

Każda **relacyjna baza danych** składa się z jednej lub wielu **tabel**. **Tabele** są **tworzone przez programistę** na podstawie **projektu bazy danych**.

Projekt bazy danych opracowujemy na podstawie **modelu wymagań informacyjnych użytkownika**, tj. jego zapotrzebowania na przechowywanie danych o **określonych przez użytkownika strukturach**.

Invoice

Product

Product_ID
Material_ID
Type
Availability
Stock
Subcontractor_ID

Material

Material_ID
Material_Type
Availability
Stock
Subcontractor_ID

Subcontractor

Subcontractor_ID
Name
Address
Postal Code
Email

Event

Event_ID
Location
Date
Address_ID

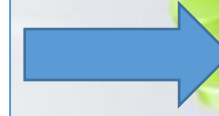
Etapy procesu konstrukcji bazy danych

Proces konstrukcji bazy danych zwykle dzielimy na następujące etapy:

pozyskanie wymagań informacyjnych użytkownika końcowego – najczęściej odbywa się w formie wywiadu prowadzonego przez analityka systemowego

1

1. Tytuł
2.
3.
4.



wyrażenie wymagań informacyjnych użytkownika końcowego w postaci graficznego diagramu ER (*Entity Relationships Diagram, diagram związków encji*)

2

przekształcenie diagramu ER do projektu tabel

3

To już robiliśmy

optymalizacja projektu tabel (normalizacja, denormalizacja)

utworzenie w bazie danych tabel według projektu – za pomocą poleceń języka SQL

Sami

1. Czego ma dotyczyć nasza baza danych – **temat**?
Np. **Salon samochodowy**
2. Wypisujemy wszystkie słowa jakie nam przychodzą na myśl
3. Grupujemy : **Klient**, **Samochody**.... Co się łączy z samochodami, itd..





Definicja – **diagram ER**

Zadaniem osoby pozyskującej **wymagania informacyjne** jest ich **wyrażenie w postaci graficznej w formie diagramu ER**.

Diagram ER składa się z trzech głównych typów elementów:

- ✓ **encji** – reprezentujących „typy rzeczy”, które będą opisywane w bazie danych,
- ✓ **atrybutów** – reprezentujących cechy opisowe encji,
- ✓ **związków** – reprezentujących informacje o powiązaniach pomiędzy obiektami encji.

Istnieje wiele **notacji graficznych** stosowanych **do zapisu diagramów ER**:

