An Overview of Query Processing on Crowdsourced Databases

Marwa B. Swidan\textsuperscript{1}, Ali A. Alwan\textsuperscript{1}, Yonis Gulzar\textsuperscript{3}, Abdallah Zaid \textsuperscript{2}  
Abualkishik\textsuperscript{1} 
\textsuperscript{1}International Islamic University Malaysia, Kuala Lumpur, Malaysia  
\textsuperscript{2}King Faisal University, Al Ahsa, Saudi Arabia  
\textsuperscript{3}American University in The Emirates, Dubai, United Arab Emirates

ABSTRACT
Crowdsourcing is a powerful solution to correctly answer expensive and unanswered queries in the database. This includes queries on a database with uncertain and incomplete data. The crowdsourcing attempts to exploit human abilities to process these difficult tasks and workers helped to provide accurate results utilizing the available data in the crowd. The crowd-sourcing database systems (CSDB) combined the ability of the crowd with the relational database by using some variant of the relational database with minor changes. This paper surveys and examines the leading studies conducted on the area of query processing for both traditional and preference queries in crowdsourcing databases. The focus is given on highlights the strengths and the weakness of each approach. A detailed discussion for the current and future trends research relevant to query processing in the area of crowd-sourced databases is also demonstrated.

1. Introduction
In many contemporary database applications, some queries cannot always be optimally answered through traditional techniques. There are many causes that lead to not answering these queries. This includes imprecise or uncertain information. Examples of such applications include but not limited to translation, handwriting recognition, image understanding, and web databases. The crowdsourced platforms became an effective solution for various types of queries by exploiting knowledge, ideas, experiences, and skills of crowd workers to process information and obtain accurate answers for queries. The hybrid human/machine systems defined as the process that integrates the crowd workers with computer systems to get high-quality results (Parameswaran \textit{et al.}, 2011; Li \textit{et al.}, 2016; Swidan \textit{et al.}, 2018; Bhaskar \textit{et al.}, 2020; Lian \textit{et al.}, 2020; Swidan \textit{et al.}, 2020). The crowd-sourced system consists of three main components that work together to provide the users with the best answers for the given queries. These components are (i) requester, who submits the query to the crowd and wait for the answers to be retrieved from the crowd, (ii) crowd platform, which contains the query execution engine that is responsible to manage the user given query and retrieve the answers, (iii) workers, who are responsible to work on the queries and delivered the results to the requester through the crowd platform (Difallah \textit{et al.}, 2015; Chai \textit{et al.}, 2019).

Figure 1 demonstrates the details structure of the typical crowdsourced platform. The requester submits the query to the crowd specifying the requirements to accomplish the task. The crowd divides the given query (task) into many sub-tasks named micro-tasks that call HIT (Human Intelligent Task). Then, the platform assigned the HITs to the workers who agree to carry out the task. Assigning the task to the workers is the most critical part of processing queries in the crowd-sourced system. Since it involves the workers who attempt to form the result based on the user query. The quality of the generated result is highly influenced by the quality of the selected workers. Therefore, not all workers are suitable to accomplish the task and the decision to select the most appropriate worker is done by the requester (Franklin \textit{et al.}, 2011; Difallah \textit{et al.}, 2015; Chai \textit{et al.}, 2019; Lian \textit{et al.}, 2020).

The crowd-sourced database (CSDB) contains a massive amount of heterogeneous data stored on various online places that are accessible through the crowd. Furthermore, these data might be in many different forms such as web data, text, image, audio, and personal data (Li \textit{et al.}, 2016; Lian \textit{et al.}, 2020). The CSDB uses many...
some workers unable to achieve a high-quality result or sometimes workers are incapable to work on certain tasks (Li et al., 2016). In this paper, we are reviewing and examining some leading studies pertaining to both traditional and preference queries in the area of crowd-sourced databases. Furthermore, we also highlighted and explained the differences between processing queries in both traditional and crowd-sourced databases.

The remainder of the paper is organized as follows. Section 2 reports the literature of traditional queries in crowd-sourced databases. Section 3 provides the literature of the preferences queries in the crowd-sourced database. The discussion encompasses some leading studies related to preference queries in traditional databases. A detailed discussion and future research challenges presented in Section 4. Finally, the conclusion of the paper has been outlined in Section 5.

2. Traditional Queries in Crowd-Sourced Databases

Traditional queries generate query results by strictly relying on conditions as given in a query. Thus, a nil result might be returned if no tuple satisfies the given conditions. Crowd database systems just like traditional databases require data manipulation functions or traditional queries such as select, aggregate, maximum, and average, except that it must rely on the crowd workers. This section discusses the approaches introduced for traditional queries in the crowd-sourced database.

2.1. Selection Operators:

The selection operator uses to retrieve data from the database that satisfy the conditions given in the user query. The filter in crowd-sourced used the human computations to identify a set of tuples that satisfy query constraint. It is like a ‘SELECT’ operator in the traditional database. For example, we may ask the crowd to select historical movies from the movie database. The CrowdDB (Franklin et al., 2011) and Qurk (Marcus et al., 2011) systems have introduced various types of heuristic filtering strategies (majority voting over a fixed number of workers), but have not studied how to implement optimal filtering strategies, these systems attempted by optimizing monetary cost. The CrowdDB (Franklin et al., 2011) has optimized join, compare and fill operators, while the Qurk (Marcus et al., 2011) system, studied the join and sort operators. Furthermore, Deco (Parameswaran et al., 2012a) has focused on missing value records in the database. In contrast, the CrowdScreen (Parameswaran et al., 2012b) system worked on only the select (filter) operator and aims to determine a strategy that minimizes the number of tasks submitted to the crowd by taking into consideration the total error less than a threshold. The CrowdFind (Sarma et al., 2014) attempted to investigate the cost and latency of filtering with the crowd. The system allows asking a batch of questions in parallel. CrowdOP (Fan et al., 2015) aimed to optimize more general operators than CrowdFind. It attempts to minimize the cost and latency on select, join, and complex operators.

