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

Конспект лекцій

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

Конспект лекцій з дисципліни «Системи управління базами даних» для підготовки фахового молодшого бакалавра за спеціальністю 029 «Інформаційна, бібліотечна та архівна справа».

Укладач: викладач вищої категорії ФКПАІТ ОНТУ Юлія БУРМАКІНА

Конспект лекцій затверджено 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

Зміст **Стор.**

Блок змістових модулів №1

Основи теорії баз даних та систем управління базами даних

Змістовий модуль 1.1 - Основи проектування баз даних

Тема 1.1.1 Вступ. Основні поняття БД та СУБД.

(2 години)

Мета: засвоєння головних понять, пов'язаних з базами даних та системами управління базами даних.

.Структура заняття:

I. Організаційний момент:
   a. Готовність групи до заняття;
   b. Психоемоційний настрій;
   c. Перевірка присутніх.

II. Повідомлення теми та мети заняття;

III. Виклад нового матеріалу:

План:

1. Поняття «бази даних» та «системи управління базами даних».
2. Призначення бази даних та системи управління базами даних.
3. Попередники баз даних.
4. Приклади баз даних.

IV. Узагальнення та систематизація знань;

V. Підведення підсумків занять;

VI. Домашнє завдання:

1. Ознайомитись з теоретичними відомостями теми 1.1.1.
2. Вивчити основні поняття лекції.
3. Дати відповіді на контрольні питання.

Контрольні питання:

1. Що таке «база даних»?

2. Яку роль виконують бази даних в теперішньому сучасному житті?

3. Навести приклади баз даних.

4. Хто або що було попередником баз даних?

5. Які переваги використання підходу з використанням баз даних?

6. Що таке «Система управління базами даних»?

7. Навести приклади систем управління базами даних.

Поява баз даних стала самим важливим досягненням в галузі програмного забезпечення. Бази даних лежать в основі інформаційних систем та це головним чином змінило характер роботи багатьох організацій. З моменту своєї появи технологія баз даних стала захоплюючою областю діяльності, а також каталізатором багатьох значних досягнень в області програмного забезпечення.

Бази даних стали невід'ємною частиною нашого повсякденного життя.

Наприклад, доступ до бази даних потрібен при покупці продуктів в супермаркеті, коли касир з покупок зчитує штрих-код. При цьому ручний сканер, пов'язаний з додатком бази даних, використовує штрих-код для пошуку ціни обраного предмету в базі даних всіх товарів. Потім програма віднімає кількість всіх тільки що проданих товарів з товарних запасів та відобразить на касовому апараті їх коштовність. Якщо запаси опустяться нижче визначеного деякого рівня, то в такому випадку система зможе автоматично спрямувати замовлення на поставку додаткової кількості цього товару.

Якщо при покупці використовується кредитна картка, касир повинен перевірити присутність кредитних коштів. Це можливо зробити за телефоном або автоматично за допомогою спеціального зчитуючого пристрою, який пов'язаний з комп'ютером. В будь-якому випадку при цьому використовується база даних, яка містить відомості про покупки, здійснювані за допомогою цієї кредитної картки. За номером кредитної картки спеціальний додаток звіряє ціну покупаємих в дану мить та куплених на протязі цього місяця товарів з кредитним лімітом. Після підтвердження припустимості такої покупки всі відомості про отримані товари заносяться в базу даних. Але ще до отримання підтвердження припустимості покупки додаток бази даних повинен переконатися в тому, що дана кредитна картка не знаходиться в переліку вкрадених або утрачених.

Коли при плануванні відпустки Ви звертаєтесь в туристичне агентство, співробітник цього агентства за Вашим запитом продивляється базу даних з відомостями про всі наявні путівки та розклад польотів. При бронюванні якої-небудь путівки система бази даних повинна виконати всі потрібні для цього дії. В даному випадку потрібно бути впевненим, що два різних співробітника агентства не бронюють одну й ту ж саме путівку або для даного рейсу не забронювані міста за межами припустимої кількості. Наприклад, якщо в літаку деякого рейсу залишилось тільки одне вільне місце та два співробітники туристичного агентства спробують його забронювати, то система повинна коректно опрацювати цю ситуацію та дозволити забронювати останнє місце тільки одному

співробітнику, відправивши іншому співробітнику повідомлення про відсутність вільних місць.

В будь-якому навчальному закладі може існувати база даних з інформацією про студентів, відвідуємих ними курсах, отримуємих стипендіях, вже пройдених та вивчаємих в теперішній час дисциплінах, а також про підсумки здачі різних екзаменів. Крім цього, може також підтримуватися база даних з інформацією про прийом студентів в наступному році, а також база даних з особистими даними про співробітників та відомості про їх заробітну плату для бухгалтерії.

Попередником бази даних була файлова система, тобто набір додатків, які виконували окремі необхідні користувачу операції, наприклад, такі як створення звітів. Кожна програма визначала та керувала своїми власними даними.

Хоча файлова система була значним досягненням у порівнянні з ручною картотекою, її використання все ж таки було пов'язано з великими проблемами, які взагалі були пов'язані з надмірністю даних та залежністю програм від даних.

Для підвищення ефективності роботи необхідно було використовувати новий підхід, а саме базу даних (**D**ata**b**ase) та систему управління базою даних (**D**ata**b**ase **M**anagement **S**ystem - **DBMS**).

База даних – це використовуємий набір логічно пов'язаних даних (та опис цих даних), призначенних для задовільнення інформаційних потреб організації.

База даних – це єдине, велике сховище даних, яке одноразово визначається, а потім використовується одночасно багатьма користувачами з різних підрозділів.

Замість розрізнених файлів з надмірними даними, тут всі дані зібрані разом з мінімальною часткою надмірністю. База даних вже не належить деякому одному відділу, а є загальним корпоративним ресурсом. Вона зберігає не тільки робочі дані цієї організації, але й їх опис. За цією причиною бази даних ще називають **набіром інтегрованих записів з самоописом.**

У сукупності опис даних називають **системним каталогом** або **словником даних**, а самі елементи опису називають **метаданими**, тобто „даними про дані". Саме наявність самоопису даних в базі даних забезпечує в ній незалежність між програмами та даними.

В підході з використанням бази даних структура даних відокремлена від додатків та зберігається в базі даних. Додавання нових структур даних або зміна існуючих ніяким чином не впливає на додатки, за умови що вони не залежать безпосередньо від змінних компонентів. Наприклад, додавання нового поля в запис або створення нового файлу

ніяким чином не вплине на роботу існуючих додатків. Але знищення поля з використовуємого додатком файлу вплине на цей додаток, а тому його також необхідно буде відповідним чином модифікувати.

Система управління базою даних (СУБД) – це програмне забезпечення, за допомогою якого користувачі мають змогу визначати, створювати та підтримувати базу даних, а також здійснювати до неї контролюємий доступ.

Тема 1.1.2 Функції СУБД

(2 години)

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

Структура заняття:

I. Організаційний момент:
   a. Готовність групи до заняття;
   b. Психоемоційний настрій;
   c. Перевірка присутніх.

II. Повідомлення теми та мети заняття;

III. Виклад нового матеріалу:

План:

1. Перелік функцій, які повинні бути реалізовані в кожній СУБД.
2. Призначення функцій СУБД.

IV. Узагальнення та систематизація знань;

V. Підведення підсумків занять;

VI. Домашнє завдання:
   1. Ознайомитись з теоретичними відомостями теми 1.1.2.
   2. Вивчити основні поняття лекції.
   3. Дати відповіді на контрольні питання.

Контрольні питання:

1. Хто запропонував перелік сервісів, які повинні бути реалізовані в будь-якій повномасштабній СУБД?
2. Якими функціями володіє кожна система управління базами даних?
3. Яка функція є самою фундаментальною функцією СУБД?
4. Що означає поняття «транзакції» при роботі з базою даних? Детально розписати все про транзакції.

5. До якої функції належить використання принципу «Все або нічого»?
6. Яка функція СУБД гарантує коректне відновлення бази даних при паралельному виконанні операцій відновлення багатьма користувачами?
7. В якій функції СУБД забезпечується механізм відновлення бази даних та повернення її в несуперечливий стан?
8. З якою метою треба приховувати інформацію, розміщену в базі даних, від інших користувачів?
9. Завдяки чому здійснюється контроль доступу до даних бази даних?
10. Що таке «цілісність бази даних» та в чому вона зображується?

Е. Ф. Кодд (французький математик) запропанував перелік сервісів, які повинні бути реалізовані в будь-якій повномасштабній СУБД:

1. **<u>зберігання, вилучення та відновлення даних</u>**

СУБД повинна надавати користувачам можливість зберігати, вилучати та відновлювати дані в базі даних. Це є найбільш фундаментальною функцією СУБД. Засіб реалізації цієї функції повинен приховувати від кінцевого користувача внутрішні деталі фізичної реалізації системи (наприклад, файлову організацію чи структури зберігання, які використовуються).

2. **<u>підтримка транзакцій</u>**

СУБД повинна мати механізм, який гарантує виконання або всіх операцій відновлення даної транзакції, або жодної з них.

Транзакція уявляє собою набір дій, які виконуються окремими користувачами або прикладною програмою з метою доступу чи зміни змісту бази даних.

Прикладами простих транзакцій може бути додавання в базу даних відомостей про співробітника. Відновлення відомостей про заробітну плату деякого співробітника чи знищення відомостей про деякий об’єкт нерухомості.

Прикладом найбільш важкої транзакції може бути знищення відомостей про співробітника з бази даних та передача відповідальності за всі керовані ним об’єкти нерухомості іншому співробітнику. У цьому випадку в базу даних необхідно буде ввести одночасно декілька змін.

Якщо під час виконання транзакції відбудеться збій, наприклад, із-за виходу зі строю комп’ютеру, база даних потрапить в суперечливий стан, так як деякі зміни вже

будуть здійснені, а решта – ще ні. Тому всі часткові зміни не будуть збережені для повернення бази даних в колишній несуперечливий стан.

ACID – ***вимоги*** гарантують правильність та надійність роботи системи.

Atomic (атомарність) - транзакція не може бути виконана частково, або все, або нічого.

Consistency (узгоджуванність) - після виконання транзакції всі дані повинні знаходитися в узгоджуваному стані. Тобто транзакція не руйнує взаємної узгоджуваності даних.

Isolation (ізольованність) - означає, що транзакції, які конкурують за доступ до бази даних, фізично обробляються послідовно, ізольовано одна від одної.

Durability (стійкість) - після завершення транзакції зміни, які були внесені, залишуться незмінними.

ACID - фундаментальні властивості систем обробки транзакцій

3. **<u>сервіси управління паралельністю</u>**

СУБД повинна мати механізм, який гарантує коректне відновлення бази даних при паралельному виконанні операцій відновлення багатьма користувачами.

Головною метою створення та використання СУБД є те, що багато користувачів мають змогу здійснювати паралельний доступ до даних, які спільно опрацьовуються. Паралельний доступ порівняно легко організувати, якщо всі користувачі виконують лише читання даних, так як в цьому випадку вони не зможуть зашкодити одне одному. Але, коли два або більш користувачів одночасно отримують доступ до бази даних, конфлікт з небажаними наслідками легко може виникнути, наприклад, якщо хоча б один з них спробує відновити дані. СУБД повинна гарантувати те, що при одночасному доступі до бази даних багатьох користувачів схожих конфліктів не відбудеться. Для цього треба використовувати термін «блокування».

4. **<u>сервиси відновлення</u>**

СУБД повинна надати засоби відновлення бази даних на випадок будь-яких ушкоджень або знищень.

При збої транзакція повинна бути повернена в несуперечливий стан. Такий збій може трапитися у випадку виходу зі строю системи чи запам’ятовуючого пристрою, помилки апаратного чи програмного забезпечення, які можуть привести до зупинки СУБД. Крім цього, користувач має змогу знайти помилку під час виконання транзакції та вимагати її відміни.

У всіх цих випадках СУБД повинна забезпечити механізм відновлення бази даних та повернення її в несуперечливий стан.

Кожна база даних має журнал транзакцій, у якому фіксуються всі зміни даних, зроблені в кожній із транзакцій.

Журнал транзакцій - це важлива складова бази даних. Якщо система дасть збій, цей журнал допоможе повернути базу даних в узгоджений стан.

