Project Deliverable 3

Querying your Database

Presented to

Bettina Kemme

COMP-421 – Database Systems

By

Patrick Desmarais (260 329 253)

Simon Hlywa (260 231 002)

Guillaume Viger (260 309 396)

McGill University

March 17th, 2011

Note on Scripts
Scripts were executed using two types of DB2 commands: `db2 -f script.clp` [1] (for scripts using line escape characters) or the `db2 -tf script.clp` [2] (for scripts using semi-colons at the end of each statement). Numbers [1] and [2] are referred next to script names presented below. A readme file is also included with the attached scripts to explain how to create tables before running scripts. If you have any problems with scripts, you can contact Patrick (patrick.desmarais@mail.mcgill.ca).

1. Four Select-From-Where Queries

The following queries exhibit some interesting features of SQL:

1 Show the name of all offered entities that User A and User B exchanged

This query requires to join two tables (Offered_entities and Transactions) and perform a union of queries.

Relations used
Offered_entities

```
Column                         Type      Type
name                           schema    name                Length   Scale Nulls
------------------------------ --------- ------------------ -------- ----- ------
OEID                           SYSIBM    BIGINT                     8     0 No
NAME                           SYSIBM    VARCHAR                  100     0 No
ENTITY_DESC                    SYSIBM    VARCHAR                 1000     0 Yes
PRICE                          SYSIBM    REAL                       4     0 Yes
STATE                          SYSIBM    SMALLINT                   2     0 Yes
OWNER                          SYSIBM    VARCHAR                  100     0 No
```

Transactions

```
Column                         Type      Type
name                           schema    name                Length   Scale Nulls
------------------------------ --------- ------------------ -------- ----- ------
DATETIME                          SYSIBM TIMESTAMP                10     0 No
PROVIDER                          SYSIBM VARCHAR                  30     0 No
RECEIVER                          SYSIBM VARCHAR                  30     0 No
OEID                              SYSIBM BIGINT                    8     0 No
AMOUNT                            SYSIBM REAL                      4     0 Yes
```

SQL Command

```
SELECT    OE.name
FROM      Offered_entities OE, Transactions T
WHERE     T.oeid = OE.oeid AND T.provider = 'lonelygirl15@example.com'
          AND T.receiver = 'sherlock_9382@aol.com'
UNION
SELECT    OE.name
FROM      Offered_entities OE, Transactions T1
WHERE     T1.oeid = OE.oeid AND T1.provider = 'sherlock_9382@aol.com'
          AND T1.receiver = 'lonelygirl15@example.com'
```

Script

See *script1-1.clp* [1] for the script used to generate the results.

2 Show the names of all users who offer at least one character of level 65 or higher

This requires to join three tables (Users, Offered Entities and Characters) and perform filtering.

Users

Column name	Type schema	Type name	Length	Scale	Nulls
EMAIL	SYSIBM	VARCHAR	100	0	No
PSWD	SYSIBM	VARCHAR	50	0	Yes
CREDIT_CARD	SYSIBM	CHARACTER	16	0	Yes

Offered_entities

Column name	Type schema	Type name	Length	Scale	Nulls
OEID	SYSIBM	BIGINT	8	0	No
NAME	SYSIBM	VARCHAR	100	0	No
ENTITY_DESC	SYSIBM	VARCHAR	1000	0	Yes
PRICE	SYSIBM	REAL	4	0	Yes
STATE	SYSIBM	SMALLINT	2	0	Yes
OWNER	SYSIBM	VARCHAR	100	0	No

Characters

Column name	Type schema	Type name	Length	Scale	Nulls
OEID	SYSIBM	BIGINT	8	0	No
SEX	SYSIBM	CHARACTER	1	0	Yes
RACE	SYSIBM	VARCHAR	20	0	Yes
SPECIALIZATION	SYSIBM	VARCHAR	20	0	Yes
CLASS	SYSIBM	VARCHAR	20	0	Yes
LEVEL	SYSIBM	SMALLINT	2	0	Yes

SQL Command

```
SELECT    U.email
FROM      Users U
WHERE     EXISTS(    SELECT *
                     FROM Characters C, Offered_entities OE
                     WHERE U.email = OE.owner AND OE.oeid = C.oeid
                          AND OE.state = 0 AND C.level > 65 )
```

Script

See *script1-2.clp* [1] for the script used to generate the results.

