Въвеждане изтриване и промяна на полетата на данните

1. Операторът INSERT

1.1. Общ синтаксис на оператора

За въвеждането на данни в таблиците чрез SQL команди се използва оператора INSERT:

INSERT INTO *име_на_таблица (списък с полета)*
VALUES *(списък с константи);*

Списъкът с полетата може да не се задава ако данните ще се въвеждат в същия ред в който са дефинирани полетата в структурата на таблицата.

Пример:

INSERT INTO *CUSTOMERS*
VALUES *(0, 'Petr','petr@example.com','888-14-19','Sofia');*

Числата(цели и реални) се пишат без кавички, Но низовите данни се поставят в кавички. В този случай ние въвеждаме данните в полетата в същия порядък в който са представени в структурата на таблицата. В качеството на стойност за поле от тим AUTO_INCREMENT може да се зададе 0 както е показано в горния пример. Тъй като това поле е зададено AUTO_INCREMET то ние можем винаги за всеки отделен запис да слагаме id 0 а необходимата стойност ще бъде изчислена от сървъра и ще бъде въведена автоматично.

Всяка стойност в „списъка със стойностите“ трябва да има такъв тип данни, който да съответства на типа данни на колоната, в която тази стойност трябва да бъде въведена. Съгласно стандарта на ANSI тези стойности не могат да включват изрази.

Пример: В качеството на числова стойност 4 не може да се окаже със израза 2+2 или който и да е друг израз. Представянето става само чрез самата числова стойност в случая 4.

1.2. NULL стойности:

Като пример нека въведем следната команда:

INSERT INTO *CUSTOMERS*
VALUES *(0, 'Petar','p_petrov@example.com','888-15-18', NULL);*

В този случай оператора на базата данни не знае адреса на клиента затова задава NULL. Трябва да се има предвид че не всички полета могат да

приемат стойност NULL. Ако в структурата на таблицата за полето е посочено че не може да приема нулеви стойности (NOT NULL), тогава даденото поле не може да приема NULL и съответно ще получите грешка при опит за въвеждането на такава стойност. Също така трябва да се има предвид че не може да се задава NULL стойност на поле отговарящо на първичен ключ. Изключение представляват полетата AUTO_INCREMENT. В нашия случай при задаване на записи в таблицата CUSTOMERS за полето IDCustomers може да зададете стойност NULL вместо 0. Сървъра отново ще изчисли необходимата стойност и ще я въведе автоматично.

Пример:

INSERT INTO *CUSTOMERS*
VALUES *(NULL, 'Petar','p_petrov@example.com','888-15-18', NULL);*

ЗАБЕЛЕЖКА: Тъй като NULL се явява специален символ а не символна стойност той се пише без единични кавички.

1.3. Произволен ред на следване на полетата.

INSERT INTO *CUSTOMERS (NAME, PHONE, EMAIL)*
VALUES *('Petar' ,'888-15-18', 'p_petrov@example.com');*

Стойността на полетата може да бъде задавана по различен начин от този посочен в структурата на таблицата. За целта полетата за които се задава стойност трябва да бъдат посочени непосредствено след името таблицата в реда в който ще се задава тяхната стойност. Възможно е да не се задава стойност за всички полета тогава те ще бъдат попълнени със стойността по подразбиране. Стойност по подразбиране може да бъде зададена или стойност NULL или напълно определена стойност зададена при създаването на таблицата.

1.4. Въвеждане на данни от една таблица в друга.

