Data Science and Engineering (2018) 3:379–397
https://doi.org/10.1007/s41019-018-0074-4

Approximate Query Processing: What is New and Where to Go?

A Survey on Approximate Query Processing

Kaiyu Li[1] · **Guoliang Li**[1]

Received: 16 May 2018 / Accepted: 3 September 2018 / Published online: 14 September 2018

Abstract
Online analytical processing (OLAP) is a core functionality in database systems. The performance of OLAP is crucial to make online decisions in many applications. However, it is rather costly to support OLAP on large datasets, especially big data, and the methods that compute exact answers cannot meet the high-performance requirement. To alleviate this problem, approximate query processing (AQP) has been proposed, which aims to find an approximate answer as close as to the exact answer efficiently. Existing AQP techniques can be broadly categorized into two categories. (1) Online aggregation: select samples online and use these samples to answer OLAP queries. (2) Offline synopses generation: generate synopses offline based on a-priori knowledge (e.g., data statistics or query workload) and use these synopses to answer OLAP queries. We discuss the research challenges in AQP and summarize existing techniques to address these challenges. In addition, we review how to use AQP to support other complex data types, e.g., spatial data and trajectory data, and support other applications, e.g., data visualization and data cleaning. We also introduce existing AQP systems and summarize their advantages and limitations. Lastly, we provide research challenges and opportunities of AQP. We believe that the survey can help the partitioners to understand existing AQP techniques and select appropriate methods in their applications.

Keywords OLAP · Approximate query processing · Online aggregation · Offline synopses

1 Introduction

Online analytical processing (OLAP) is a core functionality in data management and analytics systems [33]. The performance of OLAP is crucial for many applications that need to use OLAP to make online decisions, e.g., business intelligence. However, it is rather costly to support OLAP for large datasets, especially big data. Many systems have been proposed to support OLAP on big data, e.g., Pig, Hive, Spark SQL, and they usually take tens of minutes or even hours to answer an OLAP query. However, many applications have online requirement of OLAP that want to get results in seconds.

✉ Guoliang Li
liguoliang@tsinghua.edu.cn

Kaiyu Li
liky15@mails.tsinghua.edu.cn

[1] Department of Computer Science, Tsinghua University, Beijing, China

To alleviate this problem, approximate query processing (AQP) has been proposed, which computes approximate answers (with some quality guarantee) very efficiently to meet the high-performance requirement. Next, we use several examples to show how AQP works. Note that AQP only performs well for aggregate functions such as `SUM`, `AVG`, `COUNT`, `MAX` and `MIN` due to it should use statistical tools to give approximate results for numerical types of answers.

AQP Use Cases We consider a database with multiple tables in Fig. 1. For simplicity, we only show three tables, `orders` **O**, `customers` **C**, `states` **ST**, and the relations among the tables. We consider the following three use cases of AQP.

Case 1 (Online Aggregation) An analyst wants to know the average profit of the orders from customer c_1 within one second, and she can pose a query:

```
SELECT AVG(PROFIT) FROM O
WHERE CUSTOMERID =}c'1 WITHIN 1 second
```

OrderId	CustomerID	SupplyID	Good	Profit($)
o_1	c_1	s_1	CD	1170
o_2	c_2	s_3	IPHONE6	1250
o_3	c_1	s_2	BOOK	1417
o_4	c_3	s_4	FRUIT	1560
o_5	c_2	s_5	IPHONE8	1983
o_6	c_1	s_7	PEN	1120
o_7	c_4	s_6	COLA	1350
o_8	c_3	s_1	FISH	2000
o_9	c_2	s_3	IWATCH2	1673
o_{10}	c_1	s_2	BOOK	1230
o_{11}	c_4	s_5	SODA	1460
o_{12}	c_1	s_1	CD	1890
o_{13}	c_2	s_3	IWATCH2	1732
o_{14}	c_1	s_2	BOOK	1630
o_{15}	c_4	s_5	SODA	1470
o_{16}	c_1	s_1	CD	1290
.....	.....	.....	.....	

Order Table: O

CustomerID	StateID	Level	Gender	Name
c_1	st_1	L3	Male	Jack
c_2	st_3	L5	Female	Rose
c_3	st_2	L3	Male	Jimmy
c_4	st_1	L2	Female	Lucy
.....	.....		.....	.....

Customer Table: C

StateID	StateName	Short Name	Captial
st_1	Massachusetts	MA	Boston
st_2	New York	NY	Albany
st_3	Washington	WA	Olympia
.....	.....	.....	.....

State Table: ST

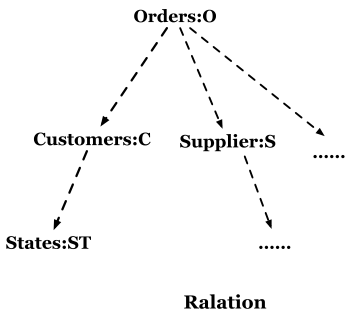

Fig. 1 Part of Relation R

A straightforward method uses random sampling to select some samples, e.g., o_1, o_9, o_{12}, and uses the samples to answer the query. The sample size can be determined based on the user-provided time constraint. For example, assume the system can process 3 samples in each second, then the sample size is 3. Here are two samples o_1 and o_{12} from customer c_1, which are used to answer the query. Then the system uses closed-form estimation (see Sect. 2.2.2) and gives an approximate result with confidence interval, e.g., $[1530.0 - 360.0, 1530.0 + 360.0]$ with a confidence of 90% based on central-limit theorem (CLT) [6].

Case 2 (Online Interactive Aggregation) An analyst wants to continuously monitor the results in an interactive manner, and she poses the following interactive query:

```
SELECT AVG(PROFIT) FROM O
WHERE CUSTOMERID =}c'_1 WITHIN 8s INTERVAL =1s
```

Then, the system incrementally selects samples in every second and iteratively updates the answer using more examples. For example, after the fourth iteration, the system totally gets 12 samples from o_1 to o_{12} and the answer is $[1392.4 - 329.4, 1392.4 + 329.4]$ with a confidence of 90%. If the analyst is satisfied with current answer, she can stop the query processing.

However, online sampling without pre-computing may lead to large errors and the quality is uncontrollable. For example, consider a rare group with few tuples (e.g., c_4), and the online methods may not get any sample from the group. Thus, the online sampling method fails to provide a high-quality result. To address this problem, we can use offline synopses.

Case 3 (Error-Bounded AQP) Given a database and a query workload, AQP aims to generate a synopsis and uses the synopsis to answer an online query. For example, suppose the queries w.r.t column CustomerID and Profit are frequently used in a query workload, the system builds synopses offline for the two columns and uses them to answer a query. The system uses the error bound and confidence to determine the synopsis size, generates the synopsis, and utilizes the synopsis to answer the query. For example, given a query

```
SELECT AVG(PROFIT) FROM O
WHERE CUSTOMERID =}c'_4
ERROR ≤ 5% and CONFIDENCE ≥ 95%
```

The system can use a pre-computed stratified sampling method [7] (i.e., selecting more samples from big groups and enough samples from rare groups) to select samples, e.g., tuples $o_2, o_6, o_7, o_8, o_{10}, o_{12}$, to answer the query. Then the query can be effectively answered, because the tuples in rare groups will be considered.

Existing AQP techniques can be broadly categorized into two categories:

1. Online aggregation. It selects samples online and uses these samples to answer OLAP queries.
2. Offline synopses generation. It generates synopses offline based on a-priori knowledge on the data statistics or query workload, and uses these synopses to answer OLAP queries.

The goal of AQP is to find an approximate answer as close as to the exact answer very efficiently. There are three main challenges. The first is to select high-quality samples (or generate the synopses) to have an error-bound guarantee. The second is to support queries with join predicates. The

third is to support distributed computing, e.g., how to collect samples from different nodes in a cluster.

Online AQP Methods To address these challenges, online aggregation focuses on computing representative statistical summaries and devising effective algorithms to support online aggregation with no assumption on future queries. As conventional naive random sampling (see Case 1) only produces good results when the data distribution is uniform, many sampling methods [22] and error estimation algorithms [6] have been proposed to get more accurate results during query processing. For online interactive queries, the system iteratively selects more examples and uses the current examples to answer queries. Note that the system can incrementally use the samples to compute the results and does not need to compute the results from scratch. When the analysts are satisfied with the result, they can stop the query processing. We will discuss more details in Sect. 2.

Offline AQP Methods Offline synopsis generation method generates synopses offline and uses these synopses to answer online queries. These methods need a-priori knowledge about the dataset and future queries. Offline synopsis methods include workload-free synopsis and workload-aware synopsis. The former selects uniform samples or stratified samples by analyzing the data [2, 21]. The latter selects query-driven samples [7], which generates a synopsis based on the previous queries. Given a query workload, the query-based-method generates a synopsis for each query and uses the synopsis to answer the future queries. This method has limited ability because it only answers the queries falling in the query workload. To address this problem, a query column set (QCS)-based method is proposed, which generates a synopsis for each QCS and uses the synopsis to answer the future queries whose columns are contained in the QCS. There are several query-driven methods, including pre-computed sampling-based approximate query (`PSAQ`) which needs to make assumption on QCS or queries, `Histogram` [88], `Wavelet` [46], and `Sketch` [14]. The advantages of these techniques are that the results are more accurate on skewed data, and the query processing is fast (as they do not need to on-the-fly select samples), but they have some limitations. First, they cannot support general queries, especially the complex nested queries. Second, they involve too much storage to store the synopses. We will discuss more details in Sect. 3.

Comparison of AQP Methods We compare existing AQP methods in Table 1. Online aggregation method (OLA) does not need a-priori knowledge of the data and queries, and it on-the-fly selects samples but may fail to provide quality guarantee for skewed data. Offline methods `PSAQ`, `Histogram`, `Wavelet` and `Sketch` need to know the queries or data in advance, and generate synopses offline. They can support skewed data well but involve large space to store the synopses.

AQP on Complex Data Besides relational data, AQP techniques [106] can also be used to support other complex data, e.g., spatial data and trajectory data. We discuss how to use AQP to support these complex data in Sect. 4.

New Applications on AQP Besides OLAP on relational data, AQP can also be used to enhance other applications, e.g., data visualization [86] and data cleaning [105]. For example, in data visualization, users would like to see the approximate result of the ratio of population in each of the state in the USA as a pie chart very efficiently, rather than waiting for minutes for an exact answer. We will discuss how to use AQP to support new applications in Sect. 5.

AQP Systems Many commercial business-critical systems support AQP, e.g., Oracle [102] and Windows Azure [19]. The challenge of building AQP system is to design effective offline indexes or summaries (by analyzing data on distributed systems), devising effective query plan during the query-time and finding effective algorithms. We will discuss well-known AQP systems in Sect. 6.

Contribution In this paper, we survey a wide spectrum of work on approximate query processing as shown in Fig. 2. We review both online aggregation and offline synopses techniques, summarize the challenges and provide the techniques to address these challenges. We also review existing AQP systems and AQP techniques on complex data types and new applications. We provide emerging challenges and opportunities in AQP.