2.2. Sorting Operators:

The sort operator relies on one or more attributes from a relational database to order all the tuples in the database. The crowd-sourced databases exploit human computation to applying the sorting operator. For instance, given some photos about a restaurant, asking the crowd to sort them according to the fact, which photo describes the restaurant better, and then the better photos may be shown in its website. To achieve the sorting operator in the crowd-sourced system the strategies changing the comparator method that used in a traditional sorting algorithm by a new method that depends on
workers to evaluate the comparison, and they decide which value is better. CrowdDB system has implemented CrowdCompare operator as CROWDEQUAL function that takes two values and asks the crowd to identify which of them are equal. CROWDDORDER function has been used to rank or order the result (Franklin et al., 2011). In Qurk (Marcus et al., 2011), the rankProducts function is implemented to perform the comparison between many tuples to sort them.

2.3. Filling Operators:
Most of the crowd-sourced systems attempt to support the queries on an incomplete database. Many systems have implemented operators that generate the missing values in the database to identify the best results. CrowdDB (Franklin et al., 2011) implements a CrowdProbe operator which aims to obtain the missing information for crowd-sourced tables by the workers. CDB (Li et al., 2017) introduced two built-in keywords for data gathering: FILL and COLLECT. FILL asks the crowd to find and fill the values of attributes with some rational values, for example, filling missing values of the car make attribute. COLLECT asks the crowd to collect the best 10 cars.

2.4. Join Operators:
The join operator combines at least two tuples from the same or diverse databases to get results based on the given conditions. Nonetheless, to perform the join operator on the crowd-sourced database, a wide range of approaches have been presented in the previous studies. In CrowdDB (Franklin et al., 2011), the CrowdJoin operator creates an index nested-loop join between two tables. At least one of the tables has to be a crowd-sourced table. While in Qurk (Marcus et al., 2011), the crowd is entrusted with choosing if tuples from two relations coordinate concerning at least one attribute. A naive implementation of this join would try to ask the workers about every possible pairing of tuples between the relations. In CrowdOP (Fan et al., 2015) the CJOIN (Crowd-Powered Join) operator has introduced to leverage the crowd to merge objects from two sources as indicated by specific constraints. For join queries, the CrowdOP incorporates FILL and JOIN operators as one operator. This algorithm asks the crowd worker to fill some incomplete values and then combines the values by using a join operator. For instance, to match mobile phones’ images with their reviews, first ask the crowd worker to fill the brand of mobile phones and then only match the mobile phones with the same brand. The work introduced in (Wang et al., 2013) focused on the crowd-sourced join query for entity resolution that objects to find all pairs of matching tuples between two collections of tuples, where asking the crowd to decide which pair of tuples is matching. CDB (Li et al., 2017) introduced two algorithms Crowdequality and CrowdJoin refer to join query on crowd-sourced database. Table 1 summarizes the crowd-sourced database systems examined in this section.

3. Preferences Queries In Crowd-Sourcing Databases
Preference queries are another essential type of queries where they have been incorporated in a large number of modern database applications. Preference queries aim to relax the query conditions by identifying the results that rely on user preferences to prefer one tuple over others. The growing complexity of contemporary database applications plus the need to support users with different preferences has further increased the need for a new type of query operators to be combined into these systems. Many applications for various domains have significantly benefited from these types of queries, for example, web services, multi-criteria decision-making applications, digital libraries, multimedia retrieval, and recommended systems. Due to the importance of preference queries that appeared in many database applications, many preference evaluation methods have been proposed. These preference techniques rely on the concept of the skyline and top-k techniques.

3.1. Top-k Queries in Crowd-Sourced Systems:
Top-k technique ranks the tuples of a dataset by collecting the values of each attribute of a tuple as a single value involving a monotonic ranking function \( F \). The best \( k \) tuples are ranked according to the best scoring value based on the function \( F(\text{Chaudhuri et al., 1999; Chang et al., 2002}). \) Top-k queries have several benefits which can be summarized as follows. (i) top-k queries are transitive. This means when a tuple \( p \) dominates another tuple \( q \), and \( q \) dominates a tuple \( r \), hence \( p \) dominates \( r \). (ii) The output of top-k queries is controlled by simply varying the \( k \) value which represents the number of tuples to be retrieved to the user. (iii) No exhaustive pairwise comparison between the individual attributes of tuples is needed when identifying the top-k queries result. However, top-k queries are subjective to the monotonic ranking function that is defined by the users. Thus, different top-k results can be generated based on different preference functions. Moreover, to determine the most interesting tuple of a query, it is not easy to analyze the data as different scores and functions possibly infer different results (Gumaei et al., 2017; Kontaki et al., 2010). The results of top-k queries are influenced negatively by the incompleteness of data. That means, if the tuples have missing values in one or more dimensions, the query answers might not be the same if the data were complete. Furthermore, relying on the application scenarios, in cases that the weights of the linear criterion may be known in advance, for future queries might be pre-ranked the results. Otherwise, it might not be feasible to pre-compute all possible combinations due to the weights are dynamic (Kontaki et al., 2010; Gumaei et al., 2017; Chang et al., 2002). To perform a top-k query in crowd-sourced databases, the work sorts the whole database and then returns the top value(s). The crowd-sourced top-k algorithms ask the crowd to compare tuples and detect the top-k tuples rely on the crowd-sourced comparison results. However, these algorithms might lead to high costs due to performing unnecessary comparisons during the process of top-k queries. The task generated by two ways first is a single choice where every task is to compare two tuples. Second is the rating, which selects multiple tuples and asks the crowd to assign a rate for each tuple. Many methods rely on tournament-sort have been presented in the literature for implementing crowd-based top-k.