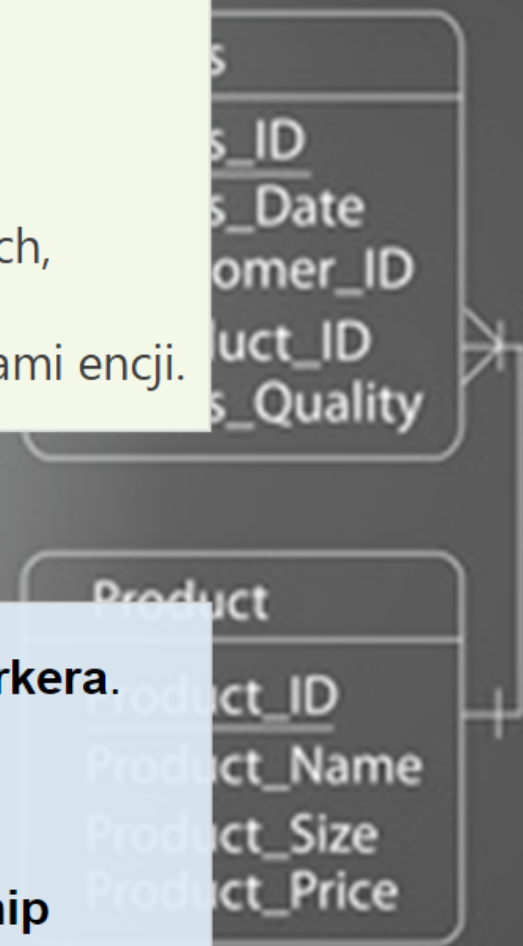
notacja Barkera

notacja Chena

notacja UML

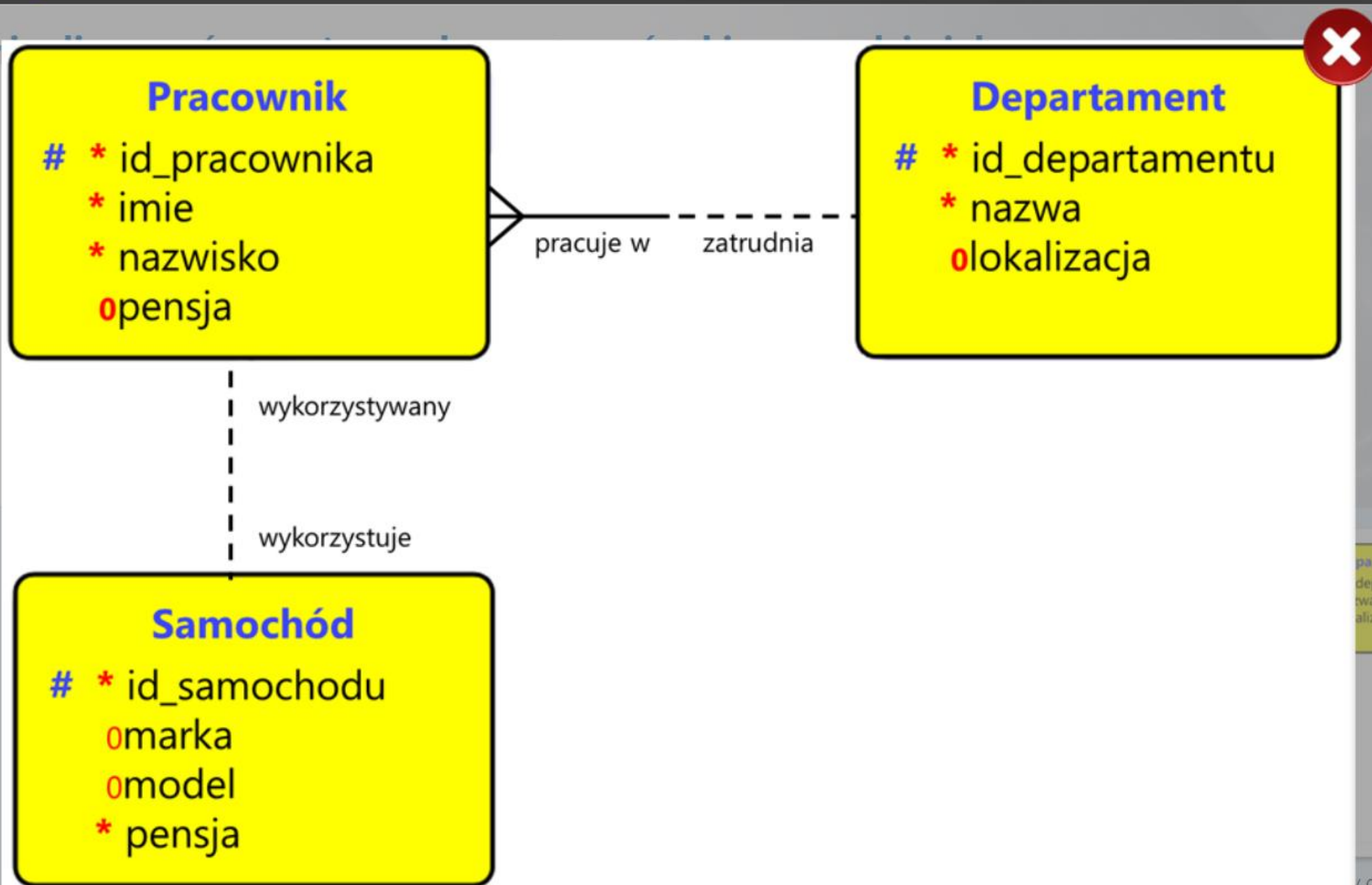
W dalszych przykładach posłużymy się **notacją Barkera**. Aby dowiedzieć się więcej na temat notacji Barkera, zapoznaj się z publikacją:

Richard Barker, CASE Method: Entity Relationship Modelling,
Addison-Wesley 1990, ISBN 0201416964.