3 Return the name of the character(s) with the most equipment and its (their) owner

This requires the use of an aggregate function, a View to manipulate the various tables and the ALL keyword to perform a comparison again the the results to determine the maximum.

Offered_entities

Column name	Type schema	Type name	Length	Scale	Nulls
OEID	SYSIBM	BIGINT	8	0	No
NAME	SYSIBM	VARCHAR	100	0	No
ENTITY_DESC	SYSIBM	VARCHAR	1000	0	Yes
PRICE	SYSIBM	REAL	4	0	Yes
STATE	SYSIBM	SMALLINT	2	0	Yes
OWNER	SYSIBM	VARCHAR	100	0	No

Equipped_with

Column name	Type schema	Type name	Length	Scale	Nulls
CID	SYSIBM	BIGINT	8	0	No
IID	SYSIBM	BIGINT	8	0	No

SQL Command

```
CREATE VIEW Temp (cid, count) AS
        SELECT EW.cid, count(*) AS count
        FROM Equipped_with EW
        GROUP BY EW.cid

SELECT OE.name, OE.owner
FROM Offered_entities OE, Temp T
WHERE       OE.oeid = T.cid AND OE.state = 0
            AND T.count >= ALL(SELECT count
                               FROM Temp)
```

Script

See *script1-3.clp* [1] for the script used to generate the results.

4 Return all the characters (with all their information) sorted by price in decreasing order

This is a simpler command requiring to order the Characters table by price from highest to lowest.

Characters

```
Column                         Type      Type
name                           schema    name                Length   Scale Nulls
------------------------------ --------- ------------------ -------- ----- ------
OEID                           SYSIBM    BIGINT                   8     0 No
SEX                            SYSIBM    CHARACTER                1     0 Yes
RACE                           SYSIBM    VARCHAR                 20     0 Yes
SPECIALIZATION                 SYSIBM    VARCHAR                 20     0 Yes
CLASS                          SYSIBM    VARCHAR                 20     0 Yes
LEVEL                          SYSIBM    SMALLINT                 2     0 Yes
```

Offered_entities

```
Column                         Type      Type
name                           schema    name                Length   Scale Nulls
------------------------------ --------- ------------------ -------- ----- ------
OEID                           SYSIBM    BIGINT                   8     0 No
NAME                           SYSIBM    VARCHAR                100     0 No
ENTITY_DESC                    SYSIBM    VARCHAR               1000     0 Yes
PRICE                          SYSIBM    REAL                     4     0 Yes
STATE                          SYSIBM    SMALLINT                 2     0 Yes
OWNER                          SYSIBM    VARCHAR                100     0 No
```

SQL Command

```
SELECT OE.oeid, OE.name, OE.entity_desc, OE.price, OE.owner, C.sex, C.race,
C.specialization, C.class, C.level
        FROM Offered_entities OE, Character C
        WHERE OE.oeid = C.oeid
        ORDER BY OE.price DESC
```

Script

See *script1-4.clp* [2] for the script used to generate the results.