The work in (Polychronopoulos et al., 2013) illustrated a solution based on human-powered top-k where the crowd asked to rank tuples directly and aggregates them using a median rank aggregation algorithm. This leads to identifying the final top-k result using the judgments of human workers. Furthermore, the approach allows workers to examine several items at a time, without prior knowledge about the errors of human workers. It adapts dynamically to the varying difficulty of comparing items and the existence of spammers. However, the top-k lists which are larger than the size of the ranking tasks given to the crowd cannot be tackled. The work presented in (Lin et al., 2017) concentrates on reduces the uncertainty of top-k ranking by using the crowdsourced. The issue of uncertainty leads to low-quality results. Thus, the algorithms that try to identify the best
tuple pairs for crowdsourced leading to highest quality improvement with the crowd-sourced task of a limited budget is proposed. Besides, the work in (Ciceti et al., 2015) aims to reduce the expected residual uncertainty of the result by giving a set of questions that, within an allocated budget to a crowd, this might lead to a unique ordering of the top $k$ results. The work introduced in (De Alfarco et al., 2016) proposed a method for the top-$k$ queries that leverage the knowledge obtained from crowd comparisons to reduce the set of candidate tuples of the top-$k$ list. The idea of the proposed approach relies on controlling the size of the crowd comparison tasks decouples from the size of the top-$k$ results list.

Moreover, the work in (Lee et al., 2017) studied the issue of top-$k$ queries in a crowd-sourced database called crowd-enabled top-$k$ queries. A novel framework that aims to increase the monetary cost when the latency is constrained is proposed. The proposed strategy consists of a two-phase parameterized framework with two parameters namely: named buckets, and ranges. Furthermore, the framework incorporates three methods called: greedy, equi-sized, and dynamic programming, to identify the buckets and ranges. By combining those three methods at each phase, four algorithms have been formed: GdyBucket, EquiBucket, EquiRange, and CrowdK. Lastly, the work presented in (Kou et al., 2017) introduced a new framework for processing top-$k$ queries in crowdsourced systems aiming at minimizing monetary cost, latency, and maximizes the quality of the results.

### Table 2: Summary of previous approaches of top-$k$ queries in a crowd-sourcing database

| Author/Year | Approach | No. of Workers | Values | Database Type | Performance Metrics |
|-------------|----------|----------------|--------|---------------|---------------------|
| Polychronopoulos et al., 2013 | Algorithm | 3-15 | 1-5 | Synthetic, real | Cost, latency |
| Lin et al., 2017 | A novel pairwise crowdsourcing model | - | 10-30 | Real | |
| Ciceti, et al., 2015 | Heuristic algorithm | Multiple workers | 1-10 | Synthetic, real | Quality |
| De Alfarco et al., 2016 | Crowd-top-$k$ | 3 | 5-50 | Synthetic, real | Cost, latency, quality |
| Lee et al., 2017 | CrowdK | 3, 5, 7, 9 | 2-20 | Synthetic, real | Cost, latency, quality |
| Kou et al., 2017 | SPR | - | 1.5, 10, 15, 20 | Real | Cost, latency, quality |

The proposed framework tries to reduce the monetary cost by decreasing the number of comparisons and improve accuracy by introducing a judgment model that allowed pairwise preference judgment. Table 2 summarizes the previous approaches of top-$k$ queries discussed in this section.

### 3.2. Skyline Queries in Crowd-sourced Databases:

Skyline technique retrieves the skyline. A skyline in a way such that any tuple in $S$ is not dominated by any other tuples in the database (Gulzar et al., 2017). Skyline queries have many powerful characteristics that can be described as follows. (i) each attribute is evaluated independently through the process of pairwise comparison and a user-defined ranking function is not required, (ii) the skyline results are identified based on the real values of the data in the database, (iii) the scale of attributes does not impact on the skylines as the comparison process merely relies on the present value in each attribute, (iv) integrating skyline operator into SQL query processor is extremely easy and simple, (v) it intuitively performs multi-objective query operations, (vi) it exhibits the transitivity property which is the main theme of skyline technique, and (vii) it is used as an effective operator for the aggregate operation (Kontaki et al., 2010; Kou et al., 2017). Nevertheless, skyline queries have some weaknesses which are described as follows. (i) There is no control on the size of skylines and it might happen that in the worst case, all the tuples in the database could be retrieved as skylines, (ii) The computation complexity of skyline process is highly influenced by the number of attributes and the size of the database, (iii) due to the massive number of skylines, the user needs to examine many skylines to determine the most suitable tuples to be selected.

