Prof. S.R. Thakare, Dr. C.A. Dhawale, Prof. A.B. Gadicha / International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com
Vol. 2, Issue 2,Mar-Apr 2012, pp.1337-1348

Transaction Control System On Various Techniques for Maintaining Consistency of Databases

Prof. S.R. Thakare [1] Dr. C.A. Dhawale [2] Prof. A.B. Gadicha [3]
Department of MCA , P.R.Pote(Patil) College of Engg &Magmt Amravati [1,2]
Department of Computer Science & Engg , P.R.Pote(Patil) College of Engg &Magmt Amravati [3]

Abstract:- As Per Requirement of Application Transaction Perform on Database.These are Transaction must maintenance Consistency .This Paper explain Various Technique of Transaction for Maintenance Consistency.

Introduction:

The transaction as used in most data management systems generalizes to the network environment, that network systems should provide of a transaction as an abstraction which

Eases the construction of programs in a distributed system. Transactions would provide the programmer with the following types of transparencies.

(1) Location Transparency. Although data are geographically distributed and may move from place to place, the programmer can act as if all the data were in one node.

(2) Replication Transparency. Although the same data item may be replicated at several nodes of the network, the programmer may treat the item as if it were stored as a single item at a single node.

(3) Concurrency Transparency. Although the system runs many transactions
concurrently, to each transaction it appears as if it were the only activity in the system. Alternatively, it appears as if there were no concurrency in the system.

(4) Failure Transparency. Either all the actions of a transaction occur or none of them occur. Once a transaction occurs, its effects survive hardware and software failures.

They are Various Techniques of Transaction for Maintenance Consistency as given below

1 Semantic Data Control in Database Environment

A centralized & distributed database is a database distributed between several sites. The reasons for the data distribution may include the inherent distributed nature of the data or performance reasons. In a distributed database the data at each site is not necessarily an independent entity, but can be rather related to the data stored on the other sites. This relationship together with the integrity assertions on the data are expressed in the form of data constraints. **Figure** 1 shows a classification of the possible data constraints that can hold on a distributed database. Those constraints as an invariant, which must hold at any given time instance (static constraints) or during any time interval of prespecified length (dynamic constraints).

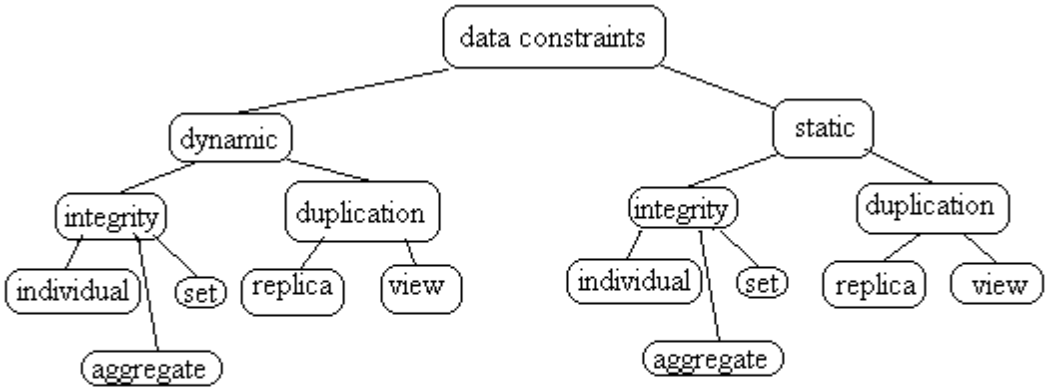

Figure 1 Distributed Database data constraints classification

At the top level there are two type of constraints dynamic and static. To check whether a *static constraint* holds, at any given time point, we need a static snapshot of the database instance. On the other hand, to verify whether a *dynamic constraint* holds we need to have information about how the database instance evolves over time. An example of a dynamic constraint is: within every 5 minutes of the life of the database instance something must happen, e.g. an update must occur. On an orthogonal plane, data constraints can be integrity or duplication. *Integrity* constraints are independent of how the data is distributed or duplicated If an integrity constraint is based on a single collection of data objects and a single variable it is called *individual*. *Set constraints* are those that are based on more than one data collections or on more than one variables. For example, in the distributed database shown in Figure 2 ,
individual integrity constraints may include the primary key constraints on the tables *R1* and *R2*, whiles a set integrity constraint may include a referential foreign key constraint between the two tables. *Aggregate constraints* are integrity constraints involving aggregate operators such as min, max, sum, count and average.

Note, that integrity constraints can be both dynamic and static. In particular a *dynamic integrity constraint* may state that if an update, which violates a constraint, is performed, a compensating update which restores the constraint should be performed within two minutes. On the other hand, static integrity constraints require the integrity constraint to be true at any given time instance.

