|  |
| --- |
|  |
|  |  |
|  |  |
|  |  |
|  | **Typy referencyjne (REF, DEREF)**  create type Osoba as object( imię varchar(10), nazwisko varchar(20)); create type Obraz as object( utworzony Date, autor REF Osoba); create table Obrazy of Obraz; create table Osoby of Osoba; insert into Osoby values ('Jan','Tarzan'); insert into Obrazy select '1-04-2006', ref(o) from Osoby o where nazwisko = 'Tarzan';  W obiektowo-relacyjnych bazach danych związki między obiektami są modelowane za pomocą nowego systemowego typu danych – typu referencyjnego. Wartościami referencyjnego typu danych są identyfikatory obiektów. Referencyjny typ danych służy do przechowywania identyfikatorów innych obiektów powiązanych z danym obiektem. Mechanizm referencyjnych typów danych w przeciwieństwie do związków w czysto obiektowym modelu danych reprezentuje jedynie związki jednokierunkowe. Modelowanie związków dwukierunkowych wymaga odrębnego użycia dwóch różnych referencji. Powyżej pokazano przykład jednokierunkowego związki jednokrotnego łączącego obiekty typu Obraz z wystąpieniami typu Osoba. Definicja typy Obraz obejmuje atrybut autor zdefiniowany na referencyjnym typie danych. Wartościami tego atrybutu będą identyfikatory obiektów typu Osoba.  DEREF - Aby uzyskać dostęp do obiektu wskazywanego przez referencję należy wykonać  operację dereferencji. Dereferencji można dokonać w sposób jawny, bądź niejawny.  SELECT NAZWISKO,DEREF(KONTAKT).NUMER AS NUMER FROM PRACOWNICY\_O P;  SELECT NAZWISKO,P.KONTAKT.NUMER AS NUMER FROM PRACOWNICY\_O P;  Pierwsze z powyższych zapytań wykonuje de referencję w sposób jawny, natomiast drugie wykorzystuje dereferencję niejawną. Operator DEREF jest operatorem dereferencji. Jako parametr przyjmuje on referencję na obiekt, a jego wartością jest wskazywany obiekt. Atrybut KONTAKT stanowi referencję na obiekt typu TELEFON, dlatego też, dla każdego pracownika, operator DEREF zwraca wskazywany przez referencję KONTAKT telefon. Mając dany obiekt typu TELEFON można w łatwy sposób, za pomocą operatora kropkowego, odczytać jego numer („.NUMER”). W niektórych sytuacjach operator DEREF można pominąć. Jeżeli nawigujemy za pomocą operatora kropkowego, poprzez referencję, do wartości atrybutów wskazywanego obiektu, to wykonywana jest automatyczna dereferencja i operator DEREF nie jest tutaj potrzebny. Operator DEREF znajduje zastosowanie właściwie jedynie w sytuacjach, kiedy chcemy uzyskać w wyniku zapytania całe obiekty, na które mamy dane referencje. Należy tutaj jeszcze zauważyć, że nazwę atrybutu przechowującego referencje poprzedzono aliasem tabeli obiektowej.  W sytuacji gdy korzystamy z niejawnej dereferencji, jest to obowiązkowe. |
|  | **Porównanie możliwości związanych z obiektowością wybranej bazy danych z jej relacyjnymi możliwościami**   |  |  |  | | --- | --- | --- | | **cecha** | **RDBMS** | **ORDBMS** | | Standard | SQL2 (ANSI X3H2) | SQL3/4 (w opracowaniu) | | współpraca z obiektowymi językami programowania | słaba, programiści musza dostosowywać program obiektowy do potrzeb bazy | graniczona głownie do nowych typów danych | | użytkowanie | łatwa do zrozumienia struktura, wiele narzędzi dla użytkowników | zapewnia niezależność danych od aplikacji, trudno odzwierciedlać złożone powiązania | | programowanie | zapewnia niezależność danych od aplikacji, trudno odzwierciedlać złożone powiązania | zapewnia niezależność danych od aplikacji, trudno odzwierciedlać złożone powiązania | | rozszerzalność | brak | głownie ograniczona do nowych typów danych | | złożone dane i powiązania miedzy nimi | trudne do zrealizowania | trudne do zrealizowania | | "dojrzałość" systemów | bardzo dojrzale, dobrze poznana i przetestowana metodologia, liczne implementacje, stabilność na rynku | niedojrzałe, rozszerzenia są nowe, wciąż ewoluujące i stosunkowo słabo poznane | | możliwość utrzymania się na rynku | przewidywana dla dużych przedsiębiorstw obecnych na rynku | przewidywana dla przedsiębiorstw znanych z RDBMS, dołączają się nowi |   **RDBMS Relacyjne bazy danych**  Zalety:   * oparte na solidnych podstawach teoretycznych (zainteresowanie świata nauki, a nie tylko biznesu) * stabilna pozycja na rynku * optymalizacja zapytań   Wady:   * z góry ustalony konstruktor, brak złożonych obiektów * brak środków hermetyzacji i modularyzacji (brak oddzielenia implementacji od specyfikacji) * brak środków do przechowywania informacji proceduralnych * niezgodność impedancji * niezgodność modelu pojęciowego z modelem implementacyjnym   **ORDBMS Obiektowo-relacyjne bazy danych**  Zalety:   * przystosowanie do multimediow (duże obiekty BLOB, CLOB i dane binarne) * dane przestrzenne (spatial) * abstrakcyjne typy danych (ADT) * metody (funkcje i procedury) definiowane przez użytkownika w rożnych językach (C++, VisualBasic, Java) * kolekcje (zbiory, wielozbiory, sekwencje, tablice zagnieżdżone, tablice o zmiennej długości) * typy referencyjne * przeciążanie funkcji * optymalizacja zapytań   Wady:   * wciąż nie uniknięto wielu błędów modelu relacyjnego (np. niezgodności impedancji) * brak perspektyw na przyszłość * produkt hybrydowy "dwa w jednym" (redundancja kodu i danych) * brak bazy intelektualnej * zmiany wprowadzane ad hoc (kumulowanie błędów koncepcyjnych) |