IOSR Journal of Computer Engineering (IOSRJCE)
ISSN : 2278-0661 Volume 1, Issue 3 (May-June 2012), PP 46-50
www.iosrjournals.org

An Algorithm to Simulate the Temporal Database Using Exponential Distribution

Shweta Sharma[1], Ekta Saini[2]
[1, 2](Computer Science Department, Doon Valley Institute of Engineering & Technology, Karnal, Haryana, India)

Abstract :
Since the early eighties, an active temporal database research community has sought new insight into the management of time-referenced, or temporal, data and has developed concepts, tools and techniques that better support the management of such data. Much of this activity has been motivated by the observations that most databases contain substantial amounts of temporal data and that conventional database technology offers precious little support for temporal data management.

Temporal data stored in a temporal database is different from the data stored in non-temporal database in that a time period attached to the data expresses when it was valid or stored in the database. In this paper we are going to design an algorithm for simulating temporal database using Exponential Distribution.
Keywords *- Exponential Distribution, Interval based Temporal Database Model, Temporal Database, Transaction Time, Valid Time.*

I. Introduction

A first step towards a temporal database thus is to timestamp the data. This allows the distinction of different database states. One approach is that a temporal database may timestamp entities with time periods. Another approach is the time stamping of the property values of the entities. In the relational data model, tuples are time stamped, where as in object-oriented data models, objects and/or attribute values may be time stamped. What time period do we store in these timestamps? As we mentioned already, there are mainly two different notions of time which are relevant for temporal databases. One is called the **valid time**, the other one is the **transaction time** [3]. Valid time denotes the time period during which a fact is true with respect to the real world. Transaction time is the time period during which a fact is stored in the database.

A temporal database is a database with built-in time aspects. Since the early eighties, an active temporal database research community has sought new insight into the management of time-referenced, or temporal, data and has developed concepts, tools and techniques that better support the management of such data. Much of this activity has been motivated by the observations that most databases contain substantial amounts of temporal data and that conventional database technology offers precious little support for temporal data management.

A wide range of database applications manage time-varying data. In contrast, existing database technology provides little support for managing such data. The research area of temporal databases aims to change this state of affairs by characterizing the semantics of temporal data and providing expressive and efficient ways to model, store, and query temporal data.

Valid time is the time for which a fact is true in the real world. In the example above, the Person table gets two extra fields, Valid-From and Valid-To, specifying when a person's address was valid in the real world.

Transaction time is the time a transaction was made. This enables queries that show the state of the database at a given time

II. Interval-based temporal data model

Interval-based temporal data model is a popular data model in temporal data- bases. It uses time intervals for representing the period of validity of a tuple, leading to unavoidable self-joins when combining tuples for objects. An interval-based data model t uses time-intervals and tuple-level time stamping, which is the most popular approach for interval-based data models [1]. It is called bitemporal data model if the data model manages transaction time and valid time of tuples . here we consider one dimensional temporal data model. In the interval-based temporal data model [2], in addition to usual attributes, Start and End attributes are used to specify the period of validity for the information in the tuple.

III. Exponential Distribution Function

In probability theory and statistics, the exponential distributions are a class of continuous probability distributions. They describe the times between events in a Poisson process, i.e. a process in which events occur continuously and independently at a constant average rate. Suppose that in order to perform a simulation experiment on some stochastic system governed by an exponential distribution function we require n random sample t1,t2…..tn. the inverse of cumulative distribution function F(t) can be obtained by equating u, a generated uniform random number :

$$u = 1-e^{-\lambda t} \qquad (1)$$

so that, $-\lambda t = \log_e(1-u)$ (2)

which gives the corresponding sample t as :

$$t = (-\log_e(1-u))/\lambda \qquad (3)$$

Since (1-u) is as good a uniformly distributed random number between 0 and 1 as u is, we can replace (1-u) with u itself. That is $t_{k} = (-1/\lambda)\log_e u_k$

We obtain the required n sample t1,t2…..tn from exponential distribution $1-e^{-\lambda t}$ by transforming n uniform random numbers u1,u2,….un in interval(0,1).

IV. imulating the temporal timings by exponential distribution

For simulating the temporal timings, that is valid and transaction times by exponential distribution, suppose a case of an employee whose valid time start and valid time end are given. And using the random sample from exponential distribution we can calculate the the transaction time start and transaction time end of the employee as described in following proposed algorithm

4.1 Terms and Notations used

Term	**Meaning**
valids	Valid time start
valide	Valid time end
trs	Transaction time start
tre	Transaction time end
x	Random number in the range (0,1)
lamda	Positive constant, no. of occurrences

4.2 Proposed Work : Algorithm for generation of temporal timings by exponential distribution

```
1.  Procedure exponential_times()
    [Generation of first random sample]
2.    for i ← 1 to 4 do
3.        seed ← (seed + 3.14159) *
                 (seed + 3.14159)
4.        temp ← seed
5.        seed ← seed – temp
6.        x ← seed
7.    end for
    [Generation of times duration by exponential distribution]
8.    for i ← 1 to 4 do
9.        p[i]= -(1/lemda)*log(x)
10.   end for
      [Transaction time start]
11.   for i ← 1 to 4 do
12.       valids=(valids+100*p[i])
13.       trs[i] ← valids
14.   end for
      [Transaction time end]
```

```
15.     for i  ← 1 to 4 do
16.         valids=(valids+100*p[i])
17.         tre[i] ← valids
18.     end for
       [Number of fields updated]
19.     for i  ← 1 to 4 do
20.         valids=(valids+100*p[i])
21.         for i  ← 1 to 20 do
22.             if[i=valids]
23.                 print number of fields updated during the
                    transaction.
24.             end if
25.         end for
26.     end for
27.   end procedure
```

V. Performance Evolution

Below Table shows Employee temporal relation that maintains the history of employees with name, salary, and department information. Every tuple has Start and End attributes indicating the period of validity of the tuple. In Employee relation, an object consists of multiple tuples, where each tuple represents an event.

