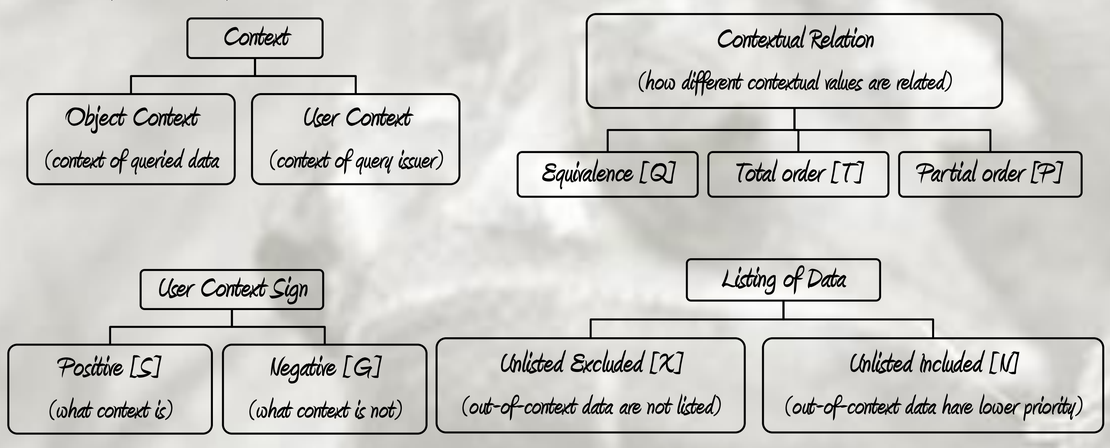
Chameleon: Context-Awareness inside DBMSs

Hicham G. Elmongui and Walid G. Aref (Purdue University)
Mohamed F. Mokbel (University of Minnesota)

Introduction

Context is any information that can be used to characterize the situation of an entity. Examples of contexts include, but are not limited to, time, location, identity, and activity of a user. This paper proposes a general context-aware DBMS, named Chameleon, that will eliminate the need for having specialized database engines, e.g., spatial DBMS, temporal DBMS, and Hippocratic DBMS, since space, time, and identity can be treated as contexts in the general context-aware DBMS. Moreover, in Chameleon, we will be able to combine multiple contexts into more complex ones using the proposed context composition, e.g., a Hippocratic DBMS that also provides spatio-temporal and location contextual services. As a proof of concept, we construct two case studies using the same context-aware DBMS platform within Chameleon. One case study treats identity as a context to realize a privacy-aware (Hippocratic) database server while the other case study treats space as context to realize a spatial database server using the same proposed constructs and interfaces of Chameleon.

Classification of Contexts



Context-Awareness SQL Constructs

```
CREATE OBJECT CONTEXT context_name (
     {col_spec | table_constraint} [, . . . ]
     , table_binding
);

table_binding: BINDING KEY ([col_name [, . . . ]])
               REFERENCES ref_table [( ref_col [, . . . ] )] WITH bool_expr

CREATE [context_sign] CONTEXT context_name (
     {col_spec | table_constraint} [, . . . ]
     , table_binding [, . . . ]
     [, substituting_key [, . . . ]]
) [AS contextual_relation clause]
[WITH UNLISTED unlisted_status];

context_sign: POSITIVE | NEGATIVE
contextual_relation: EQUIVALENCE
                   | TOTAL ORDER [USING ordering_func]
                   | PARTIAL ORDER
unlisted_status: EXCLUDED
               | INCLUDED

substituting_key: SUBSTITUTE table name(col name) BY expression;

SET ACTIVE CONTEXT [FOR USER user_name] AS context_name [, . . . ]
     { [WITH RANKING ORDER context_name [, . . . ] ]
       | [WITH RANKING EXPRESSION expression
       | [WITH SKYLINE OF expression {MAX|MIN} [, . . . ] ]
     };
```

Conceptual Evaluation

We use a simplistic preference-based system to demonstrate Chameleon's proposed syntax and semantics. Consider a table "books" that contains information about books in a certain bookstore. This table has the schema books(id, title, year, category, cover, in_stock). Each user specifies her preference as her active context. Upon submitting "**SELECT * FROM books;**", The user gets the relevant books only.

