


# Zależność referencyjna

Supplier



| <u>SUPNR</u> | SUPNAME          | SUPADDRESS                  | SUPCITY        | SUPSTATUS |
|--------------|------------------|-----------------------------|----------------|-----------|
| 21           | Deliwines        | 240, Avenue of the Americas | New York       | 20        |
| 32           | Best Wines       | 660, Market Street          | San Francisco  | 90        |
| <b>37</b>    | <b>Ad Fundum</b> | <b>82, Wacker Drive</b>     | <b>Chicago</b> | <b>95</b> |
| 52           | Spirits & co.    | 928, Strip                  | Las Vegas      | NULL      |
| 68           | The Wine Depot   | 132, Montgomery Street      | San Francisco  | 10        |
| 69           | Vinos del Mundo  | 4, Collins Avenue           | Miami          | 92        |

Supplies

| <u>SUPNR</u> | <u>PRODNR</u> | PURCHASE_PRICE | DELIV_PERIOD |
|--------------|---------------|----------------|--------------|
| 37           | 0178          | 16.99          | 4            |
| 37           | 0185          | 32.99          | 3            |
| 37           | 0468          | 14.00          | 1            |
| 37           | 0795          | 20.99          | 3            |

Purchase\_Order

| <u>PONR</u> | <u>PODATE</u> | <u>SUPNR</u> |
|-------------|---------------|--------------|
| 1511        | 2015-03-24    | 37           |
| 1513        | 2015-04-11    | 37           |
| 1523        | 2015-04-19    | 37           |
| 1577        | 2015-05-10    | 37           |
| 1594        | 2015-05-13    | 37           |

# Przykład usuwania kaskadowego

```
1 create table T (A int, B int, C int, primary key (A,B),
2               foreign key (B,C) references T(A,B) on delete cascade);
3 insert into T values (1,1,1);
4 insert into T values (2,1,1);
5 insert into T values (3,2,1);
6 insert into T values (4,3,2);
7 insert into T values (5,4,3);
8 insert into T values (6,5,4);
9 insert into T values (7,6,5);
10 insert into T values (8,7,6);
11
```

|   | a | b | c |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 2 | 2 | 1 | 1 |
| 3 | 3 | 2 | 1 |
| 4 | 4 | 3 | 2 |
| 5 | 5 | 4 | 3 |
| 6 | 6 | 5 | 4 |
| 7 | 7 | 6 | 5 |
| 8 | 8 | 7 | 6 |

```
1 delete from T where A=1;
```

|  | a | b | c |
|--|---|---|---|
|--|---|---|---|

# Przykład usuwania kaskadowego – c.d.

**create table** *manager*

(*employee\_ID* **char**(20) **primary key**,

*manager\_id* **char**(20),

**foreign key** (*manager\_id*) **references** *manager*(*employee\_ID*)  
**on delete cascade**)

Każdy pracownik ma co najwyżej jednego menadżera

FK: Każdy menadżer też jest pracownikiem

Co się stanie gdy krotka w relacji *manager* zostanie usunięta

# Przypisywanie nazw do ograniczeń

- ❑ **create table** *instructor* (  
    *ID*            **char**(5),  
    *name*         **varchar**(20) **not null**,  
    *dept\_name* **varchar**(20),  
    *salary*      **numeric**(8,2), **check** (*salary*>29000),  
    *salary*      **numeric**(8,2), **constraint minisalary check** (*salary*>29000),  
    **primary key** (*ID*),  
    **foreign key** (*dept\_name*) **references** *department*);
- ❑ usuwanie: **alter table** *instructor* **drop constraint minisalary**