Paper Structure The structure of this paper is organized as follows: We first introduce cutting-edge online AQP methods in Sect. 2 and offline AQP methods in Sect. 3. We study how to extend AQP to support complex data in Sect. 4 and support new AQP applications in Sect. 5. Besides, we review well-known AQP systems in Sect. 6 and provide emerging challenges of AQP in Sect. 7. Finally we conclude the paper in Sect. 8.

Difference with Existing Surveys Although there are some surveys [29, 78], they only focused on some aspects of AQP, but did not give a complete survey and did not cover most recent works. Cormode et al. [29] surveyed offline synopses of AQP including `Sample`, `Histogram`, `Wavelet` and `Sketch` [29]. However, it only surveyed the offline synopses but did not cover online AQP techniques. Besides, all the new techniques after 2012 were not surveyed and analyzed. There were three keynotes on AQP at SIGMOD 2017. Surajit focused on online aggregation [22]. Mozafari [78] emphasized on new challenges and opportunities, including interface, effective query planning and the theories of database learning. Tim Krastra focused on their newly built interactive data exploration system IDEA [66].

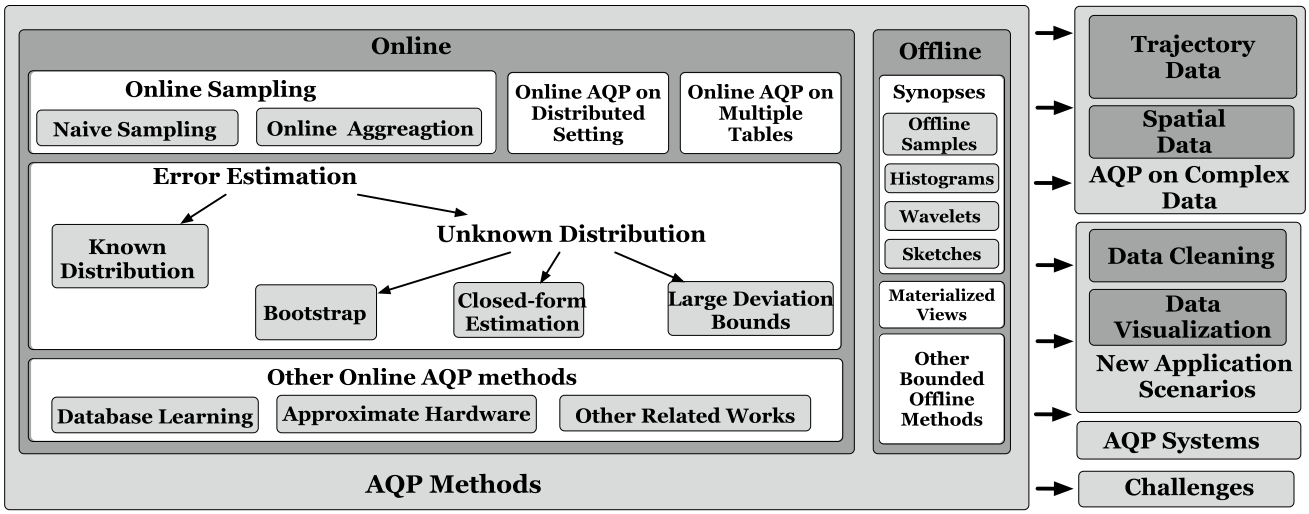

Fig. 2 AQP overview

Table 1 Comparison of existing AQP methods.

	Query	Pre-computed information	Skewed data
OLA	No	Index or data distribution	×
PSAQ	Queries or QCS	Synopses	√
Wavelet	Queries	Synopses	√
Histogram	Queries	Synopses	√
Sketch	Queries	Synopses	√

× Not support, √ support such case

2 Online Aggregation Methods

In this section, we survey the online AQP methods. The basic idea is to first select a sample **S** and then use **S** to estimate the results. We introduce how to select **S** in Sect. 2.1 and then discuss how to use **S** to estimate the error bound in Sect. 2.2. We present how to support multiple tables in Sect. 2.3 and how to work in distributed setting in Sect. 2.4. Finally, we discuss other online AQP techniques in Sect. 2.5.

2.1 Online Aggregation

Online Query Sampling techniques are widely used to support approximate query processing [84, 101]. Given a time constraint in an AQP query, a sample size can be computed by estimating how many samples the system can process within the time constraint. Then, the sampling techniques can be used to get a set of samples and the selected samples are used to answer the future queries. As the distribution of many real-world datasets are proved to be uniform-like distribution and Gaussian-like distribution, early online sampling methods use random sampling to select samples [83, 84, 88, 101]. Note that given the data-distribution assumption and random sampling can provide users with a confidence interval in most cases. For example, Case 2 selects more tuples than Case 1 and thus has a tighter confidence interval than Case 1. As many datasets satisfy the assumption, most existing systems support the sampling operator, such as Oracle and Spark SQL (see [22] for details).

The limitation of uniform sampling is that it has poor quality on skewed data. For example, it may not select samples for rare groups that have few tuples. In Case 1, if we want to know the average profit of the orders from c_4 instead of c_1, as there are only 3 tuples for customer c_4 in the dataset, randomly sampling 4 tuples may fail to select any tuple of c_4.

Online Interactive Query Online aggregation (OLA) was proposed in [52], which interactively refines the approximate results during the query processing. OLA provides users with an interface to stop the query execution when users are satisfied with the current answers. The accuracy becomes higher as OLA accesses more tuples in the database system. The system randomly selects tuples with or without replacement and computes an approximation based on the tuples seen so far. Then, it incrementally selects more tuples in every iteration. If the sample is bigger, the system can get a

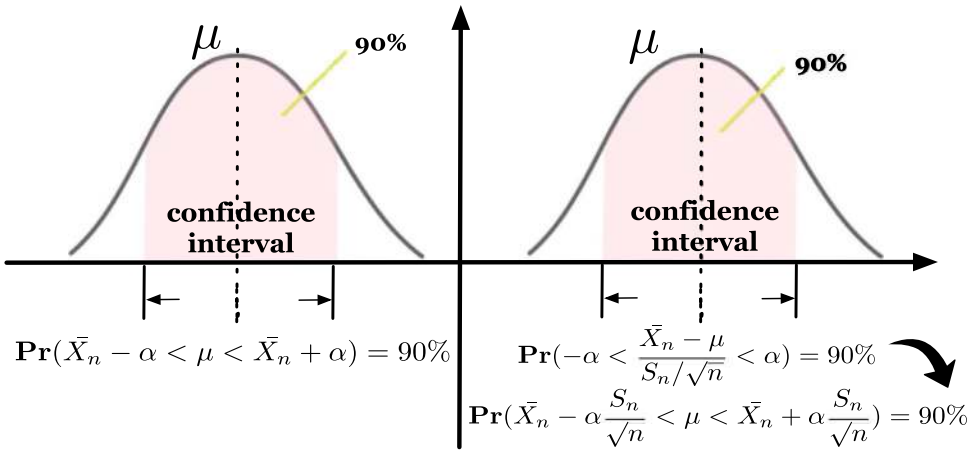

Fig. 3 Example of confidence interval

more accurate answer and tighter confidence interval. When users are satisfied with the current answer or impatient to wait, they stop the query processing. For example, considering Case 2, the system selects o_1, o_9, o_{12} in the first second and computes an approximation, say [1530.0 − 360.0, 1530.0 + 360.0] with a confidence of 90%. Then it samples three more tuples in every second and in the fourth second, the system totally selects 12 tuples from o_1 to o_{12} and the answer is [1392.4 − 329.4, 1392.4 + 329.4] with a confidence of 90%. If the analyst is satisfied with the current answer, she can stop the query processing.

Many techniques are proposed to accelerate the online query processing. For example, `Pf-ola` [93] aims to make online aggregation in parallel where the estimated results and corresponding confidence bounds are continuously refined based on the selected samples during the query processing. These parallel techniques will avoid wasting extra time for error estimation during the query processing. `G-OLA` [111] is an online aggregation architecture which can deal with arbitrarily nested aggregates using efficient delta maintenance techniques. `G-OLA` randomly partitions the dataset into smaller uniform batches, by computing a "delta update" [111] on each mini-batch of data. Then by carefully partitioning the intermediate results of nested queries, it can iteratively refine the query results.

2.2 Error Estimation

The confidence interval is widely used to estimate the result quality in most of the random-sampling methods [2], where each confidence interval gives users a numerical interval and a corresponding confidence based on the statistical theory. Initially, a set **S** of samples is computed based on sampling techniques in Sect. 2.1. Then if the data distribution is known in advance, **S** can be utilized to estimate the distribution and then the error can be estimated based on the distribution (Sect. 2.2.1). If the data distribution is unknown, it needs to first estimate the distribution of sampling data and then estimate the error (Sect. 2.2.2).

2.2.1 Error Estimation with Known Distribution

For real-world datasets, many datasets follow the normal distribution and many existing studies also assume that the data follows normal distribution. If we have a-priori knowledge about the data distribution or have a big enough sample to get the distribution, then this is a classical statistical problem—parameter estimation. We take computing the aggregation `AVG`(**S**) on a normal distribution as an example (see Figure 3).

Known Variance If we know the variance of the distribution, we can easily use the Gaussian distribution model N(μ, σ^2) to compute the confidence interval easily as shown in the left hand of Fig. 3.

Unknown Variance If we do not know the variance, we can formalize it as a *t*-distribution, then we can compute the confidence interval as shown in the right hand of Fig. 3. In such case, the bigger the sample size, the more information we will maintain about the population, then we can compute a smaller interval or higher confidence, i.e., when the sample size n is bigger, the length of confidence interval $2 * \alpha\frac{S_n}{\sqrt{n}}$ will be smaller where S_n is the standard deviation and α is a predefined statistical parameter, and thus, the answer will be more accurate.

2.2.2 Error Estimation without Known Distribution

To estimate the distribution of `AGG`(**S**), there are mainly three methods, Bootstrap, closed-form estimate, and example, when computing SUM.

