

Paweł Rajba

pawel@cs.uni.wroc.pl

<http://pawel.ii.uni.wroc.pl/>

T-SQL: Podstawy

Agenda

- Wprowadzenie
- Historia i standardy
- Podstawy relacyjności
- Typy danych
- DDL
 - tabele, widoki, zmiana struktury
- DML
- DQL
 - Podstawy, złączenia, podzapytania, grupowanie, operacje na zbiorach

Wprowadzenie

- Co to jest baza danych?
 - Wiele różnych definicji, ale najkrócej:
 - Zbiór danych trwałych, które można przeszukiwać
- Przykłady baz danych
 - Książka telefoniczna
 - Plik tekstowy
 - Arkusz Excel'a
 - Usługa katalogowa LDAP
 - Baza relacyjna w SQL Server

Wprowadzenie

- SQL, czyli Structured Query Language
- Umożliwia
 - Definiowanie struktur bazy danych
 - DDL – Data Definition Language
 - Operowanie na danych: dodawanie, udostępnianie, modyfikowanie i usuwanie
 - DML – Data Manipulation Language
 - Zarządzanie dostępem do danych
 - DCL – Data Control Language
 - Definiowanie zapytań
 - DQL – Data Query Language

Historia i standardy

- Język pojawił się oficjalnie w 1974 jako Structured English Query Language – SEQUEL
 - Twórcy: Donald D. Chamberlin i Raymond F. Boyce
- 1987: SQL-86 jako oficjalny standard ISO/ANSI
- Kolejne standardy:
 - SQL-89, SQL-92, SQL:1999
 - SQL:2003, SQL:2006, SQL:2008, SQL:2011
 - <http://en.wikipedia.org/wiki/SQL#Standardization>
- Przyjrzymy się językowi SQL w wersji SQLServer

Podstawy relacyjności

- Relacyjna baza danych składa się z:
 - Tabelek (a.k.a. encji, relacji)
 - Powiązań między nimi
 - ... i mnóstwem innych obiektów, ale podstawa to tabelki
- Tabela
 - Kolumny, wiersze
 - Klucz główny
 - Więzy integralności
- Powiązania między tabelami, czyli klucz obcy
 - Książka(isbn, tytuł, autor, rok_wydania, cena)
 - Egzemplarz(sygnatura, isbn)
 - {Egzemplarz.isbn} \subset {Książka.isbn}

Podstawy relacyjności

KSIĄŻKA

ID (PK)	Tytuł
1000	Władca Much
1001	Czarny Obelisk
1002	Gra szklanych paciorków

EGZEMPLARZ

ID (PK)	K_ID (FK)	Sygnatura
1	1000	S001/22
2	1000	S002/22
3	1001	S003/22
4	1001	S004/22
5	1001	S005/22
6	1002	S006/22



Podstawy relacyjności

- Terminy formalnego modelu relacyjnego i ich odpowiedniki w systemach komercyjnych
 - relacja → tabela,
 - krotka → wiersz,
 - atrybut → kolumna,
 - wartość → pole.

Typy danych

- Rodzaje typów danych
 - Wbudowane i użytkownika
- Wbudowane
 - int (4), bigint (8), smallint (2), tinyint (1)
 - decimal(p,s), float, real
 - money (8), smallmoney (4)
 - datetime (8), smalldatetime (4)
 - char, varchar (0-8000), text (0-2GB)
 - nchar, nvarchar (0-8000), ntext (0-2GB)
 - binary, varbinary (0-8000), image (0-2GB)
 - bit, cursor, table, sql_variant, uniqueidentifier

Typy danych

- Użytkownika
 - Zarządzanie
 - CREATE TYPE, DROP TYPE
 - Przykłady
 - CREATE TYPE PESEL FROM char(11) NOT NULL
 - CREATE TYPE PLEC FROM char(1)
 - Stare rozwiązanie:
 - sp_addtype, sp_droptype
 - Obejrzenie:
 - sp_help