5. **сервіси контролю доступу до даних**

СУБД повинна мати механізм, який гарантує можливість доступу до бази даних лише санкціонованих користувачів.

Іноді необхідно приховувати деякі відомості, які зберігаються в базі даних від інших користувачів. Наприклад, менеджерам відділень компаній можна було б надавати інформацію, яка пов'язана з заробітною платою співробітників, але бажано приховати її від інших користувачів. Крім того, базу даних необхідно було б захистити від будь-якого несанкціонованого доступу.

Термін *«безпека»* відноситься до захисту бази даних від навмисного чи випадкового несанкціонованого доступу. Припускається, що СУБД забезпечує механізми подібного захисту даних.

6. **служби підтримки цілісності даних**

СУБД повинна володіти інструментами контролю за тим, щоб дані та їх зміни відповідали вказаним правилам.

Цілісність бази даних визначає коректність та несуперечливість зберігаємих даних. Вона може розглядатися як ще один тип захисту бази даних.

Крім того, що дане питання пов'язане з забезпеченням захисту, воно має більш широкий зміст, оскільки цілісність пов'язана з якістю самих даних. Цілісність звичайно зображується у вигляді обмежень або правил зберігання несуперечливості даних, які не повинні порушуватися в базі даних.

Наприклад, можна вказати, що співробітник немає змогу відповідати одночасно більш ніж за 10 об'єктів нерухомості. В даному випадку при спробі закріпити черговий об'єкт нерухомості за деяким співробітником, СУБД повинна перевірити, чи не перевищений встановлений ліміт, та у випадку виявлення подібного перевищення закріпити новий об'єкт за іншим співробітником.

Змістовий модуль 1.2 - Моделі даних

Тема 1.2.1 Ієрархічна та мережева модель даних

(2 години)

Мета: засвоєння поняття «модель даних», її призначення, види моделей даних, властивості ієрархічної та мережевої моделей даних.

Структура заняття:

I. Організаційний момент:
   a. Готовність групи до заняття;
   b. Психоемоційний настрій;
   c. Перевірка присутніх.

II. Повідомлення теми та мети заняття;

III. Виклад нового матеріалу:

План

1. Поняття «Модель даних» та її призначення. Види моделей даних.
2. Призначення ієрархічної моделі даних. Головні інформаційні одиниці та властивості ієрархічної моделі даних.
3. Недоліки ієрархічної моделі даних.
4. Призначення мережевої моделі даних. Головні інформаційні одиниці мережевої моделі даних.
5. Переваги мережевої моделі даних.

IV. Узагальнення та систематизація знань;

V. Підведення підсумків занять;

VI. Домашнє завдання:
   1. Ознайомитись з теоретичними відомостями теми 1.2.1.
   2. Вивчити основні поняття лекції.
   3. Дати відповіді на контрольні питання.

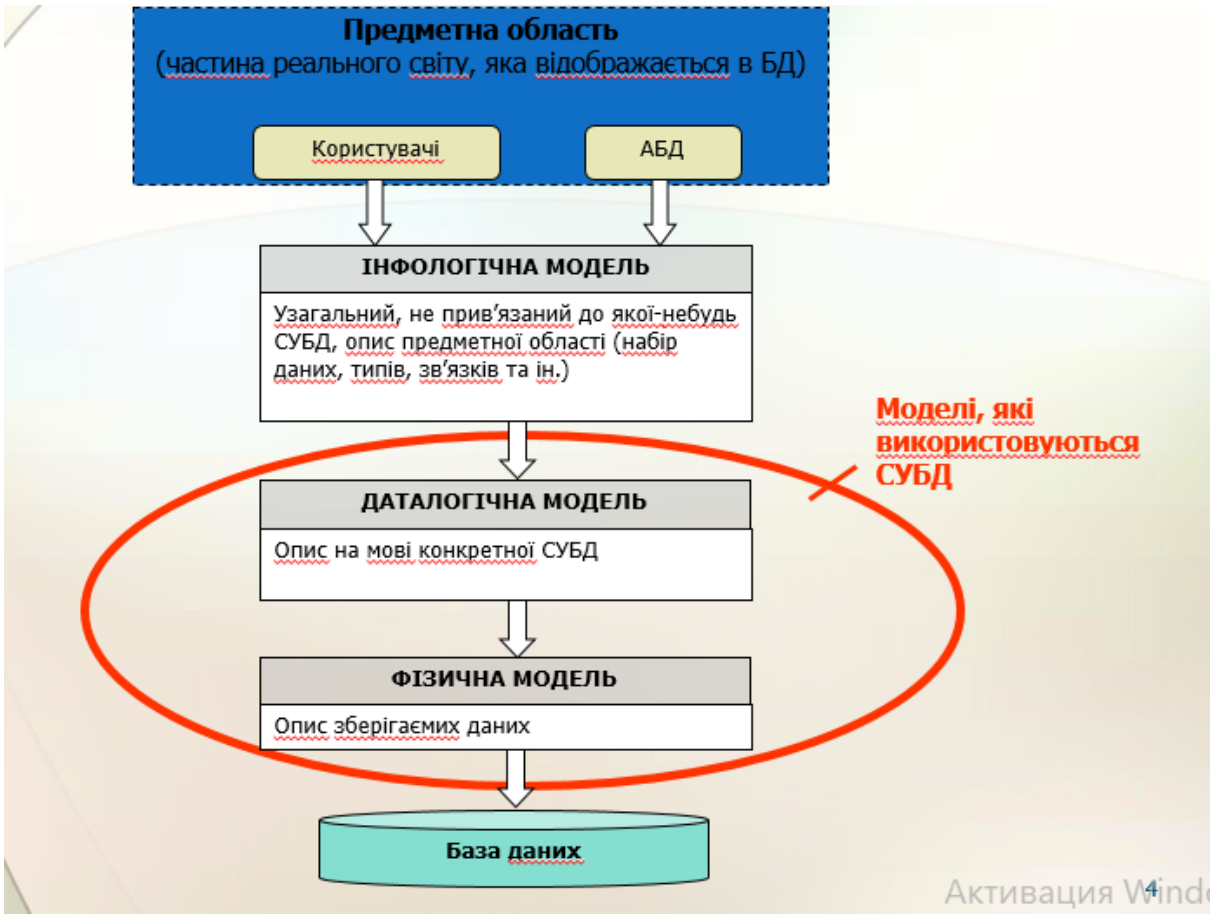
Контрольні питання:

1. Що розуміється під поняттям «модель даних»?
2. Які існують моделі даних?
3. Що таке «ієрархія»?
4. Що є головними інформаційними одиницями ієрархічної моделі даних?
5. Якими властивостями володіє ієрархічна модель даних?

6. Що розуміється під поняттям «елемент даних», «запис» та «набір даних»?
7. Чим відрізняється мережева модель даних від ієрархічної моделі даних?
8. Якими перевагами володіє мережева модель даних?

Дані, які зберігаються в базі даних, повинні бути організовані за визначеними правилами. Організація даних та способи доступу до них, які забезпечуються конкретною СУБД, називається **моделю даних**.

Рівні моделей даних (послідовність розробки БД)



Існуючі бази даних побудовані на основі ієрархічної, мережевої, реляційної та об'єктно-орієнтованої моделей даних або їх підмножин.

Ієрархічна модель даних

Ієрархія - розташування елементів від найвищого елементу до найнижчого.

Ієрархічна модель даних (ІМД) представляє собою древовидну (ієрархічну) структуру. В вершинах дерева знаходяться сукупності властивостей даних, які описують деякий об'єкт. В термінології ІМД ці сукупності називаються **сегментами**. Сегмент, у якого немає вище за нього рівня ієрархії, називається **кореневим (головним)**, а інші знаходяться на найнижчих рівнях - **підпорядковані**. Сегменти на найвищих рівнях мають назву - **логічно початкові**, **батьківські** або **предки**. Сегменти на найнижчих рівнях мають назву – **логічно підпорядковані**, **дочірні** або **потомки**.

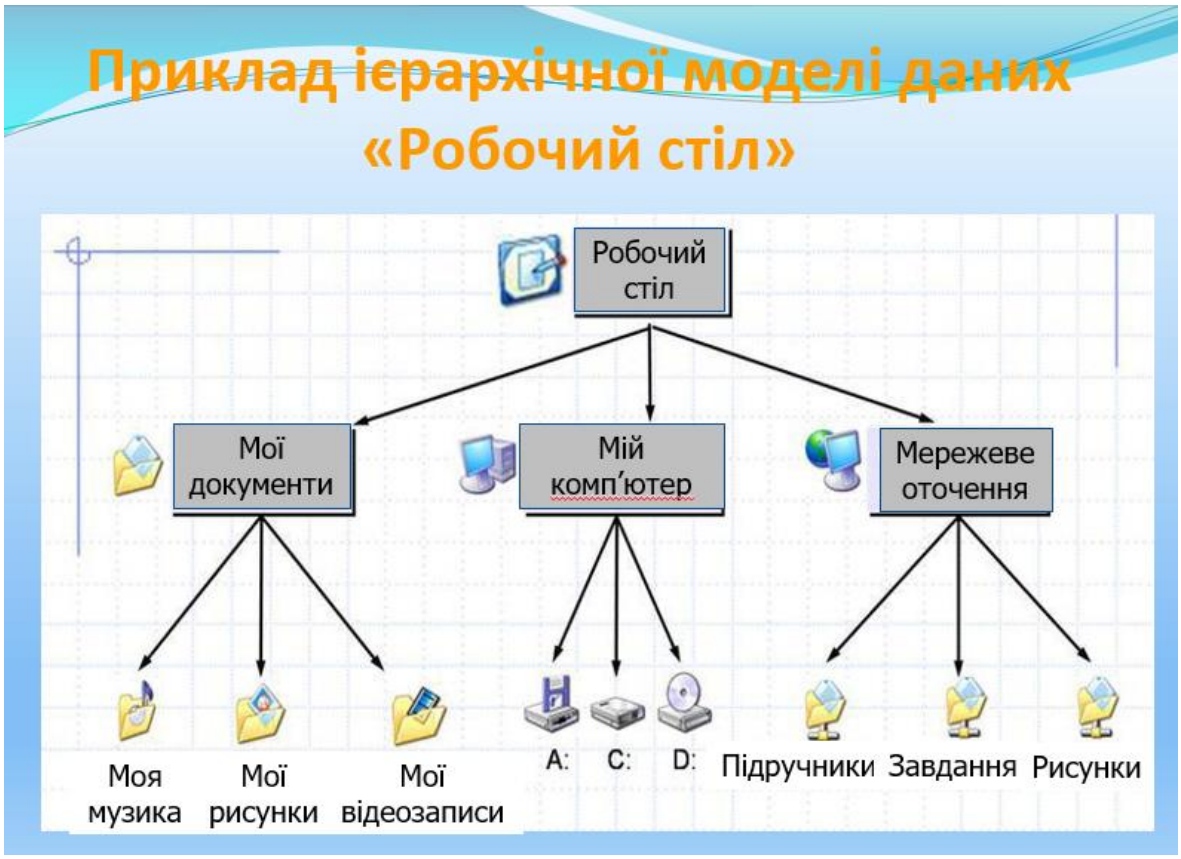
Головними інформаційними одиницями моделі є:

1) **поле** - мінімальна нерозподілена одиниця даних (прізвище, посада);
2) **сегмент (запис)** - сукупність полів, яка характеризує об'єкт.

Для відмінності окремих сегментів кожен тип сегменту має ключ.

Ключ - сукупність елементів, які значним чином визначають сегмент (номер співробітника, номер студентського квитка, номер залікової книжки, ідентифікаційний код, номер паспорту).

Прикладом ієрархічної БД є робочий стіл.



Ієрархічною БД є файлова система, яка складається з кореневої директорії, в якій є ієрархія піддиректорій та файлів.

Сукупність конкретних значень полів сегмента називається **екземпляром сегмента** або **записом**: м. Київ, вул. Перемоги, 3; 18-67-53. З цього можна сказати, що один блок на схемі відображає множину фактичних записів (екземплярів сегмента).

Пошук необхідного запису в ієрархічній БД завжди починається від кореневого запису. Наприклад, необхідно обрати запис з характеристиками конкретної партії товарів (кількість, колір, розмір, вартість та т. д.). Треба послідовно вказати філіал, магазин, склад, де ці товари знаходяться. Для спрощення пошуку записи кожного сегмента підпорядковуються за даними деякого поля.

