SQL

Structured Query Language

(Strukturalny język zapytań deklaratywny)

SQL umożliwia wykonanie operacji:

- Uzyskiwania informacji
- Modyfikowania informacji
- Dopisywania (wstawiania)
- Usuwania
- Sterownia danymi

Postać SQL-a

SQL jest „podjęzykiem danych”, ponieważ służy on tylko do komunikowania się z bazą. Za pomocą samego SQL-a nie można napisać kompletnego programu. SQL jest używany na trzy sposoby:

- **1. Interaktywny lub samodzielny SQL** używany jest do wprowadzania albo wyszukiwania informacji do/z bazy danych. Na przykład użytkownik może poprosić o listę aktywnych kont w bieżącym miesiącu. Wynik może być wysłany na ekran, skierowany do pliku albo wydrukowany.
- **2. Statyczny SQL** jest to stały kod SQL-a napisany przed wykonaniem programu.
- **3. Dynamiczny SQL** jest to kod SQL-a generowany przez aplikację w czasie jej wykonywania się. Jest używany zamiast statycznego SQL-a, gdy potrzebny kod nie może być określony podczas pisania programu, ponieważ zależy on od wy boru użytkownika.

Trójwartościowa logika *(3VL).*

W większości języków wyrażenie logiczne może być prawdziwe (TRUE) lub fałszywe (FALSE). SQL zezwala na wprowadzanie do tablic wartości NULL (nieznana, nieokreślona). Jest to znacznik wpisywany do kolumny w miejsce brakującej danej. Kiedy wartość NULL jest użyta w porównaniu, wynik nie będzie ani FALSE, ani TRUE, lecz nieznany (UNKNOWN).

Podział strukturalnego języka zapytań na podgrupy wyrażeń

DDL	JDD	Data Definition Language	CREATE, ALTER, DROP, TRUNCATE	Język definiowania danych stosowany do tworzenia schematu danych, definiuje strukturę danych
DML	JMD	Data Manipulation Language	INSERT, UPDATE, DELETE	Język manipulowania danymi służący do wypełnienia bazy danych, usuwania informacji i modyfikowania danych
DCL	JKD	Data Control Language	GRANT, DENY, REVOKE	Język kontrolowania dostępu do danych poprzez uprawnienia użytkowników
DQL	JZ	Data Query Language	SELECT	Język zapytań, którego zadaniem jest pobieranie informacji spełniających określone warunki
TCL	JSP	Transaction Control Language	COMMIT, ROLLBACK, SAVEPOINT	Język sterowania przepływem danych (kontrola transakcji)

Dialekty języka SQL

- T-SQL (Transport SQL) stosowany na serwerach Microsoft SQL Server
- PL/SQL (Procedural Language/SQL) na serwerach firmy ORACLE
- PL/pgSQL (Procedural Language/PostgreSQL) na serwerach PostgreeSQL
- SQL PL (SQL Procedural Language) wersja używana na serwerach IBM

Serwer MySQL

Cechy serwera MySQL:

- Praca w zasadzie na wszystkich platformach
- Udostępnienie różnych silników bazodanowych (np. bardzo szybkie tabele MyISAM lub HEAP)
- Wykorzystywanie systemu przesyłania skompresowanych danych pomiędzy klientem i serwerem

Serwer MySQL

Cechy serwera MySQL /cd/:

- Udostępnienie serwera w postaci osobnego programu lub biblioteki
- Obsługa zapytań rozproszonych
- Udostępnianie mechanizmów replikacji

Serwer MySQL

Zalety serwera MySQL:

- Bardzo szybki, nadający się do obsługi często odwiedzanych stron WWW
- Małe wymagania sprzętowe
- Darmowy i nielimitowany

Serwer MySQL

Wady serwera MySQL:

- Transakcje wymagają korzystania z silnika bazodanowego InnoDB
- Licencja GPL uniemożliwia sprzedaż produktów, których działanie jest powiązane z serwerem MySQL

INSTALACJA MANULANA MYSQL

Uruchamianie jako samodzielnej aplikacji

mysqld --console

mysqld

Zatrzymywanie uruchomionego ręcznie serwera

mysqladmin --user=root --password=hasło shutdown

mysqladmin -u root -p*haslo* shutdown

Ustalenie hasła dla root

mysqladmin -u root password hasło

INSTALACJA USŁUGI SYSTEMOWEJ

mysqld --install nazwa_usługi

mysqld --install-manual nazwa_uslugi

mysqld --remove nazwa usługi

URUCHAMIANIE/ZATRZYMYWANIE USŁUGI SYSTEMOWIEJ MySQL

net start nazwa_usługi

net stop nazwa_usługi

NAWIĄZYWANIE POŁACZENIA Z SERWEREM

mysql -u root --password=hasło

mysql -u root -p*hasło*

KOŃCZENIE POŁĄCZENIA

Quit

Ustawienie instancji serwera MySQL

Developer Machine – minimalna ilość pamięci (zalecane dla twórcy oprogramowania)

Server Machine – średnie zużycie pamięci (zalecane dla serwerów stron internetowych)

Dedicated MySQL Server Machine – maksymalne przydzielenie pamięci (zalecane dla wysoko wydajnych systemów)

Wybór silnika bazy danych

Multifunctional Database

Opcja ta jest propozycją zoptymalizowania instancji serwera, wykorzystuje zarówno szybkość transakcji silnika InnoDB, jak i szybkość silnika MyISAM

Transactional Database Only

Większość zasobów zostanie przydzielona silnikowi InnoDB, a silnik MyISAM zostanie uruchomiony z mniejszym przydziałem zasobów systemu – dla rozwiązań w których należy zoptymalizować bazę pod kątem transakcji generowanych przez aplikace internetowe

Wybór silnika bazy danych

Non-Transactional Database Only

Wybór tej konfiguracji aktywuje tylko silnik MyISAM przydzielając mu zasoby serwera.

Plik konfuguracyjny my-template.ini

Multifunctional Database:
default-storage-engine=InnoDB
_myisam_pct=50

Transactional Database Only:
default-storage-engine=InnoDB
_myisam_pct=5

Non-Transactional Database Only:
default-storage-engine=MyISAM
_myisam_pct=100
skip-innodb

InnoDB Tablespace Setting
Wybór miejsca alokacji danych

Pliki generowane przez InnoDB zawierające tabele mogą być umieszczane w wybranym przez użytkownika miejscu

Wybór liczby aktywnych połączeń

Decision Support (DSS/OLAP)

Opcja zalecana gdy serwer nie wymaga utrzymania zbyt dużej liczby aktywnych połączeń – maksymalna liczba zostanie ustawiona na 100, średnia liczba aktywnych połączeń ustawiona jest na 20

Wybór liczby aktywnych połączeń

Online Transaction Processing (OLTP)

Opcja wybierana jest gdy serwer wymaga dużej liczby aktywnych połączeń (jak w przypadku serwera WWW) – maksymalna liczba zostanie ustawiona na 500

Wybór liczby aktywnych połączeń

Manual Setting

Pozwala na ręczne określenie liczby aktywnych połączeń

Konfiguracja portu i zachowania serwera

Włączenie TCP/IP oraz portu, na którym będzie działał serwer (domyślnie 3306)

Strict Mode – Modyfikuje zachowanie serwera – wyświetlanie komunikatów o błędzie/ zaprzestanie operacji w przypadku próby wprowadzenia niepoprawnych wartości do kolumn tabeli bazy danych

TWORZENIE I USUWANIE BAZY

SHOW DATABASES;

SHOW DATABASES LIKE ‘wzorzec%’;

CREATE DATABASE nazwa_bazy;

DROP DATABSE nazwa_bazy;

Konfiguracja domyślnego systemu kodowania znaków

Standard Charakter Set – zestaw znaków latin

Best Support For Multilinggualism – system kodowania UTF8 (zalecane gdy dane w bazie będą przechowywane w różnych językach)

Manual Selected Default Set/Collation – ręczne określenie kodowania znaków – domyślnego języka, który będzie użyty jak tekst w bazie danych

System kodowania i porównywania znaków

SHOW CHARACTER SET;

SHOW COLLATION;

SHOW COLLATION LIKE ‘LATIN2%’;

WCZYTANIE POLECEŃ Z PLIKU ZEWNĘTRZNEGO

SOURCE test.sql

System Linux

source /usr/nowa/test.sql

System Windows

Source \bazy\test.sql

lub

mysql –u root –p*hasło* nazwa_bazy < nazwa_pliku

POLSKIE STABDARDY ZNAKÓW

- **Latin2 – standard ISO-8859-2** (kodowanie latin2__general_ci)
- **Cp1250 – standard Windows 1250** (kodowanie cp1250__general_ci)
- **Cp852 – standard DOS** (kodowanie cp852__general_ci)
- **Utf8 – standard unicode** (zgodny z dokumentem RFC3629) (kodowanie utf8__general_ci)

USTALENIE DOMYŚLNEGO ZESTAWU ZNAKÓW DLA SERWERA

mysqld --console --character-set-server=utf8

lub na stale w pliku konfiguracyjnym

[client]
port=3306
[mysql]
default-character-set=latin1

USTAWIENIE ZNAKÓW DLA POŁĄCZENIA Z SERWEREM

SET NAMES ‘nazwa zestawu znaków’ [COLLATE reguły_porównywania]

SET NAMES ‘cp1250’;

W systemie Windows zmianę strony kodowej konsoli (domyślnie cp852) można wykonać poleceniem chcp 1250, chcp852

UTWORZENIE BAZY Z DOMYŚLNYM ZESTAWEM ZNAKÓW

CREATE DATABASE nazwa_bazy CHARACTER SET zestaw_znakow;

CREATE DATABASE test2 CHARACTER SET latin2;

UTWORZENIE BAZY Z DOMYŚLNYM ZESTAWEM ZNAKÓW ORAZ Z DOMYŚLNYMI REGUŁAMI PORÓWNYWANIA

```
CREATE DATABASE test3
CHARACTER SET latin2
COLLATE latin2_bin;
```

ZMIANA ZESTAWU ZNAKÓW I REGUŁ PORÓWNYWANIA DLA ISTNIEJĄCEJ BAZY DANYCH

```
ALTER DATABASE test2
CHARACTER SET utf8
COLLATE utf8_polish_ci;
```

Pobieranie poleceń SQL z pliku

- **source create_member.sql;**
- **source create_president.sql;**
- **source insert_member.sql;**
- **source insert_president.sql;**
- **source create_student.sql;**
- **source create_grade_event.sql;**
- **source create_score.sql;**
- **source create_absence.sql;**
- **source insert_student.sql;**
- **source insert_grade_event.sql;**
- **source insert_score.sql;**
- **source insert_absence.sql**

Pobieranie informacji z bazy

```
Select Now();
Select User();
Select Version();
```

Pobieranie metadanych

SHOW DATABESES;

SHOW TABLES;

SHOW TABLES FROM baza_danych;

SHOW COLUMNS FROM tabela;

SHOW INDEX FROM tabela;

SHOW TABLES STATUS;

SHOW VARIABLES;

SHOW ENGINES;

(Zmienna default-domyślny, Yes/No – dostępny, niedostępny, Disabled-dostępny, ale wyłączony)

Silnik bazy w MySQL

- InnoDB
- MyISAM
- ARCHIVE
- BLACKHOLE
- FEDERATED
- CSV

Silnik InnoDB

(obecnie domyślny sinik MySQL)

Tabele zapewniają obsługę transakcji

Automatyczna naprawa po awarii

Sinik MyISAM

- Kompresja kluczy
- Wyszukiwanie pełnego tekstu

Silnik MEMORY

- Używa tabel przechowywanych w pamięci i posiadających rekordy o stałej wielkości
- Tabele są tymczasowe pod tym względem, że ich zawartość znika po zamknięciu serwera, tabela będzie istniała po ponownym uruchomieniu lecz będzie pusta

Silnik ARCHIVE

- Zapewnia archiwalna pamięć masową – jest przeznaczony do przechowywania ogromnej liczby rekordów
- Obsługuje zapytanie INSERT i SELECT (nie można używać DELETE i UPDATE)
- Zawiera jedynie indeksowaną kolumnę Auto_Increment (pozostałe kolumny nie mogą być indeksowane)

Silnik BLACKHOLE

- Tworzy tabele, dla których operacje zapisu są ignorowane, a operacje odczytu nic nie zwracają
- Bazodanowy odpowiednik urządzenia /dev/null w UNIX

Silnik CSV

- Przechowuje dane w formacie wartości rozdzielanych przecinkami

Silnik FEDERATED

- Zapewnia dostęp do tabel obsługiwanych przez inne serwery MySQL

- [mysqld]

Innodb_file_per_table=1

Lub

SET GLOBAL Innodb_file_per_table=1;

Przedstawienie bazy danych w systemie plików

- Każda baza danych ma własny katalog
- W katalogu znajduje się plik db.opt zawierający atrybuty bazy danych (np.domyślne kodowanie)

Przedstawienie tabel w systemie plików

- Każda tabela jest przedstawiona za pomocą co najmniej jednego pliku

plik.frm – zawiera opis struktury

Przedstawienie tabel w systemie plików – SILNIK INNODB

Systemowa przestrzeń tabel

plik.frm – zawiera opis struktury, dane i indeksy

Poszczególne przestrzenie tabel

plik.frm – zawiera opis struktury

plik.ibd – zawierający dane i jej indeksy

Przedstawienie tabel w systemie plików – SILNIK MYISAM

plik.frm – zawiera opis struktury

plik.myd – zawiera zawartość rekordów tabeli

plik.myi – zawiera informacje o indeksach utworzonych dla tabeli

Pobieranie informacji z bazy danych INFORMATION_SCHEMA

Baza danych INFORMATION_SCHEMA jest wirtualną bazą, w której tabele są widokami dla różnego rodzaju metadanych bazy danych.

