

# Podstawy języka SQL



# Rodzaje baz danych

- **Relacyjne**
- **Nierelacyjne**
- **Obiektowe**
- **Kartotekowe**
- **Hierarchiczne**

# Co to jest SQL?

- **Structured**
- **Query**
- **Language**

Język pozwalający komunikować się z silnikiem bazy danych poprzez proste zapytania/komendy

# Słownik pojęć

- Baza danych
- Schemat
- Tabela
- Wiersz
- Kolumna
- Typ danych
- Klucze (Główny, Obcy)

## Podstawowe zasady:

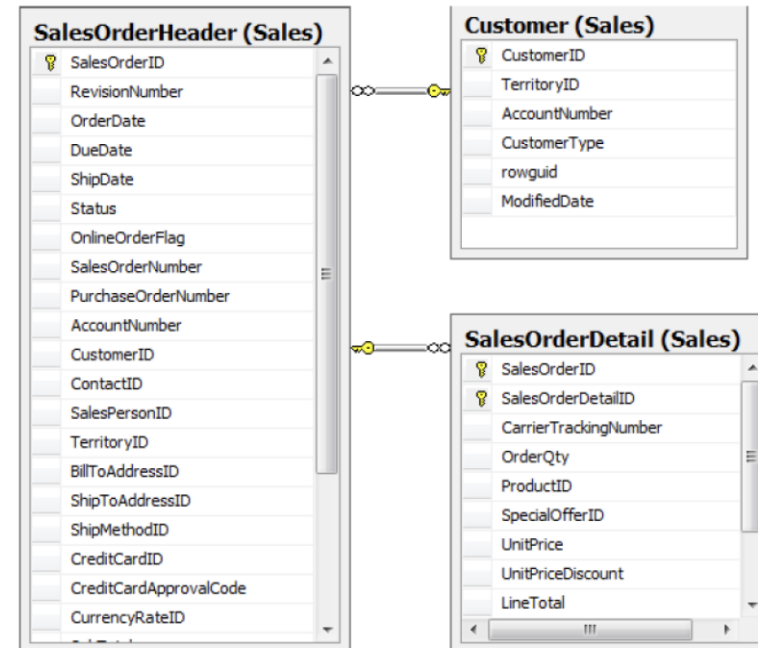
- **Każdy wiersz w Tabeli powinien być jednoznacznie identyfikowany (PK)**
- **W danej Kolumnie występują zawsze te same typy danych**
- **Domyślnym schematem Bazy danych jest schemat dbo.**

# Klucz główny – Primary KEY

- Unikatowy w całej Tabeli
- Identyfikuje każdy rekord (wiersz) w Tabeli
- Nie może zawierać wartości null
- Każda tabela **może posiadać max. 1 klucz główny**

# Klucz obcy – FOREIGN KEY

- Wskazuje na klucz główny **innej** Tabeli
- Wstawiana wartość **musi być** jedną z wartości klucza głównego, z tabeli powiązanej (Np. nowe zamówienie musi zawierać ID klienta, który już istnieje w tabeli Customer)



# Kącik pojęć na rozmowę rekrutacyjną

- Czym jest Transakcja
- Czym jest Rollback
- Co to index
- ACID



# ACID

**A – Atomowość.** To zapewnienie, że operacja lub zestaw operacji objętych **transakcją** zostanie **wykonana w całości** lub **anulowana**.

**C – Consistency.** Spójność. Baza musi być spójna – „dobra” → nie można np. przelać pieniędzy jak saldo po operacji będzie miało wartość ujemną

**I – Isolation.** Izolacja oznacza, że jednocześnie wykonywane transakcje są od siebie odizolowane - nie mogą mieć na siebie wpływu. Jest to szczególnie ważne w sytuacji dostępu do tego samego zasobu (tej samej tabeli) w przypadku dostępu do różnych obiektów konflikt nie będzie występował.

**D – Durability.** Trwałość to obietnica, że po udanym zaakceptowaniu transakcji zapisane dane nie zostaną zapomniane (bez względu na sytuacje awaryjne - awaria sprzętu / bazy). Można to uzyskać przez zapisanie danych na dyski (SSD / HDD) + dodanie operacji do dziennika logów WAL (Write Ahead Log - operacja jest najpierw zapisywana do dziennika kolejno wykonywana) lub przez replikację bazy danych.