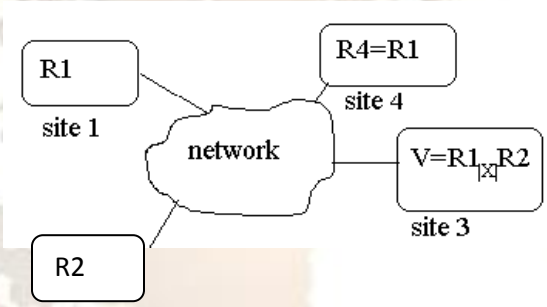

Figure 2 A Distributed database Example

2 Real-Time Replication Control for Database System on Basis Algorithms

In real-time distributed database systems, timeliness of results can be as important as their correctness The problems related to replication control such as the preservation of mutual and internal consistency become more difficult when timing constraints are imposed on transactions. Transactions must be scheduled to meet the timing constraints and to ensure data consistency. Real-time task scheduling can be used to enforce timing constraints on transactions, while concurrency control is employed to maintain data consistency. The integration of the two mechanisms is non trivial because of the trade-offs involved. Serializability may be too strong as a correctness criterion for concurrency control in database systems with timing constraints, for serializability severely limits concurrency. As a consequence, data consistency might be compromised to satisfy timing constraints.

Epsilon-serializability (ESR) is a correctness criterion that enables asynchronous maintenance of mutual consistency of replicated data . A transaction with ESR as its correctness criterion is called an *epsilontransaction* (ET). An ET is a query ET if it consists of only reads. An ET containing at least one write is an update ET. Query ETs may see an inconsistent data state produced by update ETs. The metric to control the level of inconsistency a query may return is called the *overlap*. It is defined as the set of all update ETs that are active and affecting data objects that the query seeks to access. If a query ET's overlap is empty, then the query is serializable. The overlap of an active query transaction *Q* can be used as an *upper bound of error* on the degree of inconsistency that may accumulate.

Among several replica control methods based on ESR, chosen the ordered updates approach. The ordered updates approach allows more concurrency than 1SR in two ways. First, query ETs can be processed in any order because they are allowed to see intermediate, inconsistent results. Second, update ETs may update different replicas of the same object asynchronously,but in the same order. In this way, update ETs produce results equivalent to a serial schedule; these results are therefore consistent. There are two categories of transaction conflicts that examine: conflicts between update transactions and conflicts between update and query transactions. *Conflicts between update transactions* can be either *RW* conflicts or*WW* conflicts. Both types must be strictly resolved. No correctness criteria can be relaxed here, since execution of update transactions must remain 1SR in order for replicas of data objects to remain identical. *Conflicts between update and query transactions* are of *RW* type. Each time a query conflicts with an update, say that the query overlaps with this update, and the overlap counter is incremented by one. If the counter is still less than a specified upper bound, then both operation requests are processed normally, the conflict is ignored, and no transaction is aborted. Otherwise, RW conflict must be resolved by using the conventional 1SR correctness criteria of the accommodating algorithm. The performance gains of the above conflict resolution policies are numerous. Update transactions are rarely blocked or aborted in favor of query transactions. They may be delayed on behalf of other update transactions in order to preserve internal database consistency. On the other hand, query transactions are almost never blocked provided that their overlap upper bound is not exceeded. Finally, update transactions attain the flexibility to write replicas in an asynchronous manner.

A real-time replication control algorithm, based on the majority consensus approach. he use of ESR correctness criteria to control the inconsistency of query transactions and the mechanism for conflict resolution of real-time transactions.

3.Transaction Management using Coordinator

Oracle constructs a session tree for the participating nodes. The session tree describes the relations between the nodes participating in any given transaction. Each node plays one or more of the following roles:

1. **Client**: A client is a node that references data from another node.
2. **Database Server**: A server is a node that is being referenced by another node because it
has needed data. A database server is a server that supports a local database.
3. **Global Coordinator**: The global coordinator is the node that initiated the transaction, and thus, is the root of the session tree. The operations performed by the global coordinator are as follows:

• In its role as a global coordinator and the root of the session tree, all the SQL statements,
procedure calls, etc., are sent to the referenced nodes by the global coordinator.
Instructs all the nodes, except the COMMIT point site, to PREPARE
• If all sites PREPARE successfully, then the global coordinator instructs the COMMIT
point site to initiate the commit phase
• If one or more of the nodes send an abort message, then the global coordinator instructs
all nodes to perform a rollback.

4. **Local Coordinator**: A local coordinator is a node that must reference data on another node in order to complete its part. The local coordinator carries out the following functions (Oracle8):

• Receiving and relaying status information among the local nodes
• Passing queries to those nodes
• Receiving queries from those nodes and passing them on to other nodes
• Returning the results of the queries to the nodes that initiated them.

5. **Commit Point Site**: Before a COMMIT point
site can be designated, **the COMMIT point strength** of each node must be determined. The COMMIT point strength of each node of the distributed database system is defined when the initial connection is made between the nodes. The COMMIT point site has to be a reliable node because it has to take care of all the messages. When the global coordinator initiates a transaction, it checks the direct references to see which one is going to act as a COMMIT point site. The COMMIT point site cannot be a read-only site. If multiple nodes
have the same COMMIT point strength, then the global coordinator selects one of them. In case of a rollback, the PREPARE and COMMIT phases are not needed and thus a COMMIT point site is not selected. A transaction is considered to be committed once the COMMIT point site commits locally.

4 Transaction Management using Schema

The primitives issued by a distributed transaction are begin-transaction, read, write, execute, and commit or abort. They are processed according to the schemes outlined below. The algorithms As below:

Begin-transaction: Upon invocation of this operation, the global transaction manager (GTM) writes a begin-transaction record in the log, interprets the request, and decides which nodes are going to be accessed to complete the transaction. Next, it initializes a run-time global table for the transaction. This table has a column for each anticipating node. The header of the column is

the ifentifier of the called node. Then, it issues a local- begin-transaction primitive to all participating nodes.

