Context in Databases

Evaggelia Pitoura Kostas Stefanidis
Department of Computer Science
University of Ioannina
GR 45110 Ioannina, Greece
{pitoura, kstef}@cs.uoi.gr

Arkady K. Zavlavsky
School of Computer Science and Software Engineering
Faculty of Information Technology
Monash University
Melbourne, Australia
arkady.zaslavsky@cse.monash.edu.au

Abstract

A context-aware system is a system that uses context to provide relevant information or services to its users. While, there has been a variety of context middleware infrastructures and context-aware applications, little work has been done in integrating context into database management systems. In this paper, we first exploit the various parameters of context and provide a characterization of context quality. Then, we present a classification of context-aware applications and their characteristics. Finally, we propose various ways of integrating context into a database system.

1 Introduction

Context is any information than can be used to characterize the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between a user and an application, including the user and the applications themselves [17]. There are various types of context such as time, location, computing devices and users' profiles. A system is *context-aware*, if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task. Although, there has been a lot of work on developing variety of context infrastructures and context-aware applications, there has been only little work integrating context into databases. The goal of this paper is to exploit the various ways of making a database system context-aware.

Context is a general term. We present a classification of the most common types of context used in building software systems as well as an overview of

the basic characteristics of context information. We put special emphasis on context quality as an indication of the extend to which the values of context correspond to the real world.

We discuss context-awareness and the various ways context can be used to make an application context-aware. These include the presentation of information to the user, contextual information and automatic reconfiguration. Then, we exploit the various ways context can be used to make database management systems context-aware. These include context-aware query processing, consistency management and information integration.

We survey the various conceptual models that have been proposed for context as well as relevant models for storing context. Updating context is also a central issue and we present some direction of how context updates can be supported efficiently.

The remainder of this paper is structured as follows. Section 2 presents context and its parameters and proposes models for its quality. Section 3 introduces context-awareness, the characteristics pf context-aware systems and context infrastructures. Section 4 focuses on the management of context in Database Management Systems (DBMS), while Section 5 discuss context management issues, such as modeling and storage, in more detail. Section 7 concludes the paper.

2 Context

In this section, we introduce the general concept of context and the various types of context information. Then, we discuss the issue of context quality.

2.1 Context Parameters

Context is any information than can be used to characterize the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between a user and an application, including the user and the applications themselves [17].

There are various types of context that are related to data engineering tasks. Next, we discuss them using the taxonomy introduced in [9]:

- *Computing context.* Computing context includes (i) network connectivity, communication costs, and communication bandwidth, (ii) nearby resources such as printers, displays, and workstations and (iii) local resources, such as cpu, energy and type of display. Such parameters affect data engineering since to improve performance data engineering mechanisms (such as query processing algorithms and concurrency control protocols) must take into account the underlying resources.

- *User context.* User context includes the user's profile, location, people nearby, even the current social situation. This type of context parameters

directly affect the type of information relevant to a user. Thus, the user context affects the results of query processing.

- *Physical context.* Physical context refers to the environment surrounding the user such as lighting, noise levels, traffic conditions, and temperature. Such type of context indirectly affects the type of information that is relevant for or interesting to the user.

- *Time context.* Time context refers to the typical characterizations of time such as time of a day, week, month, and season of the year. Time may affect the result of a query, since relevance may also be dependent on the time.

There are dependencies among the various types of context parameters. For instance time context may affect the computing context, for example, network traffic at weekends is less than during weekdays.

Besides this notion of context, context is also used in information modeling as a higher-level conceptual entity that describes a group of conceptual entities from a particular standpoint.

2.2 Quality of Context

The characteristics of context parameters make handling them an intricate task. Such characteristics include the following [25].

- Context exhibits a range of temporal characteristics. Some types of context (such as the user profiles) are relatively *static* whereas other types of context (such as location) are *dynamic*. Furthermore, storing the context history is important for predicting future values of context and developing appropriate models for context evolution.

- Context information is imperfect. There are various reasons for that. One reason stems for the fact that context information is often dynamic, thus it gets quickly out-dated. Then, some forms of context information is often produced by crude sensor inputs which may result in faulty information. Furthermore, due to disconnections (e.g., wireless connections becoming unavailable) or failures (e.g., sensors running out of battery power), the available information may be imprecise. Finally, there is often the need for time-consuming transformations for producing usable values of context. Often the overhead of such transformations may be avoided or reduced on the cost of producing rougher estimations.

- There are many alternative presentations of context offering varying details and depth.

- Finally, context parameters are highly interrelated. There are complex dependencies among them, that are sometimes difficult to deduce. Such dependencies may lead to conflicting and sometimes inconsistent results.

