Wykład IX

PERSPEKTYWY

W tradycyjnych systemach relacyjnych baz danych podstawowe cele stosowania perspektyw są:

- autoryzacja dostępu danych
- ułatwienie dostępu do danych
- możliwość prezentowania tych samych danych w różny sposób
- logiczna niezależność danych
- możliwość wprowadzenia dodatkowego poziomu ograniczeń integralnościowych

Perspektywy zmaterializowane pozwalają dodatkowo na:

- integrację danych pochodzących z rozproszonych źródeł danych oraz uniezależnienie użytkowników od konieczności dostępu do rozproszonych danych źródłowych
- materializowanie częściowych wyników najczęściej wykonywanych zapytań OLAP

Koncepcje perspektywy obiektowej

Komercyjne systemy zarządzania obiektowymi bazami danych np.O2, Poet, VERSANT nie wspierają mechanizmu perspektyw. Natomiast istnieje kilka systemów prototypowych, w których zaimplementowano ten mechanizm. Ponieważ nie istnieje jednak standard definiujący perspektywę obiektową trudno mówić o jednoznacznej definicji takiej perspektywy.

W systemach prototypowych i opracowaniach naukowych można wyróżnić cztery podstawowe podejścia do definicji perspektywy obiektowej:

- zapytanie (podobnie jak perspektywę relacyjną) operujące na klasach. Takie rozwiązanie ma jednak tę wadę, że złożona struktura klas bazowych zostaje „ spłaszczona „ w zapytaniu.
- funkcja składowana, której wynikiem działania jest kolekcja obiektów wyznaczonych przez tę funkcję. Nie jest możliwa restrukturyzacja danych źródłowych.
- klasa wirtualna
- schemat złożony z wielu klas wirtualnych połączonych związkami dziedziczenia i kompozycji.

W obu tych koncepcjach (klasa wirtualna i schemat) definicja perspektywy składa się z dwóch elementów:

- definicji struktury klasy wirtualnej
- zapytania lub metody wyznaczającej obiekty dostępne za pomocą tej klasy, tj. perspektywy.

Wykład IX

Klasa wirtualna jest definiowana w oparciu o tzw. klasy bazowe. Zapytanie jest formułowane w obiektowym języku zapytań OQL, składnią i funkcjonalnością przypominającym język SQL, lecz posiadającym większą funkcjonalność. Obiekty dostępne za pomocą perspektywy mogą być również wyznaczane w sposób proceduralny za pomocą obiektowego języka wspieranego przez bazę danych.

Perspektywy relacyjno-obiektowe

umożliwiają odwzorowanie relacyjnego modelu danych w model obiektowy.

- ta własność umożliwia pracę starych aplikacji relacyjnych na strukturach danych modelu obiektowego.
- perspektywy obiektowe umożliwiają budowanie obiektowych aplikacji pracujących na relacyjnych bazach danych.

przykład:

Mamy zgromadzone dane w tabelach relacyjnych o strukturach

PRACOWNICY (Numer, Nazwisko, Etat, . . . , Id_zesp**)**
ZESPOLY (Id_zesp, Nazwa, Adres **)**

/* *definicja typu obiektowego* Typ_pracownik *reprezentujący struktury obiektowe dla danych z tabeli* Pracownicy */

```
CREATE OR REPLACE TYPE Typ_pracownik AS OBJECT
(Numer NUMBER(4), Nazwisko VARCHAR2(15), Etat VARCHAR2(15))
/
```

/ definicja typu* Kolekcja_pracowników */*

```
CREATE OR REPLACE TYPE Kolekcja_pracownikow AS TABLE OF
Typ_pracownik
/
```

/ definicja typu obiektowego* Typ_zespol *wraz z metodą* Liczba_prac
reprezentujący struktury obiektowe dla danych z tabeli Zespoly,
wykorzystanie wcześniej zdefiniowanej kolekcji typu zagnieżdżona tabela */

```
CREATE OR REPLACE TYPE Typ_zespol AS OBJECT
(Id_zesp NUMBER(2),
Nazwa VARCHAR(20),
Lista_prac Kolekcja_pracownikow,
MEMBER FUNCTION Liczba_prac RETURN NUMBER)
/
```

Wykład IX

```
CREATE OR REPLACE TYPE BODY Typ_zespol AS
MEMBER FUNCTION Liczba_prac RETURN NUMBER
IS
BEGIN
  RETURN Lista_prac.COUNT( ) ;
END Liczba_prac ;
END;
/
```

składnia perspektywy relacyjno_obiektowej:

```
CREATE [OR REPLACE] VIEW <nazwa> OF <typ_obiektowy>
[ WITH OBJECT IDENTIFIER <kolumna> ]
        AS SELECT . . . ;
```

/ definicja perpektywy* Zespoly_prac

```
CREATE OR REPLACE VIEW Zespoly_prac OF Typ_zespol
WITH OBJECT IDENTIFIER (Id_zesp)
AS
SELECT Id_zesp, Nazwa,
        CAST (MULTISET (SELECT Typ_pracownik (Numer, Nazwisko, Etat)
FROM Pracownicy WHERE Id_zesp=z.Id_zesp ) AS Kolekcja_pracownikow )
        FROM Zespoly z;
```

/ przykładowe zapytanie do perspektywy* Zespoly_prac */*

```
SELECT z.Nazwa, z.liczba_prac(), z.Lista_prac FROM Zespoly_prac z;
```

