Lab.8: Podstawy języka SQL.

SQL (Structured Query Language) jest językiem zapytań służącym do obsługi relacyjnych baz danych. Współcześnie każdy SZBD posiada własną implementację języka, opartą na wspólnym standardzie ANSI: SQL-92. Wersją SQL zastosowaną w systemie bazodanowym MS SQL Server jest Transact-SQL. Wyróżniamy następujące elementy języka Transact-SQL:

- Język definicji danych (DDL – Data Definition Language),
- Język manipulacji danymi (DML – Data Manipulation Language),
- Język kontroli danych (DCL – Data Control Language),
- rozszerzenia Transact-SQL, takie jak: zmienne, operatory, funkcje, wyrażenia kontroli przepływu oraz komentarze.

1. Data Definiton Language: DDL.

Język definicji danych wykorzystywany jest do tworzenia i zarządzania bazami danych oraz ich obiektami, takimi jak: tabele, procedury, funkcje definiowane przez użytkownika, wyzwalacze, widoki, indeksy, reguły i statystyki. Dla każdego z tych obiektów zastosować można polecenia **CREATE** oraz **DROP**, odpowiednio do ich tworzenia i usuwania:

```
USE Northwind
CREATE TABLE Employeedependents
(
dependentid INT IDENTITY(1,1),
lastname VARCHAR(20),
firstname VARCHAR(20),
)
GO
```

Do zmiany istniejących obiektów służy polecenie **ALTER**:

```
USE Northwind
ALTER TABLE Employeedependents ADD birthdate DATETIME
GO
```

Tworzenie bazy danych:

```
CREATE DATABASE nazwa_bazy_danych
```

Identyfikatory regularne (nazwy własne obiektów baz danych) nie mogą zawierać spacji. Jeżeli spacje muszą być zawarte w nazwie, całość identyfikatora należy ująć w nawias kwadratowy:

```
create database [nowa baza danych]
----
use [nowa baza danych]
create table tabela1 (col_a int, col_b varchar(20))
----
```

```
alter table tabela1 add col_c varchar(30)
```

Wyświetlenie wszystkich elementów tabeli:

```
use [nowa baza danych]
print 'zawartosc tabeli pierwszej'
select * FROM tabela1
----
USE Northwind
SELECT * FROM Categories
```

2. Data Manipulation Language: DML.

Język manipulacji danymi DML służy do uzyskiwania, wstawiania, modyfikowania i usuwania danych z baz danych, zatem podstawowe polecenia DML to: **SELECT**, **INSERT**, **UPDATE** i **DELETE**. Kryteria wyboru danych okreslamy za pomocą słowa kluczowego **WHERE**.

```
USE Northwind
INSERT INTO Customers (customerid, companyname, contactname,
contacttitle)
VALUES ('ACME1','ACME Publishing','Fernando','DBA')
----
UPDATE Customers
SET contactname = 'Fernando Guerrero'
WHERE customerid = 'ACME1'
----
SELECT customerid,companyname
FROM Customers
WHERE customerid = 'ACME1'
----
DELETE Customers
WHERE customerid = 'ACME1'
```

możliwe jest stosowanie kryteriów złożonych, np. za pomocą warunków logicznych:

```
USE Northwind
select customerid, companyname from customers
where city = 'London' OR Country = 'Germany'
----
select companyname from customers
where contacttitle = 'owner' AND country = 'france'
```

3. Data Control Language: DCL.

Język kontroli danych wykorzystywany jest do zarządzania bezpieczeństwem bazy danych. W szczególności służy do przyznawania uprawnień do obiektów bazodanowych. Podstawowe polecenia DCL: **GRANT**, **DENY**, **REVOKE**. DCL nie będzie przedmiotem zajęć.

Typy danych w języku SQL.

W języku Transact-SQL istnieje 27 typów danych, za pomocą których okreslić możemy format danych, w jakim przechowywać chcemy elementy bazy danych. Wszystke typy zawiera poniższa tabela, najważniejsze z nich zostały pogrubione:

Category	Data Type	Description
Limited character	**CHAR**	Fixed-length character data (up to 8,000 characters)
	VARCHAR	Variable-length character data (up to 8,000 characters)
	NCHAR	Fixed-length Unicode character data (up to 4,000 characters)
	NVARCHAR	Variable-length Unicode character data (up to 4,000 characters)
Unlimited character	**TEXT**	Variable-length character data (up to 2,147,483,647 characters)
	NTEXT	Variable-length Unicode character data (up to 1,073,741,823 characters)
Binary	BINARY	Fixed-length binary data (up to 8,000 bytes)
	VARBINARY	Variable-length binary data (up to 8,000 bytes)
Binary large objects	IMAGE	Variable-length binary data (up to 2,147,483,647 bytes)
Integers	BIGINT	Integer from -2^63 to 2^63 - 1
	INT	Integer from -2,147,483,648 to 2,147,483,647
	SMALLINT	Integer from -32,768 to 32,767
	TINYINT	Integer from 0 to 255
	BIT	Binary integer with only two possible values (0 or 1)
Approximate numeric	**REAL**	Approximate numbers with a precision between 1 and 7 (4 bytes of storage)
	FLOAT	Approximate numbers with a precision between 8 and 15 (8 bytes of storage)
Exact numeric	DECIMAL	Exact numbers that can use up to 17 bytes to store data
	NUMERIC	Synonym of DECIMAL
Date and time	**DATETIME**	Date and time data from January 1, 1753 to December 31, 9999. Time is accurate to the 1/300 of a second
	SMALLDATETIME	Date and time data from January 1, 1900 to June 6, 2079. Time is accurate to the minute
Currency	MONEY	Currency data from -922,337,203,685,477.5808 to 922,337,203,685,477.5807
	SMALLMONEY	Currency data from -214,748.3648 to 214,748.3647

