

Advanced Database Systems

Question Bank

1.What are parallel systems? Compare parallel database architectures in detail with diagrams.

→What are parallel systems

- Parallel systems improve processing and I/O speeds by using multiple processors and disks in parallel
- Parallel machines are becoming increasingly common, making the study of parallel database systems correspondingly more important
- In parallel processing, many operations are performed simultaneously, as opposed to serial processing, in which the computational steps are performed sequentially
- A coarse-grain parallel machine consists of a small number of powerful processors; a massively parallel or fine-grain parallel machine uses thousands of smaller processors

01.Shared Memory

- All the processors share a common memory.
- In a shared-memory architecture, the processors and disks have access to a common memory, typically via a bus or through an interconnection network
- The benefit of shared memory is extremely efficient communication between processors—data in shared memory
- A processor can send messages to other processors much faster by using memory writes (which usually take less than a microsecond) than by sending a message through a communication mechanism
- Architecture is not scalable beyond 32 or 64 processors
- Adding more processors does not help after a point, since the processors will spend most of their time waiting for their turn on the bus to access memory.
- Shared-memory architectures usually have large memory caches at each processor

2.Shared Disk

- All the processors share a common set of disks
- In the shared-disk model, all processors can access all disks directly via an interconnection network, but the processors have private memories.

-There is an advantages of shared disk over the shared memory first is each processor has its own memory, the memory bus is not a bottleneck.*Second advantage Is* it offers a cheap way to provide a degree of fault tolerance

-Shared-disk systems can scale to a somewhat larger number of processors, but communication across processors is slower

3. Shared Nothing

-The processors share neither a common memory nor common disk

-In a shared-nothing system, each node of the machine consists of a processor, memory, and one or more disks

-The processors at one node may communicate with another processor at another node by a high-speed interconnection network

-shared-nothing architectures are more scalable and can easily support a large number of processors.

-Shared-nothing multiprocessors can be scaled up to thousands of processors without interference

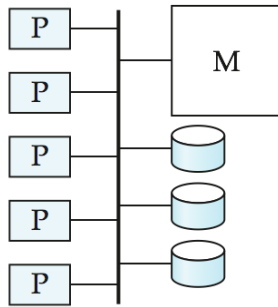
-The main drawbacks of shared-nothing systems are the costs of communication

4. Hierarchical

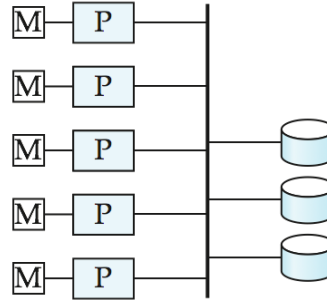
-This model is a hybrid of the preceding three architectures

-The hierarchical architecture combines the characteristics of shared-memory, shared-disk, and shared-nothing architectures

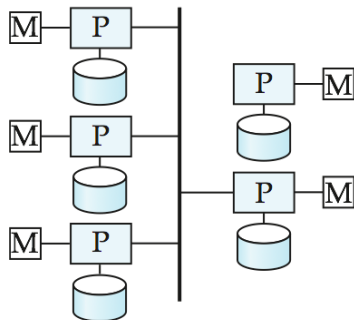
-Top level is a shared-nothing architecture – nodes connected by an interconnection network, and do not share disks or memory with each other



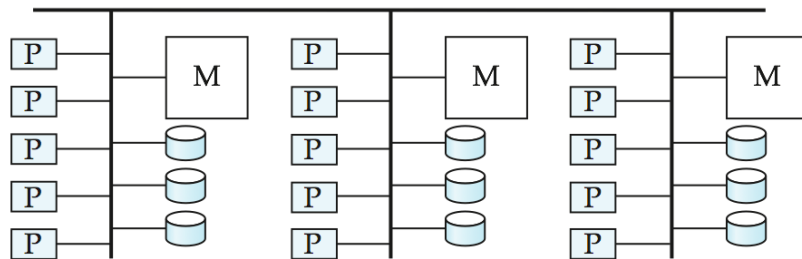
(a) shared memory



(b) shared disk



(c) shared nothing



(d) hierarchical

2.Explain data partitioning techniques used in parallel databases. Also give comparison between data partitioning techniques.