2. Three Data Modifications

1 Lower by 10% the price of all Dwarf characters (Special discount on Dwarf Day)

Characters

```
Column                         Type      Type
name                           schema    name                Length   Scale Nulls
------------------------------ --------- ------------------ -------- ----- ------
OEID                           SYSIBM    BIGINT                     8     0 No
SEX                            SYSIBM    CHARACTER                  1     0 Yes
RACE                           SYSIBM    VARCHAR                   20     0 Yes
SPECIALIZATION                 SYSIBM    VARCHAR                   20     0 Yes
CLASS                          SYSIBM    VARCHAR                   20     0 Yes
LEVEL                          SYSIBM    SMALLINT                   2     0 Yes
```

Offered_entities

```
Column                         Type      Type
name                           schema    name                Length   Scale Nulls
------------------------------ --------- ------------------ -------- ----- ------
OEID                           SYSIBM    BIGINT                     8     0 No
NAME                           SYSIBM    VARCHAR                  100     0 No
ENTITY_DESC                    SYSIBM    VARCHAR                 1000     0 Yes
PRICE                          SYSIBM    REAL                       4     0 Yes
STATE                          SYSIBM    SMALLINT                   2     0 Yes
OWNER                          SYSIBM    VARCHAR                  100     0 No
```

SQL Command

```
UPDATE Offered_entities
SET price = 0.9*(SELECT OE.price
                 FROM Offered_entities OE, Characters C
                 WHERE Offered_entities.oeid = OE.oeid AND C.oeid = OE.oeid
                       AND C.race = 'Dwarf')
WHERE EXISTS(    SELECT OE.price
                 FROM Offered_entities OE, Characters C
                 WHERE Offered_entities.oeid = OE.oeid AND C.oeid = OE.oeid
                 AND C.race = 'Dwarf' AND Offered_entities.state = 0)
```

Script

See *script2-1.clp* [1] for the script used to generate the results.

2 Ban a user: delete all his offered_entities

The user will be kept in the Users table for a complete Transactions history for other users. Only the user's passwd will be set to NULL; hence, the application will be in charge of interpreting if the user has the passwd set to null, then he is inactive.

Offered Entities

Column name	Type schema	Type name	Length	Scale	Nulls
OEID	SYSIBM	BIGINT	8	0	No
NAME	SYSIBM	VARCHAR	100	0	No
ENTITY_DESC	SYSIBM	VARCHAR	1000	0	Yes
PRICE	SYSIBM	REAL	4	0	Yes
STATE	SYSIBM	SMALLINT	2	0	Yes
OWNER	SYSIBM	VARCHAR	100	0	No

Users

Column name	Type schema	Type name	Length	Scale	Nulls
EMAIL	SYSIBM	VARCHAR	100	0	No
PSWD	SYSIBM	VARCHAR	50	0	Yes
CREDIT_CARD	SYSIBM	CHARACTER	16	0	Yes

SQL Command

[User] will be replaced by the specified user's email. Since tables in the ISA hierarchy were built with DELETE ON CASCADE, this query is simpler than manually deleting entities in the child tables.

```
DELETE
        FROM Offered_entities OE
        WHERE OE.owner = '[User]'

UPDATE Users U
        SET U.pswd = NULL
        WHERE U.email = '[User]';
```

Script

See *script2-2.clp* [2] for the script used to generate the results.

3 Set all offered_entities of an account as sold

Offered Entities

```
Column                         Type      Type
name                           schema    name               Length   Scale Nulls
------------------------------ --------- ------------------ -------- ----- ------
OEID                              SYSIBM BIGINT                    8     0 No
NAME                              SYSIBM VARCHAR                 100     0 No
ENTITY_DESC                       SYSIBM VARCHAR                1000     0 Yes
PRICE                             SYSIBM REAL                      4     0 Yes
STATE                             SYSIBM SMALLINT                  2     0 Yes
OWNER                             SYSIBM VARCHAR                 100     0 No
```

Accounts

```
Column                         Type      Type
name                           schema    name               Length   Scale Nulls
------------------------------ --------- ------------------ -------- ----- ------
OEID                              SYSIBM BIGINT                    8     0 No
EXPANSION_LEVEL                   SYSIBM SMALLINT                  2     0 Yes
```

