МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ФАХОВИЙ КОЛЕДЖ ПРОМИСЛОВОЇ АВТОМАТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ОДЕСЬКОГО НАЦІОНАЛЬНОГО ТЕХНОЛОГІЧНОГО УНІВЕРСИТЕТУ»

ЗАТВЕРДЖУЮ
Заступник директора з
навчально-методичної роботи
ФКПАІТ ОНТУ
_____ Вікторія ОКСАНІЧЕНКО
_____._____.20___ року

Методичні вказівки для самостійної роботи студентів

СИСТЕМИ УПРАВЛІННЯ БАЗАМИ ДАНИХ
(шифр за ОПП і назва навчальної дисципліни)

галузь знань _____02 «Культура та мистецтво»_____
(шифр і назва галузі знань)

спеціальність_____029 «Інформаційна, бібліотечна та архівна справа»_____
(шифр і назва спеціальності)

освітня програма_____ Інформаційна, бібліотечна та архівна справа_____
(назва освітньої програми)

(Шифр за ОПП_____)

Одеса 2022

Методичні вказівки для самостійної роботи студентів з дисципліни «Системи управління базами даних» для підготовки фахового молодшого бакалавра за спеціальністю 029 «Інформаційна, бібліотечна та архівна справа».

Укладач: викладач вищої кваліфікаційної категорії ФКПАІТ ОНТУ Ю. В. Бурмакіна

Методичні вказівки для самостійної роботи студентів затверджені на засіданні циклової комісії «Комп'ютених наук та інженерії програмного забезпечення» ФКПАІТ ОНТУ
Протокол від ___._____.20___ року, № ____

Голова циклової комісії
_____Тетяна КОСТИРЕНКО
(підпис) (власне ім'я та прізвище)

___._____.20___ року

Пояснювальна записка

Самостійна робота студентів є однією з важливих складових освітнього процесу. Їй приділяється особлива увага так як під час її виконання формуються уміння освоювати знання без сторонньої допомоги.

У вітчизняній структурі освіти самостійній роботі студента відводиться багато часу та вона стоїть на рівні з такими видами занять як лекції, практичні і лабораторні роботи.

Методичні вказівки призначені для самостійної підготовки студентів, що вивчають бази даних. На самостійний розгляд виносяться теми, які доповнюють розглянутий на лекційних заняттях матеріал. Самостійна робота припускає опрацювання теоретичного матеріалу, відповіді на питання по кожній темі.

Студенти оформляють свою роботу у вигляді конспекту. Результати роботи можуть бути використані при виконанні практичних завдань та при написанні модульного контролю.

Форми перевірки виконання можуть бути найрізноманітніші: опитування, диктант, тест, захист практичної роботи.

Завдання з самостійної роботи видаються поступово протягом семестру в кінці відповідного лекційного чи практичного заняття.

Функції самостійної роботи:

- сприяє засвоєнню знань, формуванню професійних вмінь і навиків, забезпечує формування професійної компетенції;
- виховує потребу в самоосвіті, розвиває пізнавальні і творчі здібності;
- спонукає до науково-дослідницької роботи.

Види самостійної роботи:

- опрацювання навчального матеріалу;
- виконання завдань до практичних робіт;
- вирішення варіантних завдань та вправ;
- написання рефератів, докладів за різними темами.

Тема	Кількість годин
Компоненти середовища СУБД	2
Трирівнева архітектура системи управління базами даних	2
Головні переваги та недоліки ранніх СУБД.	2
Об'єктно-орієнтована модель даних	2
Фундаментальні властивості відношень в реляційній моделі даних	2
Дванадцять правил Кодда	2
Етапи проектування баз даних.	2
Аномалії оновлення в базі даних	2
Переваги та недоліки нормалізації	2
Етапи розробки бази даних.	2
Об’єкти бази даних в MS Access.	2
Типи даних в MS Access.	2
Встановлення зв’язків між таблицями в MS Access	2
Сортування та фільтрація даних.	2
Пошук записів в MS Access	2
Використання виразів та умов відбору в запитах	2
Розрахунки в запитах	2
Мова SQL для створення запитів.	2
Створення звітів різними способами	3
Моделі архітектури «Клієнт - сервер»	2
Відкриті системи	2
Всього годин	**45**

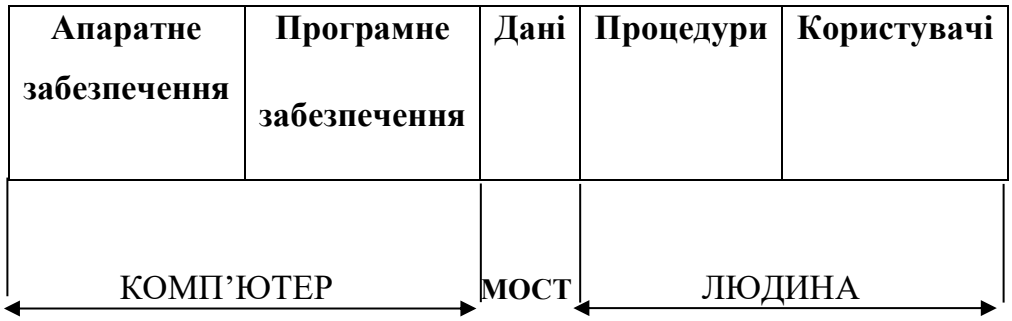
Тема «Компоненти середовища СУБД»

(2 години)

Мета: визначення головних компонентів середовища СУБД та ознайомлення з їх змістом.

В середовищі СУБД можна визначити 5 головних компонентів:

1. апаратне забезпечення;
2. програмне забезпечення;
3. дані;
4. процедури;
5. користувачі.



1. Апаратне забезпечення

Для роботи СУБД та додатку необхідне деяке апаратне забезпечення. Воно може змінюватися в широких межах – від одного ПК до мережі з декількох комп'ютерів. Апаратне забезпечення, яке використовується залежить від вимог даної організації та СУБД, яка використовується. Деякі СУБД призначені для роботи тільки з конкретними типами ОС чи обладнанням, інші мають змогу працювати з широким колом апаратного забезпечення та різними ОС.

2. Програмне забезпечення

Цей компонент охоплює програмне забезпечення самої СУБД та прикладних програм разом з ОС, охоплюючи мережеве програмне забезпечення, якщо СУБД використовується в мережі.

Звичайно, додатки створюються на мовах третього покоління. Таких як C, Cobol, Ada, Pascal чи на мовах четвертого покоління, таких як SQL, оператори якого впроваджуються до програми на мовах третього покоління. Втім, СУБД може мати свої особисті інструменти четвертого покоління, призначені для швидкої розробки додатків з використанням вбудованих непроцедурних мов запитів, генераторів звітів, форм, графічних зображень та навіть повномасштабних додатків.

3. Дані

Найбільш важливим компонентом середовища СУБД (з точки зору кінцевих користувачів) є дані, які грають роль мосту між комп'ютером та людиною. База даних містить як робочі дані, так і метадані, тобто **„данні про данні"**. Структура бази даних має назву **схема.**

4. Процедури

До процедур належать інструкції та правила, які повинні враховуватися при проектуванні та використанні БД.

Користувачам та обслуговуючому персоналу важливо надати документацію, яка містить досконалий опис процедур використання та супровід цієї системи, включаючи інструкції про правила виконання наступних дій:

- реєстрація в СУБД;
- використання деякого інструменту СУБД чи додатку;
- запуск та зупинка СУБД;
- створювання резервних копій СУБД;
- обробка збоїв апаратного та програмного забезпечення, включаючи процедури ідентифікації компоненту, який вийшов з ладу, виправлення відмовившого компоненту (наприклад, за допомогою виклику спеціаліста з ремонту апаратного забезпечення), а також відновлення бази даних після ліквідування несправностей;
- зміна структури таблиці, реорганізація бази даних, розташованої на деяких дисках, способи поліпшення продуктивності та методи архівації даних на другорядних пристроях зберігання.

5. Користувачі

Серед них можливо виділити 4 різних групи: адміністратори даних та баз даних, розробники БД, прикладні програмісти та кінцеві користувачі.

- **Адміністратори даних та баз даних**

База даних та СУБД є корпоративними ресурсами, якими слід керувати так само, як іншими ресурсами.

Адміністратор даних (АД) відповідає за керування даними, враховуючи планування бази даних, розробку та супровід стандартів, бізнес-правил та ділових процедур.

АД консультує та дає рекомендації керівнику найвищої ланки, контролює співвідношення спільного напрямку розвитку бази даних, встановленним корпоративній меті.

Адміністратор баз даних (АБД) відповідає за фізичну реалізацію бази даних, враховуючи фізичне проектування та втілення проекту, за забезпечення безпеки та цілісності

даних, за супровід ОС, а також за забезпечення максимальної продуктивності додатків та користувачів.

У порівнянні з АД, обов'язки АБД носять більш технічний характер, та для нього необхідне знання конкретної СКБД та системного оточування.

- **Розробники баз даних**

В проектуванні великих баз даних приймають участь два різних типи розробників: **розробники логічної бази даних** та **розробники фізичної бази даних**.

Розробник логічної бази даних займається ідентифікацією даних (тобто сутностей та її атрибутів), встановлює зв'язки між даними та обмеження, які накладаються на дані, які зберігаються.

Розробник фізичної бази даних отримує готову логічну модель даних та займається її фізичною реалізацією, у тому числі:

- перетворенням логічної моделі даних в набір таблиць та обмежень цілісності даних;
- обранням конкретних структур зберігання та методів доступу до даних, які забезпечують необхідний рівень продуктивності при роботі з базою даних;
- проектуванням різних вимагаємих засобів безпеки даних.

Розробник фізичної бази даних повинен розбиратися в функціональних можливостях кінцевої СУБД та розуміти переваги та недоліки кожного можливого варіанту втілення. Він повинен вміти обирати найбільш влаштовуючу стратегію зберігання даних за обліком усіх існуючих особливостей їх використання.

- **Прикладні програмісти**

Одразу після створення бази даних треба почати розробку додатків, які надають користувачам необхідні їм функціональні можливості. Саме цю роботу і виконують прикладні програмісти. Звичайно вони працюють на основі специфікацій, які створені системними аналітиками.

- **Кінцеві користувачі**

Користувачами є клієнти бази даних – вона проектується, створюється та підтримується для того, щоб обслуговувати їх інформаційні потреби.

Користувачів можна поділити за засобом використання ними системи:

1. **наївні користувачі** звичайно навіть і не підозрюють про присутність СУБД.

Вони звертаються до бази даних за допомогою спеціальних додатків, які дозволяють в максимальному ступені спростити виконання ними операції.

Такі користувачі ініціюють виконання операцій бази даних, виконуючі найпростіші команди чи обираючи пункти меню. Це значить, що таким користувачам не потрібно нічого розуміти про СУБД.

Наприклад, для того щоб дізнатися ціну товару, касир в супермаркеті використовує сканер для зчитування нанесеного на нього штрих-коду. В підсумку цієї простої дії спеціальна програма не тільки зчитує штрих-код, але й обирає на основі його значення ціну товару з бази даних, а також зменшує значення в іншому полі базі даних, який визначає залишок товарів на складі, після чого зображує ціну та загальну вартість на касовому апараті.

2. **користувачі з досвідом** знайомі зі структурою бази даних та можливостями СУБД. Для виконання вимагаємих операцій вони можуть використовувати таку мову запитів високого рівня, як SQL. А деякі користувачі з досвідом можуть створювати власні прикладні програми.