The above characteristics of context necessitate the introduction of appropriate metrics for assessing the quality of context information. Context information often involves real world entities. Thus, it makes sense to measure the quality of context information (QoI), or the extent to which the data corresponds to the real world. The quality of context information can vary, perhaps substantially, depending on the context source and the type of context.

There are many issues regarding the quality of context information. First, one must identify what are the parameters that characterize quality. Then, how are these parameters measured, that is, what is an appropriate metric for each one of the quality parameters. Furthermore, an important issue is the estimation of quality provided and more importantly how is quality guaranteed. There has not been research addressing all the above issues. Most comprehensive research regarding handling the quality of context focuses on a particular type of context, that of *location*. Various models have been developed for predicting the location of moving objects. A survey on location management can be found in [37].

In [46], a model is proposed for predicting the location of objects moving on a trajectory. Instead of storing the exact current location of the object in a location database, information (such as the speed of movement and the trajectory) is stored. This information is used to estimate the current location. The motivation is dual: to avoid the cost of continuously updating the location in the database and to be able to answer queries about the future location of the object. In this case, a quality of metric is the *accuracy* of the estimated information, that is the difference between the actual and the estimated position. Similarly, [44] handles uncertainty regarding the position of moving objects but by introducing various spatiotemporal predicates (such as always definitely-inside, sometimes definitely-inside and possible always inside). Accuracy and confidence in the prediction are the quality measures used for the location attribute also in [7].

Regarding the type of context quality for general types of context, [22] propose six quality attributes:

- coverage: the amount of the potentially sensed context about which information is delivered
- resolution: smallest perceivable element
- accuracy: range in terms of a measure of the property
- repeatability: stability of measure over time
- frequency (sample rate): the temporal equivalent of resolution
- timeliness (range of the measure in time): the range of error in terms of the time of some phenomena; the temporal equivalent of accuracy used only for requirements analysis and the exploration of design issues

These attributes are used only for requirements analysis and the exploration of design issues. There is no exploitation about how such measures may be attained and used.

In [40], it is proposed to associate with each context value an uncertain measure that captures the likelihood that the value accurately reflects reality. This measure is pertinent to data produced by sensors. Similarly, [25] proposes tagging attributes with quality indicators. The type of quality indicator depends on the type of context.

Resolving ambiguous information from sensing or the interpretation of sensed information through a mediation process involving the user is proposed in [18]. Smooth correction of those errors must occur over some time frame and over some physical space. The quality issues in [19] are freshness and confidence. Since the quality of context information is reported by context sources, the issue of trust is raised and the need for independent monitoring is highlighted.

In general, we identify the following *Quality of Service (QoS)* parameters as relevant to context information and data management:

- accuracy: the deviation of the estimated value of a context parameter from the actual value of the parameter
- level of detail: this refers to the granularity of the presented information, for instance, in the case of time, this can be measured e.g., in years, hours or seconds
- conflict-free: this measure is application-dependent and refers to the requirement of having consistent and non-contradicting values
- timeliness: the deviation of the value of the context parameter in time (how up-to-date a value is kept)

3 Context-Awareness

A system is *context-aware* if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task.

3.1 Context-Aware Applications

Each context-aware application may support one the following features [17]:

- presentation of information and services to a user;
- automatic execution of a service for a user; and
- tagging of context to information to support later retrieval.

There are various types of context-aware applications depending on the way context is used. A nice classification is provided in [39] which identifies the following categories of context-aware applications:

- *Proximate selection* is a user-interface technique where the objects located nearby are emphasized or otherwise made easier to choose.

- *Automatic contextual reconfiguration* is the process of adding new components, removing existing components, or altering the connections between components due to context changes. Typical components and connections are servers and their communication channels to clients. However reconfigurable components may also include loadable device drivers, program modules and hardware elements.
- *Contextual information and commands* can produce different results according to the context in which they are issued.
- *Context-triggered actions* are simple IF-THEN rules used to specify how context-aware systems should adapt. There are similar to contextual information and commands, except that context-triggered action commands are invoked automatically

The situation abstraction is a description of the states of relevant entities.

3.2 Infrastructures for Context

Infrastructures for context are middleware systems that address issues common to all applications that want to take advantage of context. Such issues include capturing, accessing and storing context. Efficient distribution and support for independent execution from applications are desirable features for such architectures. Some popular context infrastructures include the following.

Context Toolkit The Context Toolkit [16] is a distributed architecture that supports It is based on three abstractions: context widgets, context interpreters and context aggregators. A context widget acquires a certain type of context information and makes this information available to applications. A context interpreter accepts one or more types of context and produces a single piece of context. A context aggregator aggregates or collects context. It is responsible for all the context for a single entity. Applications can subscribe to pre-defined aggregators and supply appropriate filters.

