Multimed Tools Appl (2010) 46:399–423
DOI 10.1007/s11042-009-0372-y

A computation-oriented multimedia data streams model for content-based information retrieval

Shi-Kuo Chang · Lei Zhao · Shenoda Guirguis · Rohit Kulkarni

Published online: 20 October 2009

Abstract Multimedia applications nowadays are becoming prevalent. In the past the relational database model was generalized to the multimedia database model. More recently the relational database model was generalized to the data streams model, as the technology advanced and data became bulky and unbounded in size due to the utilization of sensor networks. In this paper we take one more step of generalization by providing a multimedia data streams model. The objective is to furnish a formal framework to design multimedia data streams (MMDS) schema for efficient content based information retrieval. We also extend the functional dependency theory and the normalization framework to handle multimedia data streams. Finally we present algorithmic methods of generating continuous multimedia queries along with examples for illustration.

Keywords Multimedia data streams model · Multimedia dependency theory · Continuous queries · Content-based information retrieval

1 Introduction

Multimedia databases have been used in many applications for over two decades now. The internet boom has accelerated this trend, introducing many new applications related to query processing and content based retrieval. In [6], a normalization framework for

S.-K. Chang (✉) · S. Guirguis
Department of Computer Science, University of Pittsburgh, Pittsburgh, PA 15260, USA
e-mail: chang@cs.pitt.edu

S. Guirguis
e-mail: shenoda@cs.pitt.edu

L. Zhao
School of Computer Science and Technology, Soochow University, Suzhou 215006, China
e-mail: zhaol@suda.edu.cn

R. Kulkarni
Department of Information Science, University of Pittsburgh, Pittsburgh, PA 15260, USA
e-mail: rok26@pitt.edu

multimedia databases was provided as a generalization to the traditional relational database model. The data dependency theory was also extended to include dependencies involving different types of multimedia data. Recently a new research area is being proposed called human computation [12], which utilizes the ultimate powerful computational units all humans possesses; i.e. human brain. Starting with CAPTCHA, researchers proposed multimedia games that are applications utilizing multimedia databases and human computational power in solving difficult problems [12]. On the other hand the data stream management concept was first introduced in [8], and later comprehensive stream management systems were proposed such as STREAM [3], Borealis [2] and Aurora [1]. Several commercial data stream management systems (DSMSs) are now available [11], [13]. Data streams management systems are expected to be able to handle huge massive updates of data collected from sensor networks or monitoring applications. In the push-model for data streams management, data updates arrive with high frequency, while a certain set of queries (called continuous queries) reside in the data stream server to process incoming data [1]. In other words, users will first register their continuous queries (CQs), and data will be pushed into these queries as they arrive. This model is in contrast to the pull-model for traditional database management. Data streams have many critical and important applications that range from network monitoring applications, military applications, to environmental monitoring applications. Data streams became even more important with the advances of the technology of sensor networks consisting of small devices that can be spread over a large area to collect time-critical information. As sensor networks become prevalent, multimedia data streams will also become massive and unbounded. It is thus critical to provide a framework for multimedia data streams (MMDS) modeling and performance estimation, based upon which we can provide performance guarantees such as minimum output rate per query, maximum supported queries per site, maximum supported streams per site and so on. The multimedia dependency theory should also be extended to include dependencies among multimedia data streams.

In this paper we provide a multimedia data streams model together with an extended functional dependency and normalization framework. We first give the formalization of the multimedia data streams problem. The objective is to furnish a formal framework to design multimedia data streams (MMDS) schema for content based information retrieval. We also extend the functional dependency theory and the normalization framework to efficiently handle multimedia data streams. Finally we present algorithmic methods of generating continuous multimedia queries, along with examples for further illustration of querying multimedia data streams in potential critical applications.

The paper is organized as follows. Three motivating examples are presented in Section 2. A focused review of selected related works are given in Section 3. Section 4 provides the mathematical model of multimedia data streams. Performance estimates are presented in Section 5, and a concrete example is explained in detail. Dependency Theory is extended in Section 6. Algorithms for Multimedia Data Streams Continuous Querying are presented in Section 7, and illustrative examples are given in Section 8 to demonstrate the application of these algorithms. Some discussions on further extension of the model, the extended dependency theory and normalization framework for ontological filtering are given in Section 9.

2 Motivating examples

As motivating examples we present the following three applications — a security system, a health application, and an aircraft database system.

Motivating example 1 — security system A security system usually has video cameras installed in hidden places in some secured building or store. Typically a guard or two keep rotating between the captured videos and they interfere when something suspicious or a threat occurs. However, this security system has two major problems: first, inevitable human error, and second, if a WANTED person shows up, it is less likely that the security guard can pay attention, and hence interfere before the crises (as opposed to after the crises). However, if we have a multimedia data stream management system that receives the stream of videos (or stream of frames) and has a relation (i.e. a table) with images of all WANTED criminals, and another relation with images of all weapons. Then one can pre-register couple of useful continuous queries to automate the task of the guards, and help them do their job and avoid human errors. Such CQs may include:

1) Tell me whenever you see an object similar to any weapon.
2) Tell me whenever you see a WANTED person.
3) Tell me whenever an object (or more) in several frames within the last 30 seconds moves in a very "violent" manner.
4) If any of the above, start recording the video on a "Threat" clip for future reference.

Thus, if the guard missed anything, or had to leave his place for whatever reason, the system can help him. Moreover, new things that he was not able to do, such as identifying WANTED criminals, is now achievable.

Motivating example 2 — health-care system One of the major data streams applications is the health monitoring systems, where the patient's heart rate, body temperature, etc. are continuously monitored, and a CQ to report when these values go beyond the normal values. Imagine the case for some diseases when an X-Ray photo needs to be inspected to show progress or to identify anomalies. This is a multimedia data stream that can be fed to a multimedia data stream management system with the appropriate continuous queries.

Motivating example 3 — airport security Assume we have a multimedia database with multimedia objects being still images of object types. And then we have a multimedia stream of video frames taken by some airport's security cameras. The images are taken inside the terminal, and outside as well. So, the input multimedia stream might contain frames that contain objects. It might be useful to recognize the object type and the location, to make sure that everything is going accordingly, and there are no violations of the airport rules. This could be modeled as a multimedia continuous query. The advantage of applying a multimedia data stream approach is the ability to have up-to-date rules, and allowing exceptions in emergency cases. Yet the precision and the 24 hours monitoring advantages are also gained as opposed to relying only on human monitoring.

3 A focussed review of related works

There are a massive amount of research on both multimedia databases and data streams. For multimedia databases, the work [6] is the inspiring work of this paper, as in [6] the dependency theory was generalized for multimedia databases, and a normalization framework was proposed. Other approaches in content-based retrieval and algorithms have also been proposed in the multimedia database literature. The literature of data streams is focused on performance fine tuning; minimizing response time (as a QoS metric), or

improving freshness (as a QoD metric) [15]. Also, scaling with burst arrivals of data and peak loads gained some attention (known as the problem of load shedding) [17]. Some work was also proposed for mining data streams [9]. There are indeed many issues and interesting applications over data stream, such as the representation of infinite stream of data, continuous queries over data streams, and performance tuning and so on. Since the literature is vast, rather than giving a superficial review on many papers, we will provide a focused review by discussing in detail three representative papers that address some interesting and basic issues on continuous query with substantial achievements. Hopefully this focused review will enable the reader to gain more insight into the complexities of the research topics.

Sensor networks and monitoring applications provide increasingly fine-grained data, which will be encoded as data stream. Such data is huge and largely transient — so much of it will exist only as streams rather than be permanently recorded in raw form. Moreover, multimedia applications nowadays proliferated in many fields and gave birth to multimedia databases. We are interested in modeling multimedia data streams and constructing continuous multimedia queries over multimedia data streams. So we did a systematic investigation about data stream and continuous query, and this focused review is concerned with these issues.

The problem of scheduling multiple heterogeneous CQs (Continuous Queries) in a DSMS (Data Stream Management System) with the goal of optimizing QoS for end users and applications is considered in [16]. To quantify such QoS, the authors first used the traditional metric of response time, which is defined over multiple CQs, including CQs that contain joins of multiple data streams. The authors also considered slowdown as another QoS metric, since they believe it to be a more fair metric for heterogeneous workloads, and, as such, more suitable for a wide range of monitoring applications.

