SQL - DML

Data Manipulation Language

SQL – DML

JĘZYK DEFINIOWANIA DANYCH

- Jednym z zadań realizowanych przez język SQL jest pobieranie i modyfikowanie danych. Instrukcje do tego przeznaczone tworzą tzw. język manipulowania danymi (Data Manipulation Language – DML).

SQL – DML

JĘZYK DEFINIOWANIA DANYCH

- SELECT – wybiera dane z bazy danych;
- INSERT – umieszcza nowe wiersze w tabeli;
- UPDATE – zmienia wartość istniejącego wiersza;
- DELETE – usuwa wiersze z tabeli;

SQL – INSERT

JĘZYK DEFINIOWANIA DANYCH

- Instrukcja ma postać:

INSERT INTO nazwa_tabeli (kolumna1, kolumna2, …)

VALUES (wartość1, wartość2,...)

SQL – INSERT

JĘZYK DEFINIOWANIA DANYCH

- W wyniku działania instrukcji do tabeli zostanie dodany nowy wiersz.
- W polu **kolumna1** zostanie zapisana wartość **wartość1** itd.
- W poleceniu można nie podawać nazw kolumn:

INSERT INTO nazwa_tabeli VALUES (wartość1,…)

- W takim wypadku wartości zostaną wprowadzone do kolejnych kolumn.

SQL – INSERT

JĘZYK DEFINIOWANIA DANYCH

- Kolumny z atrybutami auto increment, identity zostaną wypełnione automatycznie.
- Jeżeli w poleceniu nie zostaną uwzględnione wszystkie kolumny pozostaną puste chyba, że zdefiniowano dla nich wartość domyślną.

SQL – INSERT

JĘZYK DEFINIOWANIA DANYCH

- Jeżeli podczas definiowania tabeli dla określonej kolumny zostało zabronione zapisywanie wartości nieznanej (NULL), to próba zapisania nowego wiersza bez podania wartości dla tej kolumny zakończy się niepowodzeniem.

SQL – INSERT

JĘZYK DEFINIOWANIA DANYCH

- Ze względu na to, że wprowadzanie pojedynczych wierszy jest uciążliwe, niektóre serwery pozwalają na wpisanie w klauzuli VALUES wartości, które zostaną umieszczone w kilku wierszach.

```
INSERT INTO nazwa_tabeli VALUES
(wartość1_rekord1, wartość2_rekord1, …),
(wartość1_rekord2, wartość2_rekord2, …),
(wartość1_rekord3, wartość2_rekord3, …);
```

ĆWICZENIE 1

JĘZYK DEFINIOWANIA DANYCH

- Wypełnij tabelę **klienci** przykładowymi danymi.
- Wypełnij tabelę **pracownicy** przykładowymi danymi.
- Wstaw co najmniej 5 rekordów do każdej z powyższych tabel.

SQL – SELECT

JĘZYK DEFINIOWANIA DANYCH

- Instrukcja SELECT określa jakie dane zostaną zwrócone w wyniku jej wykonania.
- Ogólna postać polecenia:

SELECT nazwa_kolumny FROM nazwa_tabeli;

- Polecenie zwróci zawartość kolumny nazwa_kolumny z tabeli nazwa_tabeli.

SQL – SELECT

JĘZYK DEFINIOWANIA DANYCH

- Wybieranie niektórych kolumn z tabeli nazywane jest **projekcją** lub **selekcją pionową** tabeli.
- W poleceniu SELECT zamiast wpisywania listy wszystkich pól tabeli można użyć symbolu *

SELECT * FROM nazwa_tabeli;

SQL – ĆWICZENIE 2

JĘZYK DEFINIOWANIA DANYCH

- Wyświetl wszystkie rekordy z tabel: **klienci i pracownicy**.

SQL – UPDATE

JĘZYK DEFINIOWANIA DANYCH

- Instrukcja UPDATE służy do aktualizowania danych w tabeli.

UPDATE nazwa_tabeli

SET nazwa_pola=”nowa_wartość”;

- W klauzuli UPDATE podaje się nazwę modyfikowanej tabeli, a w klauzuli SET nazwę modyfikowanej kolumny oraz nową wartość.

