



The Preliminary Courses are:

- Data StructureDatabase Principles
- Database Design and Application
- The students should already have the basic concepts about database system, such as data model, data schema, SQL, DBMS, transaction, database design, etc. Now we will introduce the implementation techniques of Database Management Systems. The goal is to

and to through the study of this course.

Database Management Systems and Their Implementation, Xu Lizhen



Introduce the inner implementation technique of every kind of DBMS, including the architecture of DBMS, the support to data model and the implementation of DBMS core, user interface, etc. The emphasis is the basic concepts, the basic principles and the implementation methods related to DBMS core.

Database Management Systems and Their Implementation, Xu Lizhen



Because the relational data model is the mainstream data model, and distributed DBMS includes all aspects of classical centralized DBMS, the main thread of this course is

. The implementation of every aspects of DBMS are introduced according to distributed DBMS. Some contents of other kinds of DBMS are also introduced, including federated database systems, parallel database systems and object-oriented database systems, etc. Along with the continuous progress of database technique, new contents will be added at any time.

Database Management Systems and Their Implementation, Xu Lizhen

÷.,

- Database System Principles (before 1984) ---relational model theory, some query optimization algorithms
- Distributed Database Systems (1985~1994) ---introduce implementation techniques in DDBMS thoroughly and systemically
- Database Management Systems and Their Implementation (after 1995) ---- not limited to DDBMS, hope to introduce the implementation techniques of DBMS more thoroughly. Along with the progress of database technique, new contents can be added without changing the course name.

Database Management Systems and Their Implementation, Xu Lizhen

- 1) Popckl bof) Afpopfroba A q pbp
- 2) T kd bkd fk) Mofk fmibplcAq pbPvppbjp
- O der O j hofpek k) Ge kkbp Dbeohb) A q pb J k dbj bkqPvppj p)00a Baffilk) J Dot -Hill Companies, 2002
- 4) Hector Garcia-Jlifk)Ofaccoby+ARiijk)Aqpb Pvppjpqebljmibopllh
- 5) P+ fkd 1 bq i) Nr bov L mfj f|| qfl k fk AA P
- 6) Courseware: http://cse.seu.edu.cn/people/lzxu/resource

e



The history, classification, and main research contents of database systems; Distributed database system

The composition of DBMS and its process structure; The architecture of distributed database systems

Physical file organization, index, and access primitives

The fragmentation and distribution of data, distributed database design, federated database design, parallel database

7

Database Management Systems and Their Implementation, Xu Lizhen

÷

- (1) According to the development of data model
- No management(bd ob 63): Scientific computing
- File system: Simple data management
- Demand of data management growing continuously, DBMS emerged.
 - > 1964, the first DBMS (American): IDS, network
 - > 1969, the first commercial DBMS of IBM, hierarchical
 - > 1970, E.F.Codd(IBM) bring forward relational data model
 - > Other data model: Object Oriented, deductive, ER, ...

10

11

- (2) A coording to the development of DBMS architectures
- Centralized database systems
- Parallel database systems
- Distributed database systems (and Federated database systems)
- Mobile database systems
- (3) A coording to the development of architectures of application systems based on databases
- Centralized structure : Host + Terminal
- Distributed structure
- Client/ Server structure
- Three tier/ multi-tier structure
- Mobile computing
- Grid computing (Data Grid), Cloud Computing

Database Management Systems and Their Implementation, Xu Lizhen

(4) According to the expanding of application fields

- OLTP
- Engineering Database
- Deductive Database
- Multimedia Database
- Temporal Database
- Spatial Database
- Data Warehouse, OLAP, Data Mining
- XML Database
- Big Data, NoSQL, New SQL

Aucompt Aucompt Lucene/Solf Elasticsearch Mortar D	Infochimps Qubole Compute Amagon Engine EMR Continuuity Metasca Zettaset EMC Greenplum IBM ^{MARR} HortonworksCloudera	451 Research
Dis-Automotive Dis-Automotive	HOMM segurings Channels Labora Laborational zone Laboration form Laboration form Laboration form Laboration form Laboration Principsed Princip	Anaporto Bala Ince Trackare Random Markow



What is DDB?

A DDB is a collection of correlated data which are spread across a network and managed by a software called DDBMS.

Two kinds:

- (1) Distributed physically, centralized logically (general DDB)
- (2) Distributed physically, distributed logically too (FDBS)

We take the first as main topic in this course.

Database Management Systems and Their Implementation, Xu Lizhen

14



- Distribution
- Correlation
- DDBMS

Database Management Systems and Their Implementation, Xu Lizhen

÷.,

- Local autonomy
- Good availability (because support multi copies)
- Good flexibility
- Low system cost
- High efficiency (most access processed locally, less communication comparing to centralized database system)
- Parallel process
- Hard to integrate existing databases
- Too complex (system itself and its using, maintenance, etc. such as DDB design)

Database Management Systems and Their Implementation, Xu Lizhen



Compared to centralized DBMS, the differences of DDBS are as follows:

- Query Optimization (different optimizing goal)
- Concurrency control (should consider whole network)
- Recovery mechanism (failure combination)
- Data distribution

Database Management Systems and Their Implementation, Xu Lizhen

17



- ÷.
 - The components of DBMS core
 - The process structure of DBMS
 - The components of DDBMScore
 - The process structure of DDBMS

Database Management Systems and Their Implementation, Xu Lizhen





- -

- Single process structure
- Multi processes structure
- Multi threads structure
- Communication protocols between processes / threads

Database Management Systems and Their Implementation, Xu Lizhen

program is α [:] ile, running a	mpiled with DBMS core a single process.
Application co	
Application co	
ments	Result
DBMS core (as a f	nction)
.exe file	
mentation, Xu Lizhen	

process corre sponding to one DBMS





 Application or embedde communication control: 	progr ed SQL tion p	ams ac offere rotocol	ccess da d by D to real	atabase BMS, a lize syr	esthrou accordi nchroni	ugh A PI ng to izing	
Ad-hoc interface or application program]	Pip	e0 e1		DBMS co	ore	
Pipe0: Senc Pipe1: retu	Pipe0: Send SQL statements, inner commands; Pipe1: return results. The result format:						
State TupNum	AttNum	AttName	AttType	AttLen		TmpFileName	
		Definitio	n of one attri	bute Defi	nition of othe	r attributes	
Database Management Systems and The	ir Implement	tation, Xu Liz	hen				25



× × ×	
R1R2Site1SiteSelect *From R1,R2Where R1.a = R2.t	Global query optimization may get an execution plan based on cost estimation, such as: (1)send R2 d ptdp)O (2)execute on site1: Select * Col j O)O T e bobO + : O+
Database Management Systems an	Their Implementation, Xu Lizhen 28



The access to database is transferred to the operations on files (of OS) eventually. The file structure and access route offered on it will affect the speed of data access directly. It is impossible that one kind of file structure will be effective for all kinds of data access

- Access types
- File organization
- Index technique
- Access primitives

Database Management Systems and Their Implementation, Xu Lizhen

έ,

- Query all or most records of a file (>15%)
- Query some special record
- Query some records (<15%)
- Scope query
- Update

Database Management Systems and Their Implementation, Xu Lizhen

3

-

- Heap file: records stored according to their inserted order, and retrieved sequentially. This is the most basic and general form of file organization.
- Indexed file: index + heap file/ cluster
- Dynamic hashing: p115
- Grid structure file: p118 (suitable for multi attributes queries)
- Raw disk (notice the difference between the logical block and physical block of file. You can control physical blocks in OSby using raw disk)

Database Management Systems and Their Implementation, Xu Lizhen

- B+ Tree ()
- Clustering index ()
- Inverted file
- Dynamic hashing
- Grid structure file and partitioned hash function
- Bitmap index (used in data warehouse)
- Others