-In its simplest form,I/O parallelism refers to reducing the time required to retrieve relations from disk by partitioning the relations over multiple disks

-Horizontal partitioning is the most common form of data partitioning technique in a parallel database

Several partitioning Techniques are

1. Round-robin Partitioning
2. Hash partitioning
3. Range partitioning

1.Round-robin Partitioning

-This strategy scans the relation in any order and sends the i th tuple to disk number $D_i \bmod n$

- Round-robin scheme ensures that each disk has approximately the same number of tuples as the others.

-The scheme is ideally suited for applications that wish to read the entire relation sequentially for each query. With this scheme, both point queries and range queries are complicated to process, since each of the n disks must be used for the search.

2.Hash partitioning

-This declustering strategy designates one or more attributes from the given relation's schema as the partitioning attributes

- A hash function is chosen whose range is $\{0, 1, \dots, n - 1\}$

- If the hash function returns i , then the tuple is placed on disk D_i

-This scheme is best suited for point queries based on the partitioning attribute. For example, if a relation is partitioned on the telephone number attribute, then we can answer the query "Find the record of the employee with telephone number = 555-3333" by applying the partitioning hash function to 555-3333 and then searching that disk. Directing a query to a single disk saves the start-up cost of initiating a query on multiple disks, and leaves the other disks free to process other queries

3.Range partitioning

- This strategy distributes tuples by assigning contiguous attribute-value ranges to each disk.

- It chooses a partitioning attribute, A , and a partitioning vector $[v_0, v_1, \dots, v_{n-2}]$, such that, if $i < j$, then $v_i < v_j$. The relation is partitioned as follows: Consider a tuple t such that $t[A] = x$. If $x < v_0$, then t goes on disk D_0 . If $x \geq v_{n-2}$, then t goes on disk D_{n-1} . If $v_i \leq x < v_{i+1}$, then t goes on disk D_{i+1} . For example, range partitioning with three disks numbered 0, 1, and 2 may assign tuples with values less than 5 to disk 0, values between 5 and 40 to disk 1, and values greater than 40 to disk 2.

-This scheme is well suited for point and range queries on the partitioning attribute. For point queries, we can consult the partitioning vector to locate the disk where the tuple resides. For range queries, we consult the partitioning vector to find the range of disks on which the tuples may reside. In both cases, the search narrows to exactly those disks that might have any tuples of interest

3.Explain two phase commit (2PC) protocol in brief. Also explain how 2PC protocol handles failure of a participating site and failure of a coordinator.

-Consider a transaction T initiated at site S_i , where the transaction coordinator is C_i

The Commit Protocol

When T completes its execution—that is, when all the sites at which T has executed inform C_i that T has completed— C_i starts the 2PC protocol.

• **Phase 1.** Ci adds the record <prepare T> to the log, and forces the log onto stable storage. It then sends a prepare T message to all sites at which T executed. On receiving such a message, the transaction manager at that site determines whether it is willing to commit its portion of T. If the answer is no, it adds a record <no T> to the log, and then responds by sending an abort T message to Ci . If the answer is yes, it adds a record <ready T> to the log, and forces the log (with all the log records corresponding to T) onto stable storage. The transaction manager then replies with a ready T message to Ci .

• **Phase 2.** When Ci receives responses to the prepare T message from all the sites, or when a prespecified interval of time has elapsed since the prepare T message was sent out, Ci can determine whether the transaction T can be committed or aborted. Transaction T can be committed if Ci received a ready T message from all the participating sites. Otherwise, transaction T must be aborted. Depending on the verdict, either a record <commit T> or a record <abort T> is added to the log and the log is forced onto stable storage. At this point, the fate of the transaction has been sealed. Following this point, the coordinator sends either a commit T or an abort T message to all participating sites. When a site receives that message, it records the message in the log

2 Handling of Failures

The 2PC protocol responds in different ways to various types of failures:

• **Failure of a participating site.** If the coordinator Ci detects that a site has failed, it takes these actions: If the site fails before responding with a ready T message to Ci , the coordinator assumes that it responded with an abort T message. If the site fails after the coordinator has received the ready T message from the site, the coordinator executes the rest of the commit protocol in the normal fashion, ignoring the failure of the site.