Pobieranie informacji z bazy danych INFORMATION_SCHEMA

SCHEMATA, TABLES, VIEWS, ROUTINES, TIGGERS, EVENTS, PARAMETERS, PARTITIONS, COLUMNS – *informacje o bazach, tabelach, wyzwalaczach, zdarzeniach, parametrach*

CHARAKTER_SETS, COLLATION – *informacje o obsługiwanych kodowaniach*

USER_PREVILIGES, SCHEMA_PREVILGES – informacje o uprawnieniach pochodzące z tabel user,db,tables_priv,columns_priv w bazie danych SQL

Plik przekierowany do i z mysql

- mysql baza_danych < plik.sql
- mysql baza_danych < plik.sql > plik.out

Lub

- mysql –t baza_danych < plik.sql > plik.out

Przeniesienie całego katalogu danych

1. Zatrzymać serwer MySQL
2. Przenieść katalog danych do nowego położenia
3. Uruchomić serwer z opcją --datadir=‘nowe położenie’ lub zmienić plik konfiguracyjny my.ini

mysqld --console --datadir=‘polozenie bazy’

Przeniesienie poszczególnych baz UNIX

1. Zatrzymać serwer MySQL
2. Przenieść katalog bazy danych do nowego położenia, czyli jego skopiowanie do nowej lokalizacji i usuniecie z początkowej
3. Utworzenie w katalogu danych MySQL dowiązania symbolicznego o nazwie takiej samej jak oryginalna baza danych i prowadzącego do nowej lokalizacji
4. Ponowne uruchomienie serwera

Przeniesienie poszczególnych baz UNIX

Przykład: przeniesienie bazy danych test z katalogu /usr/mysql/data do katalogu /var/db

% mysqladmin –u root –p shutdown

Enter password:****

% cd /usr/local/mysql/data

% tar cf – test | (cd /var/db; tar xf -)

% rm –rf test

% ln –s /var/db/test test

% mysqld_safe &

Przeniesienie poszczególnych baz Windows

1. Zatrzymać serwer MySQL
2. Przenieść katalog bazy danych do nowego położenia
3. Utworzenie w katalogu danych MySQL pliku działającego w charakterze dowiązania symbolicznego –utworzony plik powinien nazywać się jak baza i mieć rozszerzenie .sym

Przykład: Przeniesienie bazy test z katalogu C:\mysql\data\test do katalogu E:\baza\test

Utwórz w katalogu C:\mysql\data plik test.sym zawierający zapis: E:\baza\test

Tworzenie konta użytkownika bez nadawania hasła

CREATE USER andrzej;

(użytkownik andrzej będzie się logował bez hasła)

Usuwanie konta użytkownika

DROP USER nazwa_użytkownika

Modyfikacja nazwy konta użytkownika

RENAME USER ‘stara nazwa’ TO ‘nowa nazwa’

Zmiana hasła do konta

SET PASSWORD FOR nazwa_uzytkownika = PASSWORD(‘1234’)

Tworzenie i usuwanie kont użytkownika

CREATE USER nazwa [IDENTIFIED BY ‘haslo’]

CREATE USER edzio IDENTIFIED BY ‘1234’;

Tworzenie konta użytkownika z jednoczesnym nadaniem hasła

CREATE USER janek IDENTIFIED BY PASSWORD ‘1234’;

lub

CREATE USER janek IDENTIFIED BY ‘1234’;

Tworzenie i usuwanie kont użytkownika

Nazwa użytkownika

‘nazwa_uzytkownika’@’nazwa_hosta’

Dla użytkownika lokalnego user1, który będzie się logował z komputera lokalnego

‘user1’@’localhost’

lub

user1@localhost

Tworzenie i usuwanie kont użytkownika

Dla użytkownika user1, który będzie się logował z komputera host1.mojadomena.com należy użyć konstrukcji

‘user1’@’host1.mojadomena.com’

Dla użytkownika user1, który będzie się logował z dowolnego komputera w danej domenie należy użyć konstrukcji

‘user1’@’%.mojadomena.com’

Tworzenie i usuwanie kont użytkownika

Nazwa hosta może być podana w postaci adresu IP, który określa komputer z którego użytkownik user1 będzie się logował

‘user1’@’212.77.100.101’

Dla użytkownika user1, który będzie miał prawo logować się z dowolnego komputera w danej sieci

‘user1’@’212.77.100.%’

Tworzenie i usuwanie kont użytkownika

Konta użytkowników, którzy będą mieli prawo logowania się z komputerów o dowolnych nazwach i adresach

user1

Tylko nazwa user1 jest traktowana jako:

‘user1’@’%’

SELECT

```
SELECT * FROM Klient;
SELECT DISTINCT kod_pocztowy, miejscowosc
    FROM Klient;
SELECT tytul, cena, cena * 0.07 AS Marza FROM
    Ksiazki;
SELECT tytul,cena FROM Ksiazki ORDER BY tytul;
SELECT tytul,cena FROM Ksiazki ORDER BY 2;
```

SELECT

SELECT tytul,cena FROM Ksiazki ORDER BY tytul ASC, cena DESC;

SELECT TOP 1 tytul,cena FROM Ksiazki ORDER BY cena DESC;

SELECT TOP 1 WITH TIES tytul,cena FROM Ksiazki ORDER BY cena DESC;

SELECT * FROM Ksiazki WHERE cena>50;

SELECT * FROM Ksiazki WHERE cena>=50 AND cena<=100;

SELECT * FROM Ksiazki WHERE cena between 50 and 100;

SELECT * FROM Ksiazki WHERE tytul LIKE „Fizyka%”

SELECT * FROM Ksiazki WHERE tytul in (‘element1’,’element2’)

Agregowanie danych

COUNT(nazwa kolumny)-zwraca liczbę wierszy w grupie

SUM(nazwa kolumny) - suma

AVG(nazwa kolumny) – wartość średnia

MAX(nazwa kolumny)- największa wartość

MIN(nazwa kolumny) – najmniejsza wartość

Grupowanie danych

SELECT COUNT(*) AS „liczba uczniów” FROM tblUczniowie

SELECT COUNT(nazwisko) AS „liczba uczniów” FROM tblUczniowie

SELECT AVG(Year(Date())-Year(Dataurodzenia)) AS ‘’średni wiek” FROM tblUczniowie WHERE idzawodu=104;

Grupowanie danych

```
SELECT count(nazwisko), idzawodu
FROM tblUczniowie
GROUP BY idzawodu;
```

Grupowanie danych

```
SELECT MAX(poziom)-MIN(poziom) FROM
    tblUczniowie

SELECT  TOP 2 COUNT(idzawodu) AS 'liczba
    uczniow', idzawodu FROM tblUczniowie
    GROUP BY idzawodu ORDER BY 1 DESC;
```

Klauzula HAVING

```
SELECT count(nazwisko), idzawodu
FROM tblUczniowie WHERE idzawodu in (103,108)
GROUP BY idzawodu
HAVING COUNT(nazwisko)>20
ORDER BY 1;
```

Łączenie tabel – połączenie zwężające

```
Tabela1 INNER JOIN tabela2 ON
   tabela1.kolumna1=tabela2.kolumna2
```

```
SELECT tbluczniowie.nazwisko,
   tbluczniowie.idzawodu, tblzawody.nazwazawodu
   FROM tbluczniowie INNER JOIN tblzawody ON
   tbluczniowie.idzawodu=tblzawody.idzawodu;
```

Łączenie tabel – lewe połączenie rozszerzające

Tabela1 LEFT JOIN tabela2 ON
tabela1.kolumna1=tabela2.kolumna2

SELECT tbluczniowie.nazwisko,
tblzawody.nazwazawodu FROM tbluczniowie
LEFT JOIN tblzawody ON
tbluczniowie.idzawodu=tblzawody.idzawodu;

Łączenie tabel – prawe połączenie rozszerzające

```
Tabela1 RIGHT JOIN tabela2 ON
   tabela1.kolumna1=tabela2.kolumna2
```

```
SELECT tbluczniowie.nazwisko,
   tblzawody.nazwazawodu FROM tbluczniowie
   RIGHT JOIN tblzawody ON
   tbluczniowie.idzawodu=tblzawody.idzawodu;
```

Tworzenie prostej tabeli

```
CREATE TABLE Klient
(
   ID INTEGER,
   Nazwa VARCHAR(20)
);
```

Tworzenie tabeli o ile nie istnieje w bazie

```
CREATE TABLE IF NOT EXISTS Klient
(
   ID INTEGER,
   Nazwa VARCHAR(20)
);
```

Tworzenie tabeli tymczasowej

```
CREATE TEMPORARY TABLE Test
(
   ID INTEGER,
   Wartosc INTEGER
);
```

Tworzenie tabeli na podstawie innej

```
CREATE  TABLE Test2 LIKE Test
```

Typy całkowitoliczbowe

- BIT
- BOOL
- BOOLEAN
- TINYINT (od -128 do 127 lub od 0 do 255)
- SMALLINT (od -32768 do 32767)
- MEDIUMINT (od -8 388 608 do 8388 607)
- INT (od -2 147 483 648 do 2 147 483 647)
- INTEGER (4 bajt)
- BIGINT (8 bajt)

Tabela zawiera kolumnę typu INTEGER przechowująca wartość z przedziału 0-65535, liczba wyświetlanych znaków będzie zawsze równa 5, a wolne miejsca dla wartości krótszych niż 5 znaków będą wypełnione zerami

```
CREATE TABLE Test1
(
   ID SMALLINT(5) ZEROFILL
);
```

Typy zmiennoprzecinkowe

- FLOAT
- DOUBLE
- REAL (synonim DOUBLE)
- DECIMAL (DEC, NUMERIC)

Tabela zawierająca wartości rzeczywiste

```
CREATE TABLE Test3
(
    ID INTEGER,
    Wartosc FLOAT
);
```

Tabela zawierająca wartości rzeczywiste o określonej precyzji

```
CREATE TABLE Test4
(
    Wartosc DECIMAL(6, 3)
);
```

Typy daty i czasu

- DATE
- DATETIME
- TIMESTAMP
- TIME
- YEAR

Tabela zawierająca datę

```
CREATE TABLE Test5
(
    ID INTEGER;
    Data DATE
);
```

Tabela zawierająca datę i czas

```
CREATE TABLE Test6
(
    Dataiczas  DATETIME
);
```

Typy łańcuchowe (przechowują ciągi znaków i dane binarne)

- CHAR (długość od 0 do 255 znaków-stała szerokość)
- VARCHAR (od 0 do 65535 znaków – zmienna długość)
- BINARY i VARBINARY (jak CHAR i VARCHAR – parametr określa liczbę bajtów)
- BLOB i TEXT (do przechowywania dużej ilość danych binarnych/tekstowych)
- ENUM i SET (pozwala ograniczyć wartości do określonego zbiory wyliczeniowego)

Tabela zawierająca dane tekstowe

```
CREATE TABLE Test8
(
    ID INTEGER,
    Imie CHAR(7),
    Nazwisko VARCHAR(8)
);
```

Tabela zawierająca text

```
CREATE TABLE Test10
(
   ID INTEGER,
   Opis TEXT
);
```

Tabela zawierająca typ wyliczeniowy (kolumna kolor przyjmuje jedną wartość z listy)

```
CREATE TABLE Test11
(
   ID INTEGER,
  Kolor ENUM(‘zielony’, ‘czerwony’)
);
```

Tabela zawierająca typ wyliczeniowy (kolumna kolor może przyjmować wiele wartość z listy)

```
CREATE TABLE Test12
(
   ID INTEGER,
  Kolor SET('zielony', 'czerwony')
);
```

Atrybuty kolumn

- PRIMARY KEY (prosty)

```
CREATE TABLE nazwa_tabeli
( nazwa_kolumny1 typ_kolumny PRIMARY KEY,
   definicje pozostałych zmiennych);
```

Atrybuty kolumn

- PRIMARY KEY (prosty)

```
CREATE TABLE test21
( Id INTEGER PRIMARY KEY AUTO_INCREMENT,
  Nazwa VARCHAR(20));
```

Atrybuty kolumn

- PRIMARY KEY (złożony)

```
CREATE TABLE nazwa_tabeli
( nazwa_kolumny1 typ_kolumny1,
   nazwa_kolumny2 typ_kolumny2,
   definicje pozostałych zmiennych,
   PRIMARY KEY (nazwa_kolumny1, nazwa_kolumny2)
);
```

Atrybuty kolumn

- PRIMARY KEY (złożony)

```
CREATE TABLE test22
( Id INTEGER,
   Nazwa VARCHAR(20),
   PRIMARY KEY (Id, Nazwa)
);
```

Atrybuty kolumn

- Automatyczne generowanie kolejnej wartości

```
CREATE TABLE test22a
( Id INTEGER  AUTO_INCREMENT,
   Nazwa VARCHAR(20),
   PRIMARY KEY (Id, Nazwa)
);
```

Atrybuty kolumn

- Testowanie AUTO_INCREMENT

INSERT INTO test22a (Nazwa) VALUES (‘jeden’);

INSERT INTO test22a (Nazwa) VALUES (‘dwa’);

INSERT INTO test22a (Nazwa) VALUES (‘dwa’);

Atrybuty kolumn

- Unikatowe wartości w kolumnie

```
CREATE TABLE test24
( Id INTEGER PRIMARY KEY AUTO_INCREMENT,
  Nazwa VARCHAR(20) UNIQUE
 );
```

Atrybuty kolumn

- Testowanie unikatowych wartości w kolumnie

INSERT INTO test24 (Nazwa) VALUES (‘jeden’);

INSERT INTO test24 (Nazwa) VALUES (‘dwa’);

INSERT INTO test24 (Nazwa) VALUES (‘dwa’);

Atrybuty kolumn

- brak wartości pustych

```
CREATE TABLE test25
( Id INTEGER PRIMARY KEY AUTO_INCREMENT,
   Nazwa VARCHAR(20) NOT NULL
 );
```

Atrybuty kolumn

- Testowanie brak wartości pustych

INSERT INTO test25 (Nazwa) VALUES (NULL);

Atrybuty kolumn

- Wartość domyślna

```
CREATE TABLE Uczniowie
( Id_ucznia INTEGER PRIMARY KEY
   AUTO_INCREMENT,
  Nazwisko VARCHAR(20) NOT NULL,
  Miasto VARCHAR(20) DEFAULT ‘Chelm’
 );
```

Atrybuty kolumn

- Testowanie wartości domyślnych

INSERT INTO Uczniowie (Nazwisko) VALUES (‘Kowalski’);

INSERT INTO Uczniowie (Nazwisko) VALUES (‘Nowak’);

INSERT INTO Uczniowie (Nazwisko,Miasto) VALUES (‘Piotrowski’,’Lublin’);

KLUCZ OBCY

[CONSTRAINT nazwa] FOREIGN KEY (kolumna1, kolumna2, …..) REFERENCES nazwa_tabeli (kolumna1, kolumna2, ……)

KLUCZ OBCY

```
CREATE TABLE OCENY
( Id_oceny  INT PRIMARY KEY AUTO_INCREMENT,
   uczen INT,
   Ocena  CHAR(1) NOT NULL,
   Data DATE DEFAULT ‘2014-09-15’,

CONSTRAINT   fkuczniowie   FOREIGN  KEY   (uczen)
REFERENCES   Uczniowie  (id_ucznia)

);
```

KLUCZ OBCY

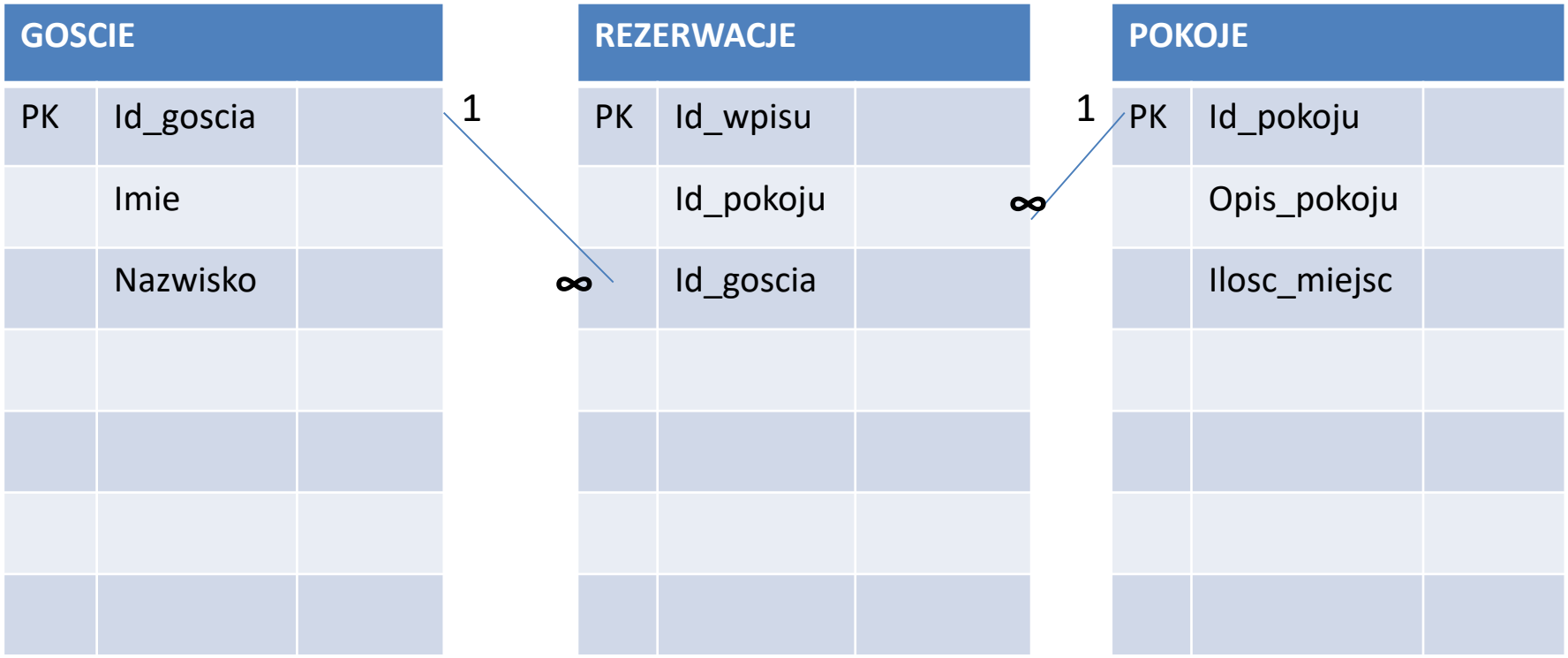
- Testowanie klucza obcego

```
INSERT INTO Oceny  (uczen,ocena)  VALUES (2,5);

INSERT INTO Oceny  (uczen,ocena)  VALUES (3,3);

INSERT INTO Oceny  (uczen,ocena)  VALUES (5,4);
```

Projekt


GOSCIE
PK
Id_goscia
Imie
Nazwisko
1
∞
REZERWACJE
PK
Id_wpisu
Id_pokoju
Id_goscia
1
∞
POKOJE
PK
Id_pokoju
Opis_pokoju
Ilosc_miejsc

Wyświetlenie struktury tabel

- SHOW COLUMNS FROM test26;

Odtwarzanie struktury tabel

- SHOW CREATE TABLE test26

Tworzenie indeksów

CREATE INDEX nazwa_indeksu ON nazwa_tabeli
(kolumny składowe)

Wyświetlenie indeksów

SHOW INDEX FROM Nazwa_tabeli

Tworzenie indeksów

```
CREATE TABLE Mieszkancy
( Id INTEGER PRIMARY KEY AUTO_INCREMENT,
   Nazwisko VARCHAR(20) NOT NULL,
   Imie VARCHAR(20) NOT NULL,
   Miasto VARCHAR(20) DEFAULT ‘Chelm’,
   Ulica VARCHAR (30) NOT NULL,
   Nr_domu VARCHAR(10),
   INDEX in_nazwisko (Nazwisko),
   INDEX in_Imie (Imie)
  );
```

Tworzenie indeksów

Indeks dla pojedynczej kolumny

CREATE INDEX in_miasto ON Mieszkancy (Miasto)

Tworzenie indeksów

Indeks złożony z kilku kolumn

```
CREATE INDEX in_ulica ON Mieszkancy (Ulica,
    Nr_domu)
```

Modyfikowanie tabel

ALTER TABLE nazwa_tabeli zmiana1, [zmina2],….