ПРИМЕР: Да си представим че имаме таблица vip_customers и в нея искаме да нанесем данни от таблица customers:

INSERT INTO *vip_customers*
SELECT *
FROM *CUSTOMERS*
WHERE *address = 'Sofia';*

Разбира се условието (WHERE) може да бъде зададено по ваше усмотрение. При изпълнението на този оператор всички редове, получени в резултата на заявката (т.е. информацията за всички купувачи от град София), ще бъдат добавени в таблицата vip_customers, която трябва да отговаря на следните изисквания:

- Трябва да съществува, т.е. предварително да е създадена
- Структурата на тази таблица трябва да бъде същата каквато и на таблица customers

Да разгледаме следния пример: Имаме таблица Email в която се съхраняват имената на клиентите и техните електронни пощи. Т.е. структурата на новата таблица е различна от тази на таблица customers. Копирането на данните от таблица customers в новата таблица може да бъде направено посредством посочване на полетата:

INSERT INTO *EMAILS (IDCustomers, Name, Email)*
SELECT *IDCustomers, Name, Email*
FROM *CUSTOMERS*

Тук имената на полетата съвпадат, но това не е задължително, важното е да съвпадат типовете на полетата.

2. ОПЕРАТОР DELETE

ЗАБЕЛЕЖКА: *ПОЛЗВАЙТЕ ТОЗИ ОПЕРАТОР С ПОВИШЕНО ВНИМАНИЕ. ИМА ОПАСНОСТ ДА ЗАТРИЕТЕ ЦЯЛАТА ТАБЛИЦА*

Синтаксис :

DELETE *име_на_таблица*
*[**WHERE** условие*
LIMIT *n];*

Ако не укажете WHERE рискувате за изтриете цялата таблица. Също така е желателно да задавате и LIMIT ограничаващ количеството на изтриваните записи.

Примери:

DELETE FROM *CUSTOMERS*
WHERE *IDCustomers>100*

Изтрива всички потребители които имат ID >100

DELETE FROM *CUSTOMERS*
WHERE *IDCustomers>100*
LIMIT *3;*

Ако записите в таблицата вървят поред то тази заявка ще изтрие потребителите с ID 101, 102 и 103. Същото нещо може да се направи и със следната заявка и би било по коректно в случай че записите не вървят подред.

DELETE FROM *CUSTOMERS*
WHERE *(IDCustomers>100)* ***AND*** *(IDCustomers<104)*

Ако искате да изтриете всички записи от таблицата Customers то може да използвате следната команда:

DELETE FROM *CUSTOMERS*

При изтриването на записи трябва да се има предвид че броячат на полето от типа AUTO_INCREMENT не се нулира. Т.е. да си представим че преди изтриването на всички записи сме имали 8 клиента. Изтриваме всички записи от таблицата и въвеждаме нов. То ID-то на новия клиент няма започне от 1 а от 9. За да може номерацията на полето да започне от начало то трябва да нулираме брояча. Това става със командата ALTER която ще има следния синтаксис:

ALTER TABLE *name_table* ***AUTO_INCREMENT****=0;*

3. Оператор UPDATE

Синтаксис:

UPDATE *Име_на_таблица*
SET *Поле1=Стойност1, поле2=стойност2, …*
*[****WHERE*** *условие];*

Ако не бъде зададена последната част от заявката определяща кои именно записи на полетата трябва да бъдат обновени то ще бъдат обновени всичките записи.

Примери:

UPDATE *CUSTOMERS*
SET *address=’LONDON’*
WHERE *address=’n/a’;*

Тази заявка ще обнови всички записи в таблицата CUSTOMERS където полето адрес има стойност ‘n/a’. Новата стойност на полето ще е ‘LONDON’.

Списъкът на полетата се отделя със запетая

UPDATE *CUSTOMERS*
SET *name=’Jack’, email= ‘jack@example.com’*
WHERE *IDCustomers=3;*

В оператора UPDATE могат да се използват аритметични израз, което много удобно ако сме решили да направим преоценка.

ЗАДАЧИ:

Задача 1: Реализирайте повишение на цените на някои от продуктите с 10%.

Упътване: Стойността на полето трябва да се умножи с 1.1 за повишение с 10%.

Задача 2: Променете структурата на таблицата ORDERS като добавите още едно поле date, в което да се съхранява датата на поръчката. Попълнете данни за това поле с коректно зададени дати за всеки от записите. Създайте нова таблица totals с информация за продажбите по конкретни дни. Т.е. новата таблица трябва да съдържа общата сума за стойността на поръчките за всеки ден. Попълнете новата таблица с командата INSERT като използвате таблица ORDERS.