数据库总复习

Lecture 3 Advanced indexing

Spatial index空间索引

Databases can also store data types such as lines and polygons, in addition to raster images.

Allows relational databases to store and retrieve spatial information.

Queries can use spatial conditions (e.g.contains or overlaps).

Queries can mix spatial and non-spatialconditions

两种query一个join：

Nearest neighbor queries: given a point or an object, find nearest objects that satisfy given conditions.

Range queries: deal with spatial regions, e.g.ask for objects that lie partially or fully inside a specified region.

Spatial join of two spatial relations, with location playing role of join attribute.

三种techniques:

1.K-d Tree

2.Quadtree

3.R-Tree: R-tree is useful for indexing sets of rectangles and other polygons.

Lecture 4-6 Query evaluation and optimisation

Materialisation:

Generate results for an expression whose inputs are relations or relations that are already computed.Temporary relations must be materialised(stored) on disk.

Pipelining:

Pass on tuples to parent operations even as operation is being executed (no need to store temporaryresults onto disk).

Lecture 7-8 Transactions, concurrency control and failure recovery

1.Transaction

A transaction is a unit of program execution that accesses and possibly updates various data items.

2.ACID

Atomicity: Either all operations of a transaction are properly reflected in database or none is.

Consistency: Execution of a transaction in isolation preserves consistency of database.

Isolation: Although multiple transactions may execute concurrently, each transaction must be unaware of other concurrently executing transactions. Intermediate transaction results must be hidden from other concurrently executed transactions.

Durability: After a transaction completes successfully, changes it has made to database persist, even if there are system failures.

3.Transaction State

Active: initial state; transaction stays in this state while it is executing.

Partially committed: after final statement has been executed (but commit has not executed).

Failed: normal execution can no longer proceed.

Aborted: after transaction has been rolledback and database restored to its state prior to start of transaction.

Committed: after successful completion.

4.Schedules

A sequence of instructions that specifies chronological order in which concurrent transactions are executed

5.Concurrent Executions

Advantages of concurrent execution of multiple transactions:

increased processor and disk utilisation: better transaction throughput: one transaction can use CPU while another is reading from or writing to the disk

reduced average response time: short transactions need not wait behind long ones.

5.Serialisability

A concurrent schedule is serialisable if it is equivalent to a serial schedule.

(1)**Conflict Serialisability:**

If a schedule S can be transformed into a schedule S´ by a series of swaps of non-conflicting instructions, then S and S´are conflict equivalent. A schedule S is conflict serialisable if it is conflict equivalent to a serial schedule.

(2)**View Serialisability:**

Let S and S´ be two schedules with same set of transactions. S and S´ are view equivalent if following three conditions are met, for each data item Q:

-If in schedule S, transaction Ti reads initial value of Q, then in schedule S' also transaction Ti must read initial value of Q.

-If in schedule S, transaction Ti executes read(Q), and that value was produced by transaction Tj (if any), then in schedule S' also transaction Ti must read value of Q that was produced by same write(Q) operation of transaction Tj .

-Transaction (if any) that performs final write(Q) operation in scheduleS must also perform final write(Q) operation inscheduleS'.

A scheduleS is view serialisable if it is view equivalent to aserial schedule.

Every view serialisable schedule that is not conflictserialisable has blind writes.

Every conflict serialisable schedule is also view serialisable but not vice versa.

6.Recoverability and Cascadelessness

Recoverable schedule: if a transaction Tj reads a data item previously written by a transaction Ti , then commit operation of Ti appears before commit of Tj.

Cascading rollback: a single transaction failure leads to a series of transaction rollbacks.

Cascadeless schedules: for each pair of transactions Ti and Tj such that Tj reads a data item previously written by Ti, commit operation of Ti appears before read operation of Tj.

7.Deadlock and starvation

Deadlock: System is deadlocked if there is a set of transactions such that every transaction is waiting for another transaction.