Context 1: The user has a preference for only books of a certain category (e.g., Science fiction).

```
CREATE POSITIVE CONTEXT ctxt_category_SQX (
     category varchar(20),
     BINDING KEY (category) REFERENCES books(category)
) AS EQUIVALENCE WITH UNLISTED EXCLUDED;

SET ACTIVE CONTEXT AS ctxt_category_SQX;
```

Context 2: The user's preference is for books published in 2005, and then those published in 2006 before all other books.

```
CREATE POSITIVE CONTEXT ctxt_year_STI (
     year integer,
     BINDING KEY (year) REFERENCES books(year)
) AS TOTAL ORDER WITH UNLISTED INCLUDED;

SET ACTIVE CONTEXT AS ctxt_year_STI;
```

Context 3: The user prefers hardcover books over paperback ones.

```
CREATE POSITIVE CONTEXT ctxt_cover_STX (
     cover integer,
     BINDING KEY (cover) REFERENCES books(cover)
) AS TOTAL ORDER WITH UNLISTED EXCLUDED;

SET ACTIVE CONTEXT AS ctxt_cover_STX;
```

Context 4: The user does not prefer (wants to avoid) any science fiction books.

```
CREATE NEGATIVE CONTEXT ctxt category GQI (
     category integer,
     BINDING KEY (category) REFERENCES books(category)
) AS EQUIVALENCE WITH UNLISTED INCLUDED;

SET ACTIVE CONTEXT AS ctxt_category_GQI;
```

Context 5: The user prefers books published in 2005, and then those published in 2006 before all other books. For the books that are similarly ranked, the user prefers hardcover books over books with paperback cover.

```
SET ACTIVE CONTEXT FOR user1
AS ctxt_year_STI, ctxt_cover_STX
WITH RANKING ORDER ctxt_year_STI, ctxt_cover_STX;
```

Effect of Contexts in Queries

Context Class	**ORDER BY** Clause	Join Operation
GQN	X	NOT IN
SQN	X	LEFT OUTER JOIN
SQX	X	INNER JOIN
STN	√	LEFT OUTER JOIN
STX	√	INNER JOIN
SPN	√	LEFT OUTER JOIN
SPX	√	INNER JOIN

Instantiating Hippocratic Databases

Using Chameleon, we limit both disclosure and retention of patients data in a healthcare facility as what happens in Hippocratic Databases. Whenever a patient is admitted to the facility, he/she has to sign a privacy policy. The privacy policy specified which information is to be released to which recipient. Moreover, the policy also specifies for which purposes the information is to be released. On an opt-in basis, the healthcare facility also allows patients to choose if they want any of their personal information to be released to other recipients. By the end of the retention period, the patient data should have fulfilled the purposes for which the data has been collected. After this period, different recipients cannot retrieve the data.

```
CREATE OBJECT CONTEXT patient_privacy_pref (
     recipient varchar(30),
     purpose varchar(30),
     pid integer,          pid_pref boolean,          name_pref boolean,
     age_pref boolean,     address_pref boolean,      phone_pref boolean,
     BINDING KEY(pid) REFERENCES patient(pid)
);
CREATE OBJECT CONTEXT policy_signature (
     pid integer,
     sign_date date,
     BINDING KEY(pid) REFERENCES patient(pid)
);
CREATE POSITIVE CONTEXT identity_activity (
     job varchar(30),
     activity varchar(30),
     BINDING KEY(job, activity)
          REFERENCES patient_privacy_pref(recipient, purpose)
     SUBSTITUTE patient(pid)
          WITH (CASE WHEN patient_privacy_pref.pid_pref AND today() <= policy_signature.sign_date + 90
                THEN patient.pid ELSE NULL),
     SUBSTITUTE patient(name)
          WITH (CASE WHEN patient_privacy_pref.name pref AND today() <= policy_signature.sign date + 90
                THEN patient.name ELSE NULL)
          ...
) AS EQUIVALENCE WITH UNLISTED EXCLUDED;
```