Головні властивості ієрархічної моделі даних

1. Кожний з підлеглих сегментів пов'язаний лише з одним сегментом рівня ієрархії, який знаходиться вище нього. Зв'язки між сегментами одного уровня не допускаються.

2. Між сегментами двох рівнів можуть підтримуватися лише зв’язки «один до багатьох» (один філіал - багато магазинів, один склад - багато товарів) або «один до одного» (один філіал - один директор).

Недоліки ієрархічної моделі даних:

1. ***Асиметричність запитів***. В цьому можна впевнитися, порівнюючи два запити. За допомогою першого з них обирається запис з інформацією про партії товарів, для якої відомі назви філіала, магазина та склада. В рамках другого запиту за відомостями про партії товарів треба визначити назву філіала, в якому вона знаходиться.

2. ***Складність змін***. Наприклад, при закритті деякого складу товари перерозподіляються поміж іншими складами. Ця операція потребує кардинальної зміни структури БД.

3. ***Жосткість структури***, яка не дозволяє врахувати в БД окремі реальні ситуації. Наприклад, товари, які зберігаються на одному складі, призначені декільком магазинам.

4. ***Складність експлуатації***. Робота з ієрархічними БД потребує значної кваліфікації користувачів в області програмування. В частності, в СКБД IMS фірми IBM для опису загальної схеми БД та блоку зв'язка кожного користувача з БД використовувалась мова програмування Assembler. Для відбору даних з БД - спеціалізована мова DL/1, для обробки отриманої інформації – мови PL/1 або Кобол.

Перелічені причини призвели до того, що ієрархічна модель даних в теперішній час практично не використовується.

Мережева модель даних

Одним із основоположників мережевої моделі даних є американський учений Чарльз Бахман. В 1973 р. за керування роботою Data Base Task Group (робоча група по базам даних, США) він був нагороджений премією Тьюрінга – найпрестижнішою премією в галузі гнформатики.

Приклад: СУБД Integrated Database Management System (IDMS) компанії Cullinet Software, Inc. (1972 р.)

Мережева модель бази даних являє собою сукупність об'єктів різного рівня, проте схема зв'язків між об'єктами може бути будь-якою.

Мережеві БД більш гнучкі: немає явно вираженого головного елемента й існує можливість встановлення горизонтальних зв'язків. Наприклад, організація інформації в Інтернеті (WWW).

Елементами цієї моделі є: **рівень, вузол та зв'язок**.

В мережевій моделі даних не існує підпорядкованості рівнів «один до одного».

В мережевій моделі даних не накладаються жодні обмеження на кількість зв'язків, які входять до однієї вершини. Тобто, зв'язки можна встановлювати не лише між вузлами сусідніх за підлеглістю рівнів, але й різних рівнів.

Приклади мережевої моделі даних

Перевагами мережевої моделі даних у порівнянні з ієрархічною моделю є її гнучкість, можливість утворення довільних зв'язків, економічність.

Недоліки - висока складність, яка практично виключає можливість її експлуатації користувачами, які не є спеціалістами в області інформаційних технологій, слабкий контроль цілісності зв'язків між об'єктами БД.

Тема 1.2.2 «Реляційна модель даних»

(2 години)

Мета: засвоєння поняття «реляційна модель даних», її призначення, властивості реляційної моделі даних.

Структура заняття:

I. Організаційний момент:
   a. Готовність групи до заняття;
   b. Психоемоційний настрій;

c. Перевірка присутніх.

II. Повідомлення теми та мети заняття;

III. Виклад нового матеріалу:

План

1. Структура реляційних даних.

2. Властивості реляційної таблиці.

IV. Узагальнення та систематизація знань;

V. Підведення підсумків занять;

VI. Домашнє завдання:

1. Ознайомитись з теоретичними відомостями теми 1.2.2.
2. Вивчити основні поняття лекції.
3. Дати відповіді на контрольні питання.

Контрольні питання:

1. Ким була запропонована реляційна модель даних?

2. Чим реляційна модель даних відрізняється від інших моделей даних?

3. Чому реляційна модель даних стала найпоширенішою?

4. Які головні поняття використовуються при роботі з реляційною моделю даних? Дати їм визначення.

5. Яка характеристика реляційної моделі даних визначається кількістю атрибутів, які містяться в базі даних?

6. Для чого в базі даних треба створювати первинний ключ?

7. Навіщо при роботі з пов’язаними таблицями використовується зовнішній ключ?

8. Які вимоги надаються до таблиць реляційної моделі даних?

Враховуючи всю складність ієрархічної та мережевої моделей, найбільше поширення отримала **реляційна модель даних** (**relation** - відношення - математичний термін для визначення непідпорядкованої сукупності записів одного типу або таблиць визначеного специфічного вигляду).

Вся інформація в реляційній моделі даних організована у вигляді таблиць, які пов’язані між собою за допомогою зв’зків.

До реляційних СУБД можна віднести - MS Access, Firebird, SQL Server, Oracle, DB2.

Реляційні системи беруть початок в математичній теорії множин. Вони були запропоновані наприкінці 1968 р. доктором Е. Ф. Коддом з фірми IBM, який першим

усвідомив, що можна використовувати математику для надання надійної основи та суворості області керування базами даних.

Нечіткість багатьох термінів, які використовуються в сфері обробці даних, змусила Е. Ф. Кодда відмовитися від них та вигадати нові або дати найбільш точні визначення існуючим. Таким чином, він не міг використовувати поширений термін «Запис», який в різних ситуаціях може визначати екземпляр запису, або тип записів, запис у стилі Коболу (який припускає групи, які півторюються) або плоский запис (який їх не припускає), логічний запис або фізичний запис, зберігаємий запис або віртуальний запис та т.д. Замість цього він використовував термін **«Кортеж довжини n»** або просто **«Кортеж»**.

Реляційна модель даних (РМД) покладена в основу більшості сучасних СУБД. Перевагами моделі вважаються простота розміщення даних та зручність їх інтерптетації.

Реляційна модель орієнтована на організацію даних у вигляді таблиць (відношень).

Для РМД існує достатньо суворе теоретичне обгрунтування. Представлення даних у вигляді відношень дозволяє використовувати для обробки даних формальний математичний апарат реляційної алгебри відношень та реляційного числення. Поняття таблиці та відношення з практичної точки зору представляють собою одне й теж саме, тому в подальшому будуть використовуватися обидва ці терміни.

Кожна таблиця реляційної БД має ім’я та рядок заголовків.

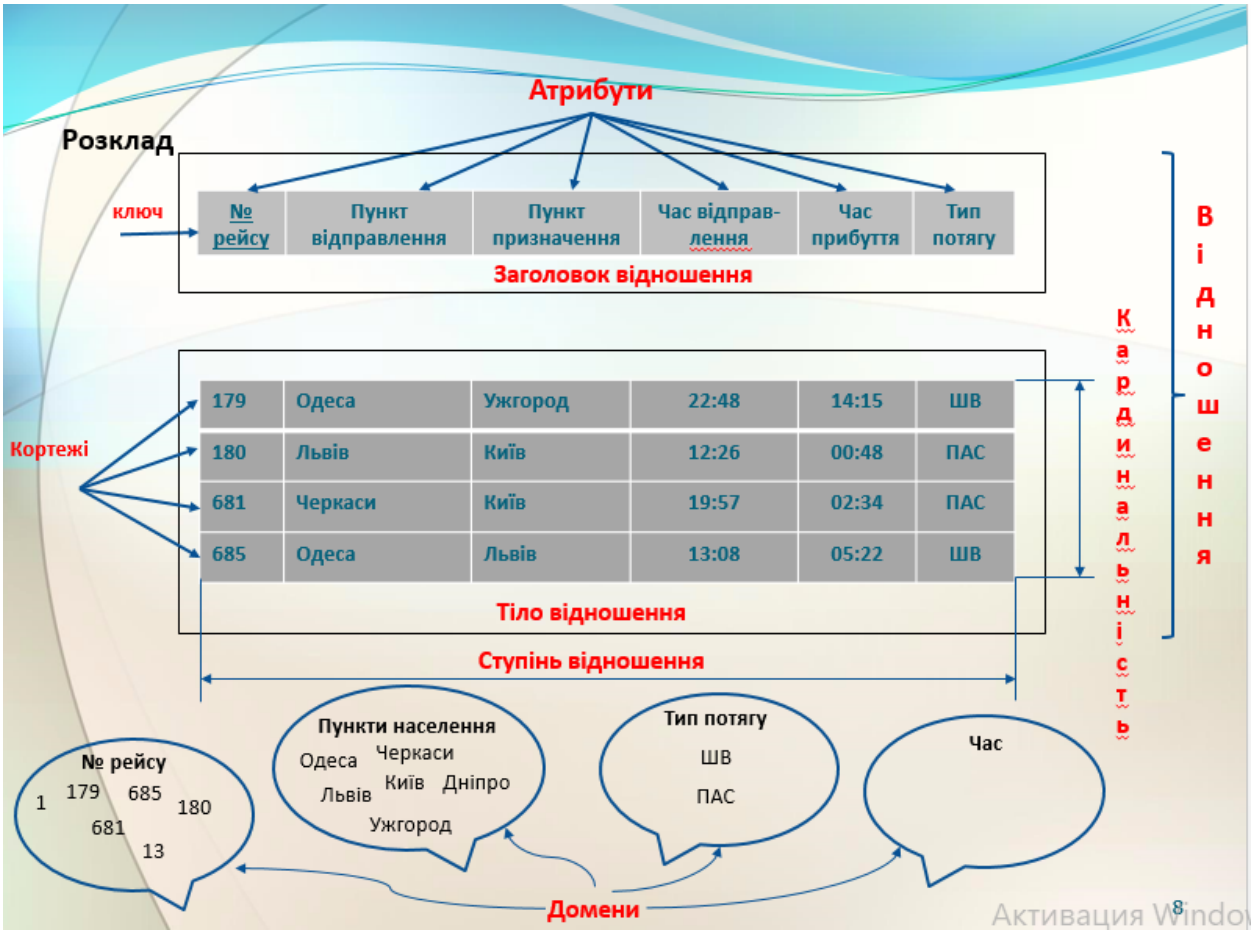
Структура реляційних даних

Відношення - плоска таблиця, яка складається з рядків та стовпців.

В будь-якій реляційній СУБД передбачається, що користувач сприймає базу даних як набір таблиць.

Атрибут - стовпець відношення з власним іменем.

В реляційній моделі відношення використовується для зберігання інформації про об’єкти, які зображуються в базі даних. Відношення має вигляд двомірної таблиці, в якій рядки відповідають окремим записам, а стовпці - атрибутам. При цьому атрибути можуть бути розташовані в будь-якому порядку - незалежно від їх перерозташування відношення залишиться тим самим та матиме той самий зміст.



Кортеж - рядок відношення.

В відношенні «Відділення» кожен рядок містить шість значень, по одному для кожного атрибуту. Кортежі можуть розтошовуватися в будь-якій черзі, при цьому відношення буде залишатися тим самим та мати той самий зміст.

Кортежі називають ***поширенням***, ***станом*** або ***тілом відношення***, яке постійно змінюється.

Ступінь - визначається кількістю атрибутів, яку містить відношення.

Відношення «Відділення» має 6 атрибутів та його ступінь дорівнює 6. Це означає, що кожен рядок таблиці є 6-арним кортежем, тобто кортежем, який містить 6 значень.

Відношення лише з одним атрибутом має ступінь 1 та називається ***унарним відношенням*** (чи 1-арним кортежем). Відношення з двома атрибутами має назву ***бінарне***, відношення з трьома атрибутами - ***тернарне***, а для відношень з більшою кількістю атрибутів використовується термін ***n-арний***.

Кординальність - кількість кортежів, яку містить відношення.

Ця характеристика змінюється при кожному додованні або знищенні кортежів. Кординальність є властивістю тіла відношення та визначається поточним станом відношення в окрему мить.

Для того щоб відрізняти один рядок від іншого використовується поняття *первинного ключа*.

Первинним ключем (Primary key) називається атрибут відношення, значення якого унікальним чином ідентифікує кожен кортеж відношення. У відношення може бути лише один первинний ключ.

