SRI VENKATESHWARAA COLLEGE OF ENGINEERING & TECHNOLOGY
(Approved by AICTE, New Delhi & Affiliated to Pondicherry University, Puducherry.)
13-A, Villupuram – Pondy Main road, Ariyur, Puducherry – 605 102.
Phone: 0413-2644426, Fax: 2644424 / Website: www.svcetpondy.com

Department of Computer Science and Engineering

Subject Name: **MOBILE COMPUTING** Subject Code: **CS E84**

UNIT IV

Mobile Data Management: Mobile Transactions - Reporting and Co Transactions –Kangaroo Transaction Model - Clustering Model –Isolation only transaction – 2 Tier Transaction Model – Semantic based nomadic transaction processing.

2 Marks

1. Classify the mobile data management.

Mobile data access can be broadly classified into two categories:

(1) Data access in mobile client/server.

(2) Data access in ad-hoc networks.

2. What is meant by mobile transaction?

A mobile transaction is a long-live transaction whose locus of control moves along with the mobile user. Mobile transactions may access remote data wirelessly, t h r o u g h a weak connection, or may access local replicas of data in disconnected mode. The differences between mobile and distributed transaction management are significant because their goals are different.

3. Write about ACID properties.

ACID (Atomicity, Consistency, Isolation, Durability)

ACID (Atomicity Consistency Isolation Durability) is a concept that Database Professionals generally look for when evaluating databases and application architectures. For a reliable database all this four attributes should be achieved. **Atomicity** is an all-or-none proposition.

Consistency guarantees that a transaction never leaves your database in a half-finished state.

Isolation keeps transactions separated from each other until they're finished.

Durability guarantees that the database will keep track of pending changes in such a way that the server can recover from an abnormal termination.

4. Differentiate between distributed transaction and mobile transaction.

In distributed transactions, the main goal is maximizing availability while achieving ACID properties. In mobile transactions, maximizing reliability while achieving some sort of consistency is the main goal.

5. Write short notes on reporting and co-transaction.

This model is based on the Open Nested transaction model. A computation in the mobile environment is considered to consist of a set of transactions, some of which may execute on the mobile node and some of which may execute on the fixed host. The model addresses sharing of partial results while in execution, and maintaining computation state in a fixed node so that the communication cost is minimized while the mobile host relocates. The model proposes to modify Reporting and Co-Transactions to suit mobile environments. The model defines a mobile transaction to be a set of relatively independent transactions which interleave with other mobile transactions.

6. What are the types of transactions?

Atomic transactions

Compensable transactions

Reporting transactions Co-

transactions

7. What is mean by Kangaroo transaction model? APRIL/MAY2014

This model introduced in is based on global transactions and the split transaction models, where transaction relocation is achieved by splitting the transaction at the point of hand-off. A mobile transaction (called Kangaroo transaction) is considered a global transaction in a multi database environment. A Kangaroo transaction (KT) is a global transaction that consists of a set of Joey Transactions (JT). A JT is associated with the base station or the cell in which it executes.

8. Define clustering model.

This model described in assumes a fully distributed system. The database is divided into clusters . A cluster defines a set of mutually consistent data. Bounded inconsistencies are allowed to exist between clusters. These inconsistencies are finally reconciled by merging the clusters. The model is based on the open nested transaction model, extended for mobile computing.

9. What is meant by m-degree consistent?

Consistency between clusters can be defined by an m-degree relation, and the clusters are said to be m-degree consistent. The m-degree relation can be used to define the amount of deviation allowed between clusters. In this model, a mobile transaction is decomposed into

a set of weak and strict transactions. The decomposition is done based on the consistency requirement.