# Blokowanie warunków integralnościowych

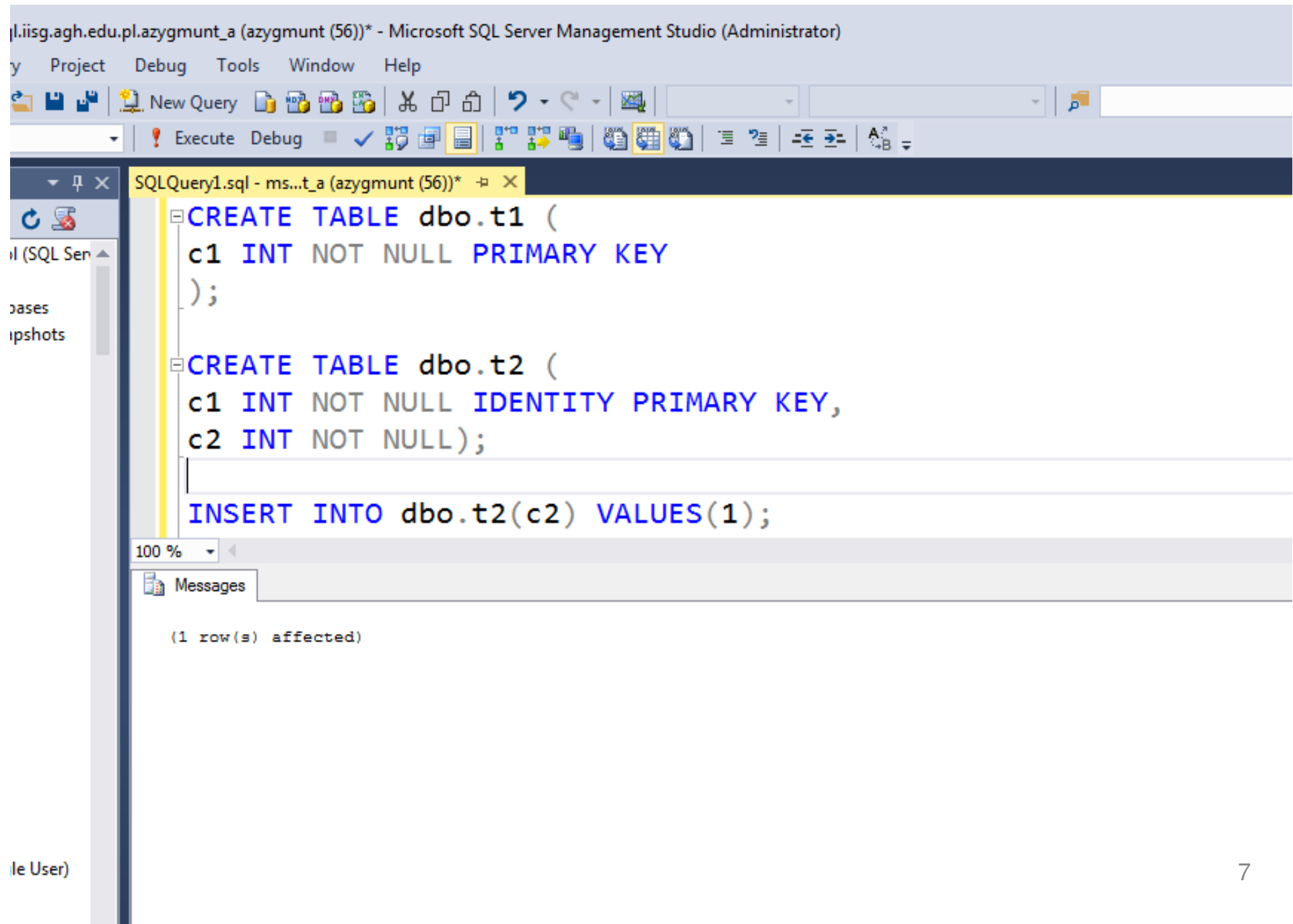
- Przy dodawaniu nowego warunku do istniejącej tabeli automatyczne sprawdzanie, czy istniejące dane spełniają warunki
  - Blokowanie sprawdzania ze względów wydajnościowych, np. przy ładowaniu dużej ilości danych
- Blokowanie sprawdzania warunków
  - na istniejących danych
  - przy ładowaniu nowych danych

# Blokowanie sprawdzanie warunków na istniejących danych

- ❑ Zastosowanie do warunków CHECK i FOREIGN KEY – inne warunki muszą być usunięte i utworzone od nowa
- ❑ Użycie opcji WITH NOCHECK gdy dodawany nowy warunek
- ❑ Można zmienić istniejące dane przed dodaniem warunku

```
USE Northwind
ALTER TABLE dbo.Employees
WITH NOCHECK
    ADD CONSTRAINT FK_Employees_Employees
    FOREIGN KEY (ReportsTo)
    REFERENCES dbo.Employees(EmployeeID)
```

# Przykład blokowania constraintów na istniejących danych



The screenshot displays the Microsoft SQL Server Management Studio (SSMS) interface. The title bar indicates the connection is to 'pl.iisg.agh.edu.pl.azygmunt\_a (azygmunt (56))\* - Microsoft SQL Server Management Studio (Administrator)'. The menu bar includes 'File', 'Project', 'Debug', 'Tools', 'Window', and 'Help'. The toolbar contains various icons for file operations, query execution, and formatting. The 'SQLQuery1.sql - ms...t\_a (azygmunt (56))\*' tab is active, showing the following SQL script:

```
CREATE TABLE dbo.t1 (  
  c1 INT NOT NULL PRIMARY KEY  
);  
  
CREATE TABLE dbo.t2 (  
  c1 INT NOT NULL IDENTITY PRIMARY KEY,  
  c2 INT NOT NULL);  
  
INSERT INTO dbo.t2(c2) VALUES(1);
```

The 'Messages' pane at the bottom shows the execution result: '(1 row(s) affected)'.

# Przykład blokowania constraintów na istniejących danych (1)

## constraint nieaktywny

g.agh.edu.pl.azygmunt\_a (azygmunt (56))\* - Microsoft SQL Server Management Studio (Administrator)

Project Debug Tools Window Help

New Query

Execute Debug

SQLQuery1.sql - ms...t\_a (azygmunt (56))\*

```
ALTER TABLE dbo.t2 WITH NOCHECK
ADD CONSTRAINT FK_t2_t1 FOREIGN KEY (c2)
REFERENCES dbo.t1(c1);

SELECT [name], type_desc, is_disabled, is_not_trusted
FROM sys.foreign_keys
WHERE parent_object_id = OBJECT_ID('dbo.t2');
```