The local transaction managers (LTMs) of these nodes respond by writing a local-begin-transaction record in the log and creating and initializing new columns in their run-time local tables. Each of the headers of these columns contains the global transaction number and the identifier of the calling node.

Read(Xva1ue): The GTM looks in its run-time global tables to find the table of the involved transaction. Next, It looks for the column of the participating node that contains Xvalue. If it finds a value for Xvalue then it returns it to the calling transaction. If not, it issues a localLread(

Xva1ue) primitive to the appropriate node. The LTM of the latter, in turn, issues a disk_read(Xvalue) primitive to the data mana er of its data storage files. It

gets the requested value, adds it as an entry in the corresponding column of the calling global transaction, and sends it to the GTM. The latter adds the information as an entry in the appropriate column of its table and returns the value to the transaction.

5. Database Transactions on basis Integrity Constrain

For the embedding language assume a programming language. It includes most of the usual constructs, arithmetic expressions, assignment statements, jumps, conditional-, repeating- and compound- statements. In addition it is necessary to describe the database operations. Each database application has a data model **as** a basis. We will restrict our discussion to the relational model and SQL **as** database language. The interesting operations are INSERT, DELETE, UPDATE and SELECT, which we currently restrict to the untested versions. However, we admit arithmetic expressions, comparison operators, logical connectors and aggregate functions (count, sum, avg, min, max). These constructs are sufficient for most of the transaction programs. consistency constraints are much more powerful than those mentioned in the SQL standard proposal. They include the same constructs **as** the predicates in the SQL statements. In addition the existential and the universal quantifier are possible.

The calculus used for the integrity constraints is an extension of predicate calculus extended by some formulas of a nested predicate calculus. This is necessary for constraints including aggregate functions. For example, integrity constraints corresponding to multiple tuples of more than one relation often have a form like: This says that for each element tl of relation rl there exist at least "in' elements and at most "ax' elements in r2 related to tl. Note that this is a simple but important generalization of the 'referential integrity' of **SQL** by specifying minimal and maximal cardinalities.The operator ***card*** is used to describe the number of elements of a set. It has the same meaning **as** the aggregate function count.

The basic method used for analyzing the transaction program is **symbolic execution.** Since we analyze at compile time, the actual values of the variables are not known. Therefore we execute the program with symbolic values assigned to the input variables. They end with the letter **'s'** in the following examples. The output variables are computations over the input variables.

To execute a program at the symbolic level it **is** represented by a control flow graph. A path in a control flow graph is a sequence of basic blocks, such that there exists a possible connection between adjacent nodes. The output of the symbolic execution of **a** path **is a** set **of** output variables consisting **of** computations over the input variables and a conjunction of the path predicates resulting from the branches of the control flow graph.

0 Loops:

The number of iterations often depend on input data. Therefore, determining all paths is not possible. One solution is to derive a closed form **of** the body of the loop, which is not always possible. Another approach is to execute the body K times, where K is determined by the user **or** by the system.

Arrays:

The index of an array is often not known at execution time. This makes it impossible to decide, whether predicates containing different indices like 'a[i]=ab]' are the same.

0 Procedure Calls:

There are two methods to handle procedure calls; the first one is the inline substitution. Using this approach, all procedure calls are replaced by the program code. This method significantly increases .

6 Database Transactions on basis Object

The schemas of three databases: DB1, DB2, and DB3, which are located in different sites and used to store the personal information at the same school. The global schema is constructed by integrating these three component schemas. The schemas only include the class composition hierarchies without the class hierarchy. The missing data mainly come from the existence of missing attributes for the constituent classes, The missing attributes for a class can be divided into *primitive missing attributes* and *complex missing attributes* according to their types of being primitive or complex attributes. **A** predicate which iinv'olves a nested attribute of the range class is called a *nested predicate.* The nested attributes are represented by path expressions in a query. consider the queries containing one range class, whose predicates contain nested predicates combined in conjunctive form. The range class is the *root class* of the composition hierarchy involved in the predicates; the other classes specified in the hierarchy are named *branch classes* for the query. The constituent classes of a root class and a branch class are called *local root classes* and *local branch classes,* respectively.

The following three phases **are** necessary for processing the queries involving missing data.

0 *phaseO*: this phase examines the **GOid** mapping tables to find the isomeric objects which can provide
the missing data. In other words, this is the step for looking up the assistant objects. For the localized approach, the task for checking the assistant objects is also included in this phase.

0 *phase* **I:** this phalse integrates the information of one object and its isomeric objects. For the localized
approach, it means the step for certifying the local maybe resullt~ia nd the unsolved items.

0 *phase* **P:** this phatse does the predicate evaluation. It is possible that, the concerned predicates are local
predicates when the localized approach is applied.

the number of executable paths. The second possibility is to check the procedures separately and to replace all procedure calls by their path conditions.

0 Selection of Paths:

Often it is impossible to execute all paths, because of the complex structure of the program. Restricting the number of paths checked, on the other hand, reduces the quality of the result

phase **I** has to be executed after phase 0. the basic processing algorithms are provided by analyzing the combination of these three phases.