10. What is mean by isolation only transaction.

The Coda file system at CMU provides an application-transparent file system for mobile clients by using file hoarding and optimistic concurrency control. A proxy logs all updates to the file system during disconnection and replays the log on reconnection. Automatic mechanisms for conflict resolution are provided for directories and files through the proxy and the file server. Hoarding is based on user-provided, prioritized list of files. Periodically, the proxy walks the cache to ensure that the highest priority files are present and consistent with the server. Coda provides Isolation-only Transactions (IOT) to automatically detect read/write conflicts that could occur during disconnection.

11. Define 2 tier transaction model.

A two-tier replication scheme has been proposed in whereby mobile disconnected applications are allowed to propose tentative update transactions. On connection, tentative transactions are applied to (re-processed at) the master data copy in the fixed network. At the re-processing stage, application semantics are used (such as finding commutative operations) to increase concurrency. To reduce re-processing costs that can be high in certain occasions, the work in uses a history-based approach

12. Write short notes on semantic based nomadic transaction process.

The semantics-based mobile transaction processing scheme views mobile transactions as a concurrency and cache coherence problem. It introduces the concepts of fragment able and reorders able objects to maximize concurrency and cache efficiency exploiting semantics of object operations. The model assumes the mobile transaction to be long-lived with unpredictable disconnections. Traditional definitions of concurrency and serializability are too restrictive for most operations. Commutatively of operations is an important property which allows concurrent operations on an object. If certain operations on an object are commutative, then the database server can schedule these operations in an arbitrary manner.

13. Explain briefly about compensable transaction.

Atomic transactions whose effects cannot be undone at all. When ready to commit, the transaction delegates all operations to its parent. The parent has the responsibility to commit or abort the transaction later on.

14. What is atomic transaction?

Atomic transactions are the Normal components and may be compensable with atomic compensating dual steps.

15. What is reporting transaction?

Reporting transactions can make its results available to the parent at any point of its execution. It could be a compensating or a non-compensating transaction.

16. Define co-transaction.

Co-transactions: behave in a manner similar to the co-routine construct in programming languages. Co-transactions retain their current status across executions; hence they cannot be executed concurrently.

17. What is meant by RDO?

An RDO is an object (code and data) with a well-defined interface that can be dynamically loaded into a mobile client from a server computer, or vice versa, to reduce client-server communication requirements, or to allow disconnected operation.

18. What is meant by QRPC?

Queued remote procedure call is a communication system that permits applications to continue to make non-blocking remote procedure calls even when a mobile client is disconnected; requests and responses are exchanged upon network reconnection.

19. Explain Mobile Data Management strategy.

A mobile data management strategy is a structure imposed on a complex data model that is to be navigated by a user on a mobile device. This is a relatively new process born from the popularity of mobile applications that requires a flexible and in-depth navigation structure. There are several methods for such strategy, each describing approaches to a variety of tasks or activities.

11 Marks

1. EXPLAIN MOBILE DATA MANAGEMENT.

Mobile data access can be broadly classified into two categories: (1) data access in mobile client/server, and (2) data access in ad-hoc networks. Several research projects from each category are presented in the following subsections.

Mobile Client/Server Data Access

In the first category, mobile data access enables the delivery of server data and the maintenance of client-server data consistency in a mobile and wireless environment. Efficient and consistent data access in mobile environments is a challenging research area because of the weak connectivity and resource constraints. The data access strategies in a mobile information system can be characterized by delivery modes, data organizations, and consistency requirements, among other factors. The mode for server data delivery can be *server-push, client-pull,* or a hybrid of both. The server-push delivery is initiated by server functions that push data from the server to the clients. The client-pull delivery is initiated by client functions

which send requests to a server and "pull" data from the server in order to provide data to locally running applications. The hybrid delivery uses both server-push and client-pull delivery. The data organizations include mobility-specific data organizations like mobile database fragments in the server storage and data multiplexing and indexing in the server-push delivery mode. The consistency requirements range from weak consistency to strong consistency.

