



MOBA SQL

д.е.н., профессор
Ставицкий А.В.



План

- Загальні поняття
- Оператори маніпуляції даними
- Оператори опису даних



1. Загальні поняття



SQL

- За допомогою запитів здійснюється відбір і сортування інформації, вони використовуються для побудови звітів та форм. Запити Access реалізуються через інструкції мови структурованих запитів Structured Query Language (SQL), які можна змінювати в процесі роботи.



Режим SQL

- При побудові запиту в вікні конструктора Access працює у фоновому режимі, записуючи еквівалентні інструкції SQL. Для перегляду програми SQL необхідно переключитися в меню

Конструктор-Режим-Режим SQL



Запис операторів

- Формат запису операторів SQL вільний. Можна писати усе підряд на одному рядку, один оператор на декількох рядках, слова операторів можна розділяти довільною кількістю пробілів і коментарів. Закінчення операторів визначається по контексту. Компілятору мови байдуже, великими або маленькими буквами пишуться оператори.



Зарезервовані слова

- Як і в інших мовах, весь набір ключових слів мови SQL зарезервований, їх не можна займати для інших цілей (на імена об'єктів і змінних SQL).



Коментарі

- У випадку написання складної команди іноді додаються коментарі, які позначаються знаками „{” та „}”, або знаком „--” (два знаки мінус) до кінця рядка.



Обмеження

- Слід враховувати обмеження на довжину назв змінних: ім'я бази даних повинно бути не довше 10 символів, імена інших об'єктів SQL - таблиць, стовпчиків, view(псевдотаблиць), синонімів - не довше 18 символів.



Групи операторів

- Оператори маніпуляції даними
 - INSERT
 - DELETE
 - SELECT
 - UPDATE
- Оператори опису даних
 - CREATE
 - DROP
 - ALTER



2. Оператори маніпуляції даними



База даних

- Для прикладів обрано базу даних SQL_EX.

Студенти							
ID	Прізвище	Ім'я	Дата народ	Місто	Курс	Стипендія	Учасник клу
1	Петровський	Іван	10.10.1985	Дніпропетров	3	90,00 грн.	<input checked="" type="checkbox"/>
0	Петровський	Іван	10.10.1985	Дніпропетров	3	90,00 грн.	<input checked="" type="checkbox"/>
1	Петров	Андрій	16.01.1900	Київ	2	119,00 грн.	<input type="checkbox"/>
2	Квашнін	Семен	17.01.1900	Чернігів	2	81,00 грн.	<input type="checkbox"/>
5	Петровський	Іван	10.10.1985	Дніпропетров	3	90,00 грн.	<input checked="" type="checkbox"/>
6	Петровський	Іван	10.10.1985	Дніпропетров	3	29,00 грн.	<input checked="" type="checkbox"/>
8	Петров	Андрій	16.01.1900	Київ	1	85,00 грн.	<input type="checkbox"/>
9	Квашнін	Семен	17.01.1900	Чернігів	1	85,00 грн.	<input type="checkbox"/>
10	Петровський	Іван	10.10.1985	Дніпропетров	1	85,00 грн.	<input type="checkbox"/>

Курсові роботи				
ID	СтудентID	Назва	Керівник	Click to Add
1	1	Прогнозуванн	Семшов В.Ф.	
*(New)	0			

Студенти				
ID	Прізвище	Ім'я	Дата народ	Місто
Нові студенти				
1	Петровський	Іван	10.10.1985	Дніпро
0	Петровський	Іван	10.10.1985	Дніпро
1	Петров	Андрій	16.01.1900	Київ
2	Квашнін	Семен	17.01.1900	Чернігів
5	Петровський	Іван	10.10.1985	Дніпро
6	Петровський	Іван	10.10.1985	Дніпро
*(New)				



Оператори маніпуляції даними

До неї входять оператори:

- вибору (SELECT) рядків із таблиці (або таблиць),
- знищення (DELETE) рядків у таблиці,
- вставки (INSERT) рядків,
- зміни (UPDATE) значень в існуючих у таблиці рядках.



Оператор SELECT

- Найуживанішим оператором є оператор вибору, в якому вказується, які саме поля потрібно вивести. Таблиця, з якої виводяться дані вказується після службового слова FROM.