SQL – UPDATE

JĘZYK DEFINIOWANIA DANYCH

- Wykonanie instrukcji w takiej postaci spowoduje zmianę we wszystkich rekordach w danej tabeli.
- Jeżeli modyfikacja ma dotyczyć tylko wybranych wierszy należy do instrukcji dodać klauzulę **WHERE**

UPDATE nazwa_tabeli

SET nazwa_kolumny=”nowa_wartość”

WHERE warunek;

SQL – UPDATE

JĘZYK DEFINIOWANIA DANYCH

- Warunek związany z klauzulą WHERE to przeważnie porównanie wartości danego pola z wartością oczekiwaną. Przykłady:

WHERE nazwisko=‘Nowak’

WHERE id=‘5’

SQL – UPDATE

JĘZYK DEFINIOWANIA DANYCH

- Warunków może być więcej niż jeden. Przykłady:

WHERE nazwisko=‘Nowak’ AND imie=‘Andrzej’

WHERE data=‘2016-02-24’ AND kwota=‘500’

SQL – UPDATE

JĘZYK DEFINIOWANIA DANYCH

- Przy użyciu instrukcji UPDATE możliwe jest modyfikowanie wielu kolumn jednocześnie:

UPDATE nazwa_tabeli SET kolumna1=”wartość1”, kolumna2=”wartość2” WHERE warunek1 AND warunek2;

SQL – ĆWICZENIE 3

JĘZYK DEFINIOWANIA DANYCH

- Zmień strukturę tabeli klienci i dodaj pole **miasto**
- Zmodyfikuj wszystkie rekordy tak by każdemu klientowi przypisać miasto: **3 klientów: Warszawa, 2 klientów: Kielce**
- Za pomocą polecenia SELECT wyświetl wszystkie rekordy z tabeli klienci;

SQL – ĆWICZENIE 3

JĘZYK DEFINIOWANIA DANYCH

- Zmodyfikuj 3 rekord w tabeli **klienci** i wprowadź do niego dane: **Marek Marecki, Błotna 3, 900678456, Rzeszów**
- Zmodyfikuj za pomocą jednego polecenia rekordy 1,2 i 5 w tabeli **pracownicy** i wprowadź do nich dane:
  Jacek Jackowski, 200456982, referent
  Michał Michalak, 800657345, referent
  Daniel Danielewski, 700756234, konsultant

SQL – DELETE

JĘZYK DEFINIOWANIA DANYCH

- Instrukcja DELETE służy do usuwania wierszy z tabeli.

DELETE FROM nazwa_tabeli;

- Wykonanie powyższego polecenia spowoduje usunięcie wszystkich rekordów z tabeli.
- Aby usuwać wybrane rekordy należy ponownie skorzystać z klauzuli **WHERE**.

SQL – ĆWICZENIE 4

JĘZYK DEFINIOWANIA DANYCH

- Usuń z tabeli pracownicy wszystkie rekordy zawierające dane pracowników zatrudnionych na stanowisku **referent**.
- Usuń z tabeli klienci wszystkich klientów z **Warszawy**.
- Wyświetl za pomocą polecenia SELECT wszystkie rekordy z tabeli **pracownicy**.
- Wyświetl za pomocą polecenia SELECT wszystkie rekordy z tabeli **klienci**.

SQL – SELECT

JĘZYK DEFINIOWANIA DANYCH

- Klauzula DISTINCT: klauzula eliminuje z wyświetlania wyniku zapytania powtarzające się wiersze.
- Przykładowo: chcąc otrzymać informację o tym z jakich miejscowości pochodzą klienci wystarczy wydać polecenie:

SELECT DISTINCT miasto FROM klienci;

SQL – SELECT

WYRAŻENIA W INSTRUKCJI SELECT

- W instrukcji SELECT oprócz nazw kolumn mogą występować wyrażenia.
- Tworzone są one z nazw kolumn, funkcji systemowych, stałych i operatorów i muszą zwracać pojedyncze wartości.
- Do tworzenia wyrażeń służy słowo kluczowe **AS**

