COS 326
Database Systems

Lecture 5
Object-Relational Databases (1)
Chapter 29
4 August 2015
Admin matters

- marking of prac 1?
- Documents for assignment essays & presentations (Friday)
- next 2 weeks

<table>
<thead>
<tr>
<th>Week</th>
<th>Date</th>
<th>Day</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4 Aug</td>
<td>Tues</td>
<td>L5: Object-Relational DBMS</td>
</tr>
<tr>
<td>5</td>
<td>Aug</td>
<td>Wed</td>
<td>L6: Object-Relational DBMS</td>
</tr>
<tr>
<td>7</td>
<td>Aug</td>
<td>Fri</td>
<td><strong>Practical 2:</strong> Object-Relational DBMS (PostgreSQL)</td>
</tr>
<tr>
<td>4</td>
<td>11 Aug</td>
<td>Tues</td>
<td>L7: Object-Relational DBMS</td>
</tr>
<tr>
<td>12</td>
<td>Aug</td>
<td>Wed</td>
<td>L8: Object-Relational DBMS</td>
</tr>
<tr>
<td>14</td>
<td>Aug</td>
<td>Fri</td>
<td><strong>Practical 3:</strong> Object-Relational DBMS (PostgreSQL)</td>
</tr>
</tbody>
</table>
In this lecture

1. Introduction to ORDBs
   - Why use ORDBs?
   - advantages & disadvantages of ORDBMSs
     - Reading for the student

2. Database manifestos: Reading for the student

3. Postgres – an early ORDBMS

4. SQL:2008

5. Relationship between postgres, SQL:2008 & PostgreSQL

6. PostgreSQL basics
Some arguments for & against RDBMS

Arguments against RDBMSs:
• Relational DBMS are satisfactory for normal business applications but cannot support complex applications (support for: OODBMS)

Arguments for RDBMSs:
• Relational model is part of any real-world DBMS
• Businesses have made major investments in RDBMS
• complex applications can be handled by extensions to the relational model (support for: ORDBMS)
• Smooth transition through upgrades to newer DBMS version which is ORDBMS
• Database Market share
  • RDBMS: very large (billions of dollars)
  • OODBMS: very small but growing
1. ORDBMS

- extend relational model with OO features
  - classes & objects,
  - aggregation & inheritance
  - polymorphism & dynamic binding
  - etc

- Bridges the gap
  - OO programming languages and RDBMS data
  - OO data modeling and RDBMS
2. Reading for the student

- Advantages & disadvantages of the ORDBMS approach

- Database manifestos
  - Third-generation database manifesto
  - Third manifesto
3. Postgres – early ORDBMS

- Postgres (‘Post Ingres’)
  - research DBMS designed to be potential successor to INGRES.
  - query language for Postgres was called postquel

- Postgres extends the Relational model to include:
  - Abstract Data Types (ADTs) (DEFINE TYPE)
  - Data of type ‘procedure’ (DEFINE PROCEDURE)
  - Operators and precedence + associativity of operators (DEFINE OPERATOR)

- Supports OO constructs such as:
  - inheritance
  - aggregation
    - attributes that reference tuples (objects) in other relations.
4. SQL:2008 - New OO Features

• **Row types**

• **User-defined types**
  • distinct types, structured types
  • supertype/subtype relationship

• **User-defined routines**
  • procedures, functions, *methods*, operators

• **Collection types**
  • arrays, sets, lists, and multisets
4. SQL:2008  Row type

- **Row type**
  - sequence of \((field\_name, data\_type)\) pairs
- **Column of table can contain row values**
- **A complete row can be:**
  - stored in a variable
  - passed as an argument to a function
  - returned as return value from a function call