Przykład - diagram ER

Do tworzenia...



departament
departamentu
nazwa
lokalizacja

/ go powiększyć.

Zasady konstrukcji diagramów ER

Oto kilka ogólnych **zasad konstrukcji diagramów ER w notacji Barkera**:

encje

atrybuty

związek

Zasady dotyczące encji:

- ✓ **encje** są nazywane **w liczbie pojedynczej**
- ✓ **każda encja powinna posiadać atrybut** (atrybuty), który pełni **rolę identyfikatora** (klucza), **oznaczony znakiem „#”**



Przykład

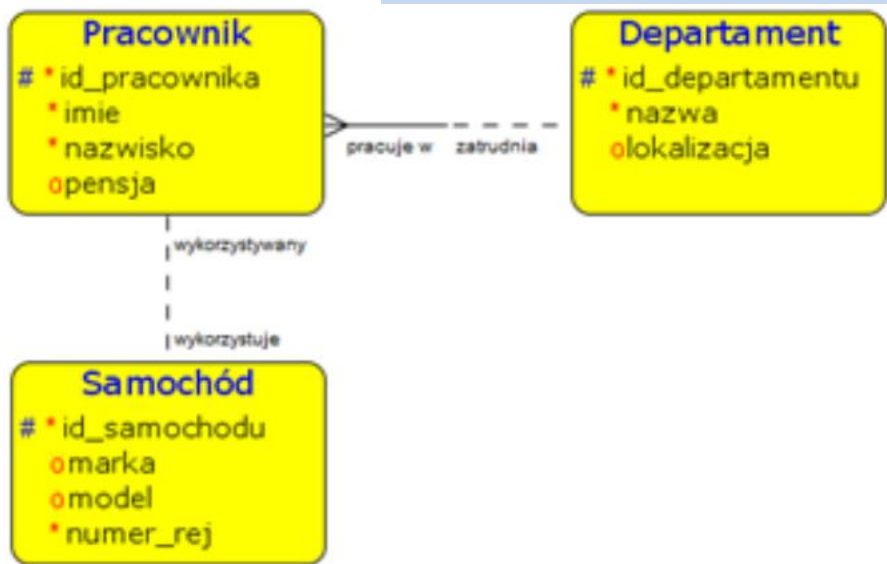
1 Co ma się znajdować w encjach

Interpretacja przykładowego diagramu ER byłaby zatem następująca:

- ✓ **istnieje potrzeba przechowywania danych o pracownikach:** imienia, nazwiska, pensji (wartość pensji jest nieobowiązkowa),
- ✓ **istnieje potrzeba przechowywania danych o departamentach:** nazwy, lokalizacji (wartość lokalizacji jest nieobowiązkowa),
- ✓ **istnieje potrzeba przechowywania danych o samochodach:** marki, modelu, numeru rejestracyjnego (wartości marki i modelu są nieobowiązkowe),


2 Relacje

- ✓ **istnieje potrzeba przechowywania informacji o powiązaniach pracowników z wykorzystywanymi samochodami** (z każdym pracownikiem może być związany tylko jeden samochód, z każdym samochodem może być związany tylko jeden pracownik),
- istnieje potrzeba przechowywania informacji o powiązaniach pracowników z zatrudniającymi ich departamentami** (z każdym pracownikiem musi być związany dokładnie jeden departament, z każdym departamentem może być związanych wielu pracowników).



Zasady konstrukcji diagramów ER

Oto kilka ogólnych **zasad konstrukcji diagramów ER w notacji Barkera**:

 Kliknij na etykiety, aby zobaczyć szczegółowy opis.

encje

atrybuty


związek

Zasady dotyczące atrybutów:

- ✓ **atrybuty** są nazywane w liczbie pojedynczej
- ✓ w nazwie atrybutu nie zagnieżdżamy nazwy encji (np. pracownik_imie)
- ✓ **atrybuty**, których **wartości muszą być zawsze wprowadzane** (obowiązkowe), są oznaczone **znakiem „*”**, a atrybuty, których **wartości mogą być niewprowadzone** (opcjonalne) – **znakiem „o”**

Zasady konstrukcji diagramów ER

Oto kilka ogólnych **zasad konstrukcji diagramów ER w notacji Barkera**:

 Kliknij na etykiety, aby zobaczyć szczegółowy opis.