# Transakcja

Jest to mechanizm w bazach danych pozwalający na zarządzanie grupą zapytań. Można porównać ją do pudełka do którego wkładamy produkty i wysyłamy do odbiorcy – jeżeli coś jest uszkodzone zwraca nam on całe pudełko z jego zawartością. Możemy oczywiście transakcją zarządzać. Niesie to ze sobą wiele możliwości. Pozwala to nam samemu stwierdzić, czy po wystąpieniu określonego błędu to co się wykonało ma zostać zapisane, czy wycofane. Transakcje kończymy słowem kluczowym COMMIT

## Rollback

Podobnie jak COMMIT, **ROLLBACK** jest również instrukcją SQL i sygnalizuje, że transakcja **nie** zakończyła się **pomyślnie**. Dlatego transakcja jest **przerywana**, aby cofnąć zmiany dokonane przez transakcję. Po wykonaniu ROLLBACK nie zachodzi żadna modyfikacja wykonana przez bieżącą transakcję. ROLLBACK transakcji staje się konieczny, jeśli wystąpi błąd podczas wykonywania transakcji. Błąd może polegać na awarii systemu, zaniku zasilania, błędzie w instrukcjach transakcji, awarii systemu. W przypadku awarii zasilania lub awarii systemu, ROLLBACK występuje, gdy system zostanie ponownie uruchomiony. ROLLBACK może wystąpić tylko wtedy, gdy COMMIT nie jest jeszcze wykonany.

# Indeks

*Indeks* jest to struktura danych na dysku umożliwiająca szybkie wyszukiwanie danych w bazie danych na podstawie wartości klucza wyszukiwania takiego jak np. *nazwisko osoby*. Indeks w bazie danych ma takie samo znaczenie jak skorowidz (indeks) w książce! **Korzyść** wydajnościowa ze stosowania indeksów **jest największa w przypadku dużych tabel** (zawierających najwięcej rekordów) **oraz zapytań, które wykonywane są najczęściej**. Można się kierować prostą zasadą, polegającą na tym, że nie tworzymy indeksu jeżeli nie jesteśmy przekonani, że faktycznie będziemy z niego korzystać.

## Kiedy nie używać indeksów?

Należy pamiętać, że indeks drastycznie spowalnia dodawanie, modyfikowanie i usuwanie danych, ponieważ indeksy muszą być aktualizowane za każdym razem, gdy tabela ulega nawet najmniejszej modyfikacji. Najlepszą praktyką jest dodanie indeksu dla wartości, które są często używane do wyszukiwania, ale nie ulegają częstym zmianom.

# RDBMS

*Relational Database Management System*  
*(System zarządzania bazą danych )*

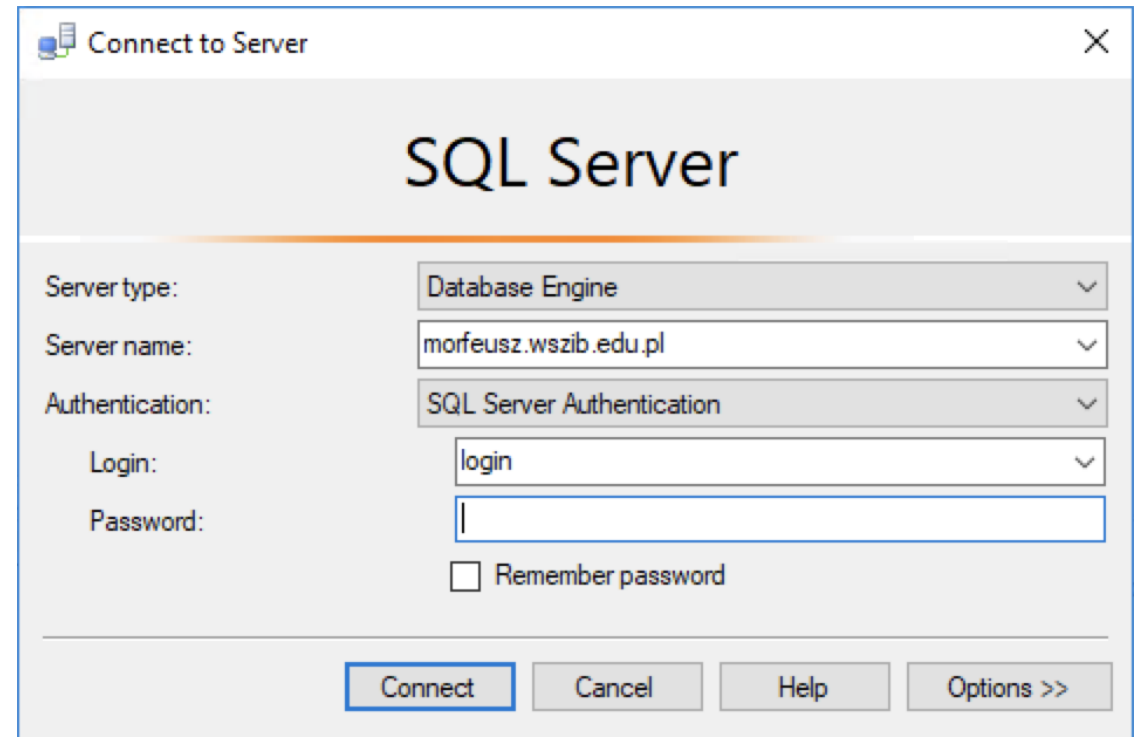
Oprogramowania otwarte	Oprogramowania zamknięte
MySQL	Oracle
<b>PostgreSQL</b>	<b>Microsoft SQL Server</b>
SQLite	Microsoft Access

