OPRACOWANIE DO KOLOKWIUM ZALICZENIOWEGO
BAZY DANYCH 2

1. Charakterystyka relacyjnych baz danych i hurtowni danych.

Hurtownia danych – to trwała analityczna baza danych, która jest podstawą systemu wspomagania decyzji. Jest ona projektowana dla dużej liczby danych stałych (historycznych). Zapewnia ona stały dostęp do danych w trakcie podejmowania decyzji. Jest to przedsięwzięcie rozwijające się, które zapewnia dostęp do właściwych danych właściwym użytkownikom we właściwym czasie. Często tworzone są hurtownie danych, które są zorientowane na konkretny temat np. analiza sprzedaży. Wtedy takie hurtownie danych nazywamy hurtowniami tematycznymi.

Centrum hurtowni danych stanowi zbiór danych:

- **zintegrowanych**, czyli wyrażonych we wspólnym języku na poziomie pojęć i schematu bazy danych;
- **nieulotnych**, czyli przeznaczonych do długiego przechowywania mających wartość archiwalną w zasadzie niepodlegającą zmianom niż dodawanie nowych porcji danych;
- **ukierunkowanych**, czyli zorganizowanych w sposób mający ułatwić zadania analityczne zoptymalizowanych pod kątem przetwarzania konkretnych rodzajów zapytań.

Analityczna baza danych, czyli hurtownia danych ma za zadanie:

- przyjmować i archiwizować duże ilości danych;
- dokonywać wstępnych analiz poprzez mechanizm migawek;
- być bazą danych tylko do odczytu uniemożliwiając modyfikację danych;
- zapewniać stały dostęp do informacji w procesie podejmowania decyzji

Bazy danych – system o przetwarzaniu operacyjnym. Najczęściej obsługują one operacje *online*, a więc można stwierdzić, że są to operacje czasu rzeczywistego. System operacyjny oparty o klasyczną bazę danych nie posiada możliwości dokonywania analiz porównawczych lub posiada je w bardzo ograniczonym zakresie. Próba analizy przeprowadzona z użyciem bazy danych kończy się zaabsorbowaniem wszystkich zasobów systemu operacyjnego co może doprowadzić do czasowego paraliżu systemu informatycznego firmy. Na podstawie powyższych rozważań można wysnuć wniosek, że dane przechowywane w bazach danych są nietrwałe.

Od systemu operacyjnego opartego na bazie danych wymaga się:

- przyjmowania danych i bieżącego wprowadzania zmian w danych zawartych w bazie danych;
- śledzenia operacji;
- przedstawiania raportów;
- utrzymywania spójności danych;
- szybkiego wykonywania operacji.

2. Modele danych – reprezentacja danych w hurtowni.

Dane w hurtowni danych można reprezentować i przechowywać w oparciu o model relacyjny zwany również jako ROLAP i wielowymiarowy zwanym również MOLAP lub MDOLAP. Istnieje również sposób reprezentacji nazywany jako sposób hybrydowy – HOLAP. Oznacza to, że dane reprezentuje się częściowo w modelu ROLAP, a częściowo w MOLAP.

ROLAP – magazyn danych w technologii ROLAP jest implementowany w postaci tabel, których schemat posiada najczęściej strukturę gwiazdy, płatka śniegu, konstelacji faktów, lub strukturę gwiazda-płatek śniegu. W przypadku implementacji ROLAP zalecane jest zdefiniowanie dodatkowych logicznych obiektów opisujących ww. schematy. Obiektami tymi są: wymiar, hierarchia wymiaru i zależności funkcyjne.

MOLAP/MDOLAP – hurtowania danych zaprojektowana w technologii MOLAP do przechowywania danych wykorzystuje wielowymiarowe tablice. Tablice te zawierają wstępnie przetworzone dane pochodzące z wielu źródeł. W systemie Oracle dane wielowymiarowe są przechowywane w tzw. przestrzeni analitycznej.

