page 8kb=8192bytes

page stores records

A group of 8 adjacent pages is called an extent

locks: sql server --> resources

--SQL Server locks resources using different lock modes

that determine how the resources can be accessed by concurrent transactionsh

-- a. if there is read operation on data, then sql server assings shared lock(S) on that resources

in this case, other transaction can read the data but no modification alllowed

-- b. if .........dml ...........................................exclusive lock(X)

in this case, other transaction can not do anything on resources

pabatch01

Pcsglobaltech01

--update lock prevent locking

--login server scope

--user database scope

--a login in can have multiple users

--differentl databased

--system

--user defined database

--1 default instance 49 named instances

--cursor

-5system databse, four, 1 hidden

-msdb ->scheduling

--resource database

--BCP(bulk copy program) and BI (Bulk insert)

select \* from

In a 8KB pages, maximum of 8060 bytes can be used by data

--What are internal fragmantation and ecternal fragmentation?

Storing data non-contiguously on disk is known as fragmentation

Internal Fragmentation: this happens when the leaf nodes of clustered index not filled to full campacity

due to memeory bubbles.When records are stored non-contiguously inside the page, then

it is called internal fragmentation. In other words, internal fragmentation is said to

occur if there is unused space between records in a page. This fragmentation occurs through

the process of data modifications (INSERT, UPDATE, and DELETE statements) that are made against

the table and therefore, to the indexes defined on the table.

Internal fragmentation is measured in average page fullness of the index(Page density).

A page that is 100% full has no internal fragmentation.

consequence:This unused space causes poor cache utilization and more I/O, which ultimately leads

to poor query performance.

External Fragmentation: this happens when the physical storeage of pages is non-contiguous. When you intert a new row

into full page. SQl server has to spilt the page into two pages and move half of the rows to the

second page. The new page can be reserved anywhere in a datafile. physical order of pages and extents

of a clusterd table don't correspond to it's logical order

consequences:???

How Internal Fragmentation will affect the performance of the SQL server ?

Internal Fragmentation will increase the I/O. When you run queries that scan part or complete table/index, if you have internal fragmentation on that table/index, it causes additional page reads. In our example, the entire data can be stored in 5 pages. When the query needs to do index scan it has to read 10 pages instead of 5 pages. Which means 50% more I/O.

Internal Fragmentation reduce the efficiency of buffer cache.When indexes has internal fragmentation, it need more space to fit in the buffer.In our case this single index will use 5 additional pages to fit into the buffer which should have used to store other index pages. This will reduce the cache hit ratio. In turn it will increase the physical I/O. It also increase the logical reads.

This also increase the size of the database file. It need more disk space to store the additional pages and reduce the performance of Backup and Restore.

How External Fragmentation will affect the performance of the SQL server ?

While reading individual rows, external fragmentation will not affect the performance as it directly go to the page and fetch the data.Unordered scans also will not affected by the external fragmentation as it use the IAM pages to find which extents need to be fetched. In the case of ordered index scan ,external fragmentation might become a degrading factor for performance. The degradation of the performance is because the disk drive's heads have to jump around on the physical disk, rather than performing just contiguous read operations.Also note that external fragmentation will not affect the performance once the pages are loaded into the buffer pool.

How to measuring The Index Fragmentation?

create nonclustered index idex\_covering on T9(Name)

include(id)

-sys.dm\_db\_index\_physical\_stats

sys.dm\_db\_index\_physical\_stats (

{ database\_id | NULL | 0 | DEFAULT }

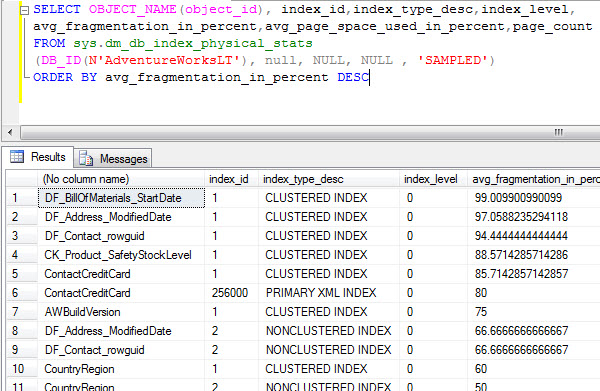
, { object\_id | NULL | 0 | DEFAULT }

, { index\_id | NULL | 0 | -1 | DEFAULT }

, { partition\_number | NULL | 0 | DEFAULT }

, { mode | NULL | DEFAULT }

)