Питання щодо самоконтролю:

1. Які компоненти можна визначити в середовищі СУБД?
2. Який компонент середовища СУБД вважається найважливішим з точки зору користувачів?
3. Що таке „Метадані”?
4. Для чого призначена схема бази даних?
5. Які функції в середовищі СУБД виконує програмне забезпечення?
6. На які групи можна розподілити користувачів?
7. Чим відрізняються функції адміністратору даних від функцій адміністратору баз даних?
8. Чим займаються розробники баз даних?

Тема «Трирівнева архітектура системи управління базами даних»

(2 години)

Мета: впровадження ідентифікації трьох різних рівнів опису елементів даних, які формують архітектуру систем управління базами даних.

Найбільш фундаментальним моментом у звітах дослідницьких груп про архітектуру системи керування базами даних є ідентифікація трьох різних рівней опису елементів даних. Ці рівні формують трьохрівневу архітектуру, яка охоплює зовнішній, внутрішній та концептуальний рівні.

Рівень, на якому сприймають дані користувачі, називається ***зовнішнім рівнем***, тоді як система керування базами даних та операційна система сприймають дані на ***внутрішньому***

рівні. ***Концептуальний рівень*** зображення даних призначений для відображення зовнішнього рівня на внутрішньому та забезпечення необхідної незалежності одне від одного.

Зовнішній рівень – це зображення бази даних з точки зору користувачів. Цей рівень описує ту частину бази даних, яка належить до кожного користувача.

Зовнішнє зображення містить тільки ті сутності, атрибути та зв'язки «реального» світу, які цікаві користувачу, інші сутності, атрибути або зв'язки, які йому не цікаві, також можуть бути зображені в базі даних, але користувач може навіть не підозрювати про їх існування. Різні зображення можуть по-різному відобразити одні й ті ж дані. Наприклад, один користувач може продивлятися дати в форматі *день, місяць, рік*, а інший – в форматі *рік, місяць, день*.

Концептуальний рівень – узагальнене зображення бази даних. Цей рівень описує те, які дані зберігаються в базі даних, а також зв'язки, існуючі між ними. Фактично, це повне зображення вимог до даних з боку організації, які не залежать від будь-яких міркувань відносно способу їх зберігання.

На концептуальному рівні зображені такі компоненти:

- усі сутності, атрибути та зв'язки;
- обмеження на дані;
- семантична інформація про дані;
- інформація про заходи безпеки та підтримка цілісності.

Внутрішній рівень – фізичне зображення бази даних в комп'ютері. Цей рівень описує те, яка інформація зберігається в базі даних.

Внутрішній рівень описує фізичну реалізацію бази даних, призначену для досягнення оптимальної продуктивності та забезпечення економного використання дискового простору. Він містить опис структур даних та організації окремих файлів, які використовуються для зберігання даних в запам'ятовуючих пристроях.

На внутрішньому рівні зберігається така інформація:

- розподіл дискового простору для зберігання даних та індексів;
- опис подробиць зберігання записів (з вказівкою реальних розмірів зберігаємих елементів даних);
- відомості про розміщення записів;
- відомості про стиснення даних та обраних методах їх шифрування.

Мета трирівневої архітектури – це відокремлення користувацького зображення бази даних від її фізичного зображення.

Питання щодо самоконтролю:

1. Що є найбільш фундаментальним моментом у звітах дослідницьких груп про архітектуру СУБД?
2. Які рівні формують трирівневу архітектуру СУБД?
3. На якому рівні сприймають дані користувачі?
4. На якому рівні сприймають дані СУБД та операційна система?
5. Для чого призначений концептуальний рівень зображення даних?
6. Яка інформація міститься на кожному з рівнів архітектури СУБД?
7. Навіщо архітектура СУБД була поділена на три рівні?

Тема «Переваги та недоліки СУБД»

(2 години)

Мета: визначення переваг та недоліків кожної СУБД та ознайомлення з їх змістом.

СУБД володіють як потенціальними перевагами, так й недоліками.

Переваги СУБД:

1. контроль за надмірністю даних;
2. несуперечливість даних;
3. спільне використання даних;
4. підтримка цілісності даних;
5. підвищена безпека.

Контроль за надмірністю даних

При використані бази даних передбачається спроба вилучення надмірності даних за рахунок інтегрування файлів для того, щоб уникнути зберігання декількох копій того ж самого елементу інформації. Але повністю надмірність даних не виключається, а лише контролюється її ступінь. В одних випадках ключові елементи даних треба дублювати для моделювання зв’язків, а в інших випадках деякі дані треба буде продублювати з міркувань підвищення продуктивності системи.

Несуперечливість даних

Вилучення надмірності даних або контроль за нею дозволяють скоротити ризик виникнення суперечливих станів.

Якщо елемент даних зберігається в базі даних в одному примірнику, то для зміни його значення треба буде виконати лише одну операцію оновлення, до того ж нове значення стане припустимим одразу всім користувачам бази даних. А якщо цей елемент даних зберігається в базі даних в декількох примірниках, то така система має можливість стежити за тим, щоб копії не суперечили одна одній.

На жаль, в багатьох сучасних СУБД такий тип несуперечності даних автоматично не підтримується.

Спільне використання даних

Файли звичайно належать окремим особам або цілим відділам, які використовують їх в своїй роботі. В той же самий час, база даних належить всій організації в цілому та має можливість спільно використовуватися всіма зареєстрованими користувачами. При такій організації роботи більша кількість користувачів може працювати з великим об'ємом даних. До того ж, при цьому можна створювати нові додатки на підставі вже існуючої в базі даних інформації та додавати до неї тільки ті дані, які в теперішню мить ще не зберігаються в ній, а не визначати знов вимоги до всіх даних, які потрібні новому додатку.

Нові додатки можуть також використовувати такі функціональні можливості, як визначення структур даних та керування доступом до даних, організація паралельної обробки та забезпечення засобів копіювання / відновлення, виключаючи необхідність реалізації цих функцій зі свого боку.

Підтримка цілісності даних

Цілісність бази даних визначає коректність та несуперечливість зберігаємих в ній даних. Цілісність звичайно описується за допомогою обмежень, тобто правил підтримки несуперечливості, які не повинні порушуватися в базі даних. Обмеження можна пристосовувати до елементів даних усередині одного запису або до зв'язків поміж записами.

Наприклад, обмеження цілісності може казати, що заробітна плата співробітника на місяць не повинна перевищувати 50000 грн. або те, що в запису з даними про співробітника номер відділення, в якому він працює, повинен відповідати існуючому відділенню компанії.

Таким чином, інтеграція даних дозволяє адміністратору баз даних завдавати вимоги по підтримці цілісності даних, а СУБД використовувати їх.

Підвищена безпека

Безпека бази даних міститься в захисті бази даних від несанкціонованого доступу з боку користувачів. Без залучення відповідних заходів безпеки інтегровані дані стають найбільш уразливими, ніж дані в файловій системі.

Система забезпечення безпеки може бути висловлювана в формі обліку імен та паролів для ідентифікації користувачів, які зареєстровані в цій базі даних. Доступ до даних з боку зареєстрованого користувача може бути обмежений деякими операціями (видобуванням, додаванням, оновленням та знищенням). Наприклад, адміністратору баз даних може бути надано право доступу до всіх даних бази даних, менеджеру відділення компанії – до всіх даних, які належать до його відділення, а інспектору відділу реалізації – тільки до даних про нерухомість, в результаті цього він немає доступу до конфіденційних даних, наприклад, заробітної плати співробітників компанії.

Недоліки СУБД:

1. важкість;
2. розмір;
3. вартість СУБД;
4. витрати на перетворення;
5. продуктивність.

Важкість

Забезпечення функціональності, якою повинна володіти кожна добра СУБД, супроводжується значним ускладнюванням програмного забезпечення СУБД.

Для того, щоб скористатися всіма перевагами СУБД, проектувальники та розробники баз даних, адміністратори даних та адміністратори баз даних, а також кінцеві користувачі повинні добре розуміти функціональні можливості СУБД. Нерозуміння принципів роботи системи може призвести до невдалих результатів проектування, що буде мати дуже серьйозні наслідки для всієї організації.

Розмір

Важкість та широчінь функціональних можливостей призводить до того, що СУБД стає надзвичайно важким програмним продуктом, який може займати багато місця на диску та вимагати великого об’єму оперативної пам’яті для ефективної роботи.

Вартість СУБД

В залежності від існуючого обчислюваного середовища та потребуємих функціональних можливостей, вартість СУБД може знаходитися в дуже широких межах.

Витрати на перетворення

В деяких ситуаціях вартість СУБД та додаткового апаратного забезпечення може виявитися несуттєвою у порівнянні з вартістю перетворення існуючих додатків для роботи з новою СУБД та новим апаратним забезпеченням. Ці витрати також містять вартість підготовки

персоналу для роботи з новою системою, а також сплачування послуг фахівців, які будуть здійснювати допомогу в перетворенні та запуску нової системи.

Все це є однією з головних причин, за якою деякі організації залишаються прибічниками колишніх систем та не бажають здійснювати перехід до найбільш сучасних технологій управління бази даних.

Продуктивність

Звичайно файлова система створюється для деяких спеціалізованих додатків, наприклад, для оформлення рахунків, а тому саме її продуктивність може бути дуже високою.

Але СУБД призначені для вирішення найбільш загальних задач та обслуговування одразу декількох додатків, а не якогось одного з них. В підсумку багато які додатки в новому середовищі будуть працювати не так швидко, як раніше.

Питання щодо самоконтролю:

1. Якими перевагами володіє СУБД?
2. Що потрібно зробити для того, щоб вилучити надмірність даних?
3. В яких випадках виникають несуперечливі дані?
4. Завдяки чому описується цілісність бази даних?
5. У вигляді чого може бути представлена система забезпечення безпеки баз даних?
6. Якими недоліками володіє СУБД?
7. Чому СУБД вважається складним програмним продуктом?
8. Продуктивність відноситься до переваг чи недоліків СУБД? Пояснити свою відповідь.

Тема «Об'єктно-орієнтована модель даних»

(2 години)

Мета: дізнатися про об’єктно-орієнтовану модель даних та її властивості.

Створення об'єктно-орієнтованих СУБД вважається одним з найбільш перспективних напрямків в галузі розробки нових типів баз даних.

Об'єктно-орієнтовані СУБД базуються на ідеях, сформованих в об'єктно-орієнтованих мовах програмування (**наслідування**, **інкапсуляції** та **поліморфізма**). Предметна область представляється у вигляді множини класів об'єктів. Структура та поведінка об'єктів одного класу (наприклад, товарів бази даних торгового підприємства) є однаковими.

Об'єкт володіє наступними характеристиками:

1. Має унікальний ідентифікатор, який однозначно визначає об'єкт.

2. Належить до деякого класу, який володіє визначеними поведінкою та властивостями.

3. Може обмінюватися повідомленнями з іншими об'єктами.

4. Має деяку внутрішню структуру. Об'єкти, внутрішня структура яких прихована від користувачів (відомо лише, які функції може виконувати даний об'єкт), мають назву **інкапсульованими**.

Поведінка об'єкта задається за допомогою методів його класу – операцій, які можна застосовувати до об'єкту. Здібність застосовувати один й той же метод для різних класів називається **поліморфізмом**.

В об'єктно-орієнтованій моделі можливе створення нового классу об’єктів на основі вже існуючого классу. Цей процес називається **наслідуванням**. Новий клас, який має назву підклас існуючого класу (**суперкласу**), наслідує всі властивості та методи суперкласу. Крім того, для нього можуть бути визначені додаткові властивості та методи.

