4-Mar Presentation Note Trust but Verify

Presenter: Zachary Huang Scribe: Xupeng Li

(1) Sample + Seek

Provides background of samplers. Sampling what users want and telling the error bound.

(2) Optimistic Data Visualization

Run queries at background and once the query is done, it is sent back to user and let user verify

whether the conclusions drawn from the previous approximated charts are invalidated.

User interface shown in the paper:

(1) select the column

(2) specification of the visualization

(3) approximated value

(4) Uncertanty

(5) what people did before

(6) query running on the background

Evaluation: three case study

Takeways

AQP works

Optimism works

Draft query

Discussions

How do we evaluate the contribution of this paper? Sampling is very useful, but is not a contribution of this paper. The first auther is from visualization research field. He applied the sampling techniques to build his software and try to close a gap in visualization research. The contribution of this paper is optimistic visualization that is a way of how to visualize. Is it a large contribution, is it a strong paper? Or just a documentation for a software?

The contribution of this paper is that it proposes a way of using sampling to solve a severe problem of database, which is database sometimes need very long time to converge for a single query, while using sampling users can get assumptions and distributions very quickly. It is a very useful application and can be commercialized.

This is a visualization paper. The focus would be what the design tradeoff is, why they design or how they design. The contribution of this paper is the interface design and why they choose this design.

If I were to design such a system, what kind of things are considered to be important? How many other things did they do? The idea of optimistic visualization is not hard to imagine that we are going to run approximate queries and eventually come to check to see if it is correct. It is a nice design but how do you show that and how do you decide what to check. It is even possible to do it automatically. It is a lot of subtleties that If you do sampling, how does it affect visualizations?

Prior papers are focusing a lot on dealing with one query. How to approximate a query and how to encoding uncertainty.

This paper is no like Voyager that guild you what kind of query to consider. There's no guidence here. It is closer to Polaris. But unlike Polaris where queries can be done quickly, here we consider that data is big and query will run for a long time. So we use approximation. And here's tradeoff between accuracy and efficiency.

Another aspact to consider is how to design an interface to fire a lot of queries at a time and how should you manage them.

Users may forget what they ran or why they ran because they ran a lot of queries. Even though a query finally return with some difference from the approximation, users can hardly to find out what does it mean. Maybe a query should run for two days, and when it eventually done, users already forgot what happened.

A user may run a lot of queries in order. The second query he ran is based on the approximate result of the first query. And the third query is based on the second. Finally, the user find the first queries's approximate result is wrong, all the following queries are meaningless. Maybe we can organize all queries in a tree to model dependencies. Once a query is shown to be wrong, all the following queries can be highlighted.

COMS W6998

Wander join

Presenter: Xupeng Li

Scribe: Zachary Huang

Basic idea: Approximate aggregation result join.

Join is common but expensive. Join of small tables leads to large table.

Fast from hours to seconds with high confidence.

Ripple join vs Wander join

Ripple join dependent, uniform,

Ripple join maintain sample pool, record all samples and result,

Stored In memory,

Sample from A and B, and get the result,

Sample each row is uniform, but dependent on previous sampling.

Wander join independent, non-uniform,

Cases for cyclic or acyclic

1. Tree decomposition. 2 choose walk. 3. Sanity check

Experiment

1. converge time is much less for wander join for different queries

2. low time to achieve different confidence levels

Main contribution:

Propose a new, efficient algorithm, implemented in popular database.

Discussions:

Q: How to compete the confidence interval?

Because sample dependent, becomes very complex.

Wander join ~2015 won the best paper because blurs online algorithm, which is advanced for a long time.

Q: Why is ripple join bad? Is it because random? No Guide? What bad means?

Purpose of sampling is for performance.

Current semantics: sample from one final time, and further do query.

Use some magic gives us sample without runs join.

Sampling: how many times join and throw, which is inefficient.

One approach: random samples from left and right, do a join.

With likelihood, they will join, otherwise waste effort.

Likelihood depends on join and table.

Sample(L)join Sample(R) not equal Sample(L join R).

Also related to the hardware improvement.

80s: store on disk. Random access cost is large.

For disk only 5ms, page 8 kb, which you can sample. Scan 300 m/s.

Q: Why sample when can read and run the query?

Sample + Seek: know query, do sample ahead of time. Offline sampling.

Do the random seeking before won't be expensive.

Wander join: has large memory, can build indexes.

Index: receives a key, see records that have the key.

Therefore, can directly ask how many records match.

Likelihood of getting a good join much better.

Both online or offline indices are possible.

Random walk: hope that join, otherwise waste resource.

Q: What is independence sample? Why it is important? If so many matches for the last table, why not take all of them? Afterall, we want sample pool to be large!

Because they are not independent.

Random sample in memory is not cheap.

Filter will also cause waste if it's not in the index.

Group by is also filter. Filter by the group by values.

Indendence helps calculate the confidence. (Ripple has many long pages talking about calculating confidence interval, which is hard)

Q: How to use pangloss to run online aggregation?

Pangloss use sample seek, very quickly we get a good approximate result, wait a long time, and have the full result. Online aggregation gives you answer bit by bit. Wander join can increase every seconds.

Sample seek doesn't have anything in between. Get approximate quickly, and a long while for final result.

Q: How to visualize join?

No visualization currently considers join.

Wait ahead of time until the final result.

Why is visualization of join hard?

Because size, time…

Can we visualize it using data density for joins?

If joining tables inherently, there will be even larger tables.

Rendering table would also be a problem.

User may want to try join.

First join, if very large, then don't want them.

Query will shrink things down like in common website.

Therefore, interactive visualize join is very hard.