Solar The Solar system [11, 10] advocates a graph-based abstraction for context aggregation and dissemination. Context information is modeled through events which are produced by sources. Events flow through a directed acyclic graph of events-processing operators and are delivered to subscribing applications. Applications subscribe by describing their desired event stream as a tree of operators that aggregate low-level context information published by existing sources into the high-level context information needed by the application.

Cooltown The Cooltown project [28] proposed a web-based model of context in which each entity (person, place or thing) has a corresponding description that can be retrieved via a URL. Using URLs for addressing, physical URL beaconing and sensing of URLs for discovery, and localized web servers for directories,

they create a location-aware but ubiquitous system to support nomadic users. On top of this infrastructure, the Internet connectivity is used to support communications services.

CoolAgent CoolAgent [12] is a context-aware multi-agent system. Ontology sharing, sensing and reasoning is supported through the use of the Resource Description Framework (RDF) and a Prolog-based system.

4 Context-aware DBMS

In this section, we consider how context can be integrated into a Database Management System. Context information related to a DBMS includes user-related information (such as information provided through a user profile), computational-related (such supporting small device, limited energy, quality of network connection (e.g., in terms of reliability, frequent disconnections, intermittent connectivity an low bandwidth) and environmental conditions (weather, location, time of the day).

4.1 Context-aware Query Processing

Context-aware query processing has many aspects. We consider how context affects (i) the results returned by a query, (ii) query optimization and (iii) the way the results are presented to the users.

Although, there is some research on location-aware query processing, integrating other forms of context in query processing is a new issue. The only related work that we are aware of is the context-aware querying processing framework of [20]. In this framework, context-aware query processing is divided into three-phases: query pre-processing, query execution and query post-processing. Query pre-processing is performed in two steps: a query refinement and a context binding step. The goal of the query refinement step is to further constraint the query condition by means of different contextual information. Context binding instantiates with exact values the contextual attributes involved in the refined query. After query execution, at the query post-processing phase, the results are sorted. External services may then be invoked for the delivering of results to the users. Five context-aware strategies are defined. Strategy 1 refers to queries that consider the current value of context as their reference point, for example such queries include looking for the closest restaurant, the next flight, the shortest route. To implement them, the contextual attributes are bound to their current values. Strategy 2 includes queries that access facts about the past (i.e., history data) which are recalled based on the relevant context. In this case, archived data are linked based on their common contextual attributes. Strategy 3 considers context as an additional constraint to the query. A given query is refined to include relevant constraint rules. Strategy 4 reduces the result set by ordering the produced results based on the user profile. This is

achieved by using an associated sorting rule. Strategy 5 considers the delivery and presentation of results to the user by observing related delivery rules.

In the following, we shall use as a simple running example a database of information about restaurants. The type of the record entries for restaurants are tuples of a relation schema Restaurant(id, type-of-food, address, outdoors, opening-hours, price). The context parameters are weather, location, time and the user profile.

4.1.1 Context-Aware Results

Context may affect the results produced by a query. In this case, the same query may produce different results depending on the context in which it is executed. Context-aware query processing may be seen as a two-step process. In the first phase, the context relevant to a specific query is initially identified and then acquired. During the second phase, the relevant context is integrated within the query.

Querying Context Parameters. One way to involve context within queries is by allowing explicit access to context parameters within a query. Context parameters are treated as attributes of a virtual relation; let us call this relation Context. The attributes of Context are bounded to the current value of context when the query is executed. In the following example, we assume that the attribute time of Context is bound to the current time when the query is executed. The query returns all restaurants that are currently open.

select Restaurant.id
from restaurants, Context
where Context.time *in* Restaurant.opening-hours

Context as a Predicate. Another way to achieve context-awareness is to augment the query with appropriate predicates. In particular, a given query is transformed to a different one by adding additional constraints to it. One way this may be achieved is by adding constraints using the contextual attributes. Another way is by associating rules with specific attributes or relations and adding these rules to the query. For example assume that a user specifies that when the weather is good, the user likes to eat outdoors. The following example returns all restaurant with an outdoor facility when the weather is good and any restaurant otherwise.

An initial query submitted by the user:

select Restaurant.id
from Restaurants

is transformed to

select Restaurant.id
from Restaurants, Context

where (Context.weather = "sunny" and Restaurant.outdoors = "available") or Context.weather = "rainy"

Context as Preference. Context can be used to confine database querying by selecting as results the best matching tuples. This can be achieved by defining preferences based on context, so that under a specific context a tuple is preferred over another. The research literature on preferences is extensive. In particular, in the context of database queries, there are two different approaches for expressing preferences: a quantitative and a qualitative one.