When a participating site Sk recovers from a failure, it must examine its log to determine the fate of those transactions that were in the midst of execution

834 Chapter 19 Distributed Databases

when the failure occurred. Let T be one such transaction. We consider each of the possible cases:

- The log contains a <commit T> record. In this case, the site executes redo(T).
- The log contains an <abort T> record. In this case, the site executes undo(T).
- The log contains a <ready T> record. In this case, the site must consult C_i to determine the fate of T. If C_i is up, it notifies S_k regarding whether T committed or aborted. In the former case, it executes redo(T); in the latter case, it executes undo(T). If C_i is down, S_k must try to find the fate of T from other sites. It does so by sending a querystatus T message to all the sites in the system. On receiving such a message, a site must consult its log to determine whether T has executed there, and if T has, whether T committed or aborted. It then notifies S_k about this outcome. If no site has the appropriate information (that is, whether T committed or aborted), then S_k can neither abort nor commit T. The decision concerning T is postponed until S_k can obtain the needed information. Thus, S_k must periodically resend the querystatus message to the other sites. It continues to do so until a site that contains the needed information recovers. Note that the site at which C_i resides always has the needed information.
- The log contains no control records (abort, commit, ready) concerning T. Thus, we know that S_k failed before responding to the prepare T message from C_i . Since the failure of S_k precludes the sending of such a response, by our algorithm C_i must abort T. Hence, S_k must execute undo(T).
- **Failure of the coordinator.** If the coordinator fails in the midst of the execution of the commit protocol for transaction T, then the participating sites must decide the fate of T. We shall see that, in certain cases, the participating sites cannot decide whether to commit or abort T, and therefore these sites must wait for the recovery of the failed coordinator.
- If an active site contains a <commit T> record in its log, then T must be committed.
- If an active site contains an <abort T> record in its log, then T must be aborted.

- If some active site does not contain a <ready T> record in its log, then the failed coordinator Ci cannot have decided to commit T, because a site that does not have a <ready T> record in its log cannot have sent a ready T message to Ci. However, the coordinator may have decided to abort T, but not to commit T. Rather than wait for Ci to recover, it is preferable to abort T.
- If none of the preceding cases holds, then all active sites must have a <ready T> record in their logs, but no additional control records (such as <abort T> or <commit T>). Since the coordinator has failed, it is impossible to determine whether a decision has been made, and if one has, what that decision is, until the coordinator recovers. Thus, the active sites must wait for Ci to recover. Since the fate of T remains in doubt, T may continue to hold system resources. For example, if locking is used, T may hold locks on data at active sites. Such a situation is undesirable, because it may be hours or days before Ci is again active. During this time, other transactions may be forced to wait for T. As a result, data items may be unavailable not only on the failed site (Ci), but on active sites as well. This situation is called the blocking problem, because T is blocked pending the recovery of site Ci

4.What are parallel systems? Explain Speedup and Scaleup in parallel systems with the help of diagram.

5.What are the different types of distributed database system? Explain Semijoin strategy used in distributed query processing.

In a homogeneous distributed database system, all sites have identical database management system software, are aware of one another, and agree to cooperate in processing users' requests. In such a system, local sites surrender a portion of their autonomy in terms of their right to change schemas or database-management

Distributed Databases

branch(branch name, branch city, assets)

account (account number, branch name, balance)

depositor (customer name, account number)

system software. That software must also cooperate with other sites in exchanging information about transactions, to make transaction processing possible across multiple sites.

