 143-148

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Converting Relational Database into Temporal Database

Yogiek Indra Kurniawan[1]*, Kusuma Ayu Laksitowening[2], Ade Romadhony[3] , Muhammad Fachrie[4], Dimas Aryo Anggoro[5]

[1]Informatics, Universitas Muhammadiyah Surakarta
[2]informatics, Telkom University
[3]informatics, Telkom University
[4]Department of Informatics, Universitas Teknologi Yogyakarta
[5] Informatics, Universitas Muhammadiyah Surakarta
**Corresponding author E-mail: yogiek@ums.ac.id*

Abstract

In this paper, perform the implementation of temporal database which take into account the historical aspects of data and how to convert relational database into temporal database. After performing the implementation, an analysis of queries is done in accessing each data that takes into account the historical aspects of data, for Data Definition Language (DDL), Data Manipulation Language (DML) and queries for retrieve data. After the analysis, it is concluded that the use of queries for temporal database can be done on the relational database with some adjustments. In addition, temporal databases have advantages in terms of storage usage and response time for DDL and DML queries, but for response time in query retrieve data takes longer than relational database.

Keywords: *historical data; relational database; temporal database; time dimension.*

1. Introduction

In this era, many applications require data from the past and need data for the future. This data is usually used to track events that happen to see trends and find errors in the past so user can prevent the occurrence of the same error in the future. Unfortunately, on the relational database that is often used now, less support for things like this[8].
One solution to the problem of track events from the past is use temporal databases. Temporal database is a database that supports temporal aspects beyond user defined time, with one or more time dimensions [2][6]. This allows data changes in a period will not delete data from the past. On the other hand, another approaches that can be done to solve the problem above is to add the time attribute on the relational database[5].
In this paper, implement historical data by using temporal database as well as relational database modified and analyzing the queries in both databases, also analyze the aspect of storage usage and response time to measure the effectiveness of each database.
The problem that is used as research object in this paper is to implement and analyze query in temporal database compared to query in relational database which take into account the historical aspect of data, and also to know the effectiveness of temporal database usage.

2. Research Method

The temporal database is a database that supports temporal aspects beyond the time defined by the user [1][7]. The temporal database allows associating facts with time. The temporal database is an enrichment of a relational database that takes into account the time aspect. In the conventional database, does not take care of the historical aspects of data validity in the database.
The temporal database supports handling of complex time aspects and can store historical aspects of data. The data will be marked by the time of its validity in the real world, so that data will have "history" from the past to the future. In addition, there are special operators for time that do not exist in conventional databases, such as operators to know two overlapping periods of time.
The research method used for problem solving and research in this study is divided into several stages as follows[4]:

2.1 Literature Review

Bowman et all[3] created a standard for using structured query language(SQL) in relational database that using until now as SQL92. This language is used in all Database Management Systems (DBMS) because easy to use and easy to understand.
In 2012, Snodgrass[9] created a language to facilitate historical aspect of data, named Temporal Structured Query Language (TSQL2) that is used in temporal database. After that, the temporal database developed and began to be used by the public. Research on the temporal database also began to develop with the use of the database in real cases. In 2018, [7] made a survey of temporal database research used by subsequent researchers to develop research in the temporal database.

Copyright © 2019 Authors. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Recent development in temporal database make [8] created temporal database into modern DBMS and data warehouses. Temporal database also can be used in other real case in the world. It shows the rapid research about development of temporal database.

2.2 Systems Design and Model

At this stage, a model of temporal database was designed. After that, relational database also was designed based on temporal database design.
There are many applications that can implement temporal databases, such as the Office app, weather monitoring apps, hospital apps, stock exchange apps and more. Such applications can use temporal databases due to data changes every time. So for the historical aspects of data, these applications can be used.
In this research, application that implemented is application of data monitoring at shops. There are 2 pieces of application with different databases. This is to support the historical aspect of each database.
Figure 1 is a schema diagram design in the temporal database of the shopping system case study.