Приклад – 1

- Приклад вибирає всі рядки з таблиці Студенти і всі стовпчики:

```
SELECT *
```

```
FROM Студенти
```



Приклад – 2

- Приклад знаходить у таблиці Студенти рядки, у якому стовпчик Курс=3. З цього рядка беруться тільки три вказаних стовпчики.

```
SELECT Прізвище, Ім'я, Місто
```

```
FROM Студенти
```

```
WHERE Курс=3
```

- Зверніть увагу, що умова вибору була вказана за службовим словом WHERE.



Приклад – 3

- Приклад вибирає прізвища студентів, міста їхнього проживання з таблиці Студенти, а прізвища наукових керівників – із таблиці Курсові роботи:

```
SELECT Студенти.Прізвище, Студенти.[Ім'я], Студенти.Місто,  
[Курсові роботи].Керівник
```

```
FROM Студенти, [Курсові роботи]
```

```
WHERE Студенти.ID=[Курсові роботи].ID
```



Додаткові параметри оператора SELECT

- SQL дозволяє вибирати з усіх необхідних записів лише деяку частину. Додаткові параметри оператора SELECT керують подібним виводом.



ALL

- Вибирає всі записи, що задовольняють умові в інструкції SQL. Цей аргумент використовується, якщо не вказані жодні параметри.



DISTINCT

- Вибирає тільки ті записи, всі значення яких у полях, перерахованих в інструкції SELECT, є унікальними. Комбінація всіх обраних полів має бути унікальною. Наприклад, якщо два студенти мають однакові прізвища, то буде виведено тільки один запис.



DISTINCTROW

- Пропускає дубльовані записи, всі поля яких співпадають, а не одне обране поле.



TOP n

- Повертає визначене число записів зверху чи низу впорядкованого списку. Замість n підставляється необхідне число записів.



TOP n PERCENT

- Повертає визначений відсоток записів зверху чи низу впорядкованого списку. Замість n підставляється необхідний відсоток записів.



Створення нових полів

- SQL дозволяє створювати нові поля, які комбінують інформацію з інших полів. Наступний приклад на виводить номер студента та в одному полі ПІБ його прізвище та ім'я.



Приклад – 4

```
SELECT Студенти.ID, Студенти.Прізвище & " " & Студенти.[Ім'я] AS  
ПІБ
```

```
FROM Студенти
```



Оператор DELETE

- Оператор DELETE знищує записи, які задовольняють певній умові.



Приклад – 5

- Приклад знищує в таблиці Студенти усі рядки, у яких номер курсу дорівнює 5:

```
DELETE
```

```
FROM Студенти
```

```
WHERE Курс=5
```



Приклад – 6

- Така інструкція знищить всі рядки в таблиці Студенти, але не саму таблицю:

```
DELETE
```

```
FROM Студенти
```



Оператор INSERT

- Оператор INSERT вставляє до таблиці один або декілька рядків.



Приклад – 7

- Приклад дозволяє вставити до таблиці Студенти запис про нового студента:

```
INSERT INTO Студенти
```

```
VALUES (6, "Сидорчук", "Павло", "7.3.1988", "Харків", 2, 120, 0)
```



Приклад – 8

- Якщо використовувати складений вираз, то оператор INSERT INTO може вставити цілий набір, обраних підзапитом SELECT з іншої таблиці записів з іншої таблиці.

```
INSERT INTO Студенти ([Прізвище], [Ім'я], [Дата народження],  
[Місто], [Курс], [Стипендія], [Учасник клубу])  
SELECT [Прізвище], [Ім'я], [Дата народження], [Місто], 1, 85, 0  
FROM [Нові студенти]  
WHERE [Прізвище] IS NOT NULL
```



Приклад – 8А

- Наведений приклад додає всі непусті записи з таблиці Нові студенти до таблиці Студенти. Оскільки перша таблиця не містить деяких полів, то дані вказуються явно: новий студент буде вчитися на першому курсі, отримуючи стипендію – 85 грн., не відвідуючи клуб.
- Оскільки не вказано поле лічильника, то нові записи пронумеруються автоматично.
- Якщо не у всі поля рядка, що вставляються, вноситься значення (як це зроблено в прикладі), то незаповнені стовпчики будуть містити значення NULL.