Об'єктно-орієнтована СУБД дозволяє зберігати об'єкти та забезпечує їх спільне використання різноманітними додатками. Для цього вона повинна володіти наступними компонентами:

1. мовою баз даних, яка дозволяє декларувати класи об'єктів, а потім створювати, зберігати, вилучати та знищувати об'єкти.

2. Сховищем об'єктів, до яких можуть отримати доступ різні додатки. Для ссилок на об'єкти використовуються їх ідентифікатори.

Для практичної реалізації об'єктно-орієнтованих баз даних застосовуються два підходи:

1. Використовується мова об'єктно-орієнтованого програмування (наприклад, C++), доповнений засобами, які дозволяють при необхідності зберігати об'єкти після завершення програми, за допомогою якої вони були створені.

2. Основою є реляційна система, до якої додаються об'єктно-орієнтовані компоненти.

Недоліки об'єктно-орієнтованих баз даних:

1) відсутнє необхідне уніфіковане теоретичне обгрунтування та стандартизована термінологія;

2) не існує формально сформульованої методології проектування баз даних;

3) відсутні засоби створення нерегламентованих запитів;

4) немає загальнихправил підтримки узгоджуванності даних.

В завершення можна відмітити, що об'єктно-орієнтовані бази даних в сучасний час дуже важкі в проектуванні та експлуатації, що обмежує їх практичне застосування. Тому, недивлячись на інтенсивні дослідження, які продовжуються, об'єктно-орієнтована модель даних доки підтримується лише декількома СУБД (POET, Jasmine, Versant, Iris).

Питання щодо самоконтролю:

1. На яких ідеях базуються об'єктно-орієнтовані СУБД?
2. Що означає поняття наслідування**,** інкапсуляції та поліморфізму в об'єктно-орієнтованих моделях даних?
3. Які підходи застосовуються для практичної реалізації об'єктно-орієнтованих баз даних? Описати ці підходи.
4. Які Ви можете назвати недоліки об'єктно-орієнтованих баз даних?

Тема «Фундаментальні властивості відношень в реляційній моделі даних»

(2 години)

Мета: дізнатися про фундаментальні властивості відношень в реляційній моделі даних.

В реляційній моделі підтримуються такі фундаментальні властивості відношень:

- відсутність кортежів-дублікатів;
- відсутність впорядкованості кортежів;
- відсутність впорядкованості атрибутів;
- атомарність значень атрибутів.

Розглянемо детальніше кожне з цих властивостей.

1. Відсутність кортежів-дублікатів

Та властивість, що відношення не містять кортежів-дублікатів, виходить з визначення відношення як безлічі кортежів. У класичній теорії множин за визначенням кожна множина складається з різних елементів.

З цієї властивості витікає наявність у кожного відношення так званого первинного ключа - набору атрибутів, значення яких однозначно визначають кортеж відношення. Для кожного відношення принаймні повний набір його атрибутів має цю властивість. Проте при формальному визначенні первинного ключа потрібно забезпечення його «мінімальності», тобто в набір атрибутів первинного ключа не повинні входити такі атрибути, які можна відкинути без збитку для основної властивості, - однозначно визначати кортеж. Поняття первинного ключа є виключно важливим у зв'язку з поняттям цілісності баз даних.

2. Відсутність впорядкованості кортежів

Властивість відсутності впорядкованості кортежів відношення також є наслідком визначення відношення-екземпляра як множини кортежів. Відсутність вимоги до підтримки порядку на безлічі кортежів відношення дає додаткову гнучкість СУБД при зберіганні баз даних в зовнішній пам'яті і при виконанні запитів до бази даних. Це не суперечить тому, що при формулюванні запиту до БД, наприклад, на мові SQL можна зажадати сортування результуючої таблиці відповідно до значень деяких стовпців. Такий результат, взагалі кажучи, не відношення, а деякий впорядкований список кортежів.

3. Відсутність впорядкованості атрибутів

Атрибути відношень не впорядковані, оскільки за визначенням схема відношення є безліч пар {ім'я атрибуту, ім'я домена}. Для посилання на значення атрибуту в кортежі відношення завжди використовується ім'я атрибуту. Ця властивість теоретично дозволяє, наприклад, модифікувати схеми існуючих стосунків не лише шляхом додавання нових атрибутів, але і шляхом видалення існуючих атрибутів. Проте у більшості існуючих систем така можливість не допускається, і хоча впорядкованість набору атрибутів відношення явно не потрібно, часто як неявний порядок атрибутів використовується їх порядок в лінійній формі визначення схеми відношення.

4. Атомарність значень атрибутів

Значення усіх атрибутів є атомарними. Це витікає з визначення домена як потенційної множини значень простого типу даних, тобто серед значень домена не може міститися множина значень (відношення). Прийнято говорити, що в реляційних базах даних допускаються тільки нормалізовані відношення або відношення, представлені в першій нормальній формі. Потенційним прикладом ненормалізованого відношення є наступне:

Номер відділення	***Відділення***		
	Номер співробітника	***ПІБ співробітника***	***ЗП співробітника***
310	2934	Євдокімов М. К.	3000
	2935	Афоніна О. М.	3200
	2936	Петров А. А.	4000
313	2937	Константинов П. Р.	4500
315	2938	Мішина Г. Є.	3900

Можна сказати, що тут ми маємо бінарне відношення, значеннями атрибуту ВІДДІЛЕННЯ якого є відношення. Помітимо, що початкове відношення СПІВРОБІТНИКИ є нормалізованим варіантом відношення ВІДДІЛЕННЯ:

Номер співробітника	***ПІБ співробітника***	***ЗП співробітника***	***Номер відділення***
2934	Євдокімов М. К.	3000	310
2935	Афоніна О. М.	3200	310
2936	Петров А. А.	4000	310
2937	Константинов П. Р.	4500	313
2938	Мішина Г. Є.	3900	315

Нормалізовані відношення складають основу класичного реляційного підходу до організації баз даних. Вони мають деякі обмеження (не будь-яку інформацію зручно представляти у вигляді плоских таблиць), але істотно спрощують маніпулювання даними. Розглянемо, наприклад, два ідентичних оператори занесення кортежу:

Зарахувати співробітника Кузнєцова А. В. (номер співробітника - 3000, ЗП співробітника - 3700) у відділення номер 320

Зарахувати співробітника Кузнєцова А. В. (номер співробітника - 3000, ЗП співробітника - 3700) у відділення номер 310.

Якщо інформація про співробітників представлена у вигляді відношення СПІВРОБІТНИКИ, обидва оператори виконуватимуться однаково (вставити кортеж у відношення СПІВРОБІТНИКИ). Якщо ж працювати з ненормалізованим відношенням ВІДДІЛЕННЯ, то перший оператор виразиться в занесення кортежу, а другий - на додавання інформації про Кузнєцова в множинне значення атрибуту ВІДДІЛЕННЯ кортежу з первинним ключем 310.

Питання щодо самоконтролю:

1. Які фундаментальні властивості відношень підтримуються в реляційній моделі?
2. Що означає властивість відношення «Відсутність кортежів-дублікатів»?
3. Що означає властивість відношення «Відсутність впорядкованості кортежів»?
4. Що означає властивість відношення «Відсутність впорядкованості атрибутів»?
5. Що означає властивість відношення «Атомарність значень атрибутів»?

Тема «Дванадцать правил Е. Ф. Кодду»

(2 години)

Мета: ознайомлення зі змістом правил французського вченого та математика Е. Ф. Кодду, яким повина відповідати кожна реляційна СУБД.

Французський вчений та математик Е. Ф. Кодд запропанував 12 правил, яким повинна відповідати кожна реляційна система управління базами даних. Ними стали:

1. Правило інформації

Вся інформація в базі даних повина бути зображена виключно на логічному рівні та тільки одним способом – у вигляді значень, які є в наявності в таблицях.
Фактично це неформальне визначення реляційної бази даних.

2. Правило гарантійного доступу

Логічний доступ до всіх та кожного елементу даних (атомарному значенню) в реляційній базі даних повинен забезпечуватися шляхом використання комбінації імен таблиць, первинного ключа та імені стовпця.

Правило 2 вказує на роль первинного ключу при пошуку інформації в базі даних. Ім'я таблиці дозволяє знайти потрібну таблицю, ім'я стовбця дозволяє знайти потрібний стовбець, а первинний ключ дозволяє знайти строку, яка містить необхідний елемент даних.

3. Правило підтримки недійсних значень

В теперішній базі даних повина бути реалізована підтримка недійсних значень, які відрізняються від рядків символів нулевої довжини, рядки символів пробілу та від нуля або іншого числа та використовуються для зображення відсутніх даних незалежно від типу цих даних.

Правило 3 вимагає, щоб відсутні дані можна було б зобразити за допомогою недійсних значень (NULL).

4. Правило дінамічного каталогу, заснованого на реляційній моделі

Опис бази даних на логічному рівні повинен бути зображений в такому ж саме вигляді, що й головні дані, щоб користувачі, володіючі відповідними правами, мали змогу працювати з ним за допомогою тієї ж реляційної мови, яку вони використовують для роботи з головними даними.

Правило 4 говорить, що реляційна база даних повина сама себе описувати. Іншими словами, база даних повина вміщувати набір системних таблиць, які описують структуру самої бази даних.

5. Правило вичерпної підмови даних

Реляційна система може підтримувати різні мови та режими взаємодії з користувачем (наприклад, режим питань та відповідів). Але повина існувати хоча б
одна мова, оператори якої можна було б зобразити у вигляді строк символів у відповідність з деяким чітко зазначеним синтаксисом та який в повній мірі підтримує наступні елементи:

- визначення даних;
- визначення уявлень;
- обробка даних (інтерактивна та програмна);
- умови цілісності;
- ідентифікація прав доступу;
- межі транзакцій (початок, завершення, відміна).

Правило 5 потребує, щоб СУБД використовувала мову реляційної бази даних.
Така мова повина підтримувати всі головні функції СУБД – створення бази даних, читання та введення даних, реалізація захисту бази даних та ін.

6. Правило оновлення уявлень

Всі уявлення, які теоретично можна оновити, повинні бути доступні для оновлення.

Правило 6 відноситься до уявлень, які є віртуальними таблицями, дозволяючими показувати різним користувачам різні фрагменти структури бази даних.

7. Правило додавання, оновлення та знищення

Можливість працювати з відношеннями як з одним операндом повина існувати
не тільки при читанні даних, але й при їх додаванні, оновленні та знищенні даних.

Правило 7 акцентує увагу на тому, що бази даних за своєю природою орієнтовані на множину. Воно потребує, щоб операції додавання, оновлення та
знищення можна було б виконати над множиною рядків.

8. Правило незалежності фізичних даних

Прикладні програми та утіліти для роботи з даними повині на логічному рівні залишатися недоторканими при будь-яких змінах засобів зберігання даних або методів доступу до них.

9. Правило незалежності логічних даних

Прикладні програми та утіліти для роботи з даними повині на логічному рівні залишатися недоторканими при внесенні в базові таблиці будь-яких змін, які теоретично дозволяють зберігти недоторканими наявні в цих таблицях дані.

Правила 8 та 9 визначають відокремлення користувача та прикладної програми
від низькорівневої реалізації бази даних.

10. Правило незалежності умов цілісності

Повина існувати можливість визначення умов цілісності, специфічніх для кожної конкретної реляційної бази даних, на підмові реляційної бази даних та зберігання їх в каталозі, а не в прикладній програмі.

