Тема лекції 6:

Інструкції маніпуляції з таблицями. Інструкції модифікації даних.

- Створення таблиць
- Обмеження на дані
- Модифікація таблиць
- Видалення таблиць
- Внесення даних
- Зміна даних
- Видалення даних

Мова визначення даних (DDL – Data Definition Language)

Інструкції DDL:

- CREATE TABLE – створення таблиці
- ALTER TABLE - модифікація структури таблиці
- DROP TABLE – видалення таблиці

Інструкція створення таблиці (CREATE TABLE)

- Створює порожню таблицю без записів (значення вводяться засобами DML)
- CREATE TABLE, як правило, визначає ім’я таблиці та набір стовпців, вказаних у певному порядку, тобто визначає структуру таблиці
- Типи даних стовпців повинні бути сумісними зі стандартом ANSI
- Значення розміру стовпця залежить від типу даних
- Кожна таблиця повинна мати хоча б один стовпець

Інструкція створення таблиці (CREATE TABLE)

- Приклад 1. Наступна інструкція створює таблицю Salers:

```
CREATE TABLE Salers
  (snum integer,
   sname char(20),
   city char(10),
   comm decimal);
```

Інструкція створення тимчасової таблиці (CREATE TEMPORARY TABLE)

- Інструкція подібна до CREATE TABLE , крім ключового слова TEMPORARY.
- Тимчасова таблиця, на відміну від постійної, існує лише на час сеансу роботи з базою даних, в якому вона була створена.
- Тимчасова таблиця може бути доступна іншим користувачам, як і постійна таблиця.
- Тимчасові таблиці створюються для представлення в них поточних підсумкових (звітних) даних, які є доступними декільком користувачам бази даних.

Обмеження на значення даних

- Обмеження – частина визначення таблиці, яке задає певні критерії значенням, які вводяться в таблицю
- Обмеження – є одним із засобів організації підтримки цілісності даних безпосередньо в БД
- СУБД перевіряє відповідність встановлених обмежень і значень при модифікації даних

Обмеження на значення даних

- NOT NULL
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY
- CHECK

Обмеження стовпця та обмеження таблиці

- Обмеження стовпця застосовується лише до індивідуальних стовпців
- Обмеження таблиці застосовується до груп стовпців
- Визначаються обмеження наступним чином:

```
CREATE TABLE <ім’я табл>
  (<ім’я стовп> <обмеж на стовп>,
   <ім’я стовп> <тип даних> <обмеж на стовп>, ...
   CONSTRAINT <ім’я обмеж на табл >
     < обмеж на табл> (<ім’я стовп> [,<ім’я стовп>]));
```

Обмеження NOT NULL

- Забороняє використання NULL значень у стовпцях
- Приклад 1 (продовж.): Створимо таблицю Salers, не дозволяючи поміщати NULL у стовпці snum або sname:

```
CREATE TABLE Salers
 (  snum integer NOT NULL,
    sname char(20) NOT NULL,
    city char(10),
    comm decimal);
```

Обмеження унікальності UNIQUE

- Використовується, щоб уникнути певного безладдя в БД, коли необхідно надавати унікальності разом з обмеженнями.
- Якщо задати обмеження UNIQUE для стовпця, то СУБД відхилить будь-яку спробу у певному рядку ввести у це поле значення, яке вже було в іншому рядку.
- UNIQUE може застосовуватись лише до полів, які визначені як NOT NULL.