Modyfikowanie tabel

ADD [COLUMN] (definicja kolumny) – dodaje nową kolumnę

ADD [COLUMN] definicja kolumny [FIRST | AFTER nazwa_kolumny] – umożliwia na umiejscowienie nowej kolumny w tabeli

ADD [CONSTRAINT nazwa] PRIMARY KEY (kolumna1,…) – dodaje klucz główny do tabeli

Modyfikowanie tabel

ADD [CONSTRAINT nazwa] FOREIGN KEY (kolumna1,…) REFERENCES Tabela1 (kolumna1,…) definicja_odniesienia– dodaje klucz obcy

ALTER [COLUMN] nazwa SET DEFAULT wartość – określa wartość domyślną dla kolumny nazwa

ALTER [COLUMN] nazwa DROP DEFAULT – usuwa wartość domyślną dla kolumny nazwa

Modyfikowanie tabel

CHANGE [COLUMN] stara_nazwa definicja_nowej_kolumny [FIRST/AFTER naza_kolumny] – zmienia kolumnę starą na zdefiniowaną przez definicja_nowej_kolumny

DROP [COLUMN] nazwa –usuwa kolumnę nazwa

DROP PRIMARY KEY – usuwa z tabeli klucz główny

DROP INDEX nazwa – usuwa index nazwa

DROP FOREIGN KEY nazwa – usuwa klucz obcy nazwa

Modyfikowanie tabel

ORDER BY nazwa – pozwala na ustawienie wierszy w porządku określonym przez dane w kolumnie nazwa

RENAME [TO] nowa_nazwa – zmienia nazwę tabeli na nowa_nazwa

Modyfikowanie tabel

CONVERT TO CHARACTER SET nazwa [COLLATE collation_name] – wykonuje konwersję zestawu znaków (strony kodowej)

Modyfikowanie tabel

```
CREATE TABLE Ksiazki
(
Id INTEGER,
AutorId INTEGER,
Tytul VARCHAR (15)
);
```

Modyfikowanie tabel

Dodawanie klucza podstawowego

ALTER TABLE Ksiazki ADD PRIMARY KEY (Id);

Usuwanie klucza podstawowego

ALTER TABLE Ksiazki DROP PRIMARY KEY

Modyfikowanie tabel

Dodawanie klucza podstawowego z więcej niż z jednej kolumny

ALTER TABLE Ksiazki ADD PRIMARY KEY (Id, Tytul);

Zmiana atrybutu AUTO_INCREMENT

ALTER TABLE Ksiazki MODIFY COLUMN Id INTEGER AUTO_INCREMENT

Modyfikowanie tabel

Zmiana właściwości pola

```
ALTER TABLE Ksiazki MODIFY COLUMN Tytul
    VARCHAR (50);
```

Modyfikowanie tabel

Dodawanie do tabeli nowej kolumny

```
ALTER TABLE Ksiazki ADD COLUMN ISBN
    VARCHAR(13);
```

Usuwanie kolumny

```
ALTER TABLE Ksiazki DROP COLUMN ISBN;
```

Modyfikowanie tabel

Dodawanie do tabeli nowej kolumny w konkretnym miejscu

ALTER TABLE Ksiazki ADD COLUMN ISBN VARCHAR(13) AFTER Id;

Modyfikowanie tabel

Dodanie indeksu

ALTER TABLE Ksiazki ADD INDEX in_tytul_isbn (Tytul, ISBN);

Modyfikowanie tabel

Zmiana nazwy tabeli

ALTER TABLE Ksiazki RENAME Books;

Dodanie wartości domyślnej

ALTER TABLE Books ALTER COLUMN ISBN SET DEFAULT ‘brak’;

Usuwanie wartości domyślnej

ALTER TABLE Books ALTER COLUMN ISBN DROP DEFAULT ;

Modyfikowanie tabel

Konwersja zestawu znaków

```
ALTER TABLE Books CONVERT TO CHARACTER
    SET latin2;
```

Usuwanie tabel

```
DROP TABLE Books
```

Zadanie – Modyfikowanie tabel

Osoba
Id
Imie
Nazwisko
Data_urodzenia
Miejsce_urodzenia
Adres_Id

Adres
Id
Ulica
Nr_domu
Nr_mieszkania
Kod
Miejscowosc

Zadanie – Modyfikowanie tabel

Utwórz tabelę Osoba:

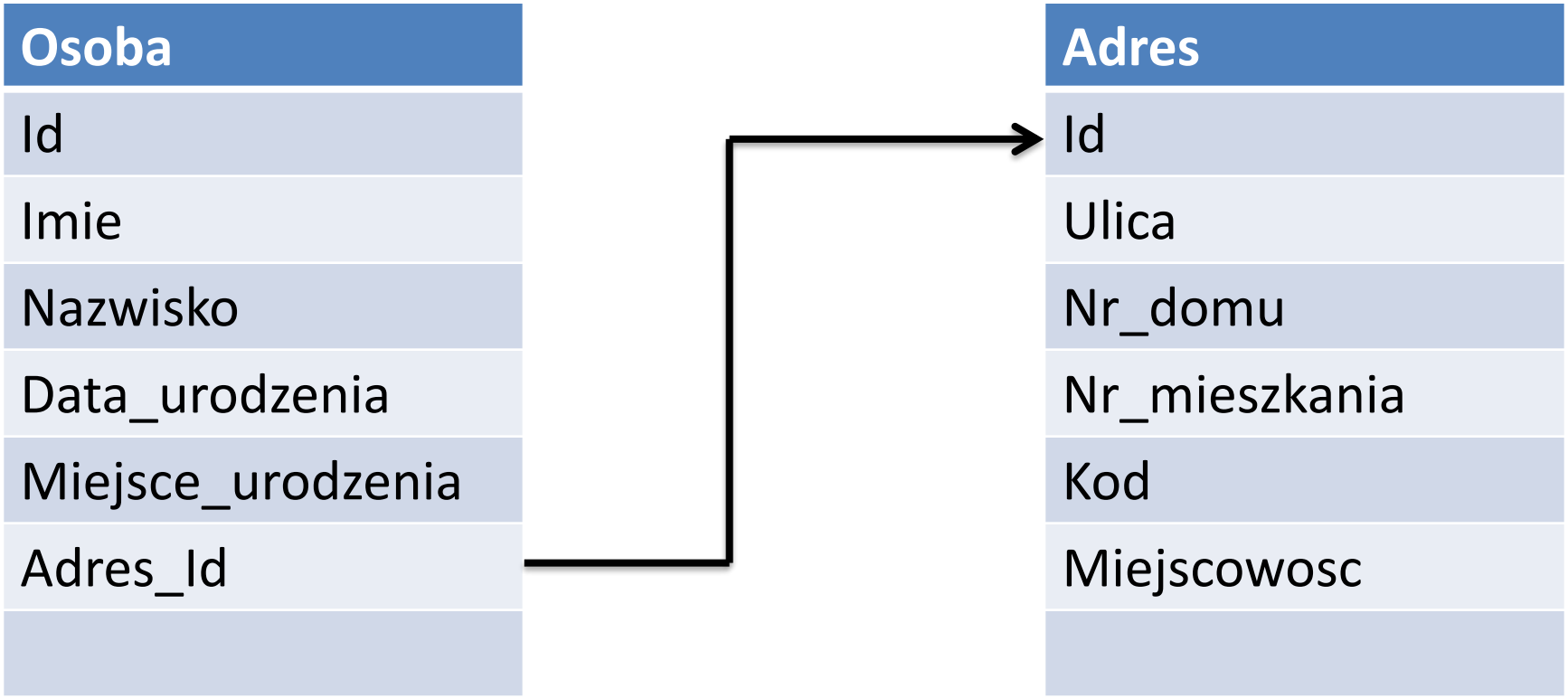
- Id – Integer, PK, autonumerowanie
- Imie – VARCHAR(20), nie puste
- Nazwisko – VARCHAR(45), nie puste
- Data_urodzenia – typ DATE
- Miejsce_urodzenia – typ VARCHAR(30)
- Adres_Id – INT

Zadanie – Modyfikowanie tabel

Utwórz tabelę Adres:

- Id – Integer
- Ulica – VARCHAR(30)
- Nr_domu – VARCHAR(5)
- Nr_mieszkania – VARCHAR(5)
- Kod – VARCHAR(6)
- Miejscowosc – VARCHAR(30)

Zadanie – Modyfikowanie tabel



Zadanie – Modyfikowanie tabel

Zadanie 3

Wstaw klucz obcy łączący tabele

Zadanie – Modyfikowanie tabel

Zadanie 4

Zmodyfikuj tabele w taki sposób, aby możliwe było odzwierciedlenie sytuacji, kiedy adres zamieszkania jest inny niż adres korespondencyjny:

- Usuń klucz obcy z poprzedniego zadania
- Usuń kolumnę Adres_Id z tabeli Osoba
- Dodaj kolumnę Adres_zam_id
- Dodaj kolumnę Adres_kor_id

Zadanie – Modyfikowanie tabel

Zadanie 5

Dodaj klucz obcy do kolumn Adres_zam_id i Adres_kor_id wiążący te kolumny więzami integralności referencyjnej z kolumną Id tabeli Adres

Modyfikowanie tabel

Utwórz tabelę pracownicy

- Id_pracownika – integer
- Imie – Varchar(20)
- Nazwisko – Varchar(30)
- Wiek - integer

Modyfikowanie tabel

Wprowadź dane do tabeli:

Jan Nowak – 24 lata

Wojciech Kowalski – 30 lat

Marian Nowak – 33 lata

Marcin Tracz – 26 lat

Ewa Werner – 23 lata

Ustaw id_pracownika jako klucz główny z autonumeracją

Dodaj pole Wyksztalcenie typ tekstowego – długość 15

Uzupełnij dla wszystkich rekordów wartość pola Wyksztalcenie na „Zawodowe”

Zmień kolumnę Wyksztalcenie by wartosc domyślna była ustawiona na „Średnie”

Dodaj do tabeli wiersze

Marian Nowak – 33 lata

Marcin Tracz – 26 lat

Ewa Werner – 23 lata

Dodaj do tabeli kolumnę Premia typu numeric

Wpisz do kolumny Premia wartość 100 dla pracowników z wiekiem ponad 30 lat, dla pozostałych Premia wynosi 50

Modyfikowanie tabel

Zadanie 1 – Zmień typ kolumny wiek na numeric

Wprowadzanie danych

INSERT [INTO] tabela [(kolumna1, kolumna2, …)]

VALUES (wartość1, wartość2, …)

Wprowadzanie danych

Dla przykłady utwórz tabelę Klienci:

```
CREATE TABLE Klienci
(
   KlientId INT PRIMARY KEY AUTO_INCREMENT,
   Imie VARCHAR(20),
   Nazwisko VARCHAR(25),
   Adres VARCHAR (60)
);
```

Wprowadzanie danych z użyciem nazw kolumn

```
INSERT INTO Klienci
(Nazwisko ,KlientId, Imie, Adres)
VALUES
('Kowalski', 1, 'Jan', 'Bezek');
```

Wprowadzanie danych wszystkich kolumn

```
INSERT INTO Klienci
VALUES
(2, ‘Ewa’, ‘Nowak’, ‘Warszawa’);
```

Wprowadzanie danych wszystkich kolumn z automatycznym generowaniem wartość

```
INSERT INTO Klienci
VALUES
(NULL, ‘Piotr’, ‘Wrona’, ‘Bezek’);
```

Wprowadzanie danych – druga postać instrukcji INSERT

INSERT [INTO] tabela

SET kolumna1=wartość1, kolumna2=wartość2,…

Wprowadzanie danych – druga postać instrukcji INSERT

```
INSERT INTO Klienci SET
Imie='Agata',
Nazwisko='Szpak',
Adres='Warszawa';
```

Wprowadzanie wielu wierszy

INSERT [INTO] tabela [(kolumna1, kolumna2, …)]

VALUES

(wartość1a, wartość2a, …),

(wartość1b, wartość2b, …),

(wartość1c, wartość2c, …);

Wprowadzanie wielu wierszy

```
INSERT INTO Klienci
(Imie, Nazwisko, Adres)
VALUES
('Zygmunt', 'Kowalski', 'Gdynia'),
( 'Wojciech', 'Kowalski', 'Gdynia'),
('Wojciech', 'Ufnal', 'Gdynia');
```

Modyfikacja danych

UPDATE tabela

SET kolumna1=wartość1, kolumna2=wartość2,..

[WHERE warunek]

UPDATE Klienci

SET Imie=‘Aleksander’

WHERE Imie=‘Wojciech’;

Usuwanie danych z tabeli

DELETE FROM tabela

[WHERE warunek]

DELETE FROM KLIENCI

WHERE Imie=‘Aleksander’;

Modyfikacja danych dla istniejącego klucza głównego **(złączenie operacji DELETE i INSERT)**

REPLACE [INTO] tabela [(kolumna1, kolumna2, …)

VALUES

(wartość1a, wartość2a, …),

(wartość1b, wartość2b, …),

(wartość1c, wartość2c, …);

Modyfikacja danych dla istniejącego klucza głównego **(złączenie operacji DELETE i INSERT)**

```
REPLACE INTO Klienci
(KlientId, Imie, Nazwisko, Adres)
VALUES
(1, ‘Zygmunt’, ‘Grabik’, ‘Strupin’);
```

Kaskadowe usuwanie i aktualizowanie danych

- NO ACTION – dane w powiązanych tabelach nie będą automatycznie aktualizowane
- SET NULL – zmodyfikowane wartości klucza podstawowego mają zostać zastąpione wartością NULL
- CASCADE – modyfikacja ma zostać powtórzona we wszystkich powiązanych tabelach

Kaskadowe usuwanie i aktualizowanie danych

…….. CONSTRAINT nazwa_klucza

FOREIGN KEY (kolumna1)

REFERENCES Tabela2(kolumna2)

ON UPDATE CASCADE

ON DELETE SET NULL;

Kaskadowe usuwanie i aktualizowanie

Utwórz tabelę Kontrahenci:

- Firma – CHAR(15), Primary Key
- Miasto – VARCHAR(20), NOT NULL
- Telefon VARCHAR(9)

Kaskadowe usuwanie i aktualizowanie

Do tabeli wpisz dane:

- Fides, Chełm, 1111111111
- Albatros, Lublin, 222222222
- Dombud,Warszawa,333333333

Kaskadowe usuwanie i aktualizowanie

Za pomocą instrukcji REPLACE zmień dane firm

- Dombud,Lublin,333333333
- Starfax,Lublin,555555555

Kaskadowe usuwanie i aktualizowanie

Utwórz tabelę FAKTURY:

- Id – Integer, Primary Key, Autonumerowanie
- Firma – CHAR(15),
- Faktura– VARCHAR(20), NOT NULL
- Kwota - DECIMAL(10,2)
- Utwórz klucz obcy o nazwie fk1 dla kolumna Firma z tabelą Kontrahenci – kolumna Firma

Kaskadowe usuwanie i aktualizowanie

Wpisz do tabeli Faktury dane: (użyj instrukcji INSERT INTO ……. VALUES……)

NULL, ‘Fides’, ’F 1’, 120.25

NULL, ‘Albatros’, ’F 5’, 10.00

NULL, ‘Dombud’, ’F 25’, 125.00

NULL, ‘Albatros’, ’F 51’, 100.00

NULL, ‘Dombud’, ’F 255’, 1250.10

Kaskadowe usuwanie i aktualizowanie

Wpisz do tabeli Faktury dane: (użyj instrukcji INSERT INTO ……. SET……)

‘Fides’, ’F 88’, 95.60

‘Albatros’, ’F 55’, 19.00

‘Starfax’, ’F 66’, 55.10

Kaskadowe usuwanie i aktualizowanie

Usuń z tabeli Kontrahenci firmę Starfax

Jaki jest rezultat ?

