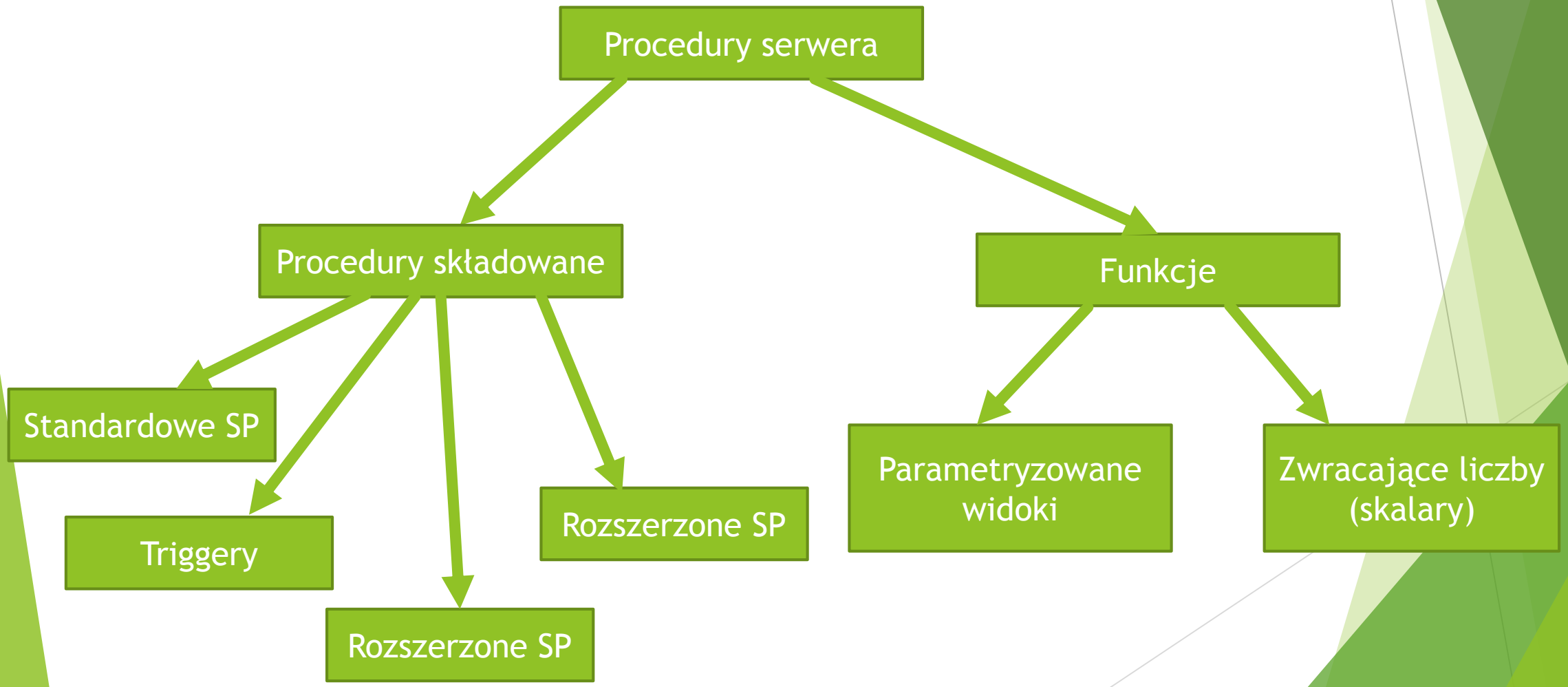
Programowanie po stronie serwera w SZBD

Robert A. Kłopotek

r.klopotek@uksw.edu.pl

Wydział Matematyczno-Przyrodniczy. Szkoła Nauk Ścisłych, UKSW

Programowanie SZBD (DBMS)