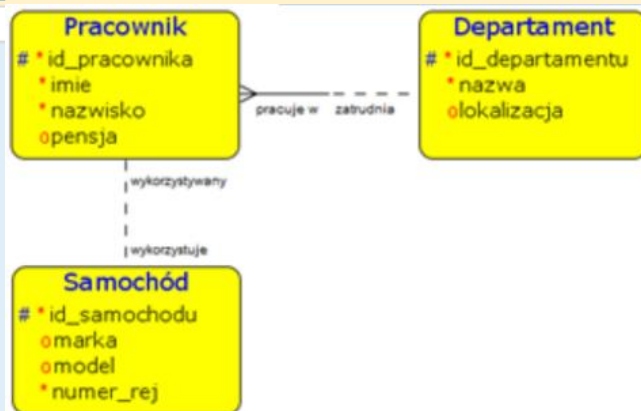
encje

atrybuty

związek

Zasady dotyczące związków:

- ✓ **każdy związek powinien posiadać dwie etykiety objaśniające jego znaczenie** – jedną po stronie każdej z wiązanych encji
- ✓ **związek, w którym musi uczestniczyć każdy obiekt encji, musi być oznaczony linią ciągłą po stronie tej encji**
- ✓ **związek, w którym niektóre obiekty encji mogą nie uczestniczyć, musi być oznaczony linią przerywaną po stronie tej encji**
- ✓ **związek, który pozwała wielu obiektom jednej encji być powiązanymi z tym samym obiektem drugiej encji, musi być oznaczony widelkami (tzw. kurza stopka) po stronie pierwszej encji**
- ✓ **związek, który wymaga, aby każdy obiekt jednej encji był powiązany z zupełnie innym obiektem drugiej encji, musi być oznaczony linią pojedynczą po stronie pierwszej encji**
- ✓ **związki mogą być rekurencyjne**, tj. wiązać obiekty encji z innymi obiektami tej samej encji





Podział związków ze względu na krotność wiązanych obiektów encji

Podział związków ze względu na obowiązkowość

Podział związków ze względu na krotność wiązanych obiektów encji:

- ✓ **związki jeden do jednego** – każdy obiekt pierwszej encji może być powiązany z dokładnie jednym obiektem drugiej encji, a każdy obiekt drugiej encji może być powiązany z dokładnie jednym obiektem pierwszej encji, **(1:1)**
- ✓ **związki jeden do wielu** – każdy obiekt pierwszej encji może być powiązany z wieloma obiektami drugiej encji, a każdy obiekt drugiej encji może być powiązany z dokładnie jednym obiektem pierwszej encji, **(1:M)**
- ✓ **związki wiele do wielu** – każdy obiekt pierwszej encji może być powiązany z wieloma obiektami drugiej encji, a każdy obiekt drugiej encji może być powiązany z wieloma obiektami pierwszej encji. **(M:N)**