SQL Command

```
CREATE VIEW TEMP1 (oeid, state) AS
        SELECT DISTINCT CO.cid, OE.state
        FROM Contain CO, Offered_entities OE
        WHERE CO.aid = 9301 AND OE.oeid = CO.cid

UPDATE Offered_entities
SET state = 1
WHERE EXISTS(SELECT OE.oeid
                    FROM Offered_entities OE
                    WHERE Offered_entities.oeid = OE.oeid AND OE.oeid = 9301)
        OR
        EXISTS(SELECT T.oeid
                    FROM TEMP1 T
                    WHERE Offered_entities.oeid = T.oeid)
        OR
        EXISTS(SELECT EW.iid
            FROM TEMP1 T1, Equipped_with EW, Offered_entities OE
            WHERE T1.oeid = EW.cid AND EW.iid = OE.oeid AND
            EW.iid = Offered_entities.oeid)
```

Script

See *script2-3.clp* [1] for the script used to generate the results.

3. Two Views

1 All Characters to be sold

This view presents the list of characters to be sold. It encompasses the information found in the Offered_entities and Characters tables.

SQL Command

CREATE VIEW OfferedCs (id, name, description, owner, price, sex, race, specialization, class, level)
AS
SELECT OE.oeid, OE.name, OE.entity_desc, OE.owner, OE.price, C.sex, C.race, C.specialization, C.class, C.level
FROM Offered_entities OE, Characters C
WHERE OE.oeid = C.oeid AND OE.state = 0

System Response

[See *script3-1.clp* for the script used to create the view. Note that this script must be run using the following command on the db2 server: `db2 -td! -f script3-1.clp`]

```
$ db2 -td! -f script3-1.clp

   Database Connection Information

 Database server        = DB2/LINUXX8664 9.1.0
 SQL authorization ID   = PDESMA6
 Local database alias   = CS421


DB20000I  The SQL command completed successfully.

db2 => describe table offeredcs

Column                         Type      Type
name                           schema    name                Length   Scale Nulls
------------------------------ --------- ------------------- -------- ----- ------
ID                             SYSIBM    BIGINT                     8     0 No
NAME                           SYSIBM    VARCHAR                  100     0 No
DESCRIPTION                    SYSIBM    VARCHAR                 1000     0 Yes
OWNER                          SYSIBM    VARCHAR                  100     0 No
PRICE                          SYSIBM    REAL                       4     0 Yes
SEX                            SYSIBM    CHARACTER                  1     0 Yes
RACE                           SYSIBM    VARCHAR                   20     0 Yes
SPECIALIZATION                 SYSIBM    VARCHAR                   20     0 Yes
CLASS                          SYSIBM    VARCHAR                   20     0 Yes
LEVEL                          SYSIBM    SMALLINT                   2     0 Yes

  10 record(s) selected.
```

```
db2 => select * from offeredcs

ID   NAME      DESCRIPTION  OWNER     PRICE  SEX RACE      SPECI... CLASS   LEVEL
---- --------- ------------ --------- ------ --- --------- -------- ------- -----
                9105 Borag ... Hailing f... battas... +5E+1  M   Worgen    DPS      Warrior 70
9106 Melfor... Silent De... battas... +6E+1  F   Night Elf DPS      Warrior 85
9107 Baltha... This char... battas... +4E+1  M   Draenei   Tank     Paladin 20

  3 record(s) selected.
```

Update Script

[See *script3-1_updates.clp* [2] for the script used to test updates.]

```
$ db2 -tf script3-1_updates.clp

   Database Connection Information

 Database server        = DB2/LINUXX8664 9.1.0
 SQL authorization ID   = PDESMA6
 Local database alias   = CS421


DB21034E  The command was processed as an SQL statement because it was not a
valid Command Line Processor command.  During SQL processing it returned:
SQL0551N  "PDESMA6" does not have the privilege to perform operation "INSERT"
on object "PDESMA6.OFFEREDCS".  SQLSTATE=42501

DB21034E  The command was processed as an SQL statement because it was not a
valid Command Line Processor command.  During SQL processing it returned:
SQL0551N  "PDESMA6" does not have the privilege to perform operation "INSERT"
on object "PDESMA6.OFFEREDCS".  SQLSTATE=42501

DB21034E  The command was processed as an SQL statement because it was not a
valid Command Line Processor command.  During SQL processing it returned:
SQL0150N  The target fullselect, view, typed table, materialized query table,
or staging table in the INSERT, DELETE, UPDATE, or MERGE statement is a target
for which the requested operation is not permitted.  SQLSTATE=42807

DB21034E  The command was processed as an SQL statement because it was not a
valid Command Line Processor command.  During SQL processing it returned:
SQL0150N  The target fullselect, view, typed table, materialized query table,
or staging table in the INSERT, DELETE, UPDATE, or MERGE statement is a target
for which the requested operation is not permitted.  SQLSTATE=42807
```