Definiowanie tej przestrzeni i zarządzanie nią realizuje się poprzez specjalne oprogramowanie np. *Warehouse Builder* lub z poziomu SQL wykorzystując do tego pakiety systemowe takie jak zbiór pakietów rodziny CWM2, w skład którego wchodzą: DBMS_AWM, DBMS_AW oraz DBMS_AW_UTILITIES.

HOLAP - stara się łączyć zalety ROLAP i MOLAP. Wykorzystane są tutaj jednocześnie wielowymiarowe oraz relacyjne bazy danych. Zapytania szczegółowe są wykonywane wolniej, ponieważ działają na strukturze ROLAP. Dane w HOLAP maja zwykle mniejszy rozmiar niż dane w MOLAP. Struktury HOLAP są przeznaczone dla potrzeb szybkiego dostępu do agregacji bazujących na dużych zbiorach danych.

3. Schematy danych modelu relacyjnego ROLAP.

W każdym ze schematów mamy do czynienia z tabelami wymiarów oraz tabelą faktów – ilość tabel faktów może być większa od jednej w zależności od opisywanego schematu danych.

Tabela faktów – przechowuje dane znormalizowane do trzeciej postaci normalnej. Składa się z dwóch typów kolumn:

- kolumny zawierające wartości liczbowe o danym fakcie tzw. **miary**;
- kolumny z kluczami obcymi pochodzącymi z tabel wymiarów.

Tabele wymiarów – są zwykle zdenormalizowane. Tabele wymiarów są strukturami złożonymi z jednej lub więcej hierarchii, które służą do kategoryzowania danych. Oprócz kluczy głównych zawierają pola opisującymi dany wymiar.

Miara – zawiera wartości liczbowe opisujące dany fakt.

Wymiar – zawiera wartości opisowe danego faktu przechowywane w atrybutach wymiaru. Wymiary są informacjami referencyjnymi i określają kontekst analiz miar.

Schemat gwiazdy – układ gwiaździsty jest specyficznym rodzajem struktury wykorzystywanym przy przetwarzaniu analitycznym. Nazwa schematu gwiazdy wzięła się z wyglądu schematu danych, w którym w centralnym miejscu znajduje się tabela faktów otoczona przez tabelę wymiarów.

Korzyści wynikające z korzystania ze schematu gwiazdy:

- szybki czas odpowiedzi na zapytania;
- prosta i przejrzysta struktura bazy danych umożliwiająca lepsze jej wykorzystanie;
- struktura bazy pozwala na łatwe i przejrzyste poznanie metadanych przez projektanta oraz późniejszego użytkownika;
- schemat gwieździsty powiększa ilość narzędzi, które mogą wspomóc tworzenie i korzystanie z hurtowni danych;
- stosunkowo długi czas ładowania danych do tabel wymiarów ze względu na denormalizację.

Schemat płatka śniegu – odmiana schematu gwieździstego, która zakłada przechowywanie wszystkich informacji dotyczących wymiarów w trzeciej postaci normalnej (3NF), a pozostawia bez zmian strukturę tabeli faktów. Znormalizowanie do trzeciej postaci normalnej oznacza, że każdy wymiar może mieć kilka własnych wymiarów. Schemat ten używany jest przede wszystkim w sytuacjach, w których mamy do czynienia z bardzo złożonymi wymiarami. Cechy charakterystyczne schematu płatka śniegu:

- spadek wydajności zapytań w porównaniu ze schematem gwiazdy, ze względu na większą liczbę połączeń tabel;
- struktura łatwiejsza do modyfikacji;
- krótki czas ładowania danych do tabel (normalizacja, czyli mniejszy rozmiar tabel poziomów, a tym samym oszczędność miejsca);
- wykorzystywany rzadziej niż schemat gwiazdy, gdyż efektywność zapytań jest ważniejsza od efektywności ładowania danych do tabel wymiarów.

Schemat konstelacji faktów – stanowi rozwiązanie pośrednie pomiędzy schematem gwiazdy, a schematem płatka śniegu. W schemacie konstelacji faktów część tabel wymiarów jest znormalizowanych, a część zdenormalizowanych oraz zwykle występuje więcej niż jedna tabela faktów. Ponadto tabele faktów powiązane są ze sobą w relacji 1:1 lub 1:N oraz mogą współdzielić te same tabele wymiarów.