In the era of the Internet of Thing (IoT) where data enormous and collected from different sensors and monitors from remote locations, while transmission of data from these devices, there are high chances that certain data items have missing values in one or more attributes making the database to be incomplete. Processing skyline queries on an incomplete database might raise some serious challenges such as the transitivity property of skyline technique that most likely to be avoided on a database with incomplete data. Moreover, losing transitivity property may lead to making the domination test to be cyclic due to the fact that some tuples are incomparable with each other and thus the skyline result could be nil (Khalefa et al., 2008; Gulzar et al., 2016a; Gulzar & Lee, 2016b).

Various algorithms have been suggested for processing skyline queries in incomplete databases. This includes but not limited to Bucket and Iskyline (Khalefa et al., 2008), Sort-based Incomplete Data Skyline (SIDS) (Bharuka et al., 2013), Incoskyline (Alwan et al., 2016), Framework (Kou et al., 2017), SOBA (Lee et al., 2016), Bucket and Iskyline (Khalefa et al., 2008) are among the first attempts that have been introduced for processing skyline queries in a database with incomplete data. Iskyline exploits the clustering technique to process skylines. Whereas SIDS algorithm (Bharuka et al., 2013), and framework proposed in (Kou et al., 2017) is designed based on the sorting technique to rearrange the tuples in such a way that those tuples having high potential to be part of the skyline are kept on the top of the sorted list. In Incoskyline (Alwan et al., 2016) algorithm, a virtual tuple named $k$-dom is created from local skylines of each group created after clustering. Incoskyline aims to reduce the execution time and decrease the domination tests during the skyline process. Lastly, the SOBA technique (Lee et al., 2016) combines the sorting and clustering technique to process skyline queries in complete databases.

The issue of processing skyline queries on a database with incomplete has been further investigated to include more complicated database architecture such as distributed and cloud. The work proposed in (Gulzar et al., 2019a) processes the skyline queries over cloud-based incomplete databases where data divided horizontally and stored in different data centers at different locations. Moreover, the work in (Alwan et al., 2017) proposed a skyline algorithm that intends to process skyline queries on incomplete distributed databases. They have proposed an algorithm named $incoskyline$ that works on a vertically divided database stored in different relations. However, there has no much attention paid to address the issue of estimating the missing values of skylines. The only works address the issue of estimating the missing values of the skylines is contributed in (Alwan et al., 2018) aiming to capture the missing values before identifying the final skylines. Most of the skyline techniques mentioned above designed to work on traditional databases. However, the area of processing skyline queries on the crowd-enabled enabled incomplete database has not received adequate attention from the researchers in the database community. Several unique features distinguish the crowd-sourced databases from traditional databases. Therefore, it might be unwise to directly apply approaches designed for the traditional database on crowd-sourced databases. To the best of our knowledge, no much attention has been paid to the issue of processing skyline queries over the crowd-enabled enabled incomplete database. Some of the previous studies focused on designing approaches to predict the missing values of skylines balancing between the monetary cost, time latency, and the accuracy of the results.

In the following, we discuss some leading studies focused on processing skyline queries on crowd-enabled enabled incomplete databases. The discussion elaborates on the strengths and
The work has been contributed by (Asudeh et al., 2015) highlights the issue of identifying Pareto-optimal tuples in crowd-sourced databases. The study is useful when the Pareto-optimal tuples do not have clear attributes and preference relations are strict partial orders. It introduces an iterative question selection algorithm that is instantiated into different methods by relying on the ideas of candidate questions, macro-ordering, and micro-ordering. The works in (Lofi et al., 2013; El Maarry et al., 2015) have suggested a hybrid method combining dynamic crowd-processing with heuristic techniques. The proposed method aims at using a set of heuristic techniques to estimate the incomplete values for all tuples in the database. Moreover, to improve the quality of the estimated values it exploits the workers for some cases. A new approach called CrowdSky has been proposed in (Lee et al., 2016). The CrowdSky intends to process skyline queries on the crowd-sourced enabled incomplete database. The focus is given on decreasing the monetary cost, minimizing the latency, and improving the quality of results. The work in (Lee et al., 2016) assumed the initial database is complete, but the data in the database might unable to provide an accurate answer for skyline queries. Hence, need to consult the crowd to estimate this missing information from a set of virtual attributes to be created when answering the query. To the best of our knowledge, the work in (Miao et al., 2019) is the most recent work that highlights the issue of skyline queries on the crowd-sourced enabled incomplete databases. They proposed a framework for solving the skyline problem over the partial complete database with crowdsourcing called Bayescrowd. Besides both of Bayescrowd and CrowdSky based only on the crowd to estimate the missing values. However, the researchers assumed that all missing values will be estimated by utilizing the crowd-sourcing databases. Table 3 summarizes the previous skykine techniques proposed in the crowd-sourcing database that are presented in this section.

Table 3: Summary of previous approaches of skyline techniques in crowd-sourcing database

| Author / Year | Approach | Data Distribution | Database Type | No. of Dimensions | Missing Data | Performance Metrics |
|---------------|----------|-------------------|---------------|-------------------|--------------|---------------------|
| Asudeh et al., 2015 | Microsourcing | independent, anti-correlated | Real | 6-8 | 1%-20% | Latency, cost, quality |
| El Maarry et al., 2016 | Heuristic | independent, anti-correlated | Real | 6-8 | 1%-20% | Cost, quality |
| Lee et al., 2016 | CrowdSky | independent, anti-correlated | Real, synthetic | 2-5 | 1-3 dimensions | Latency, cost, quality |
| Miao et al., 2019 | Bayescrow | independent, anti-correlated | Real, synthetic | 9-11 | 20%-50% | Latency, cost, quality |