*Programy służące do korzystania z bazy danych opartej na relacyjnym modelu*

# Narzędzia Pracy

- SQL Server Management Studio

System dedykowany dla administratorów i programistów baz danych



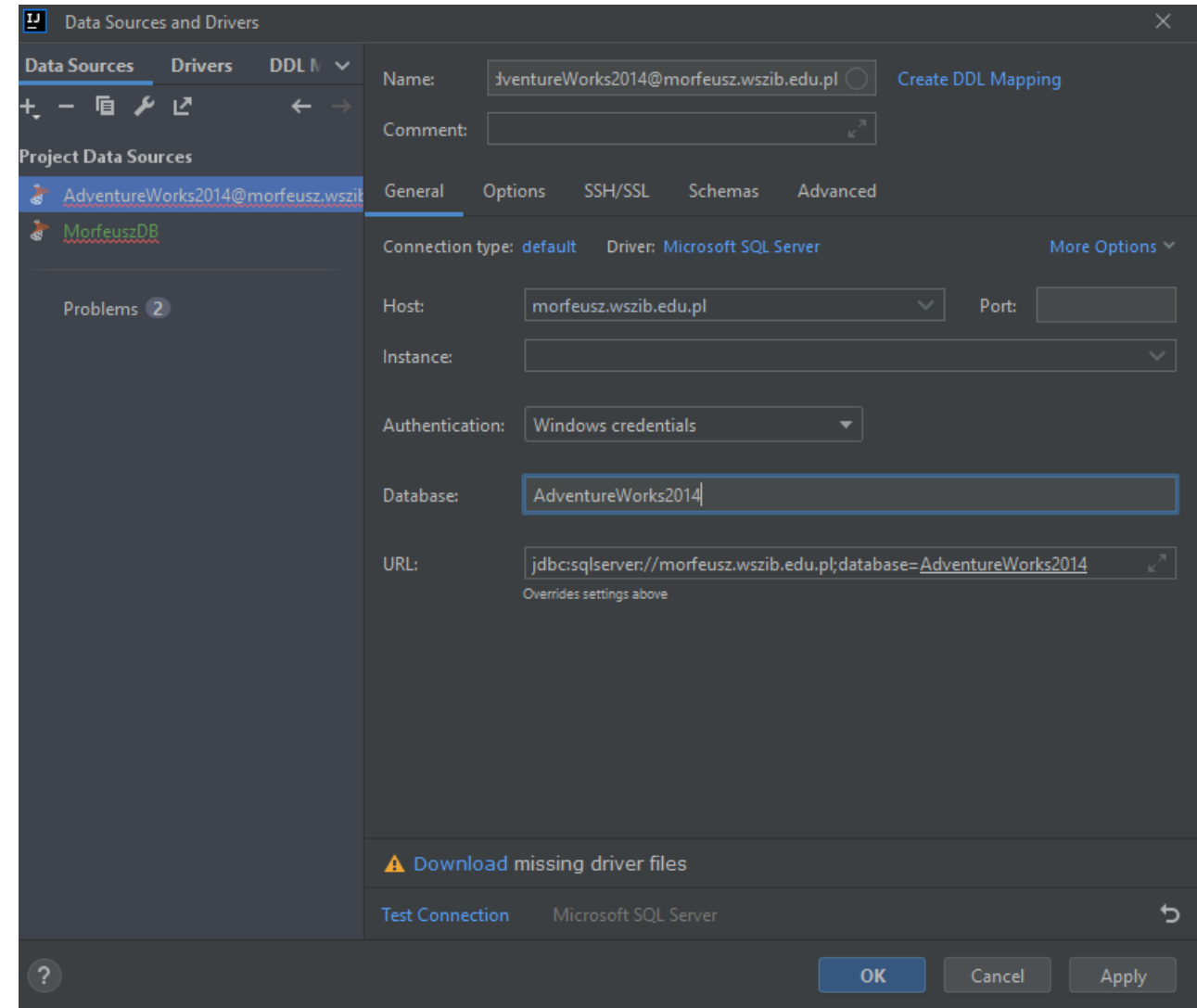
The screenshot shows the 'Connect to Server' dialog box. The title bar reads 'Connect to Server' with a close button. The main heading is 'SQL Server'. The dialog contains the following fields and controls:

- Server type:** A dropdown menu set to 'Database Engine'.
- Server name:** A text box containing 'morfiesz.wszib.edu.pl'.
- Authentication:** A dropdown menu set to 'SQL Server Authentication'.
- Login:** A text box containing 'login'.
- Password:** An empty text box.
- ☐ Remember password

At the bottom, there are four buttons: 'Connect' (highlighted with a blue border), 'Cancel', 'Help', and 'Options >>'.

# Narzędzia Pracy

- SQL Server Management Studio
- **IntelliJ IDEA/ PyCharm**



# Adventure Works

[https://akela.mendelu.cz/~jprich/vyuka/db2/AdventureWorks2008\\_db\\_diagram.pdf](https://akela.mendelu.cz/~jprich/vyuka/db2/AdventureWorks2008_db_diagram.pdf)

# Konstrukcja zapytania SELECT

SELECT select\_list

[ column\_list ] [ \* ]

FROM table\_source

[ WHERE search\_condition ]

[ GROUP BY group\_by\_expression ]

[ HAVING search\_condition ]

[ ORDER BY order\_expression

[ ASC | DESC ] ]

—

# DEMO TIME

---





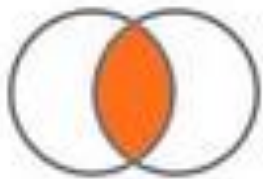
# Łączenie tabel

- INNER JOIN
- LEFT OUTER JOIN (LEFT JOIN)
- RIGHT OUTER JOIN (RIGHT JOIN)
- FULL OUTER JOIN (FULL JOIN)

# Łączenie tabel

- **INNER JOIN**
- LEFT OUTER JOIN (LEFT JOIN)
- RIGHT OUTER JOIN (RIGHT JOIN)
- FULL OUTER JOIN (FULL JOIN)

## INNER JOIN



Only the things that match on the left AND the right

```
SELECT *  
FROM TABLE_1  
INNER JOIN TABLE_2  
ON TABLE_1.KEY = TABLE_2.KEY
```

Źródło: <https://towardsdatascience.com/take-your-sql-from-good-to-great-part-3-687d797d1ede>

# Łączenie tabel

- INNER JOIN
- **LEFT OUTER JOIN (LEFT JOIN)**
- RIGHT OUTER JOIN (RIGHT JOIN)
- FULL OUTER JOIN (FULL JOIN)

## LEFT JOIN



Everything on the left  
+  
anything on the right that  
matches

```
SELECT *  
FROM TABLE_1  
LEFT JOIN TABLE_2  
ON TABLE_1.KEY = TABLE_2.KEY
```

Źródło: <https://towardsdatascience.com/take-your-sql-from-good-to-great-part-3-687d797d1ede>

# łączenie tabel

- INNER JOIN
- LEFT OUTER JOIN (LEFT JOIN)
- **RIGHT OUTER JOIN (RIGHT JOIN)**
- FULL OUTER JOIN (FULL JOIN)