Table 1: Employee Relation

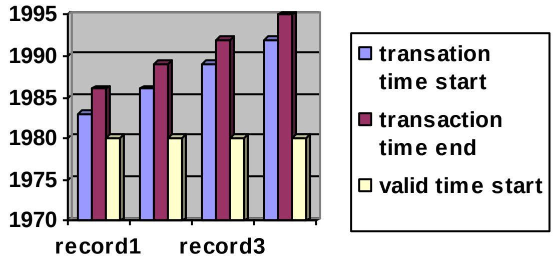
Name	Salary	DName	Start	End
Ram	4500	Commerce	0	15
Ram	8000	Computers	32	45
Ram	8000	IT	46	60
Ram	6000	Electronics	61	70
Mohan	8000	IT	11	56
Mohan	5500	Electronics	57	60
Mohan	6000	Electronics	61	75

. For example, two tuples, (Ram, 8000, Computers, [32,45]) and (Ram, 8000, IT, [46,60]), are distinct events and the latter tuple was created when Tom moved to IT department from Computers department.

Therefore, there exists no tuples whose time intervals overlap.

Table 2: Output table for Employee

	Valid time start	Transaction time start	Transaction time end	No. of fields updated
1st Transaction	1980	1983	1986	3
2nd Transaction	1980	1986	1989	4
3rd Transaction	1980	1989	1992	5
4th Transaction	1980	1992	1995	6



6.2

Fig.1 Graph for valid time start and transaction times by exponential distribution

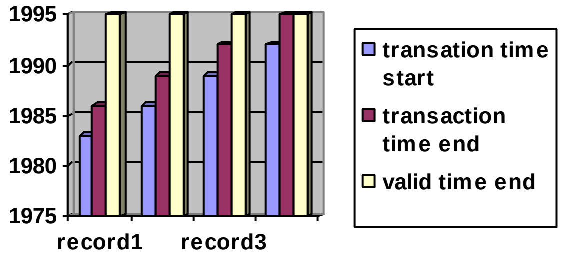


Fig.2 Graph for valid time end and transaction times by exponential distribution

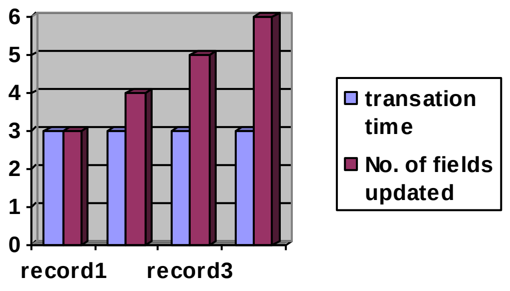


Fig.3 Graph for number of fields updated during transaction

VI. Conclusion

Join operations are one of the most expensive operations in conventional databases and they pose even more serious problems in temporal databases because temporal databases are accumulating histories and hence larger.

In the interval-based temporal data model, self-joins are unavoidable for combining tuples for an object. It requires (k + 1)-way self-join[5] for k conjunctive conditions. Interval-based temporal data model is a popular data model in temporal data- bases. an object is modeled in multiple tuples in a relation because an event is validated with an interval.

One advantage of this data model is its fast implementation on top of conventional relational databases, availing well-developed optimizer. We have discussed a join algorithm without index structures whose complexity is from a polynomial to a linear.The algorithm need multiple scans for processing k-way self-join.

Here simulation is used in block nested loop join algorithm in interval based temporal data model by using Exponential distribution to calculate the valid time and transection time of a tuple in relation.

References

[1] Christian S. Jensen, Richard T, and Michael D. Soo. Extending Existing Dependency Theory to Temporal Databases. *IEEE Transactions on Knowledge and Data Engineering*, August 1996.

[2] Claudio Bettini, Member, IEEE,X.Sean Wang, Member, IEEE Computer Society,and Sushil Jajod Senior Member, IEEE. Temporal Semantic Assumptions and Their Use in Databases. *IEEE transactions on knowledge and data engineering,* March/April 1998.

[3] C.S. Jensen, J. Clifford, S.K.Gadia, A.Segev, R.T.Snotgrass. *A Glossary of Database Concepts. Sigmod Record*, September 1992.

[4] Mohamad H. Saraee and Babis Theodoulidis. *Knowledge Discovery in Temporal Databases.* Department of Computation, at UMIST Manchester, UK, 1995.

[5] Seo-Young Noh and Shashi K. Gadia. *Efficient Self- Join algo in Interval- based Temporal Data Models*. Department of Computer Science, Iowa State University, Ames, Iowa, USA, Sept 2005.