Starvation: Common solution to recover from deadlock is to roll back one or more transactions. If a transaction is repeatedly chosen as victim, it will never complete its task, hence starvation.

Two approaches to handling deadlocks in 2PL:

Deadlock prevention protocols ensure tha tsystem will never enter a deadlock state.

Deadlock detection methods periodically check for deadlocks (usually with a background thread)and then make a decision on how to break them.

Deadlock Prevention Strategies:

Wait-die scheme — non-preemptive

Older transaction may wait for younger one to release data item. Younger transactions never wait for older ones;they are rolled back instead. A transaction may die several times before acquiring needed data item.

Wound-wait scheme — preemptive

Older transaction wounds (forces rollback) of younger transaction instead of waiting for it. Younger transactions may wait for older ones. May have fewer rollbacks than wait-die scheme.

Deadlock Detection Method: use wait-for graph. A system is in deadlock if and only if wait-for graph has acycle.

8.Two-Phase Locking Protocol

Basic 2PL:

For every transaction:

Phase 1: Growing Phase: Transaction may obtain locks. Transaction may not release locks

Phase 2: Shrinking Phase: Transaction may release locks. Transaction may not obtain locks

Cascading rollback is also possible under basic 2PL. To avoid this,use

Strict two-phase locking: Transaction must hold all its exclusive locks till it commits/aborts. Ensures schedules are recoverable and cascadeless.

Rigorous two-phase locking: is even stricter. Here all locks (including shared locks) are held till commit/abort. Transactions can be serialised in order they commit.

Conservative two-phase locking: A transaction acquires all locks it needs before execution. Prevents deadlocks but may limit concurrency.

Lock manager：

Lock manager can be implemented as a separate process to which transactions sendlock and unlock requests.

Lock manager maintains a data structure called lock table to record granted locks andpending requests. Lock table is implemented as an in-memory hash table indexed on names of data items being locked.

9.Storage Categories

Volatile storage: Does not survive system crashes.

Non-volatile storage: Survives system crashes. But may still fail, losing datan

Stable storage:A mythical form of storage that survives all failures. Usually approximated by maintaining multiple copies on distinct non-volatile media.

10.Log

A log is a sequence of log records,maintaining records of update activitieson stable storage.

A transaction is said to have committed when its commit log record is output tostable storage.

Lecture 9 Object-oriented database and distributed database

1.Object-oriented database

Motivation: Permit non-atomic domains . Allows more intuitive modeling for applications with complex data

(1)Object-relational database system: adding object-oriented features to relational database system.

(2)Object-relational mapping: converting data from native object-oriented type system of programming language to relational representation for storage, and vice versa, automatically.

(3) Object-oriented database system: supporting native object-oriented type system.

2.Distributed database

Distributed Data Independence: Users should not have to know where data is located.

Distributed Transaction Atomicity: Users should be able to write transactions that access and update data at several sites.

Types of Distributed Databases:

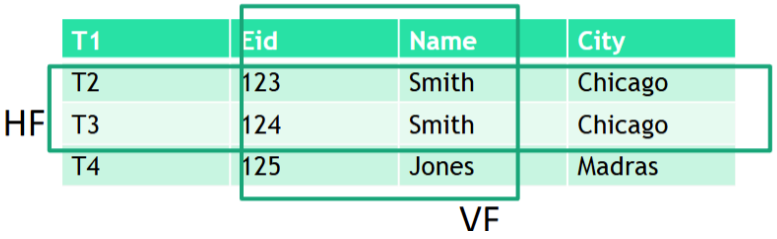
Homogeneous: data is distributed but all servers run same DBMS software.

Heterogeneous: different sites run different DBMSs separately and are connected to enable access to data from multiple sites.

分布式数据库储存的两种方式: Fragmentation and Replication

Fragmentation: breaks a relation to smaller relations and stores fragments at different sites.