With the *quantitative approach*, preferences are expressed indirectly by using scoring functions that associate a numeric score with every tuple of the query answer. Such a general quantitative framework for expressing and combining preferences is proposed in [1]. In this framework, a preference is expressed by the user for an entity. Entities are described by record types which are sets of named fields, where each field can take values from a certain type. The * symbol is used to match any elements of that type. Preferences are expressed as functions that map entities of a given record type to a numerical score. A set of preferences can be combined using a generic combine operator which is instantiated with a value function. For example, the preference of a user for restaurants can be expressed as preference(type-of-food), with values preference(chinese) = 0.1, preference(greek) = 0.8 and preference(other) = 0.1.

In the quantitative framework of [30], user preferences are stored as degrees of interests in *atomic query elements* (such as individual selection or join conditions). The degree of interest expresses the interest of a person to include the associated condition into the qualification of a given query. Specific rules are specified for deriving preference of complex queries by building on stored atomic ones. The results of a query are ranked based on the estimated degree of interest in the combination of preferences they satisfy.

Both quantitative frameworks can be readily extended to include context. One way this can be achieved is by defining preference functions based on context. Then, based on the current values of context, the associated preference functions can be selected, combined and used to rank the results of any given query. Similarly, we may either include contextual parameters in the atomic query elements or make the degree of interest for each atomic query element to depend on context.

In the *qualitative approach*, the preferences between the tuples in the answer to a query are specified directly, typically using binary preference relations. For example, one may express that restaurant1 is preferred from restaurant2 if their opening hours are the same and its price is lower. This framework can also be readily extended to include context. For instance, one may express that restaurant1 is preferred from restaurant2 if their opening hours are the same, its price is lower and it is closest to the current user's location.

A logical qualitative framework is presented in [14] for formulating preferences as *preference formulas*. The preference formula is a first-order formula defining a preference relation between two tuples.

Both the quantitative and the qualitative approached can be integrated with query processing. Relevant in this respect is research on top-k matching results and on skylines. In *top-k* queries [8], users specify target values for certain attributes, without requiring exact matches to these values in return. Instead, the result to such queries is typically a rank of the "top-k" tuples that best match the given attribute values. The *skyline* [5] is defined as those tuples of a relation that are not dominated by any other tuple. A tuple dominates another tuple if it is as good or better in all dimensions and better in at least one dimension.

Context for Associative Recall. Finally, context can be used for associative recall of past events. For example, in a "memory" database, context (such as time or location) can be used to retrieve the associated facts. For instance, it may be easier to retrieve facts (such as who was the prime minister of Greece) or objects (such as for example photographs or favorite music albums) by referring to the particular time period of one's life (e.g., when user "John" was dating "Mary") or a geographic location (e.g., during once vacation in Hawaii) closely associated with the facts or objects. To achieve such retrieval, storage of facts or objects in a database must include information about the context parameters when they occurred.

4.1.2 Context-aware Query Optimization

Besides affecting the results of a query, context information may be used in query optimization to achieve more cost-effective plans. Computing context is very relevant in this case. Instead of optimizing disk access, query plans may be derived to optimize other performance metrics such as energy (when energy power is an issue). Furthermore, the user context can also be exploited. For instance, query processing may be such that the most relevant results (based on the user's profile) are returned first.

4.1.3 Context-aware Query Presentation

The way the results are presented to a user directly depends on the device currently used by the user. In addition, energy and networking considerations may affect the way query results are delivered to the user.

4.2 Context-Aware Consistency

Besides query processing, context can be used in other data engineering tasks. These tasks include context-aware caching and replication. Context has been used in a very limited way to direct the management of caches. Only two parameters of context have been considered: (i) location of the user and (ii) user preferences expressed in a user profile.

Location and Caching. Exploiting location information in caching has been

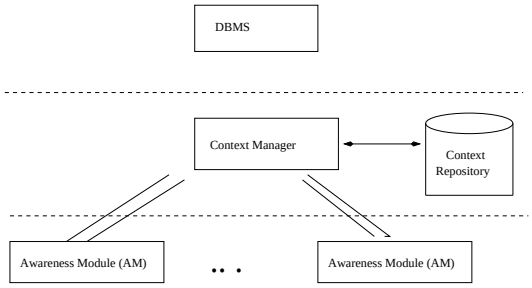

Figure 1: Connecting Context and Databases

taking into account in the case of location-dependent services. A service is location dependent, if its result depends on the location of the user who requested the service. Examples include nearest neighbor searching (e.g., finding the nearest restaurant) and local information access (e.g., local traffic news or area attractions).