Date	Store	State	Class	Sales
3/1/96	32	NY	Α	6
3/1/96	36	MA	Α	9
3/1/96	38	NY	В	5
3/1/96	41	СТ	Α	11
3/1/96	43	NY	Α	9
3/1/96	46	RI	В	3
3/1/96	47	СТ	в	7
2/4/06	49	NY	Δ	12

	Α	в
Total sales = ? (4*8+4*4+4*2+6*1=62)		
How many class A store in NY ? (3)		
> Sales of class A store in NY = ? (2*8+2*4+1*2+1*1=27)		
How many stores in CT ? (2)		
bin operation (query product list of class A store in NY)		

Database Management Systems and Their Implementation, Xu Lizhen



- int dbopendb(char * dbname)
 : open a database.
- int dbclosedb(unsigned dbid)
 : close a database.
- int dbTableInfo(unsigned rid, TableInfo * tinfo)
 : get the information of the table referenced by *lc*.
- int dbopen(char * tname,int mode, int flag)
 : open the table d and assign a rid for it.
- int dbclose(unsigned rid)
- : close the table referenced by lr and release the lr.
- int dbrename(oldname, new name)
 : rename the table.

```
Database Management Systems and Their Implementation, Xu Lizhen
```

- int dbcreateattr (unsigned rid, sstree * attrlist)
 : create some attributes in the table referenced by *lc*.
- int dbupdateattrbyidx(unsigned rid, int nth, sstree attrinfo)
 : update the definition of the nth attribute in the table referenced by *lc*.
- int dbupdateattrbyname(unsigned rid, char * attrname, sstree attrinfo)
 - : update the definition of attribute d in the table referenced by lr.
- int dbinsert(unsigned rid, char * tuple, int length, int flag)
 : insert a tuple into the the table referenced by *lr*.



- int dbdelete(unsigned rid, long offset, int flag)
 : delete the tuple specified by *w d* in the table referenced by *l*c.
- int dbupdate(unsigned rid, long offset, char * newtuple, int flag)
 : update the tuple specified by *w d* in the table referenced by *lx* with *d d*.
- int dbgetrecord(unsigned rid, int nth, char* buf)
 : fetch out the nth tuple from the table referenced by *lc* and put it into buffer *a e*.
- int dbopenidx(unsigned rid, indexattrstruct * attrarray, int flag)
 : open the index of the table referenced by *lc* and assign a *llc* for it.

Database Management Systems and Their Implementation, Xu Lizhen



- int dbcloseidx(unsigned iid)
 : close the index referenced by *lic*.
- int dbfetch (unsigned rid, char * buf, long offset)
 fetch out the tuple specified by *w d* from the table referenced by *lc* and put it into buffer *a e*.
- int dbfetchtid (unsigned iid, void * pvalue, long*offsetbuf, flag)
 : fetch out the TIDs of tuples whose value on indexed
 qpf r dpe pqpb e f obi ql k t fqe d, and put them into e d a e. He is the reference of the index used.
- int dbpack (unsigned rid)

 : re-organize the relation, delete the tuples having deleted flag physically.



e a

- (1) Centralized: distributed system, but the data are still stored centralized. It is simplest, but there is not any advantage of DDB.
- (2) Partitioned: data are distributed without repetition. (no copies)
- (3) Replicated: a complete copy of DB at each site. Good for retrieval-intensive system.
- (4) Hybrid (mix of the above): an arbitrary fraction of DB at various sites. The most flexible and complex distributing method.

Database Management Systems and Their Implementation, Xu Lizhen

Database Management Systems and Their Implementation, Xu Lizhen

1 2 3 4 flexibility complexity Advantage of DDBS Problems with DDBS

.

- (1) According to relation(or file), that means non partition
- (2) According to fragments
- Horizontal fragmentation: tuple partition
- Vertical fragmentation: attribute partition
- Mixed fragmentation: both

Database Management Systems and Their Implementation, Xu Lizhen

Database Management Systems and Their Implementation, Xu Lizhen

43

•

- (1) Completeness: every tuple or attribute must has its reflection in some fragments.
- (2) Reconstruction: should be able to reconstruct the original global relation.
- (3) Disjointness: for horizontal fragmentation.

É.

(1) Horizontal Fragmentation Defined by selection operation with predicate, and reconstructed by union operation.

SELECT * FROM R WHERE P;

 $\begin{array}{ll} R{\rightarrow}n \mbox{ fragments (use P_1, P_2...P_n)} \\ Fulfill: $P_i{\wedge}P_j$: c ipb f g $P_1{\vee}P_2{\vee}...{\vee}P_n$=true $ \end{array}$

Derived Fragmentation: relation is fragmented not $l \circ a fkd q f q p bic p q p f r q p) r q q k l q b o o b q l k p fragmentation.$



TEACHER(TNAME, DEPT) COURSE(CNAME, TNAME)

Suppose TEACHER has been fragmented according to DEPT, we want to fragment COURSE even if there is no DEPT attribute in it. This will be the fragmentation a bofs ba col j QB> E BO pco dj bkq qfl k+

Semijoin: $R = \Pi_R(R = S)$

∴ TEACHER9 = SELECT * FROM TEACHER T E BOB ABMQ 6th

COURSE9=COURSE TEACHER9

Database Management Systems and Their Implementation, Xu Lizhen

4



Defined by project operation, and reconstructed by join operation. Note:

- Completeness: each attribute should appear in at least one fragment.
- Reconstruction: should fulfill the condition of lossless join decomposition when fragmentizing.
- a. Include a key of original relation in every fragment.
- b. Include a TID of original relation produced by system in every fragment.

Database Management Systems and Their Implementation, Xu Lizhen

Apply fragmentation operations recursively. Can be showed with a fragmentation tree:



-

We can simplify a complex problem through fkd oj qfl k e fa fkd j bqe l a

Level 1: Fragmentation Transparency User only need to know global relations, he alk qe sbd hklt fc q by obco dj bkd ba and how they are distributed. In this situation, user can not feel the distribution of data, as if he is using a centralized database.

× ×

- Level 2: Location Transparency
 User need to know how the relations are
 co dj bkqi ba) r qe balk qe s bql t loov qe b
 store location of each fragment.
- Level 3 Local Mapping Transparency User need to know how the relations are fragmentized and how they are distributed, r qebalk qe s bq t loov bs bov il i database managed by what kind of DBMS, using what DML, etc.
- Level 4 No Transparency

Database Management Systems and Their Implementation, Xu Lizhen

5



Check tuple immediately while updating, if there is any inconsistency it is sent back along with ACK information and then sent to the right place.

Database Management Systems and Their Implementation, Xu Lizhen

- ÷,
 - 3) Translation of Global Queries to Fragment Queries and Selection of Physical Copies.
 - 4) Design of Database Fragments and Allocation of Fragments.

Above 1)~3) should be solved in DDBMS. While 4) is a problem of distributed database design.

Database Management Systems and Their Implementation, Xu Lizhen

e je

- 1) Distributed database
- The design of fragments
- The design of fragment distribution solution To understand user's requirements, we
- The sites where this application may occur
- The frequency of this application
- The data object accessed by this application

53

Database Management Systems and Their Implementation, Xu Lizhen





- For vertical fragmentation: Analyze the affinity between attributes, and consider:
- Save storage space and I/ O cost
- Security. Some attributes should not be seen by some users.

(2) Distribution design

Through cost estimation, decide the suitable store location (site) of each distribution unit. p252

Database Management Systems and Their Implementation, Xu Lizhen

- What is parallel database system?
- Share Noting (SN) structure
- Vertical parallel and horizontal parallel
- A complex query can be decomposed into several operation steps, the parallel process of these steps is called vertical parallel.
- For the scan operation, if the relation to be scanned is fragmentized beforehand into several fragments, and stored on different disks of a SN structured parallel computer, then the scan can be processed on these disk in parallel. This kind of parallel is called horizontal parallel.