4. Discussion and Future Research Challenges

From the literature, we conclude that the monetary cost, time latency, and result accuracy are the factors that most impact on query processing in crowd-sourced databases. Most of the previous studies concentrate on designing query processing operators work in a crowd-sourced database aiming at improving the accuracy of the query result with reasonable cost and time. Cost and result accuracy are among the most critical factors that have been considered in most of the previous works (Franklin et al., 2011; Parameswaran et al., 2012b; Lofi et al., 2013; Wang et al., 2013; Swidan et al., 2018; Chai et al., 2019; Lian et al., 2020; Swidan et al., 2020). To reduce the monetary cost of extracting and generating data from the crowd, many optimizations methods have been utilized such as decreasing the number of questions that submit to the crowd (Lee et al., 2016) or put a specific budget to process the query (Ciceri, et al., 2015). However, it might be an easy task to reduce the number of questions or limit the monetary cost on traditional queries. This is because most traditional query operators access the underlying data in the database and only certain attributes might be accessed. However, we also noticed that it is very challenging to reduce the monetary cost for preference queries such as Top-k and skyline. This is because these types of queries need to scan the entire database and incur a complex process to derive the query result. For the top-k query, a monotone function needs to be formed that aggregate the values of all dimensions and sort the result in some order. Similarly, for skyline queries an exhaustive pairwise comparisons need to be performed to determine the skylines of the database. This process requires accessing the entire database and the search space will be extremely large. Many studies tried to focus on developing query algorithms that reduce the cost and time latency when accomplishing the task. Less attention has been paid to the quality of the task performed. Quality of query result is an important aspect of preference queries such as top-k and skyline as these query operators used in many decision making and decision support systems which should give the most accurate result in helping users selecting the most appropriate decision. Due to the large number of tasks issued by the user and with the time and quality constraints, most of the studies that related to process preference query on the crowd-sourcing database have assigned the tasks to either a single or multiple qualified workers. Selecting the most suitable workers to accomplish the task is also an uneasy task for the requester. Thus, many crowd-sourced systems offer task accomplishment by multiple workers by using voting strategies as a tool for aggregating answers from workers. Doing so, helps users to make the right decision and choose the qualified workers only to ensure that the quality of the result is sustained. Likewise, in some cases, the workers are submitting the qualification tests so that low-quality workers will not be assigned the tasks. Also, some systems rely on the worker’s skills experience in the queries field (Li et al., 2016). Here, choosing the exact worker has a high impact on the quality of the results produced through the crowd platform. Most recently, some studies addressed the issue of processing skyline queries on a database with missing data and investigated the impact of these missing data on processing skyline queries (Gulzar et al., 2019b; Gulzar et al., 2019c). Indeed, missing data adds another challenge when processing queries in crowd-sourced enabled databases. Some studies suggested that workers can contribute to estimating the missing values from the crowd and provide a complete answer to the user. This is accomplished by exploiting the implicit relationship between the databases of the crowd. Among the remarkable works that address the issue of preference queries on crowd-sourcing databases are (Lofi et al., 2013; Lee et al., 2016; Swidan et al., 2018; Miao et al., 2019; Swidan et al., 2020). We suggest that many research opportunities should be explored addressing the challenges of incomplete and uncertain data when processing queries in a crowd-sourcing database. Issues such as monetary cost, time latency, and result accuracy with the uncertainty and the incompleteness of the data should be considered. In today’s era, the big data area becomes a rich area of research with many research opportunities relevant to the issue of query processing that could be explored. In the big data context, data is extremely diverse and growing rapidly. Working on big data needs powerful machines and a huge number of human resources to carry out the given task, which will make the process very expensive and takes a longer time. We also observed that some studies have been conducted concentrating on big data processing, but work is limited to some pre-processing processes such as data cleaning (Wang et al., 2014), and data labeling (Mozafari et al., 2014). Hence, much work is needed to resolve the problems with data processing in the big data context. It has been argued that working on big data particularly for handling incomplete and uncertain data...
requires powerful computing machines and extensive human resources to carry out the task of missing value estimation. Using sampling techniques to estimate the missing values are among the most effective solution to provide an accurate estimation with reasonable cost.

Another issue that has been addressed by many studies is the privacy of the worker and the requester. In a certain scenario, the requester might not wish to reveal the details of their tasks, and some data are highly confidential. The crowd is open to everyone and some workers might attempt to access the requester database with other public databases that lead to privacy leakage. Not revealing the identity of the requester might negatively affect the quality of the task as workers unable to access accurate data. The privacy issues add more challenges during data processing in crowd-sourced databases. Designing an approach that balances between the accuracy of the result and the privacy of the requester is an interesting idea that should be investigated. The issue of privacy might also be considered from the worker side. In certain cases, workers many have some privacy constraints and unwilling to share their details such as personal information, location, profession, hobby. These facts about works might be requested by the requester to help them in assigning the task. Some sort of procedure is needed to devise the privacy constraints of the worker during the task assignment.