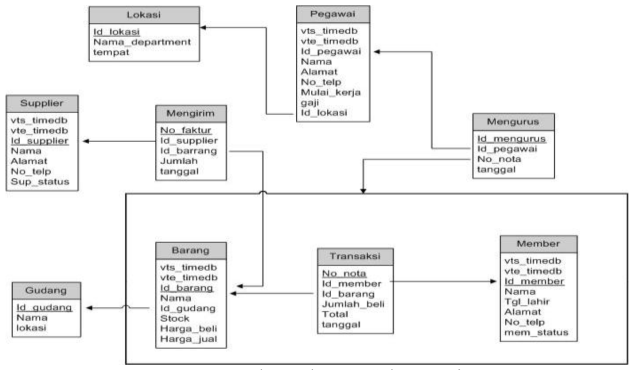


Fig. 1: Temporal Database's Schema Diagram

In figure 1, data showing the temporal database is placed in the table of employees, members, goods and suppliers. In the 4 tables are taking into account the historical aspects of data as follows:

a. *Pegawai* (Employee) table. In the employee table, will be shown historically about the salary of an employee and id_lokasi showing the position and place of employment of an employee.
b. Member table. In the member table, will be shown historically about the address and status of a member.
c. *Barang* (Goods) table. In the goods table, will be shown historically about the stock, the purchase price and the selling price of a good.
d. Supplier table. In the supplier table, will be shown historically about address and status of the supplier concerned.

To give the time dimension of valid time in 4 pieces of the table, then attribute vts_timedb (valid time start for timedb) and vte_timedb (valid time end for timedb) added.
Figure 2 is a schema diagram design in a relational database to support historical aspects of data from case studies of shopping systems that include temporal time dimensions in the table of employees, members, goods and suppliers with reference to previous temporal databases.
From figure 2, to support aspects of temporal data, so in each table will be created a historical table to store any changes in data that occur in the tables, as follows:

a. *Pegawai* (Employee) table. In the employee table will be created a table named history_pegawai which has id_pegawai column to refer to the table of employees, salary column, location and id_lokasi showing the position and place of work of an employee to show historical data in the table employees.
b. Member table. In the member table will be created a table named history_member which has a column id_member to refer to the member table and the address and status of a member to show historical data in the member table.
c. Barang (Goods) table. In the goods table, a table named history_barang has the column id_barang as the foreign key to refer to the table of goods and the stock column, harga_beli and harga_jual to show historical data in the goods table.
d. Supplier table. In the supplier table, a table named history_supplier has the id_supplier column as the foreign key to refer to the supplier table and the address and status fields as historical data from the supplier table.
e.

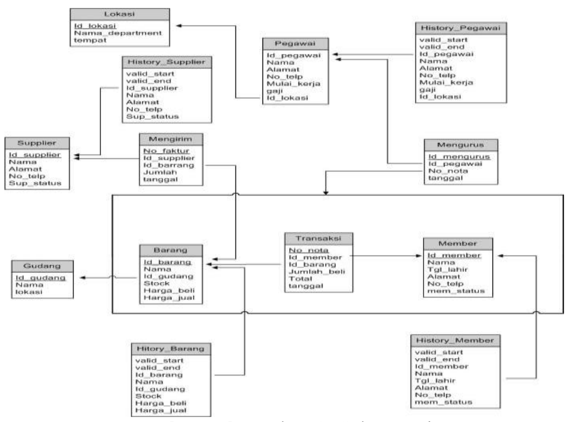


Fig. 2: Relational Database's Schema Diagram

To give the time dimension of valid time in 4 pieces of the table, then attribute valid_start and valid_end added.

2.3 Implementation

From the design of temporal database and relational database that have been created, then both are implemented to be 2 pieces of the database into 2 Java-based application with Oracle database 10g. Each application implements 1 specific database. The application will be used as testing material.

2.4 Testing

From the 2 pieces of existing applications, tested the query for each database. First, same data is added on both databases. After that, some query, divided into Data Definition Language (DDL) and Data Manipulation Language (DML), is implemented into both databases. TSQL2 as Temporal queries [9] are performed on applications with temporal databases. Once the results are obtained, SQL92, as relational queries[3], are implemented into the relational database to get the same result with the temporal query. Each query will be calculated response time as a measuring tool in testing.