Usuń z tabeli Faktury wpis dla firmy Starfax

Usuń z tabeli Kontrahenci firmę Starfax

Jaki jest rezultat ?

Kaskadowe usuwanie i aktualizowanie

Usuń klucz obcy z tabeli Faktury.

Utwórz nowy klucz obcy w tabeli Faktury dodając opcje:

ON UPDATE CASCADE

ON DELETE CASCADE

Kaskadowe usuwanie i aktualizowanie

Za pomocą instrukcji UPDATE zaktualizuj dane w tabeli Kontrahenci:

- zmień nazwę firmy Albatros na Czapla
- sprawdź zawartość tabeli Faktury
- usuń z tabeli Kontrahenci firmę Czapla
- sprawdź zawartość tabeli Faktury

Kaskadowe usuwanie i aktualizowanie

Usuń klucz obcy z tabeli Faktury.

Utwórz nowy klucz obcy w tabeli Faktury dodając opcje:

ON UPDATE SET NULL

ON DELETE SET NULL

Kaskadowe usuwanie i aktualizowanie

- za pomocą instrukcji DELETE usuń z tabeli Kontrahenci firmę FIDES
- sprawdź zawartość tabeli Faktury

Transakcje

Akcja lub seria akcji przeprowadzonych przez pojedynczego użytkownika lub aplikację w celu uzyskania dostępu związanego z odczytem lub modyfikacją zawartości bazy danych nazywamy transakcją.

Transakcje

Poprawnie wykonana transakcja powinna mieć pięć podstawowych cech:

- **ATOMIC (Niepodzielna)**
- **CONSISTENT (Spójna)**
- **ISOLATED (Izolowana)**
- **DURABLE (Trwała)**
- Współbieżna

Transakcje

Transakcja – Cecha: ATOMIC (Niepodzielna)

Transakcja może składać się z kilku akcji. Gdy zdarzy się, że któraś z akcji narazi bazę danych na utratę spójności to całość transakcji nie powinna być wykonana.

Transakcje

Transakcja – Cecha: CONSISTENT (Spójna)

Transakcja nie mogą pozostawić bazy danych w stanie nie spójnym.

Transakcje

Transakcja – Cecha: ISOLATED (Izolowana)

Baza danych musi blokować (izolować) dane w momencie trwania jednej transakcji.

Transakcje

Transakcja – Cecha: DURABLE (Trwała)

Po wykonaniu transakcji zmiany powinny zostać utrwalone na wypadek awarii oprogramowania lub sprzętu.

Transakcje

Transakcja – Cecha: Współbieżna

Cecha, która pozwala na pracę z bazą danych wielu użytkownikom w tym samym czasie z tymi samymi danymi i ma zagwarantować, że operacje wykonywane przez tych użytkowników nie wprowadzą bazy danych w stan niespójny.

Przebieg transakcji można podzielić na trzy etapy:

1. Rozpoczęcie transakcji
2. Wykonanie instrukcji składających się na transakcję
3. Zatwierdzenie lub wycofanie transakcji

Transakcje

Przebieg transakcji można podzielić na trzy etapy:

BEGIN; (START TRANSACTION;)

instrukcje wchodzące w skład transakcji

COMMIT (zatwierdzenie) lub **ROLLBACK** (wycofanie)

Transakcje

W MySQL domyślnie każda instrukcja stanowi osobną transakcję

Transakcje

Utwórz tabelę Rachunki:

- Konto Integer, Primary Key
- Wlasciciel VARCHAR(15), Not Null
- Saldo Decimal(11,2), Not Null

Transakcje

Zarejestruj w tabeli Konta operację przelewu na kwotę 40.00 z konta id=2 na konto id=1:

- UPDATE konto SET saldo=saldo-40 WHERE id=1
- UPDATE konto SET saldo=saldo+40 WHERE id=2

Transakcje

Zarejestruj w tabeli Konta **transakcję** przelewu na kwotę 40.00 z konta id=2 na konto id=1:

```
START TRANSACTION;
UPDATE rachunki SET saldo=saldo-40 WHERE konto=1;
UPDATE rachunki SET saldo=saldo+40 WHERE konto=2;
COMMIT;
```

Transakcje

Zarejestruj w tabeli Konta **transakcję** przelewu na kwotę 50.00 z konta id=2 na konto id=1:

```
START TRANSACTION;
UPDATE konto SET saldo=saldo-40 WHERE id=1;
UPDATE konto SET saldo=saldo+40 WHERE id=2;
COMMIT;
```

Punkt przywracania

SAVEPOINT nazwa_punktu_przywracania

…..

….

ROLLBACK TO nazwa_punktu_przywracania

Kontrola współbieżności

Współbieżność to zdolność do jednoczesnego realizowania wielu procesów (wątków) opartych na wspólnych danych

Kontrola współbieżności

Model optymistyczny

(założenie że modyfikowanie i odczytywanie tych samych danych przez różnych użytkowników jest mało prawdopodobne)

Przeprowadzanie transakcji odbywa się bez blokowania zasobów – jedynie w trakcie modyfikowania danych zasoby są sprawdzane w celu wykrycia konfliktów

Kontrola współbieżności

Model pesymistyczny

(założenie wystąpienia konfliktów)

Dane są blokowane za każdym razem, gdy jakaś transakcja próbuje je odczytać lub modyfikować

Kontrola współbieżności

Typy blokad

Blokady współdzielne S (Shared) są zakładane domyślnie na odczytywanych obiektach tylko na czas wykonania zapytania

Blokady wyłączne X (eXclusive) są zakładane na modyfikowanych obiektach i utrzymywane do zakończenia całej transakcji

Kontrola współbieżności

Zakres blokad – blokady zakładane mogą być na różnym poziomie szczegółowości: na poziomie wiersza , kluczy indeksów, tabel lub bazy

Zakleszczenie (Deadlock) – powstaje gdy jeden proces próbuje założyć blokadę powodującą konflikt z inną blokadą

Kontrola współbieżności

Problemy z izolowaniem transakcji:

- Utrata aktualizacji – Lost or buried updates (gdy dwie transakcje modyfikują te same dane)
- Brudne odczyty – Dirty reads (gdy jedna transakcja dokonuje zmian a druga w tym samym czasie odczytuje dane)

Kontrola współbieżności

Problemy z izolowaniem transakcji:

- Niepowtarzalne odczyty – Non repeatable reads (gdy powtórzenie w ramach transakcji tego samego odczytu daje inny wynik)
- Odczyty widma - Phantom reads (gdy pomiędzy dwoma odczytami tych samych danych w ramach jeden transakcji zmieni się liczba odczytywanych wierszy)

Kontrola współbieżności

Izolowanie transakcji – transakcja jest izolowana do momentu wydania polecenia COMMIT lub ROLLBACK

Kontrola współbieżności

Poziom izolacji transakcji określa stopień, do którego dana transakcja musi być izolowana od innych transakcji. Najniższy poziom izolacji to maksymalny stopień współbieżności, ale jest on kosztem najniższego stopnia spójności danych.

Najwyższy poziom izolacji gwarantuje najwyższy poziom spójności danych kosztem ograniczenia do minimum współbieżności

Kontrola współbieżności

Serwery bazodanowe pozwalają ustawić na poziomie serwera lub pojedynczej sesji jeden z czterech poziomów izolowania:

- Read Uncommitted
- Read Committed
- Repeatable Read
- Serializable

Kontrola współbieżności

- Read Uncommitted

Transakcja może zobaczyć modyfikacje rekordów wprowadzone przez inne transakcje, nawet jeśli jeszcze nie zostały zatwierdzone

Kontrola współbieżności

- Read Committed

Transakcja może zobaczyć modyfikacje rekordów wprowadzone przez inne transakcje, tylko jeśli zostały zatwierdzone.

Kontrola współbieżności

- Repeatable Read

Jeśli transakcja dwukrotnie wykonuje dane zapytanie SELECT wynik jest powtarzalny. Oznacza to, że za każdym razem otrzymuje ten sam wynik, nawet jeśli w międzyczasie inne transakcje zmieniły lub wstawiły rekordy.

Kontrola współbieżności

- Serializable

Poziom izolacji podobny do REPEATABLE READ, ale niemalże całkowicie izoluje transakcje. Rekordy używane przez jedną transakcję nie mogą być modyfikowane przez inne transakcje aż do chwili ukończenia pierwszej. W ten sposób, gdy jedna transakcja odczytuje rekordy, w tym samym czasie inne transakcje nie mogą ich modyfikować.

START TRANSACTION ISOLATION LEVEL poziom_izolacji

START TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

SET GLOBAL TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

SET SESSION TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

READ UNCOMMITTED

Najniższy poziom izolacji transakcji. Odczyt danych nie powoduje założenia blokady współdzielonej. Na tym poziomie pojawiają się brudne odczyty, niepowtarzalne odczyty i odczyty widma. Nie występuje jedynie problem z utratą aktualizacji. Ten tryb może być stosowany do odczytywania danych, o których wiemy że w czasie odczytywania nie będą modyfikowane.

Okno-1

START TRANSACTION;

UPDATE Klient SET ulica='Mickiewicza', nr_domu=94 WHERE nazwisko='Kawka' AND imie='Piotr';

Okno-2

SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;

SELECT miasto, ulica, nr_domu FROM Klient WHERE nazwisko='Kawka' AND imie='Piotr';

Okno-1

ROLLBACK;

UWAGA!!!Mimo że użytkownik pierwszy nie zamknął transakcji, udało się odczytać zmienione dane

READ COMMITTED

Domyślny poziom izolowania w MySQL. Podczas odczytu danych zostanie założona na nie blokada współdzielona. Założona blokada eliminuje brudne odczyty. Natomiast nadal występują niepowtarzalne odczyty oraz odczyty widma.

Okno-1
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
START TRANSACTION;
SELECT miasto, ulica, nr_domu FROM Klient WHERE nazwisko='Kawka' AND imie='Piotr';

Okno-2
UPDATE Klient SET ulica='Sienkiewicza', nr_domu=21 WHERE nazwisko='Kawka' AND imie='Piotr';

Okno-1
SELECT miasto, ulica, nr_domu FROM Klient WHERE nazwisko='Kawka' AND imie='Piotr';
COMMIT;

UWAGA!!! Adres ponownie odczytany w Oknie-1 jest inny niż odczytany wcześniej – efekt niepowtarzalnych odczytów

REPEATABLE READ

Tryb powtarzalnego odczytu. Założona na dane blokada współdzielna jest utrzymywana aż do zakończenia transakcji. Dzięki temu że inny proces nie może modyfikować odczytywanych danych, zostały wyeliminowane niepowtarzalne odczyty – występują nadal odczyty widma

Okno-1
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
START TRANSACTION;
SELECT nazwisko, imie FROM Klient WHERE miasto=‘Szczecin’;

Okno-2
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
UPDATE Klient SET miasto=‘Szczecin’ WHERE nazwisko=’Drozd’ AND imie=‘Joanna’;

Okno-1
SELECT nazwisko, imie FROM Klient WHERE miasto=‘Szczecin’;
COMMIT;

UWAGA!!! Liczba odczytanych wierszy uległa zmianie. Zmiana danych w Oknie-2 była możliwa ponieważ dotyczyła danych, które nie były odczytywane w transakcji otwartej w pierwszej sesji.

REPEATABLE READ

Tryb powtarzalnego odczytu. Założona na dane blokada współdzielna jest utrzymywana aż do zakończenia transakcji. Dzięki temu że inny proces nie może modyfikować odczytywanych danych, zostały wyeliminowane niepowtarzalne odczyty – występują nadal odczyty widma

Okno-1
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
START TRANSACTION;
SELECT nazwisko, imie FROM klient WHERE miasto='Lublin';

Okno-2
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
UPDATE Klient SET miasto='Warszawa' WHERE miasto='Lublin';

Okno-1
COMMIT;

UWAGA!!! W tym wypadku instrukcja będzie oczekiwać na zakończenie transakcji w pierwszej sesji i zdjęcie blokady z danych

SERIALIZABLE

Transakcje odwołujące się do tych samych danych są szeregowane, czyli wykonywana jedna po drugiej. Najwyższy poziom izolacji, gdzie transakcje są w pełni izolowane od siebie. Na czas transakcji izolowane są całe obiekty. Np. modyfikując jeden wiersz w tabeli blokujemy możliwość modyfikowania danych w całej tabeli.

Okno-1
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
START TRANSACTION;
SELECT miasto,ulica,nr_domu FROM Klient WHERE nazwisko= 'Wrona' AND imie='Ewa';

Okno-2
UPDATE Klient SET miasto='Katowice' WHERE nazwisko= 'Wrona' AND imie='Ewa';

Okno-1
COMMIT;

UWAGA!!! Transakcja w oknie-2 będzie mołziwa dopiero po zakończeniu transakcji z okna-1

WIDOKI - PERSPEKTYWY

- Perspektywy (widoki) to w SQL wirtualne tabele tworzone na podstawie zapytań. Składają się z kolumn i wierszy pobranych z prawdziwych tabel. Pokazywane w widoku dane są zawsze aktualne, ponieważ widoki są tworzone w momencie wykonania zapytania

WIDOKI - PERSPEKTYWY

- Perspektywy (widoki) nie przechowują zapisanych w tabelach danych. W bazie jest zapamiętywana tylko definicja widoku i związane z nią metadane.
- Widoki zapewniają bezpieczeństwo danych przez ograniczenie dostępu do danych zapisanych w tabelach.

