Tematy zajęć:

* 17. Język definiowania danych (DDL). Tworzenie i modyfikacja tabel.
* 18. Język definiowania danych (DDL).
* 20. Więzy integralności. Indeksy. Definiowanie kluczy obcych.
* 21. Zakładanie warunków integralnościowych.
* 22. Manipulowanie danymi DML-UPDATE, INSERT, DELETE.
* 23. DML. Kaskadowe usuwanie i aktualizowanie danych.

**Przypomnienie:**

* SQL DML (ang. Data Manipulation Language – „język manipulacji danymi”),
* SQL DDL (ang. Data Definition Language – „język definicji danych”),
* SQL DCL (ang. Data Control Language – „język kontroli nad danymi”).
* SQL DQL (ang. Data Query Language – „język definiowania zapytań”).
* SQL TCL (ang. Transactional Control Language) – „język kontroli transakcji”

**DML**

INSERT

UPDATE

DELETE

**DDL**

CREATE

ALTER

DROP

TRUNCATE

**DCL**

GRANT

REVOKE

DENY

**DQL**

SELECT

**TCL**

BEGIN

COMMIT

ROLLBACK

SAVEPOINT

**SQL - Structured Query Language**

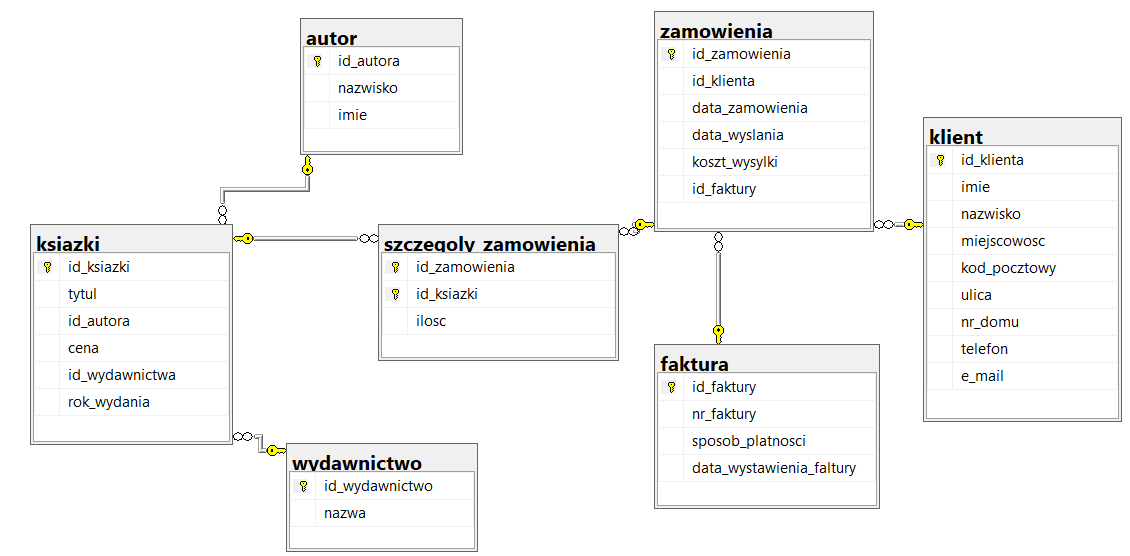
## 17. Język definiowania danych (DDL). Tworzenie i modyfikacja tabel.

## 18. Język definiowania danych (DDL).

Język DDL używany jest do tworzenia, modyfikowania i usuwania bazy danych oraz jej obiektów. Instrukcje wchodzące w skład tego języka to:

* CREATE ***nazwa\_obiektu*** — tworzy nowy obiekt,
* ALTER ***nazwa\_obiektu*** — zmienia strukturę istniejącego obiektu,
* DROP ***nazwa\_obiektu*** — usuwa istniejący obiekt.

**Przykład: bazy danych Ksiegarnia\_abc**



--Tworzenie bazy danych jest realizowane za pomocą:

CREATE DATABASE nazwa\_bazy

CREATE DATABASE Ksiegarnia\_abc;

--tworzenie widoku

CREATE VIEW widok AS

SELECT \* FROM klienci;

--Polecenie usuwające bazę danych:

DROP DATABASE nazwa\_bazy

DROP DATABASE Ksiegarnia\_abc;

--usuwanie widoku

DROP VIEW widok;

Usunąć bazę danych może tylko użytkownik, który ma odpowiednie uprawnienia, na przykład administrator serwera bazodanowego lub właściciel bazy danych. W trakcie usuwania bazy nikt nie może być z nią połączony.

## Tworzenie i usuwanie tabel

Do tworzenia tabel służy polecenie **CREATE TABLE** w postaci:

CREATE TABLE nazwa\_tabeli (

nazwa\_kolumny\_l typ\_kolumny\_l [atrybuty], ....,

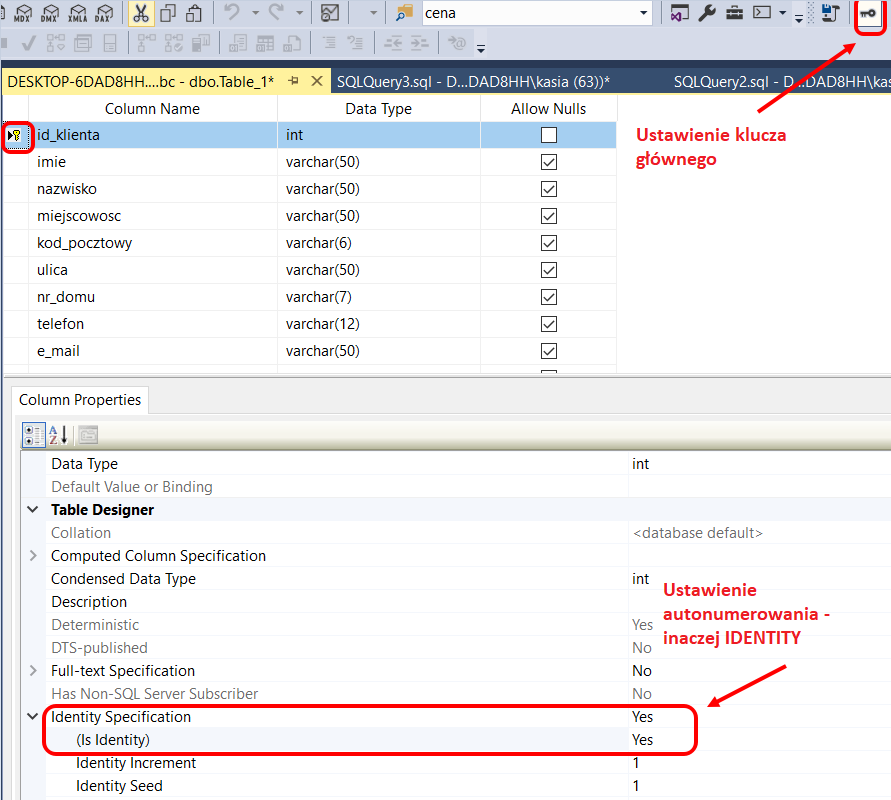
nazwa\_kolumny\_n typ\_kolumny\_n [atrybuty],

)

Do usuwania tabel służy instrukcja DROP TABLE w następującej postaci:

DROP TABLE nazwa\_tabeli;

Przykład: DROP TABLE klient;



Rysunek .Graficzne tworzenie tabeli.

CREATE TABLE klient(

id\_klienta int IDENTITY NOT NULL PRIMARY KEY,

imie varchar(50) ,

nazwisko varchar(50) ,

miejscowosc varchar(50) ,

kod\_pocztowy varchar(6),

ulica varchar(50),

nr\_domu varchar(7),

telefon varchar(12) ,

e\_mail varchar(50)

)

Podczas tworzenia tabel należy pamiętać, że:

* każda tabela musi mieć unikatową nazwę i unikatowego właściciela,
* każda kolumna musi mieć unikatową nazwę,
* nazwy tabel i kolumn muszą być zgodne z zasadami dotyczącymi standardów SQL,
* każda kolumna w tabeli musi mieć zdefiniowany typ,
* jeżeli kolumna jest typu znakowego, trzeba podać jej maksymalną długość,
* utworzone tabele są puste.

**Ćwiczenie**

Dla bazy danych Ksiegarnia\_abc tworzymy tabele: klient, zamowienia, ksiazki, autor, kategorie, szczegoly\_zmowien. Ustawiamy odpowiednie typy.

CREATE TABLE klient(

id\_klienta int IDENTITY(1,1) NOT NULL PRIMARY KEY,

imie varchar(50) ,

nazwisko varchar(50) ,

miejscowosc varchar(50) ,

kod\_pocztowy varchar(6),

ulica varchar(50),

nr\_domu varchar(7),

telefon varchar(12) ,

e\_mail varchar(50)

)

CREATE TABLE Ksiazki(

id\_ksiazki int IDENTITY(1,1) NOT NULL PRIMARY KEY,

tytul varchar (100),

autor varchar(50), --tu będzie zmiana przez alter

cena money,

wydawnictwo varchar (20),

rok\_wydania int)

Uwaga! Typ money nie występuje w MySQL , należy tam użyć - decimal jest stosowany do przechowywania kwot pieniędzy.

## Zmiana struktury tabeli -ALTER

Zmiana struktury tabeli może polegać na dodaniu kolumny, usunięciu kolumny, dodaniu atrybutu lub usunięciu atrybutu. Do modyfikowania struktury tabeli służy polecenie:

ALTER TABLE nazwa\_\_tabeli zmiana

**Przykłady:**

--Dodanie kolumny do tabeli ksiazki

ALTER TABLE ksiazki ADD liczba\_stron varchar(5);

--Dodaj do tabeli ksiazki kolumnę rok\_wydania2 typu int.

ALTER TABLE ksiazki ADD rok\_wydania2 int;

--Usunięcie kolumny z tabeli ksiazki rok\_wydania2

ALTER TABLE ksiazki DROP COLUMN rok\_wydania2;

--Modyfikowanie kolumny

--Zmiana definicji istniejącej kolumny tabeli ksiazki

ALTER TABLE ksiazki ALTER COLUMN rok\_wydania varchar (4);

--uwaga dotyczy MySQL!

--Zmiana definicji istniejącej kolumny tabeli ksiazki w MySQL:

ALTER TABLE ksiazki

MODIFY COLUMN rok\_wydania varchar(4);

--usunięcie z tabeli ksiazki kolumny autor i dodaj kolumnę id\_autora typu int

ALTER TABLE ksiazki DROP COLUMN autor;

ALTER TABLE książki ADD id\_autora int;

--Aby dodać klucz podstawowy w istniejącej tabeli, należy:

ALTER TABLE Klient

ADD PRIMARY KEY (id\_klienta);

--Jeżeli w SQL Server do definiowania klucza podstawowego użyte zostało polecenie ALTER TABLE, kolumna klucza podstawowego musi mieć ustawiony atrybut NOT NULL.