DDL: Tabele

■ Podstawowa składnia

```
CREATE TABLE <nazwa>
(
    <kolumna1> <typ1> [<więzy kolumny 1>],
    ...
    <kolumnaN> <typN> [<więzy kolumny N>],
    [<więzy tabeli>]
)
```

■ Kolumny mogą

- Zawierać dane, np. **Nazwisko VARCHAR(40)**
- Być wyliczane (computed)
 - np. *InventoryValue AS QtyAvailable * UnitPrice*

- Przykład: [https://msdn.microsoft.com/pl-pl/library/ms188300\(v=sql.110\).aspx](https://msdn.microsoft.com/pl-pl/library/ms188300(v=sql.110).aspx)

■ Pełna składnia:

[https://msdn.microsoft.com/pl-pl/library/ms174979\(v=sql.110\).aspx](https://msdn.microsoft.com/pl-pl/library/ms174979(v=sql.110).aspx)

DDL: Tabele

- Identyfikatory
 - Pole typu IDENTITY
 - składnia IDENTITY[(seed,increment)]
 - w tabeli możliwa jest tylko jedna kolumna z tą własnością
 - może być używana w połączeniu z typami całkowitymi, np. int, bigint, smallint, tinyint
 - normalnie wartości kolumny nie mogą być uaktualniane
 - nie są dozwolone wartości NULL
 - tylko generator, nie zapewnia unikalności
 - Wstawianie dowolnej wartości
 - SET IDENTITY_INSERT tablename ON
 - SET IDENTITY_INSERT tablename OFF

DDL: Tabele

■ Identyfikatory

■ Co więcej o IDENTITY?

- `IDENT_SEED('tabela')` i `IDENT_INCR('tabela')` zwracają wartość startową i krok
- zmienna `@@IDENTITY` zwraca ostatni wygenerowany numer w ramach sesji (połączenia)
- funkcja `SCOPE_IDENTITY()` zwraca ostatni wygenerowany numer w ramach tego samego zasięgu (zasięg tworzy procedura, wyzwalacz, funkcja lub wsad)
- `IDENT_CURRENT('tabela')` zwraca ostatni wygenerowany numer w ramach wszystkich sesji i zasięgów
- Zamiast nazwy kolumny z `IDENTITY` można używać `$identity`

DDL: Tabele

- Identyfikatory
 - Pole typu UNIQUEIDENTITY
 - Generowane przez funkcje NEWID() lub NEWSEQUENTIALID()
 - Identyfikator to 16 bajtowa wartość binarna w następującym formacie:
XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
 - Oznaczenie pola przez ROWGUIDCOL umożliwia dostęp przez \$rowguid (ale nie daje unikalności)
 - Typowa definicja jako identyfikatora:
[ID] UNIQUEIDENTIFIER ROWGUIDCOL PRIMARY KEY
DEFAULT NEWSEQUENTIALID()

DDL: Tabele

- Sekwencje
 - Reprezentuje ciąg liczb
 - Rozwiązanie znane z Oracle Database
 - Przykładowe zastosowania
 - Aplikacja potrzebuje ID przed wstawieniem rekordu do bazy danych
 - Wiele tabel potrzebuje listy unikalnych identyfikatorów
 - Jest potrzeba unikalnych identyfikatorów
 - Potrzebnych z góry kilka kolejnych wartości

DDL: Tabele

- Sekwencje: składnia

- CREATE SEQUENCE [schema_name .]
sequence_name
[AS [built_in_integer_type | user-
defined_integer_type]]
[START WITH <constant>]
[INCREMENT BY <constant>]
[{ MINVALUE [<constant>] } | { NO MINVALUE }]
[{ MAXVALUE [<constant>] } | { NO MAXVALUE }]
[CYCLE | { NO CYCLE }]
[{ CACHE [<constant>] } | { NO CACHE }]

DDL: Tabele

- Typowe użycie
 - CREATE SEQUENCE dbo.orderdomain_seq
 - AS BIGINT
 - START WITH 1
 - INCREMENT BY 1
 - MINVALUE 1
 - NO CYCLE
 - NO CACHE;