4. **Sposób ładowania danych do hurtowni danych.**

Sposób I:
Program SQL*Loader ładuje dane z zewnętrznych plików do tabel w bazie danych Oracle. Wykorzystuje dwa zasadnicze pliki: plik danych zawierający informacje, które mają być załadowane, oraz plik sterujący z informacjami o formacie danych, rekordów i pól w pliku danych, porządku, w jakim mają zostać załadowane. Teoretycznie informacje z pliku sterującego można by było umieścić w samym pliku danych, jednakże w praktyce się tego nie używa z uwagi na wielkokrotne wykorzystywanie pliku sterującego.

Uruchomienie programu powoduje automatyczne utworzenie pliku dziennika oraz pliku "złych" danych (plik w formacie .bad). W pliku dziennika zapisywany jest status ładowania danych, na przykład liczba przetworzonych i zatwierdzonych wierszy. Plik "złych" danych zawiera z kolei wszystkie wiersze, których załadowanie nie powiodło się z powodu błędnych danych, jak na przykład powtarzające się wartości klucza głównego. Plik sterujący tworzy się na poziomie systemu operacyjnego np. za pomocą notatnika w systemie Windows.

Przykład zawartości pliku zawierającego dane do ładowania (SALON.csv):

```
1,Sanimex,1 Maja,67,48,93-194,Krotoszyn,75
2,Anette,Krakowsa,22,68,21-27,Mielec,88
3,la Bella,Zgoda,78,26,06-881,Szczecin,87
4,Tina,Domaszowska,63,24,80-518,Bedzin,58
5,Sekret urody,Sosnowa,12,95,74-293,Poznan,27
6,la Bella,Graniczna,97,45,56-213,Plock,26
```

Przykład zawartości pliku sterującego (SALON.ctl):

```
LOAD DATA
INFILE SALON.csv
BADFILE SALON.bad
APPEND
INTO TABLE SALON
FIELDS TERMINATED BY ','
(ID_SALONU,NAZWA,ULICA,NR_BUDYNKU,NR_LOKALU,KOD_POCZTOWY,MIEJSCOWOSC,WLASCICIEL_ID_WLASCICIELA)
```

Opis poszczególnych słów kluczowych pliku sterującego:

INFILE – oznacza plik, z którego mają zostać załadowane dane
BADFILE – plik w którym będą informacje o tym co się nie udało załadować
APPEND – oznacza dodanie wierszy do tabeli
INTO – określa tabelę do której chcemy załadować dane
FIELDS TERMINATED BY – separator jakim są oddzielone kolumny w pliku z danymi (należy pamiętać aby wybrać taki separator, który nie występuje w ładowanych danych)

Uruchomienie SQL*Loadera ze wskazaniem do pliku sterującego:

```
SQLLDR HR/HR CONTROL='SALON.ctl'
```

Sposób II:
Tabele zewnętrze stanowią cenny element podczas tworzenia tabeli danych. Umożliwiają one definiowanie tabel, które nie są tabelami tradycyjnymi, czyli takimi, które są zarządzane przez serwer bazy danych. Tabele zewnętrze są przechowywane w zwykłych plikach systemu plików. Pozwala to na zadeklarowanie pliku danych jako tabeli i następnie pobieranie z niego danych.

5. Rozszerzenia SQL w kontekście hurtowni danych – rozszerzenia operatorów grupowania.

- **Operator ROLLUP** – klauzula ROLLUP rozszerza klauzulę GROUP BY o zwracanie wiersza zawierającego podsumowanie częściowe każdej grupy wierszy oraz podsumowanie całkowite wszystkich grup. Klauzula GROUP BY ROLLUP służy do grupowania wierszy w bloki z tą samą wartością we wskazanej kolumnie. Najczęściej używana z funkcjami agregującymi np. SUM().

