Questions about the contents of the final section of the course of Advanced Databases. Version 0.1 of 22/05/2018.

13 Column databases

Which are the reasons behind the success of column databases? Which have been the changes in the applicative scenario and in the technological scenario? Which are the strongest advantages of a column store over a traditional DB when we have an OLAP application? What are the problems? Why are update operations especially expensive with a column store?

In the traditional Decomposition StorageModel, a ‘column’ is actually a binary table. Can you explain this notion? Is that the same in a column store, that is do we store the RID-Value pairs, or do we just store the values? How is a tuple identified, in this case?

There two possible choices for column order – which are they? Which are the advantages of each possibility?

What is a join index? What is a projection? How many different projections may we have on the same table? Two projections that include the same columns may be different?

Consider a table Sales(FKCust, FKProd, Date, TotPrice), in a traditional DB, sequentially organized by Date, and consider the projection (FKCust, FKProd, Date, TotPrice | Date). What is the difference among the two? Are they stored in the same way?

Why do column stores compress better than traditional databases?

What does it mean to execute an access plan ‘one tuple at a time’ or ‘one operator at a time’? Why do traditional databases prefer ‘one tuple at a time’? What is the problem of that approach with modern architectures? Which intermediate approach is particularly good for modern architectures? Consider the select operator of the binary algebra. Do we need to decompress and RLE-encoded column in order to apply Select? What if it is compressed as a bitmap or with a sorted dictionary?

Consider a table that is stored by columns, and all columns are sorted by RID. What must be done in order to rebuild the original table? How much would that cost – including the cost to read the input and to write the result? Assume that, instead, it is stored as a set of projections and we have two projections such that (a) they have the same order and (b) their union covers the entire table. What is the cost, in this case? Is it higher or lower? Assume now that the two projections have different orders, that B*B is bigger than 2*NPag for each of them, and that we have a join index. What is the cost of rebuilding the entire table?

What are the Binary Algebra operators that we have seen? Can you give a formal definition of each of them, apart the Group operator? Can you give an example of application of the group operator?

Consider a join query that returns a single column such as SELECT R.A FROM R, S WHERE R.B=S.B and R.A = 12 and S.C > 10. Can you write a binary algebra access plan that expresses this query? How can you represent a BAT where the first column is a list of consecutive integers? How can the result of a Select operator be represented? What is the advantage of including a range and a bitmap in the representation of a BAT? How can a select operation be implemented in this case? How can a Reconstruct operation be implemented in this case? What is the cost of a Jive Join? Does this cost include the initial reading and writing? Under which assumptions on memory size? What is the cost of a Hash Join under the same assumptions? Why is Jive Join less expensive than Hash Join? Would you use a Jive Join in a traditional database? Why? Can you insert a Jive Join into a traditional access plan that is executed one-tuple-at-a-time? How is typically executed a group-by in a column database? Why is tuple insertion/deletion/update expensive in a column store? What is the solution to this problem?

Many column systems give a different differential store to different transactions. Why? What are the Read Store and the Write Store of a columnar data store? Which is the motivation for this approach?
How can we exploit the existence of RS and WS to implement transactions? Why do not have we indexes in a column store? Explains these two sentences: “A sorted column A with a join index corresponds to a traditional index A”, “A projection \{ A, B, C | C \} may be considered as an index on C that allows one to rapidly access A and B, with an IndexOnly plan”.

Assume that we try and simulate a column store by building, in a traditional store, one index for every different column. Would that work? Would that have some advantage wrt a traditional design? Would that suffer from some specific problem? Assume that we try and simulate a column store by decomposing a table, in a traditional store, according to the DSM, and organizing every table as a B+-tree over the RID field. Would that work? Would that have some advantage wrt a traditional design? Would that suffer from some specific problem?

Exercises

1) Consider the following tables:

\begin{verbatim}
Sales(Date,FKShop,FKCust,FKProd,UnitPrice,Q,TotPrice)
Shops(PKShop,Name,City,Region,State)
Customer(PKCust, Nome,FamName,City,Region,State,Income)
Products(PKProd,Name,SubCategory,Category,Price)
\end{verbatim}

With the following size:

- Sales: NRec: 100.000.000, Npag: 1.000.000; Shops: 500, 2; Customers: 100.000, 1.000;
- Products: 10.000, 100

Consider the following query:

\begin{verbatim}
SELECT Sh.Region, Month(S.Date),Sum(TotPrice)
FROM Sales S join Shops Sh on FKShops=PKShops
GROUP BY Sh.Region, Month(S.Date)
\end{verbatim}

Propose a data organization based on this query, where the function Month returns ‘month-year’, not just month: Month(18/05/2018)=‘05/2018’ (not ‘05’).

Consider both primary organization and indexes. Consider the possibility of denormalization and vertical partitioning.

Compute the cost of an optimal access plan based on this organization.

Add a condition: WHERE 1/1/2017 < Date.