Обмеження унікальності UNIQUE

- Приклад 1 (продовж.): удосконалимо інструкцію створення таблиці Salers через надання обмеження унікальності стовпцям:

```
CREATE TABLE Salers
(   snum integer NOT NULL UNIQUE,
    sname char(20) NOT NULL UNIQUE,
    city char(10),
    comm decimal );
```

Обмеження унікальності UNIQUE

- Приклад 2: У нашій базі даних кожного замовника обслуговує лише один продавець. Це означає, що кожна комбінація номера замовника (cnum) та номера продавця (snum) у таблиці Customers повинна бути унікальною. Отже, створимо таблицю Customers через надання обмеження унікальності групі стовпців:

```
CREATE TABLE Customers
 (   cnum integer NOT NULL,
     cname char(20) NOT NULL,
     city char(10),
     rating integer,
     snum integer NOT NULL,
   CONSTRAINT MyUniqueConstr UNIQUE (cnum,snum));
```

Обмеження первинного ключа PRIMARY KEY

- Найбільш використовуване обмеження
- Забезпечує визначення первинного ключа у таблиці
- PRIMARY KEY – це поєднання обмежень NOT NULL і UNIQUE
- В таблиці допускається лише одне обмеження PRIMARY KEY

Обмеження первинного ключа PRIMARY KEY

- Приклад 1 (продовж.): удосконалимо інструкцію створення таблиці Salers через надання обмеження первинного ключа:

```
CREATE TABLE Salers
(   snum integer NOT NULL PRIMARY KEY,
    sname char(20) NOT NULL UNIQUE,
    city char(10),
    comm decimal);
```

Обмеження первинного ключа PRIMARY KEY

- Приклад 3. Створимо таблицю Customers зі складеним первинним ключем через надання обмеження первинного ключа групі стовпців:

```
CREATE TABLE Customers
     (   cnum integer NOT NULL,
         cname char(20) NOT NULL,
         city char(10),
         rating integer,
         snum integer NOT NULL,
       CONSTRAINT MyPKConstr PRIMARY KEY
    (cnum,snum));
```

Обмеження зовнішнього ключа FOREIGN KEY

- Забезпечують зв'язок між таблицями БД
- Забезпечують правило цілісності посилань
- В таблиці можуть бути декілька обмежень FOREIGN KEY
- Задається:

  CONSTRAINT <ім’я обмеж>

  FOREIGN KEY (<ім’я стовп дочірн_табл>)

  REFERENCES <ім’я батьк_табл>(<ім’я первинного ключа батьк_табл>

Обмеження зовнішнього ключа FOREIGN KEY

- Приклад 4. Створимо таблицю Customers з первинним ключем cnum і зовнішнім ключем snum:

```
CREATE TABLE Customers
     (   cnum integer NOT NULL PRIMARY KEY,
         cname char(20) NOT NULL,
         city char(10),
         rating integer,
       CONSTRAINT CustomFKConstr
         FOREIGN KEY (snum)
         REFERENCES Salers (snum)
     );
```

Обмеження CHECK

- Задає умову, яка повинна перевірятись при внесенні даних
- Забезпечує обмеження доменів
- Умовою CHECK може будь-яка умова SQL (по аналогії з умовами фрази WHERE)
- Обмеження CHECK може використовуватись з обмеженням NOT NULL

Обмеження CHECK

- Приклад 1 (продовж.): удосконалимо інструкцію створення таблиці Salers через перевірку комісійних, які не повинні бути менші 5%:

```
CREATE TABLE Salers
(   snum integer NOT NULL PRIMARY KEY,
    sname char(20) NOT NULL UNIQUE,
    city char(10),
    comm decimal CHECK (comm>=.05));
```

Встановлення значень стовпців за замовчуванням

- NULL значення є найбільш поширеним значенням стовпця за замовчуванням
- Інша альтернатива, щоб не використовувати значення NULL, це встановити значення нуль (0) для числових полів або пропуски для символьних. У такому випадку під час запитів SQL буде обробляти їх як і будь-яке інше значення.
- Щоб надати значення за замовчуванням, використовують слово DEFAULT у команді CREATE TABLE.