  Przykład zapytania – przesyłanie jednej kolumny do klauzuli ROLLUP:

  ```
  SELECT ID_PRACOWNIKA, SUM(PENSJA) FROM PRACOWNIK
  GROUP BY ROLLUP(ID_PRACOWNIKA);
  ```

  Do klauzuli ROLLUP możemy przesłać wiele kolumn. W takim przypadku wiersze są grupowane w bloki z tymi samymi wartościami kolumn.

  Przykład zapytania – przesyłanie dwóch kolumn do klauzuli ROLLUP:

  ```
  SELECT ID_PRACOWNIKA, ID_STANOWISKA, SUM(PENSJA) FROM PRACOWNIK
  GROUP BY ROLLUP(ID_PRACOWNIKA, ID_STANOWISKA);
  ```

- **Operator CUBE** – klauzula ROLLUP rozszerza klauzulę GROUP BY o zwracanie wierszy zawierających podsumowania częściowe dla wszystkich kombinacji kolumn oraz wiersza zawierającego podsumowanie całkowite.

  Przykład zapytania:

  ```
  SELECT ID_PRACOWNIKA, ID_STANOWISKA, SUM(PENSJA) FROM PRACOWNIK
  GROUP BY CUBE(ID_PRACOWNIKA, ID_STANOWISKA);
  ```

  W powyższym przypadku wynagrodzenia są sumowane według ID_PRACOWNIKA oraz ID_STANOWISKA. Klauzula CUBE w tym przypadku zwróci wiersz z sumą wynagrodzeń dla każdego pracownika, a także sumy wszystkich wynagrodzeń dla każdego stanowiska. Na samym końcu zostanie zwrócone podsumowanie całkowite wynagrodzeń.

- **Operator GROUPING SETS** – pozwala uzyskać same wiersze podsumowań częściowych. Ponadto umożliwia on realizację wielu schematów grupowania w jednym wywołaniu instrukcji SELECT. Argumentem operatora jest lista wielu zestawów kolumn grupujących.

  Przykład zapytania:

  ```
  SELECT ID_PRACOWNIKA, ID_STANOWISKA, ID_KIEROWNIKA, SUM(PENSJA) FROM PRACOWNIK
  GROUP BY GROUPING SETS((ID_PRACOWNIKA, ID_STANOWISKA),(ID_KIEROWNIKA,ID_PRACOWNIKA));
  ```

6. Ruchome okna obliczeniowe.

Wartości funkcji grupowych mogą być również wyznaczane w oparciu o grupy wierszy przesuwających się wraz z wierszem bieżącym. Grupy takie nazywane są oknami i mogą być definiowane przy pomocy wyrażeń ROWS lub RANGE. Wyrażenia ROWS definiuje tak zwane okno fizyczne, czyli rozmiar okna jest określone liczbą wierszy. Wyrażenie RANGE definiuje tak zwane okno logiczne – rozmiar jest określany warunkiem selekcji wierszy.

Ogólny schemat budowania okna obliczeniowego:

```
FUNKCJA() OVER ([PARTITION BY kol1] ORDER BY kol2 [DESC] RANGE/ROWS BETWEEN wyr1 AND wyr2)
```

Gdzie:
FUNKCJA() – funkcja agregująca np. SUM(), AVG();
kol1 – kolumna grupująca wiersze w grupy, wewnątrz których przesuwa się okno. Jeśli nie wystąpi fraza PARTITION BY oznacza to, że okno przesuwa się przez całą tabelę;
kol2 – kolumna porządkująca wiersze wewnątrz grupy;
wyr1 – określa położenie początku okna. Może przyjmować następująca postać:

- UNBOUNDED PRECEDING – okno rozpoczyna się od pierwszego rekordu grupy;
- CURRENT ROW – okno rozpoczyna się od wartości pobranej z aktualnego rekordu;
- n PRECEDING – okno rozpoczyna się od rekordu posiadającego w kolumnie kol2 wartość o n-mniejszą niż rekord bieżący;
- INTERVAL 'n' DAYS|MONTHS|YEARS PRECEDING (**dotyczy okna logicznego RANGE**) – okno rozpoczyna się od rekordu posiadającego w kolumnie kol2 wartość o n-dni, miesięcy, lat mniejszą lub większą niż rekord bieżący.