Bootstrap It aims to get multiple samples. For each sample **S**, the estimated values of `AGG`(**S**) can be computed and these values compose a distribution which can be used to estimate the aggregation result. Then a confidence interval

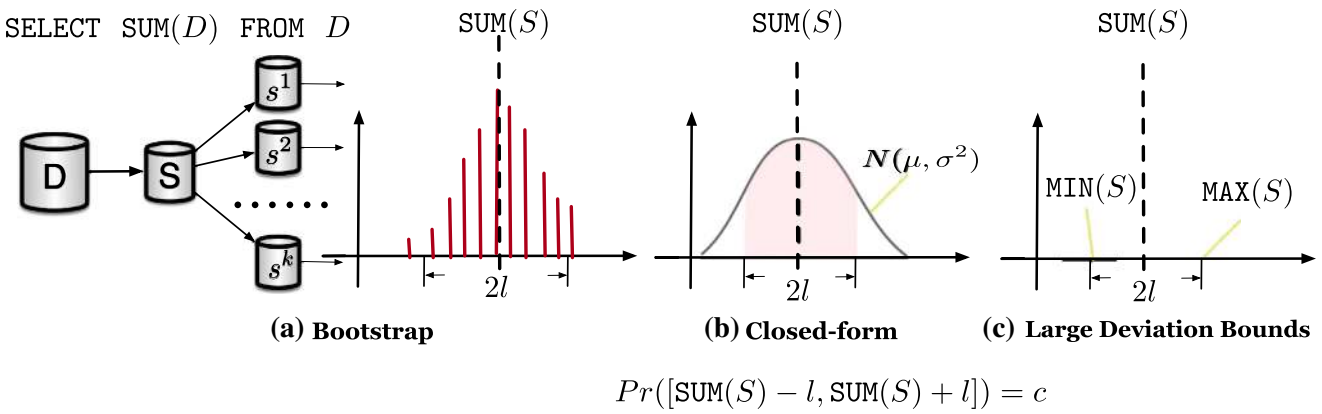

Fig. 4 Example of Error Estimations

is computed by Pr([AGG(**S**) − *l*, AGG(**S**) + *l*])=*c* to estimate the aggregation answer where 2*l* is the length of confidence interval, Pr is an estimation function (e.g., probability density function) of the distribution of AGG(**S**), and *c* is the confidence. Then the challenge is how to estimate AGG(**S**) and the distribution of AGG(**S**).

To know the distribution of AGG(**S**), we can arbitrarily sample many times (e.g., 10,000) from the population **D**, and the inferential statistics of the sample can be used to estimate the sampling distribution of AGG(**S**), e.g., standard error is an estimate of the standard deviation of that distribution. However, it is too expensive to sample many times from the whole population **D**, or even infeasible because the population is unknown and there is only one sample **S**. To address this problem, resampling such as Bootstrap is proposed to estimate the result error [82]. The concept of Bootstrap is well known in statistics for more than half a century, which has been used to estimate error in relational databases [90]. Recently Bootstrap is borrowed to estimate errors of AQP [113].

The key idea of Bootstrap is that, in order to use **S** to replace **D**, one can also draw samples from **S** instead of **D** to compose the distribution of AGG(**S**). However, if it draws samples from **S** for too many times, it is still expensive. In practice, it aims to draw reasonable number of resamples, e.g., 100. For example, in Fig. 4a, we draw the samples from **S** for *k* times and compute a distribution, then we can compute a confidence interval based on the bootstrap distribution. Interested readers are referred to [53] for more details on Bootstrap.

Bootstrap has no assumption on the data distribution and is suitable for most of the queries. However, Bootstrap has two limitations. First, most Bootstrap methods need thousands of resampling, and it is time consuming for resampling too many times. Second, Bootstrap may fail to estimate the sampling distribution when the aggregation function is sensitive to rare group (e.g., MAX) or the size of **S** is too small.

Closed-form Estimation In probability theory, the central-limit theorem (CLT) establishes that, for independent random variables, the normalized sum tends toward a normal distribution (informally a "bell curve") even if the original variables themselves are not normally distributed.[1] Thus, the distribution of AGG(**S**) can be approximated as $N(\mathtt{AGG}(\mathbf{S}), \sigma^2)$, where σ can be computed by the mean squared error Var(**S**) [20]. This method is known as closed-form estimation [6] as shown in Fig. 4b. Computing Var(**S**) for a small dataset **S** will be faster than the brute-force resampling. However, this method can only work for COUNT, SUM, AVG, VARIANCE but fail to deal with the queries whose variance is hard to compute such as MAX, MIN or user-defined functions.

Large Deviation Bounds A large deviation bound [75] wants to know the worst case of the confidence interval by estimating a value w.r.t. the "sensitivity to outliers." This value mainly depends on the aggregation function AGG. For example, when computing SUM, the bound will be MAX and MIN as shown Fig. 4. In practice, it computes a bound much bigger than the real width of the sampling distribution.

2.3 Online Aggreagtion on Multiple Tables

Many OLAP queries contain join predicates. For example, if an analyst wants to know the average profit of the orders from the MA state in Fig. 1, she should join the tables **O**, **C** and **ST**.

One major challenge of approximate online aggregation of multiple tables is that the join process is complex and time consuming, and if the dataset is too big, then the join operation takes millions of computation in real-world applications.

[1] https://en.wikipedia.org/wiki/.

OLA can be extended to support multiple tables on relational databases [49, 72]. Especially, the join queries are hard to predict in advance, and OLA is a typical method to solve such problem [23]. A naive method is to first join all of the tables and then using OLA on the joined tables to estimate the result. Obviously this method is too expensive since it is costly to join the tables. To address this problem, "ripple join" [32, 47, 57] selects a sample on each of tables and blindly join them; however, many tuples cannot be joined and some groups will be missed. Another idea is to generate a sample of the first table in a join pipeline and then select samples for other tables following join operations [3]. But it is not only time consuming but also performs bad if generating a bad sample for the first table. A most recent study proposes "Wander Join" [70–72] using online aggregation via random walk which is effective for the join operator. In a random walk process, it blindly takes tuples in different tables in the input dataset and aims to make them easily joined and get the join output. "Wander Join" designs algorithms to select the tuples which would be joined in every next step rather than blind "ripple join" [32, 47, 57]. For example, if it randomly selects samples from **O**, **C** and **ST**, it may miss the tuple st_1 in **ST** and fail to meet the query condition mentioned above. Instead, Wander Join first finds st_1 in **ST** and randomly selects tuples that can be joined in **C**, e.g., c_1, and randomly selects tuples in **O**, then it can join these tuples and compute an approximation of the profit of the orders from the MA state. However, the generation of such joined output is very expensive. To find more effective OLA methods to support join operator in distributed setting is still an open problem.

Note that if the join size is known in advance, we can get a better join plan. However, it is too expensive to compute the exact join size and we should estimate the join size. Existing techniques aim to join the synopses of each of the tables to estimate the join size and reduce the time of scanning the whole dataset [104]. A most recent study proposes a novel two-level sampling [104] by combining "independent Bernoulli sampling", "Correlated sampling" and End-biased sampling. One can use two-level sampling to estimate join size more accurately which outperforms other existing studies.

2.4 Online AQP in Distributed Setting

In many applications, the data is so large and have to be stored in a distributed cluster. For complex queries with join conditions and nested subqueries, we need to scan the dataset multiple times. Especially for join, it may need to transmit the data across cluster nodes, leading to high communication cost. For example, a user would like to know the origin place of the brands of the top-5 most popular smart phones in Amazon. To answer the query, the system should scan the tables of orders, brands and origins which are stored in distributed servers. It needs to scan the tables many times to join the tables. Moreover, it may also need to join the tables across different nodes.

There are two challenges of AQP in distributed setting. The first is to avoid scanning the data multiple times to reduce the I/O cost. The second is to reduce the communication cost among distributed nodes. Early sampling work [48] focuses on finding efficient sampling methods in distributed setting. However, it is proved to be inefficient when the predicates in a query are complex. Then, `EARL`[69](Early Accurate Result Library) generates online uniform samples from HDFS and uses Bootstrap to incrementally evaluate the accuracy computed so far. It can support complex queries well; however, it does not consider the skewness of dataset distribution. `ApproxHadoop` [44] assumes that the datasets are uniformly distributed in clusters but if the sub-datasets are not uniformly distributed, it cannot work well because random sampling on the cluster will generate a biased sample. To overcome these limitations, `Sapprox` [114] constructs probabilistic SegMap (i.e., collect the distribution of subsets) of segments offline and uses these results to generate online sampling plan.

Quickr is proposed to get samples in a cluster [60, 61], which can select a good sampling plan. It combines three sample operators together, uniform sampler, distinct sampler (selecting samples for each distinct group) and universal sampler (selecting samples for join results among multiple tables). By lazily sampling in the first pass, 'Quickr" scans the dataset spreading over clusters efficiently. To estimate the error, it handles different aggregate types using different strategies. (1) If a sampler immediately precedes the aggregation and group by operator, it extends the well-known Horvitz–Thompson (HT) to estimate the exact answer and uses central-limit theorem to compute the confidence interval. (2) otherwise, it uses the theory of "dominance transitivity" to transfer to case (1).

2.5 Other Online AQP Methods

2.5.1 Database Learning

In traditional database systems, previous query answers are not used to process future queries. If we can use previous query results to answer future queries, we can efficiently estimate an answer. Motivated by such assumption, a new AQP method called "Database Learning" (DBL) has been proposed [87].

DBL uses statistical features (e.g., computing the correlation parameters and covariances between all pairs of past queries snippets) of the dataset to train a model to represent underlying data distribution. When a sample is used to answer the queries, it is hard to know the distribution of

the whole data. However, with the help of previous query answers, one can know more about the distribution and then infer answers of new queries based on trained statistical model. The more precise the model, the less need for actual data, the smaller the sample, and consequently, the faster the response time. By increasing previous queries, one can use smaller sample and the system will become smarter and faster to process queries.

Note that DBL is different from using pre-computed views to answer new queries or QCS-based offline synopses which generate summaries for visited query columns. Views aim to use pre-computed results to exactly or approximately answer new queries. For example, if we know the query result of the average profit of orders from the MA state in Fig. 1, we can exactly answer the average profit of orders from c_1 and c_4 and approximately answer the average profit of orders from males in MA. A QCS-based offline synopses shown in Case 3 is based on the assumption on query column sets. However, during the query process, DBL learns model from past observations of past queries results (i.e., training sets) and trains a model, and when a query comes, it uses the trained model to infer the query result. For example, if we know the results of past queries of the profit of orders from c_1, c_2, c_3, c_4 in the MA state and NY state, we can use the results as training set to train a model of relation **R**. When we need to know the profit of orders from the WA state, we can sample tuples from **O** and use the trained model to compute the approximation result.

The limitation of DBL is that if the past query result is not accurate, then the quality of a training set of an online learning become worse. Thus, it will become worse and worse and finally it may mislead the approximate result.

2.5.2 Approximate Hardware

As the computation of a database system is supported by hardwares, some work aims to design new types of hardwares to trade-off the efficiency and accuracy of queries on database, reduce the energy usage and increase energy efficiency. Comparing with conventional AQP techniques, this type of technique does not need any pre-computed synopses or online approximate query scheme. In this subsection, we introduce database engines using "Approximate hardware" which mainly consists of "Approximate CPU", "Approximate Memory" and "Hybrid Hardware" and new applications of such techniques as well.

Approximate CPU "Approximate CPU" uses just part of the resources of CPU to accelerate the processing of calculation and saving a lot of electric energy, e.g., float computing. For example, a floating-point arithmetic-reduced method [103] explores ways of reducing floating-point computing power consumption by minimizing the bitwidth representation of floating-point data. Analyses of several programs that manipulate low-resolution human sensory data show that these programs achieve a significant reduction in bitwidth while not sacrificing accuracy. `Enerj` [96] is developed as an extension of Java that adds approximate data types. `Enerj` also proposes a hardware architecture that offers explicit approximate storage and computation. When using `Enerj` to program, one can use approximate data types for the computations that only need approximation instead of exact answers. Several applications to `Enerj` show that these extensions are expressive and effective because `Enerj` uses just a small number of annotations but leads to significant potential energy savings at expense of very little accuracy.