- Fragmentation: breaks a relation to smaller relations and stores fragments at different sites.
  - Horizontal fragments (HF) - rows of original data.
    - Selection queries, fragments by city
    - **Disjoint union** of HF must be equal to original relation.
  - Vertical fragments (VF) - columns of original data.
    - Projection queries, e.g. fragments of first two columns
    - Collection of VF must be a **loss-less join decomposition.**



Replication: storing several copies of a relation or fragment. Entire relation can be stored at one or more sites.

(1)**Synchronous replication – all copies of amodified relation are updated before modifying transaction commits.**

- **Voting technique** – a transaction must write a majority of copies to modify an object; read at least enough copies to make sure one of copies is current.
  - For example, 10 copies, 7 are updatable, 4 are read
  - Each copy has a version number, highest is most current.
  - Not attractive and efficient, because reading an object requires reading several copies. Objects are read more than updated.
- **Read-any-write-all technique** – a transaction can read only one copy, but must write to all copies.
  - Reads are faster than writes especially if it's a local copy
  - Attractive when reads occur more than writes
  - Most common technique

(2)**Asynchronous replication – copies of modified relation are updated over a period of time**

两种方法，差别是**Difference lies in how many copies are "updatable"**

Peer to Peer Asynchronous Replication: More than one copy can be designated as updateable (i.e. master copy). Changes to a master copy must be propagated to other copies somehow.

Primary Site Replication: In Primary site one copy is master copy. And in Secondary site, replicas of entire relation created at other sites, cannot be updated. Changes to primary copy transmitted to secondary copies are done in two steps: First capture changes made by committed transactions, then apply these changes. Capture的方法: Log Based Capture :log maintained for recovery is used to generate a Change DataTable (CDT)

Distributed Queries:

Distributed Queries

Example query with relation *S* (fragmented at Shanghai and Tokyo sites):

```
SELECT AVG(S.age)
FROM Sailors S
WHERE S.rating > 3 AND S.rating < 7
```

- Horizontally Fragmented
  - Assume tuples with rating < 5 at Shanghai, >= 5 at Tokyo.
  - When calculating average, must compute sum and count at both sites
  - If WHERE contained just S.rating > 6, just one site at Tokyo.
- Vertically Fragmented
  - *sid* and *rating* at Shanghai, *sname* and *age* at Tokyo, *tid* at both.
  - Joining two fragments by a common *tid* and execute query over this reconstructed relation
- Replicated
  - Since relation is copied to more than one site, choose a site based on local cost.

Distributed Joins:

Distributed Joins

```
SELECT *
FROM Sailors S, Reserves R
WHERE S.sid = R.sid
```

- Fetch as Needed, i.e. ship needed blocks of Reserves to London and perform join with a join algorithm.
  - ***Shipping is much more costly than join!***
- Ship to One Site – Ship entire Reserves to London and perform join with a join algorithm.
- Semi Joins and Bloom Joins (**read textbook for detailed algorithms**): assume that some tuples in Reserves do not join with any tuples in Sailors.
  - Semi-join needs to identify Reserve tuples that guarantee not to join with any Sailors tuples (i.e. shipping projection)
  - Bloom Join is similar to Semi Join but there is a bit-vector shipped in the first step instead of a projection.

Distributed Query Optimisation:

Distributed Query Optimisation

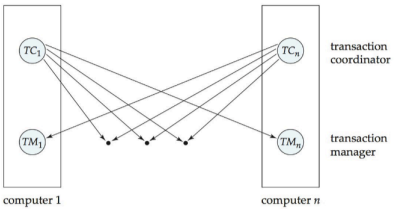
- Consider all plans, pick cheapest; similar to centralised optimisation.
  - Communication costs, if there are several copies of a relation, need to decide which to use.
  - If individual sites are running a different DBMS, autonomy of each local site must be preserved while doing global query planning.
  - Use more efficient distributed join methods (e.g. bloom join and semi-join).
  - Query site constructs a global plan, with suggested local plans describing processing at each site.
    - If a site can improve suggested local plan, free to do so.