pid	name	age	address	phone
1	Alice Adams	10	1 April Ave.	111-1111
2	Bob Blaney	20	2 Brook; Blvd.	222-2222
3	Carl Carson	30	3 Cricket Ct.	333-3333
4	David Daniels	40	4 Dogwood Dr.	444-4444

recipient	purpose	pid	pid_pref	name_pref	age_pref	address_pref	phone_pref
Charity	Solicitation	1	√	√	√	√	√
Nurse	Treatment	1	√	√	√	X	√
Account clerk	Billing	1	√	√	X	√	X
Charity	Solicitation	2	X	X	X	X	√
Nurse	Treatment	2	√	√	√	X	√
Account clerk	Billing	2	√	√	X	√	√
Charity	Solicitation	3	√	X	X	√	√
Nurse	Treatment	3	√	√	√	X	√
Account clerk	Billing	3	√	√	X	√	√
Charity	Solicitation	4	√	√	X	X	X
Nurse	Treatment	4	√	√	√	X	√
Account clerk	Billing	4	√	√	X	√	√

(job, activity)	pid	name	age	address	phone
(Charity, Solicitation)	1	Alice Adams	10	1 April Ave.	111-1111
	3			3 Cricket Ct.	333-3333
	4	David Daniels			
(Nurse, Treatment)	1	Alice Adams	10		111-1111
	2	Bob Blaney	20		222-2222
	3	Carl Carson	30		333-3333
	4	David Daniels	40		444-4444
(Account clerk, Billing)	1	Alice Adams		1 April Ave.	111-1111
	2	Bob Blaney		2 Brook; Blvd.	222-2222
	3	Carl Carson		3 Cricket Ct.	333-3333
	4	David Daniels		4 Dogwood Dr.	444-4444

Instantiating Spatial Databases

In Chameleon, we show how to model Spatial Databases. With the help of the context-awareness, we answer both range and nearest-neighbor queries. We can answer skyline queries as well. Consider a real-estate database containing information about houses. The houses table has the following schema: houses(id, bedrooms, price, city). An application developer is interested in providing some spatial queries to this database, but has no privileges to add the location of the house to this table. An object context is created to add the location of houses.

Object Context:

```
CREATE OBJECT CONTEXT house_loc (
     id integer,    x integer,    y integer,
     PRIMARY KEY(id),
     BINDING KEY id REFERENCES houses(id)
);
```

Range Query:

```
CREATE POSITIVE CONTEXT houses_in_region (
     x1 integer,    y1 integer,
     x2 integer,    y2 integer,
     BINDING KEY() REFERENCES house loc
        WITH contained(house_loc.x, house_loc.y,
                       x1, y1, x2, y2)
) AS EQUIVALENCE WITH UNLISTED EXCLUDED;
```

kNN Query:

```
CREATE POSITIVE CONTEXT nearby_houses (
     x integer,    y integer,
     BINDING KEY() REFERENCES house_loc
        WITH true
)
AS TOTAL ORDER
    USING dist(x, y, house_loc.x, house_loc.y)
WITH UNLISTED EXCLUDED;
```

Skyline Query:

```
SET ACTIVE CONTEXT FOR user2
AS houses in region, nearby houses
WITH SKYLINE OF nearby_houses.rank MIN,
                houses.price MIN;
```

Context composition

A complex context may be composed from basic ones. Such composition may involve compiling more than one context whose contextual relation is an ordering relation. We provide three mechanisms to resolve the conflict among the different orders of object imposed by these contexts.

- Using the **ORDER BY** clause
- Using ranking algorithms
- Using skyline algorithms