Broadcast Disks: A Server PUSH Approach

When a server continuously and repeatedly broadcasts data to a client commu-nity, the broadcast channel becomes a "disk" from which clients can retrieve data as it goes by. The broadcasting data can be organized as *multiple disks* of different sizes and speeds on the broadcast medium. The broadcast is created by multiplexing chunks of data from different disks onto the same broadcast channel. The chunks of each disk are evenly interspersed with each other. The chunks of the fast disks are repeated more often than the chunks of the slow disks (see Figure 7.7). The relative speeds of these disks can be adjusted as a parameter to the configuration of the broadcast. This use of the channel effectively puts the fast disks closer to the client while at the same time pushing the slower disks further away

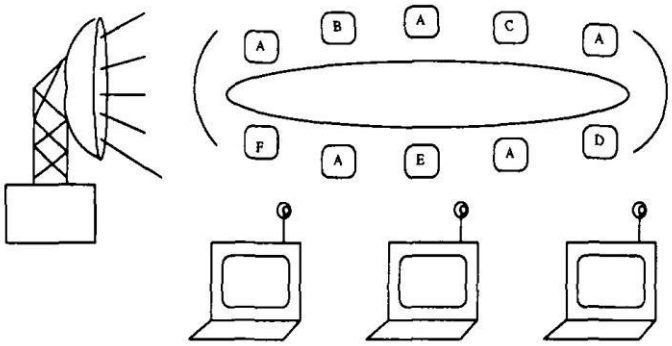

Figure 7.7 A simple broadcast disk

Odyssey: A Client PULL Approach

Odyssey is a CMU research project led by M. Satyanarayanan. It ad- dresses an application-aware adaptation approach to deal with application diversity and concurrency in mobile environments. The application-aware adaptation is implemented with the support of system-coordinated, type-specific operations. It supports concurrent execution of diverse mobile applications that execute on mobile clients, but read or update remote data on servers. The data accessed by an application may be stored in one or more general-purpose repositories such as file servers, SQL servers, or Web servers. It may also be stored in more specialized repositories such as video libraries, query-by-image- content databases, or back– ends of geographical information systems.

Ideally, a data item available on a mobile client should be indistinguishable from that available to the accessing application if it were to be executed on the server storing that item. But this correspondence may be difficult to preserve as mobile resources become scarce; some form of degradation may be inevitable. In Odyssey, *fidelity* is used to describe the degree to which data presented at a client matches the reference copy at the server. Fidelity has many dimensions. One well-known, universal dimension is consistency. For Video applications, data has at least two additional dimensions: frame rate and image quality of individual frames. Odyssey provides a framework within which diverse notions of fidelity can be incorporated.

Rover: A Mobile Objects Approach

The Rover project at MIT provides mobility support to client server applications based on two ideas: reloadable dynamic object (RDO) and queued remote procedure calls (QRPC). An RDO is an object (code and data) with a well-defined interface that can be dynamically loaded into a mobile client from a server computer, or vice versa, to reduce client-server communication requirements, or to allow disconnected operation. Queued remote procedure call is a communication system that permits applications to continue to make non-blocking remote procedure calls even when a mobile client is disconnected; requests and responses are exchanged upon network reconnection. Rover gives applications control over the location where the computation is performed. By moving RDOs across the network, applications can automate the movement of data and/or computation from the client to the server and vice versa.

Mobile Data Access in Ad-hoc Networks

The Bayou project at Xerox PARC developed a system to support data sharing among mobile users. The system is intended to support ad-hoc mobility, where no network infrastructure is assumed to be available. In particular, a user's mobile computer may experience extended disconnection from other computing devices. Bayou allows mobile users to share their appointment calendars, bibliographic databases, meeting notes, evolving design documents, news bulletin boards, and other types of data in spite of their intermittent network connectivity. The Bayou architecture supports shared databases that can be read and updated by users who may be disconnected from other users, either individually or as a group. Bayou supports application-specific mechanisms that detect and resolve the update conflicts, ensures that replicas move towards eventual consistency.