Then this paper develops new scheduling policies that optimize the average-case performance of a DSMS for response time and for slowdown. Additionally, it proposes hybrid policies that strike a fine balance between the average-case performance and the worst-case performance, thus avoiding starvation. Furthermore, this paper extends the proposed policies to exploit operator sharing in optimized multi-query plans and to handle multi-stream queries. It also argues an adaptive scheduling mechanism that allows the proposed policies to react quickly to changes in data distribution. And finally, it evaluates the proposed policies and their implementation experimentally, which shows that the scheduling policies in this paper consistently outperform previously proposed policies.

The CQL language and execution engine for general-purpose continuous queries over data streams and stored relations is presented in [4]. CQL (for continuous query language) is an instantiation of a precise abstract continuous semantics, and CQL is implemented in the STREAM prototype data stream management system at Stanford, including the “Linear Road” benchmark used as examples throughout this paper.

This paper initially defines a precise abstract semantics for continuous queries. This abstract semantics is based on two data types — streams and relations — and three classes of operators over these types: operators that produce a relation from a stream (stream-to-relation), operators that produce a relation from other relations (relation-to-relation), and operators that produce a stream from a relation (relation-to-stream). The three classes of operators are “black-box” components of this abstract semantics, which means the abstract semantics does not depend on the actual behavior of the operators in these classes, but only on their input and output types.

This paper also describes the structure and implementation of query execution plans for CQL in the STREAM system, which is available for experimentation over the Internet. For

each user, a dedicated server is started on a machine at Stanford and a client is started on the user's machine. Through the graphical user interface, users may register streams and continuous queries and view the streamed (or stored) results. Users may also inspect and alter query plans and perform visual system monitoring through "introspection": query components write statistics (such as throughput, selectivity, etc.) onto a special system stream. Graphical system monitors obtain their plotted values by registering standard CQL queries on the special system stream.

The paper [14] begins the discussion of a SQL-based standard for streaming databases. It discusses some deep model differences that exist between Oracle CQL and StreamBase StreamSQL. Then it proposes a new and unification model that uncovers these differences. The key insight of this model is that evaluation of results in both systems is triggered by the arrival of a batch of tuples. In the Oracle model the batch is defined by like values of a timestamp. In the StreamBase model, batches are always of size one tuple. By controlling the batching of tuples and the ordering between these batches, the proposed model can simulate both models plus many alternatives that neither model can capture. Moreover, this paper also presents the syntax and semantics of a new operator called SPREAD, a powerful stream-to-stream operator, and illustrates it using examples.

On the one hand, with the voluminous data streams arriving from data communications or sensor networks, many interesting applications in this area will emerge. So how to schedule continuous queries and optimize the performance becomes vital to these applications. The first paper investigated these issues and got many outstanding results. The work in this article gives the inspiration to their future works and many other related researches in this area. However it is necessary to develop policies that are able to balance the trade-off between different QoS metrics as well as QoD metrics.

On the other hand, many full-blown data stream management systems are proposed in recent years. It brings a lot of differences among these existed prototypes and products. So many researchers are devoting themselves to solve the problems on some unified or standard data stream model and the corresponding continuous query language. The other two papers are the representative works in this area. These works have the added benefit that they increase the expressive power of both languages and give the user an easy-to-use interface to work on data stream. However there are still many remaining problems that could be investigated on the road to a complete standard.

In conclusion, there are many key issues in the management of data streams, and the three papers presented in this focused review address only a handful of the topics in this area. These articles did investigate the data stream model, syntax and semantics of continuous query language, execution engine for continuous queries over data streams, scheduling multiple CQs in DSMS, and optimizing QoS for end users and applications. In addition, multimedia databases have been used for over two decades. Though data stream model and multimedia database are both well-studied concepts, it is rarely reported that they have been integrated for the purpose of modeling multimedia data streams to the best of our knowledge. We anticipate that the growth in the use of smart sensors, microprocessors, networks, and the World Wide Web will fuel an explosion of demand for multimedia data stream management systems in the coming decade. And more research is needed to support the design of such systems.

Our work differs from the papers described above in that we want to provide a computation-oriented model of multimedia data streams. In addition, the dependency theory is generalized for multimedia databases, and a normalization framework is proposed so that query processing can be carried out with respect to this framework. Our preliminary work was first reported in [10]. In the present paper the computation-oriented model is

completely formalized (see Section 4), and substantial results on performance estimation are obtained (see Section 5). Therefore instead of vaguely formulated bounds, we can now give a concrete example on performance estimation to show how this is done (see Section 5.7).

4 A computation-oriented multimedia data streams model

In this section we will give some definitions first, and then we will provide the mathematical model. In Section 5 we will give a concrete example.

After we provide a list of the terminology used, we define some mathematical terms. The definitions 1 through 6 serve the first part, while the rest of the definitions serve the second part.

Definition 1 *Primary Data Type* Primary Data Type is an indivisible data structures. Let T denote the entire set of data types and let t_i denote one type of them.

There are four types of primary data type defined in MMDS. They are : int, float, timestamp and micon. The anterior three of them are commonly used in traditional data models and mocon is a different one. Int and float are two basic calculable data types. They can be calculated under basic arithmetical operators. Timestamp is an extended calculable type. Usually it is call datetime type. Some operators were defined to calculate the difference between two values of datetime type and others are used to convert the format of them in common used data models. Many DBMS of relational data model are critical samples. They all use these three types.

Micon is not commonly used in traditional data models. But in MMDS, it is a basic type. The definition of micon is given in Definition 2.

Definition 2 *Micon* A Micon is a multimedia icon that could be: text (ticon), still image (icon), audio (earcon), video (vicon).

Definition 3 *Data Stream* A Data Stream is a constructed data type. It is a huge sequence of tuples according to a certain schema that keeps arriving to a Data Stream Management System. Each tuple has a unique identifier for verification and a timestamp for ordering. Each tuple is composed of properties and each property should be in one of the primary data type defined in Definition 1.

Definition 4 *Multimedia Data Streams* A Multimedia data stream (MMDS) is a data stream that contains at least one Micon as one of its attributes, according to a certain schema.

Definition 5 *Operator* Operators are some indivisible query operators in database management systems. Let O denote the entire set of operators and let o_i denote one operator of them.

SQL statements are the most widely used query language in traditional relational DBMSs. SQL statements can be represented by some primary operators. They are : π, σ,$\times$ and ∞. So that SQL statements can be written as algebra expressions. Followings are brief definitions of the four primary operators.

Projection(π): The projection operator is a unary operator that is parameterized by a list L of positive integers, and is denoted . When applied to a k-ary relation R, we require the elements of L to be in the range 1, ..., k, with no duplicates. The result of applying π_L to R

is the set of tuples formed by restricting the tuples in R to the attributes in L. For example, for a 5-ary relation R,

$$\pi_{124}R = \{(a,b,d)|\exists c,e : (a,b,c,d,e) \in R\}.$$

Selection(σ): The selection operator is a unary operator that is parameterized by a simple predicate θ, and is denoted σ_θ.. .When applied to a k-ary relation R, The result of applying σ_θ to R is the set of tuples in R that satisfy the predicate θ. For example, for a 5-ary relation R,

$$\sigma_{\#2<5}R = \{(a,b,c,d,e) \in R|b < 5\}.$$

where we use the notational convention of prefixing attribute indices with the # character to distinguish them from literals.

Natural Join(∞): In the named algebra, it is often convenient to use a join predicate that equates like-named attributes of its operands and discards one attribute from each pair of like-named attributes. This operator is called the natural join and denoted by ∞. Thus the absence of a predicate below the ∞ symbol does not denote a vacuously satisfied predicate as one may expect; rather, it denotes a predicate that equates the like-named attributes of the operands. For example, with relation schemes R(a, b, c, d, e) and S(d, e, f),

$$R \infty S = \pi_{a,b,c,d,R.e,f}\sigma_{R.d=S.d}(R \times S)$$

Conversion operator(ψ): Conversion operator is used to process multimedia information. It can convert information from one form to another. For example, with relation schemes R (a, b, c, d, e), and a is in type of icon, which means values of a in tuples of R are a set of images, then

$$\psi_{type}(car)(\pi_a(R)) = \psi_{type}(car)(R.a)) = true/false$$

if there is a car in the images. It is a powerful operator in multimedia information processing systems.

Mergence operator(+): Mergence operator is used to merge several streams with the same structure into one for higher output rate.