wyr2– określa położenie końca okna. Może przyjmować następująca postać:

- UNBOUNDED FOLLOWING – okno kończy się od pierwszego rekordu grupy;
- CURRENT ROW – okno kończy się od wartości pobranej z aktualnego rekordu;
- n PRECEDING – okno kończy się od rekordu posiadającego w kolumnie kol2 wartość o n-mniejszą niż rekord bieżący;
- INTERVAL 'n' DAYS|MONTHS|YEARS FOLLOWING (**dotyczy okna logicznego RANGE**) – okno kończy się od rekordu posiadającego w kolumnie kol2 wartość o n-dni, miesięcy, lat mniejszą lub większą niż rekord bieżący.

Zapytania stosujące ruchome okna obliczeniowe mogą również posługiwać się funkcjami FIRST_VALUE() oraz LAST_VALUE(). Funkcje te zwracają wartości wybranej kolumny odpowiednio z pierwszego i ostatniego rekordu okna obliczeniowego. Ponadto istnieją funkcje, które wewnętrznie korzystają z ruchomego okna obliczeniowego. Są to funkcje LEAD() oraz LAG(). Nie wymagają one definiowania wyrażenia ROWS/RANGE.

Głównym zastosowaniem ruchomych okien obliczeniowych są zapytania analityczne obliczające sumy kumulacyjne, średnie kroczące oraz średnie centralne.

7. Partycje obliczeniowe.

Tradycyjne funkcje agregujące takie jak SUM(), AVG(), COUNT() obliczają swoje wartości dla grup wierszy, które są definiowane poprzez klauzulę GROUP BY w instrukcji SELECT. Dla każdej grupy funkcja agregująca zwraca pojedynczą wartość wynikową. Od wersji Oracle 9i możliwe jest wyznaczanie wartości funkcji grupowej oddzielnie dla każdego wiersza grupy, a nie tylko raz dla całej grupy. W tym celu należy zastąpić klauzulę GROUP BY wyrażeniem PARTITION BY.

Ogólny schemat budowania partycji obliczeniowej:

```
FUNKCJA() OVER (PARTITION BY kol/wyr)
```

Gdzie:
FUNKCJA() – funkcja agregująca np. SUM(), AVG();
kol/wyr – kolumna lub wyrażenie grupujące rekordy w celu wyliczenia wartości funkcji agregujących.

Wyrażenie PARTITION BY umożliwia prostszy zapis wielu zapytań analitycznych oraz niweluje konieczność stosowania podzapytań.

Przykładowa partycja obliczeniowa:

```
SELECT region, produkt, kwota, SUM(kwota) OVER (PARTITION BY region) AS SUMA,
ROUND(100*kwota/SUM(kwota) OVER (PARTITION BY region))) AS UDZIAŁ_%
FROM SPRZEDAZ;
```

Powyższe zapytanie wylicza udział procentowy sprzedaży danego produktu względem procentu ogólnej sprzedaży w danym regionie.

Celem zastąpienia wyrażenia `ROUND(100*kwota/SUM(kwota) OVER (PARTITION BY region)))` możemy posłużyć się funkcją RATIO_TO_REPORT(). Funkcja ta zwraca proporcję wartości wybranej kolumny lub wydarzenia w bieżącym rekordzie w stosunku do sumy tej kolumny w całej partycji obliczeniowej. Należy również wspomnieć, że grupowanie rekordów przy użyciu wyrażenia PARTITION BY nie wyklucza obecności klauzuli GROUP BY w zapytaniu. Wówczas grupowanie posiada charakter dwustopniowy.

8. Funkcje rankingowe i statystyczne.

Funkcje rankingowe umożliwiają wyznaczenie położenia wiersza w odniesieniu do pozostałych wierszy grupy względem wybranej funkcji porządkującej. Wyróżniamy następujące funkcje rankingowe: RANK, DENSE_RANK, PERCENT_RANK, ROW_NUMBER, NTILE oraz CUME_DIST.