Centralized Approach (CA)

The centralized follows the 0 →**I** → **P** order for executing the three necessary phases. The procedures executed in the
global processing site and each component database are as follows

global processing site

Step CA-G1: Send the request to component databases for retrieving the objects in the local root classes and local branch classes, then wait €or the response.

Step CA-G2: To materialize each global class involved in the query, outerjoin is used over join attribute **GOid** to integrate the objects in the constituent classes (phase 0 and phase **I).**

Step CA-G3: Evaluate the predicates on the materialized global classes (phase **P).**

component database

Step CA-C1: When receiving the request from the global processing site, retrieve and send out all objects in the local root class and local branch classes of the query.

Basic Localized Approach (BL)

The basic localized approach for explaining the concept of the localized approach. The execution order for the
necessary phases **is P** → 0→ I. The tasks of the global processing site and the component databases are described
as follows

global processing site

Step BL-G1: For each component database containing the local root class, produce a local query against the local root class. The predicates remain unchanged at this step. Then send the local queries to the component databases and wait for
the response.

Step BL-G2: According to the certification rule, certify the local results using the results of checking the assistant objects (phase **I).** Get he final certain results and maybe results.

component database

Case 1: receive the local query sent from the global processing site

Step BL-C1: Only evaluate the local predicates in the local query if the missing data is involved in the original predicates (phase **P).**

Step BL-C2: Examine **GOid** mapping tables to find the **LOids** of the assistant objects for the unsolved items of the local maybe results. The **LOids** of the assistant objects and the corresponding unsolved predicates are sent to the other associated component databases for further checking (part of phase ***0).***T he local results are then sent back to the global processing site.

Case 2: receive the request from other component databases for checking the assistant objects.

Step BL-C3: Retrieve the objects for the **LOid** list of the assistant objects and evaluate the appended unsolved predicates. The **LOids** of the satisfied objects are sent back to the global processing site (phase 0)

.

Parallel Localized Approach (PL)

The execution order for the parallel localized approach is 0 → **P** → **I.** This approach makes the tasks for checking the assistant objects and evaluating local predicates to be executed in parallel in different component databases.

global processing site

Step PL-G1: This step is the same as step **BL-G1.**

Step PL-G2: This step is the same **as** step **BL-GZ.**

component database

Case 1: receive the local query sent from the global processing site

Step PL-C1: For any object o in the local root class, retrieve the objects in the nested complex attributes of ***0,*** which have associated missing attributes. There exists at least one unsolved predicate for these retrieved objects. Check **GOid** mapping tables to find the **LOids** of the assistant objects for these objects. Send the **LOids** of the assistant objects and the corresponding unsolved predicates to the associated component databases (part of phase 0)

The difference between the parallel localized approach and the basic localized approach is the order for executing

7. Database Transactions on basis Prewrite operation

A prewrite operation before **an** actual write operation is executed on design objects to increase the potential concurrency. A prewrite operation makes available the model cf the design that the object will have after the design is finally produced. A prewrite operation does not make the design but only provides the model or the picture of the design (including its dimensions, colour combination etc.) a transaction intends to make in future. Once the prewrite design of a transaction is announced, the associated transaction executes a precommit operation. After the transaction has finally produced the design (for which the prewrite design has been announced), it commits. A read transaction can read the prewrite design before the precommitted transaction has produced that design. The prewrite design is made available for reads after the associated transaction has executed a precommit but before it has been committed. Hence, prewrites increase concurrency **as** compared to the environment where only read and write operations **are** allowed on design objects.

Once a transaction has announced a precommit, it is not allowed to abort. This is due to the following reasons. First, it will help in avoiding cascading aborts since the prewrite design has been made available before the transaction **has** finally produced the design and committed. Second, it is desirable for a longduration transaction so that it does not lose all the design work at finishing stage in case there is an abort or a system failure. To accomplish this, a precommit operation is executed only after all the prewrite log records are stored on stable storage. Once write operations start, each write **log** is **also stored on stable storage. Thus, in** case of a failure after pre-commit, there is no need of executing rollback (undo) operations. The recovery algorithm has to at the most redo those operations (using the prewrite logs and the write logs) whose ***effects*** are not there on stable storage. The failed transaction then can restart from the state **as** exists at the time cf failure . If a transaction aborts before executing all prewrite operations and a precommit, it is rolled back by discarding all the announced prewrites In engineering design applications **,** by introducing prewrite designs, shortduration transactions can access the model or picture or the working copy of the design held by a long transaction.