SELECT kolumna1+’ ’+kolumna2 AS nazwa FROM nazwa_tabeli

SQL – ĆWICZENIE 5

WYRAŻENIA W INSTRUKCJI SELECT

- Wykonaj polecenie:
  SELECT concat(nazwisko,’ ’,imie) **AS** Klient FROM klienci;

- Wprowadź do tabeli **towary** pięć przykładowych rekordów z różnymi cenami (5, 15, 40, 120, 345).
- Wykonaj polecenie:
  SELECT nazwa, cena+cena*0.07 **AS** ”Cena sprzedaży”
  FROM towary;

SQL – SELECT
SORTOWANIE WYNIKÓW

- W celu posortowania danych należy dodać klauzulę ORDER BY

SELECT * FROM nazwa_tabeli

ORDER BY nazwa_kolumny;

SQL – SELECT
SORTOWANIE WYNIKÓW

- Domyślnie dane są sortowane rosnąco. Można jednak zdefiniować sposób sortowania dodając słowa kluczowe **ASC** i **DESC.**

SQL – ĆWICZENIE 6
SORTOWANIE WYNIKÓW

- Wyświetl wszystkie rekordy z tabeli **towary** posortowane malejąco wg ceny.
- Wyświetl wszystkie rekordy z tabeli **pracownicy** posortowane rosnąco wg nazwiska.
- Wyświetl wszystkie rekordy z tabeli **klienci** posortowane rosnąco wg miasta.

SQL – SELECT
SELEKCJA REKORDÓW

- Ograniczenie ilości wyników odbywa się za pomocą klauzuli WHERE
- Wybieranie tylko niektórych wierszy z tabeli nazywane jest **selekcją rekordów**.
- Klauzula WHERE musi wystąpić bezpośrednio po klauzuli FROM.

SELECT * FROM nazwa_tabeli WHERE warunek;

SQL – SELECT
SELEKCJA REKORDÓW

- Przykłady:
  WHERE rok_wydania=’2012’
  WHERE cena BETWEEN 50 AND 100
  WHERE nazwisko LIKE’A%’
  WHERE adres IS NULL
  WHERE miasto=’Warszawa’ AND adres IS NULL

SQL – ĆWICZENIE 7

SELEKCJA REKORDÓW

- Wyświetl dane wszystkich klientów z Rzeszowa.
- Wyświetl dane wszystkich klientów z Kielc.
- Wyświetl wszystkich konsultantów z tabeli pracownicy.
- Wyświetl tylko te produkty, których cena zawiera się w przedziale od 30 do 200.

SQL – SELECT

KLAUZULA LIMIT

- Klauzula LIMIT służy do wybrania określonej liczby wierszy.
- Składnia:
  SELECT * FROM nazwa_tabeli LIMIT 5;

SQL – SELECT
KLAUZULA LIMIT

- Przykład:
  SELECT * FROM towary LIMIT 7;
- Polecenie spowoduje wyświetlenie 7 pierwszych wyników zapytania.

SQL – ĆWICZENIE 8

KLAUZULA LIMIT

- Za pomocą klauzuli LIMIT wyświetl najdroższy artykuł z tabeli towary.
- Za pomocą klauzuli LIMIT wyświetl najtańszy artykuł z tabeli towary.

SQL – SELECT
KLAUZULA LIMIT

- Przykład:
  SELECT * FROM towary LIMIT 5,10;
- Polecenie spowoduje wyświetlenie wyników zapytania z pozycji od 6 do 15.

SQL – ĆWICZENIE 9

KLAUZULA LIMIT

- Za pomocą klauzuli LIMIT wyświetlaj po 2 artykuły z tabeli towary.
- Za pomocą klauzuli LIMIT wyświetl 2 klientów zaczynając od 3.

SQL – SELECT

OPERATOR IN

- Za pomocą operatora IN można sprawdzać czy wynik należy do określonego zbioru
- Przykład:
  SELECT `username` WHERE `user_id` IN (1,5,10);
- Powyższe zapytanie wyświetli rekordy, w których pole user_id przyjmuje wartości 1, 5 i 10.

SQL – SELECT

OPERATOR LIKE

- Za pomocą operatora LIKE można sprawdzać czy wynik zawiera określony fragment
- Przykład:
  SELECT `username` WHERE `username` LIKE ‘%ma’;
- Powyższe zapytanie wyświetli rekordy, w których pole username kończy się na **ma**.

