DESIGN AND IMPLEMENTATION OF THREE-DIMENSIONAL OBJECTS IN DATABASE MANAGEMENT SYSTEMS

Kazi Shamsul Arefin[1*], Mohammed Salim Bhuyan[2], Ashique Al Rahman[3]

[1,2,3]*American World University, CA, USA (Bangladesh Study Centre)*
[1*]*Research Scholar,* ***ksarefin@yahoo.com, www.ksarefin.info***

Abstract

The idea of this research is to incorporate three-dimensional information into relational database management system and hence implementation in geographical information system and computer aided design. This is because; database management system has inadequate resource to preserve three-dimensional information. On the other hand, there are a lot of potential applications that would be greatly benefited utilizing three-dimensional data. As there is no currently available three-dimensional data type therefore this research leads to identify the three-dimensional objects and develop the data type accordingly.

Keywords: *Database Management System (DBMS); Three-Dimensional (3D); Geographical Information System (GIS); Geo-DBMS; Computer Aided Design (CAD).*

----------***----------

1. INTRODUCTION

The earth is not flat; it is three-dimensional (3D). Where, we continuously work together with three objects that are width, height and depth. The design of 3D database system has to be satisfied with a number of designs criteria like: data type definition, properties of data types and entities. Finally, this research leads to the development of 3D objects in Geo-Database Management Systems (DBMS).

1.1Statement of the Problem

In case of 2D data, we have two dimensions, such as length and width; however, 3D data have an additional dimension, such as height. In DBMS, height information was always overlooked. With this, the height is approximated and included in calculations for these operations. There are some weaknesses for 3D data model in DBMS:

- DBMS does not differentiate 3D objects
- Inconsistency in continuous flat topology characteristic in GIS systems
- Problematic in the execution of spatial analyses in the GIS systems
- Problematic to provide solution for spatial overlap

1.2 Objectives of this Research

Some researchers are working on 3D geometry data types. However, there are still not recognized 3D data types by the spatial extent in DBMS. The objectives of this research are as follows:

1. To offer 3D data type in DBMS;
2. To develop a data structure that can handle with large data and offers support for analyzing, validation and querying;
3. Propose the next generation topology in DBMS for CAD and geo-DBMS.

2. BENEFITS OF 3D-DBMS DEVELOPMENT

There are a lot of advantages utilizing 3D data type in order to preserve 3D information. Now-a-days, some DBMS are trying to preserve 3D information introducing spatial concepts. As mentioned before, depth (Z-coordinate) is ignored or missing in DBMS. Introducing 3D information is a good approach although it is still insufficient. This concept is developed on the basis of geometry models. There are many advantages using 3D DBMS model:

- Avoids redundant storage
- Consistency in data editing
- Easier to maintenance
- Efficient in visualization
- Efficient in query operations
- Natural data model for applications

3. GENERAL DATA TYPE

Each column of DBMS has a name with appropriate data type. Developer decides what type of data will be stored inside the column based on the attribute. In this section, a table: T_CITY_INFO is created with 4 (Four) columns: Rank, City, Population, Latitude Longitude and Altitude for this research that leads to the 3D data type in section: 5.0.

Illustration 3.1: Table T_CITY_INFO without a spatial geometry column.

```
CREATE TABLE T_CITY_INFO
(
    Rank       NUMBER(4),
    City       VARCHAR2(20 byte),
    Population NUMBER(10),
    Latitude   NUMBER(10,7),
Longitude  NUMBER(10,7),
```

```
Altitude  NUMBER(10,7)
);
```

We gathered GIS information for 140 cities in Bangladesh and inserted data (City name, population, Longitude and Latitude) into T_CITY_INFO. After the execution of 'INSERT' statements, 'COMMIT' command is placed in order to save into database permanently then data availability is cross-checked using 'SELECT' statement.

```
SELECT RANK,CITY,POPULATION,LATITUDE,LONGITUDE
from T_CITY_INFO;
```

3.1 Limitations

Now if we would like to preserve information on geographical areas (see Figure 3.1 and 3.2) in the table: T_CITY_INFO then it is not possible in the traditional way. Besides, there is no such data type in oracle database that is able to hold this information. In this case what a traditional DBMS does:

- Additional table will be created to preserve information
- Use referential keys (Foreign key) in order to make join in different tables

Hence, there are some limitations mentioned bellows:

- Consuming more memory (Redundant information)
- Need to join in both table (Primary and Foreign key)
- Perform slower as both not in the same table
- Need to indexing in order to fetching data
- Complexity in maintaining 2 (two) different physical tables
- Complexity in finding related information