TWORZENIE WIDOKU

```
CREATE VIEW vklient AS
SELECT nazwisko, imie, miasto
FROM klient;
```

TWORZENIE WIDOKU

```
CREATE VIEW vklient2 (lastname,firstname,city) AS
SELECT nazwisko, imie, miasto
FROM klient;
```

TWORZENIE WIDOKU

```
CREATE OR REPLACE VIEW vpac2 AS
SELECT nazwisko, imie, miasto
FROM pacjenci1
```

TWORZENIE WIDOKU

```
CREATE TEMP VIEW vpacjenci AS
SELECT nazwisko, imie, miasto
FROM pacjenci
```

TWORZENIE WIDOKU

```
CREATE OR REPLACE TEMP VIEW vpacjenci AS
SELECT nazwisko, imie, miasto
FROM pacjenci
```

MODYFIKOWANIE WIDOKU

```
ALTER  VIEW vklient AS
SELECT nazwisko, imie, miasto, ulica, Nr_domu
FROM klient WHERE id_klienta>5;
```

USUWANIE WIDOKU

DROP VIEW vklient

UWAGI

- W widokach nie można stosować klauzuli ORDER BY. Jedyny wyjątek to użycie klauzuli ORDER BY do określenia wierszy, które są zwracane przez klauzulę TOP

Łączenie wyników zapytań

- Instrukcja UNION

zapytanie1 UNION zapytanie2

zapytanie1 UNION ALL zapytanie2

Warunkiem wykonania instrukcji jest aby łączone zapytania miały taką samą liczbę kolumn (wskazanie - aby typy kolumn były takie same)

Łączenie wyników zapytań

- Przykład:

```
Select nazwisko,imie
FROM klient1
WHERE miasto=`Lublin`
UNION
Select nazwisko,imie
FROM klient1
WHERE miasto=`Horodyszcze`;
```

Łączenie wyników zapytań

- Przykład:

```
Select nazwisko,imie
FROM klient1
WHERE miasto=`Lublin`
UNION
Select Nr_domu,ulica
FROM klient2
WHERE miasto=`Horodyszcze`;
```

Łączenie wyników zapytań

- Przykład:

```
Select nazwisko,imie
FROM klient1
WHERE miasto=`Lublin`
UNION ALL
Select nazwisko,imie
FROM klient1
WHERE miasto=`Lublin`;
```

Łączenie wyników zapytań

- Przykład:

```
Select nazwisko,imie
FROM klient1
WHERE miasto=`Lublin`
UNION ALL
Select nazwisko,imie
FROM klient1
WHERE imie=`Alina`;
```

Zapytanie 1 INTERSECT Zapytanie2

- Instrukcja zwraca część wspólną obydwu łączonych zapytań

Łączenie wyników zapytań

- Przykład:

```
Select nazwisko,imie
FROM klient1
WHERE telefon IS NULL
INTERSECT
Select nazwisko,imie
FROM klient1
WHERE adres_e_mail IS NULL
```

Zapytanie1 EXCEPT Zapytanie2

- Zwraca te wiersze, które wystąpiły w wyniku pierwszego zapytania, ale nie było ich w wyniku drugiego zapytania

Łączenie wyników zapytań

- Przykład:

```
Select nazwisko,imie
FROM klient1
WHERE telefon IS NULL
INTERSECT
Select nazwisko,imie
FROM klient1
WHERE adres_e_mail IS NULL
```

Podzapytania

- Zapytanie definiowane przy użyciu instrukcji SELECT umieszczone wewnątrz innej instrukcji SELECT.
- Zagnieżdżenie można stosować w instrukcjach SELECT, INSERT, UPDATE i DELETE w klauzuli WHERE lub FROM

Podzapytania

```
SELECT kolumna1, kolumna2
FROM Tabela 1
WHERE kolumna1 =
(SELECT kolumnaA1
FROM TabelaA1
WHERE warunek_podzapytanie)
```

Podzapytania

```
SELECT kolumna1, kolumna2
FROM Tabela 1
WHERE kolumna1 IN
(SELECT kolumnaA1
FROM TabelaA1
WHERE warunek_podzapytanie)
```

Tabele pochodne

```
SELECT Li.kolumna1, Li.kolumna2
FROM
(SELECT kolumna1,kolumna2
FROM Tabela
WHERE warunek_podzapytanie) AS Li
WHERE warunek_zapytania;
```

Podzapytania

```
SELECT *
FROM Pacjenci1
WHERE SELECT miasto
FROM Pacjenci2
WHERE nazwisko=`Mikulska` Pacjenci1.miasto=
();
```

Podzapytania

```
SELECT *
FROM Pacjenci1
WHERE Pacjenci1.miasto =
(SELECT miasto
FROM Pacjenci2
WHERE nazwisko=`Kawka`);
```

Podzapytania

```
SELECT *
FROM Pacjenci1
WHERE Pacjenci1.miasto=
(SELECT miasto
FROM Pacjenci2
WHERE nazwisko=`Mikulska`);
```

Podzapytania

```
SELECT *
FROM Pacjenci1
WHERE Pacjenci1.miasto IN
(SELECT miasto
FROM Pacjenci2
WHERE nazwisko=`Mikulska`);
```

Tabele pochodne

```
SELECT Li.kolumna1, Li.kolumna2
FROM
(SELECT kolumna1,kolumna2
FROM Tabela
WHERE warunek_podzapytanie) AS Li
WHERE warunek_zapytania;
```

Tabele pochodne

```
SELECT zap1.nazwisko, zap1.imie
FROM
(SELECT nazwisko, imie,miasto
FROM pacjenci1
WHERE lekarz=3) AS Zap1
WHERE miasto='Lublin';
```

Operatory zapytań

- **IN** (czy chociaż jedna wartość znajduje się na liście)
- **EXISTS** (czy zapytanie wewnętrzne zwraca jakąkolwiek wartość)
- **ANY /SOME** (sprawdza wartość wybranego wiersza w wyniku zapytania)
- **ALL** (sprawdza wartość wszystkich wierszy w wyniku podzapytania)

EXISTS

```
SELECT pacjenci.nazwisko, pacjenci.imie
FROM pacjenci
WHERE EXISTS
(SELECT * FROM wizyta
WHERE wizyta.pacjent=pacjenci.nr_kartoteki);
```

EXISTS

```
SELECT pacjenci.nazwisko, pacjenci.imie
FROM pacjenci1
WHERE NOT EXISTS
(SELECT * FROM wizyta
WHERE wizyta.pacjent=pacjenci1.nr_kartoteki);
```

EXISTS

```
SELECT wizyta.pacjent
FROM wizyta AS z1
WHERE EXISTS
(SELECT *
FROM wizyta AS  z2
WHERE z1.pacjent=z2.pacjent);
```

ANY

```
SELECT pacjenci.nazwisko, pacjenci.imie
FROM pacjenci INNER JOIN wizyta
ON pacjenci.nr_kartoteki=wizyta.pacjent
WHERE wizyta.datawizyty = ANY
(SELECT wplaty.datawplaty
FROM wplaty);
```

```
SELECT wizyta.pacjent
FROM wizyta
WHERE wizyta.kwota < ALL
(SELECT wizyty
```

Podzapytania w instrukcjach modyfikujących dane

CREATE TABLE ARCHIWUM

(wizyta(30), imie varchar(30), datawizyty date);

Podzapytania w instrukcjach modyfikujących dane

INSERT INTO archiwum

(SELECT

Instrukcja INTERSECT

- Union all
- Order by
- Limit
- Delete t1 from t1 inner join t2 ont t1.id=t2.id
- Delete t1,t2 from t1 inner join t1.id=t2.id
- Select t1.* from t1 left join t2 on t1.id=t2.id where t2.id is null
- Delete

BEGIN …END

- Konstrukcja BEGIN…END tworzy blok pozwalający na grupowanie wielu zapytań. Jeżeli część główna programu składowanego zawiera więcej niż tylko jedno zapytanie, muszą być one grupowane w ramch bloku BEGIN.

CASE

CASE [wyr]

WHEN wyr1 THEN ….

WHEN wyr2 THEN …

ELSE ….

END IF

Po znalezieniu pierwszego dopasowania po WHEN wykonuje listę zapytań po THEN

IF

- IF warunek1 THEN lista_instrukcji1
- [elseif warunek2 THEN lista_instrukcji2]
- [else lista_instrukcji3]
- End if

FUNKCJE SKŁADOWANE

```
DELIMITER $
CREATE FUNCTION liczba1( p_lekarz INTEGER)
RETURNS INT
BEGIN
RETURN (SELECT count(*) FROM PACJENCI
   WHERE lekarz=p_lekarz);
END$
DELIMITER ;

Wywołanie SELECT liczba1(2);
```

PROCEDURY SKŁADOWANE

```
DELIMITER $
CREATE PROCEDURE lista( p_lekarz INTEGER)
BEGIN
SELECT * FROM PACJENCI WHERE
   lekarz=p_lekarz;
END$
DELIMITER ;
```

Wywołanie CALL lista(2);

PROCEDURY SKŁADOWANE

```
DELIMITER $
CREATE PROCEDURE nowylekarz(zid INTEGER, zimie
    varchar(20), znazwisko varchar(20))
BEGIN
INSERT INTO Lekarze (id_lekarza, imie, nazwisko)
    VALUES (zid, zimie, znazwisko);
END$
DELIMITER ;
```

Wywołanie CALL nowylekarz(5,’Albin’ ,’Dudzial’,);

PROCEDURY SKŁADOWANE

```
DELIMITER $
CREATE PROCEDURE new5( zkod int, zimie varchar(20), znazwisko
    varchar(20))
BEGIN
IF NOT EXISTS (SELECT * FROM lekarze WHERE imie=zimie AND
    nazwisko=znazwisko) THEN
INSERT INTO Lekarze (id_lekarza,imie,nazwisko) VALUES
    (zkod,zimie,znazwisko);
Else
Select  'już jest';
End if;
END$
DELIMITER ;
```

Wywołanie CALL new5(6,'Albin', 'Dudzial');

Wartość zmiennych użytkownika mogą być przypisywane za pomocą operatorów = lub := w zapytaniach SET lub operatora := w pozostałych zapytaniach

```
Set @x =0, @y=2;
Set @color:='red';
Select @x, @color;
```

Typy parametrów procedury składowanej

- IN (domyślny) – przekazanie wartości tylko do procedury
- OUT – przekazanie wartości dla wywołującego po zakończeniu działania procedury
- INOUT – pozwala wywołującemu na przekazanie i otrzymanie wartości

```
DELIMITER $
CREATE PROCEDURE liczp6 (OUT p_lekarz INT, OUT
   p_pac INT )
BEGIN
SET p_pac= (SELECT count(*) FROM pacjenci);
SET p_lekarz = (SELECT count(*) FROM Lekarze);
END$
DELIMITER ;

CALL liczp6(@p_lekarz, @p_pac);
SELECT @p_lekarz, @p_pac;
```

```
DELIMITER $
CREATE PROCEDURE liczp1 ()
BEGIN
SET @p_pac= (SELECT count(*) FROM pacjenci);
SET @p_lekarz = (SELECT count(*) FROM Lekarze);
END$
DELIMITER ;

CALL liczp1();
SELECT @p_lekarz, @p_pac;
```

Wyzwalacze - Triggers

Wyzwalacz to program składowany powiązany z określoną tabelą i zdefiniowany do uaktywnienia się podczas wykonywania zapytania INSERT, DELETE lub UPDATE dal tej tabeli.

Wyzwalacz może być aktywowany przed przetworzeniem lub po przetworzeniu każdego rekordu przez zapytanie

Wyzwalacze - Triggers

CREATE TRIGGER nazwa_wyzwalacza - *nazwa*

[BEFORE|AFTER] - *moment aktywacji*

[INSERT|UPDATE|DELETE] - *typ zapytania*

ON nazwa_tabeli - *tabela powiązana z wyzwalaczem*

FOR EACH ROW zapytanie_wyzawalacza – *operacje wykonywane przez wyzwalacz*

Wyzwalacze - Triggers

W kodzie wyzwalacza składni

NEW.nazwa_kolumny można użyć w celu odwołania się do kolumn w nowym rekordzie przeznaczonym do wstawienia lub uaktualnienia w wyzwalaczu INSERT lub UPDATE

Wyzwalacze - Triggers

W kodzie wyzwalacza składni

OLD.nazwa_kolumny można użyć w celu odwołania się do kolumn w starym rekordzie przeznaczonym do usunięcia lub uaktualnienia w wyzwalaczu DELETE lub UPDATE

Wyzwalacze - Triggers

Aby zmienić wartość kolumny w wyzwalaczu zanim ta wartość będzie umieszczona w tabeli należy użyć zapytania

SET New.nazwa_kolumny = wartość

```
CREATE TABLE  wskaznik(id int primary key auto_increment,
    procent INT, datazapisu DATETIME);


DELIMITER $
CREATE TRIGGER spr_dane BEFORE INSERT ON wskaznik
FOR EACH ROW
BEGIN
IF NEW.procent < 0 THEN
    SET NEW.procent=0;
ELSEIF NEW.procent > 100 THEN
    SET NEW.procent =100;
END IF;
SET NEW.datazapisu = CURRENT_TIMESTAMP;
END$
DELIMITER  ;
```

```
INSERT INTO tabelka(procent) VALUES (52);
SELECT * FROM tabelka;
INSERT INTO tabelka(procent) VALUES (130);
SELECT * FROM tabelka;
INSERT INTO tabrlka(procent) VALUES (-20);
SELECT * FROM tabelka;
```

Optymalizacja zapytań

- Utworzenie indeksów w tabelach, aby pozwolić serwerowi bazy danych na szybsze wyszukiwanie indeksów
- Zmiana sposobu tworzenia zapytań, aby w maksymalny stopniu wykorzystać indeksy (w celu sprawdzenia zapytanie EXPLAIN)
- Zmiana typów danych

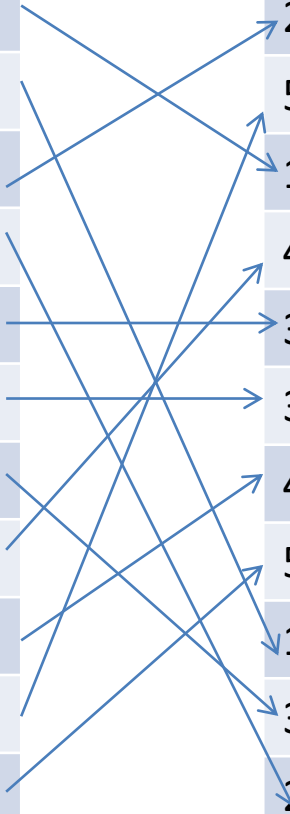
Zalety wynikające z indeksu

- Efektywność indeksu polega na możliwości określenia, kiedy nie dopasujemy większej liczby rekordów, co pozwala na pominięcie analizy pozostałych

(przykład)

Zalety wynikające z indeksu

Id_firmy
1
1
2
2
3
3
3
4
4
5
5



Id_firmy	Faktura	Kwota
2	11	50,00
5	12	120,00
1	13	60,00
4	14	90,00
3	15	200,00
3	16	30,00
4	17	80,00
5	18	10,00
1	19	160.00
3	20	20.00
2	21	50,00

Wybór indeksu

```
SELECT
        tbl1.kol_a
FROM
        tbl1 LEFT JOIN tbl2
        ON tbl1.kol_b=tbl2.kol_c
WHERE
         tbl1.kol_d=wyrażenie;
```

kol_a – zły kandydat

kol_b, kol_c - dobry koandydat

Kol_d – dobry kandydat

Optymalizator zapytań – jak działa

```
SELECT
        kol3
FROM
   mojatabela
WHERE
   kol1=wartość1 AND kol2=wartość2
```

Optymalizator zapytań – jak działa

```
SELECT
        kol3
FROM
   mojatabela
WHERE
   kol1=wartość1 AND kol2=wartość2
```

Nie używaj znaków ieloznacznych

```
WHERE TO_DAYS (data1) – TO_DAYS(CURDATE()) <  granica

WHERE TO_DAYS (data1) <  granica + TO_DAYS(CURDATE())

WHERE  data1 < DATE_ADD ( CURDATE(), INTERVAL granica DAY)
```

EXPLAIN SELECT * FROM PACJENCI1

Id:	1
Select_type:	SIMPLE
Table:	Member
Type:	ALL
Possible_keys:	**NULL**
Key:	NULL
Key_len:	NULL
Ref:	NULL
Rows:	**102**

ZDARZENIA

show variables like "event_scheduler";

SET GLOBAL event_scheduler = ON

SET GLOBAL event_scheduler = OFF

ZDARZENIA

CREATE EVENT nazwa_zdarzenia

ON SCHEDULE

(AT datetime | EVERY EXPR okres_czasu [STARTS datetime] [ENDS datetime])

DO

zapytanie

ZDARZENIA

```
CREATE EVENT mojezdarzenie
ON SCHEDULE AT '2015-02-27 09:49:00‘
DO
DELETE FROM wizyta WHERE lekarz=2;
```

```
CREATE EVENT mojezdarzenie
ON SCHEDULE EVERY 30 SECOND
DO
INESRT INTO wizyta(id,pacjent,lekarz)
   VALUES(null,1,1);
```