3. Results and Discussions

Results from this paper can be explained as follow :

3.1 Converting Temporal Queries to Relational Queries

There are 2 application built, an application using temporal database and the other using relational database. After application has built, 10,000 data has added into both databases. TSQL2 as Temporal queries and SQL 92 as relational queries are performed on both applications to get the same result. Table 1 shows how to convert temporal queries into relational queries to get the same result.

Table 1: Converting Temporal Queries into Relational Queries

No	Aspect	Temporal Query (TSQL2)	Relational Query (SQL92)
(1)	Create table gudang	create table gudang(id_gudang number(10), nama varchar(40), lokasi varchar(35), primary key(id_gudang));	create table gudang(id_gudang number(10), nama varchar2(40), lokasi varchar2(35), primary key(id_gudang));
(2)	Insert table gudang	insert into gudang values (1001,'Gudang Barang A Pintu 1 Jakarta','Jakarta');	insert into gudang values (1001,'Gudang Barang A Pintu 1 Jakarta','Jakarta');
(3)	Drop table gudang	Drop table gudang;	Drop table gudang;

(4)	Create table barang	create table barang(id_barang number(10), nama varchar(55), id_gudang number(10), stock number(5), harga_beli number(8), harga_jual number(8), primary key(id_barang), foreign key(id_gudang) references gudang(id_gudang))as validtime;	create table barang(id_barang number(10), nama varchar2(55), id_gudang number(10), stock number(5), harga_beli number(8), harga_jual number(8), constraint pk_barang primary key(id_barang), constraint fk_barang foreign key(id_gudang) references gudang);
			
			 create table history_barang(valid_start date, valid_end date, id_barang number(10), nama varchar2(55), id_gudang number(10), stock number(5), harga_beli number(8), harga_jual number(8), constraint fk_history_barang foreign key(id_barang) references barang);

(5)	Drop table barang	Drop table barang;	Drop table history_barang; Drop table barang;
(6)	Insert table barang	validtime period [2010/01/01-2011/01/02) insert into barang values (200001,'Handphone :Nokia 8800 Sapphire Arte',1001,500,14550000,16005000);	insert into barang values (200001,'Handphone :Nokia 8800 Sapphire Arte',1001,500,14550000,16005000); insert into history_barang values ('2010/01/01','2011/01/02',200001,'Handphone :Nokia 8800 Sapphire Arte',1001,500,14550000,16005000);
(7)	Update barang	Update barang set nama='Nokia 6300' where id_barang=201;	Update barang set nama='Nokia 6300' where id_barang=201;
(8)	Update barang, Change valid_start= 2010/01/01 and valid_end= 2010/10/10 for id_barang= 200001	nonsequenced validtime period [2010/10/11 – 2011/01/02) delete from barang where id_barang=200001;	Update history_barang set valid_start='2010/01/01', valid_end='2010/10/10' where id_barang=200001;
(9)	Delete barang,for barang on 2010/04/01 to 2010/06/01 for barang that have id_barang= 200001	nonsequenced validtime period [2010/04/01 – 2010/06/01) delete from barang where id_barang=200001;	Update history_barang set valid_start=2010/06/01,valid_end=2011/01/02 where id_barang=200001; Insert into history_barang values ('2010/01/01','2010/03/31','200001',(select stock from barang where id_barang=200001), (select harga_beli from barang where id_barang=200001), (select harga_jual from barang where id_barang=200001));
(10)	Delete barang,for id_barang= 200001	Delete from barang where id_barang=200001	Delete from history_barang where id_barang=200001; Delete from barang where id_barang=200001;
(11)	Show all data barang	validtime select * from barang;	Select * from history_barang;
(12)	Show data barang that include in id_gudang	Validtime select * from barang where id_gudang=1001;	Select * from history_barang where id_gudang=1001;