Approximate Storage Sampson et al. [97] propose two mechanisms to do approximation on solid-state storage. (1) The first allows errors in multi-level cells by reducing the number of programming pulses used to write them. (2) The second mechanism mitigates wear-out failures and extends memory endurance by mapping approximate data onto blocks that have exhausted the hardware error correction resources. It is claimed that [97] can improve the performance, lifetime, or density of solid-state memories by conducting some simulation experiments.

Hybrid Hardware `ApproxiDB` [51] is the first hybrid data management system based on a hybrid hardware including approximate hardware and precise hardware, and thus, it not only supports approximate query processing but also can return an exact answer. When the system can answer the query with an exact answer within the time constraints, it uses the precise hardware; otherwise, it uses the approximate hardware. `ApproxiDB` [51] is the first work that proposes the concept of "Approximate Hardware" which summarizes existing "Approximate Hardware" works and concludes the challenge and opportunity of "Approximate Hardware" well.

Applications of Approximate Hardware There have been many applications using "Approximate hardware". For example, Chen et al. [25] conduct a study of three common sorting algorithms on approximate storage. They propose an approximate-refine execution mechanism to improve the performance of sorting algorithms on the hybrid storage system to produce precise results. Moreover, a green database framework is proposed in [13] which helps the query optimizer select plans that meet the high-performance requirements with lower energy during query processing based on the resource consumption patterns.

2.5.3 Other Works

DAQ `DAQ` [92] is a variant of OLA which borrows ideas from probabilistic database and iteratively uses the high-order bits of numerical data to compute the approximation. For example, a `DAQ` scheme stores numbers in column PROFIT in Fig. 1 using "Bitsliced-Index" [92]. If we query

MAX on the column PROFIT, DAQ checks the first bit of the numbers in the 16 tuples of **O**, if there is only one tuple whose first bit is '1', we get the exact answer rather than travel all the bits (e.g., 32 bits); otherwise, we check the next bit until finding the maximum one. Unfortunately, such technique can only support simple queries over numerical columns (such as SUM and AVG) but cannot support general SQL queries.

Sample+Seek As most of previous OLA algorithms cannot work well for the rare groups during online sampling, "sample+seek" [31] proposes to design different plans for big groups and rare groups, i.e., "sample" (uniformly or measure-biased sampling) for big group and "seek" (building index) for rare groups. This work introduces a new precision metric, called distribution precision to provide error guarantee for queries. This work also provides a measure-biased sampling method to support any aggregation that can be estimated from random samples within a user-given error bound.

Incremental AQP Galakatos et al. [42] propose an AQP formulation that treats aggregate query answers as random variables to enable reusing of approximate results with reasoning about error propagation across overlapping queries. When a new query is coming, it finds previous queries which have common attributes and query conditions with the query, thus uses these results to refine the approximation. To support rare groups, this work presents a low-overhead partial index and corresponding rewriting rules based on probability model to response the queries in an interactive speed.

3 Offline Methods

If the query workload will not dynamically change, we can build offline synopses based on previous query workload and use these synopses to answer future queries efficiently. In this section, we survey four main synopses, pre-computed samples (PSAQ), Histogram, Wavelets and Sketch. Then, we introduce other recent offline methods with bounded guarantees.

3.1 Pre-computed Samples

Using offline samples as synopses for AQP has a history of 30 years in the database research community [84]. A well-known method is pre-computed sampling-based AQP, denoted as PSAQ, which generates samples offline and uses these samples to answer online queries. Note that PSAQ has an assumption that the query workload is relatively stable, i.e., the queries will not be dynamically changed.

Query-Based PSAQ A naive method builds a synopsis for each query in the workload and uses the synopsis to answer the future queries. For example, given the query in Case 1 which computes the aggregation result for customer c_1, we can build a synopsis using the samples of this query, e.g., o_1, o_3. Then, we can use the synopsis to answer the future queries which contain the same column but may use different aggregation functions. A challenge here is given a space budget, how to select the queries to build offline synopsis. One can also merge synopses of different queries to reduce the synopsis size. This method has two limitations. First, if the query workload is large, this method will generate many synopses. Second, the synopsis of a query can only be used to answer this query but cannot answer other queries. For example, it can only answer queries for customer c_1 but cannot answer queries for other customers.

QCS-Based PSAQ To address the above problems, query column set (QCS)-based PSAQ is proposed [7]. The column set of a query is the set of all columns in the query (including select, where and group clauses). This method groups the queries based on the column sets in the queries, and the queries with the same column set will be in the same group. Then for each group, the method selects samples for the columns in the group. Next it can use the synopsis to answer queries with the same QCS (or the queries whose columns are contained in the QCS). For example, suppose many queries contain columns CustomerID and Profit in the query workload, it does not need to build samples for each of the queries. Instead this method builds a sample for column (CustomerID, Profit), and thus can save a lot of space. BlinkDB [7] studies how to select samples for each QCS and uses the samples to answer online queries in distributed file system. BlinkDB also tries to share samples among different QCSs. For example, a sample for column (CustomerID, Profit) can be used to answer the queries for column CustomerID.

Sample Selection Comparing with online sampling, offline sampling can spend more time to select high-quality samples. Besides, offline methods can use a-priori knowledge of the whole dataset and they also need to store pre-computed samples. Note that the size of groups and the values in each group may be highly skewed, making many traditional uniform-sampling-based methods unreliable. Some stratified sampling(AQUA [2], START [21], BlinkDB [7], Babcock [11]) are proposed to deal with sparse data. The common idea of these work is to select more tuples for big groups meanwhile selecting enough tuples in rare groups (which may be lost in a random sample).

Error Analysis To provide high confidence, PSAQ also uses closed-form estimation and Bootstrap [6, 90, 113] to diagnose the results by using multiple samples via resampling. The method for error analysis is similar to online sampling, but they can spend more time to analyze the errors by thousands of resampling offline.

Multiple Tables Supporting queries with join predicates is also widely studied [3]. The techniques are similar to those in online sampling.

3.2 Histograms

`Histogram` summarizes a dataset and divides it into multiple buckets based on values in a numerical column [88]. For each bucket, it computes the most representative statistics which can be used to reconstruct the value of the whole dataset in this bucket, e.g., store the lower and upper bound of this bucket and count the numbers in this bucket. `Histogram` has been widely studied and incorporated into commercial relational databases which can be easily constructed and used for estimation [28, 91]. Histograms include equi-depth and equi-width histograms. The former has the same bucket size and the latter has the same bucket width (i.e., the difference of the maximal value and minimal value in each bucket). For example, the numbers of column Profit in **O** is {1120, 1170, 1230, 1250, 1290, 1350, 1417, 1460, 1470, 1560, 1630, 1673, 1732, 1890, 1983, 2000}, an equi-width `Histogram` [29] will split the numbers into buckets with the same length (e.g., 200). Then it can be divided into {(1100, 1300], 5}, {(1300, 1500], 4}, {(1500, 1700], 3}, {(1700, 1900], 2} and {(1900, 2100], 2}. For a query with the `SUM` function on the attribute, one can compute $\frac{1100*5+1300*4+1500*3+1700*2+1900*2}{16} = 1400.0$ to approximate the answer. An equi-depth histogram selects bucket boundaries so that each bucket contains the same number of data points. For example, if the bucket depth is 4, the column Profit in **O** will be divided into {(1100, 1250]}, {(1250, 1460]}, {(1460, 1700]} and {(1700, 2000]}.

The major challenge of a `Histogram` method is to find appropriate algorithms to decide the buckets. The bucketing strategy should consider both the number of buckets (the less the better) and accuracy (the higher the better). Besides equi-width and equi-depth [29], many other types of `Histograms` such as Singleton-Bucket `Histogram` [54, 55] and Maxdiff `Histogram` [91] are also widely studied. More complex methods have been designed to find bucketing scheme to trade-off efficiency and accuracy. A recent work proposed a near-optimal algorithms based on `Histogram` for describing the distribution of dataset [1]. In addition, multi-dimensional `Histograms` are proposed to support different applications. For example, `Digithist` [100] combines multi-dimensional, one-dimensional `Histograms` and grids to provide a tightly error-bounded `Histogram` for multi-dimensional data.

The drawback of `Histogram` that it only supports numerical columns and cannot support complex SQL queries accurately, e.g., multiple attributes range query. Moreover, it will cost too much space to store a synopsis for each column. The advantage is that `Histogram` can process queries instantly and has quality guarantees.

3.3 Wavelets

`Wavelet` is conceptually close to the `Histogram`. `Wavelet` transforms the data and aims to compress the most expressive features in a `Wavelet` domain but `Histogram` simply produces buckets that are subset of the original data. For example, if the numbers in column **C** in **T** are {1, 3, 4, 4}, a Haar-wavelet transform (HWT) decomposes it as {2, 4} with the loss $-1, 0$, then HWT compresses it again as {3} with the loss -1. By storing 3, {-1}, {$-1, 0$}, we can decompress it to get the original data set. By storing 3, {-1}, we can approximately represent the original dataset as {2, 2, 4, 4} with loss $1, -1, 0, 0$. Then, if we query `SUM` of this numerical column, we can decompress it to get {2, 2, 4, 4} and use it to compute the value instead of {1, 3, 4, 4}. There are many variants of `Wavelet` that have been widely studied in recent years. HWT is the most widely studied `Wavelet`, which selects the largest HWT statistics in a synopsis that provides the L_2 error for data decompression [99]. Recent work focuses more on Non-Euclidean `Wavelet` [46, 63–65]. Mytilinis et al. [80] developed parallel algorithms to generate Wavelets within an error bound.

3.4 Sketches

`Sketch` [43] models a numerical column as a vector or matrix and transforms the data by a fixed matrix to construct the synopsis. For example, the well-known bloom filter can be seen as a special case of `Sketch` which maps data into a vector of bits. `Sketch` is not suitable for general relational database but performs well when dealing with streaming data where the sketch summary must continually be updated quickly and compactly. `Sketch` is not only fast but also easy to parallelize and can provide the high approximation accuracy. `Sketch` has two main categories. The first is "Frequency-based sketch" [14] which focuses on the frequency distribution of the original dataset. The second is "Distinct-value-based sketch" [41] which counts the distinct values in a given multi-set. Different from other synopsis,`Sketch` has also been used successfully to estimate the answers of `COUNT` and `DISTINCT` queries [29].