Bayou includes consistency management methods for conflict detection called dependency checks and per-write conflict resolution based on client-provided merge procedures. To guarantee eventual consistency, Bayou servers are able to rollback the effects

of previously executed writes and redo them according to a global serialization order. Furthermore, Bayou permits clients to observe the results of all writes received by a server, including tentative writes whose conflicts have not been ultimately resolved. In the Bayou system, each data collection is replicated in full at a number of servers. Applications running as clients interact with the servers through the Bayou API, which is implemented as a client stub bound with the application. This API, as well as the underlying client-server RPC protocol, supports two basic operations: Read and Write. Read operations permit queries over a data collection, while Write operations can insert, modify, and delete a number of data items in a collection.

2. EXPLAIN MOBILE TRANSACTIONS.

A mobile transaction is a long-live transaction whose locus of control moves along with the mobile user. Mobile transactions may access remote data wirelessly, through a weak connection, or may access local replicas of data in disconnected mode. The differences between mobile and distributed transaction management are significant because their goals are different. In distributed transactions, the main goal is maximizing availability while achieving ACID properties. In mobile transactions, maximizing reliability while achieving some sort of consistency is the main goal.

In this section we describe some of the existing approaches to mobile transaction management. The transaction models considered here have been proposed by Chrysanthis, Dunham, Pitoura, Satyanarayanan, Gray, Walborn, and Nielsen. All the models described assume the mobile computing reference model.

Reporting and Co-Transactions

This model is based on the Open Nested transaction model. A computation in the mobile environment is considered to consist of a set of transactions, some of which may execute on the mobile node and some of which may execute on the fixed host. The model addresses sharing of partial results while in execution, and maintaining computation state in a fixed node so that the communication cost is minimized while the mobile host relocates.

The model proposes to modify Reporting and Co-Transactions to suit mobile environments. The model defines a mobile transaction to be a set of relatively independent transactions which interleave with other mobile transactions. A component transaction can be further decomposed into other component transactions allowing arbitrary levels of nesting. Component transactions are allowed to commit or abort independently. If a transaction aborts, all components which have not yet committed may abort. Some of the transactions may have compensating duals and may be compensated. The model classifies mobile transactions into the following four types:

- Atomic transactions: normal components and may be compensable with atomic compensating dual steps.
- Compensable transactions: atomic transactions whose effects cannot be undone at all. When ready to commit, the transaction delegates all operations to its parent. The parent has the responsibility to commit or abort the transaction later on.
- Reporting transactions: can make its results available to the parent at any point of its execution. It could be a compensating or a non-compensating transaction.
- Co-transactions: behave in a manner similar to the co-routine construct in programming languages. Co-transactions retain their current status across executions; hence they cannot be executed concurrently.

In an atomic transaction, a series of database operations either all occur, or nothing occurs. A guarantee of atomicity prevents updates to the database occurring only partially, which can cause greater problems than rejecting the whole series outright. In other words, atomicity means indivisibility and irreducibility. A compensable transaction can be formed from a pair of programs: one that performs an action and another that performs a compensation for that action if and when required. The forward action is a conventional atomic transaction: it may fail before completion, but before failure it guarantees to restore (an acceptable approximation of) the initial state of the machine, and of the relevant parts of the real world.

A reporting transaction reports its results to other transactions by delegating the results. A reporting transaction can have only one recipient at any given point of time. The changes made by a reporting transaction are made permanent only when the receiving transaction commits. If the receiving transaction aborts, the reporting transaction aborts as well. A co-transaction, on the other hand, reports its results in a way similar to reporting transactions. But upon delegation, the transaction stops execution and is resumed from the point it left off. For any pair of co-transactions, either both commit or both abort.

Co transactions are one of the conventional methods to value a company for sale. The main approach of the method is to look at similar or comparable transactions where the acquisition target has a similar business model and similar client base to the company being evaluated. This approach is fundamentally different from that of DCF valuation method, which calculates intrinsic value.