I.	(1:1) jeden do jednego	(1:M) jeden do wielu	wiele do wielu (M:N)
<p>Zapoznaj się z podziałem związków ze względu na krotność wiązanych obiektów encji (na podstawie wcześniejszego przykładu).</p>	<div data-bbox="963 104 1414 392"> Pracownik # * id_pracownika * imie * nazwisko o pensja </div> <div data-bbox="963 539 1414 828"> Samochód # * id_samochodu o marka o model * numer_rej </div> <div data-bbox="1184 406 1388 542"> wykorzystuje wykorzystywany </div>	<div data-bbox="1498 104 1949 392"> Pracownik # * id_pracownika * imie * nazwisko o pensja </div> <div data-bbox="1498 539 1949 828"> Samochód # * id_samochodu o marka o model * numer_rej </div> <div data-bbox="1694 406 1898 542"> wykorzystuje wykorzystywany </div>	<div data-bbox="2033 104 2484 392"> Pracownik # * id_pracownika * imie * nazwisko o pensja </div> <div data-bbox="2033 539 2484 828"> Samochód # * id_samochodu o marka o model * numer_rej </div> <div data-bbox="2229 406 2433 542"> wykorzystuje wykorzystywany </div>
<div data-bbox="0 849 891 1420"> <div data-bbox="0 856 318 1063"> Pracownik # * id_pracownika * imie * nazwisko o pensja </div> <div data-bbox="573 856 891 1063"> Departament # * id_departamentu * nazwa o lokalizacja </div> <div data-bbox="0 1206 318 1413"> Samochód # * id_samochodu o marka o model * numer_rej </div> <div data-bbox="343 963 547 1013"> pracuje w zatrudnia </div> <div data-bbox="165 1078 305 1206"> wykorzystywany wykorzystuje </div> </div>	<p>Każdy pracownik może wykorzystywać dokładnie jeden samochód, każdy samochód może być wykorzystywany przez dokładnie jednego pracownika.</p>	<p>Każdy pracownik może wykorzystywać dokładnie jeden samochód, każdy samochód może być wykorzystywany przez wielu pracowników.</p>	<p>Każdy pracownik może wykorzystywać wiele samochodów, każdy samochód może być wykorzystywany przez wielu pracowników.</p>

I.

Podział związków ze względu na krotność wiązanych obiektów encji

Podział związków ze względu na obowiązkowość:

- ✓ **związki obowiązkowe** – każdy obiekt encji musi być powiązany,
- ✓ **związki opcjonalne** – obiekty encji mogą, lecz nie muszą uczestniczyć w powiązaniu.

Podział związków ze względu na obowiązkowość

II.

Następny slajd



Kliknij na etykiety, aby zobaczyć szczegółowy opis.

Zapoznaj się z podziałem związków ze względu na obowiązkowość (na podstawie wcześniejszego przykładu).

1	związek obowiązkowy	2	związek opcjonalny	3	związek jednostronnie obowiązkowy
	<div>Pracownik</div> <div># * id_pracownika * imie * nazwisko o pensja</div>	<div>Pracownik</div> <div># * id_pracownika * imie * nazwisko o pensja</div>	<div>Pracownik</div> <div># * id_pracownika * imie * nazwisko o pensja</div>		
	wykorzystuje	wykorzystuje	wykorzystuje		
	wykorzystywany	wykorzystywany	wykorzystywany		
	<div>Samochód</div> <div># * id_samochodu o marka o model * numer_rej</div>	<div>Samochód</div> <div># * id_samochodu o marka o model * numer_rej</div>	<div>Samochód</div> <div># * id_samochodu o marka o model * numer_rej</div>		
	Każdy pracownik musi być powiązany z jakimś samochodem, każdy samochód musi być powiązany z jakimś pracownikiem.	Pracownik może być powiązany z samochodem, samochód może być powiązany z pracownikiem.	Pracownik może być powiązany z samochodem, samochód musi być powiązany z jakimś pracownikiem.		



Definicja – ograniczenia integralnościowe

Jedną z bardziej interesujących **funkcji** każdego serwera bazy danych jest **kontrola poprawności wprowadzanych danych**. Polega ona na **automatycznej weryfikacji reguł poprawności danych**, nazywanych **ograniczeniami integralnościowymi** (ang. *integrity constraints*), podczas wykonywania **operacji INSERT, UPDATE i DELETE**. Jeżeli polecenia takie **naruszają co najmniej jedno ograniczenie integralnościowe**, **serwer bazy danych odmawia ich wykonania**. Dzięki temu możemy zapewnić wysoką jakość gromadzonych danych.

Wyróżniamy następujące **typy ograniczeń integralnościowych**:



Kliknij na etykiety, aby zobaczyć szczegółowy opis.

NOT NULL

CHECK

UNIQUE

PRIMARY KEY

REFERENCES

Uniemożliwia wprowadzenie rekordu z wartościami pustymi we wskazanych kolumnach.