Another interesting research area that also has a high potential to be investigated is processing skyline queries over dynamic and incomplete data. The presence of incomplete data in most of the contemporary database applications is inevitable (Gulzar et al., 2018; Babanejad et al., 2020). Furthermore, databases are dynamic in nature in which their states change throughout the time to reflect the current and latest information of the applications. The frequent changes towards the initial database through the insert, delete, and update operations and the presence of incomplete data made the derived skylines before changes invalid in the new state of the database. Thus, produce a precise prediction for the missing values of the skylines after the changes made on the database is a challenging process. This is because the previous estimating for the missing values might be no longer valid in the new state of the database as many skylines might be affected by the changes made over the database. Blindly estimating the missing values of the entire skylines to identify the new set of skylines is unwise as not all data items are affected by the changes made towards the database. Thus, an efficient solution for imputing the missing values of the skylines after the changes made towards the initial incomplete and dynamic database is urgently needed. The solution should be capable to estimate only those skylines that are affected by the changes on the initial database and have high potential to be reported in the skylines result of the new state of the database (Babanejad et al., 2020). We believe that adopting some machine learning-based techniques such as neural networks and generative adversarial networks (GAN) for value estimation would be an interesting idea to be examined.

Last but not the least, it has been reported that it is very challenging to control the quality of the work done in crowd-sourced mobile platforms (Li et al., 2016). This is because of crowd-sourcing tasks influenced by some mobile platforms factors such as long-distance between the mobile phone user and the server, limitation of the power, and storage capacity (Elfaki et al., 2019). Besides, some crowd-sourced tasks need to access some spatial information about certain tuples, for instance collecting some information about nearby hotels and labeling them. Therefore, a worker may only able to accomplish such tasks for nearby hotels. However, for those far hotels, it would not be included in the task. Therefore, it would be very interesting to provide an efficient worker selection model that addresses these issues and investigate the impact of the spatial information on achieving the task. It is also worth investigating the issue of the worker model concerning the server assignment model. The server assignment model might help to overcome the limitation of the worker selection model (Li et al., 2016). An approach is needed to provide a server selection model that attempts to assign tasks based on the shortest distance and select the nearest available qualified worker to accomplish the task concerning other user objectives.

We believe that this study would be very beneficial and will open the doors for exploring numerous research opportunities relevant to the issue of processing skyline queries with incomplete data on areas such as crowd-sourcing, big data, dynamic and uncertain databases. We have presented and analyzed well-known studies focused on processing preference queries such as skyline and top-$k$ in both crowd-sourcing complete and incomplete databases. We also present and discuss some practical solutions that could be employed in handling missing data in various areas such as big data, uncertain, incomplete, and dynamic databases. Lastly, this study can be used as a departure point for many researchers who are interested in exploring the challenges of preference queries in the above-mentioned areas.

5. Conclusion

In this paper, we present and discuss the query processing in crowd-sourced database systems. We highlighted the factors that affect query processing on a crowd-sourced database. Different types of query processing including traditional and preference queries have been presented and explained. We also examine some leading studies related to query processing in the area of the crowd-sourcing database system. We summarize all of the existing techniques presented throughout the paper. Finally, we presented a detailed discussion on the current and future research challenges pertinent to preference queries in crowd-sourced database systems that could be explored in the future by interested researchers.

Bios

Marwa B. Swidan

International Islamic University Malaysia, Kuala Lumpur, Malaysia,
+60126513398, marwa_happy@yahoo.com

Mr Swidan is a PhD student at International Islamic University Malaysia, IIUM, Malaysia. She earned her B.Sc. and M.Sc. degrees in Computer Science from Tripoli University, Libya in 2004 and 2009 respectively. Her research interests include crowdsourcing, data mining/exploration, database systems and skyline queries.

Ali A. Alwan

International Islamic University Malaysia, Kuala Lumpur, Malaysia,
+60173546110, alwamer@iiu.edu.my

Dr Alwan is currently an associate professor at Kulliyyah (Faculty) of Information and Communication Technology, International Islamic University Malaysia (IIUM), Malaysia. He received his Master of Computer Science (2009) and Ph.D. in Computer Science (2013) from Universiti Putra Malaysia (UPM), Malaysia. His research interests include preference queries, skyline queries, probabilistic and uncertain databases, query processing and optimization and management of incomplete data, data integration, location-based social networks (LBSN), recommendation systems, and data management in cloud computing.
Yonis Gulzar
King Faisal University, Al Ahsa, Saudi Arabia,
+966 54 571 9118, ygulzar@kfup.edu.sa

Dr Gulzar is currently an assistant professor at King Faisal University (KFU), Saudi Arabia. He obtained his Ph.D. in Computer Science in 2018 from International Islamic University Malaysia. He received his Master in Computer Science in 2013 from Bangalore University, India. His research interests include preference queries, skyline queries, probabilistic and uncertain databases, query processing and optimization and management of incomplete data, data integration, location-based social networks (LBSN), recommendation systems, and data management in cloud computing.

Abedallah Zaid Abualkishik
American University in The Emirates, Dubai, United Arab Emirates,
+971 50 150 4130, azazoft1@gmail.com

Dr Abualkishik received a master’s and a Ph.D. degree in Software Engineering. His research interests include software functional size measurement, software functional measures conversion, cost estimation, empirical software engineering, database, Big Data, and Data Science. Currently, he is working as an assistant professor at the college of computer information technology, American University in the Emirates, Dubai, UAE. Also, he is working as a consultant for a regional software development company to prepare an accurate estimation for project deliverables.

References

Alwan, A.A., Ibrahim, H., Udzir, N.I. and Sidi, F. (2016). An efficient approach for processing skyline queries in incomplete multidimensional database. Arabian Journal for Science and Engineering, 41(8), 2927–43.

Alwan, A.A., Ibrahim, H., Udzir, N.I. and Sidi, F. (2017). Processing skyline queries in incomplete distributed databases. Journal of Intelligent Information Systems, 48(2), 399–420.

Alwan, A.A., Ibrahim, H., Udzir, N.I. and Sidi, F. (2018). Missing values estimation for skylines in incomplete database. The International Arab Journal of Information Technology, 15(1), 66–75.