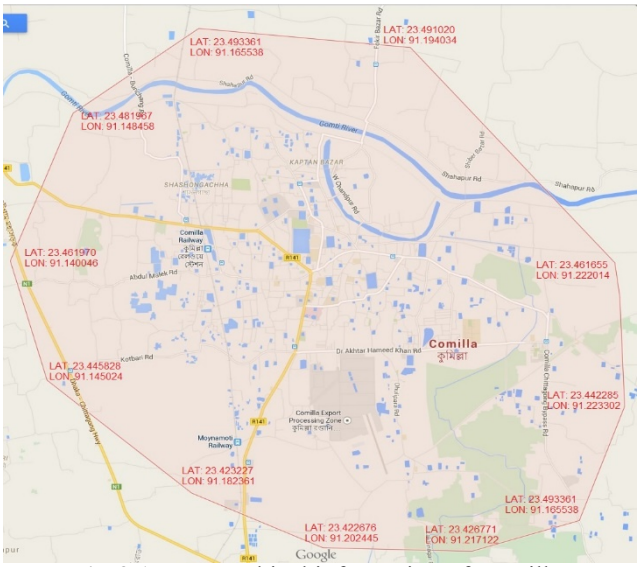

Fig-3.1: Geographical information of Comilla, Bangladesh[1]

Fig-3.2: Geographical information of Mymensingh, Bangladesh[2]

In the section 4.0: Spatial data type is designed to overcome this issue.

4. SPATIAL DATA TYPE DESIGN

Oracle Spatial is incorporated with a set of procedures, functions that supports for 3D data. Spatial data represents the essential location characteristics of real or conceptual objects as those objects relate to the real or conceptual space. Spatial can be stated to:

- Dimension
- Space
- Three-dimensional space

In this section, a new column is added in the table: T_CITY_INFO that can hold geographical information then it is tested well with sufficient input operations. The name of that new column is GEO_INFO. It can hold GIS information in the table T_CITY_INFO.

```
ALTER TABLE T_CITY_INFO ADD (GEO_INFO
SDO_GEOMETRY);


table T_CITY_INFO altered.
```

- Here updating the co-ordinate information as per figure 3.1 for the city of 'Comilla'. There is an object SDO_ORDINATE_ARRAY in SDO_GEOMETRY where longitude and latitude are being updated. This is how GEO_INFO is holding data.

```
UPDATE T_CITY_INFO
SET GEO_INFO =
  SDO_GEOMETRY(
```

[1]https://www.google.com.bd/maps/place/Comilla

[2]https://www.google.com.bd/maps/place/mymensingh

```
    NULL,
    NULL,
    NULL,
    SDO_ELEM_INFO_ARRAY(LONGITUDE, LATITUDE,
NULL),
    SDO_ORDINATE_ARRAY(23.481967,  91.148458,
……..
                        23.461970,   91.140046)
  )
  WHERE RANK=64;

1 rows updated.
```

- Now fetching information for the city of 'Comilla'. Where, GEO_INFO is containing information.

```
SELECT * FROM T_CITY_INFO WHERE RANK=64;

RANK      CITY      POPULATION  LATITUDE
          LONGITUDE   GEO_INF0

--------------------------------------------------

64        Comilla    305700        23.46
          91.17
          MDSYS.SDO_GEOMETRY(NULL,NULL,NULL,MDSYS.S
DO_ELEM_INFO_ARRAY(91.17,23.46,null),MDSYS.SDO_ORD
INATE_ARRAY(23.481967,91.148458,23.493361,91.16553
8,23.49102,91.194034,23.461655,91.222014,23.442285
,91.223302,23.493361,91.165538,23.426771,91.217122
,23.422676,91.202445,23.423227,91.182361,23.445828
,91.145024,23.46197,91.140046))
```

- The same procedure is followed to update information of 'Mymensingh' city and fetching information of that city. The rank of 'Mymensingh' is 85.

```
SELECT * FROM T_CITY_INFO WHERE RANK=85;

RANK      CITY      POPULATION  LATITUDE
          LONGITUDE   GEO_INF0

--------------------------------------------------

85        Mymensingh         244000    24.75
          90.4
          MDSYS.SDO_GEOMETRY(NULL,NULL,NULL,MDSYS.S
DO_ELEM_INFO_ARRAY(90.4,24.75,null),MDSYS.SDO_ORDI
NATE_ARRAY(24.782191,90.342722,24.780866,90.362205
,24.777593,90.370617,24.777533,90.381432,24.770813
,90.395937,24.75694,90.41482,24.739636,90.42778,24
.728644,90.437393,24.719054,90.448465,24.71422,90.
427523,24.715156,90.406752,24.782191,90.342722,24.
728488,90.395851,24.749224,90.370359,24.765358,90.
359116))
```