Reads are allowed to access the sketches once they **are** prewritten but before they are actually made.
Therefore, using prewrites, one can have a system consisting cf short and long transactions without causing delay for short-duration transactions. Thus, prewrites help in increasing the throughput of the system by making the response of the system faster for read-only operations. For read-only
transactions, the picture or the model of the design to be produced is important rather then the finally completed design. In the production, the dimension or certain colour combinations of the design may vary from the prewrite version of the design, however, the **read**only transactions are not affected. In our algorithm, the user transactions explicitly mention a prewrite operation before **an** actual write. Also, user transactions explicitly mention whether it wants to access the prewrite design (we call it pre-read operation) or the **final** design (we call it read operation). That is, the existence of a prewrite is visible to the scheduler, data manager (DM) and to user transactions. The user transaction submits prewrite, pre-read, write and read for the design objects it wants to access. Once a transaction is submitted to the Data Manager (DM), the DM
analyses the received transaction. If the transaction has a prewrite operation, the DM will store the announced design for the object in the prewrite-buffer. If the transaction has a preread operation, the DM will return the
corresponrling prewrite design from the prewritebuffer. If the operation is a read, it returns the final produced design from the write-buffer. In our algorithm, two versions of the same design may be available for reading. The first version is the final design released for manufacturing or the last design checked for correctness. The other is the most recent working copy of the design (prewrite design). However, after the final version of the design is produced, prewnte version of the design will be no longer available. That is, after the final design is
released, a read transaction can not access its prewrite design. Also, the independent writing on these two different versions of design are also not allowed.

8 Database Transactions on basis Domain

A methodology for learning across different application domains. Across domain learning involves generalizing elements for example, from Airline, Railway, and Busline domainsÐto a higher-level node, such as Transportation as shown in Fig. 3 . Whenever common knowledge is generalized and a higher level node is created,
the hierarchical structure of the ADB is reorganized. To illustrate, consider the example shown in Fig. . The original structure of the ADB is shown on the left where the Service node has four children nodes: Airline, Railway, Car-Rental, and Video-Rental. By comparing these nodes, the first two have certain similarities between them, so a new node, called Transportation, is created to capture the common

Concurrency Control Algorithm

the conflicting and nonconflicting operations in our model. The following operations on the same design object
conflict :
1. Two prewrites conflict since two prewrite design for the same object can not be announced at the same time in the same prewrite-buffer. It produces a prewrite-prewrite type of the conflict.
2. A prewrite operation conflicts with a pre-read operation since pre-read returns the value from the prewrite-buffer whereas prewrite changes the contents of the prewrite-buffer. Therefore, it will generate a conflict of the type pre-read and prewrite.
3. Two writes conflict since both of them will modify the same design object at the same time.
This will generate a write-write type of the conflict.
4. A write operation conflicts with a read operation since the read **retums** the value from the write-buffer. and a write changes its contents. Therefore, it will generate a conflict of the type read and write.
The following operations, in general, do not conflict :
1. A pre-read operation (to read a prewrite design) and a write operation do not conflict. A write operation operates on the writehEer whereas a pre-read operates on **the** prewrite-buffer.
2. A prewrite and write conflict do not conflict **as** prewrite and writes operate on their respective buffers.
3. A nead and prewrite do not conflict as they operate on different buffers; **read** operates on
write-biuffer whereas prewrite operates on **the** prewrite-buffer.
the ordered-shared locks **are** acquired. Once the prewrite-lock of T2 is converted to the writelock,
the two write operations of the transactions are executed in the order their write-locks are acquired. Another improvement over is that transaction Tz can commit before T I . In, the transactions with ordered-shared locking are allowed to commit in the order they obtained their locks.

elements. Similarly, a new node called Rental is created so that the common elements between Car-Rental and Video-Rental can be stored at the appropriate level of generality. However, the creation of these two new nodes results in the Service node being split into two subnodes. In general, whenever a few nodes are grouped into higher level nodes, the parent node is split into lower level nodes. This reorganization is automatic and ensures that the design knowledge is always stored at the proper level of generality.

Thus, for the reorganization of the hierarchical structure to occur, we need the ability to compare two nodes at the same level and determine if they have enough common elements for propagation and creation of a new node. A number of challenges are involved in making a comparison across different application domains :

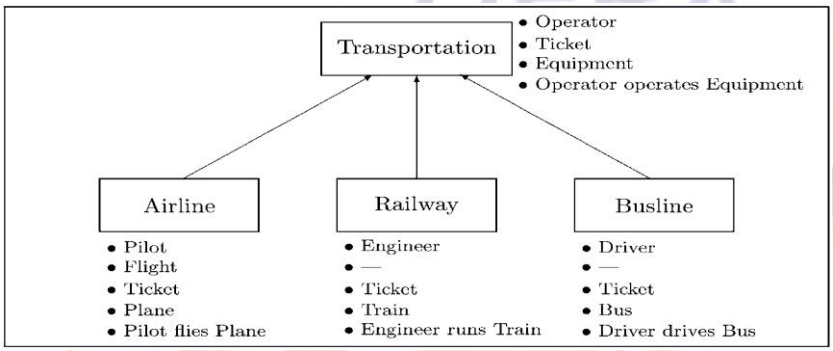

Fig. 3. Learning across application domains.

Algorithm: Across Domain Learning

Input: Entities and Relationships from Domains $\mathcal{A}$ and $\mathcal{B}$.

START

Step 1. Choose the largest possible clusters $\sigma_{\mathcal{A}}$ and $\sigma_{\mathcal{B}}$ (with the same number of entities in each cluster) from domains $\mathcal{A}$ and $\mathcal{B}$, respectively. Let n be the number of entities in a cluster.

Step 2. For all entities $A_i \in \sigma_{\mathcal{A}}$ and $B_j \in \sigma_{\mathcal{B}}$, calculate $d_E(A_i, B_j)$ from Equation (1). Solve the assignment problem to find the best (A_i, B_j) pairings. Reorder the entities in $\sigma_{\mathcal{B}}$ so that A_i is paired with B_i.