In *semantic caching*, the semantic description of data and the previous queries are kept in the cache along with data. In [38], the idea of semantic caching is applied for the case of location dependent queries. In [48, 47], caching is considered for location-dependent services. The cache refreshment policies proposed take into account the valid scope of a data value that is defined as the geographical area within which the data value is valid.

User Context and Caching. User profiles have been used both for determining the cache content [13, 31] and for selecting an appropriate cache refreshment policy (that is a policy that specifies when and how to update the cache). User profiles are used to manage the contents of caches. Profiles are expressed in a profile language that permits a high-level expression of user's data needs for the purpose of expressing the desirable contents of a cache. Advanced techniques are presented for prefetching a cache on the basis of profiles, both for basic and preemptive prefetching, the latter referring to the case where staging a cache can be interrupted at any point without prior warning.

Early work in relating user preferences and cache refreshment is quasi caching [2]. In quasi-caching, the specification of user requirements in terms of data quality are used to reduce the amount of data sent from servers to clients to update client caches.

More recent work in this context includes [3] and [6]. In [3], a cache maintenance technique is presented that focus on cache consistency guarantees. The technique is based on latency-recency profiles that allow clients to express target values for the desired latency and recency of objects. A scoring function uses the target values to determine when to download an object from the server or when to use a cached copy. The function can be tuned to meet the target values and can provide guarantees with respect to the maximum latency or recency of requested objects. User profiles are used in [6] to achieve more efficient cache

refreshment policies.

Context and Replication. There is not much current research on context-aware replication schemes. Most previous research takes into account the network connectivity between the various replication sites.

Consistency models in terms of distributed file systems (DFSs) are considered in [15]. Most DFSs implement a single consistency model to maintain one-copy equivalence. The functionality of the proposed consistency model, GLOMAR, is based on a balance between environmental constraints and the targeted level of consistency. GLOMAR is a DFS middleware layer that allows application developers to map their specific consistency models to environmental constraints. As a result, multiple consistency models can be created, with each scoped for a particular application and environmental scenario.

An adaptive replication schema based on the quality of network connection among the replication sites was proposed in [35, 36]. The proposed schema, termed weak consistency, is based on two new high level update and read operations (weak and strong). Weak reads return any available copy (even an out-of-date one). Weak updates are only tentatively committed. Weak operations are used in the case of disconnections.

In [21], a two-tier schema is proposed that allows disconnected applications to propose tentative update transactions that are later applied to a master copy.

4.3 Context in Information Systems and Multidatabases

Besides the use of context in context-aware systems, context has been used in the area of multidatabase systems to resolve semantic differences. An important issue in multidatabase systems is interoperability that is identifying objects in different databases that are semantically related and then resolving differences among these objects. Here the notion of context is that of information context. Context refers to the implicit assumptions underlying the manner in which data are represented and interpreted at each database.

In this respect, a context as *information context* may be identified or represented using the following [26].

- by association with a database or a group of databases
- as the relationship in which an entity participates
- from a schema architecture, a context can be specified in terms of an export schema (a context that is closer to the database) or an external schema (a context that is closer to the application)
- at a very elementary level, as a named collection of domains of objects

The degree of semantic similarity between a pair of objects is characterized in [27] by using the concept of semantic proximity. It is based on the premise that it is essential to associate the mappings between the objects to be compared with

the context of comparison. Context is represented as a collection of contextual coordinates and their values. The meaning of the contextual coordinates and their values are informally explained by expressing the context using description logic expressions.

In a similar spirit, context is defined in [34] as the knowledge that is needed to reason about another system, for the purpose of answering a query. It is specified as a set of assertions identifying the correspondences between various schema elements.

In [41], context is defined as the meaning, content, organization and properties of data. It is modeled using meta-data associated with the data. When using a well-defined ontology, such as Cyc [23], a well-defined partition (called Microtheory) of the ontology is assigned a context.

Context in [33] is used as a general mechanism for partitioning information bases. Analogous is the use of context in [43] where context is used to disambiguate the meaning of names. Context is considered as a pair (cid, l) where cid is the context identifier and l is a lexicon (a binding of names to objects). Nesting of contexts is possible by allowing a context to belong to the objects of the lexicon of one or more other context. Operations are defined for manipulating context.

4.4 Architecture of Context-Aware DBMS

A overview of how context can be integrated within a Database Management System (DBMS) is depicted in Figure 1. The *Awareness Modules* communicate with the sources that produce data (for instance, temperature sensors) and propagate any updates to the Context Manager. The *Context Manager* is responsible for managing (modeling, storing, updating) any context related information. The Context Repository is the module where context is stored. There are two ways of integrating context in a DBMS: (a) the context manager may be part of the DBMS or (b) the context manager may be seen as an intermediate middleware layer.