4.1 Limitations

In the figure 3.1 and 3.2 have only two dimensions. There are 2 (two) axis: horizontal and vertical (X and Y) dimensions in two-dimension (2D). However, 3D has one more dimension that is depth (Z). Now if we would like to keep altitude (Z) dimension on geographical area (figure 4.1) in the same table: T_CITY_INFO then it is not possible in this way. As there is no such type of data type in oracle database that can put this information. We can use the oracle spatial objects in order to do that. However, still there are some dependencies and limitations. Such as:

- Some DBMSs even do not support the spatial concepts
- Inappropriate in data dimensionality and structure

In the section 5.0, a custom made 3D data type is designed based on requirements in order to make life easier.

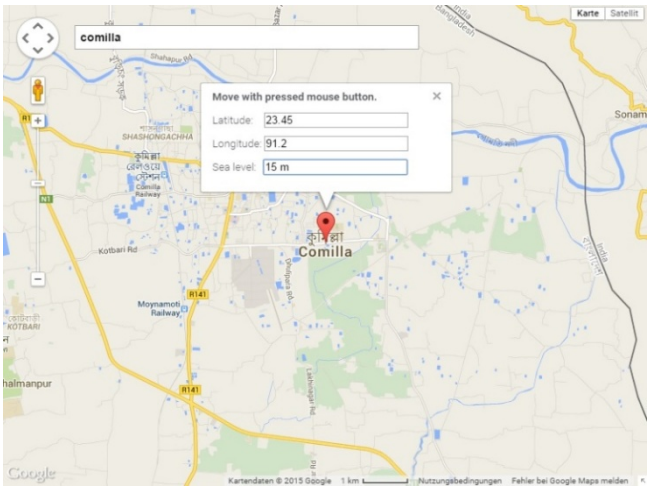

Fig-4.1: 3D (Longitude, Latitude and Altitude) Geographical information of Comilla, Bangladesh

5. THREE-DIMENSIONAL DATA TYPE DESIGN

In the figure 4.1, we are getting 3D (Longitude, Latitude and Altitude) geographical information for a city. Therefore, a new data type "GEO_3D_OBJ" is developed in order to keep 3D information in this section.

- First: an object is created with 3 (three) attributes: Longitude, Latitude and Altitude (properties of these columns are numeric). The name of this object is GEO_3D_ARRAY.

```
CREATE TYPE GEO_3D_ARRAY AS OBJECT (
   LONGITUDE NUMBER,
   LATITUDE NUMBER,
   ALTITUDE NUMBER);
```

- Second: a varray data type object: GEO_3D_DATA_TYPE is created. Where, VARRAY (1048576) is the maximum size. Here, nested (PL/SQL) table can also be used instead of varray.

```
CREATE TYPE GEO_3D_DATA_TYPE AS VARRAY
(1048576) of NUMBER;
```

- Third: a 3D object GEO_3D_OBJ is created where it holds 3 objects (NO_OF_DIM NUMBER,

GEO_3D_DATA_INFO GEO_3D_ARRAY and GEO_3D_DATA GEO_3D_DATA_TYPE). Now this data type (GEO_3D_OBJ) is able to contain 3D information.

```
CREATE TYPE GEO_3D_OBJ AS OBJECT (
 NO_OF_DIM NUMBER,
 GEO_3D_DATA_INFO GEO_3D_ARRAY,
 GEO_3D_DATA GEO_3D_DATA_TYPE);
```

- Finally: adding 3D column GEO_3D_COL into T_CITY_INFO which is an object of GEO_3D_OBJ

```
ALTER TABLE T_CITY_INFO ADD GEO_3D_COL
GEO_3D_OBJ;

table T_CITY_INFO altered.
```

- As 3D column is added, now updating information into the newly added GEO_3D_COL column in T_CITY_INFO with 3D information.

```
UPDATE T_CITY_INFO
SET GEO_3D_COL =
  GEO_3D_OBJ(
    3,  --Number of Dimension
    GEO_3D_ARRAY(LONGITUDE, LATITUDE,
ALTITUDE),
    GEO_3D_DATA_TYPE(
        24.782191, 90.342722, 21,
……
        24.765358, 90.359116, 20
    )
  )
  WHERE RANK=85;

1 rows updated.
```