SQL – SELECT

OPERATOR LIKE

- Przykład:
  SELECT `username` WHERE `username` LIKE ‘%ma%’;
- Powyższe zapytanie wyświetli rekordy, w których pole username zawiera **ma**.
- Przykład:
  SELECT `username` WHERE `username` LIKE ‘ma%’;
- Powyższe zapytanie wyświetli rekordy, w których pole username zaczyna się na **ma**.

SQL – ĆWICZENIE 8

SELEKCJA REKORDÓW

- Wyświetl dane wszystkich klientów, których imię zawiera literę A.
- Wyświetl dane wszystkich klientów, których miasto zaczyna się na K.
- Wyświetl wszystkich konsultantów z tabeli pracownicy, których imię zawiera literę A lub literę B.

SQL – SELECT

OPERATOR BETWEEN

- Przykład:
  SELECT `towar` WHERE `cena` BETWEEN 100 AND 200;
- Powyższe zapytanie wyświetli rekordy, w których cena zawiera się w przedziale od 100 do 200.

SQL – SELECT

OPERATOR REGEXP

- MySQL umożliwia także korzystanie z **wyrażeń regularnych** za pomocą polecenia REGEXP.
- Przykład:
  SELECT user_id FROM users WHERE username REGEXP BINARY ‘[a-z]’;
- Powyższe zapytanie spowoduje wyświetlenie tylko rekordów, w których pole username zawiera tylko małe litery.

SQL – SELECT

OPERATOR IS NULL i IS NOT NULL

- Przykład:
  SELECT user_id FROM users WHERE email IS NULL;
- Powyższe zapytanie spowoduje wyświetlenie tylko rekordów, w których pole email jest puste.
- Przykład:
  SELECT user_id FROM users WHERE email IS NOT NULL;
- Powyższe zapytanie spowoduje wyświetlenie tylko rekordów, w których pole email NIE jest puste.

SQL – ĆWICZENIE 9

SELEKCJA REKORDÓW

- Wyświetl dane wszystkich towarów, których nazwa jest złożona z małych liter.
- Dodaj 2 klientów bez podawania miasta.
- Wyświetl dane wszystkich klientów, dla których podano miasto.
- Wyświetl dane wszystkich klientów, dla których nie podano miasta.

SQL – SELECT

KLAUZULA GROUP BY

- Klauzula GROUP BY działa podobnie do DISTINCT
- Przykład:
  SELECT miasto FROM klienci GROUP BY miasto;
- Powyższe zapytanie spowoduje pogrupowanie wyników wg pola miasto.

SQL – SELECT

KLAUZULA HAVING

- Klauzula ta służy do wybierania grup spełniających pewne warunki.
- Jest to polecenie analogiczne do WHERE z tym, że dotyczy grup a nie pojedynczych rekordów.

SQL – SELECT

KLAUZULA HAVING

- PRZYKŁAD:
  SELECT kontynent, AVG(powierzchnia) AS srednia_powierzchnia
  FROM panstwa GROUP BY kontynent HAVING srednia_powierzchnia>500000;

SQL – SELECT

FUNKCJE AGREGUJĄCE

- COUNT(kolumna) - zlicza liczbę pól występujących w kolumnie
- SUM(kolumna) – sumuje wszystkie wartości w danej kolumnie
- MAX(kolumna) – zwraca największą wartość w danej kolumnie
- MIN(kolumna) – zwraca najmniejszą wartość w danej kolumnie
- AVG(kolumna) – zwraca średnią wartość w danej kolumnie

SQL – SELECT

FUNKCJE AGREGUJĄCE

- Przykład:
  SELECT COUNT(username) FROM users;
- Powyższe polecenie wypisze liczbę wierszy, w których wypełniono pole username.

SQL – ĆWICZENIE 10

SELEKCJA REKORDÓW

- Wyświetl liczbę wszystkich klientów.
- Wyświetl wartość wszystkich artykułów w tabeli towary.
- Wyświetl najwyższą cenę z tabeli towary.
- Wyświetl najniższą cenę z tabeli towary.
- Wyświetl średnią cenę z tabeli towary.