When executing transactions in mobile environment, it is necessary to maintain logs to enable recovery after a system crash. A MH is highly vulnerable to failures due to the loss or theft of equipment, memory loss, etc. In order to recover from such failures, it is necessary to store data objects and their logs at the mobile service stations (MSS) rather than on mobile

host (MH). A MH transfers a transaction's execution to the MSS by moving all the prewrite values and the log records. A separate pre-commit log is maintained for each transaction.

3. THE KANGAROO TRANSACTION MODEL (APRIL/MAY2014)

This model is based on the global transactions and the split transaction models, where transaction relocation is achieved by splitting the transaction at the point of hand-off. A mobile transaction (called Kangaroo transaction) is considered a global transaction in a multi database environment. A Kangaroo T ransaction (KT) is a global transaction that consists of a set of Joey Transactions (JT). A JT is associated with the base station or the cell in which it executes. When the mobile unit moves to a new cell, the JT in the previous cell is split, and one of the JTs is moved to the current cell of the mobile unit. Each JT may consist of a set of local and global transactions.

The model is built upon the existing databases. The transactions are micro- managed by the individual database transaction managers. A Joey Transaction should terminate in an abort, commit, or a split. For a KT to be successful, the last JT in the order of execution should end in a commit or abort, whereas all other JTs should be split. Based on the ability to compensate the split transaction component, a KT can be executed as a whole atomic transaction or in a relaxed mode where only component transactions are executed atomically. Uses a Data Access Agent (DAA) at each BS to manage mobile transactions and the movement of the MH. Mobile transaction's execution is coordinated by the BS with the MH is currently assigned. When MH hops from one cell to another, coordination of the mobile transaction moves to the new BS. Original transaction is split into Joey transactions (JT). One JT at each base station. Each BS coordinates the operations that are executed while the MH was in its cell.

Three Layers:

- source system,
- data access agent, and the
- mobile transaction.

Two Processing Modes: compensating and split.

DAA (Data Access Agent) acts as transaction manager at base station. For each transaction request DAA generates:

- A Kangaroo Transaction (KT) at MH
- A set of Local Transactions (LTs) & Global Transactions (GTs) at local base station called as Joey Transaction (JT).

For each hop a new Joey is created. When a Joey fails, all previous Joeys and KT will abort.

Kangaroo Transactions:

- Data Access Agent (DAA) tracks MH movement by maintaining a linked list of all BS that have been coordinators of the KT.
- List will be used in case of cascading aborts.

THE CLUSTERING MODEL

This model assumes a fully distributed system. The database is divided into clusters. A cluster defines a set of mutually consistent data. Bounded inconsistencies are allowed to exist between clusters. These inconsistencies are finally reconciled by merging the clusters. The model is based on the open nested transaction model, extended for mobile computing. A transaction submitted from a mobile host is composed of a set of weak and strict transactions. Transaction proxies are used to mirror the transactions on individual machines as they are relocated from one machine to another. A cluster is defined as a unit of consistency in that all data items inside a cluster are required to be fully consistent, while data items residing in different clusters may exhibit bounded inconsistency.

Clusters can be defined either statically or dynamically. A wide set of parameters can be used for defining clusters. This could include the physical location of data, data semantics, and user definitions. Consistency between clusters can be defined by an m-degree relation, and the clusters are said to be m-degree consistent. The m-degree relation can be used to define the amount of deviation allowed between clusters. In this model, a mobile transaction is decomposed into a set of weak and strict transactions. The decomposition is done based on the consistency requirement. The read and write operations are also classified as weak and strict. The weak operations are allowed to access only data elements belonging to the same cluster, whereas strict operations are allowed database-wide access. For every data item, two copies can be maintained–one of them strict and the other weak. As mentioned above, a weak operation can access only the local copies of a data item. Weak operations are initially committed in their local clusters. When the clusters are finally merged, they are once again committed across the clusters.