For example, there are two streams of S_1 and S_2,

$$+ : (S_1, S_2) \rightarrow S_x \text{ or } +(S_1, S_2) = S_x.$$

Fusion operators can be nested if there are more than two streams wanted to be fused. For example,

$$+(S_1, S_2, S_3) = +(+(S_1, S_2), S_3) = +(S_x, S_3) = S_y.$$

So the general form of $\phi+$ operator should be,

$$+(S_1, S_2, ..., S_3) = +(+(... + (S_1, S_2), ...), S_n)$$

Definition 6 *Continuous Query* A Continuous Query (CQ) is a constructed operator. It is a query registered by a user at the MMDSMS that is to be executed — theoretically — forever. If the CQ includes one or more multimedia operators; conversion or fusion operators — as defined before — then the CQ is called a multimedia CQ, or m-CQ for short.

Definition 7 *Cost of Operator* Cost of operator means the total time in terms of processor time to process a single tuple through an operator while the processor is exclusive. Let c_i denote the cost of o_i.

Definition 8 *Cost of CQ* Cost of continuous query means the total time to process a single tuple through an m-CQ. Let C_i denote the cost of $m–CQ_i$.

Definition 9 *Output Rate of operator* Output rate of operator means the speed of producing output tuples of an operator while the processor is exclusive. Let r_i denote the output rate of $m–CQ_i$. Apparently,

$$r_i = \frac{1}{c_i} \text{ or } c_i = \frac{1}{r_i}$$

Definition 10 *Output Rate of CQ* Output rate of CQ means the speed of producing output tuples of m-CQ. Let R_i denote the output rate of $m–CQ_i$.

The computation-oriented multimedia data stream model (abbr. COMDSM) is illustrated in Fig. 1. We are now ready to discuss the performance estimation of m-CQ's.

5 Performance estimation

5.1 A processing model

Figure 2 is the framework of our processing model. The parts with real line frame are input. The parts with dashed frame are midterm steps. The gray parts are expected results.

In Phase 1, m-CQ's are the results. In Phase 2, some computational results are what we need. We can come up with some conclusions and suggestions by the computational results.

5.2 Problem formalization

Given an m-CQ, it is required to compute a bound of performance, such as cost and output rate. We formalize an m-CQ to a weighted DAG for the convenience of computation.

Figure 3(a) shows an example of m-CQ. Each rectangle represents an operator and each arrow represents a data stream. s_1 to s_9 are source data streams and other arrows are

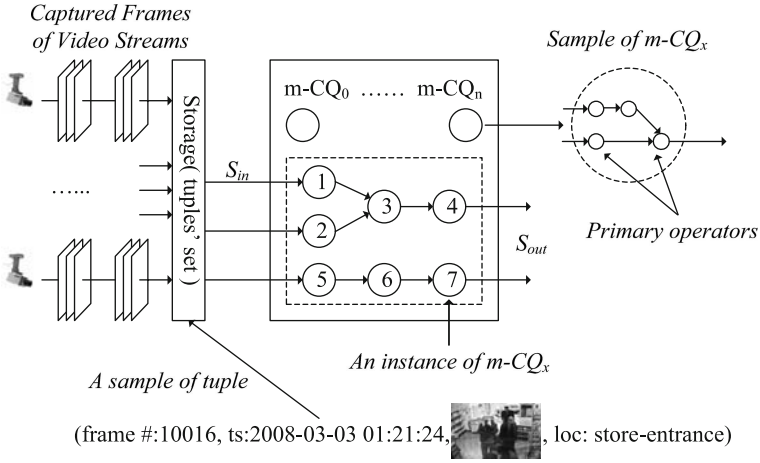


Fig. 1 Computation-oriented multimedia data stream model

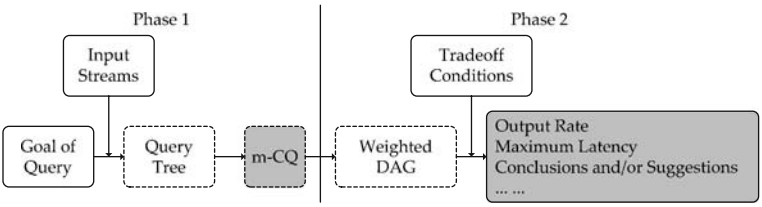


Fig. 2 Processing model

processed data streams of operators. Actually, there is no difference between these two kinds of data streams in character technically. A processed data stream of one operator can also act as a source data stream of another operator. The number marked on each arrow is the rate of the data stream.

Figure 3(b) shows a corresponding weighted DAG. Each vertex with a centered number represents an operator. We add the following kinds of vertices to the DAG. Vertices with tag s_i mean start points and the vertex with tag T means terminal. A DAG may have more than one start point for an m-CQ may have more than one source data stream. A DAG has only one terminal for an m-CQ has only one output data stream.

5.3 Cost of m-CQ

To calculate the cost of an m-CQ, firstly, we should present the definition of computation chain which comes from its corresponding weighted DAG.

Definition 11 *Computation Chain* Given a weighted DAG, corresponding to a certain m-CQ, a computation chain means a path from one vertex, except for any input source, to another vertex, except for the terminal, denoted

$$\left\{o_{k,1}, o_{k,2}, \ldots o_{k,m}\right\}, o_{k,i} \in O(1 \leq i \leq m, k = 1, 2, \ldots).$$

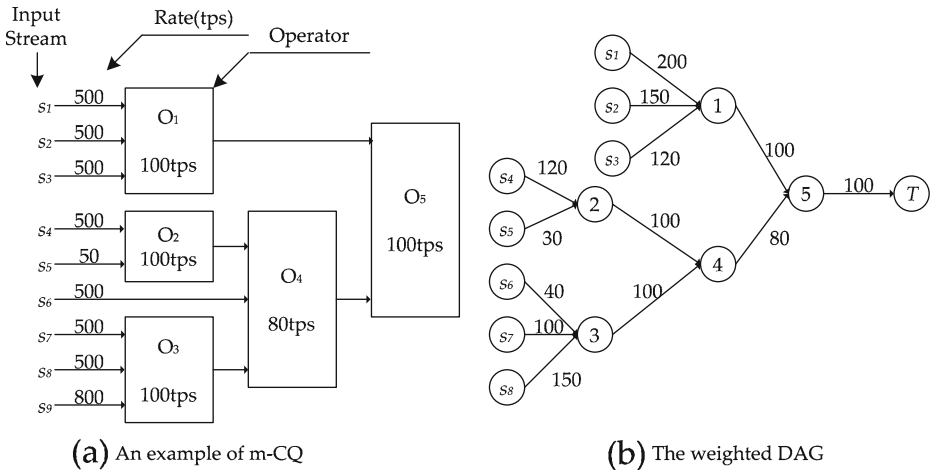


Fig. 3 Performance estimating model

Suppose, an m-CQ has p operators and $\{o_{k,1}, o_{k,2}, ...o_{k,m}\}$ is a computation chain, the cost of this chain can be figured out by

$$C_{\{o_{k,1},o_{k,2},...o_{k,m}\}} = \sum_{i=1}^{m} c_{k,i}.$$

Since

$$c_{k,i} = \frac{p}{r_{k,i}},$$

therefore

$$C_{\{o_{k,1},o_{k,2},...o_{k,m}\}} = \sum_{i=1}^{m} \frac{p}{r_{k,i}}.$$

Suppose, $m–CQ_i$ has n different computation chain, the cost of $m–CQ_i$ is the maximum cost of all computation chains, denoted as

$$C_i = \max\left(C_{\{o_{k,1},o_{k,2},...o_{k,m}\}}\right)(1 \le k \le n).$$

The corresponding computation chain is called critical computation chain. An m-CQ may have more than one critical computation chain.

For example, the cost of the m-CQ in Fig. 3 equals the cost of computation chain $\{o_2,o_4, o_5\}$. So we can get

$$\begin{aligned} C &= \tfrac{1}{r_2} + \tfrac{1}{r_4} + \tfrac{1}{r_5} \\ &= \tfrac{5}{100} + \tfrac{5}{80} + \tfrac{5}{100}. \\ &= 0.1625 \text{ sec} \end{aligned}$$

5.4 Output rate

To calculate the output rate of an m-CQ, firstly, we should present the definition of performance chain which also comes from its corresponding weighted DAG.

Definition 12 *Performance Chain* Given a weighted DAG, corresponding to a certain m-CQ, a performance chain means a path from any start point to the terminal, denoted

$$\{s_x, o_{k,1}, o_{k,2}, ...o_{k,m}, T\}, o_{k,i} \in O(1 \le i \le m, k = 1, 2, ...).$$

Two kinds of vertices can be treated as start point.