For example, for `COUNT` queries, a `sketch` may initialize a matrix **C** with $d \times w$(d and w should be tuned to proper values) zeros, for each item t in the data stream, for each integer number j from 1 to d, it increases $\mathbf{C}[j, h_j(t)]$ by 1 where h_j is a hash function. Then, when a `COUNT` query comes and it wants to count the number of t in a data stream, it first sets the current answer a as the biggest number in matrix **C**, and then it iteratively checks whether $\mathbf{C}[j, h_j(t)] \leq a$ from 1 to d.

If so, $a = \mathbf{C}[j, h_j(t)]$. After d iterations, it can find the exact answer a.

To support distributed and streaming data, a most recent `Sketch` technique [24] formulates a bias-aware linear sketching and recovery problem, and proposes algorithms to generalize the widely used Count-Sketch and Count-Median algorithms. Due to its linearity, it can be easily implemented in the streaming and distributed computation models. [89] proposes a Count-Min-Log `Sketch` method to improve the average relative error of Count-Min-Sketch within bounded storage via logarithm-based, approximate counters instead of linear counters. **CQF** [85] proposes the counting quotient filter (CQF) to support general computing operators of approximate membership query. These techniques can be easily extended into database systems.

`Sketch` can be used in many other fields, e.g., NLP [45], since `Sketch` can be used to count the frequency of words, conduct Pseudo-Words evaluation, find semantic orientation of a word, and compute distributional similarity in NLP domain. `Sketch` is also good at dealing with real-time system such as financial data streams where the data dynamically changes. More details of different types of `Sketch` can be found in [27, 29].

3.5 Materialized Views

Materialized views are also related to AQP, which generates views of some given queries and utilizes the views to answer future queries. The difference is that materialized views maintain all the data but not some samples. The materialized views are lossless, but the synopses usually have errors.

Materialized Views A materialized view [50, 67] is a pre-computed query result for some important query. When the query is stable over time, we can simply store the results of frequently used queries and use them to support future queries. The pre-computed views can be used to answer future queries (as exact answers of the queries or a subset of the answers) which have been widely used in previous query workload [8]. The most important problem here is how to use views to rewrite new queries [81]. The technique of materialized views still needs to be studied. For example, pre-computing views too much will waste resources a lot. Armbrust et al. [10] describe a scale-independent view selection and maintenance system, which uses novel static analysis techniques that ensure the created views will not become scaling bottlenecks. Moreover, a "materialized sample view" [58] is a materialized sample from the query result for some important queries which are also widely studied.

Data Cubes A data cube stores statistics for specific queries which pre-compute a list of values in the form $g(1), g(2), \ldots, g(M)$ where g is a function. For example, values in a numerical column are {12, 13 ,20, 18, 10} and values in ID column are {1, 2, 2, 3, 1}. If the query is `SUM`, g is the function of computing frequency. Then a data cube for `SUM` on table `T` is {1:22, 2:33, 3:18}. It can support many complex queries, e.g., `WHERE` clause. Many studies on different types of data cubes have been studied for supporting OLAP. A recent work [108] provides a complete set of techniques for probabilistic data cubes. Cube can also be used on ad-hoc interactive analytics over large datasets in distributed clusters [56, 59] and exploring machine learning results [9].

3.6 Other Offline Methods

As many systems require high accuracy, high speed and limited resource, existing work aims to find effective AQP techniques for providing an approximate answer within bounded error, bounded resource, or bounded response time.

3.6.1 Bounded Resources

Approximate query processing with bounded resources mean that the storage or memory is limited so that only a limited number of records can be used to approximately answer the query. Using offline computing to help decide the query plan and accelerate query processing are much practical for resource-bounded processing. For example, Cule et al. [30] studied the space-bounded query approximation. Fan et al. [16–18, 35, 36, 38] proposed a series of work based on the newly proposed concept "bounded evaluability," which can answer a specific class of queries by accessing only a subset of the whole dataset with bounded number of tuples with the help of indices built on application-induced cardinality constraints.

[35–37] make theoretical analyses on how to utilize a small subset to support queries over the whole set. [37] is a step toward understanding the tractability of queries in the context of big data by providing a formal foundation in terms of computational complexity. In [35], authors aim to study the bounded evaluability which can help decide the query plan, i.e., compute the exact answers or compute approximate answer using envelopes and bounded query specialization. [36] is the first work to formalize the notion of scale independence and study its properties. When a query **Q** is proved to be scale independent, the performance of processing query **Q** will not decrease when the scale of dataset **D** becomes bigger. [38] mainly focuses on how to implement these works on big graphs, which mainly provides approximate query processing algorithms for resource-bounded strong simulation, resource-bounded subgraph queries and resource-bounded reachability on big graph within bounded resources under access constraints. The authors evaluate the accuracy of approximate query processing on big graph data and use a small subgraph to answer graph queries within a

given bound which is a given ratio of the scale of subgraph and the whole graph.

[16, 18, 36] propose some effective algorithms to implement the techniques in [35] in real applications. [16] provides effective syntax for answering whether it is still possible to make practical use of bounded evaluability for answering relational algebra queries which is remained to be an open question, and it shows how to integrate the concept of "bounded evaluable" into real-world DBMS systems. [18] investigates effectively bounded conjunctive queries under an access schema, studies complexity of such problem and designs heuristic algorithms, which refine the concept of "scale independence" in [36].

Based on previous works, Fan et al. build resource-bounded scheme BEAS (Boundedly Evaluable Sql) [17] for querying relations within a given sampling ratio by either computing the exact answer if doable or giving an approximation by accessing no more than bounded numbers of tuples using "bounded evaluable" theories discussed above. Its novelty consists of access templates, a new accuracy measure, a resource-bounded approximation scheme and resource-bounded algorithms for answering most general queries with a deterministic accuracy lower bound. BEAS has been implemented on many industry systems.

3.6.2 Bounded Error and Bounded Time

Many systems need high precision which calls for bounded error and bounded time. [112] combines the strengths of closed-form analytic error and analytical Bootstrap method to provide bounded error for AQP systems. For sparse data, Yan et al. [109] use error-bounded stratified sampling to reduce the sample size. This technique relies on the insight that we can reduce the sampling rate with the knowledge of data distributions.

4 AQP on Complex Data

Besides relational data, many other complex data also become very large, and AQP techniques [106] can be extended to deal with these data. In this section, we take spatial data and trajectory data as examples to show how to enable AQP on these data.

4.1 AQP on Spatial Data

Smartphones and other mobile devices have generated huge amount of spatial and spatio-temporal data. There are many applications that require to support OLAP queries on spatial data, e.g., analyzing the number of cars in a business zone to predict traffic jam. The importance of spatial data analysis and aggregations is increasing and interactive exploration over these data has become a challenge. However, the cost of spatial data analysis and aggregation using the entire data is too expensive, especially on large data sets, which cannot meet the requirement of interactive spatial data exploration. Even traditional techniques such as R-tree and Grid index cannot support such big data either. Therefore, it calls for effective AQP techniques on spatial data.

4.1.1 Online Spatial AQP

An online AQP method on spatial data exploration is proposed in [98], which provides efficient spatial sampling for large spatial datasets. It models the problem of spatial sampling of large geographical tables as an integer programming formula and proposes a more efficient solution based on the spatial tree traversal of depth first search. First, it splits the geographical tables into subsets and builds a spatial tree to retrieve these data. When a query is coming, it travels the tree using depth first search and selects some samples to process the query.

Then, online aggregation on spatial data is proposed in [106]. It proposes novel indexing techniques, LS-Tree and RS-Tree which can retrieve the spatial data more efficiently. In [106], the algorithm travels the index tree from the root node, as getting more and more samples, various spatial analytics and aggregations with estimators can be applied in an online, interactive manner. In this way, it becomes more accurate and more reliable over time. Different from previous work, it considers the difference between "query-first-then sample" (which first finds the results satisfying the query condition and then get some samples to answer the query) and "sample-first-then-query." (Which first gets some samples and uses the samples satisfying the query condition to answer the query.) These algorithms are also suitable for both memory-based and disk-resident data sets and scales well toward different query and sample sizes. More importantly, the structure in [106] is dynamic, so that it can effectively handle insertions and deletions of the dataset.

4.1.2 Offline Spatial AQP

Queries on spatial data can also be quickly and approximately answered using offline techniques such as data summaries, sketches, and signatures. However, the drawback of these methods is that the accuracy is predetermined and will not improve over time. For some spatial applications, most queries are stable and we can compute results of these queries in advance. Thus, summary indexing techniques [110] can be used to answer queries on spatial databases and these

spatial indexing techniques only require linear space and extract summaries with an optimal or near-optimal query cost.

Most methods of spatial query approximation rely on quantitative usage, i.e., metric (distance based) information. And only a few methods consider qualitative information, such as topological relations and cardinal spatial relations. As `Sketch` is proved to be effective on geographic information systems, it is further investigated in the background of spatial AQP. Based on these considerations, [12] provides new types of queries that rely on qualitative relationships and discusses how to define query processing algorithms in metric space to handle qualitative information. More recent discussions of spatial databases can be found in [34].

4.2 AQP on Trajectory Data

The rapid development of location and acquisition technology promotes the generation of trajectory data that track the trajectory of a moving object, where each trajectory is a sequence of geo-located points. A wide range of applications can benefit from trajectory data mining, which brings unprecedented opportunities. In modern systems, there are various applications of trajectory data mining, e.g., frequent route discovery, location prediction, and mobile behavior analysis. The two key operations of trajectory data management are range query (e.g., finding all the trajectories which pass some given spatial ranges) and KNN trajectory matching (e.g., finding all the top-k nearest neighborhood trajectories of a given trajectory). Note that it is rather costly to support the two queries for large-scale trajectory data, and traditional R-tree or Grid index fails to deal with such big data, and moreover, they cannot meet the high-performance requirement of online trajectory aggregation.

An inverted index for spatiotemporal and trajectory data [73] is proposed which uses random exponential sampling (RIS) algorithms to estimate the answers with the guaranteed error bounds. It simply splits the trajectory into 3-degree grids and randomly samples some trajectories to answer the `COUNT` query. In order to further improve the scalability of the system, [73] extended the parallel random index sampling (CRIS) algorithm of the RIS algorithm to deal with multiple track aggregation queries to reduce time and space queries at the same time. These techniques are applied to the actual large-scale user trajectory database from "China Mobile" service providers to verify the effectiveness of the proposed sampling and estimation methods. However, this method cannot support complex queries in the trajectory database system. Designing schemas for AQP on trajectory indexing and retrieving is still an open problem. More details about trajectory query and trajectory mining can be found in [39] and [115].

5 AQP Applications

AQP can be used in many scenarios if the traditional exact methods cannot meet the high-performance requirement. In this section, we discuss two scenarios that can be enhanced by AQP, including data cleaning (Sect. 5.1) and interactive data visualization (Sect. 5.2).

5.1 AQP on Data Cleaning

Data is rather dirty, especially in big data era, and data cleaning and integration are rather important in many applications [26]. For example, in Google Scholar, we want to compute the average citations of database researchers. Since some researchers' Google Scholar pages contain publications that do not belong to them, it is incorrect to directly compute the average citations on the dirty data. A straightforward method first cleans the Google Scholar pages for every researchers and then applies the OLAP queries. Obviously this brute-force method is rather expensive. A smarter way is to utilize AQP techniques, which first cleans a sample data and then uses the sample data to compute the results.