Zarządzanie kontem MySQL

Trzy polecenia operacji na kontach MySQL

1. Zapytanie CREATE USER tworzy nowe konto użytkownika i opcjonalnie przypisuje mu hasło lub definiuje metodę uwierzytelniania

CREATE USER użytkownik [typ uwierzytelniania]

Zapytanie CREATE USER nie nadaje żadnych uprawnień

Zarządzanie kontem MySQL

Trzy polecenia operacji na kontach MySQL

2. Zapytanie DROP USER powoduje usunięcie istniejącego konta i wszystkich związanych z nim uprawnień

DROP USER użytkownik

Zarządzanie kontem MySQL

Trzy polecenia operacji na kontach MySQL

3. Zapytanie RENAME USER powoduje zmianę nazwy istniejącego konta

RENAME USER konto_źródłowe TO konto_docelowe

Zarządzanie kontem MySQL

Zakres CREATE USER:

- Nazwa użytkownika
- Komputer/y z jakich użytkownik będzie mógł nawiązać połączenie
- Metoda uwierzytelnienia

Zarządzanie kontem MySQL

Nazwa użytkownika:

‘nazwa_użytkownika’@’nazwa_komputera’

CREATE USER ‘piotr’@’localhost’ IDENTIFIED BY ‘luty’;

CREATE USER ‘pawel’@’komp1.zst.pl’ IDENTIFIED BY ‘luty’;

Przypomnienie

W zapytaniu LIKE znaki specjalne

% - dowolny ciąg

_ - jeden dowolny znak

```
CREATE USER ‘maria’@’%.zst.com’ IDENTIFIED
    BY ‘luty’;
CREATE USER ‘jan’@’192.168.128.3’ IDENTIFIED
    BY ‘luty’;
CREATE USER ‘jan’@’192.168.128.%’ IDENTIFIED
    BY ‘luty’;
CREATE USER
    ‘jan’@’192.168.128.0/255.255.255.0’
    IDENTIFIED BY ‘luty’;
```

Sposoby uwierzytelniania użytkownika

CREATE USER użytkownik [typ uwierzytelniania]

1. Identyfikacja za pomocą składni (hasło w postaci wartości hash)

IDENTIFIED BY [PASSWORD] ‘haslo’

2. Inna metoda uwierzytelniania (podaj nazwę wtyczki)

IDENTIFIED WITH wtyczka uwierzytelniania

UPRAWNIENIE

Uprawnienia można zdefiniować za pomocą zapytań:

- Zapytanie GRANT pozwala na zdefiniowanie uprawnień konta
- Zapytanie REVOKE pozwala na odebranie uprawnień

UPRAWNIENIE

Uprawnienia można zdefiniować za pomocą zapytań:

- Zapytanie SET PASSWORD pozwala na zdefiniowanie hasła dla konta MySQL
- Zapytanie SHOW GRANTS powoduje wyświetlenie uprawnień zdefiniowanych dla wskazanego konta.

Tabela uprawnień MySQL

Tabela uprawnień	Zawartość tabeli
user	Dane użytkowników, którzy mogą nawiązać połączenie z serwerem i ich globalne uprawnienia
db	Uprawnienia baz danych
tables_priv	Uprawnienia tabel
columns_priv	Uprawnienia kolumn
procs_priv	Uprawnienia procedur składowanych
proxies_priv	Uprawnienia użytkowników proxy

Nadawanie uprawnień

GRANT uprawnienia [(kolumny)]

ON obiekt

TO użytkownik [typ uwierzytelniania]

[REQUIRE wymagane szyfrowanie]

[WITH nadawanie uprawnień lub opcje zarządzania zasobami]

UWAGA !!! Jeżeli konto użytkownika nie istnieje to zapytanie GRANT utworzy je.

```
GRANT prawo [kolumny]
ON obiekt
TO użytkownik [IDENTIFIED BY hasło]
[WITH [GRANT OPTION |
MAX_QUERIES_PER_HOUR ile|
MAX_UPDATES_PER_HOUR ile |
MAX_CONNECTIONS_PER_HOUR ile|
MAX_USER_CONNECTIONS ile]
]
```

Parametr Obiekt

- Poziom globalny (global level) *.*
- Poziom bazy danych (database level) nazwa_bazy.*
- Poziom tabeli (table level) – nazwa_bazy.nazwa_tabeli
- Poziom kolumny (column level) nazwa_bazy.nazwa_tabeli oraz parametr kolumny

Parametr Prawa

- ALL - ustawia wszystkie uprawnienia
- ALTER - użycie ALTER TABLE
- ALTER ROUTINE – pozwala na zmianę oraz usuwanie procedur składowanych
- CREATE - CREATE TABLE oraz CREATE DATABASE
- CREATE TABLESPACE – pozwala tworzenie, modyfikację i usuwanie przestrzeni tabel
- CREATE TEMPORARY TABLES

Parametr Prawa

- CREATE USER - pozwala użyć zapytań CREATE USER, DROP USER, RENAME USER
- CREATE VIEW – tworzenie perspektyw
- DELETE – usuwanie wierszy z tabeli
- DROP – użycie DROP TABLE
- EVENT – zdarzenia
- EXECUTE – wywołanie procedur składowanych
- FILE – umożliwia zapis i odczyt danych zewnętrznych

Parametr Prawa

- GRANT OPTION – umożliwia nadawanie i odbieranie uprawnień dla innych kont
- INDEX – zezwala na CREATE INDEX oraz DROP INDEX
- INSERT
- SELECT
- SHOW DATABASES
- SHOW VIEW
- SHUTDOWN – pozwala na kończenie pracy serwera
- TRIGGER

Parametr Prawa

- UPDATE
- USAGE – pozwala jedynie na dostęp do bazy bez innych uprawnień

Opcje

WITH GRANT OPTION – przepisanie uprawnień innym użytkownikom

MAX_QUERIES_PER_HOUR – ile zapytań przez godzinę może wykonać użytkwnik (0 – brak limitu)

MAX_UPDATES_PER_HOUR - ile uaktualnień

MAX_CONNECTIONS_PER_HOUR – ile połączeń przez godzinę

MAX_USER_CONNECTIONS – ile równoczesnych połączeń

INFORMACJE o uprawnieniach wybranego użytkownika

- SHOW GRANTS FOR ‘piotr’@’localhost’ ;

- Sprawdzenie uprawnień dla aktualnie zalogowanego użytkownika

SHOW GRANTS;

SHOW GRANTS FOR CURRENT_USER;

SHOW GRANTS FOR CURRENT_USER();

- Nadaj użytkownikowi posługującemu się nazwą piotr prawa do logowania się do serwera

Podstawowe konto użytkownika

- GRANT usage ON *.* TO ‘piotr’@’localhost’ ;

- Użyj polecenia GRANT do zmiany hasła użytkownikowi piotr (nowe hasło ‘marzec’)

Podstawowe konto użytkownika

- GRANT usage ON * TO ‘piotr’@’localhost’ IDENTIFIED BY ‘marzec’

- Nadaj użytkownikowi piotr uprawnienia do tworzenie i modyfikacji tabel w bazie test1

Podstawowe konto użytkownika

- GRANT CREATE, SELECT, INSERT, UPDATE, DELETE
  ON test1.*
  TO ‘piotr’@’localhost’ ;

Nadaj uprawnienia wykonywania zapytań typu SELECT i INSERT dla użytkownika pawel w bazie test1

Podstawowe konto użytkownika

- GRANT SELECT, INSERT
  ON test1.*
  TO ‘pawel’@’localhost’ ;

Podstawowe konto użytkownika

Utwórz konto użytkownika maria z dowolnym hasłem. Nadaj mu uprawnienia do wykonywania zapytań SELECT w bazie test1. Ogranicz maksymalną liczbę zapytań do 50 na godzinę, a maksymalną liczbę połączeń do 5 na godzinę.

- GRANT SELECT
  ON test1.*
  TO ‘maria’@’localhost’ IDENTIFIED BY ‘haslo’
  WITH
    MAX_QUERIES_HOUR 50
    MAX_CONNECTIONS_PER_HOUR 5;

PROXY – użytkownik piotr będzie posiadał uprawnienia użytkownika ela

GRANT PROXY ON ‘ela’@’localhost’ TO ‘piotr’@’localhost’

ALTER ROUTINE i EXECUTE

- GRANT CREATE ROUTINE ON baza.* TO ‘piotr’@’localhost’

- GRANT EXECUTE ON PROCEDURE baza.pacjencilekarza TO ‘piotr’@’localhost’

usage

- W celu zmiany hasła , zdefiniowanie wymogu nawiązywania połączenia SSL lub nałożenia ograniczeń na konto bez wpływu na jego dotychczasowe uprawnienia należy
- GRANT USAGE *.* TO uzytkownik IDENTIFIED BY ‘nowe haslo’

Nadanie użytkownikowi uprawnień administracyjnych

- GRANT ALL ON baza.* to ‘piotr’@’localhost’ WITH GRANT OPTION

- GRANT GRANT OPTION ON baza.* to ‘piotr’@’localhost’

- REVOKE uprawnienia [(kolumny)] ON poziom uprawnień FROM użytkownik

- GRANT ALL ON baza.* TO ‘piotr’@’localhost’

- REVOKE DELETE,UPDATE ON baza.* FROM ‘piotr’@’localhost’
- REVOKE GRANT OPTiON ON baza.* FROM ‘piotr’@’localhost’

Kontrola dostępu klientów

ETAP 1

1. Kontrola godzinnego limitu i liczby maksymalnych połączeń
2. Gdy user wskazuje na konieczność bezpiecznego połączenia serwer ustala, czy dostarczone przez klienta dane uwierzytelniania są prawidłowe

Kontrola dostępu klientów

ETAP 2 (w każdym zapytaniu)

1. Kontrola wartości licznika wskazującego liczbę wykonanych zapytań w trakcie godziny i licznika zapytań aktualizacyjnych
2. Serwer sprawdza tabele uprawnieńw celu potwierdzenia wystarczających uprawnień klienta

Tabela uprawnień MySQL

Tabela uprawnień	Zawartość tabeli
user	Dane użytkowników, którzy mogą nawiązać połączenie z serwerem i ich globalne uprawnienia
db	Uprawnienia baz danych
tables_priv	Uprawnienia tabel
columns_priv	Uprawnienia kolumn
procs_priv	Uprawnienia procedur składowanych
proxies_priv	Uprawnienia użytkowników proxy

- SET PASSWORD FOR ‘piotr’@’localhost’ = PASSWORD(‘dsd’)

- GRANT USAGE ON *.* TO ‘piotr’@’localhost’ IDENTIFIED BY ‘dsd’;

Kategorie kopii zapasowych:

Kopie zapasowe w formacie tekstowym za pomocą narzędzia mysqldump – tworzone są pliki które zawierają zapytania SQL Create Table i Insert

Kopie binarne zapasowe tworzone są przez bezpośrednie kopiowanie plików zawierających tabele

Przenoszenie baz

- Przenośność za pomocą narzędzia mysldump
- Przenośność binarna

Przenośność binarna

- Tabele InnoDB i MyISAM – uwaga na wielkość liter w nazwach baz i tabel
- Tabele CSV – przenośność pełna gdyż to pliki tekstowe CSV
- Tabele Memory nie są przenoszone binarnie gdyż ich zawartość jest przechowywana w pamięci a nie na dysku

Narzędzie mysqldump

- Kopia zapasowa pojedynczej tabeli

mysqldump baza tabela > pliktabela.sql

Odtworzenie tabeli

mysql baza < pliktabela.sql

Narzędzie mysqldump

- Kopia zapasowa wszystkich baz

mysqldump --all-databases > /archive/dump-all-2015-03-19

Kopia zapasowa jednej bazy

mysqldump baza > /archive/dump-baza-2015-03-19

Parametry polecenia mysqldump

- --databases – kilka baz
- --all-databases – wszystkie bazy
- --no-create-info lub --no-data (tylko dane lub tylko struktura)
- --opt lub --skip-opt (optymalizacja lub brak /domyślnie zawsze z optymalizacją/)
- -- routines, --triggers, --events lub –skip-events (dołączenie do kopii procedur składowanych, wyzwalaczy i zdarzeń /domyślnie wyzwalacze są dołączane/)
- --single-transaction (kopia będzie przeprowadzona w ramach jednej transakcji – dzięki czemu uzyskasz spójną kopię zapasową)

1. Utworzenie kopii zapasowej

mysqldump --databases baza > plikbazy.sql

(opcja --databases powoduje umieszczenie zapytania CREATE DATABASE IF NOT EXISTS AND USE)

2. Skopiowanie plików do zdalnego serwera

3. Zalogowanie na zdalnym serwerze i zaczytanie plików kopii zapasowej

mysql –u root –p1234 bazadanych < plikbazy.sql

Nałożenie blokady pozwalającej tylko na odczyt tabeli

- FLUSH TABLE tabela
- LOCK TABLE tabela READ

copy nazwa_tabeli.* /backup

- UNLOCK TABLE

UWAGA !!! Nakładanie blokad na poszczególne tabele mają zastosowanie tylko dla silników bazy danych MyISAM

Nałożenie blokady pozwalającej na odczyt i zapis tabeli

- LOCK TABLE tabela WRITE
- FLUSH TABLE tabela

copy /backup/nazwa_tabeli.* /data

- FLUSH TABLE tabela
- UNLOCK TABLE

Nałożenie blokady na wszystkie tabele pozwalającej jedynie na ich odczyt

```
FLUSH TABLES WITH READ LOCK;
SET GLOBAL read_only=ON;

operacje

SET GLOBAL read_only=OFF;
UNLOCK TABLES;
```

- FLUSH TABLES WITH READ LOCK – nakłada globalną blokadę odczytu
- SET GLOBAL read_only=ON – wstrzymuje blokadę aż wszyscy klienci zwolnią nałożone blokady

Sprawdzanie i naprawianie tabel

Aby naprawić tabelę InnoDB, w której wykryto problemy:

1. Wykonać kopię zapasową
2. Usunąć tabelę
3. Ponownie ją utworzyć za pomocą wcześniej przygotowanej kopii zapasowej

Sprawdzanie i naprawianie tabel

mysqlcheck baza tabela

mysqldump baza tabela > plikkopii.sql

Mysql baza < plikkopii.sql

Sprawdzanie i naprawianie tabel

Aby naprawić tabelę MyISAM, w której wykryto problemy:

1. Wykonać zapytania CHECK TABLE i REPAIR TABLE lub użyć narzędzia mysqlcheck, które nawiązuje połączenie z serwerm i wykona wymienione zapytania
2. Użyć narzędzia myisamchk, które działa bezpośrednio na plikach tabeli

Zapytanie CHECK TABLE

- CHECK TABLE działa z tabelami InnoDB, MyISAM, ARCHIVE, CSV, a także z widokami

CHECK TABLE tabela1, tabela2 ***OPCJE***

Zapytanie CHECK TABLE

- EXTENDED przeprowadza rozszerzone sprawdzanie i próbuje zapewnić maksymalną spójność tabeli
- FAST sprawdza tabelę tylko wtedy, gdy nie została poprawnie zamknięta
- MEDIUM sprawdza indeks, skanuje rekordy, sprawdza sumę kontrolną (op.domyślna)
- QUICK – skanowanie jedynie indeksów, rekordy danych są pomijane
- FOR UPGRADE – ustala czy sprawdzana tabela jest zgodna z bieżącą wersją MySQL

Zapytanie REPAIR TABLE

- REPAIR TABLE działa z tabelami MyISAM, ARCHIVE, CSV

REPAIR TABLE tabela1,tabela2 ***OPCJE***

Zapytanie REPAIR TABLE

- EXTENDED przeprowadza rozszerzone naprawę, obejmuje ponowne utworzenie indeksów
- QUICK powoduje szybką naprawę jedynie indeksów, rekordy danych są pomijane
- USE_FRM używa pliku .frm tabeli w celu ponownej inicjalizacji pliku indeksu (opcja użyteczna w przypadku utraty indeksów – Powinna być używana tylko w wersji MySQL za jaką tabela była używana)