Distributed Transactions:

- Transaction is submitted at one site but can access data at other sites.
- Each site has its own local transaction manager and transaction coordinator.
- Function of transaction manager is to ensure ACID properties of local transactions, i.e.
  - Maintaining a log for recovery purposes;
  - Participating in an appropriate concurrency-control scheme to coordinate the concurrent execution of the transactions executing at that site;
  - Failure recovery;
  - etc.

Transaction Coordinator

- Transaction coordinator is responsible for:
  - Starting transaction execution;
  - Breaking transaction into a number of sub-transactions if needed;
  - Distributing sub-transactions to appropriate sites for execution;
  - Coordinating termination of transaction, which may result in transaction being committed at all sites or aborted at all sites.



Distributed Locking Protocols:

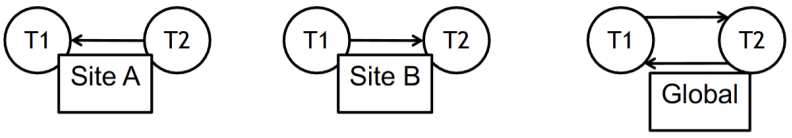
Distributed Locking Protocols

- When locks are obtained and released is determined by concurrency control protocol.
- Lock management can be distributed:
  - Single-lock manager (Centralised) – One site does all locking; vulnerable if one site goes down.
  - Primary Copy – Only one copy of each object is designated a primary copy, requests to lock/unlock are handled by lock manager at primary site regardless where copy is stored.
  - Distributed lock manager – Requests to lock/unlock a copy of an object stored at a site are handled by lock manager at site where copy is stored.

Distributed Deadlock:

Distributed Deadlock

- Each site maintains local waits-for graph, and a cycle in a local graph indicates a **deadlock**.
- A global deadlock might exist even if local graphs contain no cycles.



- Three algorithms of distributed deadlock detection
  - Centralised – send all local graphs to one site that is responsible for deadlock detection.
  - Hierarchal – organise sites into a hierarchy and send local graphs to parent in hierarchy.
  - Timeout – abort transaction if it waits too long.

Distributed Concurrency Control:

Commit Protocols: Two Phase Commit (2PC)

- Site at which transaction originated is coordinator. Other sites running sub-0transactions are subordinates.
- When a user decides to commit, command is sent to coordinator, which initiates **2PC**:
  - Phase 1: Coordinator sends a prepare message to each subordinate; when subordinate receives a prepare message, it decides to abort or commit; subordinate force-writes a no or ready log record and sends a no or yes message to coordinator accordingly.
  - Phase 2: If coordinator receives a yes message from all subordinates, force-writes a commit log record and sends commit message to all subordinates; else force-writes an abort log record and sends an abort message. Subordinates force-write abort or commit log record based on the message they receive.

In some implementations, after two phases, subordinates send acknowledgement message to coordinator; after coordinator receives ack messages from all subordinates it writes an end log record for transaction.

Distributed Recovery:

Distributed Recovery

- When a site comes back from a crash there is a recovery process that reads log and processes all transactions which executed commit protocol at time of crash.
- Failure of subordinate
  - Examine its own logs, if there is commit or abort log record for transaction T, then redo T.
  - Log only contains a <ready T> record, repeatedly contact coordinator or other active sites to find status of T, then performs redo/undo accordingly and write commit/abort log records depending on coordinator's response;
  - Log contains no control records (abort, commit, ready) concerning T, undo.
- Failure of coordinator
  - Participating sites must decide fate of T.
  - If an active site contains <commit T>/<abort T> record in its log, then T must be committed/aborted.
  - If some active site does not contain a <ready T>, abort T.
  - If active sites have a <ready T> record in their logs, but no additional control records (such as <abort T> or <commit T>), it is impossible to determine if a decision has been made and what that decision is, until coordinator recovers. (*in-doubt transaction*)

Lecture 10 Web Technologies and Data Storage

1.Basic concepts on XML

XML: Extensible Markup Language

Ability to specify new tags, and create nested tag structures make XML a great way to store and exchange data.