--Aby odebrać kolumnie atrybut PRIMARY KEY, należy w SQL Server użyć polecenia:

ALTER TABLE Klient

DROP CONSTRAINT nazwa\_klucza;

-- w MySQL użyć polecenia:

ALTER TABLE Klient DROP PRIMARY KEY;

Kontynuacja tworzenia tabel do bazy Ksiegarnia\_abc :

CREATE TABLE zamowienia (

id\_zamowienia int PRIMARY KEY IDENTITY(1,1),

id\_klienta int NOT NULL,

data\_zamowienia datetime,

data\_wyslania datetime,

koszt\_wysylki money,

id\_faktury int)

CREATE TABLE autor (

id\_autora INT IDENTITY (1, 1) NOT NULL PRIMARY KEY,

nazwisko varchar(50),

imie varchar(50)

)

# 20. Więzy integralności. Indeksy. Definiowanie kluczy obcych.

# 21. Zakładanie warunków integralnościowych.

## Integralność (spójność- zgodność z otoczeniem), rodzaje:

1. **Integralność semantyczna** - jeżeli wartości danych spełniają wcześniej zdefiniowane i nałożone ograniczenia (dopuszczalny typ, zakres długości, format). W ujęciu formalnym integralność semantyczna jest zachowana, jeżeli dane należą do zdefiniowanej wcześniej dziedziny.

2. **Integralność encji** jest zachowana, jeśli każda tabela posiada klucz główny, a jego wartości w ramach tabeli są unikatowe i różne od wartości NULL (to zapewni nam, że krotki w tabeli nie będą się powtarzać).

3. **Integralność referencyjna** jest zachowana, jeśli każda wartość klucza obcego jest równa jakiejś wartości klucza głównego w powiązanej tabeli nadrzędnej lub ewentualnie jest równa NULL.

4. **Węzły ogólne** - dodatkowe warunki poprawności danych określane przez użytkowników lub administratorów albo projektantów baz danych.

## Kaskadowe usuwanie i aktualizowanie danych

Integralność referencyjna relacyjnych baz danych pociąga za sobą konieczność określenia reguły postępowania w przypadku usuwania rekordu (wiersza/krotki) z tabeli powiązanej, co mogłoby unieważnić niektóre wartości kluczy obcych w tabelach z nią powiązanych.

* Aktualizowanie klucza podstawowego wymaga aktualizacji wartości w powiązanym z nim kluczem obcym. W relacyjnych bazach danych klucze podstawowe nie powinny być w ogóle modyfikowane, więc kaskadowe aktualizowanie definiujemy tylko w wyjątkowych przypadkach.
* Usunięcie wiersza w tabeli nadrzędnej lub wartości klucza podstawowego wymaga usunięcia lub zaktualizowania wartości w powiązanym z nim kluczu obcym. Kaskadowe usuwanie może doprowadzić do usunięcia wielu wierszy z różnych tabel, a w konsekwencji do usunięcia istotnych danych, zatem należy je definiować wyłącz­nie dla tabel pomocniczych opisujących związki „wiele do wielu”. Aktualizowanie klucza obcego jest bezpieczniejsze.

Kaskadowe usuwanie i aktualizowanie danych definiuje się w klauzulach ON UPDATE i ON DELETE z wartościami:

* CASCADE — modyfikacja ma zostać automatycznie powtórzona we wszystkich powiązanych tabelach..
* NO ACTION- dane w powiązanych tabelach nie będą automatycznie modyfikowane. Jest to domyślna wartość.
* SET NULL — zmodyfikowane wartości klucza podstawowego mają zostać zastąpione wartością NULL w powiązanych kolumnach klucza obcego.
* SET DEFAULT zmodyfikowane wartości klucza podstawowego mają zostać zastąpione w powiązanych kolumnach klucza obcego wartością domyślną.

## Atrybuty kolumn

Każda kolumna tabeli może mieć zdefiniowane za pomocą atrybutów ograniczenia, które określają, jakie dane mogą zostać w niej zapisane. Ograniczenia dotyczące kolumn mogą być definiowane w trakcie tworzenia tabeli lub w trakcie jej modyfikowania.

## PRIMARY KEY

Klucz podstawowy *(Primary Key*) to kolumna lub kombinacja kolumn, które w sposób jednoznaczny definiują wiersz w tabeli. Do określenia, która kolumna tabeli będzie kluczem podstawowym, stosuje się atrybut **PRIMARY KEY**. Kolumna z tym atrybutem jest unikatowa i automatycznie indeksowana : nie może mieć wartości NULL.

**NOT NULL**

Atrybut NOT NULL oznacza, że w kolumnie nie mogą wystąpić wartości puste. Tworzona kolumna domyślnie może zawierać wartość NULL.

--Np.

CREATE TABLE ksiazki (

id\_ksiazki int NOT NULL PRIMARY KEY, gatunek nvarchar (30) NOT NULL,) ;

--Aby ustawić w istniejącej kolumnie atrybut NOT NULL, należy wpisać:

ALTER TABLE ksiazki

ALTER COLUMN tytul varchar(50) NOT NULL;

--Do ustawienia atrybutu NOT NULL w MySQL można użyć polecenia:

ALTER TABLE Klient

MODIFY PESEL varchar(11) NOT NULL;

**IDENTITY –MS SQL /** **AUTO\_INCREMENT -MySQL**

Atrybut IDENTITY stosowany w SQL Server oznacza automatyczne ustawienie unika­towej wartości w kolumnie, powoduje automatyczny wzrost wartości dla kolejnego rekordu. Na przykład IDENTITY (1,1) oznacza zrost wartości kolumny o 1, począwszy od wartości 1. Jeśli chcemy ustawić numerowanie od innej wartości- różnej od 1, należy wpisać np. IDENTITY (50,2), gdzie wartość początkowa to 50, a krok wynosi 2.

Niemożliwe jest nadawanie atrybutu IDENTITY istniejącej kolumnie.

Podobne znaczenie w MySQL ma atrybut AUTO\_INCREMENT.

**DEFAULT**

DEFAULT- to domyślna wartość będzie wstawiana do kolumny we wszystkich nowych rekordach.