avg\_fragmentation\_in\_percent: This is a percentage value that represents external fragmentation. For a clustered table and leaf level of index pages, this is Logical fragmentation, while for heap, this is Extent fragmentation. The lower this value, the better it is. If this value is higher than 10%, some corrective action should be taken.

avg\_page\_space\_used\_in\_percent: This is an average percentage use of pages that represents to internal fragmentation. Higher the value, the better it is. If this value is lower than 75%, some corrective action should be taken.

Reducing Fragmentation in an Index: There are three choices for reducing fragmentation, and we can choose one according to the percentage of fragmentation:

If avg\_fragmentation\_in\_percent > 5% and < 30%, then use ALTER INDEX REORGANIZE: This statement is replacement for DBCC INDEXDEFRAG to reorder the leaf level pages of the index in a logical order. As this is an online operation, the index is available while the statement is running.

If avg\_fragmentation\_in\_percent > 30%, then use ALTER INDEX REBUILD: This is replacement for DBCC DBREINDEX to rebuild the index online or offline. In such case, we can also use the drop and re-create index method.

--what is a transaction?

1. A transaction is a logical unit of work. Either all the work completes as a whole unit, or none of it does.

2. Transactions very common in our daliy lives.Eg. Purchase something can be considered as a trans. When you

pay money for something but don't recieve the object. The trans is stopped and you expect to receie your money

back. Paying with your money and receive what you purchased form a logic unit of work. Either both step success

or both step must fail together.

3. For SQL server, all changes to database data take place in the context of transaction. In other word,all operations

that in any way write to the database are treated as trans. This inclue DDL & DML statement. Technically, even

single select statemtent are a type of transation. These are called readonly trans

4. In relational database, transaction should has the ACID properties.

A stands for Atomicity, this make sure there is no partially completed transaction

C stands for Consistence, this ensures that the integrity of datebase data will never be compromised

I stands for Isolation, this make sure every trainsation is functionally independent

D stands for Durability, this makes sure each transaction endure an interruption of service

--What is commit & roll back

commit: is used to finalized the statement in transaction and save changes

rollback: is undo all stantement and changes

--what is ACID properties

1. Atomicity: this requires every transaction is an atomic unit of work, meaning that all database changes in

the transaction succeed or none of them succeed.

For Example: Consider, for example, a update statement that would update 500 rows at the point in time

the transaction begins. The command will not finish until exactly all those 500 rows are updated.

if sth previents that command from updating all 500 rows, the server will abort the command and roll

back the transaction so that uncommited changes are removed.

2. Consistency: Every transaction, whether success or not, should leave the database in a consistent state as defined

by all object and database constraints. if an inconsistence occurs, the server will roll back the trans

For example, if your transaction attempts to insert a row that has has a invalid foreign key, then the

server will detect a constraint would be violated and generate an error message. in this case, you can

add logic to decide whether or not to roll back the transaction

3. Isolation: Each transaction looks as though it occurs in isolation from other transaction in regard to database

changes. This degree of isolation can vary based on isolation level.

For example, none of the objects being changed by one transaction are not allow to be changed by any other

transactions. if two trans want to change the same underlying data, one of them must what until the other

trans is finished.

4. Durability: means every transaction endures through an interruption of service. when service is restored, all committed

trans are rolled forward and all uncommited transaction are rolled backed. SQL server maintain transactional

durability by using the database transaction log.