Other	UNIQUEIDENTIFIER	16-byte GUID
	TABLE	Similar to a table database object, and used just for temporary storage
	SQL_VARIANT	Can store any Transact-SQL data type, but TEXT, NTEXT, IMAGE, TIMESTAMP, and itself
	TIMESTAMP or ROWVERSION	8-byte binary number that changes every time a column is inserted or updated
	CURSOR	Used only for variables and stored procedures output parameters

Odpowiednich typów danych należy uzywać rozważnie, z uwzględnieniem zajmowanej przez bazę danych pamięci. Przykładowo, jeśli wiemy że w danej kolumnie przechowywane będą dni miesiąca, nie będzie tu potrzebny „większy" typ niż `TINYINT` (pomijając fakt, że najlepiej zrobimy, posługując się obiektem typu data).

Przykłady deklaracji lokalnych zmiennych (tj. nie składowych tabel – znaczek '@'):

```
USE Northwind
DECLARE @unicode_data NCHAR(20)
SET @unicode_data = N'This is unicode data'
----
DECLARE @integer_data SQL_VARIANT, @char_data SQL_VARIANT
SET @integer_data = 845
SET @char_data = 'This is character data'
```

Przykłady.

Przykłady wykonywane są w programie MS SQL Server: Query Analyzer.

- otwarcie nowego okna z zapytaniami (query window): **ctrl + N**, dla wygody otworzyć można jednocześnie wiele takich okien
- wykonanie zapytań: **F5**, efekty zapytania widoczne są poniżej okna zapytań, w zakładkach: grids oraz messages,
- wywołane kwerendy zapisać można do pliku *.sql: **ctrl + S,**
- fizyczną strukturę danych można na bieżąco śledzić w oknie **object browser: F8**.
- wybranie bazy danych: słowo kluczowe **USE**, np use northwind.

1. Wyświetl wszystkich pracowników z regionu oznaczonego WA.

```
select * from employees where region = 'WA'
```

2. Policz wszystkich pracowników z regionu oznaczonego WA.

```
select count (*) from employees where region = 'WA'
```

3. Wyświetl dwa wybrane atrybuty wszystkich elementów tabeli products.

```
select productname, unitprice from products
```

4. Pokaż liczbę produktów pochodzących od dostawców ze szwecji (tabele products oraz suppliers połączone są za pomocą supplierID)

```
select unitsinstock from suppliers s, products p where
s.supplierid=p.supplierid AND country ='sweden'
```

5. Pokaż sumę wyświetlonych wg. Tego kryterium wartości atrybutów 'unitsInStock'

```
select sum(unitsinstock) from suppliers s, products p where
s.supplierid=p.supplierid AND country ='sweden'
```

6. Wyświetl najmniejszą, największą i średnią wartość atrybutu unitsInStock tabeli Products.

```
select min(unitsinstock) from products
select max(unitsinstock) from products
select avg(unitsinstock) from products
```

7. Wyświetl atrybut 'country' dla wszystkich rekordów tabeli 'customers'

```
select country from customers
```

8. Zlicz ilu klientów pochodzi z każdego z krajów:

```
select country, count(*) from customers group by country
```

9. Policz łączną sumę cen produktów należących do każdej z kategorii (tak samo jak powyżej, ale policz sumę wartości elementów a nie ich liczbę)

```
select categoryid, sum(unitprice) from products group by
categoryid
```

10. Zrób dokładnie to samo, kolumnie wynikowej nadaj nazwę 'price sum'

```
select categoryid, sum(unitprice) [price sum] from products group
by categoryid
```

11. Zlicz ile zamówień realizowanych jest przez każdego z pracowników (analogicznie do p.8, ale potrzebna jest dodatkowa tabela - aby wyświetlić nazwisko, poza tabelą 'orders', wymagana jest tabela 'employees')

```
select lastname, count(*) from employees e, orders o where
e.employeeid=o.employeeid group by lastname
```

Ćwiczenia.

Wykorzystaj bazę danych 'pubs'.

1. Wyświetl wszystkich autorów ze stanu 'CA'
2. Policz wszystkich autorów ze stanu 'CA' i nazwij kolumnę wynikową 'liczba'
3. Wyświetl atrybuty title_id oraz qty z tabeli 'sales'
4. Pokaż liczbę egzemplarzy każdego tytułu (w tabeli 'sales'), którego typ (w tabeli 'titles') to 'business'
5. Pokaż sumę w/w egzemplarzy
6. Wyświetl najmniejszą, największą i średnią liczbę egzemplarzy dla książek
7. Wyświetl nazwę stanu pochodzenia dla każdego autora
8. Zlicz ilu autorów pochodzi z każdego ze stanów
9. Policz sumę sprzedanych egzemplarzy dla każdego 'title_id' na podstawie tabeli 'sales'
10. Wyświetl taki sam wynik z nazwą 'book qty' dla kolumny wynikowej.
11. Zlicz ilu pracowników zatrudnionych jest u każdego z wydawców i pogrupuj wg. nazw wydawców (tabele 'employee' oraz 'publishers').