<table>
<thead>
<tr>
<th>branchNo</th>
<th>Address (is a row type)</th>
</tr>
</thead>
<tbody>
<tr>
<td>B005</td>
<td>street</td>
</tr>
<tr>
<td></td>
<td>22 Deer Rd</td>
</tr>
<tr>
<td></td>
<td>city</td>
</tr>
<tr>
<td></td>
<td>London</td>
</tr>
<tr>
<td></td>
<td>postcode</td>
</tr>
<tr>
<td></td>
<td>SW14Eh</td>
</tr>
</tbody>
</table>
CREATE TABLE Branch (  
    branchNo CHAR(4),  
    address ROW (     street VARCHAR(25),  
                     city VARCHAR(15),  
                     postcode VARCHAR(8) ) );  

INSERT INTO Branch  
VALUES ( 'B005',  
         ROW ('22 Deer Rd', 'London', 'SW14EH') );

<table>
<thead>
<tr>
<th>branchNo</th>
<th>Address (is a row type)</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>B005</td>
<td>22 Deer Rd</td>
<td>London</td>
<td>SW14EH</td>
<td></td>
</tr>
</tbody>
</table>
4. SQL:2008  User-defined types (UDTs == ADTs of postquel)  (1)

- UDT may be distinct (simple) type or structured type

- Distinct (simple) types (= domain), e.g.
  
  CREATE TYPE OwnerNoType AS VARCHAR(5) FINAL;
  CREATE TYPE StaffNoType AS VARCHAR(5) FINAL;
  CREATE TYPE NameType AS VARCHAR(15) FINAL;

- makes a distinction between same underlying data type
- enforce type checking
- what does FINAL mean?
4. SQL:2008  User-defined types (UDTs == ADTs of postquel)  (2)

- **structured UDT definition consists of**
  - Attribute definitions
  - Routine declarations (methods)
  - Operator declarations (still planned)
  - Equality and ordering relationships

- **e.g. of structured types:**

  CREATE TYPE PersonType AS (  
    dateOfBirth DATE, gender CHAR,  
    fName NameType NOT NULL,  
    lName NameType NOT NULL)  
  ..................
4. SQL:2008 User-defined types (UDTs == ADTs of postquel) (3)

- **structured UDT definition consists of**
  - Attribute definitions
  - **Routine declarations (methods)**
  - Operator declarations (still planned)
  - Equality and ordering relationships

- **e.g. of structured types:**

  ```sql
  CREATE TYPE PersonType AS (dateOfBirth DATE) INSTANTIABLE NOT FINAL REF IS SYSTEM GENERATED
  INSTANCE METHOD age() RETURNS INTEGER;
  INSTANCE METHOD age(DOB DATE) RETURNS PersonType;
  ```

- Stored attribute
- Instances can be created
- Subtypes can be defined
- Virtual/derived attribute
- Methods

© Pearson Education Limited 1995, 2005
4. SQL:2008  User-defined types (UDTs == ADTs of postquel)  (4)

- structured UDT definition consists of
  - Attribute definitions
  - Routine declarations (methods) etc.

```
CREATE TYPE PersonType AS (  ..........)
     INSTANCE METHOD age() RETURNS INTEGER;
```

```
CREATE INSTANCE METHOD age ()  
RETURNS INTEGER FOR PersonType  
BEGIN  
    RETURN /* calculated from SELF.dateOfBirth*/
END;
```
4. SQL:2008  UDTs – Supertypes & subtypes: Inheritance Example

CREATE TYPE StaffType UNDER PersonType AS (  
  staffNo StaffNoType,
  position VARCHAR(10) DEFAULT ‘Assistant’,
  salary DECIMAL(7, 2),
  branchNo CHAR(4)  
)
INSTANTIABLE
NOT FINAL
INSTANCE METHOD isManager()
  RETURNS BOOLEAN;

**Note: Multiple inheritance is not supported ***
4. SQL:2008  Ref types & object identity

• e.g. of structured types:

```
CREATE TYPE PersonType AS (
    dateOfBirth DATE ....... )
INSTANTIABLE
NOT FINAL
REF IS SYSTEM GENERATED
INSTANCE METHOD age() RETURNS INTEGER; ...
```

• Reference type
  • uniquely identifies a row within a table
  • Similar to OID of OODBMSs