## RIGHT JOIN



Everything on the right  
+  
anything on the left that matches

```
SELECT *  
FROM TABLE_1  
RIGHT JOIN TABLE_2  
ON TABLE_1.KEY = TABLE_2.KEY
```

Źródło: <https://towardsdatascience.com/take-your-sql-from-good-to-great-part-3-687d797d1ede>

# Łączenie tabel

- INNER JOIN
- LEFT OUTER JOIN (LEFT JOIN)
- RIGHT OUTER JOIN (RIGHT JOIN)
- **FULL OUTER JOIN (FULL JOIN)**



```
SELECT *  
FROM TABLE_1  
OUTER JOIN TABLE_2  
ON TABLE_1.KEY = TABLE_2.KEY
```

Źródło: <https://towardsdatascience.com/take-your-sql-from-good-to-great-part-3-687d797d1ede>

# Używanie Subqueries

- zagnieżdżone zapytania (zapytanie wewnątrz innego zapytania)
- podzapytanie można używać w części WHERE zapytania i na liście kolumn (selectlist)

```
1.  SELECT column-names
2.      FROM table-name1
3.      WHERE value IN (SELECT column-name
4.                          FROM table-name2
5.                          WHERE condition)
```

—

# DEMO TIME

---



# Grupowanie danych

Funkcje agregujące przykłady	Funkcje skalarne przykłady
COUNT	LEFT
SUM	ISNULL
AVG	SUBSTRING
MAX,MIN	YEAR



# Grupowanie danych – GROUP BY

- Pozwala zgrupować dane po kolumnach
- Występuje wraz z funkcją agregującą
- Chcąc filtrować dane po zgrupowaniu należy zamiast WHERE użyć HAVING
- Kolumny po których chcemy wykonać grupowanie muszą znajdować się po słowie SELECT

Zamówienia klientów którzy złożyli 5 lub więcej zamówień.

```
SELECT CustomerID, COUNT(*) AS CntOrders
FROM Sales.SalesOrderHeader
-- WHERE COUNT(*) >= 5 << wrong
GROUP BY CustomerID
HAVING COUNT(*) >= 5
```

	CustomerID	CntOrders
1	2	8
2	3	12
3	4	8
4	5	8
5	9	7
6	10	8
7	12	8
8	14	8
9	16	8
10	17	12

—

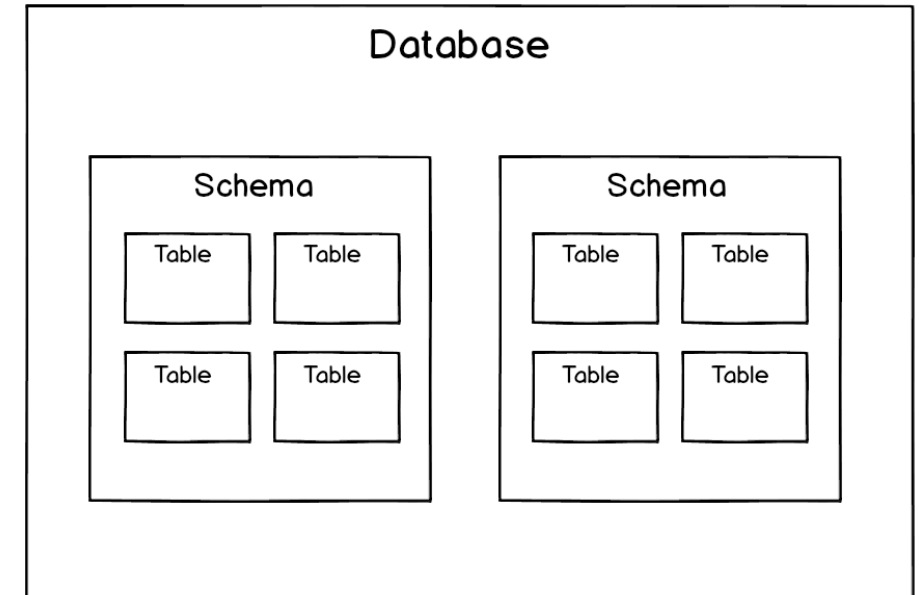
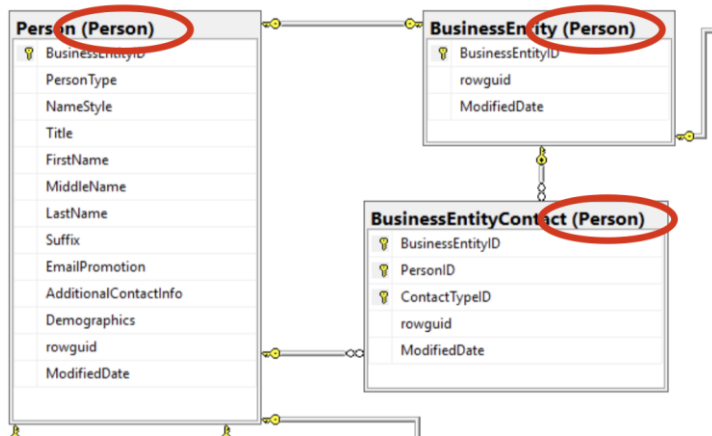
# DEMO TIME