Assume that we want to optimize some variants as well (where the SELECT clause changes according to the GROUP BY)

…GROUP BY Sh.City, Year(S.Date)
…GROUP BY S.Date
…GROUP BY Sh.Region, S.FKCust

2) Define a projection that is optimal to answer the following queries (one projection for both queries):

\begin{verbatim}
SELECT Sh.Region, Month(S.Date),Sum(TotPrice)
FROM Sales S join Shops Sh on FKShops=PKShops
GROUP BY Sh.Region, Month(S.Date)
\end{verbatim}
SELECT Date, count(*)
FROM Sales S join Shops Sh on FKShops=PKShops
WHERE Date > 1/1/2016
GROUP BY Date

3) Assume you have a sorted column whose values are 16 bytes strings, with NKey = 1000 and NRec=100.000. How many bits do you need to store it with no compression? How many with RLE? How many with bitmap encoding, without compressing the bitmap? How many with bitmap encoding, where the bitmap is RLE encoded? How many with dictionary encoding? Do all computations in bits, assuming that words do not need to be byte-aligned.

4) Assume you have an unsorted column whose values are 16 bytes strings, with NKey = 100 and NRec=100.000. How many bits do you need to store it with no compression? How many with RLE? How many with bitmap encoding, without compressing the bitmap? How many with bitmap encoding, where the bitmap is RLE encoded? How many with dictionary encoding? Words do not need to be byte-aligned.

5) Assume you have an unsorted column whose values are 16 bytes strings, with NKey = NKey and NRec=NRec. How many bits do you need to store it with no compression? How many with RLE? How many with bitmap encoding, without compressing the bitmap? How many with dictionary encoding?

6) Consider query

SELECT Sh.Region, Month(S.Date),Sum(TotPrice)
FROM Sales S join Shops Sh on FKShops=PKShops
GROUP BY Sh.Region, Month(S.Date)

Of exercise (1). Can we answer it using an access plan where every leaf is ‘IndexOnlyFilter”? Do we need new operator if we want to have such a plan? Assume that Sales and Shops are stored using the DSM approach. Draw an access plan for this query.

14 Parallel and Distributed Databases

Which are the fundamental tasks of any database, including a distributed database? Which are the central issues in a parallel (distributed) database? How would you define a parallel (distributed) database?

What are the models of parallelism? What is the different between Shared Memory/Shared Disk/Shared Nothing? What is the problem of a SM machine, and what is the advantage? Whenever possible, we do not send a tuple-at-a-time, but we collect as many tuples as we can, before sending a message of a bigger size. Why?

Describe different techniques to (horizontally) partition data in a parallel system, Considering the following techniques for the horizontal partitioning: Range partitioning, Hash partitioning, round-robin, block partitioning, co-located partitioning. What is an advantage of each? Specify some reasons to partition data in a parallel or distributed setting.
In a parallel system, do we execute access plans one-tuple-at-a-time or we usually execute one entire operator before moving to the next one? Why?

How would you execute union / intersection / difference / distinct on a parallel machine? How would you evaluate group-by and join? Can you estimate the time to complete the operation? How does that depend on the size of the buffer of the machines? What about filter? What is the relation between the way filter is executed and the technique to partition data?

If the system has 100 nodes, would you expect it to be 100 times faster, when executing one operation, with respect to a one-CPU system, or somehow less efficient than that? Or even better than that? We assume here that executing an operation includes reading the initial data from disk and writing the result on disk.

When data partitioning is very skewed, does this fact affect the efficiency of the system? Why?

What are the main differences between parallel shared-nothing systems and distributed systems? Which are the similarities? Which is the most expensive operation in a distributed system? What may you say about failures?

What is “data distribution design”? How can we design the data distribution? Which are the decisions to take, and which are the goal? What should we know in order to accomplish this task?

Data fragmentation is based on horizontal partitioning, vertical partitioning, or both? Which are the differences between this phase in a parallel database and in a distributed database?

How would you describe the problem of the distributed commit?

Assume a transaction with participants p1, p2, p3 and coordinator c. Assume that everybody is ready to commit. Which sequence of actions – log writing and messages – would take place during a normal commit process? Assume you are a participant, you receive a message (prepare T) from the coordinator, and, later, you receive another identical message. Why? What may be happened? How will you react? When is the last moment when a participant may autonomously decide to rollback a transaction?

When is the last moment when the coordinator may decide to rollback a transaction?

Assume you are a participant. You receive ‘prepare’, you answer ‘ready’ but then, for a long time, you do not get any more message from the coordinator. Why is that a problem for you (excuse the anthropomorphism of the discussion)? What can (and should) you do if the wait time gets too long?

In some implementations of the protocol, the message ‘don’t commit’ is broadcasted to all participants, rather than being sent to the coordinator only. Why may that be useful?