Database Management Systems and Their Implementation, Xu Lizhen











(1)Arbitrary

Fragmentize relation R in arbitrary mode, then stored these fragments on the disk of different processor. For example, R may be divided averagely, or hashed into several fragments, etc.

(2)Based on expression

Put the tuples fulfill some condition into a fragment. Suitable for the situation in which the most query are based on fragmentation conditions. --- excluded respectively.

Database Management Systems and Their Implementation, Xu Lizhen

62

The difference between PDB and DDB about data fragmentation and distribution

Promote parallel process degree, use the parallel 1 j m do p fifdy p adequately as possible	Promote the local degree of data access, reduce the data transferred on network
PDBM Sfeature and the feature of parallel computer system used, combining application requirements.	Application requirements, combining the feature of DDBMS used.
On multi disks of a parallel computer	On multi sites in the network





-

- In practical applications, there are strong requirements for solving the integration of multi existing, distributed and heterogeneous databases.
- The database system in which every member is autonomic and collaborate each other based on negotiation --- federated database system.
- No global schema in federated database system, every federated member keeps its own data schema.
- The members negotiate each other to decide respective input/ output schema, then, the data sharing relations between each other are established.

Database Management Systems and Their Implementation, Xu Lizhen





÷.

- FS = CS + IS
- FS is all of the data available for the users on site.
- IS is gained through the negotiation with ES of other sites (j≠i).
- $R pbo p nr bov l k CP_i \Rightarrow$ the sub-queries on CS and IS \Rightarrow the sub-queries on corresponding ES.
- The results gained from ES ⇒ the result forms of corresponding IS, and combined with the results get from the sub-queries on CS, then synthesized to the eventual result form of FS.

Database Management Systems and Their Implementation, Xu Lizhen

-

Catalog --- Data about data (Metadata).

Repj fk cr k qfl k fpql qo k peboqe br pbopl mboqfkd demands to the physical targets in system.

- 4.6.1 Contents of Catalogs
- (1) The type of each data object (such as base table, view, . . .) and its schema
- (2) Distribution information (such as fragment location, . . .)
- (3) Access routing (such as index, ...)
- (4) Grant information
- (5) Some statistic information used in query optimization

Database Management Systems and Their Implementation, Xu Lizhen

6

-

(1)~(4) are not changed frequently, while (5) will change on every update operation. For it:

- Update it periodically
- Update it after every update operation
- 4.6.2 The features of catalog
- (1) mainly read operation on it
- (2) very important to the efficiency and the data distribution transparency of the system
- (3) very important to site autonomy
- (4) the distribution of catalog is decided by the architecture of DDBMS, not by application requirements

- - (1) Centralized

Database Management Systems and Their Implementation, Xu Lizhen

- A complete catalog stored at one site.
- Extended centralized catalog: centralized at . first; saved after being used; notify while there is update
- (2) Fully replicated: catalogs are replicated at each site. Simple in retrieval. Complex in update. Poor in autonomy.
- (3) Local catalog: catalogs for local data are stored at each site. That means catalogs are stored along with data.



If want the catalog information about data on other site (look for through broadcast):

- Master catalog: store a complete catalog on some site. Make every catalog information has two copies.
- Cache: after getting and using the catalog information about data on other site through broadcast, save it for future use (cache it). Update the catalog cached through the comparison of version number.



- (4) Different Combination of the above Use different strategy to different contents of catalog, and then get different combination strategies. Such as:
- a. Use fully replicated strategy to distribution information (2), use local catalog strategy to other parts.
- b. Use local catalog strategy to statistic information, use fully replicated strategy to other parts.

```
Database Management Systems and Their Implementation, Xu Lizhen
```

- ÷.
 - Characteristics:
 - There is no global catalog
 - > Independent naming and data definition
 - > The catalog grows reposefully
 - The most important concept --- System Wide Name (SWN)
 - > ObjectName: the name given by user for the data object
 - R pbo r pbo pk j b+T for or fp) a for book of pbop k access different data object using the same name.

```
Database Management Systems and Their Implementation, Xu Lizhen
```



- > UserSite: the ID of the site where the User is. With this, different users on different sites can use the same user name.
- BirthSte: the birth site of the data object. There is no global catalog in R* system. At the BirthSte the information about the data is always kept even the data is migrated to other site.
- Print Name (PN): user used normally when they access a data object.

<PN>::=[User[@UserSite].]ObjectName[@BirthSite]

```
Database Management Systems and Their Implementation, Xu Lizhen
```

74





5) ObjectName@BirthSite

If no match for the ObjectName is found in (2), (3) or print name is in the form of (4) or (5), name completion is used.

- A missing User is replaced by current User.
- A missing UserSite or BirthSite is replaced by current site ID.

1



_

- " Obt of dp qe b nr bov pq dpj bk qp submitted by user first, and then decide the most effective operating method and steps to get the result.
- Qebdl i fpql d fk qebobpriqlcrpbop query with the lowest cost and in shortest time.

Database Management Systems and Their Implementation, Xu Lizhen



Database Management Systems and Their Implementation, Xu Lizhen



SP2 = SP S2

Database Management Systems and Their Implementation, Xu Lizhen

×,

SELECT SNAME FROM S, SP, P WHERE S.SNUM=SP.SNUM AND SP.PNUM=P.PNUM AND S. FQ : kgkd AND P.M > J B: liq AND SP.QUAN>1000;













so many possible execution strategies. So it is a complex task.

ase Management Systems and Their Implementation, Xu Lizhen Datab

That is so called algebra optimization. It takes a series of transform on original query expression, and transform it into an equivalent, most effective form to be executed. For example: $\Pi_{\text{NAME,DEPT}}\sigma_{\text{DEPT=15}}~BJ~M~\sigma_{\text{DEPT=15}}~\Pi_{\text{NAME,DEPT}}$ (EMP) (1)Query tree

For example: $\Pi_{SNUM}\sigma_{>OB>: LOCE}$ (SUPPLY DEPT)

Γ	I(SNUM)	Leaves: global relation
	>OB> : 1 cae	Middle nodes: unary/ k
	- · · · · · · · · · · · · · · · · · · ·	operations
	DEPTNUM=DEPTNUM	Leaves \rightarrow root: the exe
Supply	DEPT	order of operations

dle nodes: unary/ binary ations ves \rightarrow root: the executing of operations

📕 🙀 (2) The equivalent transform rules of relational algebra

- 1) Exchange rule of / : E1 E2 B E1
- 2) Combination rule of / : E1 (E2 E3) E1 E2) E3
- 3) Cluster rule of Π : $\Pi_{2 \to k}(\Pi_{j} B \Pi_{2 \to k}(E),$ legal when $A_{1} \ge_{n}$ is the sub set of $\{B_{1} \ _{m}\}$
- 4) Cluster rule of σ : $\sigma_{F1}(\sigma_{F2} B = \sigma_{F1} F_2(E)$
- $_{6}$ If the attributes in F are all the attributes in E1, then $\sigma_{F}(E1~B~~\sigma_{F}(E1)~E2$

Database Management Systems and Their Implementation, Xu Lizhen

Database Management Systems and Their Implementation, Xu Lizhen

 $\begin{array}{ll} \mbox{if F in the form of F1} & C \) & ka \ \mbox{e} \ bob \ \ \ obl \ kiv \ B \ p \\ & qpf \ r \ \ \ qpf \ r \ \ qpf \ qpf \ \ qpf \ qpf \ \ qpf \ qpf \ qpf \ \ qp \ \ qp \ \ qpf \ \ qpf \ \ qpf \ qpf \ \ qpf \ q$

9) Suppose $A_1 > {}_n$ is a set of attributes, in which $B_1 {}_m ob B p qqpf r dpp ka {}_1 {}_k ob B p$ attributes, then $\Pi_{> > k}$ (E1 B Π_{j} (E1) Π_{h} (E2)

 In ¬→_k(E1 B ¬→_k(E1) ¬→_k(E2) From the above we can see, the goal of algebra optimization is to simplify the execution of the query, and the target is to make the scale of the operands which involved in binary operations be as small as possible.
 In the general procedure of algebra optimization please refer to p118.