In contrast, in a heterogeneous distributed database, different sites may use different schemas, and different database-management system software. The sites may not be aware of one another, and they may provide only limited facilities for cooperation in transaction processing. The differences in schemas are often a major problem for query processing, while the divergence in software becomes a hindrance for processing transactions that access multiple site

6.Explain how to store data in distributed database systems.

There are two approaches to storing this relation in the distributed database:

- Replication. The system maintains several identical replicas (copies) of the relation, and stores each replica at a different site. The alternative to replication is to store only one copy of relation r.
- Fragmentation. The system partitions the relation into several fragments, and stores each fragment at a different site

1 Data Replication

If relation r is replicated, a copy of relation r is stored in two or more sites. In the most extreme case, we have full replication, in which a copy is stored in every site in the system

There are a number of advantages and disadvantages to replication

a)Availability- If one of the sites containing relation r fails, then the relation r can be found in another site

b) Increased parallelism- In the case where the majority of accesses to the relation r result in only the reading of the relation, then several sites can process queries involving r in parallel

c) Increased overhead on update- The system must ensure that all replicas of a relation r are consistent; otherwise, erroneous computations may result

2. Data Fragmentation

If relation r is fragmented, r is divided into a number of fragments r_1, r_2, \dots, r_n . These fragments contain sufficient information to allow reconstruction of the original relation r . There are two different schemes for fragmenting a relation: horizontal fragmentation and vertical fragmentation

-In horizontal fragmentation, a relation r is partitioned into a number of subsets, r_1, r_2, \dots, r_n .

Horizontal fragmentation is usually used to keep tuples at the sites where they are used the most, to minimize data transfer. In general, a horizontal fragment can be defined as a selection on the global relation r . That is, we use a predicate P_i to construct fragment r_i :

$$r_i = \sigma_{P_i}(r) \text{ (Draw symbol)}$$

We reconstruct the relation r by taking the union of all fragments; that is:

$$r = r_1 \cup r_2 \cup \dots \cup r_n$$

Vertical fragmentation of $r(R)$ involves the definition of several subsets of attributes R_1, R_2, \dots, R_n of the schema R so that:

$$R = R_1 \cup R_2 \cup \dots \cup R_n$$

Each fragment r_i of r is defined by:

$$r_i = \sigma_{P_i}(r) \text{ (Draw symbol)}$$

The fragmentation should be done in such a way that we can reconstruct relation r from the fragments by taking the natural join:

$$r = r_1 \star r_2 \star r_3 \star \dots \star r_n$$

7.What is stored procedure in PL/SQL? Give its advantages. Explain in detail, syntax to create stored procedure in PL/SQL with example.

Def

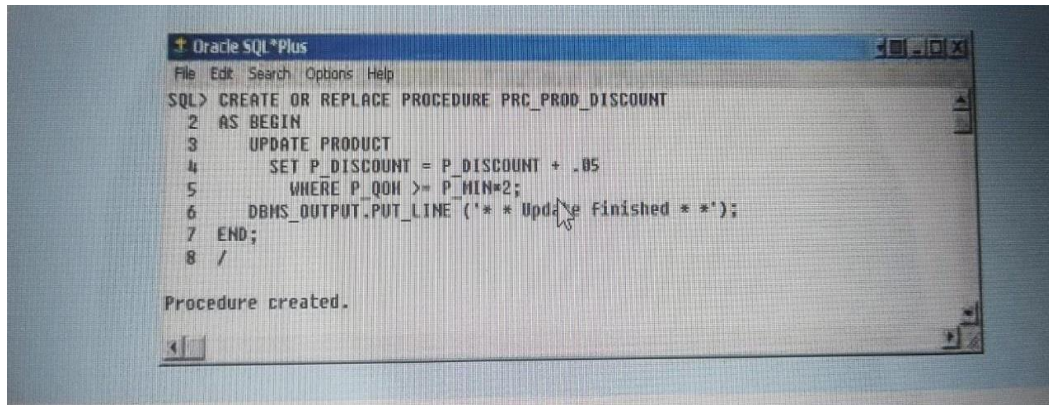
A stored procedure is a named collection of procedural and SQL statements. Just like database triggers, stored procedures are stored in the database.