System generates references

same meaning as:
OODBMS OID
4. SQL:2008 Creation of typed (UDT) tables

CREATE TABLE Person OF PersonType
REF IS personID SYSTEM GENERATED;

- Rows are objects
- One column for every attribute in the UDT
- **Cannot** add any columns to this table

<table>
<thead>
<tr>
<th>personID</th>
<th>dateOfBirth</th>
<th>fName</th>
<th>lName</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1/1/1965</td>
<td>John</td>
<td>Smith</td>
</tr>
</tbody>
</table>
4. SQL:2008 Reference types & object identity

- **e.g.**
  ```
  CREATE TABLE PropertyForRent (  
    propertyNo PropertyNumber NOT NULL,  
    street StreetType, .....  
    personID REF (PersonType) SCOPE PERSON )  
  REFERENCES ARE CHECKED ON DELETE CASCADE,  
  PRIMARY KEY (propertyNo) );
  ```

- Replace complex join definitions in queries with much simpler path expressions e.g.

- **INSTEAD OF:**
  ```
  SELECT P.street, S.fname, S.lname  
  FROM propertyForRent P, Person S WHERE P.personID = S.personID
  ```

- **USE THE DEREFERENCING OPERATOR:**
  ```
  SELECT P.street, P.personID->fname, P.personID->fname,  
  FROM PropertyForRent P;
  ```

This is a JOIN

© Pearson Education Limited 1995, 2005

This is a JOIN
5. Postgres, SQL:2008 & PostgreSQL

- Originally postgres, renamed PostgreSQL in 1996
  - query language postquel was replaced with SQL

- Uses a client/server model. A session consists:
- A server process (called postgres)
  - manages the database files
  - accepts connections to the database from client applications,
  - performs database actions on behalf of the clients.

- Users’ client (frontend) applications
  - make requests for database operations.

- Uses SQL + extensions for OO as defined in SQL:2008 + other postgres specific features
6. Using PostgreSQL in Windows

- pgAdminIII
- or from Windows command-line
6. PostgreSQL – Data types

1. Numeric Types
2. Monetary Types
3. Character Types
4. Binary Data Types
5. Date/Time Types
6. Boolean Type
7. **Enumerated Types**
8. Network Address Types
9. Bit String Types

10. **Geometric Types**
    - Points
    - Line Segments,
    - Boxes, Paths
    - Polygons
    - Circles

11. Text Search Types
12. XML Type
13. **Composite Types**
14. Arrays
15. Object Identifier Types
16. Pseudo-Types
6. PostgreSQL – user defined types (1)

- **Enumerated types**
  - similar to HLL enum types e.g.:
    ```sql
    CREATE TYPE mood AS ENUM ('sad', 'ok', 'happy');
    ```

- **e.g. of Enum Usage**
  ```sql
  CREATE TABLE person (
    name text,  
    current_mood mood
  );
  ```
  ```sql
  INSERT INTO person
  VALUES ('Moe', 'happy');
  ```
  ```sql
  SELECT * FROM person WHERE current_mood = 'happy';
  ```
6. PostgreSQL – user defined types (2a)

- **Composite types** similar to HLL record / struct types e.g.:

  ```sql
  CREATE TYPE inventoryItem AS (
    name text,
    supplierID integer,
    price numeric);
  ```

- e.g. of usage of composite type

  ```sql
  CREATE TABLE StockOnHand (  
    item inventoryItem,  
    quantity integer
 );

  INSERT INTO StockOnHand VALUES
  ( ROW ('gismo', 42, 1.99), 1000);
  ```
6. PostgreSQL – user defined types (2b)

- **Composite types contnd.**
  
  ```sql
  CREATE TYPE inventoryItem AS (
      name text, supplierID integer, price numeric);
  ```

- e.g. of usage of composite type
  
  ```sql
  CREATE TABLE StockOnHand (
      item inventoryItem, quantity integer);
  ```

  ```sql
  SELECT (StockOnHand.item).name, 
         (StockOnHand.item).price, 
         StockOnHand.quantity 
  FROM StockOnHand 
  WHERE (StockOnHand.item).price > 9.99;
  ```