-

- Methods:
- For horizontal fragmentation: R = R1 R2
- For vertical fragmentation: S = S1 S2 Sn
- Replace the global relation in query expression with the above. The expression we get is called canonical expression
- Transform the canonical expression with the equivalent transform rules introduced above. Principles:
- 1) Push down the unary operations as low as possible
- 2) Look for and combine the common sub-expression
- Definition: the sub-expression which occurs more than once in the same query expression. If find this kind of sub-expression and compute it only once, it will promote query efficiency.

91

Rn





Notice: The last query tree is which can be given by an expert at first. The goal of algebra optimization is to optimize the query expression which is not submitted in best form at first.

Database Management Systems and Their Implementation, Xu Lizhen















The executions of local sub-queries are responsible by local DBMS. The query optimization of DDBMS is responsible for the global optimization, that is the execution of assembling tree.

Because the executions of unary operations are responsible by local DBMS after algebra optimization and query decomposition, the global optimization of DDBMS only need to consider the binary operations, mainly the join operation.

Elt ql cfka dlla bpp pp dodv ql lj m dp qe b query improved by algebra optimization is introduced in this section.

Database Management Systems and Their Implementation, Xu Lizhen

9

- 1) Materialization: select suitable copies of fragments which involved in the query
- 2) Strategies for join operation

 $\begin{array}{lll} Non_distributed join: \\ (R^2 U \ R^3) & R^1 \\ Distributed join: \\ (R^1 & R^2) & (R^1 & R^3) \end{array}$

Database Management Systems and Their Implementation, Xu Lizhen

Direct join

Use of semi_join

3) Select execution of each operation (mainly to direct join)

- ÷.,
- Nested loop: one relation acts as outer loop relation (O), the other acts as inner loop relation (I). For every tuple in O, scan I one time to check join condition.

Because the relation is accessed from disk in the unit of block, we can use block buffer to improve efficiency. For R S, if let R as O, S as I, b_R is physical block number of R, b_S is physical block number of S, there are n_B block buffers in system (n_B >=2), and n_B -1 buffers used for O, one buffer used for I, then the total disk access times needed to compute R S is:

 $b_{R} \quad b_{R} / \text{ (n}_{B} \text{-1)} \quad b_{S}$

Database Management Systems and Their Implementation, Xu Lizhen

10

÷.,

- Merge scan: order the relation R and S on disk in ahead, then we can compare their tuples in order, and both relation only need to scan one time. If R and Shave not ordered in ahead, must consider the ordering cost to see if it is worth to use this method (p122)
- Using index or hash to look for mapping tuples: in nested loop method, if there is suitable access route on I (say B+ tree index), it can be used to substitute sequence scan. It is best when there is cluster index or hash on join attributes.
- Hash join: because the join attributes of R and Shave the same domain, R and S can be hashed into the same hash file using the same hash function, then R S can be computed based on the hash file.

tabase Management Systems and Their Implementation, Xu Lizhen

- The above are all classical algorithms in centralized DBMS. The idea is similar when computing join in DDBMS using direct join strategy.
- II. Three general optimization methods:
- 1) By cost comparison (also called exhaustive search)
- By heuristic rule: generally used in small systems. (p124)
- Combination of 1, 2: eliminate the solutions unsuitable obviously by using 2 to reduce solution space, then use 1 to compare cost of the rest solutions carefully.
- III. Cost estimation
 - Total query cost=Processing cost+Transmission cost

According to different environment:

- For wide area network: the transfer rate is about 100bps~50Kbps, far slow than processing speed in computer, so *bd h f b* can be omitted.
- For local area network: the transfer rate will reach 1000Mbps, both items should be considered.
- 1) Transmission cost
- $TC(x)=C_0+C_1x$

x: the amount of data transferred; C_0 : cost of initialization; C_1 : cost of transferring one data unit on network. C_0 , C_1 rely on the features of the network used.

Database Management Systems and Their Implementation, Xu Lizhen

103



 $\begin{array}{l} Processing \ cost = \ cost_{cpu} + \ cost_{I/O} \\ cost_{cpu} \ can \ be \ omitted \ generally. \\ cost \ of \ one \ I/ \ O = D_0 + D_1 \\ D_0: \ the \ average \ time \ looking \ for \ track \ (ms); \\ D_1: \ time \ of \ one \ data \ unit \ I/ \ O \ (\mu s, \ can \ be \ omitted) \\ cost_{I/O} = \ no. \ of \ I/ \ O \ D_0 \end{array}$

Notice: calculate query cost accurately is unnecessary and unpractical. The goal is to find a good solution through the comparison between different solutions, so only need to estimate the execution cost of different solutions under the same execution environment.

Database Management Systems and Their Implementation, Xu Lizhen



I. The role of semi_join

Semi_join is used to reduce transmission cost. So it is suitable for WAN only.

 $R S = \Pi_R(R S)$

if R and S are stored on site 1 and 2 respectively, the steps to realize R $\,$ S with $\,$ is as following:

- 1) Transfer $\Pi_A(S) \rightarrow site1$, A is join attribute
- 2) Execute R $\Pi_A(S) = R$ S on site1 (compress R)
- $_{3}$ Transfer R S \rightarrow site2
- 4) Execute (R S) S = R Son site2

```
Database Management Systems and Their Implementation, Xu Lizhen
```



 The reduce on transmission cost through is gained through the sacrifice on processing cost.
 There are many candidate solutions of semi_join. For example : for the query R₁ R₂ R₃ R_n, consider the to R₁, maybe: R₁ R₂, R₁ (R₂ R₁), R₁ (R₂ R₃) it is almost impossible to select the best from all possible solutions.
 bok pth k pobj oh can be regarded as reducers.
 Definition: A chain of semi_join to reduce R is called reducer program for R.

Database Management Systems and Their Implementation, Xu Lizhen

Database Management Systems and Their Implementation, Xu Lizhen

107

- RED(Q, R): A set of all reducer programs for R in query Q.
- Full reducer: the reducer



Link the two relations with a line if there is between them, then we can get the query graph. The query whose query graph like the left graph is called tree query (TQ)

Example 2: $q = (R_1.A = R_2.B)$ ($R_2.C = R_3.D$) ($R_3.E = R_1.F$)

$$R_2$$
 R_3

Database Management Systems and Their Implementation, Xu Lizhen

The query whose query graph like the left graph is called cyclic query (CQ)

Example 3: $q = (R_1.A = R_2.B)$ $(R_2.B = R_3.C)$ $(R_3.C = R_1.A)$ This is a TQ, not a CQ, because $R_3.C = R_1.A$ can be obtained from transfer relation, it is not a independent condition.