Procedury składowane

- Procedury definiowane w języku podobnym do SQL i wykonywane przez SZBD
- Pozwalają na prześmiesznie wykonania zasobożernych procesów, np. złożonych aktualizacji tysięcy dokumentów
- Systemowe procedery składowane pozwalają na konfigurację SZBD i ustawianie opcji niedostępnych poprzez zapytania SQL, np. wyświetlanie wersji MySQL: `SELECT VERSION();`

Procedury składowane – składnia podstawowa MS SQL Server

- `CREATE PROCEDURE ProcedureName [@parameterName ParameterType [OUTPUT],…] AS procedureCode`
- `EXECUTE PROCEDURE ProcedureName [@variableName|Value,…]`

- Może być użytych wiele argumentów różnych typów
- Procedura może zwrócić wiele wartości poprzez parametr OUTPUT

Przykładowa procedura MS SQL Server

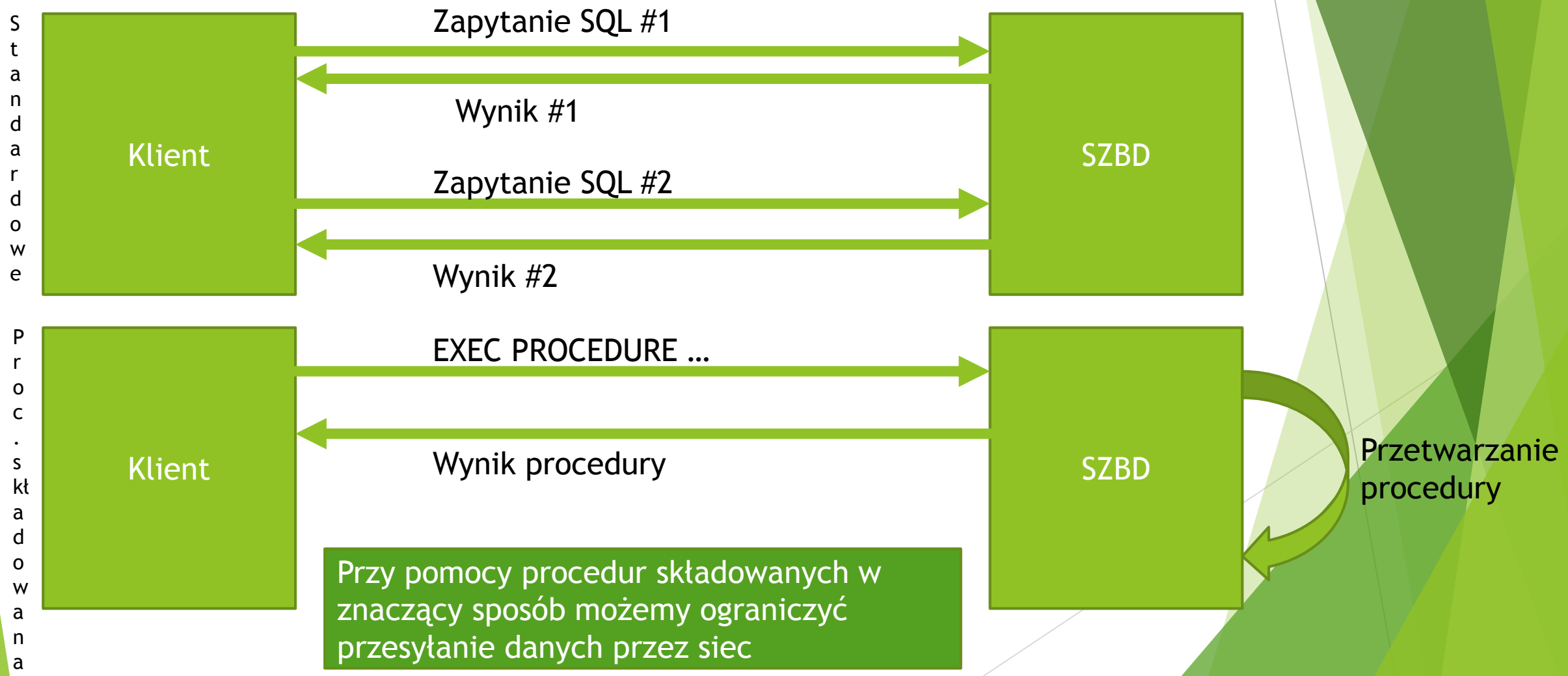
```
CREATE PROCEDURE CalculateOrderCount @country
varchar(100)
AS
DECLARE @customer varchar(100)
DECLARE custom CURSOR LOCAL FOR SELECT CustomerId FROM
Customers
WHERE country=@country
OPEN custom
FETCH NEXT FROM custom INTO @customer
WHILE @@FETCH_STATUS=0
BEGIN
UPDATE Customers SET OrderCount=(SELECT COUNT(*) FROM
Orders
WHERE CustomerId=@customer) WHERE CustomerId=@customer
FETCH NEXT FROM custom INTO @customer
END
CLOSE custom
DEALLOCATE custom
```