Оператор UPDATE

- Оператор UPDATE змінює значення полів у рядках, що задовольняють певній умові.



Приклад – 9

- Наприклад, потрібно змінити прізвище дівчини „Смалько”, що вийшла заміж, на „Лотова”:

```
UPDATE Студенти
```

```
SET Прізвище="Лотова"
```

```
WHERE Прізвище="Смалько"
```



Приклад – 10

- Наступний приклад вводить до складу клубу всіх студентів з бази Студенти, які навчаються з третього по п'ятий курси.

```
UPDATE Студенти
```

```
SET [Учасник клубу]=True
```

```
WHERE [Курс] BETWEEN 3 AND 5
```



Пропозиція WHERE

- Пропозиція WHERE може бути присутньою у будь-якому з операторів DELETE, UPDATE, SELECT, коли потрібно задати умови на записи, що потрібно опрацювати (відповідно, знищити, змінити або вибрати).



Синтаксис пропозиції WHERE – 1

- У пропозиції WHERE пишеться логічна умова, що отримується за допомогою логічних операторів AND, OR і NOT:
 - вираз1 < вираз2,
 - вираз1 >= вираз2, і т.п.,
 - назва_поля IS [NOT] NULL
 - вираз [NOT] BETWEEN вираз1 AND вираз2
 - вираз [NOT] IN (вираз1 , ... [, ...])



Синтаксис пропозиції WHERE – 2

- Можна з'ясувати, чи підходить символний рядок під визначений шаблон. Для цього використовується операція порівняння по шаблону – LIKE:

символьний-вираз LIKE "шаблон"



Синтаксис пропозиції WHERE – 3

- У операторі LIKE використовуються тільки два спеціальних символи:
 - * - заміщує довільну кількість символів,
 - ? - заміщує рівно один символ.



Приклад – 11

- Припустимо, що потрібно вибрати з таблиці Студенти усі прізвища, у що містять букву „і”, а передостання літера в ньому - "к". Оператор для вибірки буде виглядати так:

```
SELECT *
```

```
FROM Студенти
```

```
WHERE (((Студенти.Прізвище) Like "*і*к?"))
```




Умови з підзапитом

- Умови з підзапитом застосовуються для:
 - порівняння виразу з результатом іншого SELECT оператора
 - визначення належності виразу результатам іншого SELECT оператора
 - з'ясування порожності множини іншого SELECT оператора.



Приклад – 12

- В наступному прикладі підзапит повертає єдине значення – максимальний розмір стипендії, а зовнішній SELECT оператор знаходить прізвища її володарів.

```
SELECT Прізвище
```

```
FROM Студенти
```

```
WHERE Стипендія=
```

```
(SELECT MAX(Стипендія) FROM Студенти )
```



Приклад – 13

- Цей приклад виводить всю інформацію про нових студентів з тих міст, представники яких входять до клубу.

```
SELECT *
```

```
FROM [Нові студенти]
```

```
WHERE Прізвище is not NULL and Місто in (SELECT Місто FROM  
Студенти WHERE [Учасник клубу]=True)
```



Приклад – 14

- Якщо потрібно вивести всі дані про студентів, які є учасниками клубу і отримують стипендію щонайменше вдвічі більшу за найбільшого учасника клубу, то слід скористатися таким виразом:

```
SELECT *
```

```
FROM Студенти
```

```
WHERE ([Учасник клубу]) AND (Стипендія>(SELECT 2*MIN(Стипендія)  
FROM Студенти WHERE [Учасник клубу]=True))
```



Параметр ORDER BY – 1

- Для сортування записів існує параметр ORDER BY, який визначає поля, за якими слід провести сортування.



Приклад – 15

- В цьому прикладі записи таблиці Студенти будуть відсортовані спочатку за полем Прізвище, а при співпаданні прізвищ студентів – по імені.

```
SELECT Прізвище
```

```
FROM Студенти
```

```
ORDER BY Прізвище, Ім'я
```



Параметр ORDER BY – 2

- Необов'язково вказувати назви всіх полів, можна перерахувати відповідні номери стовпчиків, оскільки у операторі ORDER BY замість імені стовпчика можна вказувати його порядковий номер у списку вибірки:

```
SELECT Прізвище
```

```
FROM Студенти
```

```
ORDER BY 2, 3
```



Агрегатні функції

- Для знаходження числових характеристик відповідних стовпчиків використовуються агрегатні функції.