Правило 10 визначає, що мова бази даних повина підтримувати обмежувальні умови, які накладаються на вхідні дані та дії, які можуть бути виконані над даними.

11. Правило незалежності поширювання

Реляційна СУБД не повинна залежити від потреб конкретного користувача.

Правило 11 говорить про те, що мова баз даних повина забезпечувати можливість роботи з розподіленими даними, які розміщені на інших комп'ютерних системах.

12. Правило єдиної мови

Якщо в реляційній системі є низькорівнева мова (за один раз опрацьовує один запис), то повинна бути відсутня можливість використання її для того, щоб обійти правила та умови цілісності, виявленні на реляційной мові високого рівня (за один раз опрацьовує декілька записів).

Правило 12 попереджує використання інших можливостей для роботи з базою даних окрім мови бази даних, так як це може викликати порушення її цілістності.

Питання щодо самоконтролю:

1. Хто запропанував 12 правил, яким повина відповідати кожна реляційна СУБД?
2. Які правила були запропановані для реляційної СУБД?
3. Яке правило реляційної СУБД вказує на роль первинного ключу при пошуку інформації в базі даних?
4. Які правила реляційної СУБД визначають відокремлення користувача та прикладної програми від низькорівневої реалізації бази даних?
5. Що визначає правило незалежності умов цілісності?

Тема «Етапи проектування бази даних»

(2 години)

Мета: знати 3 етапи, які використовуються при проектуванні бази даних.

Проектуванню баз даних традиційно приділяється велика увага, так як ця робота в більшості визначає успішність експлуатації бази даних, яка створюється, можливості її модернізації та удосконалення в подальшому.

В процесі проектування БД часто виділяють три етапи.

Етап 1. Побудова концептуальної моделі предметної області

В рамках цього этапу досліджується предметна область – частина реального світу, для якої створюється БД. Вивчаються інформаційні потреби користувачів, виявляються інформаційні об'єкти та зв'язки між ними. Виходячи з отриманої інформації будується концептуальна модель предметної області, незалежна від моделі даних та програмих засобів (включаючи СУБД).

Етап 2. Логічне проектування – перетворення створенної концептуальної моделі в концептуальну схему, яка реалізується конкретною СУБД

На цьому етапі на основі концептуальної моделі розробляється структура БД, яка відповідає обраній для її створення СУБД. Для реляційної БД інформація розбивається на відношення (таблиці); для кожного відношення (таблиці) визначаються атрибути (поля), первинні ключі; відношення приводяться до нормалізованого вигляду; ідентифікуються зв'язки між відношеннями.

Етап 3. Фізичне проектування бази даних

На цьому етапі вирішуються проблеми фізичного розташування БД в зовнішній пам'яті та організації доступу до неї. Фізичне проектування БД реалізується адміністратором банку даних при конфігуруванні та налаштуванні системи. Від спеціалістів, які брали участь в проектуванні БД на попередніх етапах, цей процес може бути повністю прихований.

Питання щодо самоконтролю:

1. Які етапи виділяють в процесі проектування БД?
2. Яке призначення мають етапи проектування БД?

Тема «Аномалії оновлення в базі даних»

(2 години)

Мета: вивчення недоліків надмірності даних в базі даних та проблем, які виникають внаслідок надмірних даних.

Головною метою проектування реляційної бази даних є групування атрибутів в відношення таким чином, щоб зменшити надмірність даних та таким чином скоротити об'єм пам'яті, необхідний для фізичного зберігання відношень, які зображуються у вигляді таблиць.

Проблеми, які пов'язані з надмірністю даних, можна простежити, порівнявши відношення «Співробітники» та «Відділення» з відношенням «Співробітники в відділеннях».

Відношення «Співробітники»

Номер співробіт-ника	ПІБ	Адреса співробітника	Посада	Дата народження	ЗП	Номер відділ.
21	Філатов Андрій Петрович	Одеса, Вільямса 7, 45	Менеджер	01.05.1970	2950	5
37	Нікітіна Ганна Миколаївна	Черкаси, Корольова 67, 3	Секретар	16.09.1980	1700	3
14	Федоров Микола Сергійович	Черкаси, б-р Шевченко 19, 5	Директор	28.10.1969	5500	3
9	Краснова Олена Валеріївна	Київ, Пушкінська 21, 44	Бухгалтер	31.12.1964	4200	7
5	Петренко Оксана Вікторівна	Одеса, Левітана 4, 90	Менеджер	03.04.1972	2950	3
41	Васильєва Ганна Семенівна	Одеса, Грецька 47, 15	Бухгалтер	18.02.1968	4200	5

Відношення «Відділення»

Номер відділення	Адреса відділення	Телефон відділення
5	Одеса, Ген. Бочарова 19	551440
7	Київ, Іванова 8	285190
2	Черкаси, Шевченко 14	438615
4	Запоріжжя, Гоголя 82	648369
3	Одеса, Ген. Петрова 1	659569
1	Луганськ, Київська 67	938544

Номер співр	ПІБ	Адреса співробітника	Посада	ДН	ЗП	Номер відділ	Адреса відділення	Телефон відділення
21	Філатов Андрій Петрович	Одеса, Вільямса 7, 45	Менеджер	01.05. 1970	2950	5	Одеса, Ген. Бочарова 19	551440
37	Нікітіна Ганна Миколаївна	Черкаси, Корольова 67, 3	Секретар	16.09. 1980	2700	3	Одеса, Ген. Петрова 1	659569
14	Федоров Микола Сергійович	Черкаси, б-р Шевченко 19, 5	Директор	28.10. 1969	5500	3	Одеса, Ген. Петрова 1	659569
9	Краснова Олена Валеріївна	Київ, Пушкінська 21, 44	Бухгалтер	31.12. 1964	4200	7	Київ, Іванова 8	285190
5	Петренко Оксана Вікторівна	Одеса, Левітана 4, 90	Менеджер	03.04. 1972	2950	3	Одеса, Ген. Петрова 1	659569
41	Васильєва Ганна Семенівна	Одеса, Грецька 47, 15	Бухгалтер	18.02. 1968	4200	5	Одеса, Ген. Бочарова 19	551440

Відношення «Співробітники в відділеннях» є альтернативною формою зображення відношень «Співробітники» та «Відділення».

В відношенні «Співробітники в відділеннях» створюються надмірні дані, так як відомості про відділення компанії повторюються в записах про кожного співробітника цього відділення.

В протилежність цьому, в відношенні «Відділення» відомості про відділення компанії створюються лише в одному записі, а в відношенні «Співробітники» повторюється лише номер відділення компанії, який уявляє собою місце роботи кожного співробітника.

При роботі з відношеннями, які мають надмірні дані, можуть виникати проблеми, які називаються **аномаліями оновлення** та діляться на:

1. аномалії вставки;
2. аномалії знищення;
3. аномалії модифікації.

Розглянемо детальніше про кожну з аномалій.

1. Аномалії вставки

Існує 2 головних різновиди аномалій вставки, які можна зобразити за допомогою відношення «Співробітники в відділеннях».

1. При додаванні відомостей про нових співробітників у відношення «Співробітники в відділеннях» необхідно вказати і відомості про відділення компанії, в якому ці співробітники працюють.

Наприклад, при додаванні відомостей про нового співробітника відділення компанії №7 необхідно ввести відомості про це відділення компанії №7, які повинні співпадати з відомостями про це ж відділення компанії в інших записах відношення «Співробітники в відділеннях».

Відношення «Співробітники» та «Відділення» не постраждають від такого потенційного неспівпадання даних, так як для кожного співробітника в відношенні «Співробітники» необхідно буде ввести тільки номер існуючого відділення компанії. Окрім цього, відомості про відділення компанії з номером №7 записуються в базу даних один раз у вигляді одного рядка відношення «Відділення».

2. Для додавання відомостей про нове відділення компанії, яке ще не має своїх власних співробітників, необхідно буде присвоїти визначення Null всім атрибутам переліку персоналу відношення «Співробітники в відділеннях».

Так як атрибут «Номер співробітника» є первинним ключем відношення «Співробітники в відділеннях», то спроба ввести визначення Null в атрибут «Номер співробітника» викличе порушення цілісності сутностей та буде відхилена. Тобто, в відношення «Співробітники в відділеннях» неможливо ввести запис про нове відділення компанії, який містить визначення Null в атрибуті «Номер співробітника».

Структура відношень «Співробітники» та «Відділення» допомагає уникнути виникнення цієї проблеми, так як відомості про відділення компанії заносяться в відношення «Відділення» незалежно від введення відомостей про співробітників. Відомості про співробітників, які будуть працювати в новому відділенні компанії, можуть бути додані в відношення «Співробітники» трохи згодом.

2. Аномалії знищення

При знищенні з відношення «Співробітники в відділеннях» запису з інформацією про останнього співробітника деякого відділення компанії, відомості про це відділення компанії будуть повністю знищенні з бази даних.

Наприклад, після знищення з відношення «Співробітники в відділеннях» запису «Краснова Олена Валеріївна» з особовим номером «9» з бази даних неявним чином будуть знищенні всі відомості про відділення компанії з номером 7.

Структура відношень «Співробітники» та «Відділення» допомагає попередити виникнення цієї проблеми, так як відомості про відділення компанії характеризуються окремо від записів з відомостями про співробітників. Пов’язує ці два відношення тільки атрибут «Номер відділення».

При знищенні з відношення «Співробітники» запису з номером співробітника «9» відомості про відділення компанії №7 в відношенні «Відділення» залишаться недоторканими.

3. Аномалії модифікації

При спробі змінити значення одного з атрибутів для деякого відділення компанії в відношенні «Співробітники в відділеннях», необхідно оновити відповідні значення в записах

для всіх співробітників цього відділення. Якщо не всі записи з даними в відношенні «Співробітники в відділеннях» будуть оновленні, то в такому разі база даних буде мати суперечливі відомості.

Тобто, в нашому прикладі для відділення компанії з номером 3 в записах, які ставляться до різних співробітників, помилково можуть бути вказані різні значення номеру телефону цього відділення.

Всі наведені приклади показують те, що краще зберігати інформацію в окремих відношеннях «Співробітники» та «Відділення», а не в одному відношенні «Співробітники в відділеннях».

Питання щодо самоконтролю:

1. Що таке «Надмірність даних»?
2. Що таке «Аномалії оновлення»?
3. Які види аномалій оновлення Ви знаєте?
4. При яких обставинах виникають різні види аномалій оновлення? Довести свою відповідь на своїх прикладах.

Тема «Переваги та недоліки нормалізації»

(2 години)

Мета: знати переваги та недоліки нормалізації

Переваги нормалізації:

- краща загальна організація бази даних;
- скорочення кількості непотрібного повторювання даних;
- узгоджування даних в базі даних;
- більш гнучка структура бази даних;
- ефективні можливості забезпечення безпеки та надійності бази даних.