	1001.		
(13)	Show data barang that valid for year 2010.	Validtime period [2010-2011) select * from barang;	Select * from history_barang where valid_start > ‘2010/01/01’ and valid_end < ‘2011/01/01’;
(14)	Operator PRECEDES	Validtime select * from barang where validtime (barang) precedes period [2010/01/01-2011/06/01);	Select * from history_barang where valid_end < ‘2010/01/01’;
(15)	Operator =	Validtime select * from barang where validtime (barang) = period [2010/01/01-2011/06/01);	Select * from history_barang where valid_start=’2010/01/01’ and valid_end=’2011/06/01’;
(16)	Operator OVERLAPS	Validtime select * from barang where validtime (barang) overlaps period [2010/01/01-2011/06/01);	Select * from history_barang where valid_start > ‘2010/01/01’ or valid_end < ‘2011/06/01’;
(17)	Operator perbandingan CONTAINS	Validtime select * from barang where validtime (barang) contains period [2010-2011);	Select * from history_barang where ((valid_start > ‘2010/01/01’ or valid_end < ‘2011/01/01’) and (valid_start > ‘2010/01/01’ and valid_end < ‘2011/01/01’));
(18)	Operator MEETS	Validtime select * from barang where validtime (barang) meets period [2010/01/01-2011/06/01);	Select * from history_barang where valid_end < ‘2009/12/31’;

For the numbers (1), (2) and (3) in table 1, the queries are performed equally between temporal and relational databases. This is due to the absence of a special time dimension for the three queries. For number (5) also has the same query. This is because between SQL'92 and TSQL2 for drop tables have the same query.
To create table at number (4), on temporal simply by using TSQL2 query to create. In the relational database, must create 2 pieces of tables ie tables of ‘barang’ and table history_barang, and a trigger to insert data in table history_barang while insert into the table of goods. This is done to create a table history_barang accommodate every history of the data in the table of goods.
At number (6), the insert on the temporal table will add into the table in question, while on relational, the insert will add into the table of ‘barang’ and table history_barang caused by the trigger.
In the number (7), for the update process on the attribute, temporal database can not update, so the update process uses the same query from relational and temporal.
In the number (9), for the process of deleting the validtime table, the temporal will delete the time that is not in accordance with the needs so that it can cause the existence of 2 data in the goods table. While on relational, will change the time that is not in accordance with the needs of the goods table and history_barang and enter data into table history_barang for the time according to the needs.
In the number (10), for the process of deleting data, the temporal will delete the data in the goods table, while in temporal will delete the data in the table history_barang as well as data on the table of goods. Deletion of data starts from table history_barang because of the trigger that refers to table history_barang.
In the numbers (11) and (12), for the process of viewing the data in the temporal table, there are already own query using keyword "validtime", while in relational, the selection process is done to the history table.
At number (13), for the process of viewing data with a certain period of time, the temporal there is a separate query by using the keyword "valid time period", while the relational will use the selection process on valid start and valid end.
At number (14), the comparison operator "PRECEDES" is the operator that defines the END of the data earlier than BEGIN on the operator. In temporal, simply use the keyword "precedes period", while on relational using the selection “valid_end > BEGIN” on the operator.
At number (15), the comparison operator "=" is the operator defining that valid_start and valid_end of the data must be equal to the operator. In temporal, simply use the keyword "= period", while on relational using the selection “valid_start = BEGIN and valid_end = END” on the operator.
At number (16), the comparison operator "OVERLAPS" is the operator that defines there is a slice between the event in the data and the event on the operator. In temporal, simply use the keyword "overlaps period", while on relational using the selection “valid_start> BEGIN or valid_end <END” on the operator.
At number (17), the comparison operator "CONTAINS" is the operator that defines that every event on the operator is in the data. In temporal, simply use the keyword "contains period", while the relational uses selection “((valid_start> BEGIN or valid_end <END) and (valid_start> BEGIN and valid_end <END))” on the operator.
At number (18), the comparison operator "MEETS" is the operator that defines the END of data earlier than BEGIN on the carrier and between the END in the data with BEGIN on the continuous operator. In temporal, simply use the keyword "meets period", while the relational using “valid_end <(BEGIN-1 Chronon)” selection on the operator.
Overall, The temporal query is easier and shorter than relational query if the accessed data is historical data. In addition, for temporal queries have a more complete time operator, whereas in relational must translate the operator's time into a query that is understood by relational. In addition, the lack of temporal query in this research is temporal query does not support any update. So if there is an update on valid time, then must be adjusted by using query delete. Moreover, temporal queries are less familiar, because temporal databases are still not too exploratory.