Список агрегатних функцій

AVG(стовпчик)	Знаходить середнє значення в стовпчику
COUNT(*)	Знаходить кількість записів, що задовольняють умові
FIRST(стовпчик)	Повертає перший рядок заданого стовпчика
LAST(стовпчик)	Повертає останній рядок заданого стовпчика
MAX(стовпчик)	Знаходить максимальне значення в стовпчику
MIN(стовпчик)	Знаходить мінімальне значення в стовпчику
STDEV(стовпчик)	Повертає середньоквадратичне відхилення у стовпчику
SUM(стовпчик)	Повертає суму всіх значень у стовпчику
VAR(стовпчик)	Повертає дисперсію стовпчика



Приклад – 16

- Приклад знаходить середню стипендію студентів 2 курсу:

```
SELECT Avg(Стипендія)
```

```
FROM Студенти
```

```
WHERE Курс=2
```



Приклад – 17

- Якщо потрібно, можна дати назву вихідному полю. Наступний приклад знаходить кількість студентів 3 курсу і вміщує отримане число у поле Кількість:

```
SELECT Count(*) AS Кількість
```

```
FROM Студенти
```

```
WHERE Курс=3
```



Параметр GROUP BY

- Групові операції підтримуються параметром GROUP BY. Нехай потрібно вивести кількість студентів для кожного з курсів. В такому випадку необхідно вказати груповим полем Курс:

```
SELECT Курс, Count(*) AS Кількість
```

```
FROM Студенти
```

```
GROUP BY Курс
```



Приклад – 18

- Виведемо кількість студентів кожного курсу, середню стипендію на кожному курсі з середньоквадратичним відхиленням:

```
SELECT Курс, Count(*) AS Кількість, AVG(Стипендія) AS [Середня  
стипендія], StDev(Стипендія) AS Відхилення
```

```
FROM Студенти
```

```
GROUP BY Курс
```



Пропозиція HAVING

- Пропозиція HAVING накладає додаткові умови на групу.



Приклад

- Цей запит повертає назви міст і середні значення стипендії студентів, якщо місто представлено більше, ніж 2 студентами.

```
SELECT Місто, Avg(Стипендія) AS [Середня Стипендія]  
FROM Студенти  
GROUP BY Місто HAVING (((Count(*))>2))
```



Об'єднання об'єктів – 1

- При створенні таблиці Access автоматично встановлює внутрішнє об'єднання, що не завжди зручно при побудові запитів. Мова SQL дозволяє змінювати параметри об'єднання. Для цього вказується ключове слово JOIN у інструкції FROM:

FROM таблиці та запити [{INNER | LEFT | RIGHT} JOIN таблиця ON умова]



Об'єднання об'єктів – 2

- Оператор порівняння (умова) не обов'язково має бути знаком рівності. Можна ще використовувати знаки менше „<”, більше „>”, менше або дорівнює „<=”, більше або дорівнює „>=”, не дорівнює „<>”.
- Можна також зв'язати декілька операторі ON з оператором JOIN за допомогою ключових слів AND та OR.



Об'єднання об'єктів – 3

- SQL дозволяє вставляти всі типи об'єднань в INNER JOIN, проте неможливо вкласти INNER JOIN у зовнішнє об'єднання LEFT JOIN або RIGHT JOIN.



Приклад – 19

- Приклад виводить дані про всіх студентів, порядкові номери яких співпадають.

```
SELECT *
```

```
FROM [Студенти] INNER JOIN [Нові студенти] ON  
([Студенти].ID=[Нові студенти].ID)
```



Об'єднання об'єктів – 4

- Операції правого та лівого зовнішніх об'єднань вказують на послідовність перерахування таблиць в пропозиції JOIN. У деяких випадках це корисно, коли існують головна таблиця і допоміжна, а дані з головної таблиці потрібно одержати в будь-якому випадку.