The challenge is how to design an estimator for dirty samples. For a cleaned sample, we can use methods in Section 2 (e.g., confidence interval) to estimate the result. But for a dirty sample, the estimator has not been well studied. `SampleClean` [68, 105] aims to address this problem which only requires users to clean a sample of data, and utilizes the cleaned sample to process aggregation queries. `SampleClean` also proposes a statistical method which can use a cleaned sample to correct the bias of the query results over the dirty data. Furthermore, it uses a cleaned sample to directly estimate the query results of the cleaned data. Along the same idea, a recent work [67] efficiently cleans a sample of rows from stale materialized views and uses the cleaned samples to estimate the query results.

5.2 AQP on Data Visualization

In recent years, the performance of data visualization has became more and more crucial in commercial system (e.g., ad-hoc query). Data scientists rely on interactive data visualization to analyze the data. As the data size becomes large, traditional systems fail to provide fast interactive query result on large data. Some studies try to use AQP techniques on data visualization [62, 76, 94]. The motivation of data visualization is converting a query (e.g., SQL-like query) to a visualization result (e.g., bar chart) and AQP can generate

approximate answer over big data, and thus, using AQP on data visualization is meaningful and reasonable.

There are many AQP-based visualization methods recently [40, 62, 74, 76, 94, 107]. These methods mainly emphasize on how to provide users with online interactive visualization results and incrementally update the results using representative charts. We will discuss cutting-edge techniques in this subsection. Existing works on AQP-based visualization can be broadly classified into two categories, scatter plot approximate visualization and statistical visualization.

Scatter Plots In many data exploration tasks, the analyst needs an instant result of the spatial data distribution like scatter plots. This result can be useful for further decisions or the data pretreatment pipeline. For example, an analyst of a real estate company wants to train a model for predicting which domain in USA is worth investing. Then she needs to determine which features are used for an logistic regression. Thus, she may want to draw a heatmap of the house price of all the block with different colors (cheaper is lighter, more expensive is deeper). However, in some interactive tasks, dealing with such volume of data is challenging.

A recent visualization-aware sampling (VAS) [86] guarantees high-quality visualizations with a small subset of the entire dataset. VAS divides the whole map into small blocks, and uses stratified sampling to choose a set of tuples that minimizes a visualization-inspired loss function in each block. While existing sampling approaches minimize the error of aggregation queries, VAS focuses on using a loss function that maximizes the visual fidelity of scatter plots. It consequently selects more samples in each block and provides the users with a clearer visualization.

This sampling-based technique can also be used to solve special kinds of data exploration query. For example, ExploreSample [107] approximates scatter plots to solve an "object-centric" exploration query. In an object-centric exploration query, one may predict the performance of an NBA basketball player using the whole dataset of all the NBA basketball players, i.e., drawing a heatmap and judging whether it is an outlier. However, drawing such heatmap may consist of many potentially expensive aggregation queries over the entire database. Thus, ExploreSample selects a sample of the whole dataset and draws a heatmap to help predict the claims which is verified to be helpful.

When exploring on big data, one may pose many queries with aggregation functions. For example, an user wants to see a pie chart of the ratio of the amount of incomes of males and females in America. However, computing such amount of data is rather expensive to draw the pie chart. Thus, it is feasible to use stratified sampling to sample 5000 people in each state and use this result to draw a pie chart, because the analyst just wants to see an approximate ratio rather than an exact answer, i.e., 56 versus 44% makes no much difference with 55 versus 45% in a pie chart. Furthermore, building a visualization system with an interactive pattern will be better for data exploration. Statistical visualization mainly includes bar chart, pie chart and line chart, for better interactive visualization processing, existing works have made effort to using AQP techniques on these Statistical visualization [9, 40, 62, 74, 76, 94].

Bar chart A bar chart can simply make comparison among different groups of data. To interactively give an approximate bar chart, there are three state-of-the-art works. SampleAction [40] allows a user to formulate a query, and the system responds with a partial result, displaying a bar chart with confidence bounds. As the analyst waits, the system increases its sample size, narrows the confidence intervals and produces more precise results. By using different sampling strategy, Pangloss [76] uses an idea of Sample + Seek [31] which "sample" (uniformly or measure-biased sampling) for big groups and "seek" (building index) for rare groups. It incrementally loads more records into the sample to update the bar chart until either the confidence is higher than a predefined threshold or until a timeout. Pangloss can also be used for heatmap. Different from Pangloss, IFocus [62] mainly focuses on whether the comparison between different bars is correct, e.g., if bar A is higher than bar B on the entire data, then bar A should be higher than bar B on the sample. IFocus calculates the aggregation answer to draw a bar chart and computes the probability that bar A will be higher than bar B. When the bar A is higher than bar B and the probability of $A > B$ is higher than a threshold, it stops. Besides, FastMatch [74] is an end-to-end approach for interactively retrieving the bar chart visualizations which are most similar to a user-specified target, from a large collection of bar chart. The primary technical contribution underlying FastMatch is a probabilistic algorithm, HistSim, a theoretical sampling-based approach to sample a lot of bar charts and identify the top-k closest bar charts under "L_1" distance.

Line Chart A line chart is used to show the trend of a statistical parameter. Thus, the key idea of approximate line chart is the trend should be correct (up or down). Thus, the above-mentioned methods SampleAction [40] and IFocus can be also used for line chart. Different from these two methods, in each sampling iteration, INCVISAGE [94] proposes the concept of "improvement potential" to select better samples which can improve the line chart visualization most. Thus, INCVISAGE can generate approximations that use as few samples as possible for trend-line visualizations.

Table 2 Comparison of the AQP systems

	Online/offline	Distributed/standalone	Bounded	Platform	Algorithm	Skewed data
BlinkDB	Offline	Distributed	×	Hive/Hadoop (Shark)	Stratified sampling	√
Sapprox	Online	Distributed	×	Hadoop	Distribution-aware [44] Online sampling	×
Approxhadoop	Online	Distributed	×	Hadoop	Approximation-enabled MapReduce [44]	×
Quickr	Online	Distributed	×	N/A	ASALQA algorithm [61]	×
SnappyData	Online	Distributed	×	Spark and GemFire	Spark, as a computational engine; GemFire, as transactional store	×
FluoDB	Online	Distributed	×	Spark	Mini-batch execution [111] OLA Model	×
XDB	Online	Standalone	×	PostgreSQL	Wander join [72]	×
Verdict	Online	Standalone	×	Spark SQL	Database learning	×
IDEA	Online	Standalone	×	N/A	Reuse answers of past overlapping queries for new query	×
BEAS	Online	Standalone	√	Commercial DBMS	Approximability theorem [17]	×
ABS	Online	Standalone	×	N/A	Bootstrap	×

√ means that the systems return approximation results within bounded guarantee or can deal with skewed data, × means the methods that have no bound guarantee or cannot deal with skewed data. N/A means that the systems are not built on any existing platforms

6 AQP Systems

In big data era, most IT companies have large volume of data, and thus building AQP system is important to make online decisions. We will first discuss systems built on distributed clusters in Sect. 6.1 and then introduce other systems in Sect. 6.2. We compare the well-known AQP systems in Table 2.

6.1 Distributed AQP Systems

The big data is usually stored in distributed environment, many researchers and commercial companies would like to develop AQP systems on top of distributed system (e.g., Hadoop or Spark).

BlinkDB is a well-known AQP system which is built on top of Hadoop and devises effective strategies to select proper samples (offline generated) in distributed clusters to answer newly coming queries. BlinkDB is open sourced and public available.[2] As BlinkDB generates too many offline samples based on the assumption that the QCS is stable over time, it does not perform good for queries whose QCS is not covered by the query workload. Sapprox[114] and Approxhadoop[44] add online sampling generation during running time to overcome the shortages of BlinkDB. Besides, Sapprox is more flexible than BlinkDB and more efficient than Approxhadoop.

Most recently, Quickr [60, 61] is designed for executing ad-hoc queries on big-data clusters which perform even much better than BlinkDB, Approxhadoop and Sapprox. It does not need any pre-computing of the whole dataset spread over the clusters. In fact, in modern distributed database systems, constructing synopses for the whole dataset spreading over all the clusters is very hard. Although the ad-hoc queries are complex, Quickr does not need to scan the whole data for many times. It is verified to be practical in Microsoft's search engine, i.e., Bing. What's more, Microsoft also develops Now! [19], a progressive data-parallel computation framework, for Windows Azure. It can provide progressive SQL support over big data in Azure.

Many other "Spark"-based systems are built on real-world settings. SnappyData [79, 95] is a system built on spark which uses in-memory solution for delivering truly interactive analytics (i.e., a couple of seconds), when faced with large data volumes or high velocity streams. SnappyData can exploit state-of-the-art approximate query-processing techniques and a variety of data synopses. Besides, FluoDB [111] generalizes OLA to support general OLAP queries with arbitrarily nested aggregates which is also built on top of spark.

6.2 Other AQP Database Engines

For complex queries on multiple tables, XDB [72] is a system which supports online aggregation for complex queries including join operator by integrating "Wander Join" based on the latest version of PostgreSQL. It is the most recent AQP work supporting complex queries. XDB outperforms

[2] https://devhub.io/zh/repos/sameeragarwal-blinkdb.

the earlier online engine DBO [32, 57]. Verdict [77] uses the Database Learning [87] method to benefit the query processing in database (see details of Database Learning in Section 2.5). IDEA [66] is an interactive data exploration accelerator. It allows data scientists to immediately explore the data source without pre-computation and knowledge about the data distribution and support interactive query during the data exploration process. IDEA reuses the observations seen so far and reformulates the AQP model based on probability theory. IDEA proposes a new type of index to help getting answers within low error even in the rare subgroup of a dataset without upfront known workload.

Many other systems also rely on offline synopses generation and online estimation. BEAS (Boundedly Evaluable Sql) [17] is a system which can evaluate the feasibility of each of the query plans and select a better one. Given a sampling ratio, it can either compute the exact answer or give an approximation by accessing no more than bounded numbers of tuples using bounded evaluable theory [37]. ABS (Analytical Bootstrap) [112, 113] is a system which models bootstrap by a probabilistic relational model to predict the distributions of the AQP results. It entails a very fast computation of bootstrap-based quality measures for a general class of SQL queries.

7 Emerging Challenges and Opportunities

In this section, we summarize some research challenges and opportunities in approximate query processing.

7.1 AQP Model

Most existing AQP techniques are studied cases by cases, aiming to solve different problems by designing different techniques. In other words, AQP is not well formulated and it calls for a standard and well-formulated AQP framework that can be used in many cases. To this end, one can survey each category of AQP methods, extract the common feature of AQP, and design a common framework for AQP. Furthermore, it requires to formulate a standard SQL-like query language for AQP to make AQP easy to use.