3.2 Response Time Analysis

For the response time, from all queries that have been given, relational databases are relatively superior in query retrieve data, but for DDL and DML queries, the response time shows the temporal have advantage from relational. Figure 3 shows response time calculation results from temporal and relational queries.
Basically, queries for temporal databases as well as relational databases are the same when DMBS is accessed by each database. It's only the temporal database must pass through a middleware for parsing and query translation first.
Figure 4 shows process in temporal database while accessing a query. The process is as follows: The access time calculation begins when the Application sends TSQL2 query shown by number (1). In Executor, the query is forwarded to Converter. The most important process in middleware is shown by number (3), because in the converter the process of parsing (query solving becomes tokens) as well as the translation process (converting tokens into SQL'92 queries understood by RDBMS/Relational Database Management Systems). After

that, SQL'92 query will be forwarded to the executor to run on RDBMS. The result of SQL'92 query execution on RDBMS will be sent to Executor and proceed to Converter. In the Converter, there will be a process of parsing and translation of the results in Relational to the results on the Temporal. Finally, the result is a temporal table that is sent to the application, and is calculated from the beginning to the end of execution as response time.

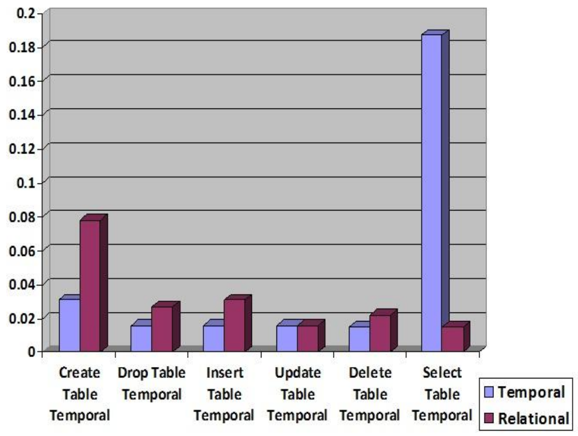


Fig. 3: Temporal and Relational's Response Time

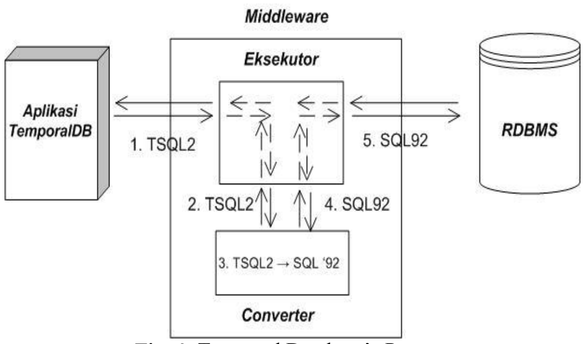


Fig. 4: Temporal Database's Process

While on relational database, the application directly accesses to RDBMS. The process of relational can be described in figure 5.

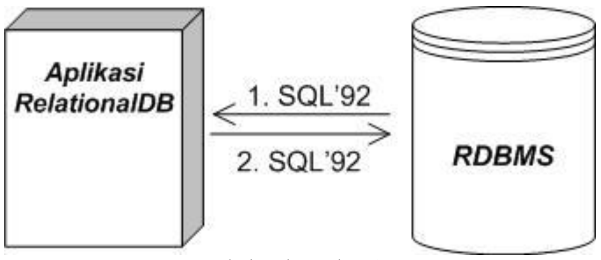


Fig. 5: Relational Database's Process

The process in relational database is as follows: The time will be calculated when the application sends SQL '92 query to the RDBMS. After that, RDBMS will send query results to the application. The time will be counted as the response time when the result has been received by the application.