Приклад – 20

- В цьому прикладі результатом запиту будуть всі записи таблиці Студенти. Для тих студентів, які повторюються в таблиці Нові студенти, буде додана інформації з цієї таблиці.

```
SELECT *
```

```
FROM [Студенти] LEFT JOIN [Нові студенти] ON ([Студенти].ID=[Нові  
студенти].ID)
```



Приклад – 21

- В цьому прикладі результатом запиту будуть всі записи таблиці Нові студенти. Для тих студентів, які повторюються в таблиці Студенти, буде додана інформації з цієї таблиці.

```
SELECT *
```

```
FROM [Студенти] RIGHT JOIN [Нові студенти] ON  
([Студенти].ID=[Нові студенти].ID)
```



Приклад – 22

- Іноді потрібно помістити результат запиту у окрему таблицю. Для цього вказується службове слово INTO. Приклад створює нову таблицю x, в яку виводяться лише прізвища всіх студентів з таблиці Студенти.

```
SELECT Студенти.Прізвище INTO x  
FROM Студенти
```



3. Оператори опису даних



Оператори опису даних

- Оператори опису даних призначені для створення опису, зміни опису і знищення об'єктів бази даних.



Види об'єктів SQL

- база даних (database)
- таблиця (table)
- стовпчик (column)
- індекс (index)
- знімок (view)
- синонім (synonym)



Ідентифікатори – 1

- Кожний об'єкт має власне ім'я – ідентифікатор, а також власника – користувача, що його створив. Якщо в якомусь запиті або операторі SQL згадуються два стовпчики з однаковими назвами, то їх потрібно уточнювати повними іменами таблиць, що їх містять.
- Перед ім'ям будь-якого об'єкта можна (а іноді і необхідно) зазначити ім'я його власника (owner-name) - вхідне ім'я користувача, який створив (CREATE) цей об'єкт.



Ідентифікатори – 2

- Студенти. Прізвище - стовпчик Прізвище із таблиці Студенти
- [Нові студенти]. Прізвище - стовпчик Прізвище із таблиці Нові студенти
- Петренко. Студенти. Прізвище - стовпчик Прізвище із таблиці Студенти, власником якої є Петренко



Оператор CREATE TABLE

- Створення таблиці здійснюється за допомогою оператора **CREATE TABLE**. Аргументами цього оператора є назви необхідних полів з вказуванням їхніх типів.



Приклад – 23

- Створимо таблицю Му та проведемо індексацію по ключовому полю ID.



Приклад – 23А

```
CREATE TABLE My (  
  [ID] Counter,  
  [вік] INT,  
  [прізвище] CHAR(20) UNIQUE,  
  [ім'я] CHAR(15),  
  [стипендія] MONEY,  
  [код] LONG,  
  [Рост] FLOAT,  
  [Вага] REAL,
```

```
  [Курс] BYTE,  
  [дата народження] DATE,  
  [дата поїздки] DATETIME,  
  [Час початку] TIME,  
  [Клуб] LOGICAL,  
  [Фото] Image,  
  [Адреса] TEXT(20),  
  [Примітки] MEMO,  
  CONSTRAINT [Index1] PRIMARY KEY  
    ([ID]))
```



Типи змінних та полів

- Мова SQL розвивалася і змінювалася довгий час, тому вона підтримує багато різних варіантів типів змінних та їх підтипів. Зокрема, типи CHAR та TEXT співпадають. Розглянемо можливі типи.



Текстовий

- CHAR або TEXT - текстові вирази довжиною до 255 байт (якщо в дужках вказано число, то воно визначає довжину поля)



ЧИСЛОВИЙ

- BYTE - ціле (1 байт)
- INT - ціле (2 байти)
- LONG - ціле (4 байти)
- FLOAT - дійсне одинарної точності з плаваючою точкою (8 байт)
- REAL - дійсне подвійної точності з плаваючою точкою (8 байт)



Часовий

- DATE - дата (8 байт)
- TIME - час (8 байт)
- DATETIME - дата та час одночасно (8 байт)



Грошовий

- MONEY - грошові значення та числові дані розміром 8 байт (до 15 знаків перед і від 1 до 4 знаків після коми)



Лічильник

- COUNTER - унікальні цілі числа розміром 4 байти (16 байт займають лише у випадку спеціального кодування)



Логічний

- LOGICAL - елементи типу „Так”–„Ні” розміром 1 байт



Поле MEMO

- MEMO - велике текстове поле розміром до 65535 байт



Об'єкт

- IMAGE або інші - посилання на довільний об'єкт, розмір залежить від розміру об'єкта



Оператор ALTER TABLE

- У вже існуючій таблиці ми можемо поміняти тип стовпчика, додати новий, знищити старий. Для цих функцій використовується оператор ALTER TABLE.