Przykładowa procedura MySQL

```
DELIMITER //
CREATE PROCEDURE country_hos(IN con CHAR(20))
BEGIN
  SELECT Name, HeadOfState FROM Country
  WHERE Continent = con;
END //
DELIMITER ;
```

```
CALL country_hos('Europe');
```

Procedury składowane vs przetwarzanie zapytań



Przy pomocy procedur składowanych w znaczący sposób możemy ograniczyć przesyłanie danych przez siec

Logika po stronie bazy danych – za i przeciw

- Funkcje baz danych oraz procedury składowane:
  - Pozwalają na tworzenie sparametryzowanych widoków oraz skomplikowanego przetwarzania po stronie serwera
  - Bardzo wydajne przetwarzanie – zredukowany narzut wymiany informacji pomiędzy klientem a serwerem
  - Plany wykonania przygotowane przez serwer – są szybsze niż plany ad hoc (podczas wysyłania zapytań)
  - Niestety istnieją różne standardy dla różnych dostawców (T-SQL, PL/SQL) – bardzo ograniczona przenośność funkcji i procedur składowanych

Problemy do rozważenia

- Wydajność przetwarzania – czy możemy przesunąć w czasie wykonanie?
- Przenośność rozwiązań:
  - Procedury składowane
  - Funkcje definiowane przez użytkownika
  - EJB
- Określenie kosztów każdego z rozwiązań wg wymagań „must do”

Triggery (wyzwalacze)

- Procedury wykonywane po modyfikacji lub zamiast modyfikacji bazy danych
- Dzielimy je na: triggery INSERT, triggery UPDATE i triggery DELETE
- Stąd możemy zdefiniować np. trigery AFTER INSERT dla tabeli Zamówień, czyli po każdej instrukcji INSERT system bazodanowy będzie wywoływał tą procedurę
- UWAGA: jeśli więcej niż jeden rekord jest objęty instrukcją grupy CRUD to trigger będzie aktywowany tylko raz – dla całej grupy rekordów objętych tą instrukcją

Triggery - uwagi

- Możliwości triggerów zależą od konkretnej implementacji SQL
- W niektórych implementacjach (np. T-SQL) triggery możemy definiować dla widoków. Stąd widok jest traktowany jako tabela i możemy określić co się stanie, gdy wstawimy rekord do widoku
- Trigger może powodować modyfikacje innych tabel, lub nawet tej, z której został wywołany co może powodować koleje wywoływania triggerów przez SZBD

Triggery - składnia

```
CREATE TRIGGER TriggerName ON TableName
{AFTER|INSTEAD OF}
{INSERT,UPDATE,DELETE}
AS
TriggerCode
```

Trigger AFTER zostaną wywołane po określonego działania, np. po wstawieniu rekordu do tabeli.

Triggery INSTEAD OF jak nazwa wskazuje zostna wywołane zamiast konkretnego działania, np. wstawiania rekordu do tabeli czy widoku, dla których zostały zdefiniowane.