--Np. Definiowanie wartości domyślnej podczas tworzenia tabeli:

CREATE TABLE ksiazki

wydawnictwo varchar (50) DEFAULT 'Sowa' );

-- Aby dodać atrybut DEFAULT w istniejącej tabeli df\_miasto -dowolna nazwa CONSTRAINTU

ALTER TABLE Klient

ADD CONSTRAINT df\_miasto

DEFAULT 'Warszawa' FOR miejscowosc;

--Aby dodać atrybut DEFAULT w istniejącej tabeli- w MySQL:

ALTER TABLE klient

ALTER miejscowosc SET DEFAULT 'Warszawa';

**UNIQUE**

Atrybut **unique** jest stosowany, jeśli wartości w kolumnie nie mogą się powtarzać Ograniczenie powtarzalności w kolumnie nie blokuje możliwości wpisania do nie wartości NULL. Atrybut UNIQUE, podobnie jak atrybut PRIMARY KEY, nie pozwala na umieszczanie w kolumnie wartości powtarzających się. Jednak w tabeli może być wiele atrybutów UNIQUE, ale tylko jeden atrybut PRIMARY KEY.

--Np. Definiowanie wartości unikatowych podczas tworzenia tabeli:

CREATE TABLE kategorie (nazwa varchar (30) NOT NULL UNIQUE);

--Atrybut UNIQUE można ustawić w istniejącej tabeli, wpisując polecenie:

ALTER TABLE kategorie ADD UNIQUE (nazwa);

--Np. Definiowanie wartości unikatowych podczas tworzenia tabeli:w MySQL

CREATE TABLE kategorie (....

nazwa nvarchar (100) NOT NULL,

....

UNIQUE (nazwa)) ;

**Warunek logiczny CHECK**

Atrybut CHECK pozwala na zdefiniowanie warunków ograniczających zakres danych wprowadzanych do kolumny. Dla każdej kolumny można definiować wiele warun­ków. Można również tworzyć za pomocą operatorów NOT, AND i OR złożone warunki ograniczające.

--Np. Definiowanie ograniczenia CHECK podczas tworzenia tabeli:

CREATE TABLE ksiazki (rok\_wydania int CHECK (...,rok\_wydania BETWEEN 2000 AND 2021),....) ;

--na istniejącej tabeli ustawienie CHECK

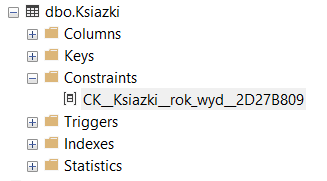
ALTER TABLE ksiazki

ADD CHECK (rok\_wydania BETWEEN 2000 AND 2021);

--usunięcie atrybutu CHECK

ALTER TABLE ksiazki DROP CONSTRAINT CK\_\_Ksiazki\_\_rok\_wyd\_\_2D27B809;

CK\_\_Ksiazki\_\_rok\_wyd\_\_2D27B809 –to jest nazwa constraint-u, można ją pobrać rozwijając strukturę tabeli i CONSTRAINTS.



--Definiowanie ograniczenia CHECK w MySQL jest możliwe od wersji 8.0.16.

--Np. W MySQL ograniczenie CHECK definiowane podczas tworzenia tabeli:

CREATE TABLE ksiazki (....

cena decimal,

CHECK (cena >20),....) ;

## Definiowanie klucza obcego

Sprawdzenie spójności w bazie danych odbywa się po jawnym zdefiniowaniu klucza obcego.

Na naszym przykładzie książki id\_autora jest kluczem obcym i łączy się z tabelą autorzy po id\_autora, będącego tam kluczem głównym.

Dodawanie klucza obcego można wykonać w poleceniu: CREATE TABLE lub ALTER TABLE.

Ogólna postać polecenia wygląda następująco:

[CONSTRAINT nazwa] FOREIGN KEY (kolumnal, kolumna 2, ...)

REFERENCE nazwa\_tabeli (kolumnal, kolumna 2, ...)

gdzie:

* CONSTRAINT nazwa jest nazwą ograniczenia, może zostać pominięta – wtedv ograniczeniu zostanie nadana nazwa systemowa;
* FOREIGN KEY (kolumnal, kolumna 2, ...) określa kolumny zawierające klucz obcy
* REFERENCE nazwa\_tabeli (kolumnal, kolumna 2, ...) określa z której tabel pochodzi klucz obcy i które kolumny są w niej kluczem podstawowym.

**Przykład ustalania klucza obcego przy tworzeniu tabeli ksiazki:**

CREATE TABLE Ksiazki(

id\_ksiazki int IDENTITY(1,1) NOT NULL PRIMARY KEY,

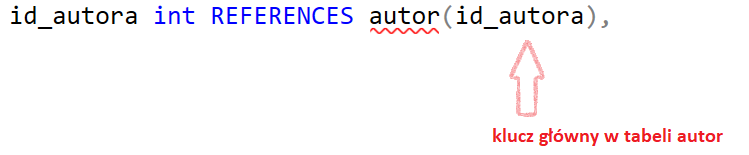
tytul varchar (100),

id\_autora int REFERENCES autor(id\_autora),

cena money,

wydawnictwo varchar (20),

rok\_wydania int);



**Przykład**

W istniejącej tabeli można definiować klauzulę ograniczeń klucza obcego dla nowej kolumny id\_autora podczas dodawania tej kolumny do tabeli.