Приклад – 24

- Додамо до таблиці поле кафедра, яке містить текстові значення довжиною до 20 символів:

```
ALTER TABLE My
```

```
ADD [кафедра] CHAR(20)
```



Приклад – 25

- Для знищення полів Вік та кафедра слід задати команду:

```
ALTER TABLE My
```

```
DROP [вік], [кафедра]
```



Зауваження

- Зміна структури таблиці призводить до фізичного перетворення даних у ній. Якщо змінений тип стовпчика, то дані в ньому перетворюються до нового типу, і якщо це неможливо здійснити, то оператор ALTER видає помилку, а таблиця залишається у незмінному стані.



Оператор DROP

- Для знищення об'єктів використовується оператор DROP.



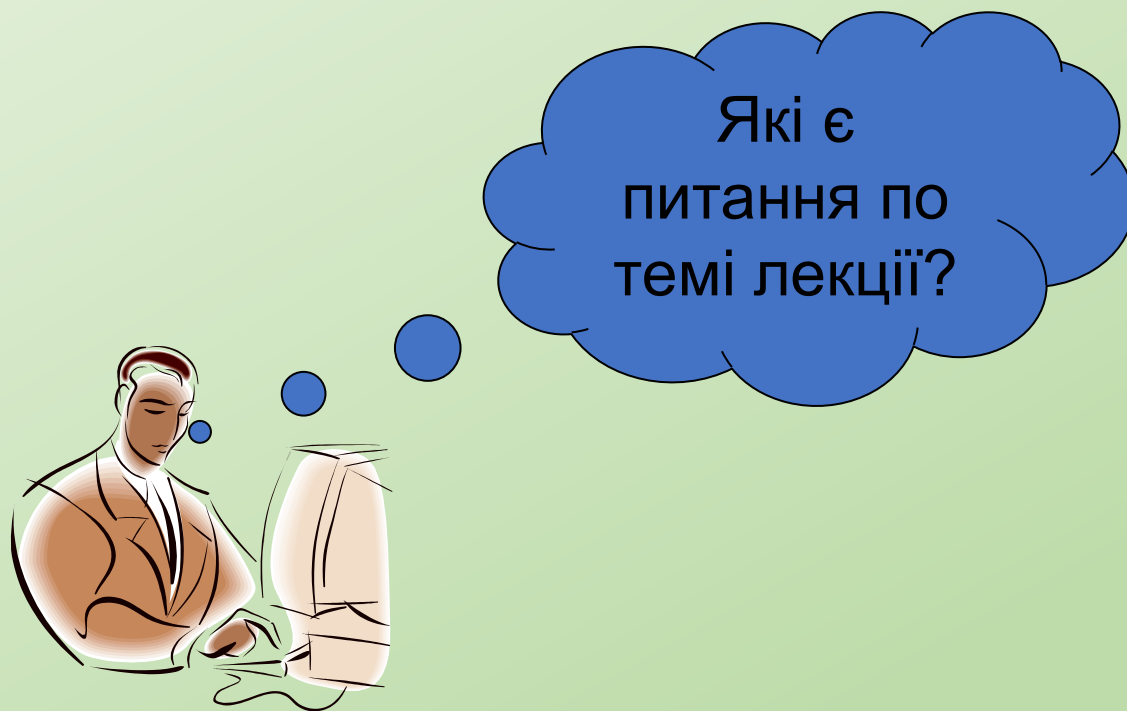
Приклад – 26

- Наступний приклад знищує таблиці My та x:

```
DROP TABLE my, x
```



Закінчення





Дякую за увагу!