 - CREATE TABLE liczby(liczba int);

 - ALTER SEQUENCE dbo.orderdomain_seq RESTART WITH 5;

 - INSERT INTO liczby VALUES (NEXT VALUE FOR dbo.orderdomain_seq);

DDL: Tabele

- Przechowywanie danych
 - Bezpośrednio w wierszu do 8KB
 - Duże dane (do 2GB) 16B wskaźnik do zewnętrznej struktury
 - Dane typu text mogą być też bezpośrednio w wierszu
 - opcja sp_tableoption
 - sp_tableoption 'tabela', 'text in row', '1000'
 - Więcej: <https://msdn.microsoft.com/en-us/library/ms173530.aspx>

DDL: Tabele

- Integralność danych
 - Na kolumny
 - DEFAULT, CHECK, RULE (deprecated), FOREIGN KEY
 - Na tabelę
 - PRIMARY KEY (zabronione nulle)
 - UNIQUE (dopuszczony null)
 - DEFAULT, RULE

DDL: Tabele

■ Więzy kolumny

```
[CONSTRAINT <nazwa więzu>]
  NOT NULL |
  DEFAULT <wartość domyślna> |
  PRIMARY KEY |
  UNIQUE |
  CHECK (<warunek>) |
  REFERENCES <tabela> [(<kolumna>)]
    [{ON DELETE | ON UPDATE}
     {NO ACTION | SET NULL | CASCADE | SET DEFAULT}]
```

■ Więzy tabeli

```
[CONSTRAINT <nazwa więzu>]
  {PRIMARY KEY | UNIQUE} (<lista kolumn>) |
  CHECK (<warunek>)
  FOREIGN KEY (<lista kolumn>)
    REFERENCES <tabela> [(<kolumna>)]
      [{ON DELETE | ON UPDATE}
       {NO ACTION | SET NULL | CASCADE | SET DEFAULT}]
```

DDL: Tabele

- Wyłączanie więzów integralności
 - Wyłączyć można tylko CHECK i FOREIGN KEY
 - Pozostałe więzy trzeba usunąć i utworzyć na nowo
 - SQL
 - ALTER TABLE nazwa { CHECK | NOCHECK }
CONSTRAINT { ALL | nazwa_więzu[,...] }
- Weryfikacja więzów integralności
 - DBCC CHECKCONSTRAINTS
[('table_name' | 'constraint_name')]
[WITH ALL_CONSTRAINTS]

DDL: Widoki

- Podstawowa składnia

```
CREATE VIEW <nazwa widoku> AS  
<select statement>
```

- Umożliwiają sterowanie uprawnieniami

- Można dać dostęp do widoku, nie dając dostępu do tabel wykorzystywanych w widoku

- Mogą być elementem optymalizacji

- Pojęcie widoku zmaterializowanego

- Kilka uwag:

- Domyślnie widoki są do odczytu, jednak po spełnieniu kilku warunków można je modyfikować

- W SELECT nie można stosować pewnych konstrukcji

- Aby uzyskać widoki zmaterializowany trzeba:

- Utworzyć go z opcją WITH SCHEMABINDING
- Utworzyć na nim Unique Clustered Index

- Pełna składnia

- [https://msdn.microsoft.com/pl-pl/library/ms187956\(v=sql.110\).aspx](https://msdn.microsoft.com/pl-pl/library/ms187956(v=sql.110).aspx)

DDL: zmiana struktury

- Podstawowa składnia
 - ALTER TABLE <nazwa tabeli>
ADD <kol> <typ>
DROP <nazwa kolumny> | <constraint>
ADD <nazwa> CHECK (<więz>)
...
 - DROP TABLE <nazwa tabeli>
 - ...
- Zmiana nazwy kolumny:
 - sp_rename 'Osoba.ID', 'OsobaID', 'COLUMN'

DML

- Podstawowa składnia

INSERT INTO t (k1,k2,...) VALUES(v1,v2,...)