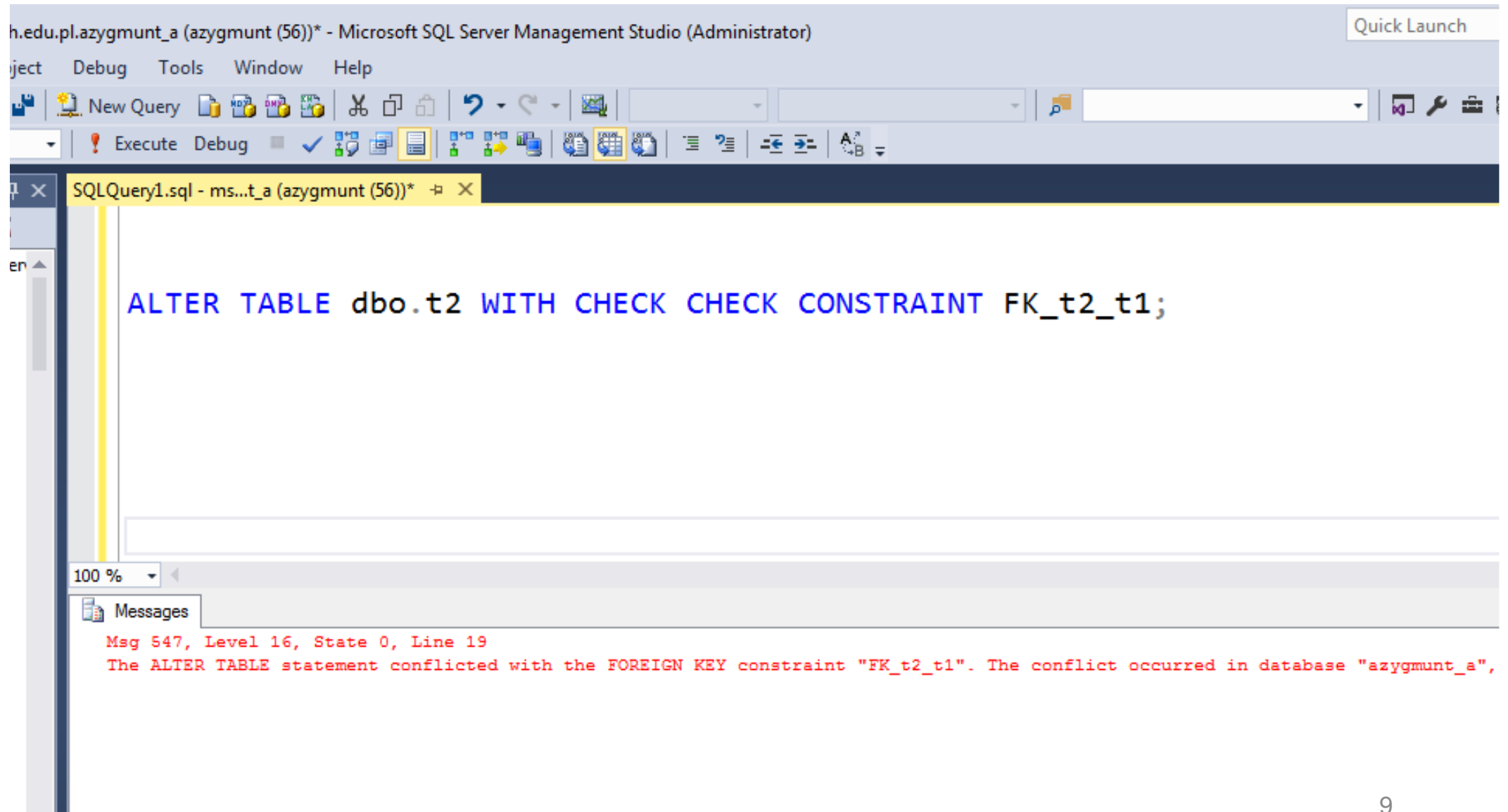
100 %

Results Messages

|   | name     | type_desc              | is_disabled | is_not_trusted |
|---|----------|------------------------|-------------|----------------|
| 1 | FK_t2_t1 | FOREIGN_KEY_CONSTRAINT | 0           | 1              |