Встановлення значень стовпців за замовчуванням

- Приклад 1 (продовж.) Припустимо, що основна філія підприємства знаходиться в місті London. Удосконалимо інструкцію створення таблиці Salers через задання значень за замовчуванням у стопці city:

```
CREATE TABLE Salers
(   snum integer NOT NULL PRIMARY KEY,
    sname char(20) NOT NULL UNIQUE,
    city char(10) DEFAULT='London',
    comm decimal CHECK (comm>=.05));
```

Модифікація структури таблиці. Інструкція ALTER TABLE

- Змінює структуру існуючої таблиці
- Стандарт ANSI SQL не дозволяє змінювати тип даних існуючого стовпця, однак багато СУБД розширяють цю версію ALTER TABLE, і дозволяють змінювати тип стовпця
- Зміна імені таблиці:

ALTER TABLE <ім’я табл>

RENAME TO <нове ім’я табл>;

Інструкція ALTER TABLE

- Додавання нового стовпця:

ALTER TABLE <ім’я табл>
    ADD COLUMN <ім’я стовп> <тип(розмір)>;

- Видалення стовпця з таблиці

ALTER TABLE <ім’я табл>
    DROP COLUMN <ім’я стовп>;

  - При цьому видаляються усі дані стовпця.
  - Якщо стовпець є первинним ключем, на який посилається зовнішній ключ з іншої таблиці, то видалення не відбудеться. Спочатку потрібно скоректувати дані в іншій таблиці.

Інструкція ALTER TABLE

- Зміна параметрів існуючого стовпця, таких як тип, розмір, обмеження:

ALTER TABLE <ім’я табл>

ALTER COLUMN <ім’я стовп> <тип(розмір)> <обмеження>;

- Приклад 5. Припустимо, що при створенні таблиці Salers поле sname не містить ніяких обмежень, які треба встановити. Крім цього, треба збільшити розмір цього поля.

ALTER TABLE Salers

ALTER COLUMN sname char(30) NOT NULL;

Інструкція ALTER TABLE

- Встановлення обмеження.

Приклад 6. Припустимо, що таблиця Customers не має ніяких обмежень таблиці. Необхідно встановити обмеження з прикладу 2.

```
ALTER TABLE Customers
    ADD CONSTRAINT MyUniqueConstr
    UNIQUE (cnum,snum);
```

- Видалення обмеження:

```
ALTER TABLE Customers
    DROP CONSTRAINT MyUniqueConstr;
```

Видалення таблиці

- Щоб видалити таблицю, необхідно мати права для цього.
- Перед видаленням таблиці необхідно видалити усе вмістиме таблиці (Вимога більшості СУБД).
- Якщо таблиця уже порожня, то видалення виконується інструкцією:

  DROP TABLE <ім’я табл>;
- Після такої інструкції ім’я таблиці більше не розпізнається, і ніяка інструкція не може бути застосована до цієї таблиці.
- Тому перш ніж вилучати таблицю, необхідно переконатись, що в неї не було зовнішнього ключа. Інакше, загубляться певні зв’язки.

Мова маніпулювання даними (DML – Data Manipulation Language)

- Інструкції DML не повертають дані у вигляді тимчасових результатних таблиць, а змінюють вмістиме вже існуючих таблиць бази даних.
- Інструкції модифікації даних можуть вміщати вкладені запити на вибірку даних з тієї ж таблиці або з інших таблиць, однак самі **не можуть бути вкладені** в інші запити
  - INSERT
  - UPDATE
  - DELETE

Інструкція внесення даних (INSERT)

- Найпростіший синтаксис :

  INSERT INTO <ім’я_табл>

  VALUES (<зн.стовп1>,...,<зн.стовпN>);
- Ім’я таблиці має бути визначеним командою CREATE TABLE.
- Кожне значення, що вводиться, має співпадати з типом даних стовпця, в який воно вставляється.
- Інструкція не виводить результатів, але СУБД повинна видати підтвердження того, що дані були введені.
- Якщо необхідно ввести невизначене значення, то у відповідне поле вводиться NULL.