UPDATE t SET k1=v1,k2=v2,... WHERE <warunek>

DELETE FROM t WHERE <warunek>

- Zestaw operacji:

CRUD=

CREATE

RETRIEVE

UPDATE

DELETE

DML

■ Ciekawostki z SQL Server

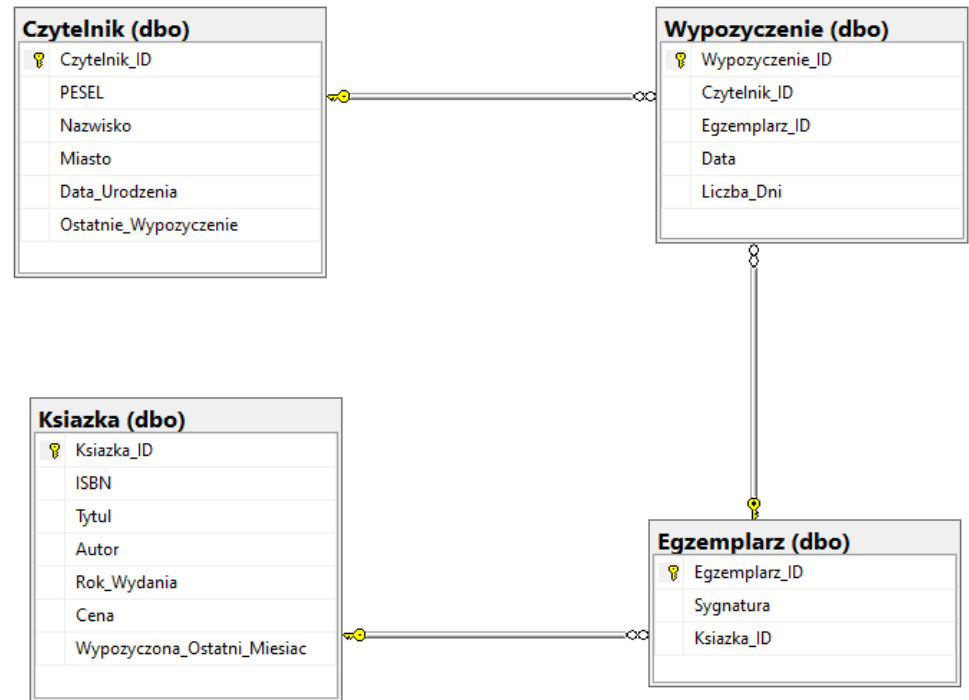
- INSERT INTO tabela VALUES (wartosci1), (wartosci2), ..., (wartosciN);
- INSERT INTO Numbers DEFAULT VALUES;
 - Np. kiedy chcemy wstawić wiersz do tabeli T(INT ID IDENTITY)
- UPDATE t SET k1=v1,k2=v2,... FROM ...
 - Do przykładu wrócimy później
- DELETE FROM t FROM ...
 - Do przykładu wrócimy później

■ Przegląd składni

- [https://msdn.microsoft.com/pl-pl/library/ff848766\(v=sql.110\).aspx](https://msdn.microsoft.com/pl-pl/library/ff848766(v=sql.110).aspx)

DEMO

- Identyfikatory.sql
- Constraints.sql
- Biblioteka.sql



DQL: Podstawy

- Podstawowa składnia
 - SELECT kolumna1, kolumna2, ...
FROM tabela1
WHERE warunki
ORDER BY 1,2,...
- Jakie mogą być warunki?
 - Operatory: =, <, >, <=, >=, IN (s1,s2,...), LIKE napis, BETWEEN ... AND ...
 - Cena < 1000,00
 - Nazwisko LIKE 'A%'
 - Gatunek IN ('muzyka', 'sf', 'biografie')
 - Wiek BETWEEN 30 AND 40

