Wydział Informatyki
Politechnika Białostocka

Zaawansowane bazy danych i hurtownie danych

semestr I

WYKŁAD 1: Wprowadzenie do przedmiotu, proceduralny język SQL

Agnieszka Oniśko

Ważne informacje

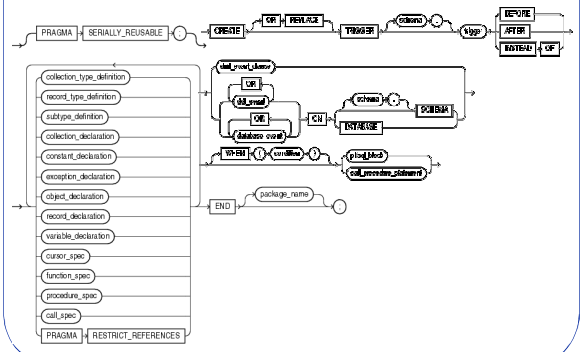
1. Sylabus w USOS'ie:
   Ogólny plan wykładów i zajęć projektowych
2. Sylabus do przedmiotu
3. http://aragorn.pb.bialystok.pl/~aonisko:
   Szczegółowy plan zajęć, materiały do wykładów i pracowni specjalistycznej, literatura, terminy zaliczeń

Plan przedmiotu

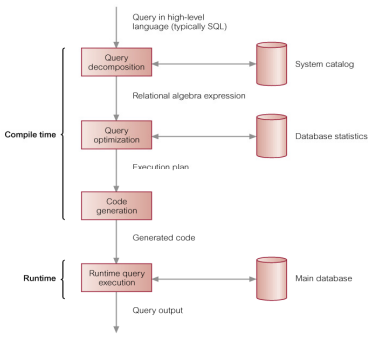
- Proceduralny język SQL (2)
- Optymalizacja zapytań w praktyce (1)
- Hurtownie danych (4)
- Integracja danych (1)
- Bazy danych typu NoSQL (2)
- Analiza danych w oparciu o modele sieci bayesowskich (3)
- Technologia Map/Reduce Apache Hadoop (2)

1. Proceduralny język SQL (ORACLE)

2. Zaawansowane obiekty baz danych (ORACLE)



3. Optymalizacja zapytań w praktyce

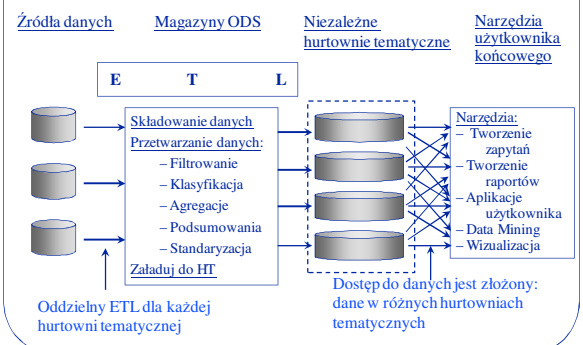


Hurtownie danych (4 wykłady)

4. Wprowadzenie do hurtowni danych, modele danych
5. Architektura hurtowni danych
6. Operacje w hurtowniach danych (zaawansowany SQL)
7. Operacje w hurtowniach danych (funkcje analityczne)

4. Modele danych w hurtowniach danych

5. Architektura hurtowni danych



6. Operaje na hurtowniach danych

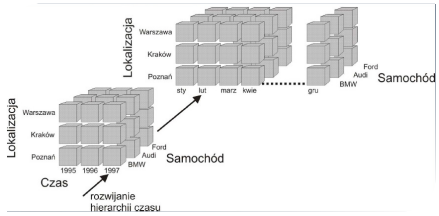
Wycinanie (Cube Slicing)

Obracanie (Pivot)

Zwijanie (Roll-up)

Agregacje (Aggregation)

Rozwijanie (Drill-down)



7. Operacje na hurtowniach danych: Funkcje analityczne

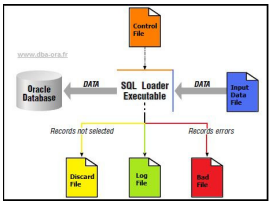
MONTH	CATEGORY	SUM(RETAIL*QUANTITY)	TOTAL_BY_MONTH
APRIL	CHILDREN	35.8	1472.7
APRIL	COMPUTER	666.6	1472.7
APRIL	COOKING	139.65	1472.7
APRIL	FAMILY LIFE	559.75	1472.7
APRIL	FITNESS	30.95	1472.7
APRIL	LITERATURE	39.95	1472.7
MARCH	BUSINESS	31.95	253.75
MARCH	COMPUTER	111.9	253.75
MARCH	COOKING	19.95	253.75
MARCH	FAMILY LIFE	89.95	253.75