7.2 Approximate Data Visualization

There are still many problems in approximate data visualization. First, how to quantify the accuracy of visualization is an open problem. Second, selecting proper chart type to fit different AQP methods is difficult. It requires to investigate effective techniques for rapidly generating visualizations for other optimization goals (including outlier detection, trend detection) and other data types (such as large networks). Finding new data visualization applications such as ExploreSample [107] is also promising.

7.3 Smarter Query Plan

A general query can be divided into three components, (1) generate query plan, (2) find tuples satisfying the query conditions, (3) aggregate the results according to (1). Traditional AQP methods focus on (2) but fail to find approximation-aware scheduler. Thus, many recent studies focus on finding smarter query planning [87]. Smart query planning can be implemented into online aggregation process, offline synopses generation or both. Online aggregation focuses on different online ideas and offline synopses aim to find a reasonable type of synopses while optimizing query planning focuses on better query strategies.

First, if the query needs to be answered with a user-given time bound, the system should be able to predict the query's latency for different sample sizes accurately, e.g., if the user needs a query processing within one second, then the sample size should be small enough to be computed within 1 second. Second, as many queries are nested and complex, a smart scheduler is crucial in query systems, e.g., the AQP engine of verdict mentioned in Sect. 2.5, and more details of such technique can be found in [87]. Third, as a single SQL query often corresponds to multiple query plans, a smart data engine needs a query optimizer to select the best plan. Traditionally, a query optimizer can estimate the computation cost of each query plan and choose the one with the minimum estimated cost. Future work could concentrate on above three aspects to generate smart query plan so as to benefit the accuracy and speed of AQP systems.

7.4 Synopse Generation in Distributed Setting

It is easy to compute a synopsis on a single computer. However, the big data are always stored in a distributed cluster. Thus, how to generate synopses on each node and merge them together is important. Although the union of synopses from different nodes can be taken as a synopsis of the whole dataset, it will involve duplicate samples and lead to low quality. Moreover, Histogram, Wavelet and Sketch cannot be easily merged. So it calls for effective algorithms to merge or approximately merge them. In addition, it is important to devise new types of mergeable synopses [4, 5, 15].

7.5 AQP on Data Cleaning

There are still many challenges of using AQP to enhance data cleaning. First, existing studies assume that the data follow the uniform distribution, but it is challenging to clean sample or synopses for skewed data and use the summaries

to estimate the aggregation results. Second, the error estimation of approximate answers from cleaned data is not accurate and it requires to devise effective techniques, e.g., Bootstrap to estimate error on dirty data. Third, existing techniques can be combined to support AQP on data cleaning. For example, motivated by database learning techniques in Sect. 2.5.1, as the results of previous queries can be used to make smarter query processing, implementing database learning techniques on dirty data may be interesting.

7.6 Online Algorithms for Non-Gaussian Distribution

Modern data analysis needs an interactive pattern for exploring data with little a-priori knowledge of dataset and newly coming queries. If the dataset is so big that one cannot compute the data distribution online, existing online aggregation studies will not work well. Thus, it requires to design new OLA algorithms to deal with non-Gaussian distribution data.

8 Conclusion

In this paper, we review extensive studies on approximate query processing. We first summarize the AQP use cases and then categorize existing techniques into online aggregation and offline synopsis generation. We survey all of existing techniques of AQP in the fields of database and data mining. We summarize the challenges and provide the techniques to address these challenges. We also discuss how to support complex data types and new applications that can be enhanced by AQP. We survey existing AQP systems and discuss their advantages and limitations. Finally, we provide emerging challenges and opportunities.

Acknowledgements This paper was supported by 973 Program of China (2015CB358700), NSF of China (61632016, 61521002, 61472198, 61661166012), and Huawei, TAL.

References

1. Acharya J, Diakonikolas I, Hegde C, Li JZ, Schmidt L (2015) Fast and near-optimal algorithms for approximating distributions by histograms. In: PODS, pp 249–263
2. Acharya S, Gibbons PB, Poosala V, Ramaswamy S (1999) The aqua approximate query answering system. In: SIGMOD, pp 574–576
3. Acharya S, Gibbons PB, Poosala V, Ramaswamy S (1999) Join synopses for approximate query answering. In: SIGMOD, pp 275–286
4. Agarwal PK, Cormode G, Huang Z, Phillips JM, Wei Z, Yi K (2012) Mergeable summaries. In: PODS, pp 23–34
5. Agarwal PK, Cormode G, Huang Z, Phillips JM, Wei Z, Yi K (2013) Mergeable summaries. ACM Trans Database Syst 38(4):26:1–26:28
6. Agarwal S, Milner H, Kleiner A, Talwalkar A, Jordan MI, Madden S, Mozafari B, Stoica I (2014) Knowing when you're wrong: building fast and reliable approximate query processing systems. In: SIGMOD, pp 481–492
7. Agarwal S, Mozafari B, Panda A, Milner H, Madden S, Stoica I (2013) Blinkdb: queries with bounded errors and bounded response times on very large data. In: EuroSys, pp 29–42
8. Agrawal S, Chaudhuri S, Narasayya VR (2000) Automated selection of materialized views and indexes in SQL databases. In: VLDB, pp 496–505
9. Alabi D, Wu E (2016) Pfunk-h: approximate query processing using perceptual models. In: HILDA@SIGMOD, p 10
10. Armbrust M, Liang E, Kraska T, Fox A, Franklin MJ, Patterson DA (2013) Generalized scale independence through incremental pre-computation. In: SIGMOD, pp 625–636
11. Babcock B, Chaudhuri S, Das G (2003) Dynamic sample selection for approximate query processing. In: SIGMOD, pp 539–550
12. Belussi A, Catania B, Migliorini S (2013) Approximate queries for spatial data. In: Advanced query processing, vol 1, issues and trends, pp 83–127
13. Binglei G, Yu J, Liao B, Yang D, Lu L (2017) A green framework for DBMS based on energy-aware query optimization and energy-efficient query processing. J Netw Comput Appl 84:118–130
14. Braverman V, Ostrovsky R (2013) Generalizing the layering method of indyk and woodruff: recursive sketches for frequency-based vectors on streams. In: APPROX, pp 58–70
15. Cafaro M, Tempesta P, Pulimeno M (2014) Mergeable summaries with low total error. CoRR, abs/1401.0702
16. Cao Y, Fan W (2016) An effective syntax for bounded relational queries. In: SIGMOD, pp 599–614
17. Cao Y, Fan W (2017) Data driven approximation with bounded resources. PVLDB 10(9):973–984
18. Cao Y, Fan W, Wo T, Yu W (2014) Bounded conjunctive queries. PVLDB 7(12):1231–1242
19. Chandramouli B, Goldstein J, Quamar A (2013) Scalable progressive analytics on big data in the cloud. PVLDB 6(14):1726–1737
20. Chaudhuri S, Das G, Narasayya VR (2001) A robust, optimization-based approach for approximate answering of aggregate queries. In: SIGMOD, pp 295–306
21. Chaudhuri S, Das G, Narasayya VR (2007) Optimized stratified sampling for approximate query processing. ACM Trans Database Syst 32(2):9
22. Chaudhuri S, Ding B, Kandula S (2017) Approximate query processing: no silver bullet. In: SIGMOD, pp 511–519
23. Chaudhuri S, Motwani R, Narasayya VR (1999) On random sampling over joins. In: SIGMOD, pp 263–274
24. Chen J, Zhang Q (2017) Bias-aware sketches. PVLDB 10(9):961–972
25. Chen S, Jiang S, He B, Tang X (2016) A study of sorting algorithms on approximate memory. In: SIGMOD, pp 647–662
26. Chu X, Ilyas IF, Krishnan S, Wang J (2016) Data cleaning: overview and emerging challenges. In: SIGMOD, pp 2201–2206