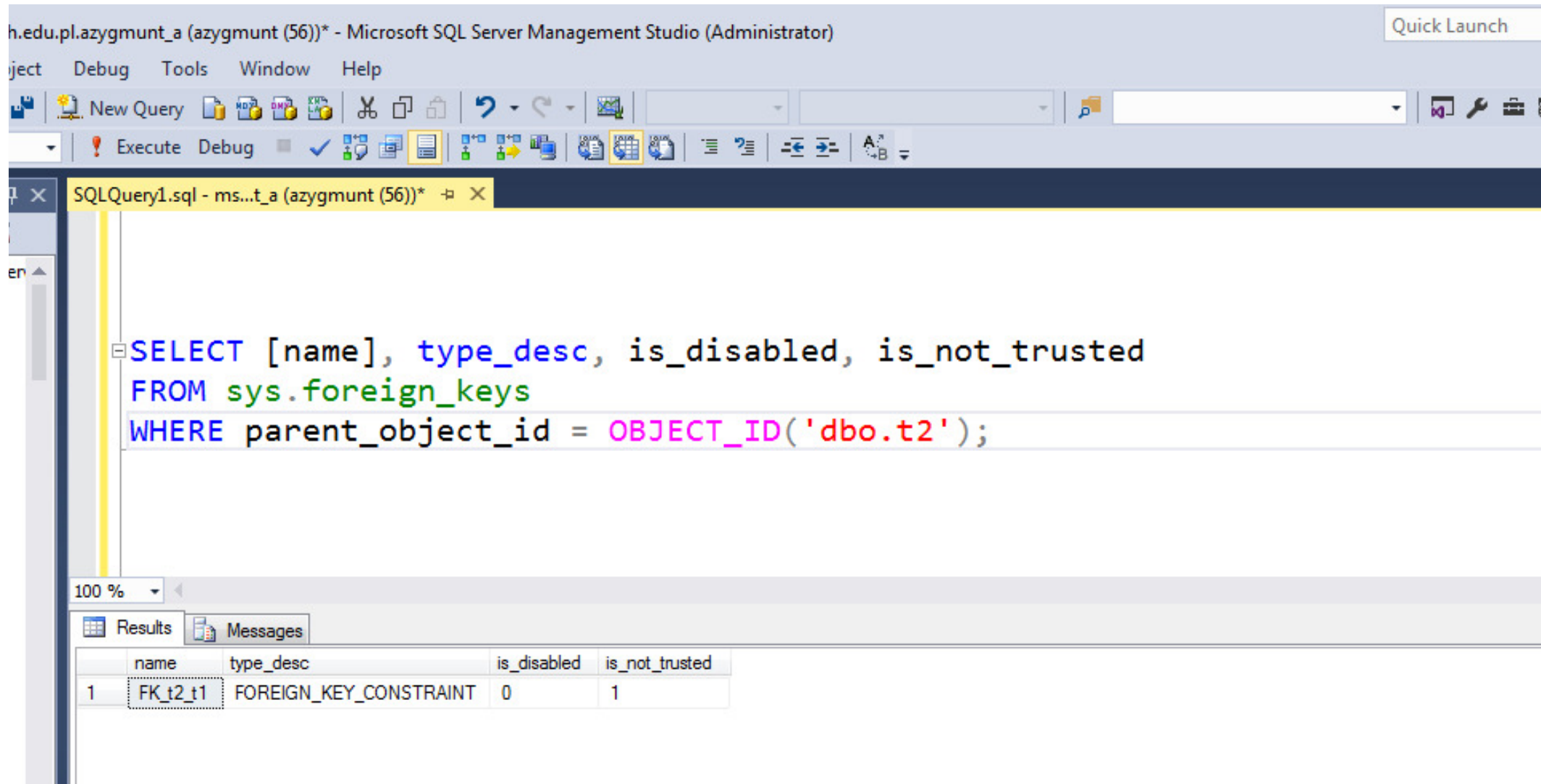


# Przykład blokowania constraintów na istniejących danych (2) próba uaktywnienia



# Przykład blokowania constraintów na istniejących danych (3)

## constraint w stanie nieaktywnym



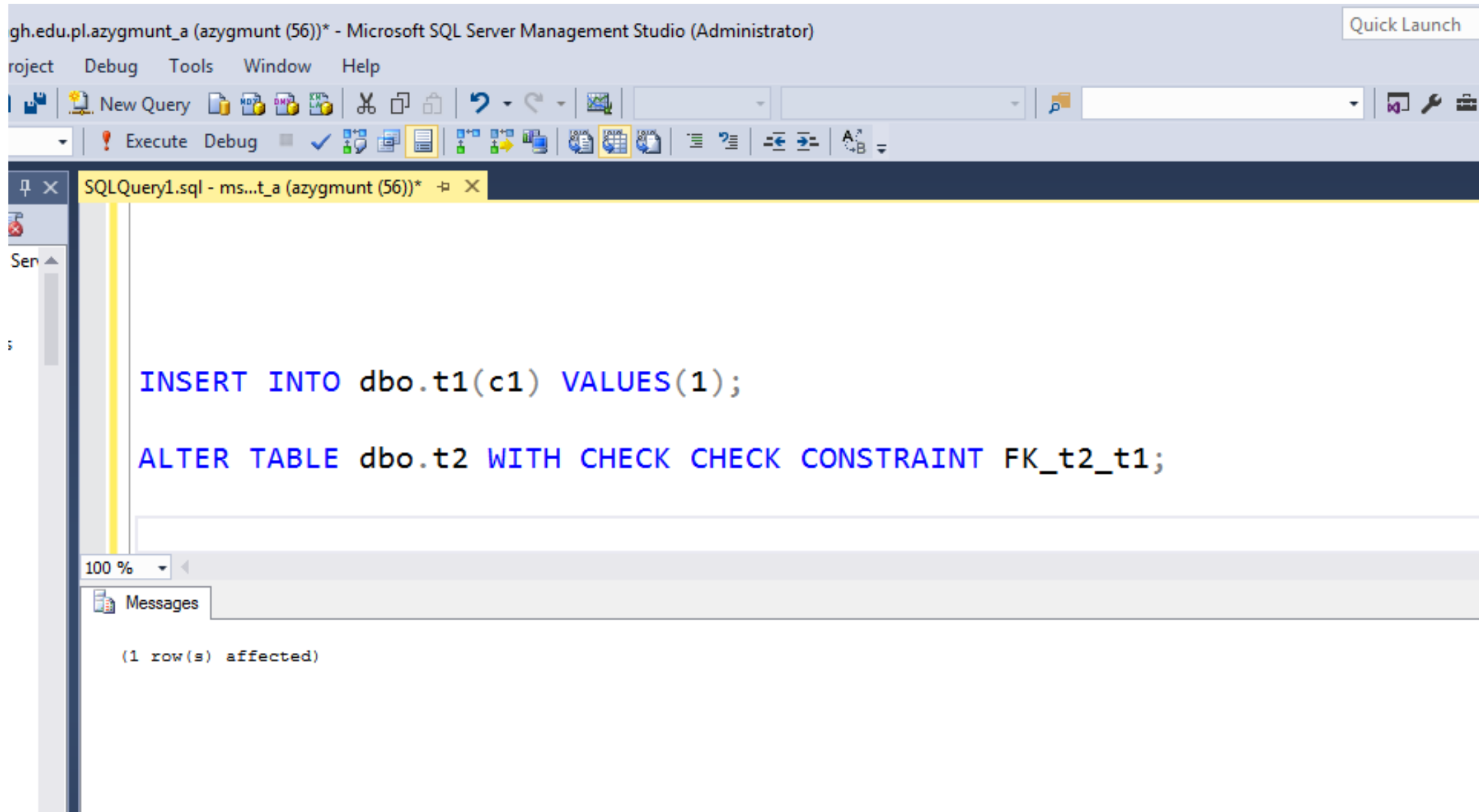
The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. The title bar indicates the connection is to 'h.edu.pl.azygmunt\_a (azygmunt (56))\* - Microsoft SQL Server Management Studio (Administrator)'. The menu bar includes 'Object', 'Debug', 'Tools', 'Window', and 'Help'. The toolbar contains various icons for file operations, execution, and formatting. The main query editor displays the following SQL query:

```
SELECT [name], type_desc, is_disabled, is_not_trusted
FROM sys.foreign_keys
WHERE parent_object_id = OBJECT_ID('dbo.t2');
```