Updatable?

This view incorporates information from multiple tables; this view is a complex view. Even when we create INSTEAD OF triggers to do insert, update and delete operations on this view, the view is non-updateable based on DB2 documentation and the description given below.

For example, here is an example of an instead of trigger for inserting a new tuple in a complex view in DB2. However, this trigger will not work because OfferedCs is based on a join of more than one base table (see the first two error messages above).

```
CREATE TRIGGER OfferedCs_insert INSTEAD OF INSERT ON OfferedCs
        REFERENCING NEW AS newoc
        FOR EACH ROW MODE DB2SQL
                BEGIN ATOMIC
                        INSERT INTO Offered_entities (oeid, name, entity_desc, price, owner)
                        VALUES (newoc.id, newoc.name, newoc.description, newoc.price,
                                COALESCE ( (SELECT U.email FROM Users U
                                                WHERE U.email = newoc.owner),
                                                RAISE_ERROR('70001', 'Unknown user')));
                        INSERT INTO Characters
                        VALUES (newoc.id, newoc.sex, newoc.race, newoc.specialization,
                                newoc.class, newoc.level);
                END
```

2 All available Offered_entities to be sold

This view restricts the access to certain columns. Users who are granted access to this view will only be allowed to see available offered_entities. This view contrasts with the previous one where it only has one base class.

SQL Command

```
CREATE VIEW AvailableOEs (id, name, description, owner, price)
AS
        SELECT oeid, name, entity_desc, owner, price
        FROM Offered_entities
        WHERE state = 0
```

System Response

[See *script3-2.clp* [2] for the script used to create the view.]

```
$ db2 -tf script3-2.clp

   Database Connection Information

 Database server        = DB2/LINUXX8664 9.1.0
 SQL authorization ID   = PDESMA6
 Local database alias   = CS421


DB20000I  The SQL command completed successfully.

db2 => describe table availableoes

Column                         Type      Type
name                           schema    name                Length   Scale Nulls
------------------------------ --------- ------------------ -------- ----- ------
ID                             SYSIBM    BIGINT                    8     0 No
NAME                           SYSIBM    VARCHAR                 100     0 No
DESCRIPTION                    SYSIBM    VARCHAR                1000     0 Yes
OWNER                          SYSIBM    VARCHAR                 100     0 No
PRICE                          SYSIBM    REAL                      4     0 Yes

  5 record(s) selected.

db2 => select id, name, price from availableoes where owner = 'lonelygirl15@example.com'

ID    NAME                 PRICE
---- ---------------------- ------------------------
9100 Borag of the Many      +5.00000E+001

  1 record(s) selected.
```