 $\begin{array}{l} q = (R_1.B=R_2.C) \quad (R_2.D=R_3.E) \quad (R_3.F=R_1.A) \\ \text{Even if the result of this query is empty, the size of any} \\ \text{one of } R_1, R_2 \text{ and } R_3 \text{ can not be decreased through} \quad . \text{ So} \\ \text{there is not full reducer for this query.} \end{array}$

Database Management Systems and Their Implementation, Xu Lizhen

-					
1	R A B	S	BC	ТСА	
_	1 a		a x	x 2	
	2 b		b y	y 3	
	3 c		c z	z 4	
q = (F	R.B=S.B) (S.0	C=T.C)	(T.A=R.A), is	there full red	ucer?
0:0	T= <u>A B</u>	P: P	0: <u>B C</u>	O: O P:	C A
	2 b		b y	~~~	у З
	3 c		C Z		z 4
	AB	D D	BC		СА
0:0	Q: 3 c	P : P	0 : <u>c</u> z	Q:Q P:	z 4
Database Ma	nagement Systems and Th	eir Implementat	ion, Xu Lizhen		11

O:O	Q : s:)Pl qel	bcrii	obar bol	cobiqflkO	fk
0:0 P · P	T	P: P $O \cdot O$	О Р	Q: Q	P 0:0	Q
	U	$Q \cdot Q$	1	0.0	Q .	

Conclusion:

- For CQ: there is no full reducer in general. Even if there is, its length increases linearly along with the number of tuples of some relation in the query. (about 3(m-1))
- For TQ: the length of full reducer < n-1 (n is the number of nodes in the query graph).
- So the full reducer can not be the optimization target when execute join operation under distributed environment through.

Database Management Systems and Their Implementation, Xu Lizhen

112

÷.

--- Implementation method of join operation in R*

- I. Two basic implementation method of join operation
- Nested Loop: the extension of corresponding algorithm in centralized DBMS
 cost = (b_R + b_R/ (n_B-1) * b_S) * D₀
- Merge Scan: the extension of corresponding algorithm in centralized DBMS
 cost = (b_R + b_S) * D₀ + Cost_{sort}(R) + Cost_{sort}(S)

Database Management Systems and Their Implementation, Xu Lizhen

113

冒 II. Transmission of relations in these two methods

• **shipped whole**": The whole relation is shipped without selections.

- For I, a temporary relation is established at the destination site for further use.
- CloL) @bobi @flkalbpk qkbba pd ofkd+
 "Fetch as need": The whole relation is not shipped. The tuples needed by the remote site are sent at its request. The request message usually contains the value of join attribute.

Usually there is an index on the join attribute at the requested site.

	Nested Loop	O: shipped whole	
		I : fetch as needed	
	Merge Scan	O: shipped whole	
		I: shipped whole	
		fetch as needed	
	Shipping whole O and I to a 3^{rd} site (NL or MS)		
	 It is obvious that O should be shipped whole. IK I) fc Ffore fumbate lib) fkabu k g brefmiba 		
	, II		
Database N	fanagement Systems and Their Impleme	entation, Xu Lizhen	

2)	If SNC does not hold, it is still possible to compute some aggregate functions of global relation distributed	
	Suppose global relation: S fragments: S_1, S_2) P_n then:	
	$\begin{split} & \text{SUM}(S) = \text{SUM}(\text{SUM}(S_1), \text{SUM}(S_2)) \text{PR J } P_n)) \\ & \text{COUNT}(S) = \text{SUM}(\text{COUNT}(S_1) \text{L R } QP_n)) \\ & \text{AVG}(S) = \text{SUM}(S)/\text{COUNT}(S) \\ & \text{MIN}(S) = \text{MIN}(\text{MIN}(S_1), \text{MIN}(S_2)) \text{J } F \ P_n)) \\ & \text{MAX}(S) = \text{MAX}(\text{MAX}(S_1), \text{MAX}(S_2)) \text{J} > \text{U} P_n)) \end{split}$	

e a

The consistency between multi copies must be considered while executing update, because any data may have multi copies in DDB.

 Updating all strategy The update will fail if any one of copies is unavailable.

p --- probability of availability of a copy. n --- No. of copies

The probability of success of the update=pⁿ

 $\lim p^n = 0$

Database Management Systems and Their Implementation, Xu Lizhen

Database Management Systems and Their Implementation, Xu Lizhen

119

÷ a

2) Updating all available sites immediately and keeping the update data at spooling site for unavailable sites, which are applied to that site as soon as it is up again.

Primary copy updating strategy
 Assign a copy as primary copy. The remaining copies called secondary copies.
 Update : update P.C, then P.C broadcast the update to S.Cs at sometimes.
 P.C maybe inconsistent with S.C temporarily.
 There is no problem if the next operation is still

update. While if the next operation is a read to some S.C, then: Database Management Systems and Their Implementation, Xu Lizhen

Compare the version No. of S.C with that of P.C, if version No. are equal, read S.C directly; else:

(1) redirect the read to P.C(2) wait the update of S.C

- Snapshot
 Snapshot is a kind of copy image not follow ed the changes in DB.
- Master copy at one site, many snapshots are distributed at other sites.
- Update: master copy only.

Database Management Systems and Their Implementation, Xu Lizhen

121



- Read: master copy snapshots
 - } is indicated by users
- The snapshot can be refreshed:
 (1) periodically
 (2) forestation from the DEEDECH and th
 - (2) forced refreshing by REFRESH command
- Snapshot is suitable for the application systems in which there is less update, such as census system, etc.

Database Management Systems and Their Implementation, Xu Lizhen





n n

While recovering:

- Some transactions maybe half done, should undo them with B.I recorded in Log.
- Some transactions have finished but the results have not been written into DB in time, should redo them with A.I recorded in Log. (finish writing into DB)

It is possible to recover DB to the most recent consistent state with Log.

Database Management Systems and Their Implementation, Xu Lizhen

127



There are multi copies for every data object. Recovered with other copies while failure occurs. The system kklq b lii mpba b rpblcplj b lmv pc fir ob+

Advantages:

(1) increase reliability

(2) recovery is very easy

Problems:

(1) difficult to acquire independent failure modes in centralized database systems.

(2) waste in storage space

So this method is not suitable for Centralized DBMS.

Database Management Systems and Their Implementation, Xu Lizhen

1

- -

A transaction T is a finite sequence of actions on DB exhibiting the following effects:

- Atomic action: Nothing or All.
- ✓ Consistency preservation: consistency state of DB → another consistency state of DB.
- Isolation: concurrent transactions should run as if they are independent each other.
- Durability: The effects of a successfully completed transaction are permanently reflected in DB and recoverable even failure occurs later.

Example: transfer money s from account A to account	B
read A	
A:=A-s	
fc>9 qebk Afpmi v fkpr ccf fbkqcr ka	
/ *undo and terminate */	
else B:=B+s	
Afpmiv qokpeboljmibop	
/ *commit the update and terminate	*/
Rollback abnormal termination. (Nothing)	
Commit normal termination. (All)	
Database Management Systems and Their Implementation, Xu Lizhen	130



Recovery information (such as Log) should be stored in nonvolatile storage. The following information need to be stored in order to support recovery:

- 1) Commit list : list of TID which have been committed.
- 2) Active list : list of TID which is in progress.





The reliability demand of Log is higher than general data, often doubled.

If the interval between two dumps is too long, lack of storage space may occur because of the accumulation of Log. Solutions:

- 1) Free the space after commit. It will be impossible to recover from disk failure in this situation.
- 2) Periodically dump to tape.
- 3) Log compression:
- Alk qe sbql pqlobIld fkcloj qflk do loqba transactions
- B.I are no longer needed for committed transactions
- Changes can be consolidated, keep the new est A.I only.