5 Context Management

There are many types of context information thus providing a unifying model for modeling and a general approach for storing context variables are challenging issues. We provide next a survey of various approaches to both problems.

5.1 Model of Context

A variety of models have been introduced for context. Discussions of the different models can be found in [9, 42]. Such models fall in one of the categories described next.

Models for Location. Location is a context parameter that has attracted

a lot of attention. Models for location are different than other values of context mainly because the location of moving objects is a parameter whose value changes continuously with time. There are basically two different way to represent location: a symbolic and a geometric model. With the symbolic model, location is represented using abstract symbols, while with the geometric model, location is represented using coordinates. A nice overview of current research on the topic can be found in [29].

Key-Value Models. The simplest model is to represent contextual information in the form of (context-variable, value) pairs. Key-value pairs can be used for efficient exact match queries for example for automatic contextual reconfiguration. Such models are general and easy to manage but lack in expressibility of semantic information.

Markup Schemes. Context is modeled using "contextual" tags. Common to such schemes is a hierarchical data structure that is express through the nesting of tags.

An example of such representation is a CC/PP profile [45]. A CC/PP profile is a description of device capabilities and user preferences that can be used to guide the adaptation of content presented to that device. CC/PP is based on RDF, the Resource Description Framework, which was designed by the W3C as a general purpose metadata description language. The Resource Description Framework (RDF) is used to create profiles that describe user agent capabilities and preferences. A CC/PP profile contains a number of CC/PP attribute names and associated values that are used by a server to determine the most appropriate form of a resource to deliver to a client. It is structured to allow a client to describe its capabilities by reference to a standard profile, accessible to an origin server or other sender of resource data, and a smaller set of features that are in addition to or different than the standard profile. A set of CC/PP attribute names, permissible values and associated meanings constitute a CC/PP vocabulary.

An example of using a markup scheme is "stick-e" notes [4] which are the electronic equivalents of post-it notes. Context information is modeled as tags and corresponding fields. The stick-e fields can recursively contain other tags and corresponding fields. The note of the <body> tag is automatically triggered when the contextual constraints in the <require> tag are met. This model has evolved into the ConteXtML model which is an XML-based protocol for exchanging contextual information.

Graphical Models. A variety of general models (such as the E/R model and UML) are graphical. Such models are very expressive and are mainly used as conceptual models. However, they convey little information at the instance level or on implementation issues.

Object-Oriented Models. Important features of object-oriented models are

encapsulation and re-usability.

Logic-Based Models. A large number of proposals to represent context are based on logic. Important in this respect is the formalization proposed in [32]. Contexts are considered as first class objects. The basic relation is ist(c,p). It asserts that the proposition p is true in the context c. The most important formulas relate the propositions true in different contexts. Introducing contexts as formal objects permits axiomatizations in limited contexts to be expanded to transcend the original limitations. This seems necessary to provide AI programs using logic with certain capabilities that human fact representation and human reasoning possess. Fully implementing transcendence seems to require further extensions to mathematical logic, i.e. beyond the nonmonotonic inference methods.

Ontology-Based Models. Ontologies have currently attracted much attention for specifying concepts and interrelations.

5.2 Storing Context

An important issue is what is an appropriate model for storing context. Besides storing the current context for building context-aware systems and applications, there is growing effort to extract interesting knowledge (such rules, regularities, constraints, patterns) from large collections of context data.

Storing context data using data cubes, called context cubes [24], is proposed for developing context-aware applications that use archive sensor data. The context cube provides a multidimensional model of context data where each dimension presents a context dimension of interest. The context cube also provides a number of tools for accessing, interpreting and aggregating context data by using concept relationships defined within the real context of the application. The basic cube operations are slice, dice, roll-up and drill-down. A slice is a selection of one dimension of an n-dimensional cube. The dice is a selection applied to all dimensions of the cube. Roll-up generates a new cube by applying an aggregate function on one dimension. Drill-down is the inverse of roll-up; it generates a context cube with finer granularity on one of the n dimensions. A cube can be used to create new context from analysis of the existing data.

5.3 Updating Context

Since context parameters change with time, deriving a model of how they change is important. This enables the prediction of future values of context. Furthermore, such information can be used to fine-tune various system-related parameters as well as a variety of protocols. Finally, having a model for context updates allow building systems that are more cost-effective. There are in general two ways for communicating updates: a push and a pull model. In the *push model*, the source of the context update push the new value of the associated context

parameter to the context-aware system. In the *pull model*, the context-aware system polls the source to learn about any updates. In both models, there is cost associated with update propagation. A model of context would reduce such cost, since it will eliminate the cost of communication between the source and the context-aware system. It will also reduce the computation cost at both ends.