There are two clear advantages to the use of stored procedures:

- Stored procedures substantially reduce network traffic and increase performance. Because the procedure is stored at the server, there is no transmission of individual SQL statements over the network. The use of stored procedures improves system performance because all transactions are executed locally on the RDBMS, so each SQL statement does not have to travel over the network.
- Stored procedures help reduce code duplication by means of code isolation and code sharing (creating unique PL/SQL modules that are called by application programs), thereby minimizing the chance of errors and the cost of application development and maintenance.

To create a stored procedure, you use the following syntax:

```
CREATE OR REPLACE PROCEDURE procedure_name [(argument [IN/OUT] data-  
type, )]  
[IS/AS]  
[variable_namedata type[:=initial_value] ]  
BEGIN  
PL/SQL or SQL statements;  
...  
END;
```

A screenshot of the Oracle SQL*Plus command-line interface. The window title is "Oracle SQL*Plus". The menu bar includes "File", "Edit", "Search", "Options", and "Help". The command prompt shows the following SQL code:

```
SQL> CREATE OR REPLACE PROCEDURE PRC_PROD_DISCOUNT
2 AS BEGIN
3   UPDATE PRODUCT
4     SET P_DISCOUNT = P_DISCOUNT + .05
5     WHERE P_QOH >= P_MIN*2;
6   DBMS_OUTPUT.PUT_LINE ('* * Update Finished * *');
7 END;
8 /
```

The output below the code is "Procedure created.".

8. Describe oracle sequence. Explain sequence in Oracle with syntax and example.

Oracle does not support the AutoNumber data type or the Identity column property. Instead, you can use a “sequence” to assign values to a column on a table. But an Oracle sequence is very different from the Access AutoNumber data type and deserves close scrutiny:

Oracle sequences are an independent object in the database. (Sequences are not a data type.)

Oracle sequences have a name and can be used anywhere a value is expected.

Oracle sequences are not tied to a table or a column.

Oracle sequences generate a numeric value that can be assigned to any column in any table.

The table attribute to which you assigned a value based on a sequence can be edited and modified.

An Oracle sequence can be created and deleted anytime.

The basic syntax to create a sequence in Oracle is:

```
CREATE SEQUENCE name [START WITH n] [INCREMENT BY n] [CACHE | NOCACHE]
```

where:

name is the name of the sequence.

n is an integer value that can be positive or negative.

START WITH specifies the initial sequence value. (The default value is 1.)

INCREMENT BY determines the value by which the sequence is incremented. (The default increment value

The CACHE or NOCACHE clause indicates whether Oracle will preallocate sequence numbers in memory.

For example, you could create a sequence to automatically assign values to the customer code each time a new customer is added and create another sequence to automatically assign values to the invoice number each time a new

invoice is added. The SQL code to accomplish those tasks is:

```
CREATE SEQUENCE CUS_CODE_SEQ START WITH 20010 NOCACHE;  
CREATE SEQUENCE INV_NUMBER_SEQ START WITH 4010 NOCACHE;
```

9.What is cursor in PL/SQL? Describe cursor attributes in detail. Write a PLSQL cursor to update salary of employee if salary is greater than 10000 then increment it by 6% otherwise by 5%.

Cursor:

- Special construct used to hold data rows returned by a SQL query
- A cursor is a temporary work area created in the system memory when a SQL statement is executed. A cursor contains information on a select statement and the rows of data accessed by it. Therefore, cursors are used as to speed the processing time of queries in large databases

ATTRIBUTE	DESCRIPTION
%ROWCOUNT	Returns the number of rows fetched so far. If the cursor is not OPEN, it returns an error. If no FETCH has been done but the cursor is OPEN, it returns 0.
%FOUND	Returns TRUE if the last FETCH returned a row, and FALSE if not. If the cursor is not OPEN, it returns an error. If no FETCH has been done, it contains NULL.
%NOTFOUND	Returns TRUE if the last FETCH did not return any row, and FALSE if it did. If the cursor is not OPEN, it returns an error. If no FETCH has been done, it contains NULL.
%ISOPEN	Returns TRUE if the cursor is open (ready for processing) or FALSE if the cursor is closed. Remember, before you can use a cursor, you must open it.