Database Management Systems and Their Implementation, Xu Lizhen

6.4.1 Commit Rule A.I must be written to nonvolatile storage before commit of the transaction.

6.4.2 Log A head Rule If A.I is written to DB before commit then B.I must first written to log.

6.4.3 Recovery strategies

(1) The features of undo and redo (are idempotent) : undo(undo(undo undo(x))) = undo(x)redo(redo(redo redo(x))) = redo(x)

133

a) A.I \rightarrow DB before commit TID →active list ↓ [B.I →Log (Log Ahead Rule) LA.I →DB commit { TID →commit list

delete TID from active list

Database Management Systems and Their Implementation, Xu Lizhen

134

Check two lists for every TID while restarting after failure:

Commit	Active	
list	list	
	>	undo, delete TID from active list
✓	>	delete TID from active list
 ✓ 		nothing to do



137

Check two lists for every TID while restarting after failure:

Commit	Active	
list	list	
	~	delete TID from active list
~	√	redo, delete TID from active list
~		nothing to do

Database Management Systems and Their Implementation, Xu Lizhen

c) A.I→DB concurrently with commit TID →active list $A.I, B.I \rightarrow Log$ (Two Rules) $A.I \rightarrow DB$ (partially done)

Database Management Systems and Their Implementation, Xu Lizhen

É,

Check two lists for every TID while restarting after failure:

Commit	Active	
list	list	
	 Image: A second s	undo, delete TID from
		activelist
✓	 Image: A second s	redo, delete TID from
		activelist
✓		nothing to do

139

		redo	undo
	a)	×	~
	b)	~	×
	c)	~	~
?	d)	×	×

Database Management Systems and Their Implementation, Xu Lizhen

Database Management Systems and Their Implementation, Xu Lizhen

14

÷.,

- Keep two copies for every page of a relation
- Keep a page table (PT) for every relation
- When updating some page, produce a new page out of place, change the corresponding pointer in page table while the transaction committing, let it point to new page.

Suppose relation R has N pages, then the length of its PT is N old new

Database Management Systems and Their Implementation, Xu Lizhen



Image: Second start system or media failure. Recovery is needed before start. Warm start Start after system shutdown. Recovery is not required. Cold start Start the system from scratch. Start after a catastrophic failure or start a new DB.

Database Management Systems and Their Implementation, Xu Lizhen









- abort by itself. Once answer OK, it can only wait for the command come from the coordinator.
- If the coordinator has failure after the participates answer OK, the participates have to wait, and is in blocked state. This is the disadvantage of 2PC.

Database Management Systems and Their Implementation, Xu Lizhen

Broadcast prepare Ready / Abort

Prepare to commit

Commit/ Abort Ack

OK

1

coordinator

11

п{

III

If the sub-transaction successes and can commit, it answers Ready, or Abort

148

149

Broadcast to all participants and this MSG is recorded on the disk

If the coordinator has not any failure, phase II is wasted
 If the coordinator has failure after the participates answer OK, the participates communicate each other and check the MSG recorded on disk in phase II, and a new coordinator is elected. If the new coordinator finds the Prepare to commit MSG on any participate, it sends out Commit command, or sends out Abort command. So the blocked problem can be solved in 3PC.

icipate

partio

Database Management Systems and Their Implementation, Xu Lizhen

Database Management Systems and Their Implementation, Xu Lizhen

In multi users DBMS, permit multi transaction access the database concurrently.

- 7.1.1 Why concurrency?
- 1) Improving system utilization & response time.
- 2) Different transaction may access to different parts of database.
- 7.1.2 Problems arise from concurrent executions

Database Management S	Systems and Their	r Implementation	Xu Lizhen

151



So there maybe three kinds of conflict when transactions execute concurrently. They are write write, write read, and read write conflicts. Write write conflict must be avoided anytime. Write read and read write conflicts should be avoided generally, but they are endurable in some applications.

Database Management Systems and Their Implementation, Xu Lizhen

15

153

Definition: suppose {T₁,T₂} Q_n } is a set of transactions executing concurrently. If a schedule of {T₁,T₂} Q_n } produces the same effect on database as some serial execution of this set of transactions, then the schedule is serializable.

Problem: different schedule different equivalent serial execution different result? (yes, n!)

T _A	Τ _B	Tc	The result of this schedule
	Read R1		is the same as serial execution $T_A = T_B = T_C$, so
Read R2	Write R2	Write R1	it is serializable. The equivalent serial execution
			ISI _A I _B I _C .

- ÷.
- Rbgdc d --- sequences that indicate the chronological order in which instructions of concurrent transactions are executed
 - a schedule for a set of transactions must consist of all instructions of those transactions
 - must preserve the order in which the instructions appear in each individual transaction.







View equivalent

Conflict equivalent

- Let R and be two schedules with the same set of transactions. Rand are hd d h d if they produce the same effect on database based on the same initial execution condition.
- Conflict operations : R-W W-W. The sequence of conflict operation will affect the effect of execution.
- Non-conflicting operations: R-R Even if there are write operation, the data items operated are different. Such as $R_i(x)$ and $W_i(y)$.
- If a schedule R can be transformed into a schedule by a series of swaps of non-conflicting operations, we say that R and are b e lb d h d

Database Management Systems and Their Implementation, Xu Lizhen

Property: if schedule R and are conflict equivalent, they must be view equivalent. It is not right contrarily. Serialization can be divided into and Example 1: for the schedule s of transaction set $\{T_1, T_2, T_3\}$ $s = R_2(x)W_3(x)R_1(y)W_2 v$ $O_1(y)R_2(x)W_2(y)W_3 u : p$

- pfplkcif qpbof if | qlk b rpbp fp pbof i bub rqlk+ Example 2: $s = R_1(x)W_2(x)W_1(x)W_3(x)$
- There is no conflict equivalent schedule of s, but we can find a p e bar ib p
 - $p: O_1(x)W_1(x)W_2(x)W_3(x)$
 - Fqfpsfbt bnrfs ibkqt fæp) kap fp pbof i bub raflk) pl pfp view serialization.

se Management Systems and Their Implementation, Xu Lizhen

The test algorithm of view equivalent is a NP problem, while conflict serialization covers the most instances of serializable schedule, so the serialization we say in later parts will point to conflict serialization if without special indication.

÷.

Directed graph G = <V,E>
V --- set of vertexes, including all transactions participating in schedule.
E --- set of edges, decided through the analysis of conflict operations. If any of the following conditions is fulfilled, add an edge T_i Q into E:
R_i(x) precedes W_j(x)
W_i(x) precedes R_i(x)

W_i(x) precedes R_j(x)
 W_i(x) precedes W_i(x)

Finally, check if there is cycle in the preceding graph. If there is cycle in it, the schedule is not serializable, or it is serializable.

Database Management Systems and Their Implementation, Xu Lizhen

-

- Because there is no cycle, there must be some vertexes whose in-degree is 0. Remove these vertexes and relative edges from the preceding graph, and store these vertexes into a queue.
- Process the left graph in the same way as above, but the vertexes removed should be stored behind the existing vertexes in the queue.
- 3) Repeat 1 and 2 until all vertexes moved into the queue.

 $\begin{array}{l} \label{eq:standard} \textit{Example: for schedule s on } \{T_1,T_2,T_3,T_4\}, \mbox{suppose}: \\ s = W_3(y)R_1(x)R_2(y)W_3(x)W_2(x)W_3(z)R_4(z)W_4(x) \\ \mbox{Is it serializable? Find out the equivalent serial execution if it is.} \end{array}$

Database Management Systems and Their Implementation, Xu Lizhen







÷,

Locking method is the most basic concurrency control method. There maybe many kinds of locking protocols. 7.2.1 X locks

Only one type of lock, for both read and write.

Compatibility matrix : NL --- no lock X --- X lock



Database Management Systems and Their Implementation, Xu Lizhen



- *dh h* In a transaction, if all locks precede all unlocks, then the transaction is called two phase transaction. This restriction is called two phase locking protocol.
- *d*:*h hh* 1 In a transaction, if it first acquires a lock on the object before operating on it, it is called wellformed.