Step 3. Solve the simultaneous equations in Equation (4) to obtain $d(A_i, B_i)$.

Step 4. Calculate the inter-cluster distance as:

$$d(\sigma_{\mathcal{A}}, \sigma_{\mathcal{B}}) = \frac{1}{n} \sum_{i=1}^{n} d(A_i, B_i).$$

If $d(\sigma_{\mathcal{A}}, \sigma_{\mathcal{B}}) < t_{\text{Same}}$, then go to Step 6. Otherwise, choose another cluster pair $(\sigma_{\mathcal{A}}, \sigma_{\mathcal{B}})$ (with n entities each) that has not been compared before. If no such pair exists, choose clusters with less number of entities and reset n to the number of entities.

Step 5. If $n \geq 2$, then go to Step 2. Otherwise, STOP.

Step 6. Create a new node as a parent of $\mathcal{A}$ and $\mathcal{B}$. For all $i \leq n$, if $d(A_i, B_i) < t_{\text{Same}}$, propagate the generalization of A_i and B_i as an entity to the new node. Propagate all the relationships among the propagated entities. STOP.

END.

shown in Fig. . Comparing the domains Airline and Railway, a higher-level node (say, Transportation) could be created with entities Operator (a generalization of Pilot and Engineer), Equipment (a generalization of Airplane and Train), Company (a generalization of Airline and Railway), and Passenger (a generalization of Freq_Flyer), and the relation ships between them. This generic model would be used to reason about subsequent transportation applications. For example, a bus company should have Driver and Bus entities, specializations of Operator and Equipment, respectively. The system would provide the generic names; the more specific terms, such as Driver and Bus, could then be elicited from a user.

The learning process involves analyzing entities and relationships from different domains to discover general facts that can be propagated to a higher-level node in the hierarchical organization of common knowledge. It is necessary to judge what facts are general and whether there is enough knowledge to warrant the formation of a new node. Our approach to making such judgments is similar to cluster analysis , where a set of objects is partitioned into disjoint subsets (called clusters) based on a distance measure. When comparing two domains, find the largest cluster (subschema) from one domain whose distance from a corresponding similar cluster (subschema) of another domain falls below a threshold. The common elements (entities and relationships) of these clusters are then propagated to form a new node. The process of clustering involves estimating the closeness between two clusters. We use a distance measure similar to the one used , to judge the closeness between two clusters. The distance between two clusters is calculated as an average of the distances of their corresponding elements (entities). The distance between two entities, in turn, is calculated as an average of the distances of the elements of the entities.

9 Database Transactions on basis Architecture

The Shared Information Platform(SIP) is an effective way to settle the information bottleneck of freeway construction. The abroad studies of ITS pay much attention to information technology, system integration and integrated information platform.The domestic studies of Shared Information Platform focus on two aspects, one is the traffic information system(TIS), the other is the urban transportation system(UTS).The Shared Information Platform of Freeway Transportation (SIPFT) is the sub-platform of SIP for transportation (SIPT).

The shared information of SIPFT is the data which need to be organized and stored by the database system, including static traffic information, dynamic traffic information and
other relevant information of management information system. The input information of database system involves a large number of highway attribute data and complex spatial
data. The mass data of GIS and the attribute data of traffic information constitute the core of platform database According to the construction of national freeway information system, the above-mentioned information belongs to different management department and application
system, and the corresponding databases are built with different management funtions. Such as the database of charge system, the database of monitoring system, the database of highway project, the database of maintenance system, the database of disaster emergency management
system and so on. Based on the user's request of data from different sources.

SYSTEM DESIGN

A. Idea of System Design

1) Construction process

As we all know, Domestic freeway infrastructure is in a period of large-scale development, the new road and the corresponding sub-system platform continue to emerge. However, the construction technology of the SIP is not mature, which need more accumulation of experience. So, the database system of platform can not be realized in one step, the way of progressive stages and gradually improving should be adopted.

2) Information processing

GIS integrates database technology and computer graphics technology, possesses powerful function of data management and space analysis, and can provide support with space property for data mining, information service and decision-making. Freeway traffic information contains a lot of geographic data, therefore, the database system of platform should be a spatial database system based on GIS.

3) Database organization

The data of SIPFT has the following characteristics: the information sources and users are geographically dispersed, but the management is relative concentrated. The fuction module of sub-system is local control and decentralized management, but platform is overall control and comanagement. In addition, the data of SIPFT meets the characteristics of the distributed database systems, which is identical logically and dispersed physically. Consequently, the distributed database is adopted to organizated the database of SIPFT, which is in line with the management ideas and management methods of SIP.

4) Integration of heterogeneous database

The information of SIPFT is derived from different subsystems, for this reason, the database design, data structure and communication way of those sub-systems are different.

Because the databases of sub-systems can not be changed when the database of platform is build, the federated database system(FDBMS) can be used to resolve such a heterogeneous distributed database. The FDBMS, under the premise of maintaining the independent management of sub-systems, realizes the integration of the heterogeneous database.