Cengage Learning © 2015

```

declare
empid emp.id%type;
empname emp.name%type;
empcity emp.city%type;
empsalary emp.salary%type;
inc number;
eid number:=&eid;
cursor s2 is select id,name,city,salary from emp ;
begin
open s2;
loop
fetch s2 into empid,empname,empcity,empsalary;
if(empsalary>10000)
then
inc:=1.06;
else
inc:=1.05;
end if;
update emp set salary=empsalary*inc where id=eid;
exit when s2%notfound;
end loop;

```

end;

10.Explain in detail, syntax to create function in PL/SQL. Write PL/SQL function to find factorial of given number.

Using programmable or procedural SQL, you can also create your own stored functions. Stored procedures and functions are very similar. A stored function is basically a named group of procedural and SQL statements that returns a value (indicated by a RETURN statement in its program code). To create a function, you use the following

CREATE FUNCTION function_name (argument IN data-type,) RETURN data-type [IS]

BEGIN

PL/SQL statements;

...

RETURN (value or expression);

END;

Stored functions can be invoked only from within stored procedures or triggers and cannot be invoked from SQL statements (unless the function follows some very specific compliance rules).

Remember not to confuse built-in SQL functions (such as MIN, MAX, and AVG) with stored functions

Program to for factorial

create or replace function factorial

(num1 in number)

return number

is

fact1 number:=1;

begin

```

    for i in 1..num1 loop
        fact1:=fact1*i;
    end loop;
    return( fact1);
end factorial;
//function call
declare
num1 number:=6;
f number(5);
begin
f:=factorial(num1);
    dbms_output.put_line(' The factorial is::::'||f);
end;

```

11.What is trigger? Explain in detail, syntax to create trigger in oracle with example.

A trigger is procedural SQL code that is automatically invoked by the RDBMS upon the occurrence of a given data

manipulation event. It is useful to remember that:

A trigger is invoked before or after a data row is inserted, updated, or deleted.

A trigger is associated with a database table.

Each database table may have one or more triggers.

A trigger is executed as part of the transaction that triggered it.

Syntax

```

CREATE OR REPLACE TRIGGER trigger_name
[BEFORE / AFTER] [DELETE / INSERT / UPDATE OF column_name] ON
table_name
[FOR EACH ROW]
[DECLARE]
[variable_namedata type[:=initial_value] ]
BEGIN

```

PL/SQL instructions;

.....

END;

As you can see, a trigger definition contains the following parts:

The triggering timing: BEFORE or AFTER. This timing indicates when the trigger's PL/SQL code executes;

in this case, before or after the triggering statement is completed.

The triggering event: the statement that causes the trigger to execute (INSERT, UPDATE, or DELETE).

The triggering level: There are two types of triggers: statement-level triggers and row-level triggers.

- A statement-level trigger is assumed if you omit the FOR EACH ROW keywords. This type of trigger

is executed once, before or after the triggering statement is completed. This is the default case.

- A row-level trigger requires use of the FOR EACH ROW keywords. This type of trigger is executed once

for each row affected by the triggering statement. (In other words, if you update 10 rows, the trigger

executes 10 times.)

The triggering action: The PL/SQL code enclosed between the BEGIN and END keywords. Each statement

inside the PL/SQL code must end with a semicolon “;”.

Example

FIGURE
8.31

Creating the TRG_PRODUCT_REORDER trigger

```
Oracle SQL*Plus
File Edit Search Options Help
SQL> CREATE OR REPLACE TRIGGER TRG_PRODUCT_REORDER
2 AFTER INSERT OR UPDATE OF P_QOH ON PRODUCT
3 BEGIN
4     UPDATE PRODUCT
5         SET P_REORDER = 1
6         WHERE P_QOH <= P_MIN;
7 END;
8 /
```

Trigger created.