Below the query editor, the 'Results' tab is active, showing a table with the following data:

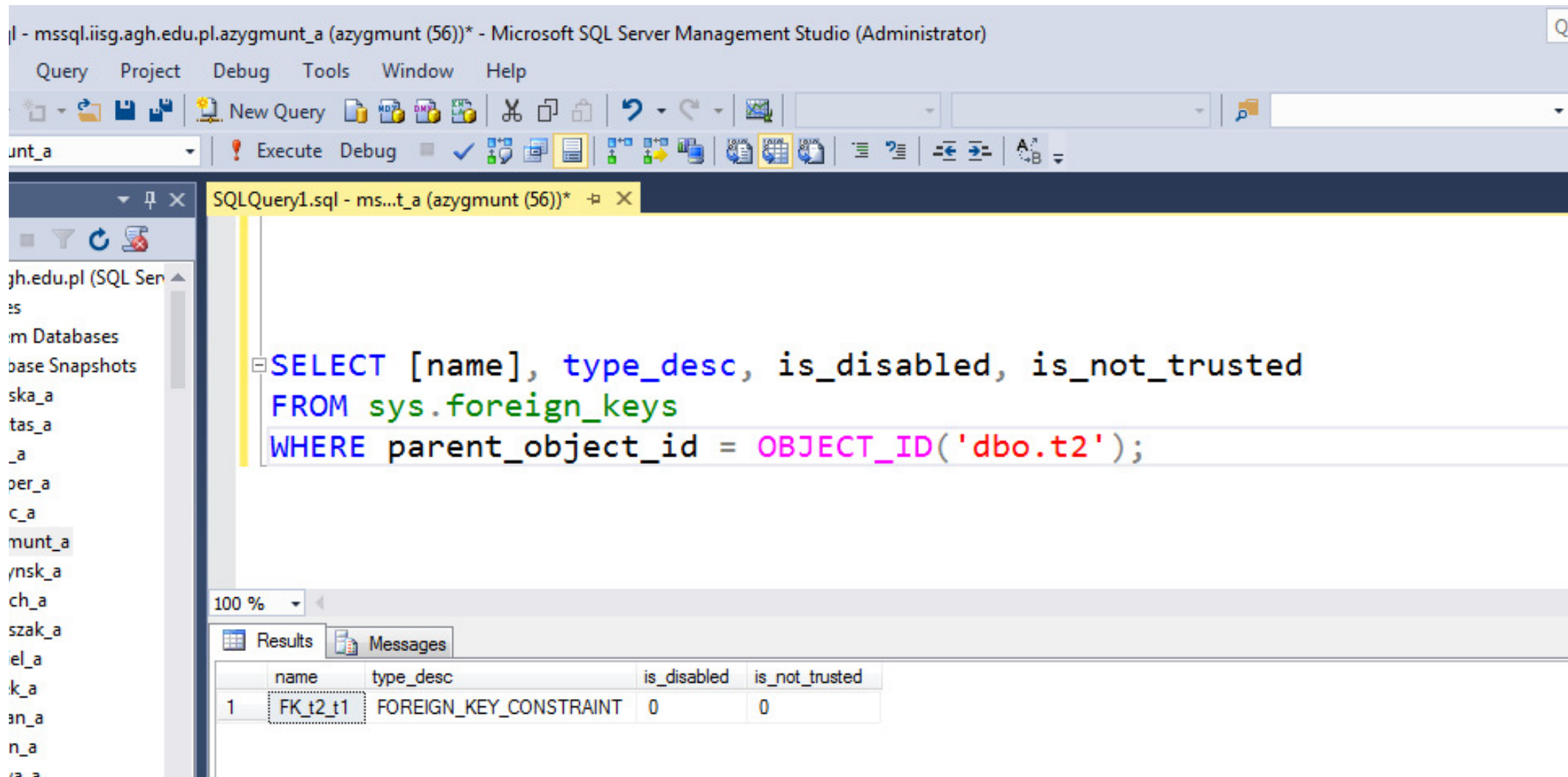
|   | name     | type_desc              | is_disabled | is_not_trusted |
|---|----------|------------------------|-------------|----------------|
| 1 | FK_t2_t1 | FOREIGN_KEY_CONSTRAINT | 0           | 1              |