Deriving general models for context updates is a formidable task, because of the great variety of context information. Models have been advanced for location updates, since it is possible to predict future locations when the moving objects follow some pattern of movement or are moving in trajectories (for example, cars in highways).

Another important issue relevant to data engineering is how to communicate the change of context to the result of querying processing. We distinguish the following three different approaches regarding how to update the query so that it takes into account context context:

- each query answer includes the valid time of the answer
- the previous results is cached and used them to prune the search for the new results
- the result is precomputed by using some model to predict the values of the context parameters.

6 Summary

In this paper, we exploit the various parameters of context and provide a characterization of context quality. Then, we present a classification of context-aware applications and their characteristics. Finally, we propose various ways of integrating context into a database system.

References

[1] R. Agrawal and E. L. Wimmers. A Framework for Expressing and Combining Preferences. In *Proc. of SIGMOD*, 2000.

[2] R. Alonso, D. Barbara, and H. Garcia-Molina. Data Caching Issues in an Information Retrieval System. *ACM TODS*, 15(3), 1990.

[3] L. Bright and L. Raschid. Using Latency-Recency Profiles for Data Delivery on the Web. In *Proc. of VLDB*, 2002.

[4] P. J. Brown, J. D. Bovey, and X. Chen. Context-aware Applications: from the Laboratory to the Marketplace. *IEEE Personal Communications*, 4(5), 1998.

[5] S. Brzsnyi, D. Kossmann, and K. Stocker. The Skyline Operator. In *Proc. of ICDE*, 2001.

[6] D. Carney, S. Lee, and S. Zdonik. Scalable Application Aware-Data Freshening. In *Proc. of ICDE*, 2003.

[7] P. Castro, P. Chiu, T. Kremenek, and R. R. Muntz. A Probabilistic Room Location Service for Wireless Networked Environments. In *Proc. of UbiComp*, 2001.

[8] S. Chaudhuri and L. Gravano. Evaluating Top-k Selection Queries. In *Proc. of VLDB*, 1999.

[9] G. Chen and D. Kotz. A Survey of Context-Aware Mobile Computing Research. Dartmouth Computer Science Technical Report TR2000-381, 2000.

[10] G. Chen and D. Kotz. Context Aggregation and Dissemination in Ubiquitous Computing Systems. In *Proc. of the 4th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA02)*, 2002.

[11] G. Chen and D. Kotz. Solar: An Open Platform for Context-Aware Mobile Applications. In *Proc. of the 1st International Conference on Pervasive Computing*, 2002.

[12] H. Chen, S. Tolia, C. Sayers, T. Finin, and A. Joshi. Creating Context-Aware Software Agents. In *Proc. of the First GSFC/JPL Workshop on Radical Agent Concepts*, 2002.

[13] M. Cherniack and M. J. Franklin. Expressing User Profiles for Data Recharging. *IEEE Personal Communications*, 2001.

[14] J. Chomicki. Preference Formulas in Relational Queries. *TODS*, 28(4), Dec 2003.

[15] S. Cuce and A. Zaslavsky. Supporting Multiple Consistency Models for a Mobility Enabled File System using a Component Based Framework. *MONET*, 8(4), 2003.

[16] A. K. Dey. Providing Architectural Support for Building Context-Aware Applications. PhD Thesis, College of Computing, Georgia Institute of Technology, December 2000, 2000.

[17] A. K. Dey. Understanding and Using Context. *Personal and Ubiquitous Computing*, 5(1), 2001.

[18] A. K. Dey, J. Mankoff, and G. D. Abowd. Distributed Mediation of Imperfectly Sensed Context in Aware Environments. GVU Technical report GIT-GVU-00-14, 2000.

[19] M. Ebling, G. Hunt, and H. Lei. Issues for Context Services for Pervasive Computing. In *Proc. of Middleware'01 Advanced Workshop on Middleware for Mobile Computing*, 2001.

[20] L. Feng, P.M.G. Apers, and W. Jonker. Towards Context-Aware Data Management for Ambient Intelligence. In *Proc. of the 15th Intl. Conf. on Database and Expert Systems Applications (DEXA)*, 2004.

[21] J. Gray, P. Helland, P. E. O'Neil, and D. Shasha. The Dangers of Replication and a Solution. In *Proc. of SIGMOD*, 1996.

[22] P. D. Gray and D. Salber. Modelling and Using Sensed Context Information in the Design of Interactive Applications. In *Proc. of the 8th IFIP International Conference on Engineering for Human-Computer Interaction*, 2001.

[23] R. V. Guha. Micro-theories and Contexts in Cyc. Basic Issues. Technical Report ACT-CYC-129-90 Microelectronics and Computer Technology Corporation, Austin, 1990.