For example, if the server shuts down unexpectedly just after the fact of successful commit has been written

to the transaction log, when the server starts up the database, the transaction will be rolled forward

and any unwritten changes to the database will be finished

commit->commit successful->data written to log->written to database

There are two type of Transaction Durability: a. full durability(commit=logged record writen to disk)

b. delayed durability(commit=record logged)

--What are the transaction modes and compare them

In sql server, we have implicit transaction mode and explicit transaction mode

Implicit transaction mode: In this mode, when you issue one or more DML or DDL statement, or a select statement

sql server starts a transaction implicitly. However, a commit or rollback must be issued

to finish the transaction.

Implicit transaction is not the sql server default. You can enter the mode by issuing

SET IMPLICIT\_TRANSACTION ON;

Explicit transaction mode: In this mode, we have to explicitly issue the BEGIN TRANSACTION command to start a transaction

The transaction ends with commit or rollback command

--What is savepoints?

There are locations with transactions that you can use to roll back a selective subset of work. You can define a

savepoint by using the SAVE TRANSACTION <save point name>. The rollback statement must reference the savepoint,

otherwise, if the statement is unqualified, it will roll back the entire transaction.

--What is @@TRANCOUNT & XACT\_STATE()

@@Transcount: can be queried to find the level of transaction.

A level of 0 indicates that at this point, the code is not within a transaction

A level >0 indicates that there is an active transaction in the session and a number

greater than 1 indicates the nexted level of nested transaction

XACT\_STATE(): can be queried to find the state of the transaction

A state of 0 indicates that there is no active action

A state of 1 indicates there is an uncommitted transaction, AND it can be committed

A state of -1 indecates there is an uncommitted transaction, BUT it can't be committed due to

prior fatal error

--Explain different types of locks in SQL Server

Shared Locks: When a transaction sets out to read data, the server will attempt to place a shared lock on the data,

Shared locks are compatible with other shared locks. Thus two or more transaction can read the same objects at the same time

But shared locks are not compatible with exclusive locks, meaning other transaction can't modify data when there is a shared lock.

IN trans which operating with the default READ COMMITED isolation level,the share lock will be released as soon as the date is read(in read commmmit)

Exclusive Locks: when s transactions sets out to change data(DML), the server will attemp to secure an

exclusive lock on the data Exclusive lock is not compatible with any other kind of lock

Thus, when a transaction has a resource locked exclusively, no other transaction can rad

or writer to the resources

Update Locks: Update (U) locks prevent a common form of deadlock. A typical update pattern consists of a transaction reading a record,

acquiring a shared (S) lock on the resource (page or row), and then modifying the row, which requires lock conversion

to an exclusive (X) lock. IF two trans are updating a resources together, because both transactions are converting to exclusive (X) locks,

and they are each waiting for the other transaction to release its shared-mode lock, a deadlock occurs.

To avoid this potential deadlock problem, update (U) locks are used. Only one transaction can obtain an update (U) lock

to a resource at a time. If a transaction modifies a resource, the update (U) lock is converted to an exclusive (X) lock.

Otherwise, the lock is converted to a shared-mode lock.

Intent Locks: Intent locks are SQL server’s way of reserving(mark) data to be locked.

Lets say we want to use a table and specify a shared lock, SQL server places an intent shared lock on the table

so no one else places a lock on it to prevent us from using it

Schema Locks: Schema locks are used when there is a DDL operation being performed on a table. This doesn’t prevent shared or exclusive locks,

so people can still use the data. It does prevent OTHER DDL operations from occurring on the same table

Bulk Update Locks: Prevents other processes that are not bulk operations copying data into a specified table from having any access to the data during

the process. Only bulk operations can have assess to data. To use this lock there must be TABLOCK specified (Hint)

--What is row versioning?

When a row versioning-based isolation level is enabled, the Database Engine maintains versions of each row that is modified.

Applications can specify that a transaction use the row versions to view data as it existed at the start of the transaction

or query instead of protecting all reads with locks. By using row versioning, the chance that a read operation will block other

transactions is greatly reduced.

--What is blocking and deadlocking?