Tags make data (relatively) **self-documenting** (to humans)

Comparison with Relational Data：

(1) Inefficient: tags, which represent schema information, are repeated

(2) Better than relational tuples as a data-exchange format

Attributes vs. Subelements：

In context of documents, attributes are part of markup, while subelements contents are part of basic document contents

In context of data representation, difference is unclear and may be confusing

Storage of XML Data：

Non-relational data stores: Flat files, XML database

Relational database: String Representationn, Tree Representationn, Map to relations

Application Program Interface:

SAX (Simple API for XML): Based on parser model, user provides handlers for parsing events

DOM (Document Object Model): XML data is parsed into a tree representation

XML Applications:

Data mediation: storing and exchanging data with complex structures

Standards for data exchange in SOAP Web services

2.Resource Description Framework(RDF)

RDF is a framework for representing information on Web.

A set of such triples is called an RDF graph: Core structure of abstract syntax is a set of triples,each consisting of a subject, a predicate and an object.

subject --predicate--> object

RDFS: Resource Description Framework Schema -Provides basic capabilities for describing RDF vocabularies.

OWL: Web Ontology Language- OWL is Semantic Web (schema) language designed to represent rich and complex knowledge about things,groups of things, and relations between things.

3.Linked Data

Collection of interrelated datasets on Web isreferred to as linked data.

4.SPARQL: Query Web of Data

SPARQL stands for "SPARQL Protocol and RDF Query Language

Express queries across diverse data sources

Contains capabilities for querying required and optional graph patterns

Queries are based on patterns. A SPARQL engine would return resources for all triples that match these patterns.

Lecture 11 Big data and blockchain based storage

1.Big data storage categorisation (lecture notes offer two, either one is acceptable)

Four major categories for NoSQL databases (according to how they store the data):

- Key-value Stores
- Wide-Column Stores
- Document Stores
- Graph Database

- *Classification from textbook*
  - Distributed file systems
  - Sharding across multiple databases
  - Key-value storage systems
    - Textbook categorises all NoSQL storage systems as key-value stores
  - Parallel and distributed databases

Map reduce

Map Reduce vs. Databases

- Map Reduce is widely used for parallel processing
  - Google, Yahoo, and 100's of other companies
  - Example uses: compute PageRank, build keyword indices, do data analysis of web click logs, ….
  - Allows procedural code in map and reduce functions
  - Allows data of any type
- Many real-world uses of MapReduce cannot be expressed in SQL
- But many computations are much easier to express in SQL
  - Map Reduce is cumbersome for writing simple queries
- Relational operations (select, project, join, aggregation, etc.) can be expressed using Map Reduce
- SQL queries can be translated into Map Reduce infrastructure for execution
  - Apache Hive SQL, Apache Pig Latin, Microsoft SCOPE

2.Concept on eventual consistency

A consistency model used in distributed computing to achieve high availability that informally guarantees that, **if no new updates are made to a given data item, eventually all accesses to that item will return last updated value.**

如果对某个数据项停止更新，那么经过一段时间后，所有对该数据项的读取操作最终都会返回最后更新的值

BASE (proposed for Scalable systems)stands for **basically available, soft state,eventually consistent.**

3.Basic concepts on consensus algorithms used in public block chain based storage

Proof of Work (public blockchain): Node needs to solve a cryptographic puzzle in order to add a block

Proof of Stake (public blockchain: Node is chosen to add next block based on amount of currency held, with probability proportionate to stake

4.Properties of blockchain

Decentralisation – majority consensus with no central authority.

Tamper resistance – infeasibility of changing contents of blocks on blockchain.

Irrefutability – users cannot deny having submitted a transaction.

Anonymity – IDs not directly tied to any real-world entities.