Narzędzie mysqlcheck

- mysqlcheck do dostępne z poziomu wiersza poleceń narzędzie do zapytań CHECK TABLE i REPAIR TABLE
- mysqlcheck baza
- mysqlcheck baza tabela1 tabela2
- mysqlcheck --databases baza1 baza2
- mysqlcheck --all-databases

Opcja narzędzia mysqlcheck i odpowiadające im opcje zapytania REPAIR TABLE

Opcje mysqlcheck	Opcje REPAIR TABLE
- -repair	----------
- -repair - -extended	EXTENDED
- -repair - -quick	QUICK
- -repair - -use-frm	USE_FRM

Opcja narzędzia mysqlcheck i odpowiadające im opcje zapytania CHECK TABLE

Opcje mysqlcheck	Opcje CHECK TABLE
- - extended	EXTENDED
- - fast	FAST
- - medium-check	MEDIUM
- - quick	QUICK

Dzienniki zdarzeń serwera

- Dziennik błędów – rejestruje zdarzenia związane z uruchamianiem i zamykaniem serwera, a także informacje o problemach i sytuacjach wyjątkowych
- Ogólny dziennik zapytań – rejestrowane są połączenia z klientami, zapytania SQL otrzymane od klientów

Dzienniki zdarzeń serwera

- Dziennik wolno wykonanych zapytań – pomaga w wykrywaniu zapytań wymagających zmodyfikowania w celu uzyskania większej wydajności (domyślnie rejestruje zapytania trwające dłużej niż 10s)

Dzienniki zdarzeń serwera

- Binarny dziennik zdarzeń – zawiera w postaci zakodowanej w formacie binarnej zapytania typu: UPDATE, DELETE, INSERT itp.
- Plik indeksu binarnego dziennika zdarzeń – zawiera listę aktualnie wykorzystywanych binarnych dzienników zdarzeń

Dzienniki zdarzeń serwera

- Dziennik przekazywania – (jeżeli serwer jest serwerem replikacji) zawiera otrzymane od serwera głównego i przeznaczone do wykonania rekordy zdarzeń modyfikujących dane
- Plik indeksu dziennika przekazywania – zawiera pliki dziennika przekazywania

Dzienniki zdarzeń serwera

Opcja	Opis
- -general_log	Włącza ogólny dziennik zdarzeń
- -general_log_file=nazwa_pliku	Nazwa pliku ogólnego dziennika zdarzeń
- -log-bin [=nazwa_pliku]	Włączenie binarnego dziennika zdarzeń
- -log-bin-index =nazwa_pliku	Plik indeksu binarnego dziennika zdarzeń
- -log_error [=nazwa_pliku]	Włączenie dziennika błędów
- -relay-log [=nazwa_pliku]	Włączenie dziennika zdarzeń przekazywania
- -relay-log-index[=nazwa_pliku]	Plik indeksu dziennika zdarzeń przekazywania
- -slow_query_log	Włączenie dziennika wolno wykonywanych zapytań
--slow_query_log_file=nazwa pliku	Nazwa pliku dziennika wolno wykonywanych zapytań

Dzienniki zdarzeń serwera

- Przykład

[mysqld]

log_error = log.err

general_log

general_log_file=glog

Dziennik błędów

- Domyślnie plik hostname.err
- Położenie pliku – katalog danych MySQL

Ogólny dziennik zapytań

- Włączenie dziennika przez zmienną systemową general_log
- Domyślnie plik hostname.log
- Położenie domyślne pliku – katalog danych MySQL
- Nazwę pliku wskazuje zmienna general_log_file

Ogólny dziennik zapytań

- Możliwość zapisywania do pliku i/lub w tabeli

(tabela general_log w bazie mysql)

- SET GLOBAL log_output=‘FILE,TABLE’;
- SET GLOBAL log_output=‘NONE’;

Dziennik wolno wykonywanych zapytań

- Włączenie dziennika przez zmienną systemową slow_query_log
- Domyślnie plik hostname-slow.log
- Położenie domyślne pliku – katalog danych MySQL
- Nazwę pliku wskazuje zmienna slow_query_log_file
- Zmienna long_query_time wskazuje czas dla jakiego są rejestrowane zapytania

Dziennik wolno wykonywanych zapytań

- Możliwość zapisywania do pliku i/lub w tabeli

(tabela slow_log w bazie mysql)

- SET GLOBAL log_output='FILE,TABLE';
- SET GLOBAL log_output='NONE';

Binarny dziennik zdarzeń

- Włączenie dziennika przez opcję --log-bin
- Pliku generowane są w sekwencji nazwy gdzie nazwa bazowa to hostname-bin (pliki hostname-bin.000001, hostname-bin.000002)
- Kolejny plik jest generowany po uruchomieniu serwera lub po osiągnięciu maksymalnej wielkości – zmienna max_binlog_size

Binarny dziennik zdarzeń

- Zawiera wskaźnik czasu wskazujący moment wykonania danego zapytania
- Dane w dzienniku są zapisywane po zakończeniu wykonywania zapytania, a nie po jego otrzymaniu

Binarny dziennik zdarzeń

mysqlbinlog nazwa-pliku > plik_tekstowy

mysql < plik_tekstowy

Lub

mysqlbinlog nazwa-pliku. | mysql

Binarny dziennik zdarzeń

W celu wyodrębnienia zapytań dotyczących jedynie wskazanej bazy danych

mysqlbinlog - -database=nazwa_bazy nazwa-pliku > plik_tekstowy

mysql < plik_tekstowy

Binarny dziennik zdarzeń

W celu wyodrębnienia zapytań wykonanych po dacie

mysqlbinlog - - start-datetime =‘’2012-10-30 10:25:25” nazwa-pliku > plik_tekstowy

mysql < plik_tekstowy

Metody zarządzania dziennikami

- Rotacja dzienników zdarzeń – do dzienników o stałych nazwach np.: dziennik błędów, ogolny dziennik zapytań, dziennik wolno wykonywanych zapytań
- Utrata ważności na podstawie wieku – powoduje usunięcie plików dzienników zdarzeń starczych niż pewny określony wiek – dla numerowanych dzienników jak binarny dziennik

Metody zarządzania dziennikami

- Opróżnianie tabeli dzienników zdarzeń (rotacja) – jeżeli zdarzenia są zapisywane w tabelach bazy danych mysql istnieje możliwość ich opróżniania lub zmiany nazwy i zastępowania pustymi tabelami

Podstawy zabezpieczeń ACCESS

Komponenty mechanizmu zabezpieczeń JET:

- Grupy robocze
- Grupy
- Użytkownicy
- Właściciele obiektów
- Uprawnienia do obiektów

Podstawy zabezpieczeń ACCESS

Cel stosowania zabezpieczeń na poziomie użytkowników:

- Zabezpieczenie wrażliwych danych zapisanych w bazie danych
- Uniemożliwienie użytkownikom przypadkowego uszkodzenia apliakcji przez modyfikację obiektów

Podstawy zabezpieczeń ACCESS

W systemie JET informacje dotyczące zabezpieczeń bazy danych są zapisywane w plikach informacyjnych grup roboczych.

Domyślny plik ma nazwę SYSTEM.MDW znajdujący się w katalogu:

C:\Document and Setting\nazwa użytkownika\Application Data\Microsoft\Access\System.MDW

Zestawienie uprawnień

Uprawnienie:

- Otwórz/uruchom
- Otwórz z wyłącznością
- Czytaj projekt
- Modyfikuj projekt
- Administruj
- Odczytaj dane
- Aktualizuj dane
- Wstaw dane
- Usuń dane

Zestawienie uprawnień

Uprawnienie: Otwórz/uruchom

- Uprawnienia: Otwieranie bazy danych, formularza, raportu lub uruchomienie makra
- Dotyczy: Bazy danych, formularze, raporty i makra

Zestawienie uprawnień

Uprawnienie: Otwórz z wyłącznością

- Uprawnienia: Otwieranie bazy danych w trybie wyłączności
- Dotyczy: Tylko bazy danych

Zestawienie uprawnień

Uprawnienie: Czytaj projekt

- Uprawnienia: Przeglądanie obiektów w widoku projekt
- Dotyczy: tabele, kwerendy, formularze i makra

Zestawienie uprawnień

Uprawnienie: Modyfikuj projekt

- Uprawnienia: Przeglądanie i modyfikacja projektu obiektów lub ich usuwania
- Dotyczy: Tabele, kwerendy, formularze i makra

Zestawienie uprawnień

Uprawnienie: Administruj

- Uprawnienia: W przypadku bazy ustawianie hasła, replikacji i modyfikacja właściwości startowych, w przypadku obiektów bazy danych – dostęp do obiektów i danych w pełnym zakresie
- Dotyczy: Bazy danych, tabele, formularze, raporty i makra

Zestawienie uprawnień

Uprawnienie: Odczytaj dane

- Uprawnienia: Przeglądanie danych
- Dotyczy: Tabele i kwerendy

Zestawienie uprawnień

Uprawnienie: Aktualizuj dane

- Uprawnienia: Przeglądanie i modyfikowanie danych bez możliwości usuwania danych
- Dotyczy: Tabele i kwerendy

Zestawienie uprawnień

Uprawnienie: Wstaw dane

- Uprawnienia: Przeglądanie i wprowadzanie, ale bez możliwości ich modyfikowania lub usuwania
- Dotyczy: Tabele i kwerendy

Zestawienie uprawnień

Uprawnienie: Usuń dane

- Uprawnienia: Przeglądanie i usuwanie, ale bez możliwości ich modyfikowania lub wprowadzania
- Dotyczy: Tabele i kwerendy

Zabezpieczenie bazy w celu dystrybucji

- Utwórz grupę roboczą z którą chcesz rozprowadzać bazę danych
- Usuń użytkownika Administrator z grupy Administratorzy
- Usuń wszystkie uprawnienia grupy Użytkownicy
- Odbierz użytkownikowi Administrator wszystkie uprawnienia projektowania wobec wszystkich obiektów w bazie danych
- Nie przypisuj hasła użytkownikowi Administrator

Replikacja

- Replikacja oznacza proces kopiowania i udostępniania informacji pomiędzy kopiami tej samej bazy danych w taki sposób, że dane i struktura wszystkich kopii pozostają jednakowe.

Replikacja

- Wzorzec projektowania (Design Master)- w schemacie replikacji Microsoft Access oryginalna baz danych
- Replika - każda kopia wzorca projektowania

Zalet u wady replikacji

- Upraszcza współużytkowanie danych przez użytkowników mobilnych
- Ułatwia udostępnianie bazy danych odległym placówkom
- Zapobiega przeciążeniu bazy danych
- Rozprowadza aktualizacje aplikacji
- Służy jako kopia zapasowa danych

Tworzenie zestaw replik

- Przygotowanie bazy danych do replikacji: konwersja istniejącej bazy danych na wzorzec projektowania
- Utworzenie replik: skopiowanie wzorca projektowania na szereg replik
- Wprowadzanie zmian w danych: rozprowadzenie zestawu replik do użytkowników i zezwolenie im na wprowadzanie zmian w danych
- Synchronizacja: okresowe synchronizowanie replik pomiędzy sobą. Podczas synchronizacji zostaną również skopiowane do wszystkich replik zmiany struktury bazy danych, wprowadzone we wzorcu projektowania
- Rozwiązywanie konfliktów i błędów replikacji

Replika częściowa

- Replikowaniu podlega tylko część rekordów w zestawie replik

Typy replik częściowych

- Globalna – zmiany w globalnej replice częściowej są w pełni rejestrowane i dostępne do synchronizacji; może wymieniać dane z replikami lokalnymi i anonimowymi, pod warunkiem że replika globalna jest „centrum” zestawu replik

Typy replik częściowych

- Globalna – zmiany w globalnej replice częściowej są w pełni rejestrowane i dostępne do synchronizacji; może wymieniać dane z replikami lokalnymi i anonimowymi, pod warunkiem że replika globalna jest „centrum” zestawu replik

Typy replik częściowych

- Lokalna – może wymieniać dane z repliką globalną, służącą jako koncentrator, lecz nie może wymieniać z dowolnym członkiem zestawy

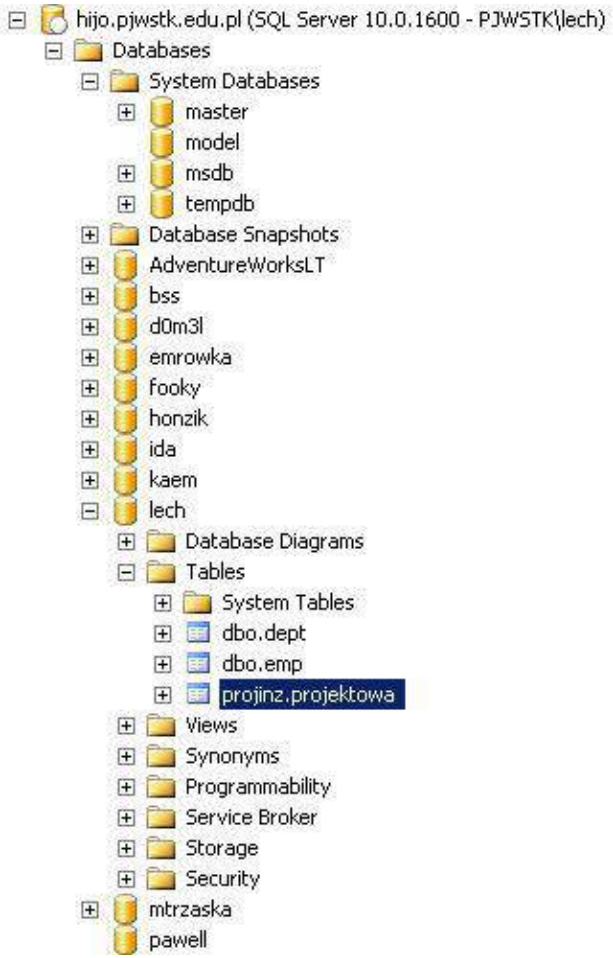
Typy replik częściowych

- Anonimowa – nie rejestruje nazw ani lokalizacji członków zestawu replik podczas operacji synchronizacji, może wymieniać dane z każdym innym członkiem zestawu replik

MS SQL Serwer

- Bazy danych są przechowywane w plikach, które odpowiadają fizycznym plikom na dysku. Domyślnie przy tworzeniu bazy danych jest tworzony jeden plik dla danych oraz jeden dla dziennika (logu) transakcji.
- Bazy danych dzielą się na systemowe (*master, model, tempdb, msdb) i użytkownika.*
- Obiekty bazy danych są podzielone na schematy. Pełna referencja do obiektu na serwerze: *bazadanych.schemat.obiekt (domyślny* schemat *dbo).*

Struktura serwera w Object Explorer



Bazy systemowe

Master – podstawowa systemowa baza danych, zawierająca metadane np. informacje o kontach użytkowników, bazach danych i innych obiektach serwera. Korzystamy za pomocą odpowiednich narzędzi oraz procedur systemowych.