(1) An input source, if there is no merging operation in the chain.
(2) The last mergence operation, which is the nearest vertex to the terminal in the chain.

Suppose, an m-CQ has p operators and $\{s_x, o_{k,1}, o_{k,2}, ...k,m, T\}$ is a performance chain, the output rate of this chain can be figured out by

$$R_{\{s_x,o_{k,1},o_{k,2},...o_{k,m},T\}} = \min\left(r_{s_x}, \frac{r_{o_{k,1}}}{p}, \frac{r_{o_{k,2}}}{p}, ..., \frac{r_{o_{k,m}}}{p}\right).$$

Suppose, m–CQ_i has n different performance chains, the output rate of m–CQ_i is the minimum output rate of all performance chains, denoted as

$$R_i = \min\left(R_{\{s_x,o_{k,1},o_{k,2},\ldots o_{k,m},T\}}\right)(1 \le k \le n)$$

The corresponding performance chain is called critical performance chain. An m-CQ may have more than one critical performance chain.

For example, the output rate of the m-CQ in Fig. 3 equals the output rate of performance chain $\{s_5,o_2, o_4, o_5, T\}$. So we can get

$$\begin{aligned} R &= \min\left(r_{s5}, \tfrac{r_2}{5}, \tfrac{r_4}{5}, \tfrac{r_5}{5}\right) \\ &= \min(30, 20, 16, 20) \\ &= 16\text{tps} \end{aligned}$$

5.5 Tradeoff in distributed processing situation

Suppose, an m-CQ has p operators which can be distributed to $g(g>1)$ computing nodes, the output rate in this situation lies on the assignment of the p operators. The assignment can be denoted as

$$A = \left\{\left\{o_{1,1}, o_{1,2}, \ldots, o_{1,d_1}\right\}, \left\{o_{2,1}, o_{2,2}, \ldots, o_{2,d_2}\right\}, \ldots, \left\{o_{g,1}, o_{g,2}, \ldots, o_{g,d_g}\right\}\right\}$$

for which $\forall o_{x_1,y_1}, o_{x_2,y_2} \in A, o_{x_1,y_1} \ne o_{x_2,y_2}$ *iff* $x_1 \ne x_2$ or $y_1 \ne y_2$ and $\sum_{i=1}^{g} d_i = p$.

Suppose, the performance chain is $\left\{s_x, o_{x_1,y_1}, o_{x_2,y_2}, \ldots o_{x_m,y_m}, T\right\}$, then the output rate of this chain can be figured out by

$$R^k_{\left\{s_x,o_{x_1,y_1},o_{x_2,y_2},\ldots o_{x_m,y_m},T\right\}} = \min\left(r_{s_x}, \frac{r_{o_{x1,y1}}}{d_{x_1}}, \frac{r_{o_{x2,y2}}}{d_{x_2}}, \ldots, \frac{r_{o_{xm,ym}}}{d_{x_m}}\right).$$

Suppose, m–CQ_i has n different performance chains, the output rate of m–CQ_i in this situation is the minimum output rate of all performance chains, denoted as

$$R_i = \min\left(R^k_{\left\{s_x,o_{x1,y1},o_{x2,y2},\ldots o_{xm,ym},T\right\}}\right)(1 \le k \le n)$$

Figure 4 shows an example of 2-processor situation. In Fig. 4, o_2 and o_3 are assigned to one processor and other three operators are assigned to another processor. The dashed chain in Fig. 4(a) means a possible critical line and that in Fig. 4(b) means a possible critical full line.

Now we can get the cost of the weighted DAG in Fig. 4 is

$$C = \frac{1}{r_2} + \frac{1}{r_4} + \frac{1}{r_5} = \frac{1}{50} + \frac{1}{27} + \frac{1}{33} = 0.087 \text{ sec},$$

and the output rate is

$$R = \min\left(r_{s5}, \frac{r_2}{2}, \frac{r_4}{3}, \frac{r_5}{3}\right) = \min(30, 50, 27, 33) = 27\text{tps}.$$

Figure 5 shows another example of 2-processor situation. In Fig. 5, o_2 and o_4 are assigned to one processor and other three operators are assigned to another processor.

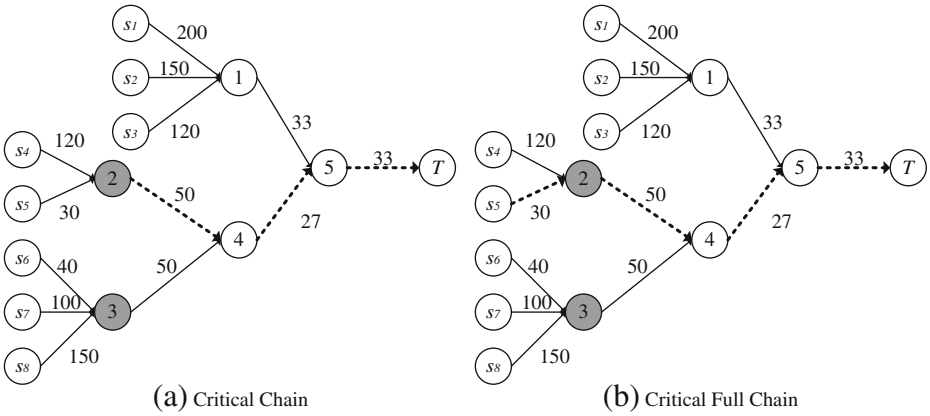


Fig. 4 Example 1 of tradeoff situation

Now we can get the cost of the weighted DAG in Fig. 5 is

$$C = \frac{1}{r_3} + \frac{1}{r_4} + \frac{1}{r_5} = \frac{1}{33} + \frac{1}{40} + \frac{1}{33} = 0.085 \text{ sec},$$

and the output rate is

$$R = \min(r_{s5}, \frac{r_2}{2}, \frac{r_4}{2}, \frac{r_5}{3}) = \min(30, 50, 40, 33) = 30\text{tps}.$$

5.6 Inverse solution

Now we can get the cost of the weighted DAG in Fig. 6(b) is

$$C = \frac{1}{r_3} + \frac{1}{r_4} + \frac{1}{r_5} = \frac{1}{50} + \frac{1}{80} + \frac{1}{50} = 0.0525\,\text{sec},$$

and the output rate is

$$R = \min\left(r_{s5}, \frac{r_2}{2}, \frac{r_4}{1}, \frac{r_5}{2}\right) = \min(60, 50, 80, 50) = 50\text{tps}.$$

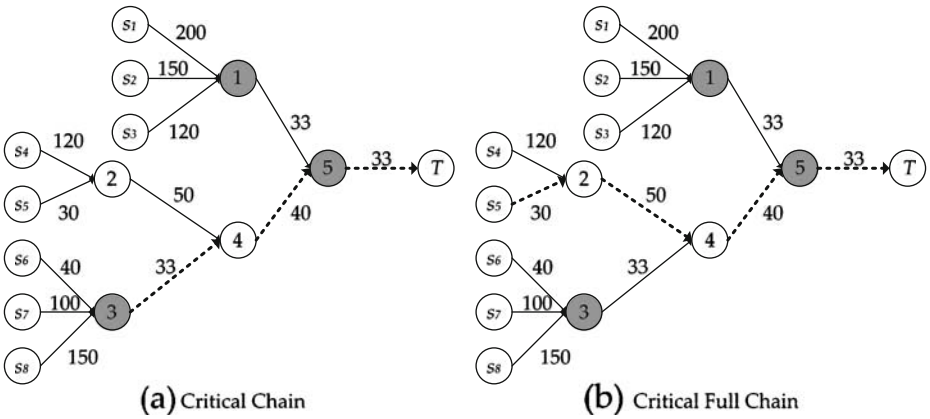


Fig. 5 Example 2 of tradeoff situation

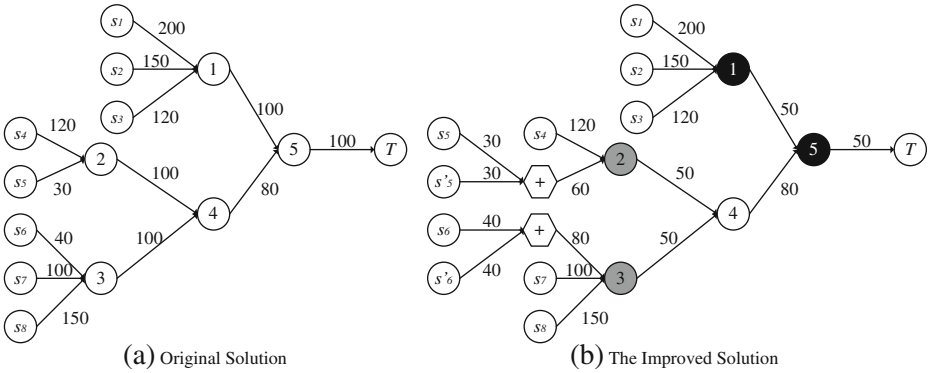


Fig. 6 Example of inverse solution

5.7 A concrete example

Consider the following scheme: Imagine a stream of video frames captured from a certain security monitoring camera:

video (frame-number, time-stamp, one-frame-of-video-data, location_id).