Процес нормалізації покращує організацію бази даних, полегшуючи роботу з нею усім, починаючи з простих користувачів до адміністратора, який відповідає за загальне керування об’єктами бази даних. Зменшується кількість повторювань даних, що спрощує структуру даних та економить дисковий простір. Через скорочення дублювання даних зменшується ймовірність їх неузгоджування. Так як в підсумку нормалізації база даних поділяється на більш дрібні таблиці, модифікувати існуючі таблиці стає легше. Набагато легше змінювати таблицю з невеликою кількістю даних, ніж велику таблицю, яка містить усі важливі для бази даних значення. Нарешті, підвищується безпека у тому розумінні, що адміністратор бази даних

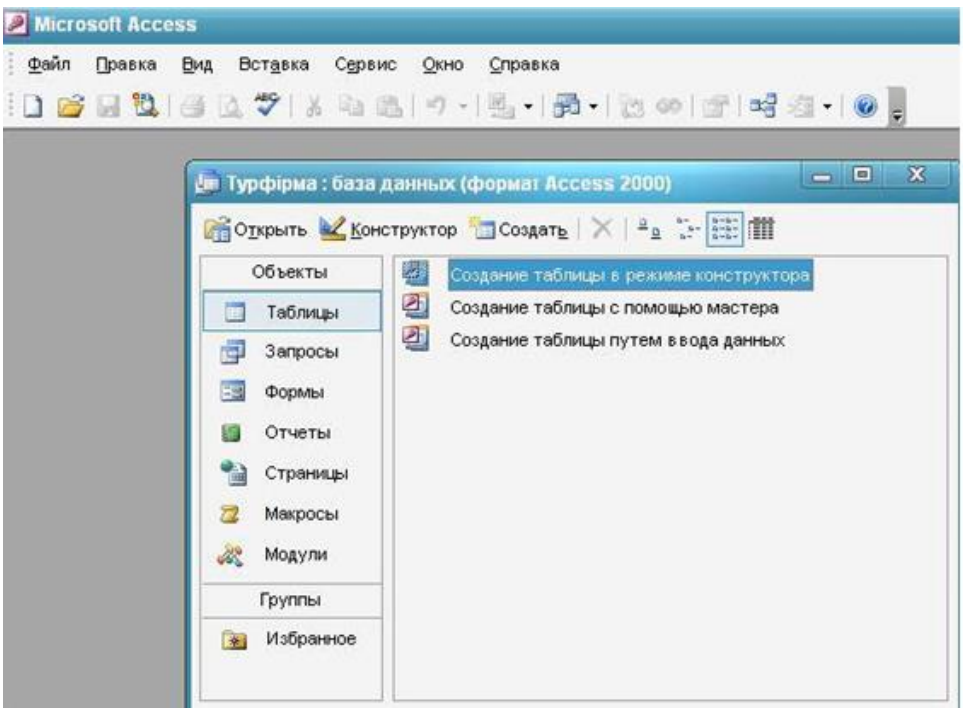
отримує можливість дозволити різним користувачам доступ лише до обмеженого переліку таблиць. Нормалізація спрощує керування безпекою.

Недоліки нормалізації

Хоча більшість вдало працюючих баз даних в деякому ступені є нормалізованими, нормалізація має один суттєвий недолік: уповільнення роботи бази даних. Виконання запиту або транзакції передбачає використання центрального процесору комп'ютера, пам'яті та операцій введення-виведення. Тобто, в нормалізованій базі даних для виконання транзакцій або запитів найбільш інтенсивно використовується центральний процесор, вимагається більше пам'яті та більша кількість операцій введення-виведення, ніж в ненормалізованій базі даних. В нормалізованій базі даних треба знаходити відповідні таблиці та пов'язувати дані для того, щоб вилучати потрібну інформацію та обробити її.

Тема «Об'єкти бази даних»

База даних в Access вміщує об'єкти різних категорій (усього таких категорій 7):



Кожній із категорій відповідає своя вкладка вікна бази даних **Таблиці, Запити, Форми, Звіти, Макроси, Модулі.**

Таблиці. Це основна категорія об'єктів у реляційній СУБД, оскільки вся інформація зберігається в базі даних у вигляді таблиць. Кожна таблиця складається з записів (рядків) і з полів (стовпців). Робота з таблицею виконується в двох основних режимах: у режимі конструктора й у режимі таблиці.

Запити. Об'єкти цього типу призначені для отримання даних з однієї або кількох таблиць. Відбір потрібних відомостей відбувається на основі сформульованих критеріїв.

Фактично за допомогою запитів створюються нові таблиці, у яких використовуються дані з існуючих таблиць.

Форми. Цей тип об'єктів використовується в основному для зручного введення даних. Форма є ніби бланком, який потрібно заповнити. Заповнити такий бланк зможе навіть початківець. Позитивним є те, що форми запобігають безпосередньому внесенню змін у таблиці.

Звіти. Об'єкти-звіти відображають дані так, що їх зручно переглядати. На основі звіту може бути створений документ, що буде роздрукований або включений у документ іншого додатка.

Макроси. Макросами називаються «макрокоманди», що запускаються простим натисканням кількох клавіш і можуть виконувати такі дії, як відкриття таблиць і форм, виконання опцій меню, керування вікнами тощо. Користувач може створювати свої макроси для послідовностей операцій, які часто застосовуються.

Модулі. Цей тип об'єктів є програмними модулями, написаними мовою VBA. Модулі - це процедури для обробки подій або виконання обчислень. Розбиття на модулі полегшує процес створення і настроювання програми.

Питання щодо самоконтролю:

1. Які об’єкти вміщує СУБД MS Access?
2. В яких режимах виконується робота з таблицею?
3. Який об'єкт СУБД MS Access призначений для отримання даних з однієї або декількох таблиць?
4. Який об'єкт використовується для зручного введення даних?
5. Що таке макроси?

Тема «Типи даних»

Тип даних визначається значеннями, які передбачається вводити до поля (наприклад, текст або число). Якщо надалі доведеться задати інший тип даних, це можна виконати у режимі конструктора.

У Access передбачено такі типи даних.

Текстовий - для введення тексту завдовжки до 255 символів. Цей тип даних установлюється за умовчанням.

Поле МЕМО - для ь-іедення заміток або довгих описів (можливе введення до 64 00 символів).

Числовий - для введення числових даних, для яких виділяється 1, 2 або 4 байти.

Дата/час - для введення дати і часу, для яких передбачено 8 байтів.

Грошовий - використовується для роботи з грошовими одиницями. Цей тип даних займає 8 байтів і допускає до 15 символів у цілій частині числа і 4 – у дробовій. Використання грошового типу запобігає помилці округлень під час обчислень.

Лічильник - для введення числа, що автоматично збільшується на одиницю при додаванні до таблиці нового запису. Дані цього типу займають 4 байти.

Логічний - для збереження логічного значення Истина чи Хиба. Таке поле займає 1 біт.

Об’єкти OLE – для збереження в таблиці OLE об’єктів (наприклад, малюнків, звуків, документів Word тощо.). Розмір збережених об'єктів OLE обмежується лищ$_e$ І ємністю диска.

Гиперпосилання - служить для запису до таблиці гіперпосилань (шляху URL)

У списку **Тип данных** конструктора є також позиція **Майстер підстановок,** за допомогою якої обираються значення з іншої таблиці або зі списку значень.

Тема «Встановлення зв’язків між таблицями в MS Access»

Розглянемо можливі відношення між таблицями бази даних. Вони бувають таких типів: "один до одного", "один до багатьох", "багато до одного" і "багато до багатьох".

Найпоширенішим у таблицях реляційних баз даних є відношення "один до багатьох".

Відношення "один до багатьох" *означає, що одному запису таблиці відповідають кілька записів в іншій таблиці.*

Розглянемо створені таблиці для бази даних «Фірма» «Подорож». Туристична фірма продає путівки, причому однакові путівки вона може продати кільком клієнтам. Тому одному запису в таблиці «Путівки» можуть відповідати декілька записів у таблиці «Замовлення»

Путівки : таблиця

Код путівки	Країна	Вид	Кількість днів	Проїзд	Ціна
1	Болгарія	відпочинок	3	авіа	1900
2	Болгарія	відпочинок	10	автобус	3500
3	Болгарія	лікування	14	авіа	4950
4	Болгарія	екскурсії	2	авіа	1450
5	Греція	відпочинок	10	авіа	4520
6	Греція	відпочинок	5	авіа	3600
7	Греція	екскурсії	3	авіа	3570
8	Греція	екскурсії	4	автобус	3800
9	Чехія	екскурсії	3	авіа	1750
10	Чехія	екскурсії	2	автобус	1650
11	Чехія	відпочинок	10	авіа	2250
12	Чехія	відпочинок	10	автобус	2800
13	Чехія	лікування	14	авіа	4880
14	Чехія	лікування	10	автобус	2400
15	Туреччина	екскурсії	3	авіа	2200
16	Туреччина	екскурсії	2	автобус	2700
17	Туреччина	відпочинок	12	авіа	3300
18	Туреччина	відпочинок	10	автобус	2900
(Счетчик)			0		0

Замовлення : таблиця

№ замовлення	Дата	Код клієнта	Код путівки	Кількість
1	01.01.2010	1	5	4
2	03.03.2010	3	2	2
3	03.03.2010	4	3	1
4	04.03.2010	6	1	3
5	04.03.2010	4	20	12
6	09.03.2010	6	3	2
7	09.03.2010	3	3	10
8	16.03.2010	5	3	2
(Счетчик)		0	0	0

Сторона "один" у відношенні "один до багатьох" називається *головною* таблицею. Сторона "багато" у цьому самому відношенні називається *зв'язаною* таблицею. У прикладі зв'язаних таблиць «Путівки» і «Замовлення» ми бачимо, що ключове поле головної таблиці зв'язується з відповідним полем зв'язаної таблиці.

Тип відношень "один до одного" трапляється в таблицях баз даних рідше ніж "один до багатьох".

У ***відношенні «один до одного"*** *одному запису в головній таблиці відповідає один запис у зв'язаній таблиці.*

У наведених нами раніше таблицях немає відношення «один до одного Щоб проілюструвати це відношення, ми могли б додатково до таблиці «Клієнти» створити таблицю «Банківські дані», у якій містилися б банківські реквізити Клієнтів. Наприклад, у полі «Рахунок» було б зазначено номери розрахункових рахунків клієнтів. Оскільки рахунок кожного клієнта унікальний, між полем «Код клієнта» таблиці «Клієнти» і полем «Рахунок» таблиці «Банківські дані» існувало б відношення «один до одного».

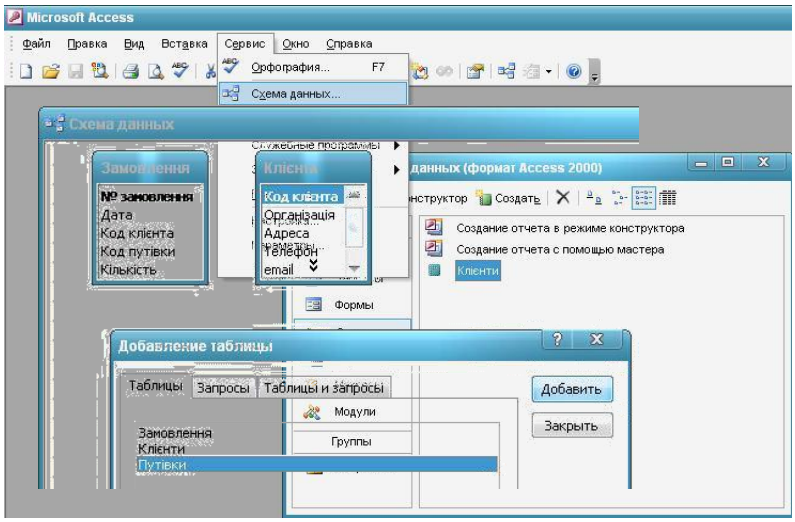
Встановлення зв'язків між таблицями

Отже, між полями таблиць можуть існувати певні відношення.

Розглянемо встановлення зв'язків на прикладі таблиць «Путівки», «Клієнти» і «Замовлення». Починаючи зв'язування таблиць, переконайтеся, що всі таблиці і форми закриті.