DQL: Podstawy

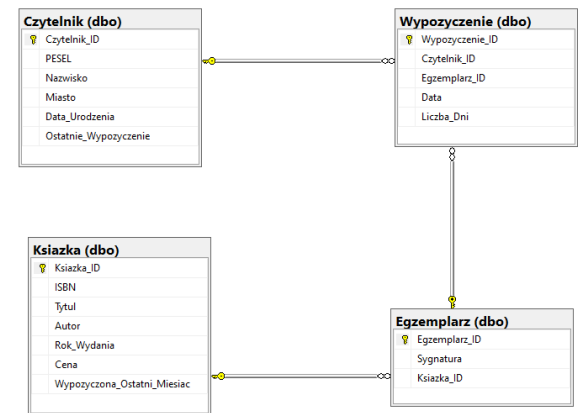
- Funkcje agregujące
 - COUNT, AVG, SUM, MIN, MAX
- Znaczenie słowa DISTINCT
- Wykonywanie obliczeń, wywoływanie funkcji
 - SELECT 2+3 AS Wynik
 - SELECT SCOPE_IDENTITY()

DQL: Podstawy

- Lista książek o cenie do 60 zł, które mają w tytule SQL Server
 - `SELECT *`
`FROM Ksiazka`
`WHERE Tytul LIKE '%SQL Server%' AND Cena < 60`
- Najdroższa i najtańsza książka
 - `SELECT MIN(Cena) AS Najtańsza, MAX(Cena) AS Najdroższa`
`FROM Ksiazka`
- Średni czas wypożyczenia
 - `SELECT AVG(LICZBA_DNI) AS "Średni czas wypożyczenia"`
`FROM Wypozyczenie`

Ćwiczenie

- Podaj rok, z którego są najstarsze książki w bibliotece
- Podaj średnią cenę książek wydanych od 2009 roku
- Wymień czytelników, którzy nigdy nie wypożyczyli książki



DQL: Złączenia

- Złączenie polega na wyświetleniu w wynikowej tabelce danych z wielu tabel
- Jak ta tabelka jest budowana? Mamy kilka rodzajów złączeń:
 - Pełne
 - Wewnętrzne
 - Zewnętrzne lewe i prawe

DQL: Złączenia

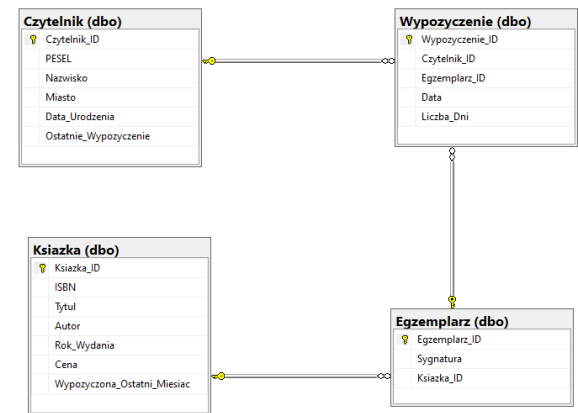
- Pełne
 - `SELECT k.*, e.*
FROM Ksiazka k, Egzemplarz e`
- Wewnętrzne
 - `SELECT k.*, e.*
FROM Ksiazka k, Egzemplarz e
WHERE k.ksiazka_id=e.ksiazka_id`
 - `SELECT k.*, e.*
FROM Ksiazka k INNER JOIN Egzemplarz e
ON k.ksiazka_id=e.ksiazka_id`
- Zewnętrzne
 - `SELECT k.*, e.*
FROM Ksiazka k LEFT JOIN Egzemplarz e
ON k.ksiazka_id=e.ksiazka_id`

DQL: Złączenia

- Tytuły i liczby dni, na które książki były wypożyczane
 - ```
SELECT k.Tytuł, w.dni
FROM Wypozyczenie w, Egzemplarz e, Ksiazka k
WHERE w.egzemplarz_id = e.egzemplarz_id AND
k.ksiazka_id = e.ksiazka_id
```
  - ```
SELECT k.tytul, w.liczba_dni  
FROM Wypozyczenie w JOIN Egzemplarz e  
ON w.egzemplarz_id = e.egzemplarz_id JOIN Ksiazka k  
ON k.ksiazka_id = e.ksiazka_id
```
- Cena wszystkich książek (10)
 - ```
SELECT SUM(Cena)
FROM Ksiazka k JOIN Egzemplarz e
ON k.ksiazka_id = e.ksiazka_id
```

# Ćwiczenie

- Podaj średnią cenę wszystkich egzemplarzy
- Podaj liczbę egzemplarzy książek wydanych od 2010 r.