The frame-number attribute here becomes the unique identifier of the video frame (a tuple in the stream).

A possible m-CQ_1 could be: Notify me all video frames, received from the security monitoring cameras, that contain a weapon. Clearly this m-CQ includes the following operations (or manipulations) over the video multimedia streams:

1) For all frames that go through a conversion operator should be utilized to detect weapon objects.
2) Merging the streams into one stream since all the streams have the same structure.
3) Selecting the frames for which the values of weapon_type attribute are not null.

The Query Graph G of the m-CQ (or the Computational structure in other words) is illustrated in Fig. 7(a), and its corresponding weighted DAG is in Fig. 7(b).

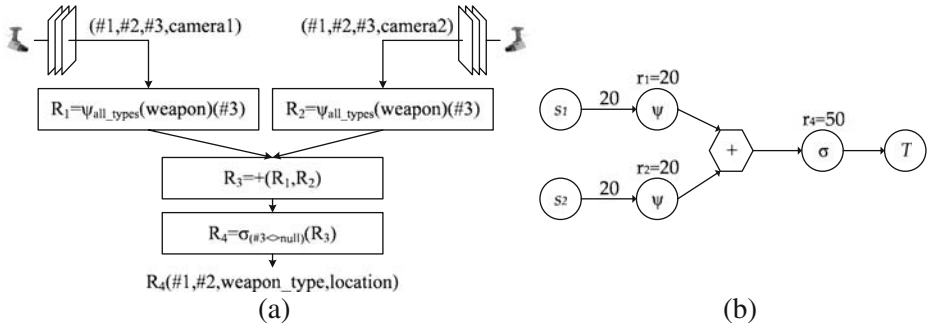


Fig. 7 Illustration of m-CQ_1. (**a**) Query graph (**b**) Corresponding weighted DAG

Now, assuming the output rate of this computational network/structure is R_1. As we can see, $\{o_1, o_3, o_4\}$ and $\{o_2, o_3, o_4\}$ are two computation chains and a merging operation o_3 precedes o_4, it follows that

$$R_1 = \min\left(\sum_{i=1}^{2} r_{\{s_i,o_i\}}, r_4\right) = \min\left(\sum_{i=1}^{2} \min(r_{s_i}, r_{o_i}), r_4\right).$$

Therefore,

$$\begin{aligned} R_1 &= \min\left(\left(\min\left(r_{s1}, \tfrac{r_1}{3}\right) + \min\left(r_{s2}, \tfrac{r_2}{3}\right)\right), \tfrac{r_4}{3}\right) \\ &= \min\left(\left(\tfrac{20}{3} + \tfrac{20}{3}\right), \tfrac{50}{3}\right) \\ &= 13.3\text{tps} \end{aligned}$$

If we consider a distributed processing situation, let us assume o_1 and o_2 share one processor, o_4 use one processor exclusively. The corresponding weighted DAG is illustrated in Fig. 8(a). Now, we can get

$$\begin{aligned} R_1 &= \min\left(\left(\min\left(r_{s1}, \tfrac{r_1}{2}\right) + \min\left(r_{s2}, \tfrac{r_2}{3}\right)\right), \tfrac{r_4}{1}\right) \\ &= \min\left(\left(\tfrac{20}{2} + \tfrac{20}{2}\right), \tfrac{50}{1}\right) \\ &= 20\text{tps} \end{aligned}$$

If assuming all the cameras have the same output rate and we have a requirement that the output rate of this m-CQ must be no less than 1 frame/(sec.camera), i.e., if we have n cameras, then

$$R_1 \geq n \text{ tps}.$$

If o_1, o_2,..., o_n always share one processor, then it is not hard to see,

$$\sum_{i=1}^{n} r_{\{s_i,o_i\}} \equiv 20\text{tps}.$$

That is, the output rate of o_3 has no chance to be above 20 tps, as a result, $R_1 \leq 20$ tps. A derived DAG is illustrated in Fig. 8(b). In this situation, the m-CQ can support up to 40 cameras with the guarantee of the requested output rate. Two processors are used to process

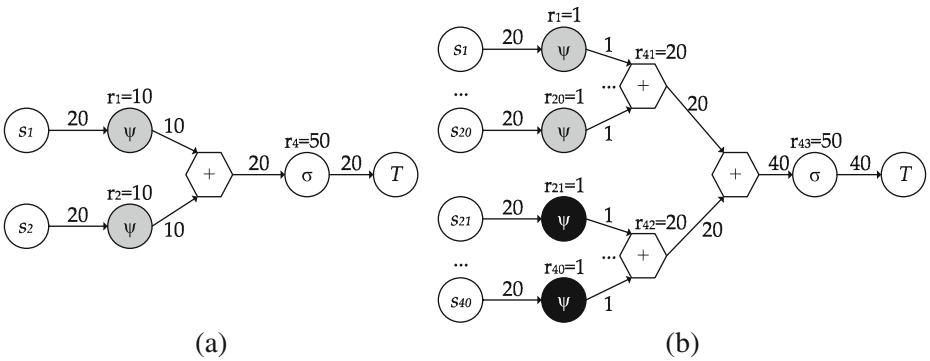


Fig. 8 Distributed processing consideration for m-CQ$_1$. (**a**) A weighted DAG (**b**) A derived DAG

two groups of cameras and each group has up to 20 cameras. Furthermore, the suggested m-CQ can support up to 50 cameras since the output rate of o_4 is 50 tps. It would become the bottleneck if no more processors were provided.

6 Extended dependency theory

In this section, we extend the multimedia dependency definitions from [6] to generalize to the multimedia streams case. As in [6], in order to evaluate the similarity between multimedia objects of two tuples, we need to use tuple-distance functions. The tuple-distance function summarizes the different distance functions applied to the elements of the 2 tuples under comparison. So, basically, a distance function is applied on corresponding attributes of the two tuples, then a function that takes these distances as input, will produce the final distance between the two tuples is called the tuple-distance function. What about tuples timestamps? Clearly, the notion of sliding windows on data streams can be utilized as follows for distance functions: if the two tuples belong to the same window then they might be considered for similarity, otherwise they may not. This way, we bound the calculations needed. And since the window specifications are set by the user, this way of utilizing the windows for calculating tuples distances does reflects user best interest. We are ready now to give some definitions.

Definition 13 *MS-similarity* Let ϖ be a tuple distance function on a relation R, and t be a maximum distance threshold, and x and y be two tuples in R, we then say that x is type MS-similar (Multimedia Stream Similar) to y with respect to ϖ, denoted $x \cong^{\varpi(t)} y$ iff x and y belongs to the same window, and $\varpi(x, y) \leq t$

Definition 14 (*type-MS functional dependency*) Let R be a relation with attribute set U, and X, Y⊆U. $X_{g1(t')} \rightarrow Y_{g2(t'')}$ is a *type-MS functional dependency* (MSFD) relation *iff* for any two tuples t1 and t2 in R, if $t_1[X] \cong_{g1(t')} t1[X]$, then $t1[Y] \cong_{g2(t'')} t2[Y]$, where t' and t'' are similarity thresholds, and $t_i[X]$ is the projection of the tuple t_i over the set of attributes X, and similarly is $t_i[Y]$.

In English, this definition typically says: there is a *type-MS functional dependency* (MSFD) from set of attributes X (under MS-Similarity g1(t')) and the set of attributes Y (under MS-Similarity g2(t'')) if and only if, any two tuples that are MS-similar under g1(t') this implies that these very two tuples are also MS-Similar under g2(t''). It reads that the set of attributes Y are *type-MS functionally dependent* on attributes X.