Потім переключіться у вікно бази даних. Клацніть по кнопці Схема даних 

на панелі інструментів вікна Access або дайте команду Сервіс – Схема даних

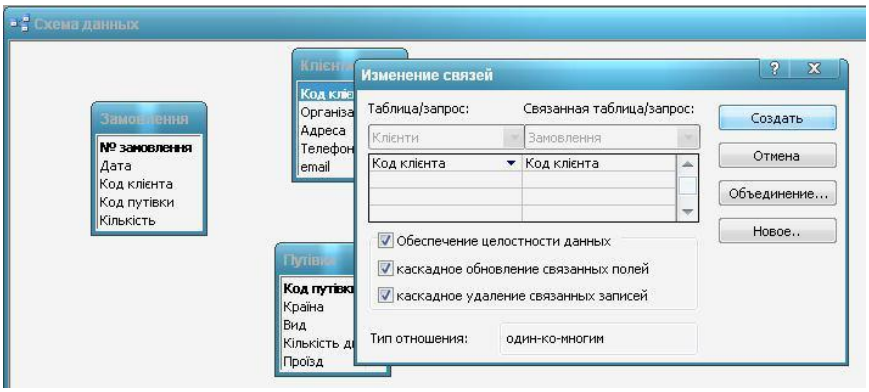


У відповідь відкриється однойменне діалогове вікно. Водночас з вікном Схема данных з'явиться діалог Добавление таблицы

Однак цей діалог може і не з'явитися, якщо раніше були створені зв'язки між таблицями (вікно Схема данных не порожнє). У цьому разі потрібно очистити вікно від зв'язків, клацнувши по кнопці Очистить макет. Потім натисніть кнопку Добавить таблицу на панелі інструментів. За умовчанням в діалозі Добавление таблицы буде відкрита вкладка Таблицы. Виділіть у списку таблицю «Замовлення» і натисніть кнопку Добавить. У вікні Схема данных з'явиться список полів таблиці «Замовлення». Аналогічно виведіть списки полів таблиць «Клієнти» і «Путівки» у вікно Схема данных, після чого закрийте діалог Добавление таблицы. Для зв'язування полів «Код путівки» оберіть це і поле у головній таблиці «Путівки» і перетягніть його мишею до зв'язаної таблиці «Замовлення».

Зазначимо, що напрямок перетягування поля завжди повинен бути «від головної таблиці до зв'язаної».

У діалозі Связи установіть перемикач Обеспечение целостност данных. Це дозволить уникнути деяких помилок при створенні експлуатації бази даних.



Клацніть по кнопці Создать, і встановлений зв'язок буде відображений у вікні Схема данных.

Зв'язок показаний лінією, позначеною цифрою 1 і символом нескінченності □, що означає відношення «один до багатьох». Аналогічно створіть зв'язок між полями «Код клієнта» таблиць «Клієнти» і «Замовлення». Ви одержите схему зв'язку, показану на малюнку.

Макет зв'язків

Зручним для вас способом (наприклад, щоб лінії зв'язків не перетиналися) розмістіть списки полів у вікні Схема данных. Списки можна перетягувати, захопивши мишею заголовок списку. Розташування списків у вікні називається *макетом зв'язків.* Після цього можете закрити вікно Схема данных. Програма виведе на екран запит щодо того, чи потрібно зберігати макет зв'язків-Клацніть по кнопці Да. Якщо ви відповісте Нет, то збережуться лише створені зв'язки, а не компонування списків.

Зверніть увагу, що в розглянутих випадках ми зв'язували ключове поле (позначене напівжирним шрифтом у списку полів) головної таблиці з відповідним полем зв'язаної таблиці. Поле зв'язаної таблиці називають *полем зовнішнього ключа.*

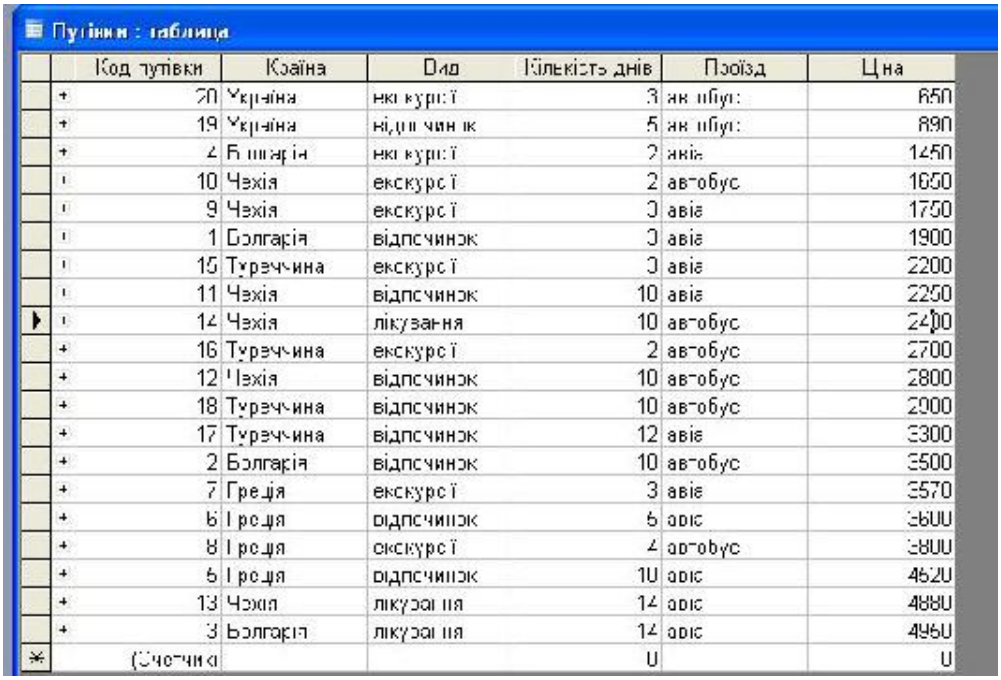
Для видалення будь-якого зв'язку не потрібно знову створювати макет зв'язав. Виділіть зв'язок клацанням миші й натисніть клавішу Delete. Після появи запиту на видалення зв'язку клацніть по кнопці Да.

Сортування записів

При введенні даних у таблиці або форми записи розташовуються у порядку введення. Це не завжди зручно при перегляді введеної інформації. Бажано згрупувати й упорядкувати інформацію, щоб вона була змістовною й у ній було легко орієнтуватися. Наприклад, у таблиці «Замовлення» доречно виділити товар з найбільшою купівельною спроможністю, помістивши відповідні записи на початок таблиці, а в таблиці «Клієнти» доцільно розподілити клієнтів за регіонами. Цього легко досягти сортуванням записів, яке виконується у такий спосіб.

Відкрийте таблицю і перейдіть у Режим таблицы. Активізуйте поле таблиці, за яким відбуватиметься сортування.

Клацніть по кнопці (сортування за зростанням) або по кнопці (сортування за спаданням).

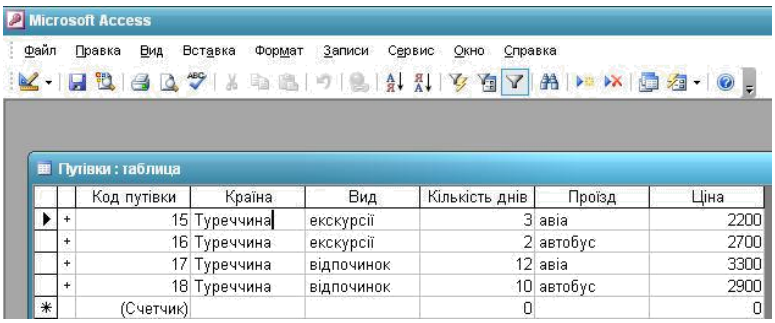


Код путівки	Країна	Вид	Кількість днів	Проїзд	Ціна
20	Україна	екскурсії	3	автобус	650
19	Україна	відпочинок	5	автобус	890
4	Болгарія	екскурсії	2	авіа	1450
10	Чехія	екскурсії	2	автобус	1650
9	Чехія	екскурсії	3	авіа	1750
1	Болгарія	відпочинок	3	авіа	1900
15	Туреччина	екскурсії	3	авіа	2200
11	Чехія	відпочинок	10	авіа	2250
14	Чехія	лікування	10	автобус	2400
16	Туреччина	екскурсії	2	автобус	2700
12	Чехія	відпочинок	10	автобус	2800
18	Туреччина	відпочинок	10	автобус	2900
17	Туреччина	відпочинок	12	авіа	3300
2	Болгарія	відпочинок	10	автобус	3500
7	Греція	екскурсії	3	авіа	3570
6	Греція	відпочинок	6	поїзд	3600
8	Греція	екскурсії	4	автобус	3800
5	Греція	відпочинок	10	поїзд	4620
13	Чехія	лікування	14	поїзд	4880
3	Болгарія	лікування	14	поїзд	4960
(Счетчик)			0		0

Прості фільтри даних

Сортування даних дозволяє упорядковувати їх, однак воно не скорочу кількості записів у таблиці, які доводиться переглядати користувачу. Зручними засобами для перегляду записів є *фільтри.*

Припустимо, що вам потрібно переглянути відомості про путівки до Туреччині Відкрийте таблицю «Путівки» у режимі таблиці. Установіть курсор *і* комірку «Туреччина» і натисніть кнопку  Фільтр по виділеному на панелі інструментів. На екрані з'являться записи про путівки до цієї країни.



Код путівки	Країна	Вид	Кількість днів	Проїзд	Ціна
15	Туреччина	екскурсії	3	авіа	2200
16	Туреччина	екскурсії	2	автобус	2700
17	Туреччина	відпочинок	12	авіа	3300
18	Туреччина	відпочинок	10	автобус	2900
(Счетчик)			0		0

Цей простий фільтр дозволяє відфільтрувати записи лише за однією умовою, що накладається на поле. Для виконання фільтрації ще за однією умовою (наприклад, за полем «Проїзд») клацніть по комірці з потрібним значенням (припустимо, авіа). Далі натисніть кнопку Фильтр по выделенному - і ви одержите усього два записи, що містять слово «Туреччина» у полі «Країна» і слово «авіа» у полі «Проїзд».