Припустимо, що ми маємо два відношення в базі даних: перше відношення містить дані про продавців, а друге - про покупців. У відношенні з даними про покупців є атрибут, в якому записане ім'я продавця, який продав товар.

Було б дуже непродуктивно писати по декілька разів теж саме ім'я продавця. Тут і виникає питання: може існує інша, найвигідніша можливість заповнення цього поля? Зараз ми й познайомимося ближче з поняттям «зовнішнього ключу».

Атрибут одного відношення, значення в якому співпадають зі значеннями атрибуту, який є первинним ключем іншого відношення, має назву **зовнішній ключ (Foreign key).**

Продавці

Код1	ПІБ
1	Симонова С.І.
2	Ветрова В.Р.

Покупці

Код2	ПІБ
1	Петров А.В
2	Коваль М.Л.

Угода

Код	Код1	Код2	Сума
1	2	2	300
2	1	2	500

В цьому випадку Код1 та Код2 в відношенні «Угоди» є зовнішніми ключами, які відповідають первинним ключам в відношенні «Продавці» (ключове поле Код1) та в відношенні «Покупці» (ключове поле Код2).

Поняття **тип даних** в реляційній моделі даних повністю адекватне поняттю типу даних в мовах програмування.

Звичайно в сучасних реляційних базах припускається зберігання символьних, числових даних (таких як «гроші»), а також спеціальних «темпоральних» даних (дата, час, часовий інтервал).

Домен - набір припустимих значень одного або декількох атрибутів.

Кожен атрибут реляційної бази даних визначається на деякому домені. Домени можуть відрізнятися для кожного з атрибутів, але два або більша кількість атрибутів можуть визначатися на тому ж самому домені.

Атрибут	Ім’я домену	Зміст домену	Визначення домену
Номер_ відділення	Номер_ відділення	Безліч усіх припустимих номерів відділень компанії	Символьний; розмір 3; діапозон „В1 – В99”
Місто	Назва_міста	Безліч усіх назв міст в Україні	Символьний; розмір 15.
Поштовий_ індекс	Поштовий_ індекс	Безліч усіх почтових кодів в Україні	Цілий
Стать	Стать	Визначення статі людини	Символьний; розмір 1; Значення „ч” чи „ж”
Дата_ народження	Дата_ народження	Усі можливі значення дат народження співробітників компанії	Дата; діапозон від 1 січня; формат дд-мм-рррр.
З_П	З_П	Усі можливі значення з_п співробітника	Грошовий; 7 цифр

Поняття **домену** має велике значення, так як завдяки йому користувач має централізовано визначити глузд та джерело значень, які можуть отримувати атрибути. В підсумку при виконанні реляційної операції системі стає доступне більше інформації, що дозволяє їй уникнути семантично некоректних операцій.

Наприклад, безглуздо порівнювати назву вулиці з номером телефону, навіть якщо для двох цих атрибутів визначеннями доменів є символьні рядки.

Властивості реляційної таблиці:

1. Таблиця являє собою двовимірну структуру, що складається з рядків і стовпців
2. Кожен рядок таблиці (кортеж) являє собою окрему сутність усередині набору сутностей
3. Кожен стовпець таблиці являє собою атрибут, і в кожного стовпця є своє ім'я
4. На кожному перетині рядка і стовпця є єдине значення
5. Кожна таблиця повинна мати атрибут, що унікально ідентифікує кожен рядок
6. Усі значення в стовпці мають відображатися в однаковому форматі. Наприклад, якщо атрибуту присвоюється формат цілого, то всі значення в стовпчику, що представляє цей атрибут, мають бути цілими
7. Кожен стовпець має певний діапазон значень, званий доменом атрибута (attribute domain)
8. Порядок слідування рядків і стовпців для СУБД не суттєвий.

Тема 1.2.3 Реляційна цілісність БД. З'язки в реляційній моделі даних

(2 години)

Мета: засвоєння двох важливих правил цілісності даних, які існують в реляційній моделі даних; визначення типів зв'язків, які підтримуються в реляційній моделі даних.

Структура заняття:

I. Організаційний момент:
   a. Готовність групи до заняття;
   b. Психоемоційний настрій;
   c. Перевірка присутніх.

II. Повідомлення теми та мети заняття;

III. Виклад нового матеріалу:

План

1. Поняття «цілісність бази даних» та її призначення.
2. Цілісність сутностей.
3. Цілісність за посиланням.
4. Поняття зв'язків в реляційній базі даних. Типи зв'язків в реляційній базі даних.
5. Приклади зв'язків в реляційній базі даних.

IV. Узагальнення та систематизація знань;

V. Підведення підсумків занять;

VI. Домашнє завдання:

1. Ознайомитись з теоретичними відомостями теми 1.2.3.
2. Вивчити основні поняття лекції.
3. Дати відповіді на контрольні питання.

Контрольні питання:

1. Що таке цілісність бази даних?
2. Які дві базові вимоги забезпечення цілісності існують в реляційній моделі?
3. Що повинне мати відношення для здійснення унікальності записів?
4. Зі значенням чого повинен співпадати зовнішній ключ?
5. Які вимоги та призначення цілісності на рівні сутності?
6. Які вимоги та призначення цілісності на рівні посилань?
7. Які типи зв'язків існують в реляційній моделі даних?
8. Яку роль виконують первинні та зовнішні ключі при створенні зв'язків в реляційній моделі даних?

9. Які приклади зв’язків Ви можете навести?

Цілісність даних (цілісність відношення) - це механізм підтримки відповідності бази даних, за якого обрані дані завжди дають один і той самий результат і відповідають певним правилам.

В реляційній моделі даних визначено дві базові вимоги забезпечення цілісності:

- цілісність сутностей (категорна сутність)
- цілісність посилань.

Обмеження цілісності

1. **цілісність сутностей (категорна сутність)**, згідно з якою відношення повинне мати первинний ключ (PRIMARY KEY) для здійснення унікальності записів, отже, у таблиці не може бути 2-х однакових об'єктів;
2. **цілісність посилань** ґрунтується на механізмі вторинних ключів (зовнішніх - Foreign Key), сенс якої полягає в тому, що деякому атрибуту (групі атрибутів) одного відношення призначається посилання на первинний ключ іншого відношення, отже закріплюються зв'язки підпорядкованості між цими відношеннями.

Цілісність на рівні сутності

У межах таблиці первинний ключ має бути унікальним, щоб однозначно ідентифікувати кожен рядок. Коли справа йде так, то кажуть, що таблиця проявляє **цілісність на рівні сутності**.

Для забезпечення цілісності на рівні сутності в первинному ключі неприпустимі порожні значення, null (тобто повна відсутність даних).

Правило категорної цілісності: жоден ключовий атрибут будь-якого рядка реляційної таблиці не може мати порожнього значення.

Вимога цілісності на рівні сутності - Всі первинні ключі унікальні й не можуть містити порожнього значення (null).

Призначення цілісності на рівні сутності - Гарантує, що кожна сутність (логічний об'єкт) матиме унікальну ідентифікацію, а значення зовнішнього ключа можуть належним чином посилатися на значення первинного ключа.

Приклад - Рахунок не може мати кілька значень, що дублюються, і не може мати порожнє значення (null). Тобто, всі рахунки унікально ідентифікуються своїм номером.

Цілісність на рівні посилань

Якщо зовнішній ключ містить значення, що збігаються з первинним ключем або порожні значення (null), кажуть, що таблиця (або таблиці), яка використовує такий ключ, проявляє **цілісність на рівні посиланнь**.

Іншими словами цілісність на рівні посилання означає, що в тому разі, якщо зовнішній ключ містить якесь значення, то це значення посилається на наявний дійсний кортеж (рядок) в іншому відношенні.

Вимога цілісності на рівні посилань - Зовнішній ключ може мати або порожнє значення, або значення, що збігається зі значенням первинного ключа у пов'язаній таблиці (Кожне непорожнє значення зовнішнього ключа повинно посилатися на наявне значення первинного ключа).

Призначення цілісності на рівні посилань - Допускається, що атрибут не має відповідного значення, але атрибут не може приймати неприпустимі значення. Виконання правила цілісності на рівні посилання унеможливлює видалення рядка в одній таблиці, де первинний ключ має обов'язкову відповідність зі значенням зовнішнього ключа в іншій таблиці.

Приклад - Клієнту може бути не призначений (ще) торговий агент, але неможливо призначити клієнту неіснуючого агента.

Зв’язки в реляційній моделі даних

У кожному зв'язку одне відношення може виступати як основне, а інше відношення виступає в ролі підлеглого.

Таким чином, один кортеж основного відношення може бути пов'язаний з кількома кортежами підлеглого відношення.

Для підтримки цих зв'язків обидва відносини мають містити набори атрибутів, за якими вони пов'язані.

В основному відношенні це **первинний ключ** відношення, який однозначно визначає кортеж основного відношення.

У підлеглому відношенні для моделювання зв'язку має бути присутнім набір атрибутів, що відповідає первинному ключу основного відношення.

Однак тут цей набір атрибутів уже є **вторинним ключем** або **зовнішнім ключем**, тобто він визначає безліч кортежів відношення, які пов'язані з єдиним кортежем основного відношення.

Типи зв’язків в реляційній моделі даних

- зв'язок "один до одного" (1:1) - кожному значенню сполучного поля відповідає один запис по обидва боки.
- зв'язок "один до багатьох" (1:N) - кожному значенню сполучного поля з одного боку відповідає декілька записів по інший бік.
- зв'язок "багато до багатьох" (M:N) - значення в полях неодноразово зустрічаються у пов'язаних відносинах. Такий тип зв'язку необхідно обходити і розділяти його на зв'язки 1:N (створюється третє відношення, яке пов'язується з початковими відносинами зв'язками 1:N).

Приклади зв’язка «один до одного»:

- у людини може бути лише один чоловік або дружина (традиційне подружжя);
- на факультеті може бути лише один декан, та навпаки, один декан може керувати лише одним факультетом.

Приклади зв’язка «один до багатьох»:

- в групі навчається багато студентів, але кожен з цих студентів навчається лише в одній групі;
- у кожного студента лише один класний керівник, а у класного керівника багато студентів в його класній групі;
- у керівника деякого підприємства багато підлеглих, але у кожного підлеглого може бути лише один керівник;
- у чоловіка може бути декілька дружин;
- у дружини може бути декілька чоловіків.

Приклади зв’язка «багато до багатьох»:

- викладач працює в різних групах, та в тій самій групі працюють різні викладачі;
- у чоловіка може бути декілька дружин та у дружини може бути декілька чоловіків.

Блок змістових модулів 2 – Теорія проектування баз даних

Змістовий модуль 2.1 - Інфологічне моделювання предметної області

Тема 2.1.1 Етапи проектування бази даних. Побудова концептуальної моделі предметної області

(2 години)

Мета: знати 3 етапи, які використовуються при проектуванні бази даних та навчитися створювати високорівневу концептуальну модель даних «Сутність – зв'язок».

Структура заняття:

I. Організаційний момент:
   a. Готовність групи до заняття;
   b. Психоемоційний настрій;
   c. Перевірка присутніх.

II. Повідомлення теми та мети заняття;

III. Виклад нового матеріалу:

План

1. Етапи проектування БД.
2. Концептуальна модель даних або модель «Сутність – зв'язок».
3. Побудова ER-діаграм.

IV. Узагальнення та систематизація знань;

V. Підведення підсумків занять;

VI. Домашнє завдання:
   1. Ознайомитись з теоретичними відомостями теми 2.1.1.
   2. Вивчити основні поняття лекції.
   3. Дати відповіді на контрольні питання.

Контрольні питання:

1. Які етапи виділяють в процесі проектування БД?
2. Яке призначення мають етапи проектування БД?
3. Для чого призначена модель «Сутність – зв’язок»?
4. Що є головними поняттями моделі «Сутність-зв’язок»?
5. Що таке «сутність»? Навести приклади сутностей.
6. Які існують типи сутностей? Як вони зображуються на діаграмах?
7. Що таке «атрибут»? Навести приклади атрибутів.

8. Які існують типи атрибутів? Як вони зображуються на діаграмах?

9. Як зображується на діаграмах атрибут, який є первинний ключем?