Using these definitions, the set of inference rules presented in [6] still holds for multimedia data streams case. And similarly the type-M Multi-valued dependency (MMD) can be generalized to the *type-MS Multi-Valued dependency* (MSMD) and the also the type-M join dependency (MJD) to the *type-MS Join dependency* (MSJD).These set of functional dependency can be utilized to normalize the schema at design time.

7 Algorithms for multimedia data streams continuous querying

In this section we describe a basic set of functions and algorithms to manipulate and handle multimedia data streams. First, we define the standard multimedia data stream. To do so, we focus our discussion below on Video and Image streams, while generalizing the algorithms to other multimedia data streams, such as audio and hyper text, is trivial.

The standard multimedia data stream, irrespective of the source, will have the following fields

1. Header- denoting start of frame
2. Timestamp
3. SourceID
4. A set of other attributes, and
5. EOF — denoting the end of frame

Note that field 3 above defines the source, as many sources (hardware) can map the tuples to the same multimedia data stream. Each source produces frames with a certain rate (frame per second) and different format. A video source will have a series of picture frames while a still camera source will have just one picture frame. Thus, the basic algorithm will be the same whether it is dealing with the picture frames or video frames. The second important attribute of the standard multimedia data stream shown above is the timestamp. Each frame will have a unique time stamp for that particular video source. There won't be a similar timestamp on two frames that belong to the same source but the timestamp can be same for two frames that belong to different sources. Note that this is different from the regular data streams. Thus, the unique tuple identifier here is the compound key: Timestamp and SourceID. This is important when we use the multimedia querying techniques like the transformational and fusion operators.

7.1 Algorithm to sort/group the incoming multimedia data

Below we describe the idea and high level steps of an algorithm to group the incoming multimedia data.

Step 1 identify the video source from the data stream. The data stream will have certain bits at fixed places which will help in identifying the video source. This is the first and the basic step required. Once the video source is identified, the corresponding algorithm will take over. We can have a separate algorithm for each video source.

Step 2 For video source: check the timestamp of the frame. As explained earlier, each frame will have a unique timestamp for the respective video source. We can group a series of frames together based on the particular time stamp. We can define a time stamp t and group all the frames from t to t+10. The next grouping will begin from t+11 which will be the new t and this group will again be till t+10. The grouping is essential in this case since the video frame will have a series of similar pictures and it will be easy to retrieve if they are divided into groups rather than an individual frame.

Step 3 For infra-red camera and other sources: check the time stamp of the frame. Since these sources will consists of only a picture and not a series of continuous pictures, we can group them accordingly based on their source and their time stamp. If we group them according to their time stamp, the retrieval can be easily managed. For example, if we need to check the pictures at a particular time, we can query it with the time limit T and T+t, and the corresponding query will fetch all the frames available in that particular time slot irrespective of the camera source. We do not need to find the source and then query it in that time slot. On the other hand, if we group them according to their source, we will have to sort them according to their time slot again. This will however, help us manage large chunks of data systematically. But again for retrieval, we need to know the video source first.

7.2 Algorithm for retrieval

The transformational and fusion operators can be used to retrieve the query response. The transformational operator is used for content-based retrieval from the incoming multimedia data streams. Again the algorithm depends on the multimedia stream source. The transformational operator requires two arguments:

1. The source of the multimedia data stream
2. Time slot for which the data is required, or in other words, the sliding window specifications.

These two parameters are essential in locating the frame which might contain the desired object. Further, we need to use the feature extraction algorithm not only in order to locate the object, but also its shape, size, color. Content based retrieval (CBR) approach is used to retrieve desired multimedia objects from a large collection on the basis of some features (such as color, texture and shape, etc.), specified within the query statement, that can be automatically extracted from the objects themselves. But major drawback of this method is that it lacks precision. That is why we early proposed the Multimedia Tool Boxes, which is, given the tradeoff between precision (QoD) and efficiency (QoS), we assume two sets of algorithms within the tool box, one that optimizes for QoD while the other optimizes for QoS. With a light weight QoS and QoD monitoring, with a user specified thresholds, the system can switch between the two multimedia tool boxes automatically.

7.3 Feature extraction algorithm

As the MMDSMS receives images from the various sources, they will be in digital forms. Every feature will be represented in a stream of 0 s and 1 s. The essential features needed to execute CBR queries are:

1. Shape of the object— can be used to identify the object
2. Color
3. Size of the object— which will be relative depending upon the location of the camera

The shape and the size of the object will also help identifying if the object in question is a real object or just a barrier. We can have a database of known objects and then compare the incoming data stream with them in order to 'identify' the object. Objects which are not in our database can be marked as 'UO or Unidentified object'. Initially, we will have a large number of objects marked as UO but gradually the system will be able to identify the object based on the user input. Care should be taken to filter out the noise from the incoming data stream. For example, new edge based feature extraction algorithm for video segmentation found at http://www.uco.es/~el1sapee/docs/research/IVCP03-1.pdf

7.4 Algorithm for the querying techniques

The retrieval will begin with the querying technique. The transformational and fusion operators can be sued for this purpose. A transformational operator is applied to any multi-dimensional source of objects in a specified set of intervals along a dimension. The generalized form of a transformational operator is for locating an object in a time period t1 to t2 from a source will be

$$\psi_{\text{type}}(\text{object name}) = (\\ \psi_{\text{type}}(\text{object name})\,\psi_{\text{xyz}}(*)\\ \sigma(\text{T})_{\text{T}>\text{t1 and T}<\text{t2}}\\ \psi_{\text{media\_sources}}\,(\text{source name})\ \text{media\_sources}$$

If we replace the source with some other media source like the video camera, laser radar, etc we can query that source in order to find the required object. The corresponding algorithm for the transformational operator will be

1. locate the media source in the query
2. check the time-slot
3. get the groups of images for that particular time slot from the media source
4. apply the feature extraction algorithm
5. compare the object with the existing objects in the database
6. check the query for additional parameters like color
7. compare it with the objects
8. output the result of the query

Sometimes we may need to compare the results of 2 or more transformational operators. In such a case, we will apply the algorithm to each of the operators independently and then compare the results. We may use the existing SQL operators for this as shown below:

1. UNION: This can be used to combine 2 transformational queries coming from different sources in order to add their results. The syntax can be of the form:

$$\Psi_{\text{motion}}(\text{moving})$$
$$\sigma(\text{T})(\text{time})$$
$$\Psi_{\text{media\_sources}}(\text{video})\text{media\_sources}$$
$$\text{UNION}$$
$$\Psi_{\text{type}}(\text{vehicle})$$
$$\sigma(\text{T})(\text{time})$$
$$\Psi_{\text{media\_sources}}(\text{laser\_radar})\text{media\_sources}$$

2. INTERSECT: Similar to UNION but will subtract the results of the 2 queries like

$$\Psi_{\text{motion}}(\text{moving})$$
$$\sigma(\text{T})(\text{time})$$
$$\Psi_{\text{media\_sources}}(\text{video})\text{media\_sources}$$
$$\text{INTERSECT}$$
$$\Psi_{\text{type}}(\text{vehicle})$$
$$\sigma(\text{T})(\text{time})$$
$$\Psi_{\text{media\_sources}}(\text{laser\_radar})\text{media\_sources}$$

   This query will find the type of the vehicle from the list of moving vehicles

3. EXCEPT: This query will find the ‘type’ of stationary vehicle, if any

$$\Psi_{\text{type}}(\text{vehicle})$$
$$\sigma(\text{T})(\text{time})$$
$$\Psi_{\text{media\_sources}}(\text{laser\_radar})\text{media\_sources}$$
$$\text{EXCEPT}$$
$$\Psi_{\text{motion}}(\text{moving})$$
$$\sigma(\text{T})(\text{time})$$
$$\Psi_{\text{media\_sources}}(\text{video})\text{media\_sources}$$

The fusion operator will perform sensor data fusion from heterogeneous data sources to generate fused objects. Fusion of data from a single sensor in different time periods is also allowed. It will combine the results of the transformational operators. The fusion parameter can have arguments and can be applied with respect to type, position and direction.