Blocking: A block occurs when one session has an exclusive lock on a resource, preventing another

session from obtainning any kind of lock on the resources. In a transaction, exclusive locks are held

to the end of the transaction. Other transaction will have to wait until the first session either commits

or rolls back. Thus writers can block other writers and readers

deadlocking: A deadlock results from mutual blocking between two or more sessions. Sometimes locking squences between sessions

can't be resolved simply by waiting for one transaction to finish. This occurs due to a cyclical relationship between

two or more commands.SQL server detects this cycle as deadlock between two transaction. aborts one of the transaction

and returns error maeesge 1025 to the client

t1 is block by t2, normally t1 just need to wait t2 to commit or rollback. But now, t2 further has statement

work on some objects that is locked by t1 previously, in this case, t2 is blocked. t1 and t2 blocks each other

thus dead lock

--Explain all the Isolation Levels

READ UNCOMMTED: This isolation level allows reader to read uncommitted data. This setting removes the shared locks taken by

select statement so that reader is no longer blocked by writer.

However the results of a select statement could read uncommitted data that was changed during a transaction

and then later was rolled back to its initial state. This is called reading dirty data

READ COMMITED: This is the default isolation level. All readers in that session will only read data changes that have been commited.

To do this, all select statement will attempt to acquir share locks, and any uderlying data resources that are being changed by

a different session, and therefore have exclusive locks, will block the read committed session

REPEATABLE READ: This isolation level, also set per session, guarantees that what-ever data is read in a transaction can be re-read later in the

the transaction. Updates and deletes of rows already selected are prevented.

Shared locked are placed on the data readed and will be held to the end of transaction

However transaction may see new rows added after its first read, which is call phantom read

SERIALIZABLE: This is the strongest level. At this level, all reads are repeatbble and new rows are not allow for later read.

Share lock placed on the range defined by the search criteria and will be held to the end of transaction

Shared Lock Read Anormaly

read uncommited no shared lock Dirty Read, Unrepeatable read, Phantom read

read commited shared lock on readed row, release after statement Unrepeatable read, Phantom read

repeatable read shared lock on readed row, release after trans Phontom read

serializable read shared lock on readed range, release after trans None

READ COMMITTED SNAPSHOT(RCSI): This is actually not a new isolation level; it is an optional way of using the default READ COMMITTED isolation

level. it reads commited data change thus will not have dirty read. Instead of using locks, RCSI is based on row

versioning; Since it doesn't need locks, data modification from other transaction can work on the underlying date

at the same time. deadlock situation will be solved. But note that RCSI set at database level and is a persistent

database property

SNAPSHOT: This isolation level also uses row versioning in tempdb(as does RCSI). It will be able to repeat any reads, no dirty read, no phaontam

reads.New row may be added to the table, but transaction will not see them. This is also a persistent database property and then set per transaction

sql server accomplished transactional isolation by means of LOCKING and ROLL VERSIONING

--Read Uncommitted: NO share lock

--Read Committed: share lock released at the end of each statement

--Reaptable read: share lock placed on all data returned by the search criteria + share lock released at the end of transaction

--SERIALIZABLE : share lock placed on the range defined by the search criteria(range lock) + share lock relaased at the end of transaction

The transaction isolation levels define the type of locks acquired on read operations. Shared locks acquired for

READ COMMITTED or REPEATABLE READ are generally row locks, although the row locks can be escalated to page or

table locks if a significant number of the rows in a page or table are referenced by the read. If a row is

modified by the transaction after it has been read, the transaction acquires an exclusive lock to protect

that row, and the exclusive lock is retained until the transaction completes. For example, if a REPEATABLE

READ transaction has a shared lock on a row, and the transaction then modifies the row, the shared row lock

is converted to an exclusive row lock.

--when a session is reading data, by default sql server will issue very brief

shared locks on the resource, such as a row or table

--In a transaction, exclusive locks are held to the end of the transaction

--In a transaction operating with the default Read Committed isolation level, shared locks are released as soon

as the data is read, shared locks are not held to the end of the transaction, except for higher isolation levels

--exclusive lock and share lock are record based