10. Для чого потрібні зв'язки між сутностями? Навести приклади зв'язків.

11. Як зображуються на діаграмах зв'язки?

12. Кількість сутностей,охоплених зв'язком – це .....? Продовжити визначення.

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.

Розглянемо більш детально кожен з етапів.

Побудова концептуальної моделі предметної області

В рамках концептуальної моделі інформаційний зміст предметної області виражається деякими абстрактними засобами. Основною вимогою, яка висувається до концептуальної моделі, є вимога адекватного відображення предметної області.

Модель має бути несуперечливою, відображати погляди і потреби всіх користувачів системи. Вона повинна володіти властивістю легкої розширюваності, що забезпечує введення нової інформації.

Розглянемо деякі засоби концептуального моделювання.

ER-модель

ER-модель (Entity-Relationship – сутність - зв'язок) була запропонована П. Ченом в 1976 р. Інформація про зміст предметної області в межах моделі зображується в структурованому графічному вигляді (ER-діаграма).

ER-модель (**E**ntity **R**elationship model) або модель «Сутність – зв’язок» – це високорівнева концептуальна модель даних, яка була розроблена з метою спрощування задач проектування баз даних. Ця модель даних являє собою набір концепцій, які описують структуру бази даних та пов’язані з нею транзакції оновлення та вилучення даних.

Для ER-моделі не існує єдиної стандартизованої системи позначень, тому характеристики ER-діаграм можуть дещо відрізнятися.

Головними поняттями моделі «Сутність - зв’язок» вважаються сутності, атрибути та зв’язки.

Під сутністю в ER-моделі розуміються об’єкт або явище, інформація про яких буде зберігатися в базі даних. При цьому розрізняють **тип сутності** та **екземпляр сутності**.

Під типом сутності розуміють набір однорідних об’єктів, який відображається як єдине ціле.

Під екземпляром сутності розуміється конкретний об’єкт.

На ER-діаграмі сутність зображується прямокутником, в якому вказане його ім’я.

Типи сутностей можна класифікувати як сильні та слабкі.

Слабкий тип сутності - тип сутності, існування якого залежить від якого-небудь іншого типу сутності.

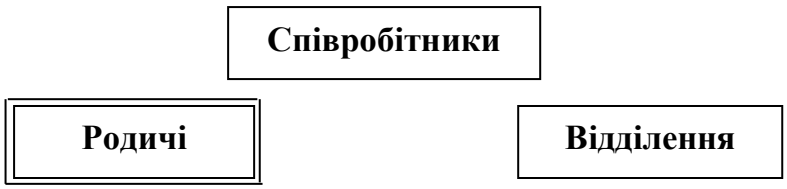
Сильний тип сутності - тип сутності, існування якого не залежить від якого-небудь іншого типу сутності.

Наприклад, сутність **«Родичі»** є сутністю слабкого типу, яка надає відомості про родичів співробітника. Сутність **«Родичі»** не може існувати в цій моделі без наявності сутності **«Співробітники»**.

Прикладами сильних сутностей є сутності **«Співробітники»** та **«Відділення»**.

Слабкі типи сутностей іноді називають ***дочірніми*** (child), ***залежними*** (dependent) або ***підпорядкованими*** (subordinate), а сильні типи сутностей - ***батьківськими*** (parent), ***сутностями-володорями*** (owner) або ***домінантними*** сутностями (dominant).

Кожен сильний тип сутності зображується у вигляді прямокутника з іменем сутності всередині нього, а кожен слабкий тип сутності - у вигляді прямокутника з подвійним контуром.



Сутності мають властивості, які називаються **атрибутами**. Атрибути повинні дозволяти розрізняти екземпляри сутності. На ER-діаграмі атрибути зображуються овалами, в яких вказуються їх імена. Атрибути поєднуються з сутностями прямими лініями.

Атрибути, які однозначно ідентифікують сутність, називаються **ключовими атрибутами**.

Ключові атрибути на ER-діаграмі підкреслюються.

В деяких ситуаціях з декількох простих атрибутів може бути сформований **складовий (складений) атрибут.**

Атрибути поділяються на прості та складові.

Простий атрибут – атрибут, який складається з одного компоненту з незалежним існуванням.

Прості атрибути не можуть бути розподілені на більш дрібні компоненти. Прикладами є атрибути статі («Ч» або «Ж») або «ЗП співробітника».

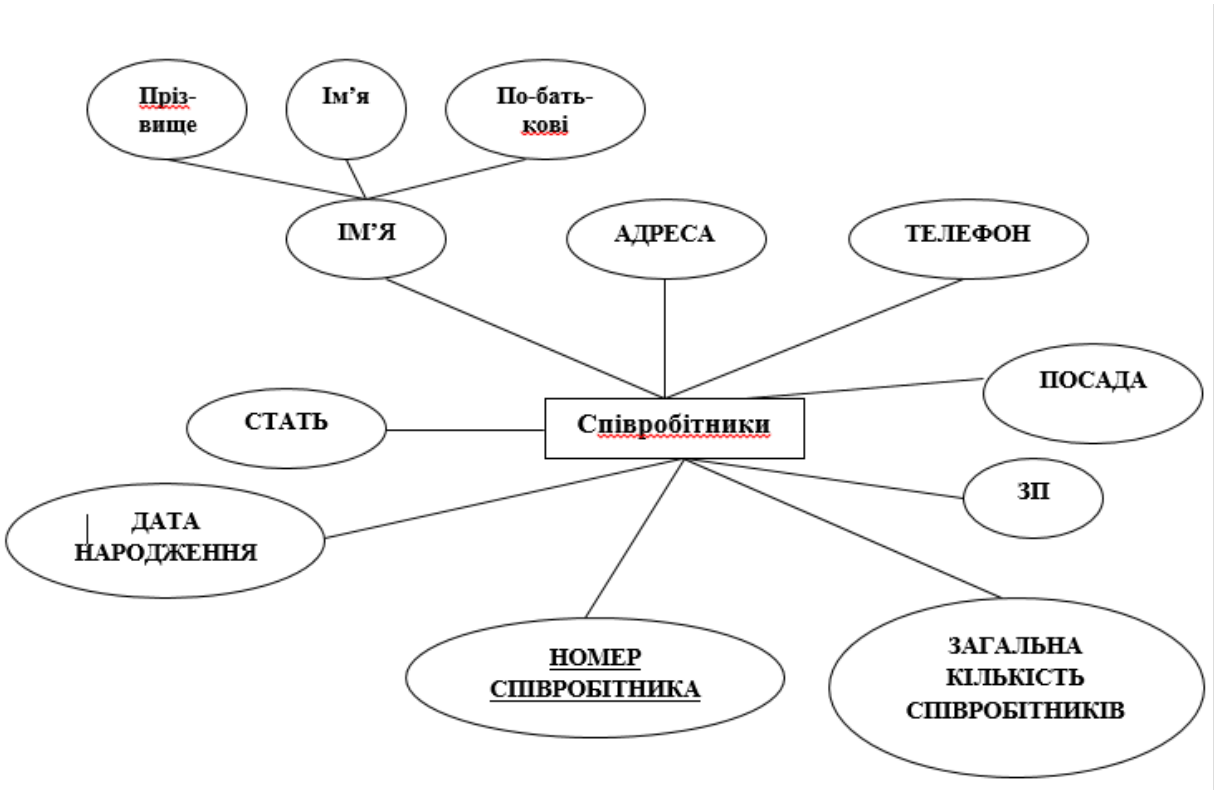
Прості атрибути іноді називають **атомарними**.

Складовий атрибут – атрибут, який складається з декількох компонентів, кожен з яких характеризується незалежним існуванням.

Деякі атрибути можуть бути розподілені на більш дрібні компоненти, які характеризуються незалежним існуванням. Наприклад, атрибут «Адреса» сутності «Відділення» зі значенням «Одеса, вул. Гер. Сталінграда 14, Суворівський, 65120, 568247» може бути розподілений на окремі атрибути «Місто» зі значенням «Одеса», «Вулиця» - зі значенням «Гер. Сталінграда 14», «Район» - зі значенням «Суворівський», «Поштовий індекс» - зі значенням «65120» та «Телефон» – зі значенням «568247».

На рисунку зображені атрибути, які пов'язані з сутностями **«Співробітники»** та **«Відділення»**.

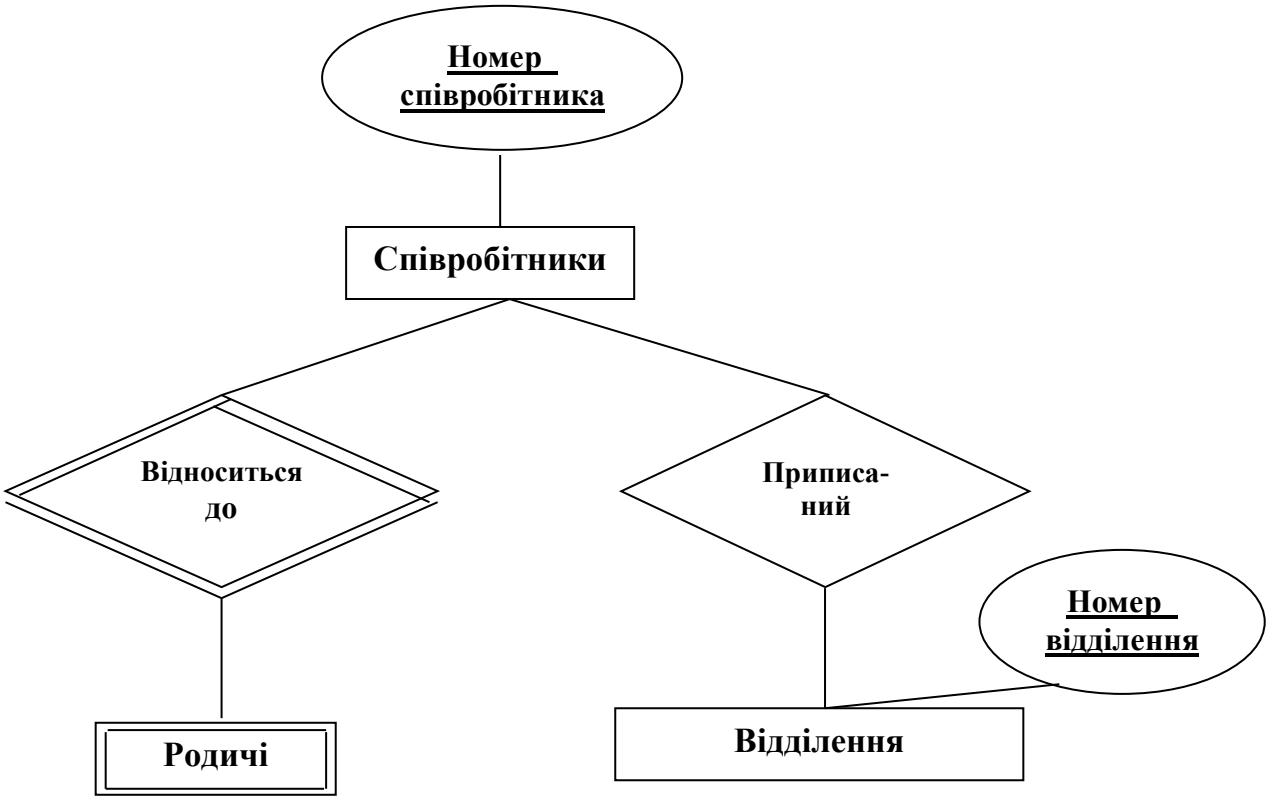
Сутність «Співробітники»



Якщо атрибут є складовим, його атрибути-компоненти зображуються у вигляді еліпсів, які приєднуються до нього (атрибут «Ім'я» сутності **«Співробітники»** є складовим атрибутом, який складається з атрибутів «Прізвище», «Ім'я», «По-батькові»).

Ім'я атрибуту, який є первинним ключем цього типу сутності, підкреслюється.

За допомогою **зв'язків** на ER-діаграмі відображається взаємодія між сутностями. Зв'язок зображується ромбом, який поєднує сутності. Всередині ромбу вказується вид зв'язка. Ромб має подвійний контур, якщо зв'язок пов'язує слабку сутність з сильною сутністю, від якої ця слабка сутність залежить.