5) Integration model

Part of the databases of domestic freeway information systems are relatively mature, so, the integrated model of platform database should combine distributed database with the comprehensive database of commanding center. The distributed database of new information systems and the comprehensive database should be built using top-down approach, On the contrary, the distributed database of the existing system should be federated into platform using down-top approach

The model of multi-level structure and co-existence is applied to the integrated database of SIPFT. The shared information which is frequently used should be storaged by the comprehensive database of commanding center. The less used shared information as well as the unshared information should be storaged by the distributed database, the stored location and the updated time of those informations are recorded in the comprehensive database, which can be extracted when they need to be queried or used. This system model, not only supporting the independent sub-system but also supporting the sub-system coordinating with each other, achieves the integration of heterogeneous database and the sharing of traffic information, at the same time, the specific management of the sub-system is maintained. The comprehensive database and heterogeneous distributed database can be connected by LAN or WAN. The system frame is shown in Fig.4 .

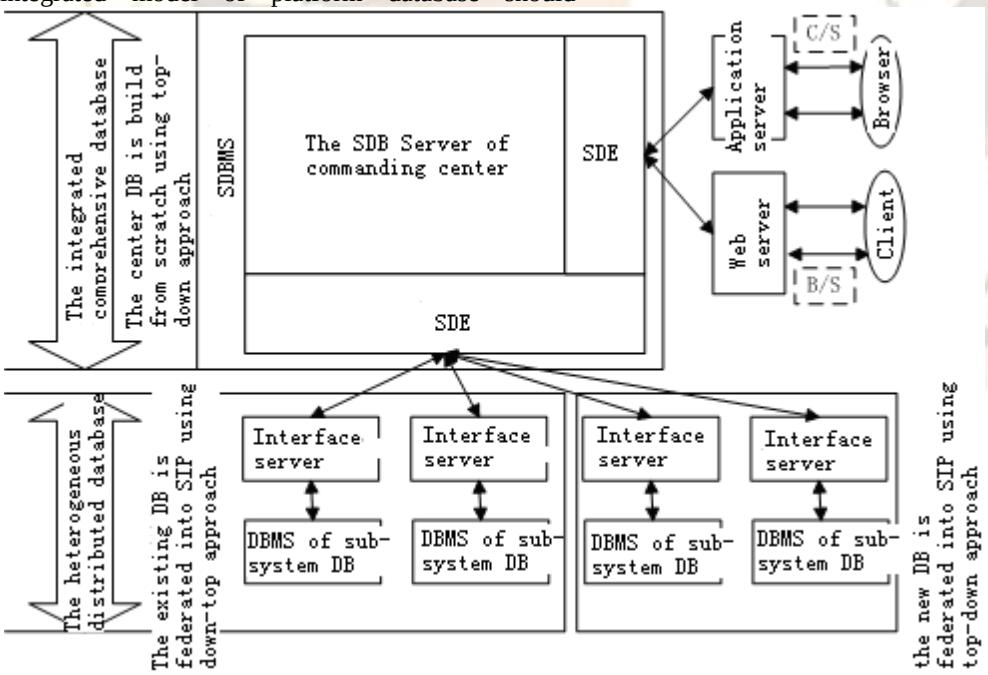

Figure 4. Frame of database system

The comprehensive database includes spatial database (SDB), spatial database management system

(SDBMS) and spatial database engine (SDE). Among those, SDE plays the role of middleware. Because the development of information system of domestic highway is in its infancy, the comprehensive database can be built from scratch. The construction of database can be overall planned in accordance with the unified thinking. The heterogeneous distributed database of the subsystem can be local visited. In order to realize the mutual visit of the
sub-systems, interface server needs to be installed. Through the development of uniform standards for information, the comprehensive database can be connected well with the database of sub-system. The mature database of sub-system can be federated into SIP using down-top approach and the new database using top-down approach.

A. Frame of Database System

Based on the space database of GIS, linked by mobile communication and network communication, the present system integrates the related systems seamlessly, including
vehicle location tracking system, electrical and mechanical equipment management system, emergency response system, information publication system, traffic flow analysis system
and network charge system. The overall structure of database system is shown in Fig. .