Asudeh, A., Zhang, G., Hassan, N., Li, C. and Zaruba, G.V. (2015). Crowdsourcing pareto-optimal object finding by pairwise comparisons. In: Proceedings of the 24th ACM International Conference on Information and Knowledge Management, Melbourne, Australia, 19-23/10/2015. p. 753-762.

Bharuka, R. and Kumar, P.S. (2013). Finding skylines for incomplete data. In: Proceedings of the 24th Australasian Database Conference, Adelaide, Australia, 24-27/5/2013. p. 109-117.

Bhaskar, N. and Kumar, P.M. (2010). Optimal processing of nearest-neighbor user queries in crowdsourcing based on the whale optimization algorithm. Soft Computing, 24(3), 1–14.

Chang, K.C.C. and Hwang, S.W. (2002). Minimal probing: Supporting expensive predicates for top-k queries. In: Proceedings of the 2002 ACM SIGMOD International Conference on Management of data, Madison, Wisconsin, USA, 3-6/6/2002. p. 346-357.

Chaudhuri, S. and Gravano, L. (1999). Evaluating top-k selection queries. In: VLDB Proceedings of the 25th VLDB Conference, Edinburgh, Scotland, United Kingdom, 7-10/9/1999. p. 397-410.

Chai, C., Fan, J., Li, G., Wang, J. and Zheng, Y. (2019). April. Crowdsourcing database systems: Overview and challenges. In: 2019 IEEE 35th International Conference on Data Engineering (ICDE) (pp. 2052-2055). IEEE, Macao, Macao, 8-11/4/2019.

Ciceri, E., Fraternali, P., Martinenghi, D. and Tagliasacchi, M. (2015). Crowdsourcing for top-k query processing over uncertain data. IEEE Transactions on Knowledge and Data Engineering, 28(1), 41–53.

De Alfaro, L., Polychronopoulos, V. and Polyzotis, N. (2017). Efficient techniques for crowdsourced top-k lists. In: The 26th International Joint Conference on Artificial Intelligence, Melbourne, Australia, 19-25/8/2017.

Difallah, D.E., Catasta, M., Demartini, G., Ipeirotis, P.G. and Cudré-Mauroux, P. (2015). The dynamics of micro-task crowdsourcing: the case of amazon mturk. In: Proceedings of the 24th International Conference on World Wide Web, Florence, Italy, 18-22/5/2015. p. 238–247.

El Maarry, K., Lofi, C. and Balke, W. (2015). Crowdsourcing for query processing on web data: A case study on the skyline operator. Journal of Computing and Information Technology, 23(1), 43–60.

Elfaki, M.A., Alwan, A.A., Abdellatie, M. and Wahaballa, A. (2019). A literature review on collaborative caching techniques in MANETs: Issues and methods used in serving queries. Engineering, Technology & Applied Science Research, 9(5), 4729–34.

Fan, J., Zhang, M., Kok, S., Lu, M. and Ooi, B.C. (2015). Crowdsop: Query optimization for declarative crowdsourcing systems. IEEE Transactions on Knowledge and Data Engineering, 27(8), 2078–92.

Franklin, M.J., Kossmann, D., Kraska, T., Ramesh, S. and Xin, R. (2011). CrowDB: answering queries with crowdsourcing. In: SIGMOD’11. Proceedings of the 2011 ACM SIGMOD International Conference on Management of data, Athens, Greece, 12-16/7/2011. p. 61-72.

Babanejad, G., Ibrahim, H., Udzir, N.I., Sidi, F., Alwan, A.A. and Gulzar, Y. (2020). Efficient computation of skyline queries over a dynamic and incomplete database. IEEE Access Journal, 8(n/a), 141523–46.

Gulzar, Y., Alwan, A.A., Salleh, N. and Al Shaikhli, I.F. (2017). Processing skyline queries in incomplete database: Issues, challenges and future trends. Journal of Computer Science, 13(11), 647–58.

Gulzar, Y., Alwan, A.A., Salleh, N. and Al Shaikhli, I.F. (2019a). Identifying skylines in cloud databases with incomplete data. Journal of Information and Communication Technology, 18(1), 19–34.

Gulzar, Y., Alwan, A.A., Abdillah, R.M., Xin, Q. and Swidan, M.B. (2019b). SCSA: Evaluating skyline queries in incomplete data. Applied Intelligence, 49(5), 1636–37.

Gulzar, Y., Alwan A. A. and Shezad Turaev. (2019c). Optimizing skyline query processing in incomplete data. IEEE Access Journal, 7(1) 178121–38.

Gulzar, Y., Alwan, A.A., Ibrahim, H. and Xin, Q. (2018). D-SKY: A Framework for Processing Skyline Queries in a Dynamic and Incomplete Database. In: Proceedings of the 20th International Conference on Information Integration and Web-based Applications & Services (iiWAS2018), Yogyakarta, Indonesia, 19-21/11/2018.

Gulzar, Y., Alwan, A.A., Salleh, N. and Al Shaikhli, I.F. and Alvi, S.I.M (2016a). A framework for evaluating skyline queries over incomplete data. Procedia Computer Science, 94(n/a), 191–8.

Gulzar, Y., Alwan, A.A., Salleh, N. and Al Shaikhli, I.F. (2016b). A Model for Skyline Query Processing in a Partially Complete Database. In: Proceedings of the 3rd International Conference on Computational Science and Technology (ICCST 2016), Kota Kinabalu, Malaysia, 29-30/11/2016.