На рисунку взаємозв'язок між сутностями «Відділення» та «Співробітники» зображений за допомогою зв'язка «Приписаний до», а між сутностями «Родичі» та «Співробітники» - за допомогою зв'язка «Відноситься до», який зображений у вигляді ромбу з подвійним контуром та вказує на те, що він встановлений між слабкою (сутність «Родичі») та сильною (сутність «Співробітники») сутностями.

Для зниження рівня деталізації на окремій ER-діаграмі часто вказують лише ті атрибути, які уявляють первинні ключі зображених сутностей, а в деяких випадках атрибути зовсім не відображаються.

Наприклад, на рисунку зображені лише ті атрибути, які є первинними ключами сильних сутностей, а саме: «Номер співробітника» та «Номер відділення».

Ступінь зв'язка – кількість сутностей, які охоплені цим зв'язком.

Охоплені деяким зв'язком сутності мають назву ***участники цього зв'язка***.

Кількість учасників деякого зв'язка має назву ступінь цього зв'язку. Тобто, ступінь зв'язка вказує на кількість типів сутностей, охоплених цим зв'язком.

Зв'язок зі ступнем 2 називається бінарним, зі ступенем 3 – тернарним, зі ступенем 4 – кватернарним.

Попередній приклад є прикладом тернарного зв'язка (приймають участь 3 типи сутності: «Співробітники», «Відділення» та «Родичі»).

Тема 2.1.2 Логічне проектування бази даних

(2 години)

Мета: навчитися здійснювати логічне проектування бази даних, тобто перетворювати модель «Сутність – зв'язок» в набір таблиць та пов'язувати ці таблиці зв'язками.

Структура заняття:

I. Організаційний момент:
   a. Готовність групи до заняття;
   b. Психоемоційний настрій;
   c. Перевірка присутніх.

II. Повідомлення теми та мети заняття;

III. Виклад нового матеріалу:

План

1. Поняття та призначення логічного проектування бази даних.
2. Приклади перетворення концептуальної моделі, представленої у вигляді ER – діаграми в логічну модель реляційної бази даних

IV. Узагальнення та систематизація знань;

V. Підведення підсумків занять;

VI. Домашнє завдання:
   1. Ознайомитись з теоретичними відомостями теми 2.1.2.
   2. Вивчити основні поняття лекції.
   3. Дати відповіді на контрольні питання.

Контрольні питання:

1. Для чого призначене логічне проектування бази даних?
2. Яким чином здійснюється логічне проектування бази даних?

Логічне (даталогічне) проектування - створення схеми бази даних на основі конкретної моделі даних, наприклад, реляційної моделі даних.

Для реляційної моделі даних даталогічна модель - набір схем відношень, зазвичай із зазначенням первинних ключів, а також "зв'язків" між відношеннями, що являють собою зовнішні ключі. Перетворення концептуальної моделі на логічну модель, як правило,

здійснюється за формальними правилами. Цей етап може бути значною мірою автоматизовано.

На етапі логічного проектування враховується специфіка конкретної моделі даних, але може не враховуватися специфіка конкретної СУБД.

Фаза логічного проектування передбачає такі дії:

✕ перетворення концептуальної моделі даних на логічну модель, унаслідок якого буде визначено схему реляційної моделі даних;

✕ перевірка моделі за допомогою концепцій послідовної нормалізації;

✕ перевірка моделі щодо транзакцій користувачів.

Спрощення концептуальної моделі даних

Перетворення концептуальної моделі даних на логічну модель, у результаті якого буде визначено схему реляційної моделі даних, може мати два підходи.

1) Проектувальник працює з концептуальною моделлю безпосередньо, не вдаючись до її попереднього перетворення. У цьому випадку йому доведеться зіткнутися з необхідністю перетворення різноманітних структур даних
2) Проектувальник, перш ніж розпочати процес переходу від концептуальної моделі до логічної моделі, прагне спочатку цей перехід максимально спростити, провівши попередні перетворення концептуальної моделі, перетворення деяких її структур даних, які не підходять для реляційних СУБД.

До таких структур даних належать:

1. зв'язки типу "багато до багатьох";
2. складні зв'язки;
3. рекурсивні зв'язки;
4. зв'язки з атрибутами;
5. множинні атрибути;
6. надлишкові зв'язки.

Якщо в моделі присутні перелічені небажані структури, то їх потрібно виключити шляхом тотожної їхньої заміни на структури даних, допустимі для логічної моделі.

Виключення зв'язку типу "багато до багатьох"

Перетворення зв'язку типу "багато до багатьох" здійснюється шляхом введення деякого проміжного об'єкта із заміною одного зв'язку двома зв'язками типу "один до багатьох" із новоствореним об'єктом. Під час організації нових зв'язків необхідно

стежити за тим, щоб максимальна потужність зв'язку "один" була спрямована до вихідного об'єкта, а максимальна потужність зв'язку "багато" - до новоствореного об'єкта.

Розглянемо основні правила перетворення ER-моделі в логічну модель бази даних на прикладі діаграми, побудованої за темою курсової роботи. Для прикладу розглянемо тему «Розробити автоматизоване робоче місце менеджера з продажу магазину комп'ютерної техніки».

На рис. 1 зображена модель «Сутність – зв'язок» для даної предметної області, а на рис. 2 ця модель вже перетворена в логічну модель реляційної бази даних.

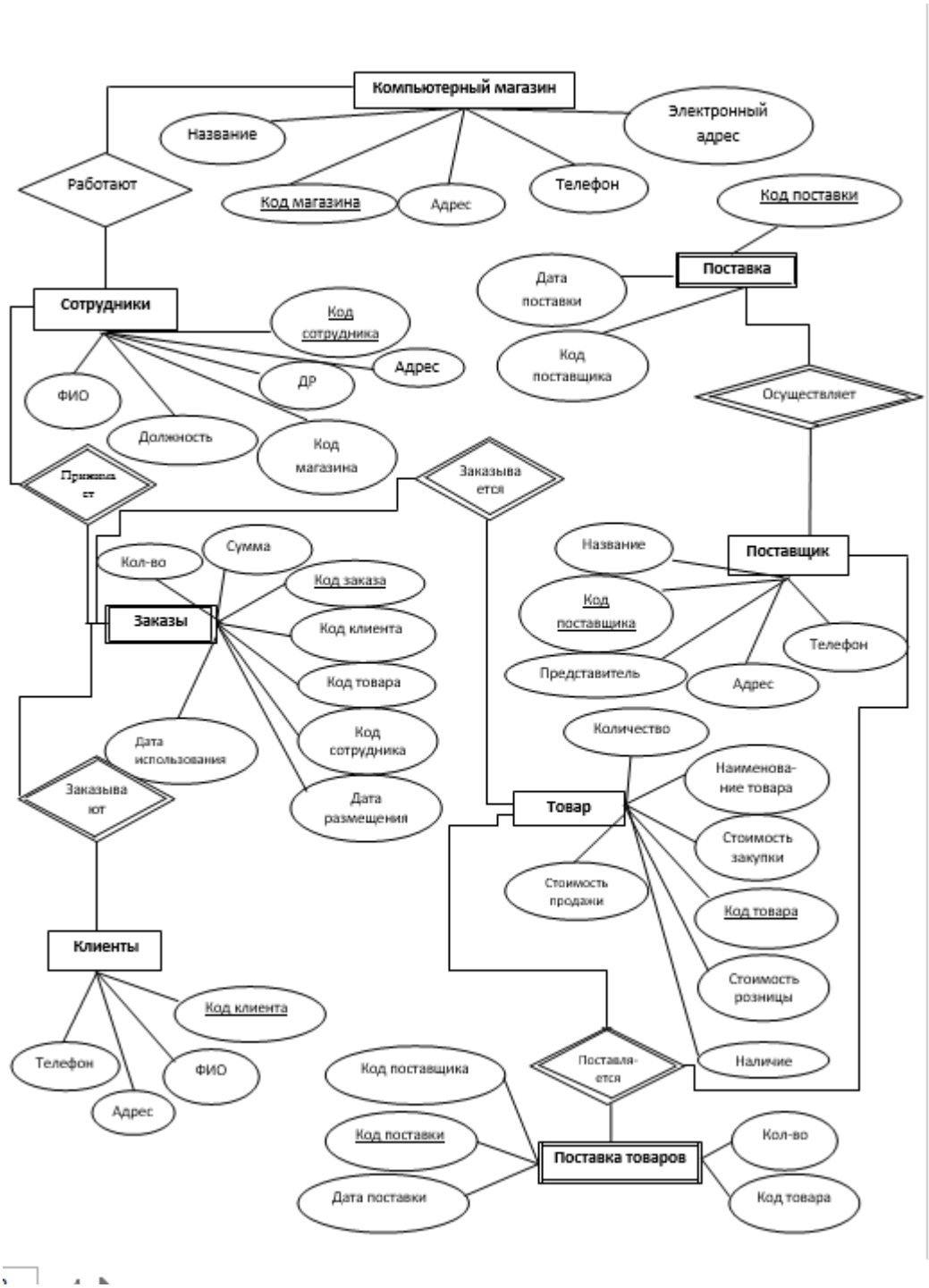


Рис. 1 - Модель «Сутність – зв'язок» для предметної області «Комп'ютерний магазин»

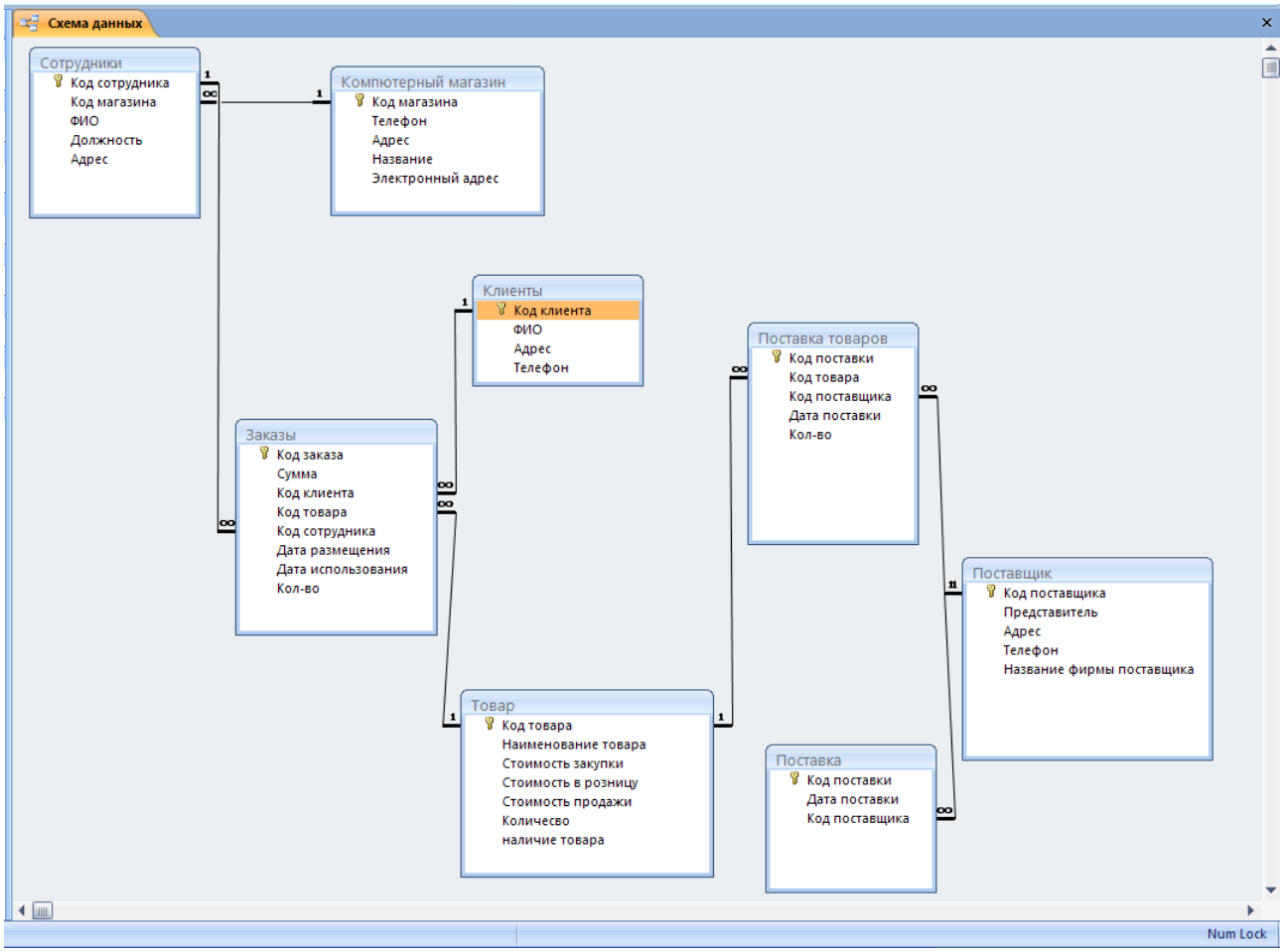


Рис. 2 Структурна схема реляційної БД для предметної області «Комп'ютерний магазин»

Розглянуті правила перетворення ER-моделі в логічну модель бази даних досить прості, наочні і дозволяють отримати добрі результати.