Ogólna składnia budowania funkcji rankingowych:

```
FUNKCJA() OVER ([PARTITION BY kol1] ORDER BY kol2 [DESC] [NULLS FIRST | LAST])
```

Gdzie:
FUNKCJA() – nazwa funkcji rankingowej;
kol1 – kolumna lub wyrażenie grupujące rekordy uczestniczące w rankingu. Brak wyrażenia PARTITION BY oznacza, że rankingowi podlegają wszystkie rekordy w tabeli;
kol2 – kolumna lub wyrażenie porządkujące rekordy wewnątrz grupy;
DESC – określa porządek rankingu od największej do najmniejszej;
NULLS FIRST – powoduje, że wartości puste trafiają na początek rankingu;
NULLS LAST – powoduje, że wartości puste trafiają na koniec rankingu;

Funkcja **RANK** zwraca numer pozycji rankingowej rekordu, przy czym wystąpienie jednakowej pozycji rankingowej dla wielu rekordów powoduje powstanie przerw w numeracji.
Funkcja **DENSE_RANK** działa podobnie jak funkcja RANK, jednak nie powoduje powstawania przerw w numeracji.
Funkcja **CUME_DIST** zwraca wartość pozycji rankingowej wyrażoną liczbą z przedziału (0,1>. Liczba ta rozumiana jest jako procent rekordów poprzedzających dany rekord w rankingu z uwzględnieniem bieżącego rekordu.
Funkcja **PERCENT_RANK** stosuje podobne podejście lecz nie uwzględnia bieżącego rekordu.
Funkcja **NTILE** zwraca numer kolejnej n-tki rankingu, do której należy bieżący rekord.
Funkcja **ROW_NUMBER** wyznacza liczbę porządkową dla każdego rekordu.

Implementacja języka SQL zawiera funkcje regresji liniowej m.in. REGR_COUNT i REGR_SLOPE. Dzięki takim zapytaniom wyznacza się prosta regresji dla danych. Na jej podstawie analityk może np. oszacować sprzedaż na wybranym poziomie cenowym.

9. Dynamiczne strony WWW.

Dynamiczne strony WWW są to strony generowane dynamicznie na podstawie parametrów lub zawartości żądania. Zawartość tych stron tworzona jest na bieżąco na podstawie informacji z baz danych. Nie są przechowywane w całości w pliku lecz tworzone są dynamicznie na podstawie wypełnienia wzorca.

Technologie tworzenia dynamicznych stron WWW:

- **CGI** – znormalizowany interfejs umożliwiający użycie dowolnego języka programowania. Jego dużą zaletą jest prostota interfejsu. Do wad należą problemy z wydajnością oraz ograniczona przenośność programu.
- **Servlety** – są to programy napisane w języku Java (klasa potomna HTTPSevlet definiująca metody doPost() i doGet()). Servlety przechowywane są w postaci skompilowanej na serwerze aplikacyjnym. Do zalet należy przenośność oraz użycie języka Java zawierającego liczne biblioteki. Do wad możemy zaliczyć brak rozdzielenia warstwy prezentacji od warstwy logiki oraz kłopoty z wydajnością.
- **Moduły serwerowe** – są to przystawki do serwera WWW pozwalające na wykonanie programów CGI. Są pozbawione wad związanych z wydajnością CGI.
- **Server Pages** – są to strony w języku HTML lub XML zawierające statyczny szablon strony, wstawki w języku programowania i zmienne w dynamicznie generowanych miejscach dokumentu. Do tej technologii możemy zaliczyć: JSP (wykorzystanie języka Java), ASP (wykorzystanie języka C#, Delphi lub Python), PHP oraz PSP.