10 rows returned in 0.00 seconds CSV Export

```
SELECT TO_CHAR(orderdate, 'MONTH') as month, category, SUM(retail*quantity),
SUM(SUM(retail*quantity)) OVER (PARTITION BY TO_CHAR(orderdate, 'MONTH')) as
total_by_month
FROM  orderitems, books, orders
WHERE orders.order#=orderitems.order# AND orderitems.isbn=books.isbn AND
                    TO_CHAR(orderdate, 'YYYY') = 2005
GROUP BY TO_CHAR(orderdate, 'MONTH'), category
ORDER BY TO_CHAR(orderdate, 'MONTH'), category;
```


8. Integracja danych

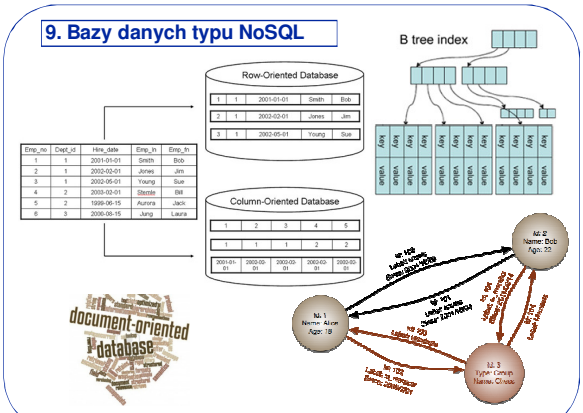
ORACLE SQL Loader



9-10. Bazy danych typu NoSQL (2 wykłady)

9. Bazy danych typu NoSQL
10. Bazy dokumentów (np. MongoDB)

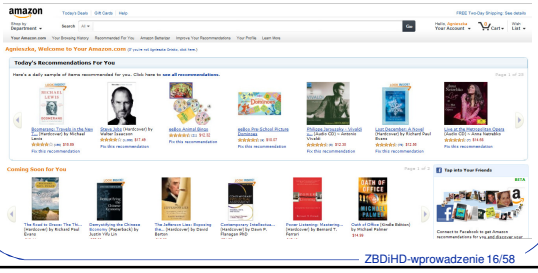
9. Bazy danych typu NoSQL



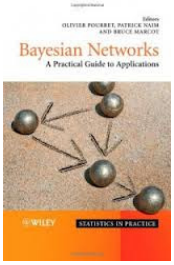
10. Bazy danych typu NoSQL: Bazy dokumentów

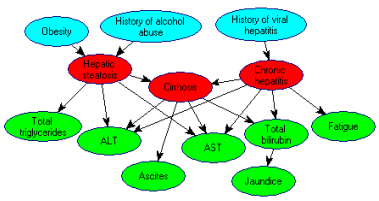
11-13. Analiza danych w oparciu o modele sieci bayesowskich (3 wykłady)

11. Konstruowanie modeli sieci bayesowskich
12. Uczenie modeli sieci bayesowskich z danych
13. Sieci bayesowskie: zastosowania

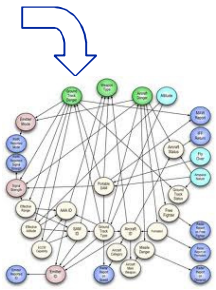


11. Konstruowanie modeli sieci bayesowskich



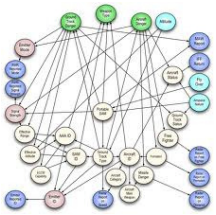


12. Uczenie modeli sieci bayesowskich z danych

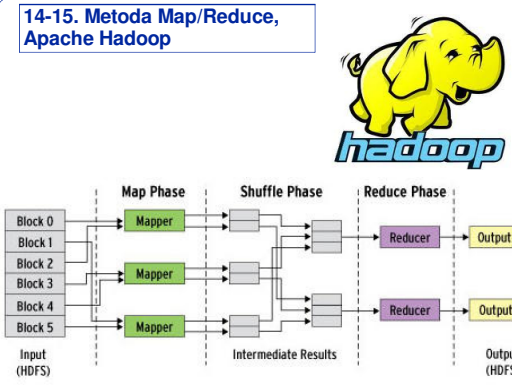


13. Sieci bayesowskie: zastosowania

Systemy diagnostyczne i prognostyczne



14-15. Metoda Map/Reduce, Apache Hadoop



PL/SQL

Relacyjny SZBD: ORACLE

- Pierwszy komercyjny SZBD (1977)
- Etapy rozwoju ORACLE
  - 1978: wersja 1.0 (128kB)
  - 1986: wersja z funkcjami bazy rozproszonej
  - 1993: implementacja hurtowni danych, wersja pod Linux'a
  - 1997: wersja 8.x - relacyjno-obiektowa (Java)
  - 2000: wersja ORACLE 9i
  - 2003: wersja ORACLE 10g
  - 2007: wersja ORACLE 11g
  - 2011: wersja ORACLE 11g, Release 2