# DQL: Złączenia

- Inny przykład
  - Nośnik reklamowy można rezerwować dokładnością do 10 minut
  - Rezerwację na zakres pewnej liczby dni chcemy modelować listą slotów 10 minutowych
    - Z której możemy potem usuwać pojedyncze sloty
- Jak to zrobić?

# DQL: Złączenia

- Rozwiązanie
  - Tworzymy dwie tabele:
    - Słownik\_Dni
    - Słownik\_Slotow\_Na\_Dzien
  - Dane: dzien\_od, dzien\_do
  - Wstawianie rezerwacji:
    - ```
INSERT INTO rezerwacja (...)  
SELECT ... FROM Słownik_Dni, Słownik_Slotow_Na_Dzien  
WHERE Słownik_Dni.dzien  
      BETWEEN dzien_od and dzien_do
```

DQL: Podzapytania

- Kiedy występuje podzapytanie?
- Rodzaje podzapytań
 - Nieskorelowane
 - `SELECT czytelnik_id, nazwisko FROM czytelnik WHERE czytelnik_id IN (SELECT DISTINCT czytelnik_id FROM WYPOZYCZENIE);`
 - Skorelowane
 - `SELECT c.czytelnik_id, c.nazwisko FROM czytelnik c WHERE EXISTS (SELECT 1 FROM WYPOZYCZENIE w WHERE w.czytelnik_id=c.czytelnik_id);`

DQL: Podzapytania

- Inny przykład
 - `SELECT * FROM
(SELECT * FROM Ksiazka
WHERE ROK_WYDANIA>2008)`

DQL: Podzapytania

■ Operator

■ ANY/SOME, ALL

- `substr(nazwisko,1,1) = ANY('A', 'B','C')`
- `pensja > ALL(SELECT pensja FROM pracownik)`

■ EXISTS, NOT EXISTS

- `EXISTS(SELECT * FROM Zamowienia WHERE ...)`

■ IN, NOT IN

- `substr(nazwisko,1,1) IN ('A', 'B','C')`
- `telefon NOT IN (SELECT telefon FROM pracownik)`