ALTER TABLE ksiazki

ADD id\_autora int REFERENCES autor(id\_autora) ;

**Przykład**

W istniejącej tabeli książki dla istniejącego pola klucza obcego id\_autora należy zdefiniować klauzulę ograniczeń klucza obcego przez zmodyfikowanie kolumny:

ALTER TABLE ksiazki

ADD CONSTRAINT ksiazki\_FK FOREIGN KEY (id\_autora)

REFERENCES autor(id\_autora);

Przy definiowaniu ograniczenia nałożonego na klucz obcy w istniejącej tabeli powoduje ono skutki wynikające z zasad kaskadowego usuwania i aktualizowania danych. I tak po zdefiniowaniu ograniczenia nałożonego na klucz obcy w tabeli *ksiazki* zmiana lub usunięcie klucza podstawowego z tabeli *autor* powoduje automatyczne zmiany w tabeli książki.

Jeżeli w połączonych tabelach występuje konflikt danych (np. w tabeli *książki dodaję id\_autora*, którego nie ma w tabeli *Autor),* to w wyniku wykonania wstawiania rekordu otrzymamy komunikat:



**Przykład** - **WITH NOCHECK**

Tworzymy klucz obcy, który będzie działał tylko dla nowo wprowadzanych danych:

ALTER TABLE Ksiazki WITH NOCHECK

ADD CONSTRAINT Ksiazki\_FR FOREIGN KEY (id\_autora)

REFERENCES autor(id\_autora);

W tej wersji zapytania została dodana klauzula WITH NOCHECK, która informuje bazę danych, że nie należy weryfikować wcześniej wprowadzonych danych.

**Przykład** – **na kaskadowe usuwanie i aktualizowanie danych:**

ALTER TABLE ksiazki DROP CONSTRAINT ksiazki\_FK;

ALTER TABLE ksiazki

ADD CONSTRAINT ksiazki\_FK FOREIGN KEY (id\_autora)

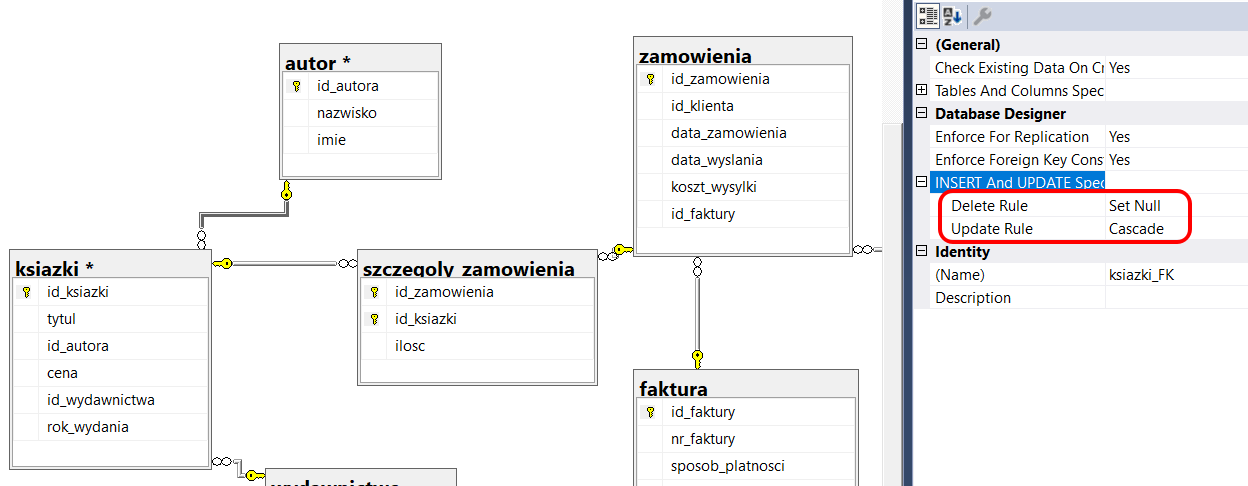
REFERENCES autor (id\_autora)

ON UPDATE CASCADE

ON DELETE SET NULL;

Polecenie DROP CONSTRAINT ksiazki\_FK usuwa wcześniejsze ograniczenie, następnie jest dodane nowe.

Podczas tworzenia diagramu również można ustawić klucze obce i właściwości dla UPDATE i DELETE:



# Indeksy

Indeksy są podstawowym mechanizmem, który służy do poprawy wydajności wykonywania zapytań w bazie danych. Jest to dodatkowa struktura danych, przechowywana obok tabeli, zoptymalizowana pod kątem wyszukiwania.

Indeks w bazie danych możemy porównać do indeksu czy skorowidzu w książce, czyli listy haseł wraz z odnośnikiem wskazującym, na której stronie znajduje się informacja na temat tego hasła. Indeksy zakładamy na wybranych kolumnach tabel. Indeksy są fizycznie przechowywane w normalnych plikach z danymi (\*.mdf lub \*.ndf). Indeksy są sposobem na przyspieszenie zapytań odczytu poprzez sortowanie wierszy tabeli według kolumny. Wpływ indeksu nie jest zauważalny w przypadku małych baz danych, takich jak przykład, ale jeśli istnieje duża liczba wierszy, może to znacznie poprawić wydajność.

Kompromisem przy tworzeniu indeksu jest szybkość zapisu i rozmiar bazy danych. Przechowywanie indeksu zajmuje miejsce. Ponadto za każdym razem, gdy wykonywana jest funkcja INSERT lub kolumna jest aktualizowana, indeks musi być aktualizowany.

Wyróżniamy 2 rodzaje indeksów:

### Indeksy klastrowane- Podczas korzystania z indeksu klastrowego wiersze tabeli są sortowane według kolumny, do której stosuje się indeks klastrowany. Dlatego w tabeli może znajdować się tylko jeden indeks klastrowany, ponieważ nie można uporządkować tabeli według dwóch różnych kolumn.