Triggery – przykład MS SQL Server

```
CREATE TRIGGER InsertOrder ON Orders AFTER
INSERT
AS
DECLARE @customer varchar(100)
SET @customer=(SELECT CustomerId FROM inserted)
UPDATE Customers SET OrderCount=OrderCount+1
WHERE CustomerId=@customer
```

- Przykład zakłada, że jedna instrukcja INSERT wstawia jedno zamówienie do tabeli
- W ogólności można wstawić w jednym zapytaniu wiele rekordów przez INSERT … SELECT
- Trigger zostanie uruchomiony dla każdego wiersza po jego wstawieniu w bazie danych Oracle

Triggery – przykład MySQL

```
mysql> CREATE TABLE account (acct_num INT, amount DECIMAL(10,2));
Query OK, 0 rows affected (0.03 sec)

mysql> CREATE TRIGGER ins_sum BEFORE INSERT ON account
       FOR EACH ROW SET @sum = @sum + NEW.amount;
Query OK, 0 rows affected (0.01 sec)

mysql> SET @sum = 0;
mysql> INSERT INTO account VALUES(137,14.98),(141,1937.50),(97,-
100.00);
mysql> SELECT @sum AS 'Total amount inserted';
+-----------------------+
| Total amount inserted |
+-----------------------+
|               1852.48 |
+-----------------------+
```

Triggery – przykład MySQL

```
mysql> delimiter //
mysql> CREATE TRIGGER upd_check BEFORE UPDATE ON account
    -> FOR EACH ROW
    -> BEGIN
    ->     IF NEW.amount < 0 THEN
    ->         SET NEW.amount = 0;
    ->     ELSEIF NEW.amount > 100 THEN
    ->         SET NEW.amount = 100;
    ->     END IF;
    -> END;//
mysql> delimiter ;
```

Ograniczenia programisty

- Wskazówki:
  - W pierwszej kolejności spróbuj użyć integralności referencji, zdefiniuj klucze główne, klucze obce i ograniczenia CHECK
  - Jeśli jest to konieczne użyj triggerów, aby wymusić bardziej skomplikowane zasady
  - Triggery mogą wycofać całą transakcję (ROLLBACK), jednak nie jest to zalecane, aby wycofywać takie transakcje, a szczególnie jeśli używamy triggerów AFTER

Procedury składowane, triggery i funkcje

- Pozwalają na najbardziej efektywne przetwarzanie – w szczególności tam, gdzie zależy nam na czasie wykonania
- Można ich używać w wielu zastosowaniach i zapewnić spójne przetwarzanie
- Triggery mogą pomóc w wykonywaniu tych samych działań na bazie danych niezależnie z jakiej aplikacji użytkownika została wysłana instrukcja grupy CRUD