Ograniczenia integralnościowe



Definicja – ograniczenia integralnościowe

Jedną z bardziej interesujących **funkcji** każdego serwera bazy danych jest **kontrola poprawności wprowadzanych danych**. Polega ona na **automatycznej weryfikacji reguł poprawności danych**, nazywanych **ograniczeniami integralnościowymi** (ang. *integrity constraints*), podczas wykonywania **operacji INSERT, UPDATE i DELETE**. Jeżeli polecenia takie **naruszają co najmniej jedno ograniczenie integralnościowe, serwer bazy danych odmawia ich wykonania**. Dzięki temu możemy zapewnić wysoką jakość gromadzonych danych.

Wyróżniamy następujące **typy ograniczeń integralnościowych**:



Kliknij na etykiety, aby zobaczyć szczegółowy opis.

NOT NULL

CHECK

UNIQUE

PRIMARY KEY

REFERENCES

Uniemożliwia wprowadzenie rekordu, którego wartości kolumn nie spełniają podanego warunku logicznego (niewspierane przez MySQL).



Definicja – ograniczenia integralnościowe

Jedną z bardziej interesujących **funkcji** każdego serwera bazy danych jest **kontrola poprawności wprowadzanych danych**. Polega ona na **automatycznej weryfikacji reguł poprawności danych**, nazywanych **ograniczeniami integralnościowymi** (ang. *integrity constraints*), podczas wykonywania **operacji INSERT, UPDATE i DELETE**. Jeżeli polecenia takie **naruszają co najmniej jedno ograniczenie integralnościowe**, **serwer bazy danych odmawia ich wykonania**. Dzięki temu możemy zapewnić wysoką jakość gromadzonych danych.

Wyróżniamy następujące **typy ograniczeń integralnościowych**:



Kliknij na etykiety, aby zobaczyć szczegółowy opis.

NOT NULL

CHECK

UNIQUE

PRIMARY KEY

REFERENCES

Uniemożliwia wprowadzenie rekordu, którego wartości kolumn powtarzają się z jakimś istniejącym rekordem.



Definicja – ograniczenia integralnościowe

Jedną z bardziej interesujących **funkcji** każdego serwera bazy danych jest **kontrola poprawności wprowadzanych danych**. Polega ona na **automatycznej weryfikacji reguł poprawności danych**, nazywanych **ograniczeniami integralnościowymi** (ang. *integrity constraints*), podczas wykonywania **operacji INSERT, UPDATE i DELETE**. Jeżeli polecenia takie **naruszają co najmniej jedno ograniczenie integralnościowe**, **serwer bazy danych odmawia ich wykonania**. Dzięki temu możemy zapewnić wysoką jakość gromadzonych danych.

Wyróżniamy następujące **typy ograniczeń integralnościowych**:



Kliknij na etykiety, aby zobaczyć szczegółowy opis.

NOT NULL

CHECK

UNIQUE

PRIMARY KEY

REFERENCES

Połączenie NOT NULL i UNIQUE służy do oznaczenia klucza głównego tabeli.



Definicja – ograniczenia integralnościowe

Jedną z bardziej interesujących **funkcji** każdego serwera bazy danych jest **kontrola poprawności wprowadzanych danych**. Polega ona na **automatycznej weryfikacji reguł poprawności danych**, nazywanych **ograniczeniami integralnościowymi** (ang. *integrity constraints*), podczas wykonywania **operacji INSERT, UPDATE i DELETE**. Jeżeli polecenia takie **naruszają co najmniej jedno ograniczenie integralnościowe**, **serwer bazy danych odmawia ich wykonania**. Dzięki temu możemy zapewnić wysoką jakość gromadzonych danych.

Wyróżniamy następujące **typy ograniczeń integralnościowych**:



Kliknij na etykiety, aby zobaczyć szczegółowy opis.

NOT NULL

CHECK

UNIQUE

PRIMARY KEY

REFERENCES

Uniemożliwia wprowadzenie rekordu, którego wartości kolumn nie występują w innej wskazanej tabeli.



Przykład

Zapoznaj się z przykładem **użycia ograniczenia integralnościowego REFERENCES**.

Kliknij na ikonę PDF, a następnie pobierz i przeanalizuj przykład.