Przykład tworzenia indeksu klastrowego w tabeli Pracownicy w kolumnie Nazwisko pracownika:

--dodanie indeksu

CREATE CLUSTERED INDEX ix\_employees\_name ON Employees(Surname);

--usuniecie indeksu

DROP INDEX ix\_employees\_name ON Employees;

* **Indeksy nieklastrowane**- są przechowywane oddzielnie od tabeli. Każdy indeks w tej strukturze zawiera wskaźnik do wiersza w tabeli, którą reprezentuje. W tabeli może znajdować się wiele indeksów nieklastrowanych. Operacje odczytu są zwykle wolniejsze w przypadku indeksów nieklastrowanych niż w przypadku indeksów klastrowych, ponieważ najpierw trzeba przejść do indeksowania, a następnie do tabeli. Jednak nie ma żadnych ograniczeń w operacjach zapisu.

Przykład tworzenia indeksu nieklastrowego w tabeli Pracownicy i kolumnie Nazwisko pracownika:

CREATE NONCLUSTERED INDEX ix\_employees\_name ON Employees(Surname);

Różnica w kodzie między klastrowanym a nieklastrowanym:

Różnica to umieszczenie przed słowem kluczowym INDEX słowa kluczowego CLUSTERED dla indeksu klastrowanego, lub słowa kluczowego NONCLUSTERED dla indeksu nieklastrowanego.

Przy tworzeniu indeksów warto wiedzieć, że domyślnie tworzone są indeksy nieklastrowane. Słowo kluczowe NONCLUSTERED można zatem pominąć.

------------------------------

--

-- INDEKSY -PRZYKŁADY

--

------------------------------

--Tworzenie indeksu, gdzie duplikaty są dozwolone

CREATE INDEX idx\_nazwisko ON Klienci(nazwisko);

--Przykład tworzenia indeksu w istniejącej tabeli:

CREATE INDEX Indeks\_tytul ON ksiazki (tytul);

--Tworzenie indeksu, gdzie duplikaty są niedozwolone

CREATE UNIQUE INDEX idx\_nazwa ON Produkty(nazwa);

--Usuwanie indeksu

DROP INDEX idx\_nazwa ON Produkty;

DROP INDEX Produkty.idx\_nazwa;

# 22. Manipulowanie danymi DML-UPDATE, INSERT, DELETE.

# 23. DML. Kaskadowe usuwanie i aktualizowanie danych.

# Polecenia (DML)

(SELECT — wybiera dane z bazy danych)- często zaliczana osobno do DQL

* **INSERT** umieszcza nowe wiersze w tabeli,
* **UPDATE** zmienia zawartość istniejącego wiersza,
* **DELETE** — usuwa wiersze z tabeli.

## Instrukcja INSERT

Instrukcja INSERT służy do wstawiania nowych wierszy do tabeli, składnia:

INSERT INTO nazwa\_tabeli (kolumnal, kolumna2, ... )

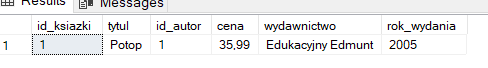
VALUES (wartości, wartość2, . . .)

Jeżeli jawnie nie podamy, do jakich kolumn powinny zostać wstawione wartości, to dane podane w klauzuli VALUES zostaną wstawione do kolejnych kolumn tabeli.

**Przykład INSERT INTO- wstawienie pojedynczego rekordu:**

INSERT INTO ksiazki(tytul, id\_autor,cena,wydawnictwo,rok\_wydania)

VALUES ('Potop',1,35.99,'Edukacyjny Edmunt',2005);



Została wstawiona również wartość dla pola id\_ksiazki 1, ponieważ podczas definiowania tabeli pole to zostało wybrane jako pole klucza podstawowego z automatycznym wstawianiem kolejnych wartości.

Jeżeli dla jakiejś kolumny została zdefiniowana wartość domyślna (klauzula DEFAULT), to w podobny sposób zostanie ona automatycznie stawiona do tabeli, chyba że podczas wstawiania wiersza podamy jej wartość.

**Przykład INSERT INTO- wstawienie kilku rekordów jednym poleceniem:**

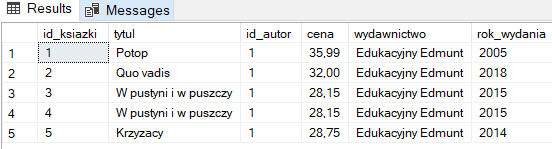
INSERT INTO ksiazki(tytul, id\_autor,cena,wydawnictwo,rok\_wydania)

VALUES ('Quo vadis',1,32,'Edukacyjny Edmunt',2018),

('W pustyni i w puszczy',1,28.15,'Edukacyjny Edmunt',2015),

('W pustyni i w puszczy',1,28.15,'Edukacyjny Edmunt',2015),

('Krzyżacy',1,28.75,'Edukacyjny Edmunt',2014);



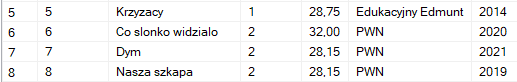
Można również krótszym sposobem, bez podania w nawiasie nazwy kolumn, choć osobiście nie polecam:

INSERT INTO ksiazki

VALUES ('Co słonko widziało',2,32,'PWN',2020),

('Dym',2,28.15,'PWN',2021),

('Nasza szkapa',2,28.15,'PWN',2019);



## Instrukcja UPDATE

Do aktualizowania danych służy instrukcja UPDATE w postaci:

UPDATE nazwa\_tabeli

SET kolumnal=wartość, kolumna2=wartość, . . .)

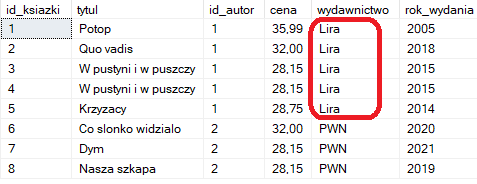
W klauzuli UPDATE podajemy nazwę tabeli, a w klauzuli SET nazwę modyfikowani kolumny oraz przypisaną jej nową wartość Warto pamiętać, że wykonanie takiej instrukcji spowoduje zmianę we wszystkich wierszach podanej kolumny. Dlatego jeżeli modyfikacja będzie dotyczyła tylko wybranych wierszy, do instrukcji należy dołączać klauzulę WHERE.