Інструкція внесення даних (INSERT)

- Приклад 7: щоб ввести перший рядок у таблицю Salers, можна використати команду:

```
INSERT INTO Salers
    VALUES (1001,‘Peel’,‘London’,.12);
```

- Можна вказувати конкретні стовпці, в які потрібно вставити значення. Це дозволяє вставляти дані в довільному порядку:

```
INSERT INTO Salers (snum,sname,comm)
    VALUES (1001,‘Peel’,.12);
```

Інструкція внесення даних (INSERT)

- Починаючи з SQL-92, з'явилась можливість працювати зі значеннями типу запис.
- Це дозволяє за фразою VALUES вказувати декілька наборів значень у дужках, які необхідно вставити. Приклад 8:

```
INSERT INTO Customers (city,cname,cnum)
      VALUES ('London','Hoffman',2001),
                  ('Rome','Giovanni',2002),
                  ('SanJose','Liu',2003);
```

  - Поля rating та snum не вказані. Це означає, що дані стовпці встановлені у значення «за замовчуванням».

Інструкція внесення даних (INSERT)

- Команда INSERT також використовується, щоб вибирати значення з однієї таблиці і поміщати їх в іншу за допомогою запиту.
- У такому випадку фраза VALUES міняється на запит SELECT.
- Приклад 9:

```
INSERT INTO Londonstaff
  SELECT *  FROM Customers WHERE
city='London';
```

  - Щоб ця інструкція працювала, необхідно, щоб таблиця Londonstaff вже була створена і мала ті ж типи полів, що й таблиця Salers, а імена полів не обов'язково мають співпадати.

Інструкція внесення даних (INSERT)

- Приклад 10. Необхідно сформувати нову таблицю Daytotals, в якій буде інформація про кошти, які надходять кожен день. Такі дані можна вести незалежно від таблиці Orders, але спочатку потрібно заповнити таблицю Daytotals інформацією, яка вже міститься у таблиці Orders:

```
INSERT INTO Daytotals (ddate,total)
  SELECT odate, SUM(amt)
    FROM Orders
    GROUP BY odate;
```

Інструкція видалення записів (DELETE)

- Інструкція видаляє цілі рядки таблиці, а не індивідуальні значення полів.
- Наприклад, щоб видалити все вмістиме таблиці Salers, можна ввести команду:

  DELETE FROM Salers;
- Якщо необхідно видалити певні рядки з таблиці, використовують фразу WHERE.
- Наприклад, щоб видалити продавця Axelrod, можна ввести:

  DELETE FROM Salers WHERE snum=1003;
- Тут вказано поле snum, а не поле sname, оскільки використання первинних ключів надійно забезпечує видалення лише одного рядка.
- Це можна забезпечити і через підзапит, якщо snum невідоме (виконати самостійно).

Інструкція зміни даних (UPDATE)

- Інструкція UPDATE змінює значення стовпців.
- Щоб змінити значення в одному стовпці таблиці, використовується синтаксис:

  UPDATE <ім’я таблиці>

  SET <ім’я стовпця>=<значення>;
- Приклад 11. Замінити рейтинг усіх замовників на 200:

  UPDATE Customers SET rating=200;

Інструкція зміни даних (UPDATE)

- Якщо потрібно змінити значення стовпця у певному рядку, то використовується фраза WHERE.
- Приклад 12. Змінити рейтинг на 200 усіх замовників для продавця Peel:

```
UPDATE Customers  SET rating=200
  WHERE snum=1001;
```

  - Через підзапит (snum - невідоме) – самостійно

Інструкція зміни даних (UPDATE). Правила використання фрази SET

- В одній команді можна модифікувати лише одну таблицю, оскільки вираз SET не сприймає префіксів.
- У фразі SET можна використовувати вирази. Приклад 13. Продавцям у Лондоні збільшити їх комісійні у два рази:

```
UPDATE Salers  SET comm=comm*2
  WHERE city='London';
```

- Вираз SET не є предикатом. Тому треба вводити NULL значення без синтаксису предикату.

  Приклад 14. Встановити рейтинг замовників у Лондоні в NULL:

```
UPDATE Customers  SET rating=NULL
  WHERE city=’London’;
```

Дякую за увагу