Plany wykonania

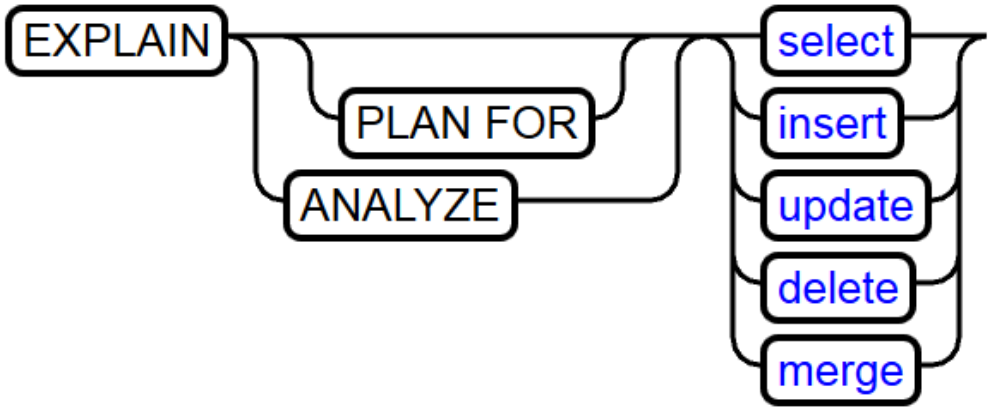
- Nasuwa się myśl, że rozwiązanie wybrane przez optymalizator nie musi być wcale rozwiązaniem optymalnym. Jeśli na przykład bardzo dobrze znamy bazę danych, może się zdarzyć, że będziemy w stanie przygotować lepszy plan zapytania.
- Przed wykonaniem można się przyjrzeć planowi wykonania zapytania (EXPLAIN PLAN). Jest to swego rodzaju symulacja która pokazuje w jaki sposób silnik bazy danych będzie wykonywał zapytanie na bazie. Z planu wykonania zapytania można odczytać jakie indeksy będą użyte, które tabele będą w całości skanowane, jaka będzie kolejność wykonania zapytania, itp.
- Jeśli to samo zapytanie lub podobne do procedury składowanej będzie wywołane ad hoc to SZBD może użyć gotowego planu wykonania
- SZBD może przetrzymywać plany wykonania dla SP, które pozwolą na dalsze zmniejszanie czasu wykonania poprzez caching najlepszych przewidywanych metod dostępu do danych oraz wykonywania żądanych operacji

Plany wykonania różnych dostawców

- Funkcja planu wykonania jest w różny sposób obsługiwana przez różnych producentów:
- MS SQL Server dostarcza narzędzia Query Analyzer
- W Oracle jest to polecenie EXPLAIN PLAN.
- Użytkownicy frontendu Oraclowego Toad mogą w łatwy sposób analizować plan wykonania każdego zapytania (poprzez naciśnięcie Ctrl+E)
- Baza hurtowni danych Teradata ma bardzo mocno rozbudowany mechanizm optymalizacji zapytań
- Generalnie wystarczy umieścić słowo EXPLAIN przed poleceniem SQL i optymalizator wypisze na ekranie dokładny przebieg wykonania zapytania SQL

Plan wykonania – baza h2 Server

```
EXPLAIN select w.kod_woj, w.nazwa,
count (*) as liczba_miast
from woj w
join miasta m on m.kod_woj=w.kod_woj
group by w.kod_woj, w.nazwa
having count (w.kod_woj) > 2;
```




```
SELECT
    W.KOD_WOJ,
    W.NAZWA,
    COUNT(*) AS LICZBA_MIAST
FROM PUBLIC.WOJ W
    /* PUBLIC.WOJ.tableScan */
INNER JOIN PUBLIC.MIASTA M
    /* PUBLIC.CONSTRAINT_INDEX_8:
KOD_WOJ = W.KOD_WOJ
          AND KOD_WOJ = W.KOD_WOJ
     */
    ON 1=1
WHERE M.KOD_WOJ = W.KOD_WOJ
GROUP BY W.KOD_WOJ, W.NAZWA
HAVING COUNT(W.KOD_WOJ) > 2
```

Czego nie dowiemy się z planu zapytania?

- Plany zapytań:
  - Nie informują o założonych blokadach
  - Nie pozwolą na zbadanie statystyk oczekiwania na dostępność obiektów
  - Nie stwierdzają, czy dane znajdują się w pamięci podręcznej czy nie
  - Nie porównują rzeczywistego kosztu wykonania z estymowanym kosztem
  - W niektórych sytuacjach pewne operacje mogą być niewidoczne w planie zapytania

Normalna praca SZBD

- Nowoczesne DBMS działają nieprzerwanie
- Oferują wbudowane funkcje wykonywania regularnie różnych działań, np. co godzinę
- Tak więc procedury składowane mogą być wykonywane automatycznie - na przykład każdej nocy, aby przenosić tymczasowe dane między systemami, usuwać nieaktualne informacje itp.
- W MS SQL Server 2000 program SQL Server Agent jest usługą odpowiedzialną za zarządzanie zadaniami

Pytania?