**Przykład UPDATE**

UPDATE ksiazki SET wydawnictwo ='Lira'

WHERE wydawnictwo='Edukacyjny Edmunt';

W podanym przykładzie w tabeli *książki w* kolumnie *wydawnictwo* zostanie zmieniona wartość *z Edukacyjny Edmunt na Lira.*



Przy użyciu instrukcji UPDATE możliwe jest modyfikowanie wielu kolumn jednocześnie.

**Przykład UPDATE –kilka wartości jednocześnie**

UPDATE ksiazki SET tytul ='Nie mów nikomu', id\_autor ='3', cena=42.50,

wydawnictwo='Albatros', rok\_wydania= 2018

WHERE id\_ksiazki=1 ;



## Instrukcja DELETE

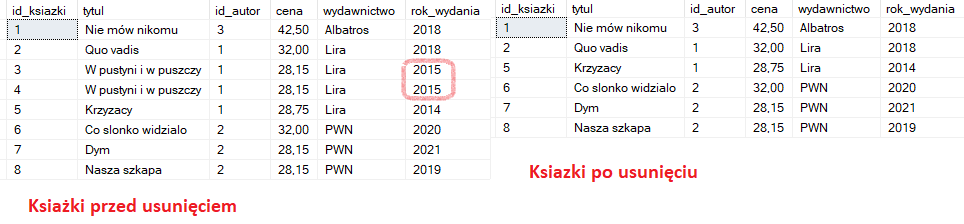
instrukcja DELETE usuwa wiersze z wybranej tabeli.

Aby usunąć wszystkie wiersze z tabeli, wystarczy w klauzuli FROM podać nazwę tabeli.

**Przykład na Delete –usunięcie z tabeli książki z roku 2015**

DELETE FROM ksiazki

WHERE rok\_wydania= 2015;



**Przykład na Delete –usunięcie wszystkich rekordów z tabeli ksiazki:**

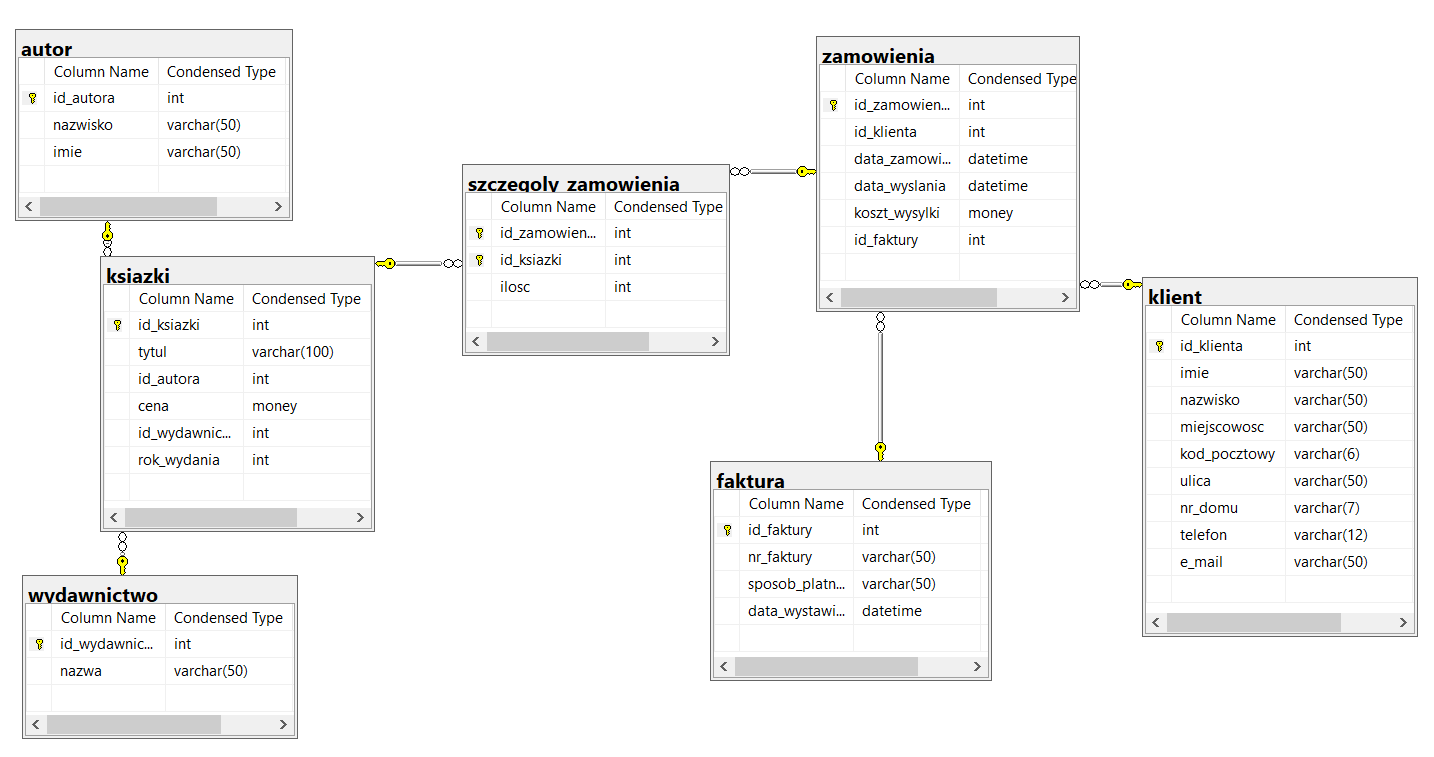
DELETE FROM ksiazki;