Consider a participant Pi that restarts after a failure, and the last log record says ‘ready,T’. Do we know what is the last message that Pi received? Do we know what is the last messages that it sent? Do we know whether that message has been received or not? Assume that the participant decides to rollback T. Is this decision safe? Assume that it decides to send a message ‘don’t commit’, is that safe? Assume that it decides to send a message ‘(ready T)’ to the coordinator, is that safe? Assume that it decides not to send any message, is that safe? In order to answer these questions, it is necessary to reason by cases: what if Pi already sent a message before crashing? What if it did not? What may the others have done, while Pi was down?

Assume that, at restart, Pi sees that the last log record was ‘don’t commit,T’. Consider which of the following choices are safe: rollback the transaction? send a message (don’t commit T)? do not send any message?

Assume that, at restart, the coordinator C sees that the last log record was ‘Prepare,T’. Which was the last message the C has sent (list all the possibilities)? Consider which of the following choices are safe: do not do anything, send a message (Prepare T), send a message (Abort T), send a message (Commit T) (writing a log record if needed).

Assume that, at restart, the coordinator C sees that the last log record was ‘Abort,T’. Which was the last message the C has sent (list all the possibilities)? Consider which of the following choices are safe: do
not do anything, send a message (Prepare T), send a message (Abort T), send a message (Commit T) (writing a log record if needed).
Assume that, at restart, the coordinator C sees that the last log record was ‘Commit,T’. Which was the last message the C has sent (list all the possibilities)? Consider which of the following choices are safe: do not do anything, send a message (Prepare T), send a message (Abort T), send a message (Commit T) (writing a log record if needed).
Assume that we want to implement a distributed lock protocol that ensures that no two processes may hold two conflicting locks over the same logical entity. Which possibilities exist? Which are the advantages and the problems of a centralized solution? May we adopt a solution where every logical object has one specific lock manager but the lock managers do not communicate the ones with the others? In a situation of data replication, is it possible to devise a solution where every process only communicates with a lock manager that is on the same node as the process? Describe the write-locks-all and the majority-lock approaches. Which are the advantages of one and of the other? Is it necessary that the lock managers are in the same nodes as the managed objects? Is that useful? Define the quorum approach. What is the relationship between the primary copy, the write-locks-all, the majority-lock, and the quorum approaches? Can ones be described as a special case of another?
If we adopt the lock protocol – that is, only write or read on data after you got the necessary locks – may we run into consistency problems, such as non-repeatable read, or reading two different values from two different nodes? Is distributed deadlock a problem? Is it possible to keep a global wait-for graph? Is it complicated? Can we generalize the prevention approaches – such as wait-die or would-wait 2PL – to the distributed setting?

Consider a participant Pi that restarts after a failure, and the last log record says ‘don’t commit’. Do we know what is the last message that Pi received? Do we know what is the last messages that it sent? Do we know whether that message has been received or not? Assume that the participant decides to rollback T. Is this decision safe? Assume that it decides to send a message ‘don’t commit’, is that safe? Assume that it decides not to send any message, is that safe? In order to answer this questions, it is necessary to reason by cases: what if Pi already sent that message before crashing, what if

Exercises

1) Assuming that you have k nodes, connected in a Shared-Noting network, that you take time 10ms to read 4KB of data, that you send data in blocks of 4KB which need 30 ms each to be transmitted. Assume that R and S both measure 10 GB, that R is hash-distributed around nodes 1-100 according to h1(R.A), while S is hash-distributed around nodes 1-50 according to h2(S.B). Describe how would you proceed to compute R Intersect S. Is it possible to parallelize the operations of reading data from disks and sending messages? The result must be written on disk, and you may leave it distributed on the disks of the different machines in any way. Compute the elapsed total time of the computation, including reading data, distributing it for computation, and storing the result.

2) Assume you have relation R(X,Y) at site @R of size NPag(R) and relation S(Y,Z) at site @S of size NPag(S). Assume that sf(SJ(S,R)) is the selectivity factor of SJ(S,R), that is NRec(SemiJoin(S,R))/NRec(S), L(Y) is the size if the Y attribute, and LRec(S) (respectively LRec(R)) is the size of a record of S (resp. R). Write a formula for the amount of data communication that is needed in order to compute the following plan:
O1@R = TableScan(R)@R
O2@S = Send(O1,@S)
O3@S = TableScan((S)@S)
O4@S = SemiJoin(O2,O3)
O5@R = Send(O4,@R)
O6@R = TableScan(R)@R
O7@R = Join(O5,O6)@R

3) Assume that a group of three participants and one coordinator must perform a 2PC. At the beginning every node may fail at every moment, but it is guaranteed to restart after 10 seconds, and, after the first failure, to stay up for at least one second, which is enough to do any *sequence* of read/write log or send/receive message, if the send/receive partner is up. Under this hypothesis, try and find the longest possible delay for protocol termination. We assume that no partner ever gives up and that, when an expected message is not received, all partners request a re-send at random intervals never longer than 2 seconds.