Update Script

[See *script3-2_updates.clp* [2] for the script used to test updates.]

```
$ db2 -tf script3-2_updates.clp

   Database Connection Information

 Database server        = DB2/LINUXX8664 9.1.0
 SQL authorization ID   = PDESMA6
 Local database alias   = CS421


DB20000I  The SQL command completed successfully.

DB20000I  The SQL command completed successfully.

DB20000I  The SQL command completed successfully.

db2 => select * from availableoes

ID    NAME          DESCRIPTION                 OWNER                 PRICE
----- ------------  --------------------------- --------------------  -------------
9101  Melfor the... Silent Death. This...       merlin428@example.com  +9.00100E+001
9102  Balthazar07   This character has ...      sherlock_9382@aol.com  +3.99900E+001

  2 record(s) selected.

db2 => select * from offered_entities

OEID  NAME          ENTITY_DESC                 PRICE          STATE   OWNER
----- ------------  --------------------------- -------------  ------  -------------
9101  Melfor ...    Silent Death. This charac... +9.00100E+001  0       merlin428@...
9102  Balthazar07   This character has comple... +3.99900E+001  0       sherlock_9...
9103  Bobo_the...   Buy this character it is ... +2.55000E+001  1       tinyT1m@ex...
9104  Galadrie...   Completes the collection ... +7.59900E+001  1       tomB0mbadi...
9109  Gold Potion   Help for your current quest. +2.00100E+001  0       lonelygirl...

  5 record(s) selected.
```

Updatable?

Even though inserts in this table need to be coupled with other data modifications operations, from a technical point of view, this view is updatable since it refers to only one base table (delete), and it includes the primary key of that base table (update, insert). Note that this view is best suited for select-from-where, delete and certain update (name, description, price) operations.

Summary on updatable views in DB2

From executed tests and the documentation found online, DB2 Version 9.1 allows insert, update and delete operations on a view when they are derived from one base table where the view's attributes must contain the primary key(s) and where the view's attributes can be referred via a one-to-one mapping to the base table attributes. Instead of triggers can be used to do insert, update and delete operations on a non-updateable view based on one base table where the SELECT clause in the view definition contains an arithmetic or string expression, a built-in function, or a constant.

A more detailed description of the limit of DB2 to allow the different operations was found at this web page: http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=/com.ibm.db29.doc.codes/n150.htm. The following frame is the copy of the content of interest. We see that updates on views are very restrictive.

- Inserts into a view are prohibited if:
  - The view definition contains a join, a GROUP BY, or a HAVING clause.
  - The SELECT clause in the view definition contains the DISTINCT qualifier, an arithmetic expression, a string expression, a built-in function, or a constant.
  - Two or more columns of the view are derived from the same column.
  - A base table of the view contains a column that does not have a default value and is not included in the view.
- Updates to a view are prohibited if:
  - The view definition contains a join, a GROUP BY, or a HAVING clause.
  - The SELECT clause in the view definition contains the DISTINCT qualifier or a built-in function.

  Also, a given column in a view cannot be updated (that is, the values in that column cannot be updated) if the column is derived from an arithmetic expression, a constant, a column that is part of the key of a partitioned index, or a column of a catalog table that cannot be updated.
- Deletes against a view are prohibited if:
  - The view definition contains a join, a GROUP BY, or a HAVING clause.
  - The SELECT clause in the view definition contains the DISTINCT qualifier or a built-in function.
- Truncates against a view are always prohibited.

4. One Check Constraint

Accounts can only have expansion levels from 0 up until level 3 (0 corresponding to the base World of Warcraft game and 3 corresponding to the latest expansion called “Cataclysm”).

Revised Schema (SQL Command)

```
CREATE TABLE Accounts(
        oeid BIGINT NOT NULL PRIMARY KEY,
        expansion_level SMALLINT CONSTRAINT expansions CHECK( expansion_level >= 0
        FOREIGN KEY (oeid) REFERENCES Offered_entities ON DELETE CASCADE
)
```

System Response

(creation)

```
DB20000I  The SQL command completed successfully.
```

(trying to insert an account with expansion_level 4)

```
DB21034E  The command was processed as an SQL statement because it was not a valid Command Line
Processor command.  During SQL processing it returned: SQL0545N  The requested operation is not
allowed because a row does not satisfy the check constraint "GVIGER.ACCOUNTS.SQL110315215332701".
SQLSTATE=23513
```

(repercussion on Contain table)

```
DB21034E  The command was processed as an SQL statement because it was not a valid Command Line
Processor command.  During SQL processing it returned: SQL0530N  The insert or update value of
the FOREIGN KEY "GVIGER.CONTAIN.SQL110315215332881" is not equal to any value of the parent
key of the parent table. SQLSTATE=23503
```

Script

See *script4.clp* [1] for the script used to generate the results.