- After adding 3D data, checking from T_CITY_INFO for Rank = 85

```
SELECT RANK, CITY, POPULATION, LATITUDE,
      LONGITUDE, GEO_3D_COL from T_CITY_INFO
WHERE RANK=85;

RANK  CITY      POPULATION  LATITUDE
      LONGITUDE   GEO_3D_COL
-----------------------------------------------
85    Mymensingh244000   24.75     90.4
      GEO_3D_OBJ(3,GEO_3D_ARRAY(90.4,24.75,NULL
),GEO_3D_DATA_TYPE(24.782191,90.342722,21,24.78
0866,90.362205,19,24.777593,90.370617,17,24.777
533,90.381432,18,24.770813,90.395937,15,24.7569
4,90.41482,12,24.739636,90.42778,19,24.728644,9
0.437393,20,24.719054,90.448465,14,24.71422,90.
427523,16,24.715156,90.406752,20,24.782191,90.3
42722,15,24.728488,90.395851,17,24.749224,90.37
0359,19,24.765358,90.359116,20))
```

6.PERFORMANCE ANALYSIS

In this section, we analyzed the performance for traditional way and our newly developed 3D data type. Here, we used 'EXPLAIN PLAN' for both of cases and found huge performance improvement.

6.1 Traditional Datatype

In order to the performance testing, we created 2 (two) additional tables (T_CITY_INFO_TRADITIONAL and GEO_3D_DATA) and did the "EXPLAIN_PLAN" and found the following statistics:

```
EXPLAIN PLAN FOR SELECT T_CITY_INFO_TRADITIONAL.*
FROM T_CITY_INFO_TRADITIONAL INNER JOIN GEO_3D_DATA
ON GEO_3D_DATA.RANK=T_CITY_INFO_TRADITIONAL.RANK
WHERE T_CITY_INFO_TRADITIONAL.RANK=85;

SELECT PLAN_TABLE_OUTPUT FROM
TABLE(DBMS_XPLAN.DISPLAY());
-----------------------------------------------------
Plan hash value: 3863888095

-----------------------------------------------------
| Id  | Operation          | Name
| Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------
|   0 | SELECT STATEMENT   |
|    15 |  1350 |     7  (15)| 00:00:01 |
|*  1 |  HASH JOIN         |
|    15 |  1350 |     7  (15)| 00:00:01 |
|*  2 |   TABLE ACCESS FULL| T_CITY_INFO_TRADITIONAL
|     1 |    77 |     3   (0)| 00:00:01 |
|*  3 |   TABLE ACCESS FULL| GEO_3D_DATA
|    15 |   195 |     3   (0)| 00:00:01 |
-----------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------


   1 -
access("GEO_3D_DATA"."RANK"="T_CITY_INFO_TRADITIONAL"
."RANK")
   2 - filter("T_CITY_INFO_TRADITIONAL"."RANK"=85)
   3 - filter("GEO_3D_DATA"."RANK"=85)

Note
-----
   - dynamic sampling used for this statement
(level=2)
```

6.2 Developed 3D Datatype

In section 5, we developed a 3D data type. Now doing the "EXPLAIN_PLAN" and found the bellow statistics:

```
EXPLAIN PLAN FOR SELECT * FROM T_CITY_INFO WHERE
RANK=85;
SELECT PLAN_TABLE_OUTPUT FROM
TABLE(DBMS_XPLAN.DISPLAY());
-------------------------------------------------

Plan hash value: 2045439915

-------------------------------------------------
| Id  | Operation         | Name        | Rows  |
Bytes | Cost (%CPU)| Time     |
-------------------------------------------------
|   0 | SELECT STATEMENT  |             |     1 |
27 |     3   (0)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| T_CITY_INFO |     1 |
27 |     3   (0)| 00:00:01 |
-------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter("RANK"=85)
```

6.3 Comparisons

As per illustration on section 6.1 and 6.2, it is clear the improved performance of our developed 3D data type. We just added our newly developed a 3D column instead of creating a different table what the traditional way followed and got the dramatically performance improvement in terms of Rows, Bytes, Cost (%CPU), Times (Seconds).

Table-6.1: Performance analysis of Traditional and newly developed data type

	Data type		
Category	Traditional	3D (Developed)	Improvements (%)
Rows	15	1	93%
Bytes	1,350	27	98%
Cost (%CPU)	7	3	57%
Time (Seconds)	1	1	0%