```
<ANY() - less than maximum
>ANY() - more than minimum
=ANY() - equivalent to IN
>ALL() - more than the maximum
<ALL() - less than the minimum
```

<http://stackoverflow.com/questions/2298550/oracle-any-vs-in>

DQL: Podzapytania

- Operatory
 - Ciekawa składnia Oracle dla operatora IN
 - WHERE (TYTUL, AUTOR, ROK_WYDANIA) IN
(
 ('T1', 'A1', 1980),
 ('T2', 'A2', 1985),
 ('T3', 'A3', 1984),
 ...
)

DQL: Grupowanie

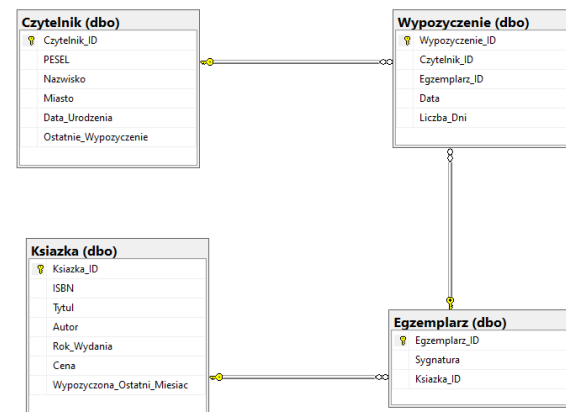
- Wynik zapytania dzielimy na grupy
- Dla każdej grupy wykonujemy agregację
- Przykłady
 - Proste podliczenie liczby egzemplarzy
 - `SELECT ksiazka_id, COUNT(egzemplarz_id) as "Liczba egzemplarzy"`
`FROM egzemplarz`
`GROUP BY ksiazka_id;`
 - `SELECT k.tytul, COUNT(e.egzemplarz_id) as "Liczba egzemplarzy"`
`FROM ksiazka k JOIN egzemplarz e ON k.ksiazka_id = e.ksiazka_id`
`GROUP BY k.tytul;`
 - Jaka jest różnica?

DQL: Grupowanie

- Klauzula HAVING umożliwia określanie grup, które mają się znaleźć w wyniku
- Jaka jest zależność pomiędzy WHERE i HAVING?
- Przykłady
 - ```
SELECT k.tytul, COUNT(e.egzemplarz_id) as "Liczba egzemplarzy"
FROM ksiazka k JOIN egzemplarz e ON k.ksiazka_id = e.ksiazka_id
GROUP BY k.tytul
HAVING COUNT(e.egzemplarz_id)>=3
```
  - ```
SELECT k.tytul, COUNT(e.egzemplarz_id) as "Liczba egzemplarzy"  
FROM ksiazka k JOIN egzemplarz e ON k.ksiazka_id = e.ksiazka_id  
WHERE e.egzemplarz_id NOT IN  
  (SELECT egzemplarz_id FROM wypozyczenie)  
GROUP BY k.tytul  
HAVING COUNT(e.egzemplarz_id)>=3
```

Ćwiczenie

- Podaj średnią liczbę dni, na jaką poszczególni czytelnicy wypożyczali książki
- Wymień czytelników, którzy wypożyczyli średnio na więcej dni, niż średnia biblioteczna



DQL: Operacje na zbiorach

- UNION [ALL], INTERSECT, MINUS
 - `SELECT tytul FROM ksiazka WHERE substr(tytul, 1, 1)='S'`
 - UNION | UNION ALL | INTERSECT | MINUS
 - `SELECT tytul FROM ksiazka WHERE substr(tytul, -1) in ('w','a','k')`

Dalej przyjrzymy się nieco ciekawszym konstrukcjom specyficznym dla SQL Server