Змістовий модуль 2.2 - Процес нормалізації

Тема 2.2.1 Нормалізація відношень

(2 години)

Мета: знати про процес нормалізації та призначення кожної нормальної форми.

Структура заняття:

I. Організаційний момент:
   a. Готовність групи до заняття;
   b. Психоемоційний настрій;
   c. Перевірка присутніх.

II. Повідомлення теми та мети заняття;

III. Виклад нового матеріалу:

План

1. Поняття та призначення нормалізації.
2. Нормальні форми.
3. Переваги та недоліки нормалізації.

IV. Узагальнення та систематизація знань;

V. Підведення підсумків занять;

VI. Домашнє завдання:

4. Ознайомитись з теоретичними відомостями теми 2.2.1.
5. Вивчити основні поняття лекції.
6. Дати відповіді на контрольні питання.

Контрольні питання:

1. Що розуміється під поняттям нормалізації?
2. Які нормальні форми існують в реляційній теорії?
3. В чому зміст 1 нормальної форми. Навести приклад бази даних, доведеної до 1 нормальної форми.
4. В чому зміст 2 нормальної форми. Навести приклад бази даних, доведеної до 2 нормальної форми.
5. В чому зміст 3 нормальної форми. Навести приклад бази даних, доведеної до 3 нормальної форми.
6. Якими перевагами володіє нормалізація?
7. Завдяки якій перевазі нормалізації спрощується структура бази даних та економиться дисковий простір?
8. Якими недоліками володіє нормалізація?

Під **нормалізацією** розуміється процес приведення відношення до однієї з нормальних форм (НФ). Але перед тим як розглядати нормальні форми, треба розповісти, для чого потрібна нормалізація.

При проектуванні баз даних в першу чергу робиться наголос на вірогідність та несуперечливість зберігаємих даних, до того ж, ці властивості не повині втрачатися в процесі роботи з даними, тобто після багаточисельних змін, знищень та доповнень даних по відношенню до першопочаткового стану бази даних.

Для підтримки бази даних в стійкому стані використовується ряд механізмів, які отримали узагальнену назву **засобів підтримки цілістності**. Ці механізми

використовуються **статично** (на етапі проектування бази даних), так й **динамічно** (в процесі роботи з базою даних).

База даних повинна володіти обмеженнями, які будуть задовільняти її в процесі створення, незалежно від її наповнення даними. Доведення структури бази даних до відповідності цим обмеженням – це **нормалізація**. В цілому суть цих обмежень дуже проста: кожен факт, який зберігається в базі даних, повинен зберігатися один єдиний раз, так як дублювання може призвести до незгоди між копіями однакової інформації. Треба уникати невизначеностей, а також надмірності зберігаємої інформації. Отже, наша **мета** – **доведення відношень до НФ**. Треба відмітити, що в процесі нормалізації постійно зустрічається ситуація, коли відношення необхідно розкласти на декілька інших відношень. Тому коректніше було б казати про нормалізацію не окремих відношень, а всієї їх сукупності в базі даних.

Нормалізація може не викликати проблему, якщо база даних проектується одразу ж за визначеними правилами. Іншими словами, можна спочатку створити базу даних як-небудь, а потім нормалізувати її, або ж з самого початку будувати її за правилами для того, щоб в подальшому не треба було б її переробляти.

Нормалізація відношень забезпечує ефективність структур даних в реляційній БД. Цей процес зменшує надмірність даних (зберігання однакових даних в декількох місцях). В результаті більш раціонально використовується зовнішня пам’ять, зменшується ймовірність порушення узгоджуваності даних.

Нормалізація - це послідовне перетворення початкової бази даних до НФ, при цьому кожна наступна НФ обов’язково містить у собі попередню.

В реляційній теорії нараховують 6 НФ:

- 1 НФ;
- 2 НФ;
- 3 НФ;
- НФ Бойса-Кодда (НФБК);
- 4 НФ;
- 5 НФ.

На практиці обмежуються 3 НФ, її достатньо для створення надійної схеми бази даних. НФ більш високих порядків уявляють академічну зацікавленість із-за надмірної складності. Крім того, при реалізації абстрактної схеми бази даних у вигляді реальної бази іноді розробники змушені зробити крок назад – провести денормалізацію з метою

підвищення ефективності, так як ідеальна з точки зору теорії структура може статися занадто накладною на практиці.

Основні властивості нормальних форм:

• кожна наступна нормальна форма покращує властивості попередньої нормальної форми;

• при переході до наступної нормальної форми властивості попередніх нормальних форм зберігаються.

Перша нормальна форма (1НФ)

Відношення знаходиться в 1 НФ, якщо на перетині кожного стовпця з кожним рядком міститься лише атомарне (нерозподілене) значення. Іншими словами, кожен атрибут відношення повинен зберігати одне єдине значення та не бути ні списком, ні множиною значень. Атрибут, який є атомарним в одному додатку, може виявитися складовим в іншому.

Приклад. В базі даних відділу кадрів фірми в таблиці, в якій зберігаються особисті відомості про співробітників, є атрибут «Домашня адреса», в якому адреса зберігається в форматі: місто, вулиця, будинок[/корпус], квартира (в даному випадку адреса зберігається у вигляді єдиного текстового рядка, так як малоймовірно, щоб необхідно було б обрати співробітників, наприклад, за номером квартири). Таким чином, в контексті бази даних відділу кадрів адреса є атомарним поняттям, та його ділення на складові частини немає сенсу, так як лише привнесе до бази даних надмірну громіздкість. Але та сама адреса для додатку, призначеного для сортування пошти у поштовому відділенні компанії, не є атомарним, тому що бажано було б згрупувати конверти в окремі купи по вулицям, так як кожну вулицю обслуговує свій листоноша. Крім цього, з метою оптимізації переміщення листоноші в межах вулиці, кожну купу бажано було б відсортувати за номерами будинків, для того щоб зробити можливим рознесення пошти за один прохід по вулиці без повернення.

Доведення відношення до 1 НФ – проста операція. Треба проглянути відношення та розподілити складові атрибути на різні рядки/стовпці. Можливо, цю ситуацію доведеться повторити декілька разів до тих пір, доки кожен із атрибутів не стане атомарним.

Приклад. В базі даних міститься таблиця підприємств, в якій зберігаються такі відомості:

- найменування підприємства;
- місто;

- адреса;
- електронна адреса;
- веб-сторінка;
- вид агенту (постачальник або клієнт);
- контактні особи (може бути декілька), посада контактної особи, телефон контактної особи.

Табл. 1

Найм-ня	Місто	Адреса	Ел. пошта	Веб-стор.	Вид	Конт. особа
Поршневий з-д	Київ	Вул. 2-а Кольцева, 17	info@ plunger.gmail.com	www. plunger.ua	Постачаль-ник	Іванов І. І., заступник директора, тел (044)76-15-95 Петров П. П., начальник від. збуту, тел (044)76-15-35
ПП«Вимпел»	Одеса	Вул. Гоголя, 25	pennon@ gmail.com		Клієнт	Сидоров С. С., директор, тел. (048)66-65-38
ПП «Альфа»	Київ	Вул. Пушкінська, 37, оф. 565	alpha@ gmail.com		Клієнт	Васильєв В.В., директор, тел (044)74-57-45

В даному випадку атрибут «Контактна особа» не є атомарним, тому що в ньому зустрічаються списки з декількох осіб. Розподілимо ці кортежі таким чином, щоб кожен кортеж містив дані тільки про одну особу:

Табл. 2

Найм-ня	Місто	Адреса	Ел. пошта	Веб-стор.	Вид	Конт. особа
Поршневий з-д	Київ	Вул. 2-я Кольцева, 17	info@ plunger.gmail.com	www. plunger.ua	Постачаль-ник	Іванов І. І., заступник директора, тел (044)76-15-95
Поршневий з-д	Київ	Вул. 2-я Кольцева, 17	info@ plunger.gmail.com	www. plunger.ua	Постачаль-ник	Петров П. П., начальник від. збуту, тел (044)76-15-35
ПП «Вимпел»	Одеса	Вул. Гоголя, 25	pennon@ gmail.com		Клієнт	Сидоров С.С., директор, тел. (048)66-65-38
ПП «Альфа»	Київ	Вул. Пушкінська, 37, оф. 565	alpha@ gmail.com		Клієнт	Васильєв В.В., директор, тел (044)74-57-45

Трохи краще, хоча якщо придивитися виявляється, що атрибут «Контактна особа» може бути атомарним знов з натягуванням, так як містить дані різного роду, хоча й про одну особу. Розподілемо його на декілька атрибутів:

Табл. 3

Найм-ня	Місто	Адреса	Ел. пошта	Веб-сторінка	Вид	Посада	ПІБ	Код міста	Тел.
Поршневий з-д	Київ	Вул. 2-я Кольцева, 17	info@ plunger.gmail.com	www. plunger.ua	Постачальник	Заст. дир.	Іванов І.І.	044	76-15-95
Поршневий з-д	Київ	Вул. 2-я Кольцева, 17	info@ plunger.gmail.com	www. plunger.ua	Постачальник	Нач. від. збуту	Петров П.П.	044	76-15-35
ПП «Вимпел»	Одеса	Вул. Гоголя, 25	pennon@ gmail.com		Клієнт	Директор	Сидоров С.С.	048	66-65-38
ПП «Альфа»	Київ	Вул. Пушкінська, 37, оф. 565	alpha@ gmail.com		Клієнт	Директор	Васильєв В.В.	044	74-57-45

Зараз можна вважати, що кожне значення кожного з атрибутів нашого відношення є атомарним. Отже, відношення знаходиться в 1 НФ.

Друга нормальна форма (2НФ)

Зараз наше відношення має коректний вигляд, але не досконалий. Значення, які повторюються в таблиці, є потенційним джерелом проблем.

По-перше, при заповненні таких записів легко помилитися. Наприклад, достатньо змінити лише один символ в атрибуті «Адреса», та це стане зовсім другою адресою, яка немає нічого спільного з першою адресою. Знайти такі помилки в величезній таблиці – дуже складна задача.

По-друге, назва вулиці може змінитися. Може змінитися й адреса веб-сайту фірми. Тоді треба буде проглядати всю таблицю та змінювати відповідні значення.

По-третє, не можна зневажати ефективністю зберігання інформації. З нашої таблиці можна було б як мінімум двічі дізнатися адресу поршневого заводу, хоча достатньо було б одного разу. Таким чином, боротьба з надмірністю даних є найбільш вигідною з усіх боків – як з боку підвищення зручності користування даними та підтримки їх в цілісному вигляді, так й з боку ефективності обробки та зберігання даних апаратними засобами серверу баз даних. Саме на усунення цієї надмірності спрямована подальша нормалізація відношень.

Відношення знаходиться в другій нормальній формі, якщо воно знаходиться в першій нормальній формі, і кожен неключовий атрибут (тобто що не є складовою частиною первинного ключа) функціонально повно залежить від первинного ключа.