 Sgd d If Sis any schedule of well- formed and two phase transactions, then Sis serializable. (proving is on p151) 	Growing phase Shrinking phase Unlock A Unlock A Unlock A Unlock C 2PL	Lock A Lock B Unlock A Unlock B Lock C Unlock C Unlock C
Database Management Systems and Their Implementation, Xe	u Lizhen	



- 1) Well-formed + 2PL : serializable
- Well-formed + 2PL + unlock update at EOT: serializable and recoverable. (without domino phenomena)
- 3) Well-formed + 2PL + holding all locks to EOT: strict two phase locking transaction.

7.2.2 (S,X)	locks	/
Slock	if read access is	NL
V I a als	intended.	s
X IOCK	is intended.	Х



U lock --- update lock. For an update access the NL s U Х transaction first acquires Υ Υ Υ Υ NL a U-lock and then promote it to X-lock. S Υ Υ Y Ν Purpose: shorten the time U Y Ν Ν Y of exclusion, so as to Х Y Ν Ν Ν boost concurrency degree, and reduce deadlock. Х (S,X) (S,U,X) Concurrency degree Overhead



Database Management Systems and Their Implementation, Xu Lizhen

Database Management Systems and Their Implementation, Xu Lizhen

- Timeout: If a transaction waits for some specified time then deadlock is assumed and the transaction should be aborted.
- ²⁾ Detect deadlock by wait-for graph G=<V,E> V : set of transactions {T_i| T_i is a transaction in DBS f.)) k
 - $\mathsf{E}:\{{<}\mathsf{T}_i,\mathsf{T}_j{>}|~\mathsf{T}_i~w~aits~for~\mathsf{T}_j~f~g$
- If there is cycle in the graph, the deadlock occurs.
- When to detect?
- 1) whenever one transaction waits.
- 2) periodically

÷.

- What to do when detected?
- 1) Mfh sfqfj vlrkdbpq)jfkfjrj loq lpq)
- 2) Abort the victim and release its locks and resources
- 3) Grant a waiter
- 4) Restart the victim (automatically or manually)
- 7.3.2 Deadlock avoidance
- 1) Requesting all locks at initial time of transaction.
- 2) Requesting locks in a specified order of resource.
- 3) Abort once conflicted.
- 4) Transaction Retry

Every transaction is uniquely time stamped. If T_A requires a lock on a data object that is already locked by T_B, one of the following methods is used:

- Wait-die: T_A waits if it is older than T_B, otherwise it afbp) ftb-tfqfp 1 cdpa ka rdj of iiv obcpfba with original timestamp.
- b) Wound-wait: T_A waits if it is younger than T_B , l qc bot fpb fq t l r ka Q_B , i.e. T_B is aborted and automatically retried with original timestamp.

In above, both have only one direction wait, either liabovlrkdbolovlrkdboliabo+Eqfpfjmlppfib to occur wait in cycle, so the dead lock is avoided.

Database Management Systems and Their Implementation, Xu Lizhen



- 7.4.1 Locking in multi granularities
 - To reduce the overhead of locking, the lock unit should be the bigger, the better; To boost the concurrency degree of transactions, the lock unit should be the smaller, the better.
- In large scale DBMS, the lock unit is divided into several levels: DB File Record Field
- In this situation, if a transaction acquires a lock on a data object of some level then it acquires implicitly the same lock on each descendant of that data object.
- So, there are two kinds of locks in multi granularity lock method: Explicit lock
- Implicit lock



÷,

- Transactions access concurrently not only the data in DB, but also the indexes on these data, and apply operations on indexes, such as search, insert, delete, etc. So indexes also need concurrency control while multi granularity locking is supported.
- Database Management Systems and Their Implementation, Xu Lizhen





e în

- While traversing, apply Slock on root first, then apply Slock on the child node selected. Once get Slock on the child, the Slock $l k m obkq \ k$ bobib pba) b r pbqp sloptkd k qdl h+ Search like this until arrive leaf node. After traversing, only S lock on wanted leaf is left. Keep this Slock till EOT.
- While inserting new index item, traverse first, find the leaf node where the new item should be inserted in. Apply X lock on this leaf node.
 - If it is not full, insert directly.
 - If it is full, split node according to the rule of B+ tree. While splitting, besides the original leaf, the new leaf and their parent should add X lock. If parent is also full, the splitting will continue.
 - In every splitting, must apply X lock on each node to be changed. These X locks can be released when the changes are finished.
 - After the all inserting process is completed, the X lock on leaf node is changed to Slock and kept to EOT.

Database Management Systems and Their Implementation, Xu Lizhen

179

³ While deleting an index item from the tree, the procedure is similar as inserting. Deleting may cause the combination of nodes in B+ tree. The node changed must be X locked first and X lock released after finishing change. The X lock on leaf node is also changed to Slock and kept to EOT.

Database Management Systems and Their Implementation, Xu Lizhen

The that the DB is a fixed collection of objects is not true when multi granularity locking is permitted. Then even Strict 2PL will not assure serializability:

- T1 locks all pages containing sailor records with h f = 1, and finds <u>oldest</u> sailor (say, fd = 71).
- Next, T2 inserts a new sailor; h f = 1, fd = 96.
- T2 also deletes oldest sailor with rating = 2 (and, say, fd = 80), and commits.
- T1 now locks all pages containing sailor records with h f = 2, and finds <u>oldest</u> (say, fd = 63).

l lkp/fpolpkqA pqolptebobQ fp loobq nent Systems and Their Implementation, Xu Lizhen



- T1 implicitly assumes that it has locked the set of all sailor records with h f = 1.
 - Assumption only holds if no sailor records are added while T1 is executing!
 - Need some mechanism to enforce this assumption. (Index locking and predicate locking)
- Example shows that conflict serializability guarantees serializability only if the set of objects is fixed!
- Fcqebpvpqbj alk qprmml oqj riqf do kri ofqvil hfkd) or even if support multi granularity locking, the query need to scan the whole table and add Slock on the table, then there is not this problem. For example : select s#, average(grade) from SC group by s#;

ase Management Systems and Their Implementation, Xu Lizhen



÷.,

- Grant lock on all records that satisfy some logical predicate, e.g. fd=1)
- Index locking is a special case of predicate locking for which an index supports efficient implementation of the predicate lock.
 - What is the predicate in the sailor example?
- In general, predicate locking has a lot of locking overhead. It is almost impossible to realize it.

184

é,

 Support for isolation level of transaction is added from SQL-92. Each transaction has an access mode, a diagnostics size, and an isolation level.
 SET TRANSACTION statement

Isolation Level	Possible result				
	Dirty Unrep Read Re	Unrepeatable Read	Phantom Problem	Lock demand	
Read Uncommitted	Maybe	Maybe	Maybe	Nolock when read; Xlock when write keep until EOT	
Read Committed	No	Maybe	Maybe	Slock when read, release after read; X lock when write, keep until EOT	
Repeatable Reads	No	No	Maybe	According to Strict 2PL	
Serializable	No	No	No	Strict 2PL and keep Slock on leaf of index until EOT	

Database Management Systems and Their Implementation, Xu Lizhen

Database Management Systems and Their Implementation, Xu Lizhen



SET TRANSACTION READ ONLY ISOLATION LEVEL REPEATABLE READ;

SET TRANSACTION ISOLATION LEVEL { READ COMMITTED | READ UNCOMMITTED | REPEATABLE READ | SERIALIZABLE }

- -
 - Lock granularity: object is the smallest lock granularity in OODB generally. DB Class Object
 - 2) Single level locking: lock the object operated with Sor X lock directly. Suitable for the OODBMS faced to CAD application, etc. not suitable for the application occasion in which association queries are often.
 - ³⁾ multi granularity lock: use S, X, IS, IX, SIX locks introduced in last section. It is a typical application of multi granularity lock. But in this situation, the class level lock can only lock the objects directly belong to this class, can not include the objects in its child classes. So it is not suitable for cascade queries on inheriting tree or schema update.
 - 4) Complex multi granularity lock: two class hierarchy locks are added.