10. PSP – zastosowanie.

Wygodnym sposobem tworzenia procedur składowanych generujących tekstowe strony WWW, najczęściej w języku HTML, jest przygotowywanie ich w postaci dokumentów *PL/SQL Server Pages*, czyli PSP. Dokument PSP ma postać dokumentu HTML, w którym za pomocą specjalnych znaczników umieszczone są w nim polecenia w kodzie PL/SQL generujące dynamiczne części dokumentu w oparciu o dane pobrane z bazy danych.

Istnieją cztery dyrektywy w PSP:

- page – dostarcza informacji o bieżącej stronie;
- pl/sql – wyznacza nazwę procedury reprezentującej dany dokument PSP;
- parametr – ustawia parametry procedury;
- include – powoduje dołączenie w wybranym miejscu dokumentu PSP zawartości pliku wskazanego przez atrybut file.

Typy znaczników w języku PSP:

- **<%@ page ... %>** - określa metadane zawierające informacje o stronie w języku PSP np. język procedur, zestaw kodowania znaków, typ zawartości strony itd.
  Np. **<%@ page language = "PL/SQL" %>** - określenie pliku jako źródła kodu dla PL/SQL;
- **<%@ pl/sql parametr = "..." type = "..." %>** - określa parametry przyjmowane przez procedurę;
- **<%@ pl/sql procedure = "..." %>** - określa nazwę procedury;
- **<%@ include ... %>** - określa nazwę pliku załączanego do pliku PSP (np. plik CSS);
- **<%! ... %>** - blok deklaracji zmiennych wykorzystywanych w procedurze zapisany w języku PL/SQL;
- **<% ... %>** - blok wykonywalny poleceń PL/SQL;
- **<%=... %>** - miejsce odwołania się do zmiennej lub wyrażenia w PL/SQL;
- **<% --... --%>** - komentarz;

Procedurę składowaną można skompilować i załadować do bazy danych Oracle z poziomu wiersza poleceń programem loadpsp np.:

```
loadpsp -replace -user HR/HR "strona_glowna.psp"
```

Powyższe polecenie załaduje procedurę składowaną utworzoną na podstawie pliku 'strona_glowna.psp', a jeśli już istnieje to ją zastępuję (dzięki użyciu parametru -replace).

Uruchamianie procedur składowanych powstałych z plików PSP następuje z poziomu protokołu HTTP.

W celu uruchamiania procedur składowanych z poziomu protokołu HTTP należy utworzyć deskryptor 'DAD', który zawiera każde zapytanie kierowane do bramki PL/SQL. Deskryptory DAD są tworzone przez administratora, lecz można nadać takie prawa również innym użytkownikom bazy danych np. dla konta HR.

W celu nadania praw należy wywołać polecenie z poziomu konta administratora:

```
GRANT XDBADMIN TO HR;
```

Następnie z poziomu konta posiadającego uprawnienia XDBADMIN należy napisać procedurę tworzącą deskryptor i ją wywołać:

```
BEGIN
DBMS_EPG.CREATE_DAD(dad_name=>'dad',path=>'/plsql/*');
END;
/
```

Wywoływana jest procedura z pakietu DBMS_EPG o nazwie CREATE_DAD, która przyjmuje nazwę tworzonego deskryptora oraz ścieżkę wywołania. Tu odpowiednio: 'dad' oraz '/plsql/'.

Aby wywołać procedurę PL/SQL znajdującą się w bramce PL/SQL należy z poziomu przeglądarki wejść na stronę: `http://127.0.0.1:8080//dad/plsql/NAZWA_PROCEDURY`.

Zalety PSP:

- mechanizm PSP jest sposobem wykorzystania języka PL/SQL w środowisku WWW;
- technologia jest dość prosta i efektywna, składnia przypomina języki Pascal i Ada;
- możliwość wykorzystania zaawansowanych funkcji systemu Oracle;
- dobre wsparcie dla plików XML.

Wady PSP:

- ograniczona możliwość integracji z innymi rozwiązaniami programistycznymi;
- częsta konieczność niskopoziomowych mechanizmów np. w przypadku protokołu SMTP lepszym rozwiązaniem wydaje się być składowana procedura biblioteki JavaMail.