/ wynik powyższego zapytania */*

```
NAZWA                Z.LICZBA_PRAC()   LISTA_PRAC
-------------------- --------------- --------------------------------------------------------------------------------
Administracja                     2
KOLEKCJA_PRACOWNIKOW(TYP_PRACOWNIK(1000,'Lech','Dyrektor'),TYP_PRACOWNIK(1080,'Koli
berek','Sekretarka'))
Bazy danych                       4
KOLEKCJA_PRACOWNIKOW(TYP_PRACOWNIK(1010,'Podgajny','Profesor'),TYP_PRACOWNIK(1040,'
Rus','Adiunkt'),TYP_PRACOWNIK(1070,'Muszyński','Adiunkt'),TYP_PRACOWNIK(1060,'Misiecki','Asysten
t'))
Sieci komputerowe         4
KOLEKCJA_PRACOWNIKOW(TYP_PRACOWNIK(1020,'Delecki','Profesor'),TYP_PRACOWNIK(1030,'M
aleja','Adiunkt'),TYP_PRACOWNIK(1100,'Warski','Asystent'),TYP_PRACOWNIK(1110,'Rajski','Stażysta'))
Systemy operacyjne        3
KOLEKCJA_PRACOWNIKOW(TYP_PRACOWNIK(1050,'Lubicz','Adiunkt'),TYP_PRACOWNIK(1120,'Ork
a','Asystent'),TYP_PRACOWNIK(1130,'Kolski','Stażysta'))
Multimedia                        0 KOLEKCJA_PRACOWNIKOW()
Audyt                             0 KOLEKCJA_PRACOWNIKOW()
```

Wykład IX

Operacje masowe

Sposób przekazywania danych pomiędzy kolekcjami a bazą danych (ang. *bulk bind*).

- sprawiają, że operujemy wieloma zmiennymi jednocześnie jak jednostką.
- operacje te są znacznie wydajniejsze od wykonywanych w pętli pojedynczych odczytów.

Instrukcja **FORALL**

- służy do wykonywania instrukcji DML na podstawie zawartości kolekcji
- umożliwia znaczne skrócenie zapisu

FORALL j **IN** <*tablica*>.**FIRST** . . <*tablica*>.**LAST**
{ [DELETE FROM <*tabela*> WHERE <*kolumna*> = <*tablica(j)*>**] |**
[INSERT INTO <*tabela*>(<*kolumny*>) **VALUES** <*tablica(j)*>**] }**

przykłady: */* z http://docs.oracle.com/cd/E11882_01/appdev.112/e25519/tuning.htm#LNPLS882 */*

```
DROP TABLE pracownicy_temp;
CREATE TABLE pracownicy_temp AS SELECT * FROM pracownicy;

DECLARE
 TYPE NumList IS VARRAY(20) OF NUMBER;
 Zesp  NumList := NumList(10, 30, 70);  -- id zespolu
BEGIN
 FORALL i IN Zesp.FIRST..Zesp.LAST
  DELETE FROM pracownicy_temp WHERE id_zesp = Zesp(i);
END;
/
```

```
DROP TABLE parts1;

CREATE TABLE parts1 (
  pnum INTEGER,
  pname VARCHAR2(15)
);

DROP TABLE parts2;

CREATE TABLE parts2 (
  pnum INTEGER,
  pname VARCHAR2(15)
);

DECLARE
  TYPE NumTab IS TABLE OF parts1.pnum%TYPE INDEX BY PLS_INTEGER;
  TYPE NameTab IS TABLE OF parts1.pname%TYPE INDEX BY PLS_INTEGER;
  pnums   NumTab;
  pnames  NameTab;
  iterations  CONSTANT PLS_INTEGER := 50000;
  t1  INTEGER;
  t2  INTEGER;
  t3  INTEGER;
```

```
BEGIN
  FOR j IN 1..iterations LOOP  -- populate collections
    pnums(j) := j;
    pnames(j) := 'Part No. ' || TO_CHAR(j);
  END LOOP;

  t1 := DBMS_UTILITY.get_time;

  FOR i IN 1..iterations LOOP
    INSERT INTO parts1 (pnum, pname)
    VALUES (pnums(i), pnames(i));
  END LOOP;

  t2 := DBMS_UTILITY.get_time;

  FORALL i IN 1..iterations
    INSERT INTO parts2 (pnum, pname)
    VALUES (pnums(i), pnames(i));

  t3 := DBMS_UTILITY.get_time;

  DBMS_OUTPUT.PUT_LINE('Execution Time (secs)');
  DBMS_OUTPUT.PUT_LINE('---------------------');
  DBMS_OUTPUT.PUT_LINE('FOR LOOP: ' || TO_CHAR((t2 - t1)/100));
  DBMS_OUTPUT.PUT_LINE('FORALL:   ' || TO_CHAR((t3 - t2)/100));
  COMMIT;
END;
```

```
/* odpowiedź*/
Execution Time (secs)
---------------------
FOR LOOP: 2.16
FORALL:   .11
```

klauzula **BULK COLLECT**

- służy do odczytu wyniku zapytania dotyczącego kolekcji w jednej operacji
- może być użyta w klauzuli SELECT
- **SELECT . . . BULK COLLECT INTO** <lista_tablic >
- może być stosowana w metodzie kursora **FETCH**
- może być użyta w klauzuli **RETURNING**
- nie można jej zagnieździć w pętli **FORALL**

przykład:

```
CREATE OR REPLACE PROCEDURE Fast_proc
IS
TYPE TObjectTab IS TABLE OF ALL_OBJECTS%ROWTYPE;
ObjectTab$ TObjectTab;
BEGIN
 SELECT * BULK COLLECT INTO ObjectTab$ FROM ALL_OBJECTS;
 FORALL j IN ObjectTab$.FIRST..ObjectTab$.LAST
   INSERT INTO t1 VALUES ObjectTab$( j );
END Fast_proc;
```

w wykładzie wykorzystano http://www.ploug.org.pl/konf_00/pdf/wrembel.pdf