Database Management Systems and Their Implementation, Xu Lizhen

187



RL --- add a Slock at a class and all of its child classes
 WL --- add a X lock at a class and all of its child classes

	NL	IS	IX	S	SIX	Х	RL	WL
NL	Y	Y	Y	Y	Y	Υ	Y	Ν
IS	Y	Y	Y	Y	Y	Ν	Y	Ν
IX	Y	Y	Y	Ν	N	Ν	Ν	Ν
s	Y	Υ	Ν	Υ	N	Ν	Υ	Ν
SIX	Y	Y	Ν	Ν	N	Ν	Ν	Ν
Х	Y	Ν	Ν	Ν	N	Ν	Ν	Ν
RL	Y	Y	Ν	Y	Ν	Ν	Y	Ν
WL	Y	Ν	Ν	Ν	Ν	Ν	Ν	Ν

Database Management Systems and Their Implementation, Xu Lizhen



Database Management Systems and Their Implementation, Xu Lizhen

нîн

- T.S---> krj bodbkbodpa v ljmrdpopfkdpok i clock in chronological order.
- 2 T.Sfor a transaction --- the current T.S when the transaction initials.
- T.Sfor an data object:
 1) Read time () --- highest T.Spossessed by any transaction to have read the object.
 2) Write time () --- highest T.Spossessed by any transaction to have written the object.
- 4. The key idea of T.S method is that the system will enforce the concurrent transactions to execute in the schedule equivalent with the serial execution according to T.S order.

Database Management Systems and Their Implementation, Xu Lizhen

Let R --- a data object with T.Str and tw. T --- a transaction with T.St.
Transaction T reads R: read R if (t >= tw) then /* OK */ tr = Max (tr, t) else /* a transaction younger than T has already write R ahead of T, conflict */ restart T with a new T.S

Database Management Systems and Their Implementation, Xu Lizhen

×.					
Transaction T	Fwrites R:				
if $(t \ge tr)$					
then if (t	t >= tw)				
	then / * OK */				
	writeR	No other reads			
	tw = t	_ <u></u>			
	else / * tr <= t < tw */	tr t tw			
	do nothing				
else	/ * a transaction yo read R ahead of	unger than T has already T, conflict */			
restart T with a new T.S					
Database Management Systems and Th	heir Implementation, Xu Lizhen	192			

- - t. Compared with lock method, the most obvious advantage is that there is no dead lock, because of no wait.
 - $_{\rm 2}$ Disadvantage: every transaction and every data object has T.S, and every operation need to update tr or tw, so the overhead of the system is high.
 - 3. Solution:
 - Enlarge the granularity of data object added T.S. (Low concurrency degree)
 - T.Sof data object are not actually stored in nonvolatile storage but in main memory and preserved for a specified time and the T.Sof data objects whose T.Sis not in main memory are assumed to be zero.



The key idea of optimistic method is that it supposes there is rare conflict when concurrent transactions execute. It doesn't take any check while transactions are executing. The updates are not written into DB directly but stored in main memory, and check if the schedule of the transaction is serializable when a transaction finishes. If it is serializable, write the updating copies in main memory into DB; Otherwise, abort the transaction and try again.

The lock method and time stamp method introduced lsb ob iiba mbppfj fppf j bppla +

Database Management	Avetoms and	Their Imp	lementation	Xulizhen
Database in anagement	by bronno cento	THOM MIP	ranonunon,	

194

d c g d read data from database and execute every kind of processing, but update operations only form update copies in memory.

hc dg d check if the schedule of the transaction is serializable.

V hd g d if pass the check successfully, write the update copies in memory into DB and commit the transaction; Or throw away the update copies in memory and abort the transaction.

-

- 1. Read set of each transaction
- 2. Write set of each transaction
- 3. The start and end time of each phase of each transaction

Database Management Quaterns and Their Implementation	Vullahoo

196

-

When transaction T_i ends, only need to check if there is conflict among T_i and the transactions which have committed and other transactions which are also in checking phase. The transactions which are in read phase don't need to be considered. Suppose T_j is any transaction which has committed or is being checked, T_i passes the check if it fulfills one of the following conditions for all T_i :

1. T_j had finished write phase when T_i began read phase, $T_j = Q$

- 2. The intersection of T_i's read set and T_j's write set is empty, and T_i began write phase after T_j finifhed write phase.
- 3. Both Ti's read set and write set don't intersect with Ti's write set.

4. There is not any access conflict between T_i and T_j.

Database Management Systems and Their Implementation, Xu Lizhen

197



The concurrency control in DDBMS is the same as that in centralized DBMS, demand concurrent transactions to be scheduled serializably. The problems in DDBMS are:

- Multi_copy
- Communication overhead
- 7.11.1 write lock all, read lock one
- Read R --- S_lock on any copy of R
- Write R---X_lock all copies of R
- Hold the locks to EOT Communication overhead: suppose n---No. of copies

Database Management Systems and Their Implementation, Xu Lizhen



Write: (n+1)/ 2 lock MSG	Read: (n+1)/ 2 lock MSG
(n+1)/ 2 lock grants	(n+1)/ 2 lock grants
n update MSG	1 read MSG
ACK	n+1
 3n+1 In 7.11.1, if there are two lock for update, maybe even to be an are the set of the set of	transaction compete the X sach get a part, but no one

Database Management Systems and Their Implementation, Xu Lizhen

20

.

- Write R---X_lock on k copies of R, k>n/ 2
- Read R ---S_lock on n-k+1 copies of R
- Hold the locks to EOT
- For read-write conflict: k+(n-k+1)=n+1>n, so the conflict can be found on at least one copy.
- For write-write conflict: 2k>n, so it is also sure that the conflict can be detected.
- The above two methods are the special situations of it: $\sqrt{7.11.1}$ is k=n; 7.11.2 is k=(n+1)/2
- k can be changed between (n+1)/ 2 ~ n, the bigger of k, the better for read operations.

R---data object

Assign the lock responsibility for locking R to a given site. This site is called primary site of R. Communication overhead :

Database Management Systems and Their Implementation, Xu Lizhen

Nrite: 1 lock MSG	Read: 1 lock MSG
1 lock grants	1 lock grants ⊃
n update MSG	1 read MSG 了
n ACK	2
2n+1	-

It is efficient but liable to fail, so there are many variations. It is often used together with primary copy updating strategy (see 5.9).

202



Wait for lock

Site B T_{1B} ←

Suppose both T1 and T2 are distributed transactions, and have two sub transactions on site A and B respectively. T_{1A} and T_{1B}must commit simultaneously T_{2A} and T_{2B} must commit simultaneously

The above shows a global dead lock. How to find out this kind of dead lock?

 $\mathsf{T}_{2\mathsf{B}}$

Global wait-for graph: add EXT nodes based on general wait-for graph. If transaction T is a distributed transaction, and has sub transactions on other sites, and T is the head of wait-for chain of current site, add BUQ Qfkql qebdome fcQfpqebqfilct fqforchain of rcobkqpfqb) aa Q BUQfkql qebdome+

Database Management Systems and Their Implementation, Xu Lizhen

FclkpljbpfcpepBUQQQ Q, BUQ

- 1) ebhlæbopfdøpfce pBUQQ_kQ Q BUQ
- ²⁾ if $T_x = T_i$: global deadlock is detected.
- if T_x Q: merge two wait-for graphs:
- BUQ Q Q Q, BUQ Q, Q
- 3) Repeat step 1 and 2, check if T_x will result in global dead lock like T_k when the condition in 2 is true. If wait-for graph on all sites have been check like above and no global cycle is found, no global dead lock occur.

```
Database Management Systems and Their Implementation, Xu Lizhen
```



Database Management Systems and Their Implementation, Xu Lizhen