# Przykład blokowania constraintów na istniejących danych (4) próba uaktywnienia c.d.



# Przykład blokowania constraintów na istniejących danych (5)

## constraint aktywny



The screenshot shows the Microsoft SQL Server Management Studio (Administrator) interface. The title bar indicates the connection is to 'mssql.iisg.agh.edu.pl.azygmunt\_a (azygmunt (56))\*'. The menu bar includes Query, Project, Debug, Tools, Window, and Help. The toolbar contains icons for New Query, Open, Save, Undo, Redo, and other standard database operations. The left sidebar shows the server tree with 'azygmunt\_a' selected. The main query editor displays the following SQL query:

```
SELECT [name], type_desc, is_disabled, is_not_trusted
FROM sys.foreign_keys
WHERE parent_object_id = OBJECT_ID('dbo.t2');
```

Below the query editor, the 'Results' tab is active, showing a table with the following data:

|   | name     | type_desc              | is_disabled | is_not_trusted |
|---|----------|------------------------|-------------|----------------|
| 1 | FK_t2_t1 | FOREIGN_KEY_CONSTRAINT | 0           | 0              |

# Blokowanie sprawdzanie constraintów przy ładowaniu nowych danych

- Zastosowanie do constraintów CHECK i FOREIGN KEY
- Używany gdy:
  - Dane są zgodne z constraintami
  - Ładujesz nowe dane, które nie są zgodne z constraintami
  - Potem należy uruchomić frazę modyfikującą dane zgodnie z definicją constrainta i uaktywnić constraint

```
USE Northwind
ALTER TABLE dbo.Employees
NOCHECK
    CONSTRAINT FK_Employees_Employees
```

- blokowanie wszystkich: *NOCHECK CONSTRAINT ALL*
- *ALTER TABLE dbo.Employees CHECK CONSTRAINT FK\_Employees\_Employees*

# Duże typy danych (Large-Object)

- Duże obiekty (zdjęcia, filmy, zdjęcia medyczne wysokiej rozdzielczości...) są przechowywane jako *large object (lob)*:
  - **blob**: binary large object – obiekt jest ogromną kolekcją nieinterpretowalnych danych binarnych (których interpretacja jest po stronie aplikacji, poza systemem bazodanowym)
  - **clob**: character large object – obiekt jest ogromną kolekcją danych znakowych
  - Jeżeli zapytanie zwraca duży obiekt, zwracany jest wskaźnik (nie sam obiekt), który jest wykorzystywany w języku macierzystym, w którym napisana jest aplikacja (np. JDBC pozwala na ściąganie dużego obiektu po kawałku)
- Np.:
  - book\_review* **clob**(10KB)
  - image* **blob**(10MB)
  - movie* **blob**(2GB)

# Typy zdefiniowane przez użytkownika (user-defined types) (SQL:1999)

- SQL wspiera dwie formy typów zdefiniowanych przez użytkownika:

- distinct types
- *structured data types* – złożone typy z zagnieżdżoną strukturą rekordów, tablicami i wielozbiorami

- **create type**

**create type** *Dollars* **as numeric** (12,2) **final**;

**create type** *Pounds* **as numeric** (12,2) **final**;

przypisanie wartości typu *Dollar* do *Pounds* błąd kompilacji

- **create table** *department*  
    (*dept\_name* **varchar** (20),  
    *building* **varchar** (15),  
    *budget* *Dollars*);

- wyrażenie (*department.budget*+20) spowoduje błąd
  - ▶ **cast** (*department.budget* **to** *numeric*(12,2))

- **drop type, alter type**

# Dziedziny (SQL-92)

## ❑ **create domain**

**create domain** *person\_name* **char**(30) **not null**

## ❑ **Typy i dziedziny** są podobne, ale:

- ❑ **dziedziny** mogą mieć ograniczenia, takie jak **not null** oraz **default**
- ❑ **typy** są przeznaczone nie tylko do specyfikowania typu atrybutu ale w proceduralnych rozszerzeniach SQL gdzie może nie być możliwości wymuszenia ograniczeń
- ❑ wartości jednej dziedziny mogą być przypisane do drugiej o ile zgadzają się typy podstawowe

## ❑ **create domain** *degree\_level* **varchar**(10)

**constraint** *degree\_level\_test*

**check** (**value in** ('Bachelors', 'Masters', 'Doctorate'));



# Wsparcie dla typów i dziedzin

- ❑ *PostgreSQL* – wspiera **create domain**, a **create type** – inna składnia i interpretacja
- ❑ *Microsoft SQL Server* - implementuje wersję **create type**, która wspiera dziedzinowe ograniczenia (podobne do **create domain**)
- ❑ *IBM DB2* – wspiera wersję **create type** (**create distinct type**), nie wspiera **create domain**
- ❑ *Oracle* – nie wspiera

# Generowanie unikalnych wartości klucza

- Systemy bazodanowe oferują automatyczne zarządzanie generowaniem unikatowych kluczy
  - różna syntaktyka
  - Oracle/DB2
    - ▶ *ID numer(5) generated always {by default} as identity*
    - ▶ **insert into** *instructor(name,dept\_name,salary)*  
**values**('Newproof','Comp.Sci',100000);
  - PostgreSQL
    - ▶ **serial**
  - MySQL
    - ▶ **auto\_increment**
  - SQL Server
    - ▶ **identity**

# Tworzenie rozszerzeń tabeli

- Aplikacje często wymagają stworzenia tabeli o podobnym schemacie jak istniejąca