DQL: WITH (CTE)

- Pozwala zdefiniować tymczasowe dane do wykorzystania w kolejnym zapytaniu
 - dane znane są jako common table expression (CTE)
- Definicja CTE może być rekurencyjna
- DEMO: with.sql
- Więcej:
 - <http://technet.microsoft.com/en-us/library/ms175972.aspx>
 - <http://technet.microsoft.com/en-us/27cfb819-3e8d-4274-8bbe-cbbe4d9c2e23>

DML: Update i Delete

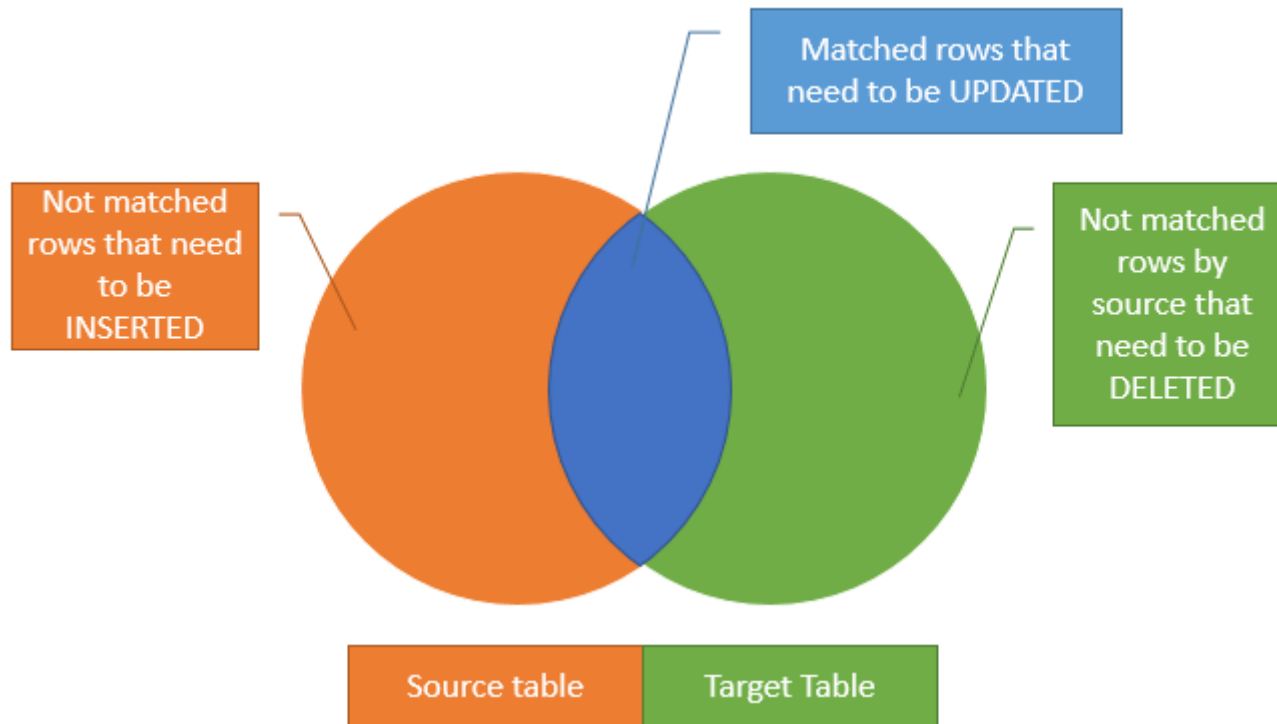
- Jak wcześniej wspomniano, DEMO
 - Update_Delete.sql
- Więcej:
 - Update
[https://msdn.microsoft.com/pl-pl/library/ms177523\(v=sql.110\).aspx](https://msdn.microsoft.com/pl-pl/library/ms177523(v=sql.110).aspx)
 - Delete
[https://msdn.microsoft.com/pl-pl/library/ms189835\(v=sql.110\).aspx](https://msdn.microsoft.com/pl-pl/library/ms189835(v=sql.110).aspx)

DML: Merge

- Dane są tabele Source i Target
- Chcemy zaktualizować Target w oparciu o stan Source
- Polecenie MERGE
 - Pozwala na wykonanie operacji INSERT, UPDATE, DELETE w ramach jednego zapytania
 - Znacznie zwiększa wydajność i pozwala bardziej zwięźle zdefiniować zadanie
 - Pozwala również na unikanie błędów

DML: Merge

- Scenariusz



DML: Merge

- Scenariusz

```
1 MERGE target_table USING source_table
2 ON merge_condition
3 WHEN MATCHED
4     THEN update_statement
5 WHEN NOT MATCHED
6     THEN insert_statement
7 WHEN NOT MATCHED BY SOURCE
8     THEN DELETE;
```

- DEMO: merge1.sql, merge2.sql

<https://www.sqlservertutorial.net/sql-server-basics/sql-server-merge/>

DQL: Stronicowanie

- Stronicowanie można zrobić na kilka sposobów
 - Za pomocą tabel tymczasowych (SQL2000)
 - Za pomocą podzapytań lub CTE (SQL2005/2008)
 - Tutaj wykorzystywana jest funkcja ROW_NUMBER
<http://msdn.microsoft.com/en-us/library/ms186734.aspx>
 - Za pomocą OFFSET i FETCH (SQL2012)
- DEMO
- Bardzo dobre podsumowanie wydajności
<http://www.mssqltips.com/sqlservertip/2696/comparing-performance-for-different-sql-server-paging-methods/>