---



# Przestrzenie nazw - Schemas

- kontenery na obiekty bazodanowe (tabele, widoki, funkcje..)
- tworzone przez polecenie **CREATE SCHEMA**
- użytkownik ma domyślny schemat
- wbudowane schematy: *dbo*, *guest*, *sys*



# Schema – Rozwiązywanie nazwy

Jeśli nazwa schematu jest pominięta poniższe zasady decydują o tym, który obiekt będzie odczytany:

- najpierw obiekt (np. tabela) jest szukana w domyślnym schemacie użytkownika
- jeśli nie zostanie odnaleziony, sprawdzany jest schemat *dbo*

# Tworzenie Tabel

- Podczas tworzenia **muszą** zostać określone kolumny
- Powinien zostać określony klucz główny
- Można określić dodatkowe ograniczenia na kolumnach

```
CREATE TABLE table_name (  
    column1 datatype,  
    column2 datatype,  
    column3 datatype,  
    ....  
);
```

```
CREATE TABLE new_table_name  
AS  
    SELECT column1, column2,...  
    FROM existing_table_name  
    WHERE ....;
```

# Modyfikowanie Tabel

- Używamy słowa kluczowego ALTER TABLE
- Gdy dodajemy nowe kolumny do istniejącej tabeli trzeba pamiętać o zapewnieniu wartości domyślnej dla istniejących rekordów – lub pozwolić kolumnie na wartości null

```
ALTER TABLE table_name  
DROP COLUMN column_name;
```

```
ALTER TABLE table_name  
ADD column_name datatype;
```

```
ALTER TABLE table_name  
RENAME COLUMN old_name to  
new_name;
```

```
ALTER TABLE table_name  
ALTER COLUMN column_name datatype;
```

# Usuwanie Tabeli

- Słowo kluczowe: DROP TABLE
- Tabele do których odwołują się inne obiekty (poprzez klucze obce) nie mogą zostać usunięte
- Wraz z usunięciem tabeli kasowane są wszystkie uprawnieni, indeksy i dane

# Typy danych w SQL Server

- Liczbowe (int, bigint, float)
- Tekstowe (char, nchar, varchar, nvchar)
- Data i czas (datetime2, datetime)
- Binarne (blob)
- Pozostałe

Liczbowe	INT	-2147483648 to 2147483647
	BIGINT	-9223372036854775808 to 9223372036854775807
	DOUBLE	Liczna zmiennoprzecinkowa
Tekstowe	CHAR(SIZE)	Ciąg znaków o STAŁEJ długości (może zawierać litery, cyfry i znaki specjalne)
	VARCHAR(SIZE)	Ciąg o ZMIENNEJ długości (może zawierać litery, cyfry i znaki specjalne). Parametr size definiuje rozmiar ciągu w bajtach.



# Ograniczanie wartości kolumn - Nullability

- określa czy wartości muszą być podawane
- klucze główne (PKs) nie mogą zawierać **NULL**
- wartości **NULL** mogą być wstawiane jawnie lub niejawnie (przez pominięcie nazwy kolumny w instrukcji INSERT)

# Autoinkrementacja

- Właściwość kolumny
- Autoinkrementowana wartość **nie może** być wprowadzana ręcznie
- Podczas usuwania wierszy z kolumną oznaczoną jako **IDENTITY** mogą pojawić się przerwy w ciągłości numeracji

—

# DEMO TIME

---