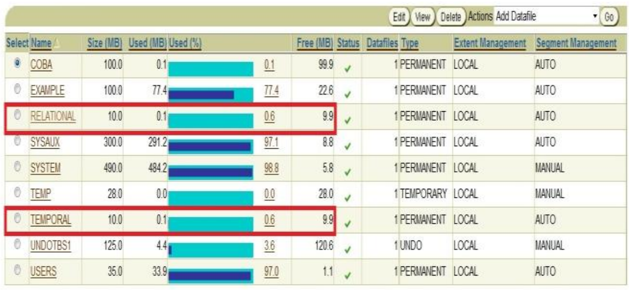
In DDL and DML queries, the temporal database process only takes 1 direction from the application to the RDBMS, without returning to the application, can be shown by process number (1) through number (5). Therefore, temporal's response time is faster than relational, since relational databases occur twice in every temporary table access, ie in the main table and in the history table.

While in the query retrieve data, the process on Temporal database is row per row of data generated by RDBMS, will always be translated into TSQL2 form into the application. The results of the translations are shown in the application. This causes the time required by temporal database to be longer than relational database because on relational, the process only happens once. When the result table is generated by the RDBMS, it is sent to the relational application and the result is displayed by the application.

3.3 Storage Analysis

Storage discussed here is the use of storage media in each database with the same data usage. It is used to measure the effectiveness of the use of storage media in each database model.

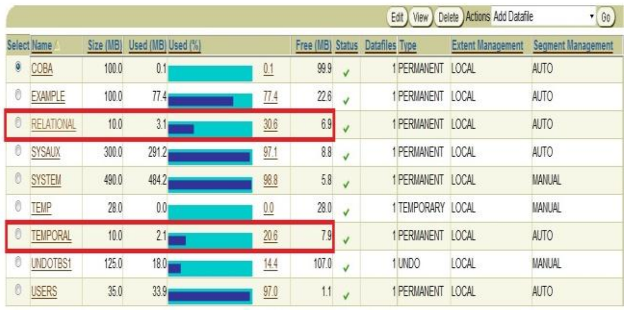
At the beginning before the data is added, 2 tablespaces are created that represent the initial storage usage for 2 databases to be compared. Obtained data can be seen in figure 6 as follows:



Select	Name	Size (MB)	Used (MB)	Used (%)	Free (MB)	Status	Datafiles	Type	Extent Management	Segment Management
●	COBA	100.0	0.1	0.1	99.9	✓	1	PERMANENT	LOCAL	AUTO
○	EXAMPLE	100.0	77.4	77.4	22.6	✓	1	PERMANENT	LOCAL	AUTO
○	RELATIONAL	10.0	0.1	0.6	9.9	✓	1	PERMANENT	LOCAL	AUTO
○	SYSAUX	300.0	291.2	97.1	8.8	✓	1	PERMANENT	LOCAL	AUTO
○	SYSTEM	490.0	484.2	98.8	5.8	✓	1	PERMANENT	LOCAL	MANUAL
○	TEMP	28.0	0.0	0.0	28.0	✓	1	TEMPORARY	LOCAL	MANUAL
○	TEMPORAL	10.0	0.1	0.6	9.9	✓	1	PERMANENT	LOCAL	AUTO
○	UNDOTBS1	125.0	4.4	3.6	120.6	✓	1	UNDO	LOCAL	MANUAL
○	USERS	35.0	33.9	97.0	1.1	✓	1	PERMANENT	LOCAL	AUTO

Fig. 6: Early Tablespace

After the data are added with the same data usage, the comparison of storage usage can be seen in figure 7 is as follows:



Select	Name	Size (MB)	Used (MB)	Used (%)	Free (MB)	Status	Datafiles	Type	Extent Management	Segment Management
●	COBA	100.0	0.1	0.1	99.9	✓	1	PERMANENT	LOCAL	AUTO
○	EXAMPLE	100.0	77.4	77.4	22.6	✓	1	PERMANENT	LOCAL	AUTO
○	RELATIONAL	10.0	3.1	30.6	6.9	✓	1	PERMANENT	LOCAL	AUTO
○	SYSAUX	300.0	291.2	97.1	8.8	✓	1	PERMANENT	LOCAL	AUTO
○	SYSTEM	490.0	484.2	98.8	5.8	✓	1	PERMANENT	LOCAL	MANUAL
○	TEMP	28.0	0.0	0.0	28.0	✓	1	TEMPORARY	LOCAL	MANUAL
○	TEMPORAL	10.0	2.1	20.6	7.9	✓	1	PERMANENT	LOCAL	AUTO
○	UNDOTBS1	125.0	18.0	14.4	107.0	✓	1	UNDO	LOCAL	MANUAL
○	USERS	35.0	33.9	97.0	1.1	✓	1	PERMANENT	LOCAL	AUTO

Fig. 7: Comparison Tablespace at the End

From the figure 7, it can be seen that the use of storage for relational database is larger than the use of relational temporal storage. This is because on the relational database must create a new table to store historical data. In addition, some triggers should be created to include historical data in the new table. While in the temporal database, the data is sufficiently represented on a temporal table only. In the temporal table there can be 2 pieces of data with the same primary key as long as it has a different valid start and valid end.

4. Conclusion

Based on the query analysis that has been done, the use of queries in temporal database can be implemented on relational database with some changes in each query performed. Furthermore, the use of storage for temporal databases is less than the use of storage for relational databases that apply historical aspects of data because each table and the contents of the data in the relational database must be copied to obtain the history of the data.

One disadvantage of temporal databases is that it takes longer time to access than relational databases in query retrieve data because the temporal database must pass through a middleware for parsing and query translation first for each result of the data table. As for DDL and DML queries on the temporal table will be faster in the temporal database, this is due to the temporal database accessing only 1 table, while the relational database will access 2 tables. On the other hand, temporal database has advantages in accessing queries to access temporal data because if using relational database must be done some modifications to the query to get the same results with the query on the temporal database.

Acknowledgement

The work was prepared with the support of the TimeDB as a tool for this research.

References

[1] Andreas, B., Dieter, G., Sahar, V., & Hua, L. Z. (2018, January). Reasoning Under Situation Awareness in a Temporal Database. In 2018 Second IEEE International Conference on Robotic Computing (IRC) (pp. 387-391). IEEE.
[2] Böhlen, M. H., Dignös, A., Gamper, J., & Jensen, C. S. (2017, July). Temporal Data Management–An Overview. In Europe-an Business Intelligence and Big Data Summer School (pp. 51-83). Springer, Cham.
[3] Bowman, J. S., Emerson, S. L., & Darnovsky, M. (1996). The practical SQL handbook: using structured query language. Addison-Wesley Longman Publishing Co., Inc..
[4] Kurniawan, Y. I., Soviana, E., & Yuliana, I. (2018, June). Merging Pearson Correlation and TAN-ELR algorithm in rec-ommender system. In AIP Conference Proceedings (Vol. 1977, No. 1, p. 040028). AIP Publishing.
[5] Kvet, M., & Kvet, M. (2017, July). Temporal database management: Temporal registration. In Information and Digital Technologies (IDT), 2017 International Conference on (pp. 227-233). IEEE.
[6] Orru, M., Paolillo, R., Detti, A., Rossi, G., & Melazzi, N. B. (2017, June). Demonstration of OpenGeoBase: the ICN NoSQL spatio-temporal database. In Local and Metropolitan Area Networks (LANMAN), 2017 IEEE International Symposium on (pp. 1-2). IEEE.
[7] Pant, N., Fouladgar, M., Elmasri, R., & Jitkajornwanich, K. (2018, March). A Survey of Spatio-Temporal Database Re-search. In Asian Conference on Intelligent Information and Database Systems (pp. 115-126). Springer, Cham.
[8] Poscic, P., Babic, I., & Jaksic, D. (2018, May). Temporal functionalities in modern database management systems and data warehouses. In 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). IEEE.
[9] Snodgrass, R. T. (Ed.). (2012). The TSQL2 temporal query language (Vol. 330). Springer Science & Business Media.