SQL (Structured Query Language)

- Definiowanie danych (DDL:Data Definition Language)
- Definiowanie zapytań (DQL: Data Query Language)
- Modyfikowanie danych (DML: Data Modification Language)
- Sterowanie danymi (DCL: Data Control Language)

ORACLE: Języki programowania

- SQL
- PL/SQL
- Java

PL/SQL (Procedural Language/ Structured Query Language)

- PL/SQL: język o strukturze blokowej
  - bloki nazwane
  - bloki anonimowe
  - bloki podrzędne

PL/SQL Language Reference

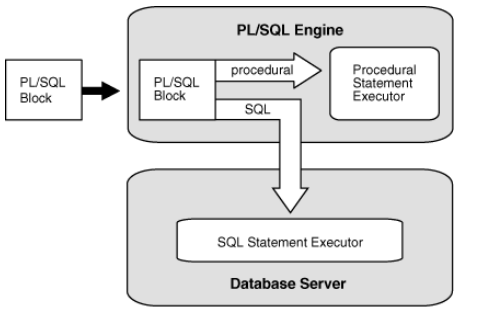
Oracle® Database PL/SQL Language Reference
11g Release 1 (11.1)
Part Number B28370-05

Next

Contents

źródło: http://docs.oracle.com/

Wykonanie bloku w PL/SQL



źródło: http://docs.oracle.com/

Wykonanie bloku w PL/SQL w środowisku SQL*Plus

```
[DECLARE
deklaracje]

BEGIN

  [EXCEPTION
  obsługa wyjątków]

END;
```

RUN
/

START
@

Struktura bloku w PL/SQL

zmienne, stałe, kursory, wyjątki zdefiniowane przez użytkownika

instrukcje języka SQL
instrukcje sterujące PL/SQL

działania, które mają być podjęte w razie wystąpienia błędów

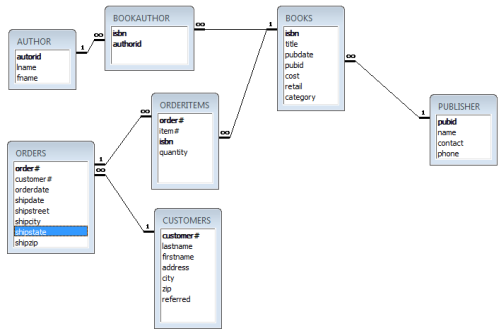
Obsługa SQL w PL/SQL

```
SELECT
INSERT
UPDATE
DELETE
```

```
COMMIT
ROLLBACK
SAVEPOINT
```

Funkcje, operatory i pseudokolumny dostępne w SQL

Schemat bazy



Zmienne i stałe

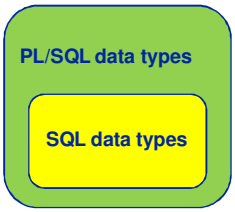
ZMIENNA:
identyfikator typ_danych [**NOT NULL**] [:=wart_pocz | **DEFAULT** wart_pocz];

STAŁA:
identyfikator **CONSTANT** typ_danych [:=wart_pocz | **DEFAULT** wart_pocz];

```
DECLARE
v_zmienna NUMBER(5);
v_nr NUMBER(3) NOT NULL := 10;
c_stala CONSTANT NUMBER(2) DEFAULT 13;
v_sprawdz BOOLEAN NOT NULL := TRUE;
```


Typy danych

- **Typy danych SQL**
  - Number()
  - Varchar2()
  - Date
  - .....
- **Typy danych PL/SQL**
  - Boolean
  - %TYPE
  - %ROWTYPE
  - .....