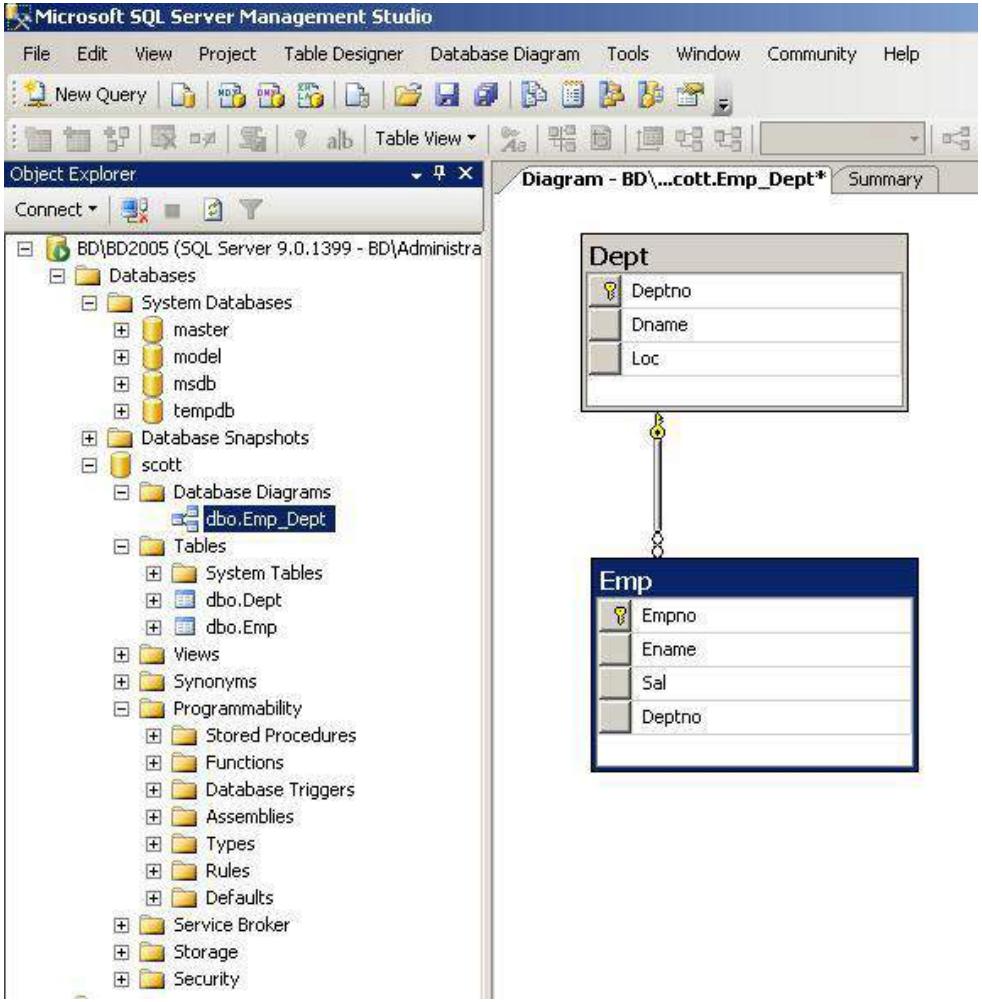
Model – jest szablonem bazy danych. Gdy tworzymy nową bazę danych użytkownika, jest ona kopią bazy *model. Dzięki niej* możemy uprościć tworzenie wielu identycznych baz danych. Jeżeli w bazie *Model utworzymy tabele, procedury składowane i* inne obiekty, to każda nowa baza danych będzie te obiekty zawierać.

Bazy systemowe

Tempdb – baza w której serwer trzyma informacje tymczasowe np. tabele i procedury tymczasowe użytkowników.

Msdb – przechowuje informacje na temat zaplanowanych zadań. Korzystamy za pomocą odpowiednich narzędzi orazprocedur systemowych.

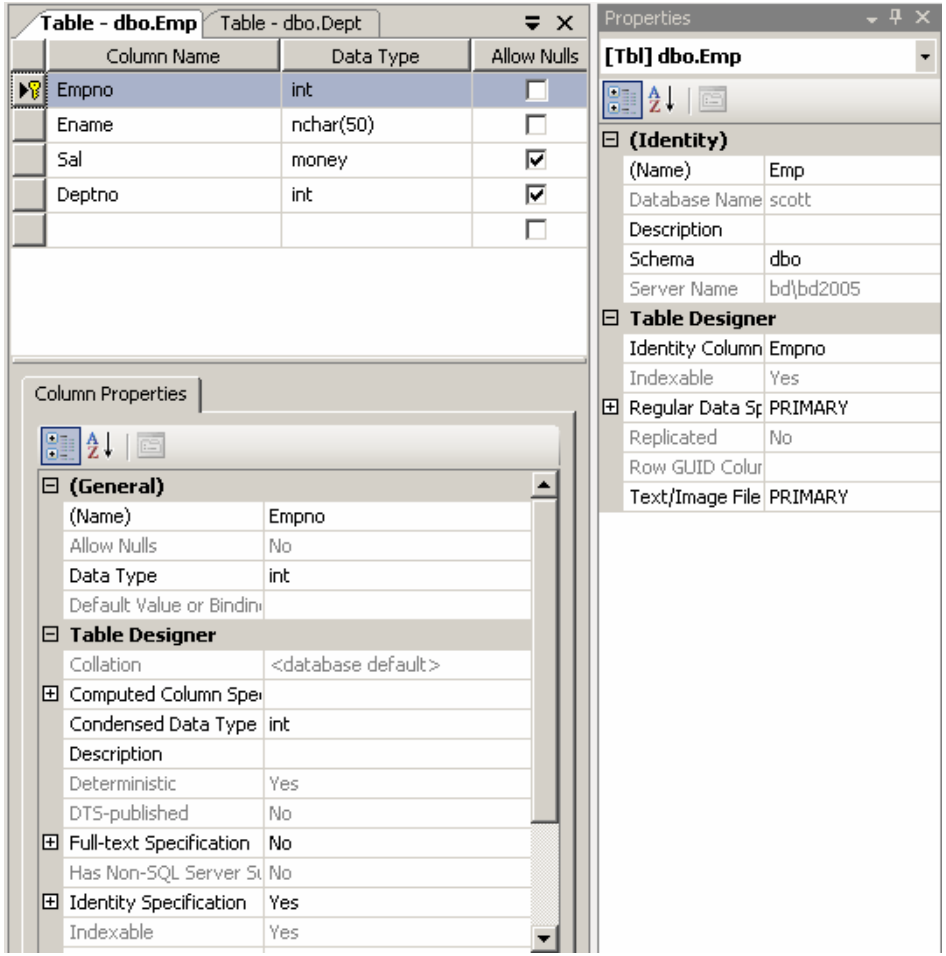
SQL Server Management Studio – graficzny interfejs do baz danych na serwerze



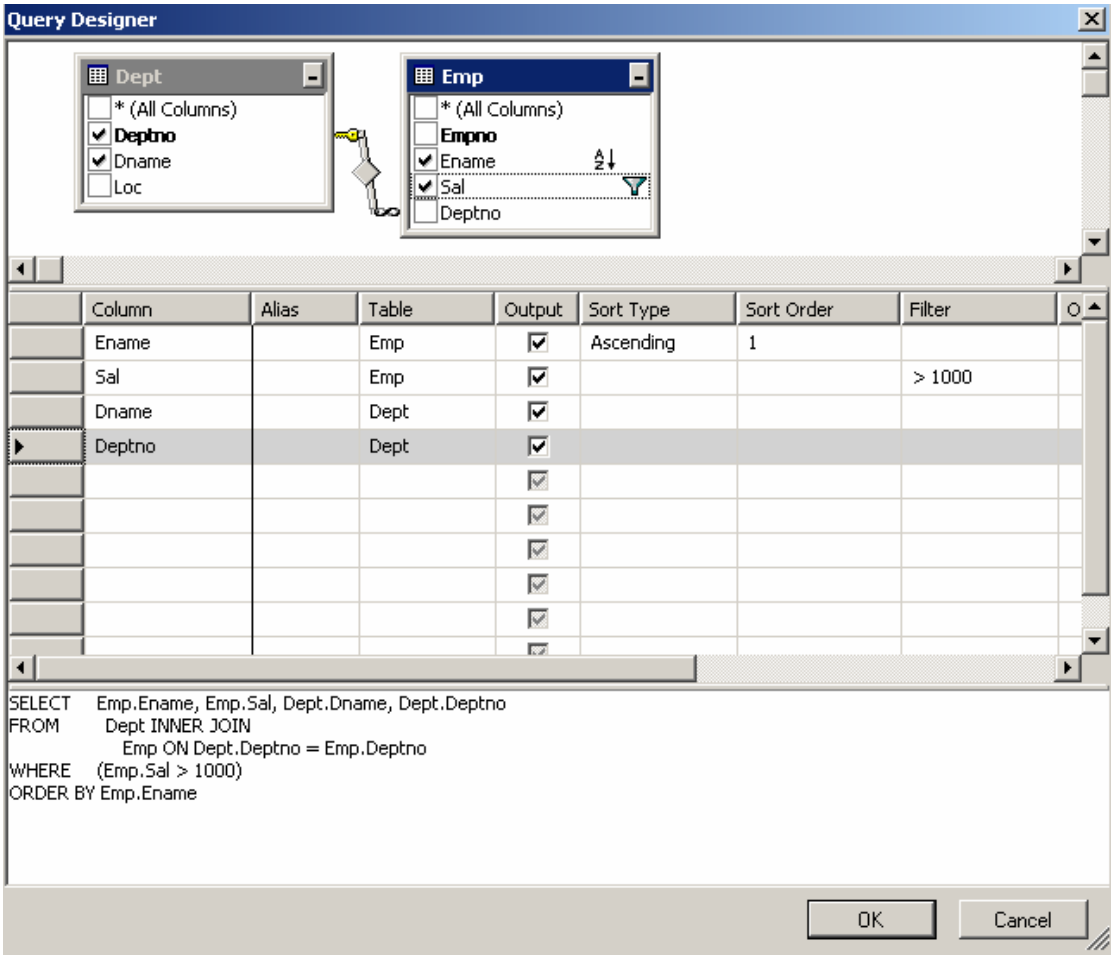
SQL Server Management Studio – graficzny interfejs do baz danych na serwerze

- Tworzenie i administrowanie bazami danych.
- Łatwiejsze niż z linii poleceń tworzenie i modyfikowanie tabel, procedur, wyzwalaczy, perspektyw, indeksów, typów danych, użytkowników, uprawnień.
- Wprowadzanie, wyświetlanie, import i eksport danych.

Widok projekt



Tworzenie zapytania do bazy danych



SQL

W oprogramowaniu Microsoft SQL Server zaimplementowano język SQL w oparciu o standard SQL-92. Większość instrukcji omówionych w wykładzie "Systemy baz danych", działa również na serwerze SQL Server.

CREATE TABLE

```
CREATE TABLE Osoby
(IdOsoby INT PRIMARY KEY IDENTITY,
Imie VARCHAR(20),
Nazwisko VARCHAR(30));
```

W tak utworzonej tabeli, wartości klucza głównego *IdOsoby są* generowane automatycznie.

```
INSERT INTO Osoby VALUES ('Jan', 'Kowalski');
SELECT * FROM Osoby;
```

CREATE SCHEMA

CREATE SCHEMA myschema;
Dodanie nowej tabeli do schematu:

CREATE TABLE myschema.mytable1(x int);
Odwołanie do tabeli w schemacie:

SELECT * FROM myschema.mytable;
Usunięcie schematu (najpierw obiekty):

DROP myschema.mytable;
DROP myschema;

Domyślny schemat dbo

Tabele tymczasowe

• *Lokalne tabele tymczasowe - nazwa takiej tabeli ma postać #nazwa_tabeli. Jest ona widoczna tylko w sesji, w której została* utworzona. Po zakończeniu sesji tabela jest usuwana.
• *Globalne tabele tymczasowe różnią się tym, że są widoczne* również spoza sesji, w której zostały utworzone. Nazwa takiej tabeli ma postać *##nazwa_tabeli. Po zakończeniu sesji wszystkich* użytkowników korzystających z tabeli, jest ona usuwana.
• Tworzone w systemowej bazie **tempdb**
• Do tworzenia tabel tymczasowych używamy polecenia CREATE TABLE w sposób analogiczny do normalnych tabel.

Tabele tymczasowe

```
CREATE TABLE #MyTempTable
(cola INT PRIMARY KEY);
INSERT INTO #MyTempTable VALUES (1);
```

- Można używać wszystkich deklaratywnych więzów spójności z wyjątkiem FOREIGN KEY.

Użycie dziennika transakcji dla tabel tymczasowych

Operacje na tabelach tymczasowych są wpisywane do dziennika transakcji w bazie *tempdb w sposób uproszczony, więc mogą* zostać wycofane w razie potrzeby. Uproszczenie oznacza brak możliwości odtworzenia przy awarii ale jednocześnie istotnie zwiększoną wydajność operacji na nich (do 4 razy szybciej).

Transakcje

BEGIN TRAN - rozpoczęcie transakcji

COMMIT TRAN - zatwierdzenie transakcji

ROLLBACK TRAN - wycofanie transakcji

Prawa dostępu do serwera MS SQL Serwer

Użytkownik domyślnie nie ma żadnych uprawnień. MS SQL Server umożliwia zarządzanie uprawnieniami poprzez tworzenie ról.

Wyróżniamy w MS SQL Server trzy rodzaje ról:

- Serwerowe
- Bazodanowe
- Definiowane przez użytkowników

Prawa dostępu do serwera MS SQL Serwer

Role serwerowe:

Sysadmin: rola umożliwia wykonać każdą operację na każdej bazie danych

Serweradmin: rola nie pozwala na administrowanie danymi, pozwala na konfigurowanie sytemu

Setupadmin: rola pozwala konfigurować system i nim zarządzać

Securityadmin: rola pozwala zarządzać użytkownikami i zarządzać bezpieczeństwem

Prawa dostępu do serwera MS SQL Serwer

Role serwerowe (ciąg dalszy):

Processadmin: rola umożliwia kontrolowanie procesów, np.kasowanie działających zapytań

DBCreator: rola umożliwia tworzenie, kasowanie, modyfikowanie, odzyskiwanie bazy danych

DiskAdmin: rola umożliwia zarządzanie plikami

Bulkadmin: rola umożliwia wykonanie polecenia BULK INSERT (masowego wstawiania danych)

Prawa dostępu do serwera MS SQL Serwer

Role bazy danych:

dbowner: rola pozwala wykonywać zapytania DDL

db_accessadmin: rola pozwala decydować, które konta logowania mają prawa dostępu do bazy danych

Prawa dostępu do serwera MS SQL Serwer

Tworzenie ról:

CREATE ROLE Pracownicy;

Usuwanie roli:

DROP ROLE Pracownicy;

Prawa dostępu do serwera MS SQL Serwer

Tworzenie ról:

CREATE ROLE Pracownicy;

Usuwanie roli:

DROP ROLE Pracownicy;

Prawa dostępu do serwera MS SQL Serwer

Przypisywanie do ról:

GRANT Pracownicy to Kowalski;

Prawa dostępu do serwera MS SQL Serwer

UPRAWNIENIA dzielimy na dwie kategorie:

- Uprawnienia do wykonywania określonych instrukcji
- Uprawnienia obiektowe

Prawa dostępu do serwera MS SQL Serwer

Uprawnienia do wykonywania określonych instrukcji:

- ALTER ANY ROLE – uprawnienia do modyfikowania ról
- ALTER ANY USER – uprawnienia do modyfikowania kont użytkowników
- BACKUP DATABASE – wykonywanie kopii bezpieczeństwa
- CREATE FUNCTION – tworzenie funkcji
- CREATE PROCEDURE – tworzenie procedur
- CREATE SCHEMA – tworzenie schematów
- CREATE TABLE – tworzenie tabel
- TAKE OWNERSHIP – przejmowanie obiektów na własność
- VIEW DEFINITION – odczytywanie metadanych obiektów

Prawa dostępu do serwera MS SQL Serwer

Uprawnienia obiektowe (użytkownicy mogą odwoływać się do wybranych obiektów bazy danych i wykonywać określone operacje):

- SELECT, INSERT, UPDATE, DELETE, REFERENCE – uprawnienia pobierania, wstawiania, modyfikowania i usuwania danych oraz tworzenia kluczy obcych
- SELECT, UPDATE – uprawnienia do kolumn
- EXECUTE – do funkcji i procedur składowanych

MS SQL Server – kopie bezpieczeństwa

- Pełna kopia bazy danych:

BACKUP DATABASE Księgarnia TO DISK = 'C:\kopia\1'

- Przyrostowa kopia danych (zapisy zmian od czasu pełnej kopii bazy danych):

BACKUP DATABASE Księgarnia TO ksiegarnia_kopia_przyrost WITH DIFFERENTIAL

- Przywracanie pełnej bazy danych

RESTORE DATABASE Księgarnia FROM kopia.bak

MS SQL Server – Sprawdzanie spójności bazy

- **DBCC CHECKDB** – sprawdzenie spójności bazy
- **DBCC CHECKCATALOG** – sprawdzenie integralności referencyjnej tabel systemowych
- **DBCC CHECKTABLE** – sprawdzenie spójności na poziomie pojedynczej tabeli