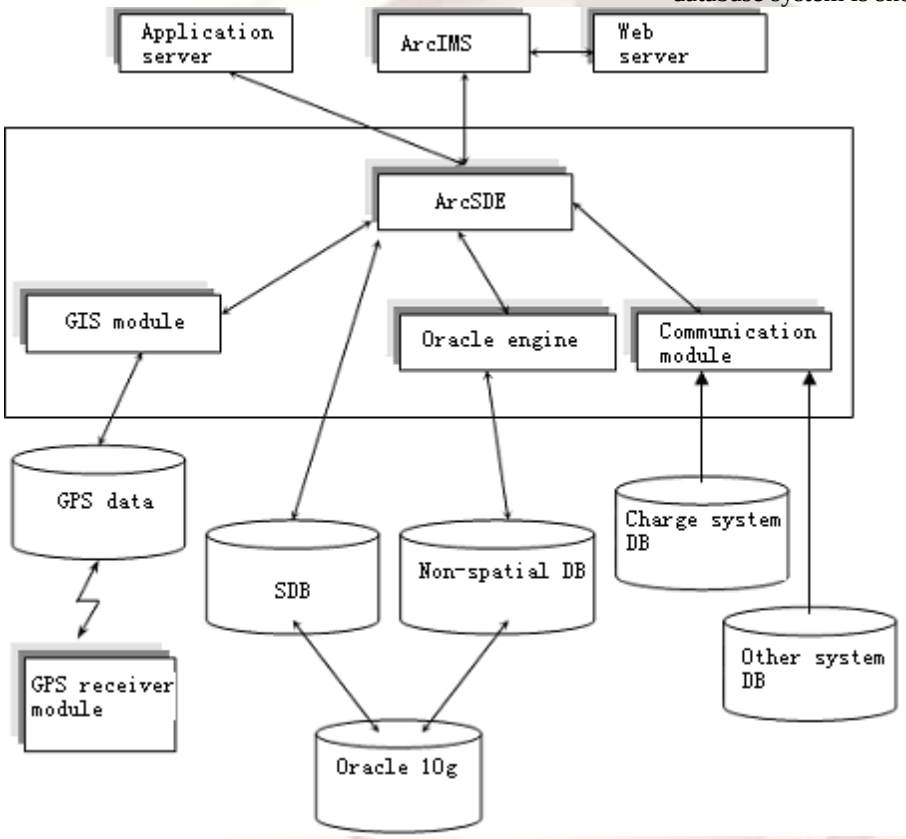

Figure 5 . Overall structure of database system

B. Data Classification of Platform

The platform data can be divided into non-spatial data and spatial data according to data features. Spatial data includes topographic maps, route data, and facilities locations of traffic engineer and so on. Non-spatial data includes the information of electrical and mechanical equipment management, network charge, real-time location of vehicle tracking, road maintenance and office automation.

Conclusion :- All above Techniques suggested the convenient way of Transaction view of databases as per requirements of user to enhanced the Performance

References :-

1) Christina Liebelt "Designing Consistency-Preserving Database Transactions" 0730-` 3157/89/OOO0/0300 **1989 IEEE**

2) Jia-Ling Koh and Arbee L.P. Chen" Query Execution Strategies for Missing Data in Distributed Heterogeneous Object Databases" **Proceedings of the 16th ICDCS1063-6927/96 ,1996** IEEE.

3) C.C. Chibelushit, **S.** Gandon, **J.S.D.**Masont, F. Deravit, R.D. Johnston" DESIGN ISSUES FOR A DIGITAL AUDIO-VISUAL INTEGRATED DATABASE" **1996 The Institution of Electrical Engineers. Printed and published by the IEE, Savoy Place, London WC2R OBL.**

4) Sanjay Kumar Madria and Abdullah Embonig "User Defined Prewrites for Increasing Concurrency in Design Databases" **International Conference on Information, Communications and Signal Processing ICICS '97 Singapore, 9-12 September 1997,0-7803-3676-3/97, 1997** IEEE

5) Veda C. Storey,Debabrata Dey "A Methodology for Learning Across Application Domain for Database Design System" IEEE TRANSACTIONS on Knowledge and Data Engineering,Vol 14,No.1,January/February 2002,1041-4347,2002 IEEE

6) Jia-Ling Koh and Arbee L.P. Chen, Member, IEEE "Efficient Query Processing in Integrated Multiple Object Databases with Maybe Result Certification" IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 14, NO. 4, JULY/AUGUST 2002

7) Zhao Han-tao, Zhai Jing "Database System Design and Application of Freeway Shared Information Platform" 2009 First International Workshop on Database Technology and Applications, 2009 IEEE

8) Anastasia Ailamaki " Embarrassingly Scalable Database Systems" , ICDE Conference 2011 978-1-4244-8960-2/11, 2011 IEEE

9) Vaidehi V, Sharmila Devi D "Distributed Database Management and Join of Multiple Data Streams in Wireless Sensor Network using Querying Techniques", IEEE-International Conference on Recent Trends in Information Technology, ICRTIT 2011 IEEE MIT, Anna University, Chennai. June 3-5, 2011

10) Kjetil Nørvag, Olav Sandsta, and Kjell Bratbergsengen," Concurrency Control in Distributed Object-Oriented Database Systems" Advances in Databases and Information Systems, 1997

11) David J. DeWitt, Jim Gray," Parallel Database Systems: The Future of High Performance Database Processing" Appeared in Communications of the ACM, Vol. 36, No. 6, June 1992.

12) Arun Kumar Yadav, Dr. Ajay Agarwal," A Distributed Architecture for Transactions Synchronization in Distributed Database Systems" Arun Kumar Yadav et. al. / (IJCSE) International Journal on Computer Science and Engineering Vol. 02, No. 06, 2010.

13) Ali Asghar Alesheikh" Design and Implementation of a Moving Object Database for Truck Information Systems" Department of GIS, Faculty of Geomatics Eng., K.N. Toosi University of Technology ValiAsr St., Vanak Sq., Tehran, Iran , MME08 PN 32

14) **Paul Hardy, Marc-Olivier Briat, Cory Eicher, Thierry Kressmann,"** DATABASE-DRIVEN CARTOGRAPHY FROM A DIGITAL LANDSCAPE MODEL, WITH MULTIPLE REPRESENTATIONS AND HUMAN OVERRIDES" ICA Generalisation Workshop, Leicester, August 2004 – Paul Hardy, ESRI.

15) S.BING YAO, ALAN R.HEVNER,"Analysis of Database System Architectures Using Benchmarks" IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. SE-13, NO. 6, JUNE 1987