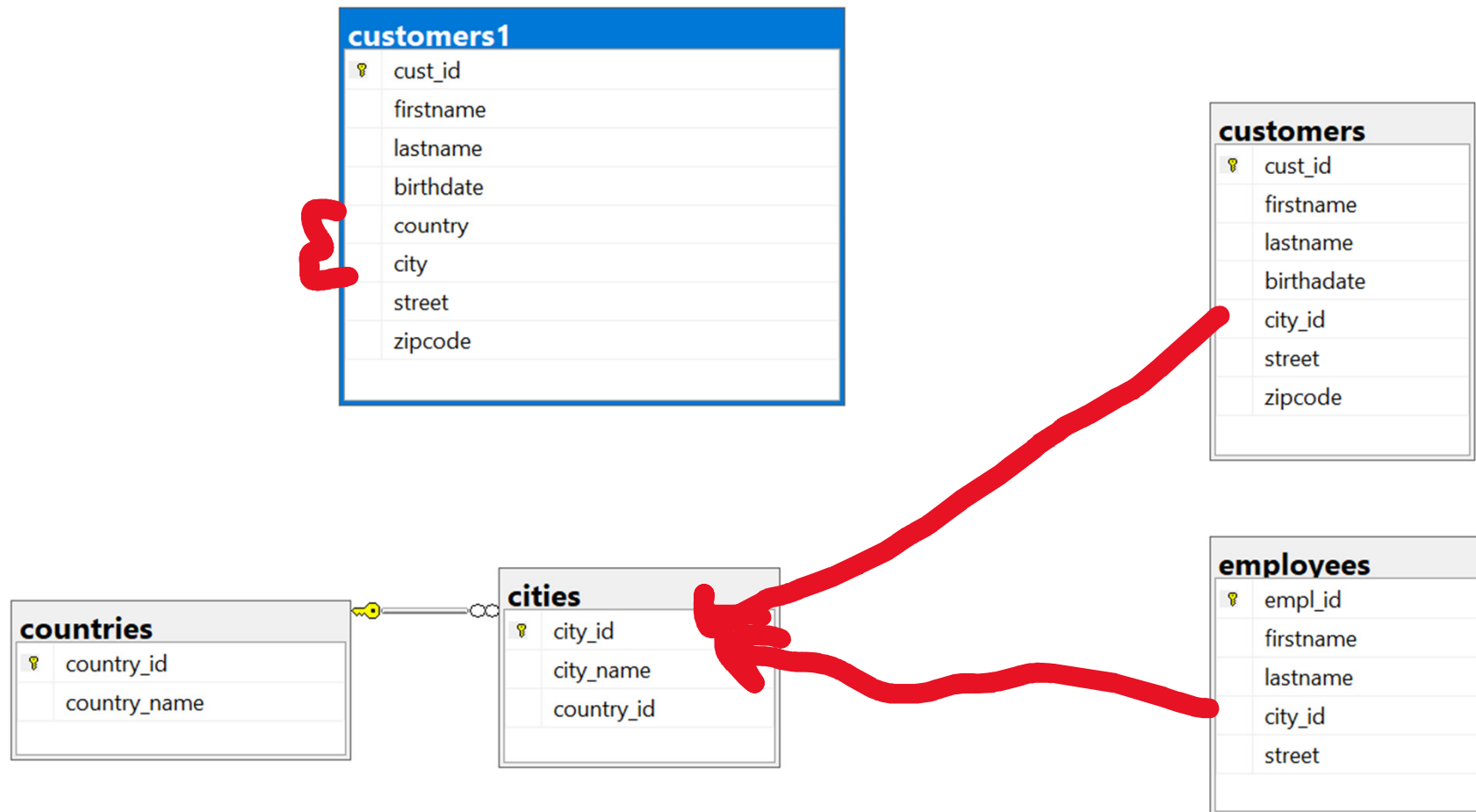
**create table** *temp\_instructor* **like** *instructor*

- Przy złożonych zapytaniach wygodnie jest przechowywać wynik zapytania jako nową tabelę (zazwyczaj temporalną), potrzebne 2 wyrażenia:
  - tworzenie tabeli z odpowiednimi kolumnami
  - wstawianie wyniku zapytania do tabeli
- Tworzenie tabeli zawierającej wyniki zapytania (SQL:2003):

**create table** *t1* **as**  
    (**select** \*  
    **from** *instructor*  
    **where** *dept\_name* = 'Music')  
    **with data**;

- nazwy i typy danych *t1* są wywnioskowane z wyniku zapytania

# Tworzenie słowników jako warunek integralności



## Widoki SQL

- Są częścią zewnętrznego modelu danych
- Widok definiowany jest za pomocą zapytania SQL, a jego zawartość jest generowana po wywołaniu widoku przez aplikację lub inne zapytanie
- Widok to wirtualna tabela bez fizycznych krotek
- Widoki pozwalają na logiczną niezależność danych, co czyni je kluczowym elementem w trójwarstwowej architekturze baz danych



## Widoki SQL

```
CREATE VIEW TOPSUPPLIERS  
AS SELECT SUPNR, SUPNAME FROM SUPPLIER  
WHERE SUPSTATUS > 50
```

```
CREATE VIEW TOPSUPPLIERS_SF  
AS SELECT * FROM TOPSUPPLIERS  
WHERE SUPCITY='San Francisco'
```



## Widoki SQL

```
CREATE VIEW ORDEROVERVIEW(PRODNR, PRODNAME,  
TOTQUANTITY)  
AS SELECT P.PRODNR, P.PRODNAME, SUM(POL.QUANTITY)  
FROM PRODUCT AS P LEFT OUTER JOIN PO_LINE AS POL  
ON (P.PRODNR = POL.PRODNR)  
GROUP BY P.PRODNR
```



## Widoki SQL

```
SELECT * FROM TOPSUPPLIERS_SF
```

```
SELECT * FROM ORDEROVERVIEW  
WHERE PRODNAME LIKE '%CHARD%'
```