Typy danych

- **zmienna%TYPE**
- **tabela.kolumna%TYPE**
- **tabela%ROWTYPE**

```
v_imie VARCHAR2(15);
v_moje_imie v_imie%TYPE;
v_cena books.retail%TYPE;
ksiazki_rekord books%ROWTYPE;
```


Blok PL/SQL: Przykład

```
DECLARE
   v_data DATE NOT NULL := SYSDATE;
   v_data_str VARCHAR2(10);

BEGIN
   v_data_str := v_data; -- konwersja niejawna
   v_data_str := TO_CHAR(v_data); /* konwersja
END;                                jawna */
```


Reguły składniowe w PL/SQL

- **Instrukcje mogą się ciągnąć przez kilka linii (słowa kluczowe!)**
- **Jednostki leksykalne mogą być separowane spacjami**
- **Literały znakowe i datowe muszą być ujęte w pojedynczy cudzysłów**
- **Liczby mogą być reprezentowane przez wartości proste lub notację wykładniczą (np. 2E5 = 200000)**

Instrukcja warunkowa

```
IF warunek THEN
     instrukcja-1;
END IF;
```

```
IF warunek THEN
     instrukcja-1;
ELSE
    instrukcja-2;
END IF;
```

```
IF warunek-1 THEN
     instrukcja-1;
ELSIF warunek-2 THEN
     instrukcja-2;
ELSE
     instrukcja-3;
END IF;
```


Instrukcja warunkowa: Przykład

```
……...
IF v_cena > 100 THEN
     RETURN (v_cena*1.1);
ELSIF v_cena >= 50 THEN
     RETURN (v_cena*1.2);
ELSE
     RETURN (v_cena*1.25);
END IF;
……….
```


Instrukcja iteracyjna LOOP

```
LOOP
     instrukcje;
EXIT [WHEN warunek];

END LOOP;
```

```
WHILE warunek LOOP
     instrukcje;

END LOOP;
```

```
FOR licznik IN [REVERSE] min..max LOOP
     instrukcje;
END LOOP;
```


Instrukcja FOR: Przykład

```
DECLARE
     v_licznik NUMBER(1) :=0;
     v_ostatni NUMBER(1);
     v_min NUMBER(1) := 1;
     v_max NUMBER(1) := 5;
BEGIN
     FOR i IN v_min..v_max LOOP
          v_licznik := v_licznik + 1;
          v_ostatni := i;
     END LOOP;
     dbms_output.put_line(`Ostatni indeks:'
     || TO_CHAR(v_ostatni) || `. Liczba petli:'
     || TO_CHAR(v_licznik));
END;
```


Przetwarzanie wyników zapytań

SELECT *lista*
INTO *nazwa_zmiennej* | *nazwa_rekordu*
FROM *tabela*
WHERE *warunek*;

```
……….
BEGIN
     SELECT MAX(retail)
     INTO v_max
     FROM books;
END;
```


Kursory niejawne

- Pozwalają na analizowanie i wykonywanie komend SQL
- PL/SQL zarządza automatycznie kursorem niejawnym
- Kursory niejawne: deklarowane dla wszystkich instrukcji DML oraz SELECT

Atrybuty kursora SQL

SQL%ROWCOUNT	Liczba wierszy, których dotyczyła ostatnia instrukcja SQL
SQL%FOUND	Atrybut logiczny, który przyjmuje wartość TRUE, gdy ostatnia instrukcja dotyczyła jednego lub więcej wierszy
SQL%NOTFOUND	Atrybut logiczny, który przyjmuje wartość TRUE, gdy ostatnia instrukcja dotyczyła zerowej liczby wierszy
SQL%ISOPEN	Zawsze przyjmuje wartość FALSE ponieważ PL/SQL zamyka kursory natychmiast po ich wykonaniu

Atrybuty kursorów niejawnych

```
PROCEDURE KASUJ_KSIAZKE (p_isbn books.isbn%TYPE)
IS

BEGIN
     DELETE FROM books
     WHERE isbn = p_isbn;
     IF SQL%NOTFOUND THEN
           dbms_output.put_line
                 ('Nie znaleziono rekordu');
     END IF;
END;
```


Kursory jawne: Deklaracja

CURSOR *nazwa_kursora* [(*parametr* [, *parametr*] …)] **IS**
instrukcja_select;
parametr ::= nazwa_parametru typ danych [{:= | **DEFAULT** wyrażenie}]

```
CURSOR kur_books (p_cost NUMBER(5)) IS
              SELECT *
              FROM books
              WHERE cost <= p_cost;
```


Kursory jawne

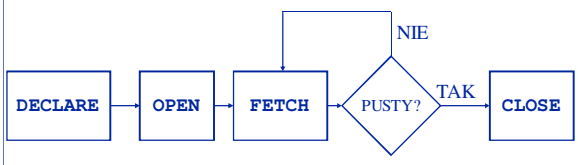
DECLARE ..... ;

OPEN *nazwa_kursora* [(*parametr* [, *parametr*] ...)];

FETCH *nazwa_kursora* **INTO** *zmienna* [, *zmienna* ...];