ISOLATION-ONLY TRANSACTIONS (APRIL/MAY2014) (APRIL 2013)

The Coda file system at CMU provides an application-transparent file system for mobile clients by using file hoarding and optimistic concurrency control. A proxy logs all updates to the file system during disconnection and replays the log on reconnection. Automatic mechanisms for conflict resolution are provided for directories and files through the proxy and the file server. Hoarding is based on user-provided, prioritized list of files. Periodically, the proxy walks the cache to ensure that the highest priority files are present and consistent with the server. Coda provides Isolation-only Transactions (IOT) to automatically detect read/write conflicts

that could occur during disconnection. Unlike traditional transactions, it does not guarantee failure atomicity and only conditionally guarantees permanence.

The SEER hoarding system developed at UCLA is based on the Coda file system. It operates without user intervention by observing user activities and predicting future needs. It defines and uses a measure called "semantic distance" between files to determine how best to cluster files together in preparation for hoarding. The semantic difference between two files is based on the time elapsed between the events of opening the files, and on how many reference to other files occurs in between. SEER does not actually hoard files, but instead interfaces with Coda (and other replicated systems) to do the hoarding. SEER also detects hoard misses during disconnection.

THE TWO-TIER TRANSACTION MODEL (APRIL 2013)

A two-tier replication scheme has been proposed in whereby mobile disconnected applications are allowed to propose tentative update transactions. On connection, tentative transactions are applied to (re-processed at) the master data copy in the fixed network. At the re-processing stage, application semantics are used (such as finding commutative operations) to increase concurrency. To reduce re-processing costs that can be high in certain occasions, the work in uses a history-based approach. On reconnection, tentative transactions, which are represented as histories, are merged with base transactions' histories. The merging process quickly identifies the set of tentative transactions that need to be backed out to resolve conflicts.

SEMANTIC-BASED NOMADIC TRANSACTION PROCESSING (APRIL 2013)

The semantics-based mobile transaction processing scheme views mobile transactions as a concurrency and cache coherence problem. It introduces the concepts of fragment able and reorders able objects to maximize concurrency and cache efficiency exploiting semantics of object operations. The model assumes the mobile transaction to be long-lived with unpredictable disconnections. Traditional definitions of concurrency and serializability are too restrictive for most operations. Commutatively of operations is an important property which allows concurrent operations on an object. If certain operations on an object are commutative, then the database server can schedule these operations in an arbitrary manner. Recovery also becomes quite simplified. Operations may be commutative either for all states or part of the states of the objects. The I/O values of the operations can be used to redefine serial dependencies of the operations. Though this may improve concurrency, it may require complex recovery mechanisms than normal schemes. Organization of the object can be used for selective caching of the object fragments, necessary

for continuing the operation during the disconnected state. This approach reduces the demand on the limited wireless bandwidth and provides better utilization of the cache space available on the mobile host. Application semantics can also be utilized to define the “degree of inconsistency,” “degree of isolation,” and the “degree of transaction autonomy”. Techniques like epsilon serializability and quasi copies can be used to specify allowable inconsistencies in the system.

This approach utilizes the object organization to split large and complex objects into smaller easily manageable pieces. The semantic information is utilized to obtain better granularity in caching and concurrency. These fragments are cached and/or operated upon by the mobile hosts and later merged back to form a whole object. A stationary server sends out the fragments of an object when requested by mobile units. The objects are fragmented by a split operation. The split is done using a selection criteria and a set of consistency conditions. The consistency conditions include the set of allowable operations on the object and the conditions of the possible object states. On completion of the transaction, the mobile hosts return the fragments to the server. These fragments are put together again by the merge operation at the server. If the fragments can be recombined in any order, then the objects are termed “re- orderable” objects. Aggregate items, sets, and data structures like stacks and queues are examples of fragment able objects.