Путівки : таблица

	Код путівки	Країна	Вид	Кількість днів	Проїзд	Ціна
+	17	Туреччина	відпочинок	12	авіа	3300
+	15	Туреччина	екскурсії	3	авіа	2200
*	(Счетчик)			0		0

Фільтрація даних стосується лише записів на екран, сама таблиця залишається без змін. Для скасування фільтра й відновлення попереднього вигляду таблиці клацніть

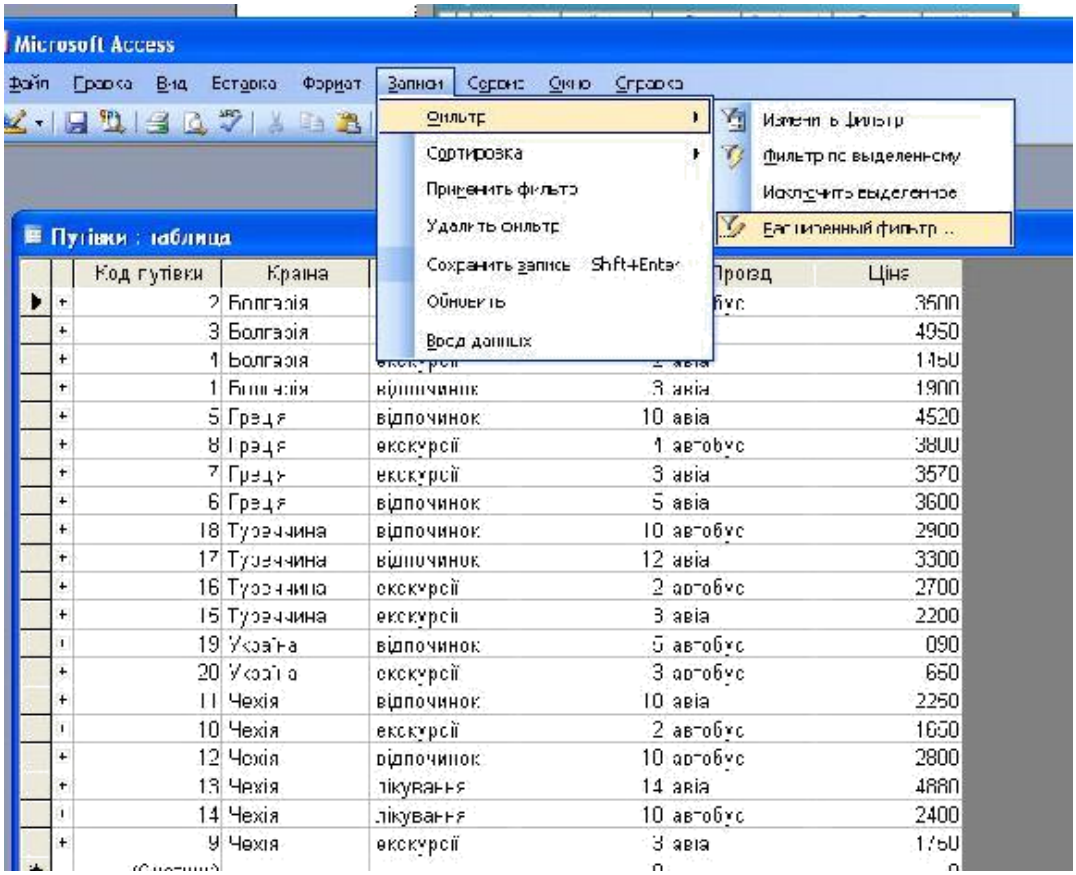
по кнопці Знищити фільтр:  , розміщеній на панелі інструментів.

При установці фільтра, що містить кілька умов, зручно скористатися дещо іншим інструментом. Відкривши таблицю, у якій збираєтеся виконати фільтрацію, клацніть по кнопці Змінити фільтр на панелі інструментів. На екран буде виведено лише один рядок таблиці. У полі «Країна» з'явиться трикутна стрілка розкривного списку. За допомогою цього списку задайте одну умову. Клацніть по іншому полю й аналогічно задайте другу умову (наприклад, вид путівки). Задайте всі необхідні умови і натисніть кнопку Применение фильтра.

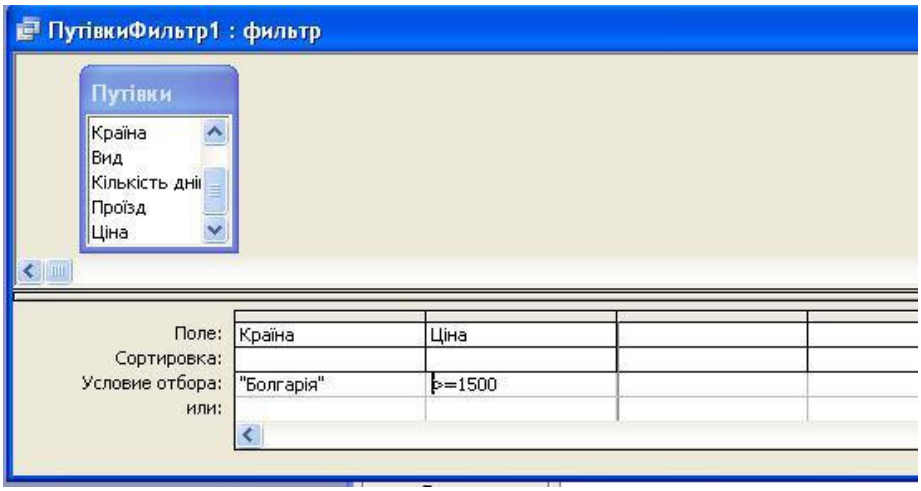
Розширений фільтр

Для задания складних умов фільтрації використовується розширений фільтр. Припустимо, ви хочете переглянути у таблиці «Путівки» записи про путівки до Болгарії вартістю не більше 1500 грн. Для цього зручно створити розширений фільтр.

Відкрийте таблицю «Путівки» бази даних «Тур фірма» «Подорож» і перейдіть до режиму таблиці (клацання по кнопці Вид).Оберіть команду меню Записи - Фильтр – Розширений фильтр.У діалозі



Клацніть по першій комірці рядка Поле й у розкривному списку оберіть позицію «Країна». У рядку Умова відбору наберіть«Болгарія».



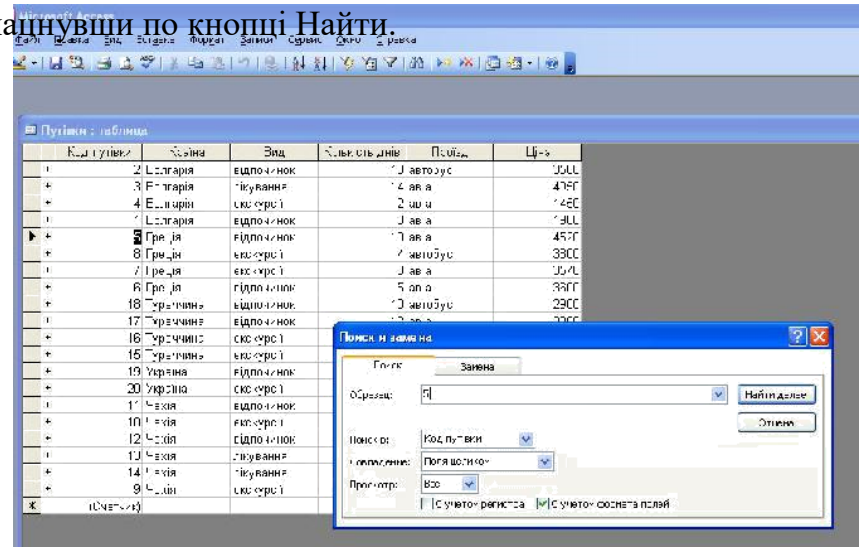
У сусідній комірці рядка Поле оберіть зі списку позицію «Ціна», а в умові вибору зазначте «<=1500». Клацніть по кнопці Применение фильтра (або задайте команду Фильтр -Применить фильтр). На екрані з'являться відфільтровані відомості про путівки до Болгарії в заданому інтервалі вартості.

Тема «Пошук запису»

Якщо таблиці баз даних великі і важко знайти той чи інший запис, можна вернутися за допомогою до засобу Пошук. Працюючи з таблицею (наприклад, «Путівки»), перейдіть у режим таблиці (якщо він не встановлений).

Клацніть мишею в тому полі, значення якого буде використано для пошуку (наприклад, «Код»). Натисніть на кнопку Знайти на панелі інструментів.

У діалозі введіть значення поля (наприклад, число 5). Запустіть процедуру пошуку, клацнувши по кнопці Найти.



Якщо необхідно виконати пошук, враховуючи регістр слова або формат полів, установіть перед натисканням кнопки Найти відповідні перемикачі в діалозі пошуку. У цьому самому діалозі є розкривні списки Просмотр і Совпадение, що дозволяють задати зону пошуку (все, вверх, вниз) і ознаку збігу (поля целиком, с любой частью поля, с начала поля).

Використання виразів у запитах

При формуванні запиту ви можете задати обробку даних, наприклад знайти суму або середнє значення для будь-якого поля. Для цього використовуються *вирази,* за якими виконуються обчислення, а результати обчислень заносяться в окреме поле. У виразах можна вживати знаки арифметичних операцій +, -, *, /, оператори порівняння =, <, >, <=, >=, а також імена полів, які взяті у квадратні дужки. Наприклад, вираз [Ціна]*1,25 означає що вміст поля Цена збільшується у 1,25 разу.

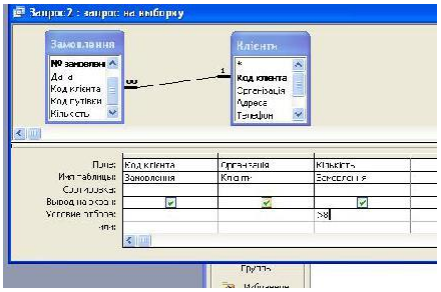
Надалі ми використовуватимемо вирази при заданні умов відбору і під час формування поля, що обчислюється в запиті.

Умови відбору

Запит, сформований згідно з вказівками пункту «Створення запиту», містить всі замовлення путівок. Якщо ж вас цікавлять лише великі замовлення (кількість путівок перевищує певне число), краще сформувати запит із заданиям умови відбору у такий спосіб:

У вікні бази даних перейдіть на вкладку Запросы і клацніть двічі по піктограмі «Замовлення: Запрос».

У відповідь відкриється вікно запиту. Перейдіть у режим конструктора запитів, клацнувши по кнопці Вид на панелі інструментів.



У діалозі наведено схему даних для розглянутих таблиць, нижче - бланк запиту. Клацніть по комірці, розташованій на перехресті рядка Условие отбора і стовпця «Кількість». Введіть із клавіатури вираз «>8» і натисніть Enter.

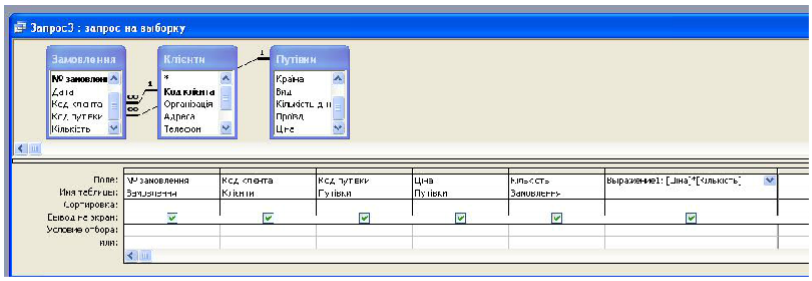
Тема «Розрахунки в запиті»

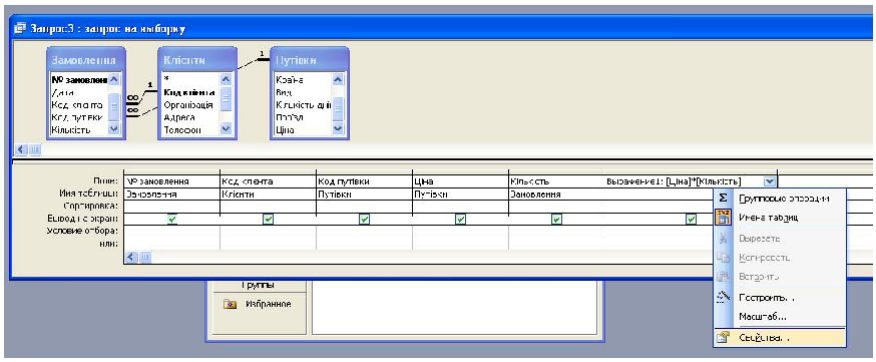
Проілюструємо виконання розрахунків на прикладі запиту, сформованого на основі таблиць «Замовлення» і «Путівки». Нас цікавитиме сума кожного замовлення, що обчислюється як добуток ціни путівки та кількості путівок: [Ціна]*[Кількість]. Виконується подібний запит таким чином.