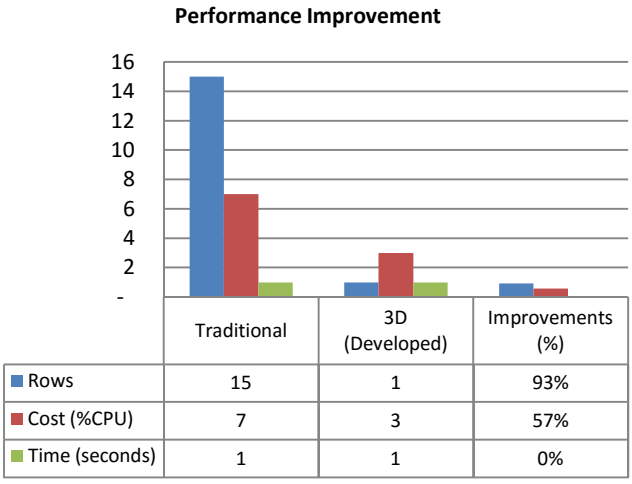

Fig-6.1: Performance analysis (Rows, Cost) for traditional and developed 3D data type

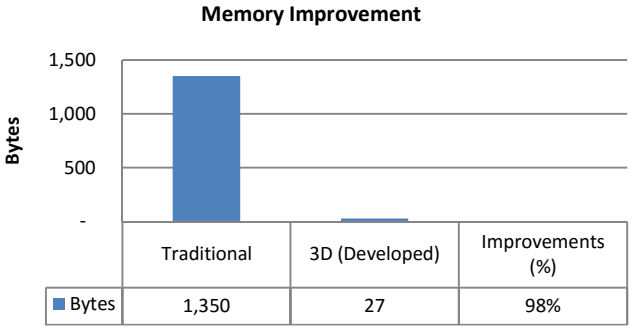

Fig-6.2: Memory analysis (Bytes) for traditional and developed 3D data type

7. CONCLUSION AND FUTURE RESEARCH

In this paper, we presented a clear research evidence of development 3D data type in DBMS. Where, this approach is implemented and contributed to the rising 3D data type for all DBMSs. We can get all information in a single table rather doing multiple pieces and joining from multiple tables; just adding a custom made column able to query. We presented how to develop a custom made data type and also showed how 3D data can be inserted without using existing spatial data type. Hence, it speeds up the process, reduce memory and also easy to manage data.

To conclude, our concept is developed, well tested and also provided promising outcomes in Oracle 11g database environment. Future research is to develop 4D data type for Computer Aided Design (CAD) systems using PL/SQL tables.

ACKNOWLEDGEMENT

This is a unique and interesting research work where it mainly focused geographical area of Bangladesh as this research is done in this country. We would like to thanks google map and Oracle database.

REFERENCES

[1] Foley, J., van Dam, A., Feiner, S., and Hughes, J. 1995. Computer graphics: principles and practice. Addison Wesley, 2nd Ed. Geodata Infrastructure North-Rhine Westphalia (GDI NRW).

[2] Arens, C. A. 2003. Modeling 3D spatial objects in a geo-DBMS using a 3D primitives. Msc thesis, TU Delft, The Netherlands. 76 p.

[3] Arens, C., Stoter, J.E., and van Oosterom, P.J.M. 2005. Modeling 3D spatial objects in a geo-DBMS using a 3D primitive. In Computers & Geosciences, volume 31, 2. pp. 165-177.

[4] CalinArens et al., Modeling 3D spatial objects in a geo-DBMS using a 3D primitive, Computers & Geosciences, Vol.31, pp. 165-177, 2005.

[5] T.K Chen et al., 3D Spatial Operations for geo-DBMS: Geometry vs. Topology, International Archives of the Photogrammetry and Spatial Information Sciences, Vol.37, pp.549-554, 2008.

[6] JantienStoter et al., Visualization and Editing of 3D Objects Organized in a DBMS, Geo-Database Management Center, 2003.

[7] JantienStoter et al., 3D Modeling with National Coverage: Bridging the Gap, 3D Products at NMCAs, Open Issues.

[8] Prof. Abbas Rajabifard et al., Advance Principles of 3D Cadastral Data Modeling, 2nd International Workshop on 3D Cadastres, 2011.

[9] CalinArens et al., Modeling 3D objects in a geo-DBMS using a 3D primitive, 6th AGILE, France, 2003.

[10] Narayan, K. Lalit (2008). Computer Aided Design and Manufacturing. New Delhi: Prentice Hall of India. p. 4. ISBN 812033342X.

[11] Madsen, David A. (2012). Engineering Drawing & Design. Clifton Park, NY: Delmar. p. 10. ISBN 1111309574.

[12] http://en.wikipedia.org/wiki/Spatial

[13] http://en.wikipedia.org/wiki/Three-dimensional_space

[14] http://en.wikipedia.org/wiki/Data_type

BIOGRAPHIES

Kazi Shamsul Arefin is engaged in research activities throughout his undergraduate years and has published more than 22 international research papers. Moreover, he is a member of International Association of Computer Science and Information Technology (IACSIT), Membership No: 80337708.