11. Bazy danych XML.

Bazy danych możemy podzielić na bazy płaskie (proste), w których obiekty reprezentowane są za pomocą struktury rekordów zgrupowanych w strukturach plików. Głównymi operacjami są odczytywanie rekordu i zapisywanie rekordu. Do płaskich baz danych możemy zaliczyć bazy płaskie XML. Drugą kategorią baz danych są systemy zarządzania bazą danych, czyli DBMS.

Podział baz danych ze względu na typy dokumentów XML:

- bazy danych pozwalające na przechowywanie dokumentów XML (*Enabled Database Systems*) – są to bazy niezorientowane na przetwarzanie dokumentów XML. Z reguły pozwalają na przechowywanie określonego typu dokumentów XML. Są to plikowe i tekstowe bazy danych (bazy płaskie) lub obiektowe i relacyjne bazy danych;
- bazy danych dokumentów XML (*Native Database Systems*) – są to bazy zorientowane na przetwarzanie dokumentów XML.

Przechowywanie plików XML:

- w systemie plików;
- w bazach danych w strukturach dużych obiektów binarnych typu CLOP lub BLOP;
- w bazach danych w postaci zdekomponowanej.

Podstawowy podział języków zapytań rysuje się na linii baz danych obiektowo-relacyjnych i baz danych dokumentów XML – baz danych płaskich. Do języków zapytań w obiektowo-relacyjnych bazach danych zaliczamy języki oparte na szablonach XML, oparte na funkcjach SQL lub oparte na konwersji zgodnej z mapowaniem. Języki zapytań w bazach danych XML są to języki zorientowane za przetwarzanie dokumentów XML. Działają one na zbiorach dokumentów XML oraz konstruują je.

Baza danych dokumentów XML – definiuje model dla dokumentów XML, które są jej podstawową jednostką składowania. Do jej funkcjonalności należy składowanie dokumentów XML, definiowanie i przechowywanie schematów, obsługa zapytań poprzez XQuery, XPath, oraz obsługa interfejsów programistycznych.

12. Optymalizacja zapytań.

Do metod optymalizacji zapytań możemy zaliczyć perspektywy zmaterializowane oraz migawki.

Perspektywa zmaterializowana stanowi środek prezentacji różnych danych przechowywanych w różnych tabelach w hurtowni danych w sposób zmaterializowany. Dzięki perspektywom zmaterializowanym system bazy danych może podjąć decyzję o tym, czy zapytanie powinno zostać zrealizowane na podstawie danych przechowywanych w tabelach, czy też wygenerować zbiór wynikowy na podstawie utworzonego wcześniej zbiorczego podsumowania danych.

W systemie Oracle można stworzyć dwa typu **migawek**: prostą i złożoną. Migawka prosta odwołuje się do pojedynczej tabeli oraz nie są wykonywane funkcje agregujące w stosunku do kolumn tabeli źródłowej. Ponadto migawka prosta może skorzystać z tak zwanego dziennika migawki. Migawka złożona, w odróżnieniu od migawki prostej, odwołuje się do wielu tabel oraz nie może skorzystać z dziennika migawki. Zaletą migawek (tzw. widoków zmaterializowanych) jest możliwość określenia metody odświeżania migawki. Programista może narzucić odświeżanie w trybie *Fast*, co oznacza, że dane w migawce będą aktualizowane codziennie względem zmian w tabelach, z których dana migawka korzysta.

13. Źródła.

1. *Oracle Database 12c i SQL – PROGRAMOWANIE*, Jason Price;
2. *Hurtownie danych, teoria i praktyka*, Agnieszka Chodkowska- Gyurics;
3. *Hurtownie danych. Od przetwarzania analitycznego do raportowania*, Adam Pelikant;
4. *Oracle Database 12c – PROGRAMOWANIE W JĘZYKU PL/SQL*, Michael McLaughlin;
5. *SQL*, Marcin Szeliga
6. *Oracle Database Online Documentation 11g Release 2 (11.2)*, dostęp online;

Opracowano dn. 18.06.2017
Tomasz Odzimek, Bartłomiej Osak