Спочатку сформуйте запит, показаний на рисунку.

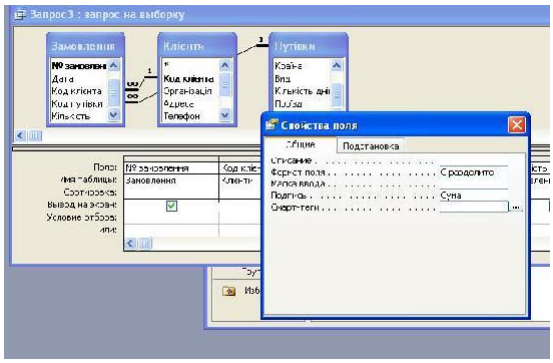
Передемо в режим конструктора запитів, клацнвши на кнопці Вид н панелі інструментів.

У вікні клацніть по полю праворуч від поля «Кількість». Введіть вираз [Ціна]*[Кількість] і натисніть клавішу Enter. Перед введеним виразом з'явиться текст Выражение1:

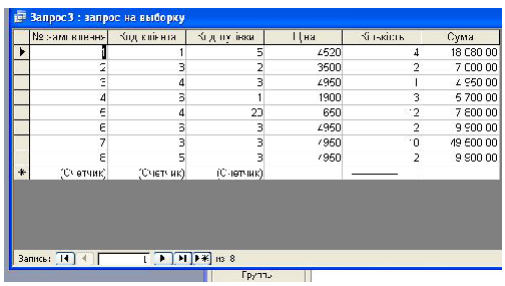




Клацніть правою кнопкою миші в зоні поля з виразом і оберіть у контекстному меню команду Свойства. У діалозі Свойства поля задайте формат поля С разделителем (два десяткові знаки після коми) і назву поля «Сума».



Натисніть кнопку Вид і перейдіть у Режим таблиці. Ви отримаєте запит, в останньому стовпці якого буде зазначена сума кожного замовлення.



Отже, на основі таблиць бази даних ви отримали запит, у якому було виведено обчислюване поле - сума всіх зроблених замовлень на путівки. Розрахунки виконуються безпосередньо при виведені запиту. Результати обчислень у таблицях не зберігаються. Тому результати запиту завжди представляють поточний вміст бази даних.

Тема «Мова SQL для створення запитів»

(2 години)

Мета: дізнатися про історію розвитку структурованої мови запитів SQL.

SQL (**S**tructured **Q**uery **L**anguage) – це структурована мова запитів до реляційних баз даних.

На початку 1970-х років в одній із дослідницьких лабораторій компанії IBM розробили експериментальну реляційну СУБД IBM System R, для якої потім створили спеціальну мову SEQUEL, яка дає змогу відносно просто керувати даними в цій СУБД.

Абревіатура SEQUEL розшифровувалася як Structured English QUEry Language - "структурована англійська мова запитів". Пізніше з юридичних міркувань мову SEQUEL було перейменовано на SQL.

Метою розробки було створення простої непроцедурної мови, якою міг скористатися будь-який користувач, навіть той, хто не має навичок програмування. Власне розробкою мови запитів займалися Дональд Чемберлін і Рейс Бойс.

Стандарти мови SQL

Оскільки до початку 1980-х років існувало кілька варіантів СУБД від різних виробників, причому кожен із них мав власну реалізацію мови запитів, було ухвалено рішення розробити стандарт мови, що гарантуватиме переносимість ПЗ з однієї СУБД на іншу (за умови, що вони підтримуватимуть цей стандарт).

В 1983 році Міжнародна організація зі стандартизації (ISO) та Американський національний інститут стандартів (ANSI) приступили до розробки стандарту мови SQL. Стандарт SQL було вперше опубліковано у 1986 р. (SQL-86), він забезпечував мінімальну функціональність.

Оновлювався:

1989 - (SQL-89) механізм підтримки цілісності посилань,

1992 - (SQL-2) розширена функціональність,

1999 - (SQL-3) додано підтримку регулярних виразів, рекурсивних запитів, підтримку тригерів, і деякі об'єктно-орієнтовані можливості.

2003 - (SQL:2003) - введено розширення для роботи з XML-даними, генератори послідовностей і засновані на них типи даних,

2006 – (SQL:2006) - функціональність роботи з XML-даними значно розширено. З'явилася можливість спільно використовувати в запитах SQL і Xquery,

2008 – (SQL:2008) - поліпшено можливості віконних функцій, усунуто деякі невизначеності стандарту SQL:2003.

Переваги мови SQL

Незалежність від конкретної СУБД Незважаючи на наявність діалектів і відмінностей у синтаксисі, здебільшого тексти SQL-запитів можна досить легко перенести з однієї СУБД в іншу. Існують системи, розробники яких від самого початку орієнтувалися на застосування щонайменше кількох СУБД.

Наявність стандартів Наявність стандартів і набору тестів для виявлення сумісності та відповідності конкретної реалізації SQL загальноприйнятому стандарту тільки сприяє "стабілізації" мови.

Декларативність За допомогою SQL програміст описує тільки те, які дані потрібно витягти або модифікувати. Те, яким чином це зробити, вирішує СУБД безпосередньо під час обробки SQL-запиту. Однак не варто думати, що це повністю універсальний принцип - програміст описує набір даних для вибірки або модифікації, однак йому водночас корисно уявляти, як СУБД буде розбирати текст його запиту.

Недоліки мови SQL

Невідповідність реляційній моделі даних Творці реляційної моделі даних Едгар Кодд, Крістофер Дейт та їхні прихильники вказують на те, що SQL не є істинно реляційною мовою.

Складність Хоча SQL і замислювався як засіб роботи кінцевого користувача, зрештою він став настільки складним, що перетворився на інструмент програміста.

Відступи від стандартів Незважаючи на наявність міжнародного стандарту, багато компаній, що займаються розробкою СУБД (наприклад, Oracle, Sybase, Microsoft, MySQL AB), вносять зміни в мову SQL, застосовувану в розроблюваній

СУБД. Таким чином, з'являються специфічні для кожної конкретної СУБД діалекти мови SQL.

Складність роботи з ієрархічними структурами Раніше діалекти SQL більшості СУБД не пропонували способу маніпуляції деревоподібними структурами.

Тема «Створення звітів в СУБД Access»

(2 години)

Мета: навчитися створювати звіти в СУБД Access

Звітом називають будь-який набір даних, призначений для друку.

Так як звіт призначений спеціально для виведення на друк, MS Access у випадку необхідності виведе звіт на екран в вікні попереднього перегляду. Користувач немає можливості редагувати дані, які відображаються в звіті.

Після натискання по кнопці «Создать» MS Access відображає на екрані діалогове вікно «Новый отчёт». Як й при роботі з формами, треба використовувати відкриваючий список цього вікна для обрання таблиці або запиту, для яких створюється звіт.

Потім необхідно обрати одну з опцій:

- **«Конструктор»** – створення користувацького звіту з нуля;
- **«Мастер отчётов»** – використання «Майстру звітів» для створення звіту;
- **«Автоотчёт в столбец»** – створення автозвіту (обраний за умовчуванням автозвіт містить списки імен полів та змісту цих полів, які розташовані один під одним. Таке розташування не зовсім є зручним для звітів), в якому поля розташовані один під одним;
- **«Автоотчёт ленточный»** – створення автозвіту, в якому поля розташовані один за одним. Такий звіт корисний в тому випадку, якщо б він базувався на запиті, який містить лише декілька полів, та вони вміщуються на екрані та на сторінці при друку;
- **«Мастер диаграмм»** – відображає на екрані «Майстер діаграм», який використовується для створення графіків;
- **«Почтовые наклейки»** – виводить на екран вікно «Создание почтовых наклеек», за допомогою якого можна створювати наклейки або інші етикетки.

Питання щодо самоконтролю:

1. Що таке «Звіт»?
2. Для чого призначені звіти?
3. Яким способом можна створити новий звіт?
4. Як називається звіт, в якому поля розташовані одне під одним?
5. Для чого призначений спосіб створення звітів «Поштові наклейки»?

Тема «Відкриті системи»

(2 години)

Мета: дізнатися про концепцію відкритих систем.

Реальне поширення архітектури "клієнт-сервер" стало можливим завдяки розвитку та широкому впровадженню в практику концепції відкритих систем.

Основним змістом підходу відкритих систем є спрощення комплексування обчислювальних систем за рахунок міжнародної та національної стандартизації апаратних і програмних інтерфейсів. Головною спонукальною причиною розвитку концепції відкритих систем стали повсюдний перехід до використання локальних комп'ютерних мереж і ті проблеми комплексування апаратно-програмних засобів, що їх спричинив цей перехід. У зв'язку з бурхливим розвитком технологій глобальних комунікацій відкриті системи набувають ще більшого значення і масштабності.

Ключовою фразою відкритих систем, спрямованою в бік користувачів, є незалежність від конкретного постачальника. Орієнтуючись на продукцію компаній, що дотримуються стандартів відкритих систем, споживач, який купує будь-який продукт такої компанії, не потрапляє до неї в рабство. Він може продовжити нарощування потужності своєї системи шляхом придбання продуктів будь-якої іншої компанії, що дотримується стандартів. Причому це стосується як апаратних, так і програмних засобів і не є необґрунтованою декларацією. Реальну можливість незалежності від постачальника перевірено у вітчизняних умовах.

Практичною опорою системних і прикладних програмних засобів відкритих систем є стандартизована операційна система. Нині такою системою є UNIX. Фірмам-постачальникам різних варіантів ОС UNIX у результаті тривалої роботи вдалося дійти згоди про основні стандарти цієї операційної системи. Зараз усі

поширені версії UNIX в основному сумісні в частині інтерфейсів, що надаються прикладним (а в більшості випадків і системним) програмістам. Як здається, незважаючи на появу системи Windows NT, що претендує на стандарт, саме UNIX залишиться основою відкритих систем у найближчі роки.

Технології та стандарти відкритих систем забезпечують реальну і перевірену практикою можливість виробництва системних і прикладних програмних засобів із властивостями мобільності (portability) та інтероперабельності (interoperability). Властивість мобільності означає порівняльну простоту перенесення програмної системи в широкому спектрі апаратно-програмних засобів, що відповідають стандартам. Інтероперабельність означає спрощення комплексування нових програмних систем на основі використання готових компонентів зі стандартними інтерфейсами.

Використання підходу відкритих систем вигідне і виробникам, і користувачам. Насамперед відкриті системи забезпечують природне розв'язання проблеми поколінь апаратних і програмних засобів.

Виробники таких засобів не змушені розв'язувати всі проблеми заново; вони можуть принаймні тимчасово продовжувати комплексувати системи, використовуючи наявні компоненти.

Зауважимо, що при цьому виникає новий рівень конкуренції. Усі виробники зобов'язані забезпечити деяке стандартне середовище, але змушені домагатися його якомога кращої реалізації. Звісно, через якийсь час наявні стандарти почнуть відігравати роль стримування прогресу, і тоді їх доведеться переглядати.

Перевагою для користувачів є те, що вони можуть поступово замінювати компоненти системи на більш досконалі, не втрачаючи працездатності системи. Зокрема, у цьому криється розв'язання проблеми поступового нарощування обчислювальних, інформаційних та інших потужностей комп'ютерної системи.