CLOSE *nazwa_kursora*;

Sterowanie kursorami jawnymi



Atrybuty kursorów niejawnych

%ROWCOUNT	Liczba wierszy, sprawdzonych w kursorze w danym momencie
%FOUND	Atrybut logiczny, który przyjmuje wartość TRUE, jeśli instrukcja FETCH zwróci wiersz
%NOTFOUND	Atrybut logiczny, który przyjmuje wartość TRUE, jeśli ostatnia instrukcja FETCH nie zwrócila żadnego wiersza
%ISOPEN	Atrybut logiczny, który przyjmuje wartość TRUE, jeśli kursor jest otwarty

```
DECLARE
CURSOR kur_books(p_pubid publisher.pubid%TYPE)
        IS SELECT * FROM books
           WHERE pubid = p_pubid;
k_record books%ROWTYPE;
v_cost books.cost%TYPE := 0;
BEGIN
     OPEN   kur_books;
     LOOP
          FETCH kur_books(1) INTO k_record;
          IF kur_books%NOTFOUND THEN
               EXIT;
          END IF;
          v_cost := v_cost + k_record.cost;
     END LOOP;
     -- kur_books%ROWCOUNT ???;
     dbms_output.put_line(v_cost);
     CLOSE   kur_books;
END;
```


Pęta kursorowa FOR

```
FOR nazwa_rekordu IN nazwa_kursora LOOP

     instrukcja-1;
     instrukcja-2;
     …….
END LOOP;
```

Niejawne instrukcje **OPEN**, **FETCH**, **CLOSE**
zmienna *nazwa_rekordu* -- zadeklarowana niejawnie

Kursory jawne: Przykład

```
DECLARE
     CURSOR k_order
          (p_zam orders.order#%TYPE) IS
          SELECT *
          FROM orderitems
          WHERE order# = p_zam;
BEGIN
     FOR i IN k_order(1) LOOP
          dbms_output.put_line(i.isbn);
     END LOOP;
END;
/
```


Kursor w pętli FOR

```
DECLARE
    CURSOR a_cursor IS SELECT * FROM customers;

BEGIN
   FOR i IN a_cursor LOOP
       dbms_output.put_line(i.lastname);
   END LOOP;
END;
```

```
BEGIN
   FOR i IN (SELECT * FROM customers) LOOP
       dbms_output.put_line(i.lastname);
   END LOOP;
END;
```


Kursor w pętli FOR

```
BEGIN
-- Inefficient, calls function for every row
  FOR item IN
       (SELECT DISTINCT(SQRT(department_id)) x
        FROM employees)
       LOOP
    DBMS_OUTPUT.PUT_LINE(item.x);
  END LOOP;
END;
/
```

źródło: http://docs.oracle.com/

Kursor w pętli FOR

```
BEGIN
-- Efficient, only calls function once
-- for each distinct value.
 FOR item IN
       ( SELECT SQRT(department_id) x
         FROM (SELECT DISTINCT department_id
               FROM employees)
        )
        LOOP
          DBMS_OUTPUT.PUT_LINE(item.x);
 END LOOP;
END;
/
```

źródło: http://docs.oracle.com/

Polecenie FORALL

```
DECLARE
TYPE NumList IS VARRAY(20) OF NUMBER;
-- department numbers:
depts NumList := NumList(10, 30, 70);

BEGIN
  FORALL i IN depts.FIRST..depts.LAST
    DELETE FROM employees_temp
    WHERE department_id = depts(i);
  COMMIT;
END;
/
```


Użycie klauzuli WHERE CURRENT OF

```
DECLARE
     CURSOR k_cursor(v_cost books.cost%TYPE) IS
          SELECT cost FROM books
          WHERE cost > v_cost
          FOR UPDATE;
BEGIN
     FOR i IN k_cursor(40) LOOP
          UPDATE books SET cost = cost* 1.1
          WHERE CURRENT OF k_cursor;
     END LOOP;
     COMMIT;
END;
/
```


Użycie klauzuli WHERE CURRENT OF

- Używając kursorów można aktualizować lub usuwać rekordy
- Klauzula `FOR UPDATE` blokuje wiersze
- Klauzula `WHERE CURRENT OF` odwołuje się do bieżącego rekordu z kursora jawnego
- Nie wolno zatwierdzać instrukcji z kursora jawnego, jeżeli użyta jest klauzula `FOR UPDATE`

Kursory: Podsumowanie

- Kursory niejawne używane dla wszystkich instrukcji DML oraz zapytań zwracających jeden wiersz
- Kursory jawne używane dla zapytań zwracających zero lub więcej wierszy