Once we have the data from various sources, we can use the existing fusion operator to combine the results of the two or more sources obtained through the transformational operators as

$$
\begin{aligned}
&\phi_{\text{type,position,direction}}\\
&\quad\left(\psi_{\text{motion}}(\text{moving})\,\psi_{\text{type}}(\text{Object})\right)\\
&\quad\sigma_{\text{xy}}(*)\\
&\quad\sigma(\text{T})_{\text{T mod 10=0 and T>t1 and T<t2}}\\
&\quad\psi_{\text{media\_sources}}(\text{source 1})\text{media\_sources}\\
&\quad.\\
&\quad.\\
&\quad.\\
&\quad\psi_{\text{type}}(\text{Object})\,\psi_{\text{xyz}}(*)\\
&\quad\sigma(\text{T})_{\text{T>t1 and T<t2}}\\
&\quad\psi_{\text{media\_sources}}(\text{source n})\text{media\_sources}
\end{aligned}
$$

The generalized algorithm combining the transformational and fusion operators will be

1. Locate the transformational operators in the query
2. Apply the transformational algorithm to each of the transformational operator independently
3. Check the parameters of the fusion operator
4. Apply the fusion operator with respect to the parameter specified to the results of the transformational operators
5. Output the result

8 Illustrative examples of continuous query processing

The examples mentioned below are an extension of the algorithms proposed. They can be used as a practical application of these algorithms. The logical SQL queries are used in a database and be used in the multimedia querying techniques.

Scenario: parking lot (Security system)
Cameras installed: Video, Laser radar
The basic steps in order to find an object or locate an object in a given area according to the algorithms defined, will be

- ✓ Query all the cameras situated in the area
- ✓ Apply the feature extraction algorithm to find the required object
- ✓ Query the result to find the desired object with the required parameters

- Aim: To find a Blue car(s) at a given time interval t1 and t2

Solution:

$$
\begin{aligned}
&\psi_{\text{type}}(\text{Car})\,\psi_{\text{xyz}}(*)\\
&\sigma(\text{T})_{\text{T>t1 and T<t2}}\\
&\psi_{\text{media\_sources}}(\text{Video})\ \text{media\_sources}\\
&\text{INTERSECT}\\
&\psi_{\text{color}}(\text{blue})\,\psi_{\text{xy}}(*)\\
&\sigma(\text{T})_{\text{T>t1 and T<t2}}\\
&\psi_{\text{media\_sources}}(\text{Video})\ \text{media\_sources}
\end{aligned}
$$

- Aim: To find a moving car(s) at a given time interval t1 and t2

  Solution:

$$\psi_{\text{motion}}(\text{moving})\psi_{\text{type}}(\text{car})$$
$$\sigma_{\text{xy}}(*)$$
$$\sigma(\text{T})_{\text{T mod 10=0 and T>t1 and T < t2}}$$
$$\psi_{\text{media\_sources}}(\text{Video})\text{media\_sources}$$

- Aim: are there any car(s) in the given location which are not similar to the blue cars for the time period t1 to t2

  Solution:

$$\psi_{\text{type}}(\text{car})\psi_{\text{xyz}}(*)$$
$$\sigma(\text{T})_{\text{T>t1 and T < t2}}$$
$$\psi_{\text{media\_sources}}(\text{Video})\text{media\_sources}$$
$$\text{NOT IN}$$
$$($$
$$\psi_{\text{type}}(\text{Car})\psi_{\text{xyz}}(*)$$
$$\sigma(\text{T})_{\text{T>t1 and T < t2}}$$
$$\psi_{\text{media\_sources}}(\text{Video})\text{media\_sources}$$
$$\text{INTERSECT}$$
$$\psi_{\text{color}}(\text{blue})$$
$$\psi_{\text{xy}}(*)$$
$$\sigma(\text{T})_{\text{T>t1 and T < t2}}$$
$$\psi_{\text{media\_sources}}(\text{Video})\text{media\_sources}$$
$$)$$

- Aim: is there any vehicle which is above the specified speed limit in the parking lot for a particular time period?

  Solution:

$$\psi_{\text{type}}(\text{car})\psi_{\text{xyz}}(*)$$
$$\sigma(\text{T})_{\text{T>t1 and T < t2}}$$
$$\psi_{\text{media\_sources}}(\text{Video})\,\text{media\_sources}$$
$$\text{where}$$
$$\psi_{\text{parameter}}(\text{speed}) > 15$$

- Aim: To check if there is any WANTED person in the parking lot

  Solution:

  First of all we will have to get all the objects present in the parking lot, then compare those objects with the existing database of WANTED persons.

$$\psi_{\text{type}}(\text{object})\psi_{\text{xyz}}(*)$$
$$\sigma(\text{T})_{\text{T>t1 and T < t2}}$$
$$\psi_{\text{media\_sources}}(\text{Video})\text{media\_sources}$$
$$\text{where}$$
$$\psi_{\text{type}}(\text{object}) = \psi_{\text{type}}(\text{object})\text{database}$$

- Aim: To check continuously if there is any vehicle going through the entrance at a given time interval t1 and t2 (30 times or more per second).

Suppose:
The output rate of each camera is 30fps.
The output rate of ψ operation is 20fps.
Solution:

$$\begin{aligned}&\Big(\psi_{\text{type}}(\text{car})\psi_{\text{xyz}}(*)\\&\psi_{\text{media\_sources}}(\text{Video\_1})\text{media\_sources}\\&+\\&\psi_{\text{type}}(\text{car})\psi_{\text{xyz}}(*)\\&\psi_{\text{media\_sources}}(\text{Video\_2})\text{media\_sources}\Big)\\&\sigma(\text{T})_{\text{T}>\text{t1 and T}<\text{t2}}\end{aligned}$$

Scenario: Medical Center (Healthcare System)
Cameras installed: X-ray, Video,
The basic steps in order to find an object or locate an object in a given area according to the algorithms defined will remain the same,

- ✓ Query all the cameras situated in the area
- ✓ Apply the feature extraction algorithm to find the required object
- ✓ Query the result to find the desired object with the required parameters

- Aim: To detect any presence of foreign body inside the patient's body

Solution:
We can take the images from the X-Ray and compare them with some standard images to detect any anomaly.

$$\begin{aligned}&\psi_{\text{type}}(\text{object})\psi_{\text{xyz}}(*)\\&\psi_{\text{media\_sources}}(\text{X}-\text{ray})\text{media\_sources}\\&\text{where}\\&\psi_{\text{type}}(\text{object})=\psi_{\text{type}}(\text{object})\text{database}\end{aligned}$$

- Aim: To monitor a person's behavior

Solution:
Compare the images from the video camera, from different time periods.

$$\begin{aligned}&\psi_{\text{type}}(\text{object})\psi_{\text{xyz}}(*)\\&\sigma(\text{T})_{\text{T}>\text{t1 and T}<\text{t2}}\\&\psi_{\text{media\_sources}}(\text{video})\,\text{media\_sources}\\&\text{where}\\&\psi_{\text{type}}(\text{object})=(\\&\psi_{\text{type}}(\text{object})\psi_{\text{xyz}}(*)\\&\sigma(\text{T})_{\text{T}>\text{t3 and T}<\text{t4}}\\&\psi_{\text{media\_sources}}(\text{video})\text{media\_sources})\end{aligned}$$

Scenario: Airport Security
Cameras installed: Video

The basic steps in order to find an object or locate an object in a given area according to the algorithms defined will remain the same,

- ✓ Query all the cameras situated in the area
- ✓ Apply the feature extraction algorithm to find the required object
- ✓ Query the result to find the desired object with the required parameters

- Aim: To detect the type of an aircraft

Solution:

We can take the images from the Video camera and compare them with some standard images to detect any anomaly.

$$\begin{aligned}&\psi_{\text{type}}(\text{aircraft})\psi_{\text{xyz}}(*)\\&\psi_{\text{media\_sources}}(\text{Video})\text{media\_sources}\\&\text{where}\\&\psi_{\text{type}}(\text{aircraft}) = \psi_{\text{type}}(\text{aircraft})\text{database}\end{aligned}$$

- Aim: To detect any unidentified flying object around the airport

Solution:

Compare the images from the video camera, with those in the database and ignore the images which are already in the database.

$$\begin{aligned}&\psi_{\text{type}}(\text{object})\psi_{\text{xyz}}(*)\\&\psi_{\text{media\_sources}}(\text{Video})\text{media\_sources}\\&\text{where}\\&\psi_{\text{type}}(\text{object})\ \text{NOT IN}\\&\Big(\psi_{\text{type}}(\text{aircraft})\psi_{\text{xyz}}(*)\\&\psi_{\text{type}}(\text{aircraft})\text{database}\Big)\end{aligned}$$

If we want to get a video stream with the output rate of at least 30fps, the query may need modification.