5. Two Triggers

1 After a transaction has been made the buyer becomes the new owner.

```
CREATE TRIGGER Change_owner
AFTER INSERT ON Transactions
REFERENCING NEW AS newrow
FOR EACH ROW
  UPDATE Offered_entities
  SET owner = newrow.receiver
  WHERE oeid = newrow.oeid
```

Script

See *script5-1.clp* [1] for the script used to generate the results.

2 Before inserting a new character, the offered_entity id must be unique from other tables (ISA covering contraint)

```
CREATE TRIGGER characters_insert
BEFORE INSERT ON characters
REFERENCING NEW AS newc
FOR EACH ROW
        WHEN (          newc.oeid IN (SELECT oeid FROM Custom_character_levelings)
                OR      newc.oeid IN (SELECT oeid FROM Items)
                OR      newc.oeid IN (SELECT oeid FROM Accounts))
            BEGIN ATOMIC
                SIGNAL SQLSTATE '75001'
                    SET MESSAGE_TEXT = 'This id is already assigned to another
        offered_entity of different type.';
            END
```

Script

[See *script5-2.clp* for the script used to create the view. Note that this script must be run using the following command on the db2 server: `db2 -td! -f script5-2.clp`]

6. XML-Extension

1 Relation added

The relation added is described in *admins.dtd*. This relation will keep track of administrators information for the company director. Information such as since when the administrator is part of the development and its rights are stored in this table.

2 CREATE

The create script is presented in *admins_create.clp* [2].

```
CREATE TABLE admins (id INT NOT NULL PRIMARY KEY,
                     Info XML);
```

3 INSERTs

The insert operations is presented *admins_inserts.clp* [2].
10 administrators are inserted with variable names and rights, and different since attributes.

```
INSERT INTO admins (id, Info) VALUES
(1, '<admin since="1997">
        <name>Joe</name>
        <right>Ban_user</right>
    </admin>'),
(2, '<admin since="1999">
        <name>Julien</name>
        <right>Reset_password</right>
    </admin>'),
(3, '<admin since="2011">
        <name>Jonathan</name>
        <right>Ban_user</right>
        <right>Reset_password</right>
    </admin>'),
(4, '<admin since="2004">
        <lname>Fio</lname>
        <right>Remove_oes</right>
    </admin>'),
(5, '<admin since="2004">
        <lname>Fit</lname>
        <right>Remove_oes</right>
    </admin>'),
(6, '<admin since="2004">
        <lname>Fip</lname>
        <right>Remove_oes</right>
    </admin>'),
```

```
(7,     '<admin since="2005">
            <name>Pat</name>
            <lname>Copu</lname>
            <right>Ban_user</right>
        </admin>'),
(8,     '<admin since="2005">
            <name>Pit</name>
            <lname>Copu</lname>
            <right>Reset_password</right>
            <right>Clean_transactions</right>
        </admin>'),
(9,     '<admin since="2005">
            <name>Lel</name>
            <lname>George</lname>
            <right>Remove_oes</right>
        </admin>'),
(10, '<admin since="2008">
            <name>Lio</name>
            <lname>Koli</lname>
            <right>Ban_user</right>
        </admin>');
```

4 SELECTs

Select operations are presented in *admins_selects.clp* [2].

This select operation returns the name of administrators that have been part of the admins team since 2006.

```
SELECT XMLQUERY (
'for $d in $doc/admin
return $d/name'
passing INFO as "doc")
FROM admins m WHERE XMLEXISTS (
'$i/admin[@since>2005]' passing m.info as "i");
```

This select operation returns all administrators with a name. Those that only have a lastname or no name are not returned.

```
SELECT XMLQUERY (
'for $d in $doc/admin
return $d/name'
passing INFO as "doc")
FROM admins m WHERE XMLEXISTS (
```

'$i/admin/name' passing m.info as "i");