Gumaei, A., Sammouda, R. and Al-Salman, A.S. (2017). An Efficient Algorithm for K-Rank Queries on Large Uncertain Databases. International Journal of Computer Science and Network Security (IJCSNS), 17(4), 129–32.

Khalefa, M.E., Mokbel, M.F. and Levandoski, J.J. (2008). Skyline query processing for incomplete data. p. 556-566. In: In: Proceedings of the 24th International Conference on Data Engineering, Cancun, Mexico, 7-12/4/2008.

Kontaki, M., Papadopoulos, A.N. and Manolopoulos, Y. (2010). Continuous processing of preference queries in data streams. In: Proceedings of the International Conference on Current Trends in Theory and Practice of Computer Science, Berlin, Heidelberg, 23-29/1/2010. p. 47-60.
Kou, N. M., Li, Y., Wang, H., U, I.H, and Gong, Z. (2017). Crowdsourced top-k queries by confidence-aware pairwise judgments. In: Proceedings of the 2017 ACM International Conference on Management of Data, Chicago, Illinois, USA, 14-19/5/2017. p. 1415–30.

Lee, J., Im, H. and You, G.W. (2016). Optimizing skyline queries over incomplete data. Information Sciences, 361(n/a), 14–28.

Lee, J., Lee, D. and Hwang, S.W. (2017). CrowdK: Answering top-k queries with crowdsourcing. Information Sciences, 399(n/a), 98–120.

Lee, J., Lee, D. and Kim, S.W. (2016). Crowdsky: Skyline computation with crowdsourcing. p. 125-136. In: EDBT. Proceedings of the 19th International Conference on Extending Database Technology, Bordeaux, France, 15–16/3/2016.

Li, G., Chai, C., Fan, J., Weng, X., Li, J., Zheng, Y., Li, Y., Yu, X., Zhang, X. and Yuan, H. (2017). CDB: optimizing queries with crowd-based selections and joins. p. 1463–78. In: Proceedings of the 2017 ACM International Conference on Management of Data, Chicago, IL, USA, 14–19/5/2017.

Li, G., Wang, J., Zheng, Y. and Franklin, M.J. (2016). Crowdsourced data management: A survey. IEEE Transactions on Knowledge and Data Engineering, 28(9), 2296–319.

Lian, H., Qiu, W., Yan, D., Huang, Z. and Tang, P. (2020). Efficient and secure k-nearest neighbor query on outsourced data. Peer-to-Peer Networking and Applications, n/a(n/a), 1–10.

Lin, X., Xu, J., Hu, H. and Fan, Z. (2017). Reducing uncertainty of probabilistic top-k ranking via pairwise crowdsourcing. IEEE Transactions on Knowledge and Data Engineering, 29(10), 2990–303.

Lofi, C., Maary, K. and Balke, W. (2013). Skyline queries in crowd-enabled databases. p. 465-476. In: EDBT ’13. Proceedings of the 16th ACM International Conference on Extending Database Technology, Genoa, Italy, 18-22/3/2013.

Marcus, A., Wu, E., Karger, D.R., Madden, S. and Miller, R.C. (2011). Crowdsourced databases: Query processing with people. In: Proceedings of the 5th Biennial Conference on Innovative Data Systems Research (CIDR 2011), Asilomar, California, USA, 9-12/1/2011. p. 211-214.

Miao, X., Gao, Y., Guo, S., Chen, L., Yin, J. and Li, Q. (2019). Answering Skyline Queries over Incomplete Data with Crowdsourcing. IEEE Transactions on Knowledge and Data Engineering, 31(4), 1.

Mozaftari, B., Sarkar, P., Franklin, M., Jordan, M. and Madden, S. (2014). Scaling up crowd-sourcing to very large datasets: a case for active learning. Proceedings of the VLDB Endowment, 8(2), 125–36.

Parameswaran, A. and Polyzotis, N. (2011). Answering queries using humans, algorithms and databases. In: Proceedings of the 35th Biennial Conference on Innovative Data Systems Research, Asilomar, California, USA, 9-12/1/2011. p. 160–6.

Parameswaran, A., Park, H., Garcia-Molina, H., Polyzotis, N. and Widom, J. (2012). Deco: Declarative Crowdsourcing. p. 1203-1212. In: CIKM ’12. Proceedings of the 21st ACM International Conference on Information and Knowledge, Maui, HI, USA, 29/10–2/11/2012.

Parameswaran, A.G., Garcia-Molina, H., Park, H., Polyzotis, N., Ramesh, A. and Widom, J. (2012b). Crowdscreen: algorithms for filtering data with humans. In: SIGMOD ’12. Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, Scottsdale, Arizona, USA, 20-24/5/2012. p. 361–72.

Polychronopoulos, V., DeAlfaro, L., Davis, J., Garcia-Molina, H. and Polyzotis, N. (2013). Human-Powered Top-k Lists. In: WebDB. Proceedings of the 16th International Workshop on the Web and Databases, New York, NY, USA, 23/6/2013. p. 25–30.

Sarma, A.D., Parameswaran, A., Garcia-Molina, H. and Halevy, A. (2014). Crowd-powered find algorithms. In: Proceedings of the 30th International Conference on Data Engineering, Chicago, IL, USA, 31/3–4/4/2014. p. 964-975.

Swidan, M.B., Alwan, A.A., Turaev, S. and Gulzar, Y. (2018). A model for processing skyline queries in crowd-sourced databases.