**Usunięcie całej zawartości tabeli poleceniem TRUNCATE TABLE nazwa\_tabeli;**

TRUNCATE TABLE ksiazki;

Uwaga! Po zastosowaniu TRUNCATE ponowne wstawianie rekordów zaczyna się od numerowania od 1, przy poleceniu np. DELETE FROM książki- kolejne wstawianie rekordów kontynuuje numerowanie auto numerowania;

**Zadanie do przetestowania na kaskadowe usuwanie i aktualizowanie danych.**



1. **Utwórz bazę danych zgodnie ze schematem i opisem z przykład.**
2. **Ustal klucze relacje/związki pomiędzy tabelami, czyli klucze obce.**
3. **Ustal dla kliku relacji kaskadowe usuwanie i aktualizowanie i wartości NULL –np. dla usuwania.**
4. **Wprowadź do każdej z tabel po ok. 10 rekordów.**
5. **Zmodyfikuj kilka dowolnych rekordów.**
6. **Usuń dowolne rekordy, sprawdź co się dzieje z rekordami, gdzie było ustawione kaskadowe usuwanie lub aktualizacja.**
7. **Przetestuj polecenia pokazane w przykładach.**

Różne przykłady na integralność danych przy zastosowaniu przykładu: tabela produkty i klienci:

------------------------------

--

-- CHECK

--

----------------------------

--Tworzenie tabeli z ograniczeniami pól - CHECK(warunek) musi być spełniony, aby dodać nowy rekord

CREATE TABLE Produkty (

Id\_produktu int NOT NULL IDENTITY(1,1) PRIMARY KEY,

nazwa VARCHAR(50) NOT NULL,

cena FLOAT(10) NOT NULL,

CHECK (cena>1.0));

--Tworzenie tabeli z ograniczeniami wraz z nazewnictwem ograniczenia przez CONSTRAINT nazwa, następnie 2 warunki w CHECK połączone AND

CREATE TABLE Klienci (

Id\_klient int NOT NULL IDENTITY(1,1) PRIMARY KEY,

Nazwisko varchar(50) NOT NULL,

Imie varchar(50),

Wiek int,

Miasto varchar(50),

CONSTRAINT CK\_KlienciWiek CHECK (Wiek>=18 AND Miasto='Rzeszów'));

--Dodanie ograniczenia do istniejącej już tabeli

ALTER TABLE Produkty ADD CHECK (cena>1.0);

--Dodanie ograniczenia do istniejącej tabeli z nazwą ograniczenia

ALTER TABLE Klienci ADD CONSTRAINT CK\_KlienciWiek CHECK (Wiek>=18 AND Miasto='Rzeszów');

--Usuwanie istniejącego ograniczenia

ALTER TABLE Klienci DROP CONSTRAINT CK\_KlienciWiek;

------------------------------

--

-- DEFAULT

--

------------------------------

--Tworzenie tabeli z wartością domyślną pola

CREATE TABLE Produkty (

Id\_produktu int NOT NULL IDENTITY(1,1) PRIMARY KEY,

nazwa VARCHAR(50) NOT NULL,

opis VARCHAR(255) DEFAULT 'Brak opisu',

cena FLOAT(10) NOT NULL);

--Wartością domyślną może być też funkcja matematyczna np. getdate()

CREATE TABLE Zamowienia (

ID\_zamowienia int NOT NULL IDENTITY(1,1) PRIMARY KEY,

idProduktu INT,

data\_zamowienia DATETIME DEFAULT getdate() );

--Dodanie wartości domyślnej do istniejącej już tabeli (w niektórych systemach może nie działać ze względu na konwersje danych)

ALTER TABLE Zamowienia ADD CONSTRAINT CK\_ZamowieniaDataZamowienia DEFAULT getdate() FOR data\_zamowienia;

--Usuwanie wartości domyślnej z istniejącej już tabeli

ALTER TABLE Zamowienia DROP CONSTRAINT CK\_ZamowieniaDataZamowienia;

--w moim wypadku nazwa ograniczenia to, pokazane wyżej w dokumencie jak odszukać DF\_\_Zamowieni\_\_data\_\_\_1920BF5C więc usuwamy go

ALTER TABLE Zamowienia DROP CONSTRAINT DF\_\_Zamowieni\_\_data\_\_\_1920BF5C;

------------------------------

--

-- UNIQUE

--

------------------------------

--Ograniczenie kolumny do posiadania tylko wartości unikalnych

CREATE TABLE Produkty (

ID\_produktu int NOT NULL IDENTITY(1,1) PRIMARY KEY,

nazwa VARCHAR(50) NOT NULL UNIQUE, --Tutaj dodajemy UNIQUE

opis VARCHAR(255) DEFAULT 'Brak opisu',

cena FLOAT(10) NOT NULL,

);

CREATE TABLE Produkty (

ID\_produktu int NOT NULL IDENTITY(1,1) PRIMARY KEY,

nazwa VARCHAR(50) NOT NULL,

opis VARCHAR(255) DEFAULT 'Brak opisu',

cena FLOAT(10) NOT NULL,

UNIQUE(nazwa) --Tutaj dodajemy UNIQUE

);

--Można go też nazwać

CREATE TABLE Produkty (

ID\_produktu int NOT NULL IDENTITY(1,1) PRIMARY KEY,

nazwa VARCHAR(50) NOT NULL,

opis VARCHAR(255) DEFAULT 'Brak opisu',

cena FLOAT(10) NOT NULL,

CONSTRAINT ck\_nazwa UNIQUE(nazwa)

);

--Usuwanie ograniczenia

ALTER TABLE Produkty DROP CONSTRAINT ck\_nazwa;