[24] L. Harvel, L. Liu, G. D. Abowd, Y-X. Lim, C. Scheibe, and C. Chathamr. Flexible and Effective Manipulation of Sensed Contex. In *Proc. of the 2nd Intl. Conf. on Pervasive Computing*, 2004.

[25] K. Henricksen, J. Indulska, and A. Rakotonirainy. Modeling Context Information in Pervasive Computing Systems. In *Proc. of the 1st International Conference on Pervasive*, pages 167–180, 2002.

[26] V. Kashyap and A. Sheth. So Far (Schematically) yet So Near (Semantically). In *Proc. of IFIP WG 2.6 Database Semantics Conference on Interoperable Database Systems (DS-5)*, 1992.

[27] V. Kashyap and A. Sheth. Semantic and Schematic Similarities between Database Objects: a Context-based Approach. *VLDB Journal*, 5(4), 1996.

[28] T. Kindberg, J. Barton, and J. Morgan et al. People, Places, Things: Web Presence for the Real World. *MONET*, 7(5), 2002.

[29] M. Koubarakis, T. K. Sellis, and A. U. Frank et. al. *Spatio-Temporal Databases: The CHOROCHRONOS Approach.* Lecture Notes in Computer Science 2520 Springer, 2003.

[30] G. Koutrika and Y. Ioannidis. Personalization of Queries in Database Systems. In *Proc. of ICDE*, 2004.

[31] E. F. Galvez M. Cherniack, M. J. Franklin, and S. Zdonik. Profile-Driven Cache Management. In *Proc. of ICDE*, 2003.

[32] J. McCarthy. Notes in Formalizing Context. In *Proc. of the 13th International Joint Conference in Artificial Intelligence*, 1993.

[33] J. Mylopoulos and R. Motschnig-Pitrik. Partitioning Information Bases with Contexts. In *Proc. of CoopIS*, 1995.

[34] A. Ouksel and C. Naiman. Coordimnating Context Building in Heterogeneous Information Systems. *J. Intell Inf Systems*, 3, 1993.

[35] E. Pitoura and B. Bhargava. Maintaining Consistency of Data in Mobile Distributed Environments. In *Proc. of ICDCS*, 1995.

[36] E. Pitoura and B. Bhargava. Data Consistency in Intermittently Connected Distributed Systems. *IEEE TKDE*, 11(6), 1999.

[37] E. Pitoura and G. Samaras. Locating Objects in Mobile Computing. *IEEE TKDE*, 13(4), 2001.

[38] Q. Ren and M. H. Dunham. Using Semantic Caching to Manage Location Dependent Data in Mobile Computing. In *Proc. of MOBICOM*, 2000.

[39] B. N. Schilit, N. I. Adams, and R. Want. Context-Aware Computing Applications. In *Proc. of the Workshop on Mobile Computing Systems and Application*, 1994.

[40] A. Schmidt, K. A. Aidoo, and A. Takaluoma et al. Advanced Interaction in Context. In *Proc. of the International Symposium on Handheld and Ubiquitous Computing (HUC99)*, 1999.

[41] E. Sciore, M. Siegel, and A. Rosenthal. Using Semantic Values to Facilitate Interoperability Among Heterogeneous Information Systems. *ACM TODS*, 1994.

[42] T. Strang and C. Linnhoff-Popien. A Context Modeling Survey. In *Proc. of the Workshop on Advanced Context Modelling, Reasoning and Management associated with the Sixth International Conference o n Ubiquitous Computing (UbiComp 2004)*, 2004.

[43] M. Theodorakis, A. Analyti, P. Constantopoulos, and N. Spyratos. A Theory of Contexts in Information Bases. *Information Systems*, 27(3), 2002.

[44] G. Trajcevski, O. Wolfson, K. Hinrichs, and S. Chamberlain. Managing Uncertainty in Moving Objects Databases. *ACM TODS*, Sept. 2004.

[45] W3C. Composite Capabilities/Preferencs Profile (CC/PP). http://www.w3.org/TR/2004/REC-CCPP-struct-vocab-20040115/, 2004.

[46] O. Wolfson, P. Sistla, S. Chamberlain, and Y. Yesha. Updating and Querying Databases that Track Mobile Units. *Distributed and Parallel Databases Journal*, 7(3), 1999.

[47] J. Xu, X. Tang, and D. L. Lee. Performance Analysis of Location-dependent Cache Invalidation Schemes for Mobile Environments. *IEEE TKDE*, 15(2), 2003.

[48] B. Zheng, J. Xu, and D. L. Lee. Cache Invalidation and Replacement Strategies for Location-Dependent Data in Mobile Environments. *IEEE Transactions on Computers*, 51(10), 2002.