27. Cormode G (2011) Sketch techniques for approximate query processing. Foundations and trends in databases. NOW Publishers, Breda
28. Cormode G, Deligiannakis A, Garofalakis MN, McGregor A (2009) Probabilistic histograms for probabilistic data. PVLDB 2(1):526–537
29. Cormode G, Garofalakis MN, Haas PJ, Jermaine C (2012) Synopses for massive data: samples, histograms, wavelets, sketches. Found Trends Databases 4(1–3):1–294
30. Cule B, Geerts F, Ndindi R (2015) Space-bounded query approximation. In: ADBIS, pp 397–414
31. Ding B, Huang S, Chaudhuri S, Chakrabarti K, Wang C (2016) Sample + seek: Approximating aggregates with distribution precision guarantee. In: SIGMOD, pp 679–694
32. Dobra A, Jermaine C, Rusu F, Xu F (2009) Turbo-charging estimate convergence in DBO. PVLDB 2(1):419–430
33. Duan L, Pang T, Nummenmaa J, Zuo J, Zhang P, Tang C (2018) Bus-olap: A data management model for non-on-time events query over bus journey data. Data Sci Eng 3(1):52–67
34. Eldawy A, Mokbel MF (2017) The era of big spatial data. PVLDB 10(12):1992–1995
35. Fan W, Geerts F, Cao Y, Deng T, Lu P (2015) Querying big data by accessing small data. In: PODS, pp 173–184
36. Fan W, Geerts F, Libkin L (2014) On scale independence for querying big data. In: PODS, pp 51–62
37. Fan W, Geerts F, Neven F (2013) Making queries tractable on big data with preprocessing. PVLDB 6(9):685–696
38. Fan W, Wang X, Wu Y (2014) Querying big graphs within bounded resources. In: SIGMOD, pp 301–312
39. Feng Z, Zhu Y (2016) A survey on trajectory data mining: techniques and applications. IEEE Access 4:2056–2067
40. Fisher D, Popov IO, Drucker SM, schraefel mc (2012) Trust me, i'm partially right: incremental visualization lets analysts explore large datasets faster. In: CHI, pp 1673–1682
41. Flajolet P, Martin GN (1985) Probabilistic counting algorithms for data base applications. J Comput Syst Sci 31(2):182–209
42. Galakatos A, Crotty A, Zgraggen E, Binnig C, Kraska T (2017) Revisiting reuse for approximate query processing. PVLDB 10(10):1142–1153
43. Garofalakis MN, Gehrke J, Rastogi R (2002) Querying and mining data streams: you only get one look a tutorial. In: SIGMOD, p 635
44. Goiri I, Bianchini R, Nagarakatte S, Nguyen TD (2015) Approxhadoop: bringing approximations to mapreduce frameworks. In: ASPLOS, pp 383–397
45. Goyal A, III HD, Cormode G (2012) Sketch algorithms for estimating point queries in NLP. In: EMNLP-CoNLL, pp 1093–1103
46. Guha S, Harb B (2005) Wavelet synopsis for data streams: minimizing non-euclidean error. In: SIGKDD, pp 88–97
47. Haas PJ, Hellerstein JM (1999) Ripple joins for online aggregation. In: SIGMOD, pp 287–298
48. Haas PJ, Koenig C (2004) A bi-level bernoulli scheme for database sampling. In: SIGMOD, pp 275–286
49. Haas PJ, Naughton JF, Seshadri S, Swami AN (1996) Selectivity and cost estimation for joins based on random sampling. J Comput Syst Sci 52(3):550–569
50. Halevy AY (2001) Answering queries using views: a survey. VLDB J 10(4):270–294
51. He B (2014) When data management systems meet approximate hardware: challenges and opportunities. PVLDB 7(10):877–880
52. Hellerstein JM, Haas PJ, Wang HJ (1997) Online aggregation. In: SIGMOD, pp 171–182
53. Hesterberg TC (2014) What teachers should know about the bootstrap: resampling in the undergraduate statistics curriculum. Am Stat 69(4):371–386
54. Ioannidis YE (1993) Universality of serial histograms. In: VLDB, pp 256–267
55. Ioannidis YE, Christodoulakis S (1993) Optimal histograms for limiting worst-case error propagation in the size of join results. ACM Trans Database Syst 18(4):709–748
56. Jayachandran P, Tunga K, Kamat N, Nandi A (2014) Combining user interaction, speculative query execution and sampling in the DICE system. PVLDB 7(13):1697–1700
57. Jermaine C, Arumugam S, Pol A, Dobra A (2008) Scalable approximate query processing with the DBO engine. ACM Trans Database Syst 33(4):23:1–23:54
58. Joshi S, Jermaine CM (2008) Materialized sample views for database approximation. IEEE Trans Knowl Data Eng 20(3):337–351
59. Kamat N, Jayachandran P, Tunga K, Nandi A (2014) Distributed and interactive cube exploration. In: ICDE, pp 472–483
60. Kandula S (2017) Errata and proofs for "quickr". In: Technical Report TR-2017-14, MSR
61. Kandula S, Shanbhag A, Vitorovic A, Olma M, Grandl R, Chaudhuri S, Ding B (2016) Quickr: lazily approximating complex adhoc queries in bigdata clusters. In: SIGMOD, pp 631–646
62. Kim A, Blais E, Parameswaran AG, Indyk P, Madden S, Rubinfeld R (2015) Rapid sampling for visualizations with ordering guarantees. PVLDB 8(5):521–532
63. Kim WH, Adluru N, Chung MK, Charchut S, GadElkarim JJ, Altshuler LL, Moody T, Kumar AR, Singh V, Leow AD (2013) Multi-resolutional brain network filtering and analysis via wavelets on non-euclidean space. In: MICCAI, pp 643–651
64. Kim WH, Chung MK, Singh V (2013) Multi-resolution shape analysis via non-euclidean wavelets: applications to mesh segmentation and surface alignment problems. In: CVPR, pp 2139–2146
65. Kim WH, Singh V, Chung MK, Hinrichs C, Pachauri D, Okonkwo OC, Johnson SC (2014) Multi-resolutional shape features via non-euclidean wavelets: applications to statistical analysis of cortical thickness. NeuroImage 93:107–123
66. Kraska T (2017) Approximate query processing for interactive data science. In: SIGMOD, p 525
67. Krishnan S, Wang J, Franklin MJ, Goldberg K, Kraska T (2015) Stale view cleaning: getting fresh answers from stale materialized views. PVLDB 8(12):1370–1381
68. Krishnan S, Wang J, Franklin MJ, Goldberg K, Kraska T, Milo T, Wu E (2015) Sampleclean: fast and reliable analytics on dirty data. IEEE Data Eng Bull 38(3):59–75
69. Laptev N, Zeng K, Zaniolo C (2012) Early accurate results for advanced analytics on mapreduce. PVLDB 5(10):1028–1039
70. Li F, Wu B, Yi K, Zhao Z (2016) Wander join: Online aggregation for joins. In: SIGMOD, pp 2121–2124
71. Li F, Wu B, Yi K, Zhao Z (2016) Wander join: online aggregation via random walks. In: SIGMOD, pp 615–629
72. Li F, Wu B, Yi K, Zhao Z (2017) Wander join and XDB: online aggregation via random walks. SIGMOD Rec 46(1):33–40
73. Li Y, Chow C, Deng K, Yuan M, Zeng J, Zhang J, Yang Q, Zhang Z (2015) Sampling big trajectory data. In: CIKM, pp 941–950
74. Macke S, Zhang Y, Huang S, Parameswaran AG (2017) Adaptive sampling for rapidly matching histograms. CoRR, abs/1708.05918
75. McDiarmid C (1998) Concentration. In: Probabilistic methods for algorithmic discrete mathematics
76. Moritz D, Fisher D, Ding B, Wang C (2017) Trust, but verify: optimistic visualizations of approximate queries for exploring big data. In: CHI, pp 2904–2915
77. Mozafari B (2015) Verdict: a system for stochastic query planning. In: CIDR
78. Mozafari B (2017) Approximate query engines: commercial challenges and research opportunities. In: SIGMOD, pp 521–524

79. Mozafari B, Ramnarayan J, Menon S, Mahajan Y, Chakraborty S, Bhanawat H, Bachhav K (2017) Snappydata: a unified cluster for streaming, transactions and interactice analytics. In: CIDR
80. Mytilinis I, Tsoumakos D, Koziris N (2016) Distributed wavelet thresholding for maximum error metrics. In: SIGMOD, pp 663–677
81. Nash A, Segoufin L, Vianu V (2010) Views and queries: determinacy and rewriting. ACM Trans Database Syst 35(3):21:1–21:41
82. Nirkhiwale S, Dobra A, Jermaine CM (2013) A sampling algebra for aggregate estimation. PVLDB 6(14):1798–1809
83. Olken F (1993) Random Sampling from Databases. Ph.D. thesis, University of California at Berkeley
84. Olken F, Rotem D (1986) Simple random sampling from relational databases. In: VLDB, pp 160–169
85. Pandey P, Bender MA, Johnson R, Patro R (2017) A general-purpose counting filter: making every bit count. In: SIGMOD, pp 775–787
86. Park Y, Cafarella MJ, Mozafari B (2016) Visualization-aware sampling for very large databases. In: ICDE, pp 755–766
87. Park Y, Tajik AS, Cafarella MJ, Mozafari B (2017) Database learning: toward a database that becomes smarter every time. In: SIGMOD, pp 587–602
88. Piatetsky-Shapiro G, Connell C (1984) Accurate estimation of the number of tuples satisfying a condition. In: SIGMOD, pp 256–276
89. Pitel G, Fouquier G (2015) Count-min-log sketch: approximately counting with approximate counters. CoRR, abs/1502.04885
90. Pol A, Jermaine C (2005) Relational confidence bounds are easy with the bootstrap. In: SIGMOD, pp 587–598
91. Poosala V, Ioannidis YE, Haas PJ, Shekita EJ (1996) Improved histograms for selectivity estimation of range predicates. In: SIGMOD, pp 294–305
92. Potti N, Patel JM (2015) DAQ: a new paradigm for approximate query processing. PVLDB 8(9):898–909
93. Qin C, Rusu F (2014) PF-OLA: a high-performance framework for parallel online aggregation. Distrib Parallel Databases 32(3):337–375
94. Rahman S, Aliakbarpour M, Kong H, Blais E, Karahalios K, Parameswaran AG, Rubinfeld R (2017) I've seen "enough": incrementally improving visualizations to support rapid decision making. PVLDB 10(11):1262–1273
95. Ramnarayan J, Mozafari B, Wale S, Menon S, Kumar N, Bhanawat H, Chakraborty S, Mahajan Y, Mishra R, Bachhav K (2016) Snappydata: a hybrid transactional analytical store built on spark. In: SIGMOD, pp 2153–2156
96. Sampson A, Dietl W, Fortuna E, Gnanapragasam D, Ceze L, Grossman D (2011) Enerj: approximate data types for safe and general low-power computation. In: PLDI, pp 164–174
97. Sampson A, Nelson J, Strauss K, Ceze L (2014) Approximate storage in solid-state memories. ACM Trans Comput Syst 32(3):9:1–9:23
98. Sarma AD, Lee H, Gonzalez H, Madhavan J, Halevy AY (2012) Efficient spatial sampling of large geographical tables. In: SIGMOD, pp 193–204
99. Sazish AN, Amira A (2008) An efficient architecture for HWT using sparse matrix factorisation and DA principles. In: APCCAS, pp 1308–1311
100. Shekelyan M, Dignös A, Gamper J (2017) Digithist: a histogram-based data summary with tight error bounds. PVLDB 10(11):1514–1525
101. Song G, Qu W, Liu X, Wang X (2018) Approximate calculation of window aggregate functions via global random sample. Data Sci Eng 3(1):40–51
102. Su H, Zait M, Barrire V, Torres J, Menck A (2016) Approximate aggregates in oracle 12c. pp 1603–1612
103. Tong JYF, Nagle D, Rutenbar RA (2000) Reducing power by optimizing the necessary precision/range of floating-point arithmetic. IEEE Trans VLSI Syst 8(3):273–286
104. Vengerov D, Menck AC, Zaït M, Chakkappen S (2015) Join size estimation subject to filter conditions. PVLDB 8(12):1530–1541
105. Wang J, Krishnan S, Franklin MJ, Goldberg K, Kraska T, Milo T (2014) A sample-and-clean framework for fast and accurate query processing on dirty data. In: SIGMOD, pp 469–480
106. Wang L, Christensen R, Li F, Yi K (2015) Spatial online sampling and aggregation. PVLDB 9(3):84–95
107. Wu Y, Harb B, Yang J, Yu C (2015) Efficient evaluation of object-centric exploration queries for visualization. PVLDB 8(12):1752–1763
108. Xie X, Hao X, Pedersen TB, Jin P, Chen J (2016) OLAP over probabilistic data cubes I: aggregating, materializing, and querying. In: ICDE, pp 799–810
109. Yan Y, Chen LJ, Zhang Z (2014) Error-bounded sampling for analytics on big sparse data. PVLDB 7(13):1508–1519
110. Yi K, Wang L, Wei Z (2014) Indexing for summary queries: theory and practice. ACM Trans Database Syst 39(1):2:1–2:39
111. Zeng K, Agarwal S, Dave A, Armbrust M, Stoica I (2015) G-OLA: generalized on-line aggregation for interactive analysis on big data. In: SIGMOD, pp 913–918
112. Zeng K, Gao S, Gu J, Mozafari B, Zaniolo C (2014) ABS: a system for scalable approximate queries with accuracy guarantees. In: SIGMOD, pp 1067–1070
113. Zeng K, Gao S, Mozafari B, Zaniolo C (2014) The analytical bootstrap: a new method for fast error estimation in approximate query processing. In: SIGMOD, pp 277–288
114. Zhang X, Wang J, Yin J, Ji S (2016) Sapprox: enabling efficient and accurate approximations on sub-datasets with distribution-aware online sampling. PVLDB 10(3):109–120
115. Zheng Y (2015) Trajectory data mining: an overview. ACM TIST 6(3):29:1–29:41