Suppose:

The output rate of each camera is 30 fps.

The output rate of ψ operation is 20 fps.

Solution:

$$\begin{aligned}&\Big(\psi_{\text{type}}(\text{object})\psi_{\text{xyz}}(*)\\&\psi_{\text{media\_sources}}(\text{Video\_1})\text{media\_sources}\\&+\\&\psi_{\text{type}}(\text{object})\psi_{\text{xyz}}(*)\\&\psi_{\text{media\_sources}}(\text{Video\_2})\text{media\_sources}\Big)\\&\text{where}\\&\psi_{\text{type}}(\text{object})\text{NOT IN}\\&\Big(\psi_{\text{type}}(\text{aircraft})\psi_{\text{xyz}}(*)\Big)\\&\psi_{\text{type}}(\text{aircraft})\text{database})\end{aligned}$$

- Aim: To find all the flying airplane(s) at a given time interval t1 and t2

  Solution:

$$\psi_{\text{motion}}(\text{moving})\psi_{\text{type}}(\text{airplane})$$
$$\sigma_{\text{xy}}(*)$$
$$\sigma(\text{T})_{\text{T mod 10=0 and T>t1 and T < t2}}$$
$$\psi_{\text{media\_sources}}(\text{Video})\text{media\_sources}$$

9 Discussion

In this paper we presented a computation-oriented model to model multimedia data streams as a potential tool for many useful applications such as security and health systems as described in the motivating examples earlier. Based on the proposed model, we provided performance estimates of the output rate, maximum number of supported streams, and the maximum number possible queries. We also generalized the multimedia dependency theory to include those involved in multimedia data streams and provided some algorithms of how to efficiently query multimedia data streams, along with illustrative examples. We believe this should provide a new research dimension in the database and the multimedia communities.

Compared to other models, our model is a computation-oriented model. This is the main difference. With the computation-oriented model we can obtain performance estimates to deal with QOS and QOD issues, which cannot be easily accomplished in some other models. Since the focus of this paper is the model for multimedia data stream, many important features of SigmalQL [5] such as the cluster operator are not discussed in the present paper. In fact, the query language SigmaQL is as expressive as CQL. All reasonable query languages should be equally expressive, just like all reasonable programming languages should be equally expressive. Therefore the issue is not whether one query language is more expressive than another. The issue is whether a query language can lend itself easily to performance estimates and therefore performance improvements.

Another consideration is further extensions. Our next goal is to further extend the multimedia dependency theory to include the ontological dimension. This would allow the ontological filtering to improve the performance of multimedia stream query processing. At issue is a suitable ontology that is expressive and yet amenable to transformations in order to be incorporated into efficient query processing algorithms.

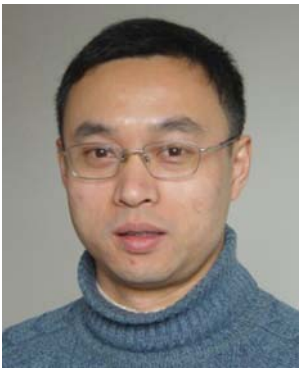
References

1. Abadi DJ, Carney D, Çetintemel U, Cherniack M, Convey C, Lee S, Stonebraker M, Tatbul N, Zdonik S (2003) Aurora: a new model and architecture for data stream management. The VLDB Journal 12(2):120–139
2. Abadi D J, Ahmad Y, Balazinska M, Çetintemel U, Cherniack M, Hwang J.-H, Lindner W, Maskey A, Rasin A, Ryvkina E, Tatbul N, Xing Y, Zdonik S B (2005) The design of the borealis stream processing engine. CIDR, 277–289
3. Arasu A, Babcock B, Babu S, Datar M, Ito K, Nishizawa I, Rosenstein J, Widom J (2003) Stream: The Stanford stream data manager (demonstration description). In SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data, New York, NY, USA
4. Arasu A, Babu S, Widom J (2006) The CQL continuous query language: semantic foundations and query execution. The VLDB Journal 15(2):121–142
5. Chang S K, Costagliola G, Jungert E Orciuoli F (2004)"Querying Distributed Multimedia Databases and Data Sources for Sensor Data Fusion", IEEE Trans. on Multimedia, Vol. 6, No. 5, 687–702
6. Chang S K, Deufemia V, Polese G (2007) A Normalization Framework for Multimedia Databases. IEEE Transactions on Knowledge and Data Engineering, 19(12)

7. Chang S K, Costagliola G, Jungert E, Camara K (2009) Intelligent Querying Techniques for Sensor Data Fusion. Chapter from the book Intelligent Techniques for Warehousing and Mining Data Streams, (Alfredo Cuzzocrea, ed.), IGI Global
8. Chen J, DeWitt D J, Tian F, Wang Y (2000) Niagaracq: a scalable continuous query system for internet databases. In *SIGMOD '00:* Proceedings of the 2000 ACMSIGMOD international conference on Management of data, pages 379–390. ACM
9. Gaber M, Zaslavsky A, Krishnaswamy S (2005) Mining data streams: a review. SIGMOD Record, 34(2)
10. Girguis S, Kulkarni R, Chang S K (2008)"A Multimedia Data Streams Model for Content-Based Information Retrieval", Proceedings of 2008 International Conference on Distributed Multimedia Systems (DMS2008), Boston, USA, 232–239
11. http://www.coral8.com/, 2004.
12. http://www.cs.cmu.edu/~biglou/research.html, 2008.
13. http://www.streambase.com, 2006.
14. Jain N, Mishra S, Srinivasan A, Gehrke J, Widom J, Balakrishnan H, Çetintemel U, Cherniack M, Tibbetts R, Zdonik S (2008) Towards a streaming SQL standard. Proc. VLDB Endow. 1(2):1379–1390
15. Sharaf M A, Chrysanthis P K, Labrinidis A, Pruhs K (2008) Algorithms and Metrics for Processing Multiple Heterogeneous Continuous Queries. ACM Transactions in Database Systems (TODS)
16. Sharaf MA, Chrysanthis PK, Labrinidis A, Pruhs K (2008) Algorithms and metrics for processing multiple heterogeneous continuous queries. ACM Trans. Database Syst. 33(1):1–44
17. Tatbul N, Zdonik S (2006). Window-aware load shedding for aggregation queries over data streams. In Proc. of VLDB Conference

Shi-Kuo Chang received the B.S. degree from National Taiwan University in 1965. He received the M.S. and Ph.D. degrees from the University of California, Berkeley, in 1967 and 1969, respectively. He was a research scientist at IBM Watson Research Center from 1969 to 1975. From 1975 to 1982 he was Associate Professor and then Professor at the Department of Information Engineering, University of Illinois at Chicago. From 1982 to 1986 he was Professor and Chairman of the Department of Electrical and Computer Engineering, Illinois Institute of Technology. From 1986 to 1991 he was Professor and Chairman of the Department of Computer Science, University of Pittsburgh. He is currently Professor and Director of Center for Parallel, Distributed and Intelligent Systems, University of Pittsburgh.

Dr. Chang is a Fellow of IEEE. He was a consultant to IBM, Bell Laboratories, Standard Oil, Honeywell, Naval Research Laboratory and Siemens. His research interests include distributed multimedia systems, visual information systems, visual languages and visual computing. Dr. Chang has published over two hundred and forty papers and wrote or edited twelve books.



Lei Zhao received the Ph.D. degree in the area of database access control in 2006 from Soochow University, Suzhou, China. From 1998 to now, he is a faculty of the school of computer science and technology of Soochow University. He is now an Associate Professor in the field of database technology and parallel computing. His research interests include database modeling, parallel model of data processing and data mining.

Shenoda Guirguis is a graduate student of the Department of Computer Science, University of Pittsburgh. His research interests are Sensor Networks, Multimedia Database management, Database Security.

Rohit P. Kulkarni was born in Mumbai, India in 1982. He received the B.Eng degree in Electronics engineering from the University of Mumbai, Mumbai, India in 2005. He worked for 2 years in India in various fields like Automation, Database, System administrations for companies like Reliance Industries, Citigroup, Bloomberg LP before coming to US to pursue MS degree from the School of Information Science, University of Pittsburgh, in the field of Database Management and Web-Systems. His research interests are Web-aware data management, Multimedia Database management, Database Security.