## Widoki SQL

- Modyfikacja zapytań: RDBMS automatycznie modyfikuje zapytania, które odpytują widoki na zapytania dotyczące bazowych tabel podstawowych

```
CREATE VIEW TOPSUPPLIERS  
AS SELECT SUPNR, SUPNAME FROM SUPPLIER  
WHERE SUPSTATUS > 50  
SELECT * FROM TOPSUPPLIERS  
WHERE SUPCITY='San Francisco'
```

```
SELECT SUPNR, SUPNAME FROM SUPPLIER  
WHERE SUPSTATUS > 50 AND SUPCITY='San Francisco'
```



- Materializacja widoku: fizyczna tabela jest tworzona podczas pierwszego zapytania o widok
- Jeżeli relacje użyte w zapytaniu zostaną zaktualizowane, wynik zmaterializowanego widoku stanie się nieaktualny
- Potrzeba **utrzymania** widoku zmaterializowanego w stanie aktualnym
  - triggery, przyrostowe zarządzanie widokiem (niektóre DBMS wspierają)
  - jak dużo (całościowy, przyrostowy), kiedy (natychmiastowe, opóźnione)
- Brak standardu SQL, np. CREATE MATERIALIZED VIEW

- **zalety**

- przydatne dla aplikacji wymagających szybkich odpowiedzi na pytania wyliczających agregaty na dużych zbiorach danych
  - zagregowany wynik dużo mniejszy niż relacja, na której widok jest zbudowany
- dobre, gdy widok często wykorzystywany
- szybszy dostęp do złożonych złączeń

- **wady**

- koszty wydajnościowe zarządzania widokiem
- koszt przechowywania danych na dysku



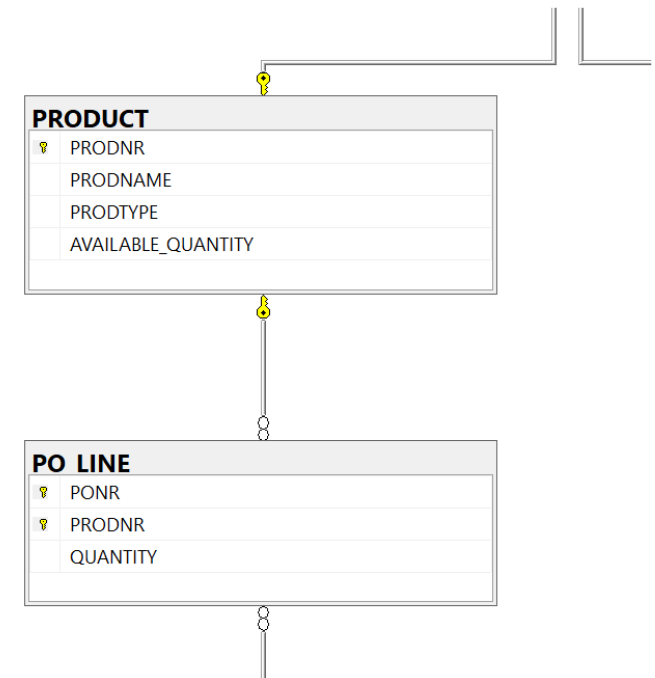
# Widoki SQL

- Większość implementacji SQL pozwala na modyfikacje tylko prostych widoków (=widoki modyfikowalne) – zapytanie tworzące widok musi spełniać wszystkie warunki (standard SQL):
  - Klauzula **from** ma tylko jedną relację
  - Klauzula **select** zawiera tylko nazwy atrybutów relacji i nie zawiera żadnych wyrażeń, agregatów ani specyfikacji **distinct**
  - Atrybut niewyspecyfikowany w klauzuli **select** może być ustawiony na null (brak ograniczenia **not null** ani nie należy do PK) (albo mieć wartości domyślne)
  - Zapytanie nie zawiera klauzul **group by** ani **having**
- Modyfikacja triggerami **INSTEAD OF**

```
CREATE VIEW ORDEROVERVIEW(PRODNR, PRODNAME,  
TOTQUANTITY)  
AS SELECT P.PRODNR, P.PRODNAME, SUM(POL.QUANTITY)  
FROM PRODUCT AS P LEFT OUTER JOIN PO_LINE AS POL  
ON (P.PRODNR = POL.PRODNR)  
GROUP BY P.PRODNR
```

```
UPDATE VIEW ORDEROVERVIEW  
SET TOTQUANTITY=10  
WHERE PRODNR= '0154'
```

**ERROR!**



- Opcja WITH CHECK sprawdza instrukcje UPDATE i INSERT pod kątem zgodności z definicją widoku

```
CREATE VIEW TOPSUPPLIERS  
AS SELECT SUPNR, SUPNAME FROM SUPPLIER  
WHERE SUPSTATUS > 50  
WITH CHECK OPTION
```

```
UPDATE TOPSUPPLIERS  
SET SUPSTATUS =20  
WHERE SUPNR='32'
```

**NOT OK!**

```
UPDATE TOPSUPPLIERS  
SET SUPSTATUS =80  
WHERE SUPNR='32'
```

**OK!**

# Usuwanie widoku

- ❑ DROP VIEW *nazwa\_widoku* [RESTRICT|CASCADE]
- ❑ RESTRICT - jeżeli istnieją inne widoki, zależne od usuwanego widoku to polecenie ma zakończyć się błędem (domyślnie)
- ❑ CASCADE - wszystkie widoki zależne od usuwanego są również usuwane.