Припустимо, що при постановці задачі замовник повідомив нам, що в межах кожного міста найменування підприємств є унікальним, але в різних містах найменування можуть співпадати. Таким чином, підприємство характеризується складовим ключем «Найменування + Місто». Подивимося на наше відношення з точки зору того, як ми тільки що дізналися про 2 НФ. Явно, що телефонний код міста залежить виключно від самого міста та жодним чином не пов'язаний з найменуванням підприємства. Звідси й одне з джерел надмірності даних – скільки разів в таблиці зустрічається рядок, який містить відомості про контактну особу будь-якого підприємства, яке знаходиться в даному місті, стільки й повторюється інформація про код міста. Для усунення цієї надмірності, треба розподілити наше відношення на декілька відношень:

Табл. 4а

Найм-ня	Місто	Адреса	Ел. пошта	Веб-сторінка	Вид	Посада	ПІБ	Тел.
Поршневий з-д	Київ	Вул. 2-я Кольцева, 17	info@ plunger.gmail.com	www. plunger.ua	Постачальник	Заст. дир.	Іванов І.І.	76-15-95
Поршневий з-д	Київ	Вул. 2-я Кольцева, 17	info@ plunger.gmail.com	www. plunger.ua	Постачальник	Нач. від. збуту	Петров П.П.	76-15-35
ПП «Вимпел»	Одеса	Вул. Гоголя, 25	pennon@ gmail.com		Клієнт	Директор	Сидоров С.С.	66-65-38
ПП «Альфа»	Київ	Вул. Пушкінська, 37, оф. 565	alpha@ gmail.com		Клиієт	Директор	Васильєв В.В.	74-57-45

Табл. 4б

Місто	Код міста
Київ	044
Одеса	048

Отже, ми усунули часткову залежність атрибута «Код міста» від складового ключа, перемістивши коди міст в окреме відношення з ключем «Місто». Таким чином, ми отримали два відношення, кожне з яких знаходиться в 2 НФ.

Не дивлячись на здійснені зусилля, таблиця 4а все ж таки містить надмірність – достатньо поглянути на значення в стовбцях «Адреса», «Ел. пошта» та «Веб. сторінка»,

які повторюються. Це значить, що процес нормалізації не є завершуваним, та треба приступати до його наступного етапу.

Третя нормальна форма (3НФ)

Існує функціональна залежність між атрибутами «ПІБ», «Посада» та «Телефон». Є очевидним, що на підприємстві деяка людина займає визначену посаду та розпоряджається визначеним робочим телефоном. Зворотнє в спільному випадку не є вірним – на підприємстві може існувати декілька аналогічних штатних одиниць, наприклад, менеджери за збутом, та декілька чоловік мають можливість користуватися одним робочим телефоном.

Для того, щоб уникнути цю функціональну залежність, треба зробити декомпозицію таблиці 4а на дві таблиці. Перша з них буде зберігати факти, які відносяться безпосередньо до самого підприємства:

Табл. 5а

Найм-ня	Місто	Адреса	Ел. пошта	Веб-сторінка	Вид
Поршневий з-д	Київ	Вул. 2-я Кольцева, 17	info@ plunger.gmail.com	www.plunger.ua	Постачаль-ник
ПП «Вимпел»	Одеса	Вул. Гоголя, 25	pennon@gmail.com		Клієнт
ПП «Альфа»	Київ	Вул. Пушкінська, 37, оф. 565	alpha@ gmail.com		Клієнт

Друга таблиця буде зберігати факти, які відносяться до конкретної особи, яка виконує обов’язки на цьому підприємстві:

Табл. 5б

Найм-ня	Місто	ПІБ	Посада	Тел.
Поршневий з-д	Київ	Іванов І.І.	Заступник директора	76-15-95
Поршневий з-д	Київ	Петров П.П.	Начальник від. збуту	76-15-35
ПП «Вимпел»	Одеса	Сидоров С.С.	Директор	66-65-38
ПП «Альфа»	Київ	Васильєв В.В.	Директор	74-57-45

Разом з таблицею 4б цей набір таблиць уявляє собою нашу першопочаткову базу даних, наведену до 3 НФ.

Відношення знаходиться в третій нормальній формі, якщо воно знаходиться в другій нормальній формі та кожен його неключовий атрибут безпосередньо (не транзитивно) залежить від первинного ключа.

В більшості випадків досягнення третьої нормальної форми вважається достатнім для реальних проектів баз даних, проте в теорії нормалізації існують нормальні форми

вищих порядків (НФБК, 4НФ, 5НФ), деякі з яких пов'язані вже не з функціональними залежностями між атрибутами стосунків, а відображають тонші питання смислового вмісту наочної області.

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.

Блок змістових модулів 3 – Робота з базами даних

Змістовий модуль 3.1 - Створення бази даних в СУБД Access

Тема 3.1.1 СУБД MS Access. Інтерфейс програми.

(2 години)

Мета: знати призначення СУБД MS Access. Знати який інтерфейс має СУБД MS Access.

Структура заняття:

I. Організаційний момент:
   a. Готовність групи до заняття;
   b. Психоемоційний настрій;
   c. Перевірка присутніх.

II. Повідомлення теми та мети заняття;

III. Виклад нового матеріалу:

План

1. Призначення СУБД MS Access.
2. Об'єкти бази даних MS Access.

IV. Узагальнення та систематизація знань;

V. Підведення підсумків занять;

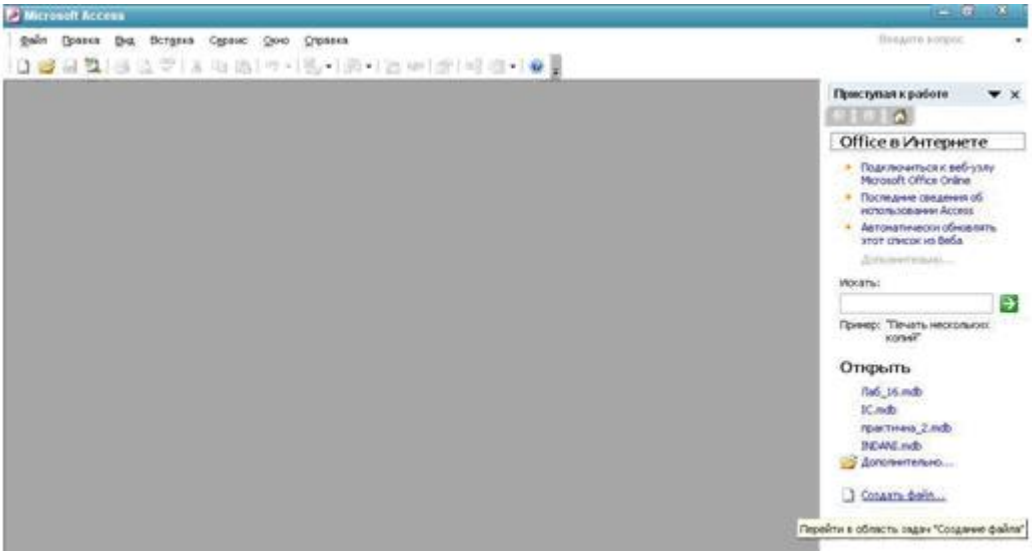
VI. Домашнє завдання:
   1. Ознайомитись з теоретичними відомостями теми 3.1.1.
   2. Вивчити основні поняття лекції.
   3. Дати відповіді на контрольні питання.

Контрольні питання:

1. Для чого призначена СУБД MS Access?
2. Яке розширення має файл бази даних, створеної в СУБД MS Access?
3. Які об’єкти вміщує СУБД MS Access?
4. В яких режимах виконується робота з таблицею?
5. Який об'єкт СУБД MS Access призначений для отримання даних з однієї або декількох таблиць?
6. Який об'єкт використовується для зручного введення даних?
7. Що таке макроси?

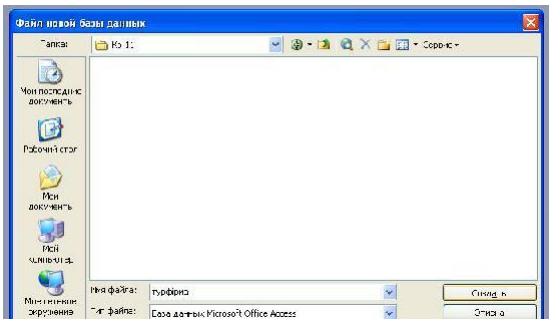
Access - це повноцінна реляційна СУБД, у якій може працювати як пересічний користувач, котрий не володіє основами програмування, так і фахівець-розробник, який створює додатки на мові Visual Basic for Applications (скоромно VBA). Популярність СУБД Access при вивченні баз даних зумовлена тим, що ця система є найпростішою для засвоєння початківцями.

Робота з базою даних починається з запуску СУБД. Для запуску Access натисніть кнопку **Пуск** на панелі задач і оберіть у головному меню команду **Всі програми** - **Microsoft Access**. У відповідь з'явиться діалогове вікно Microsoft Access.



Як було зазначено, роботу в СУБД Access ми будемо вивчати на прикладі створення бази даних, яка описує діяльність якоїсь туристичної фірми «Подорож». Нехай ця фірма приймає заявки на туристичні путівки і продає їх.

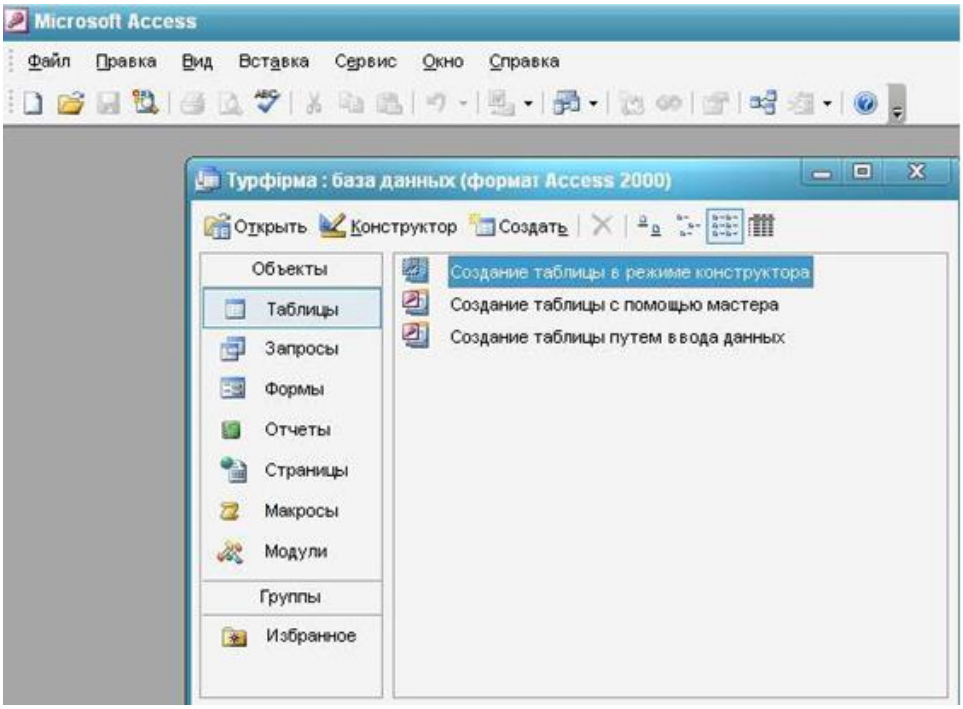
Отже, у первинному вікні ви обрали опцію створення нової БД і натиснули кнопку ОК. У наступному стандартному діалозі вам потрібно ввести ім'я файла для нової бази даних (у нашому прикладі це «Фірма «Подорож») і позначити папку, у якій цей файл розміститься. Йому буде надано розширення, mdb.



Після задання імені і розташуван файла клацніть по кнопці **Создать**, і на екрані з'явиться вікно бази даних

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.

Тема 3.1.2 Створення таблиць в MS Access.

(2 години)

Мета: знати призначення таблиць в MS Access та якими способами можна їх створити.

Структура заняття:

I. Організаційний момент:
   a. Готовність групи до заняття;
   b. Психоемоційний настрій;
   c. Перевірка присутніх.

II. Повідомлення теми та мети заняття;

III. Виклад нового матеріалу:

План

1. Режими створення таблиці в MS Access.
2. Завдання структури таблиці.
3. Типи даних в MS Access.

IV. Узагальнення та систематизація знань;

V. Підведення підсумків занять;

VI. Домашнє завдання:
   1. Ознайомитись з теоретичними відомостями теми 3.1.2.
   2. Вивчити основні поняття лекції.
   3. Дати відповіді на контрольні питання.

Контрольні питання:

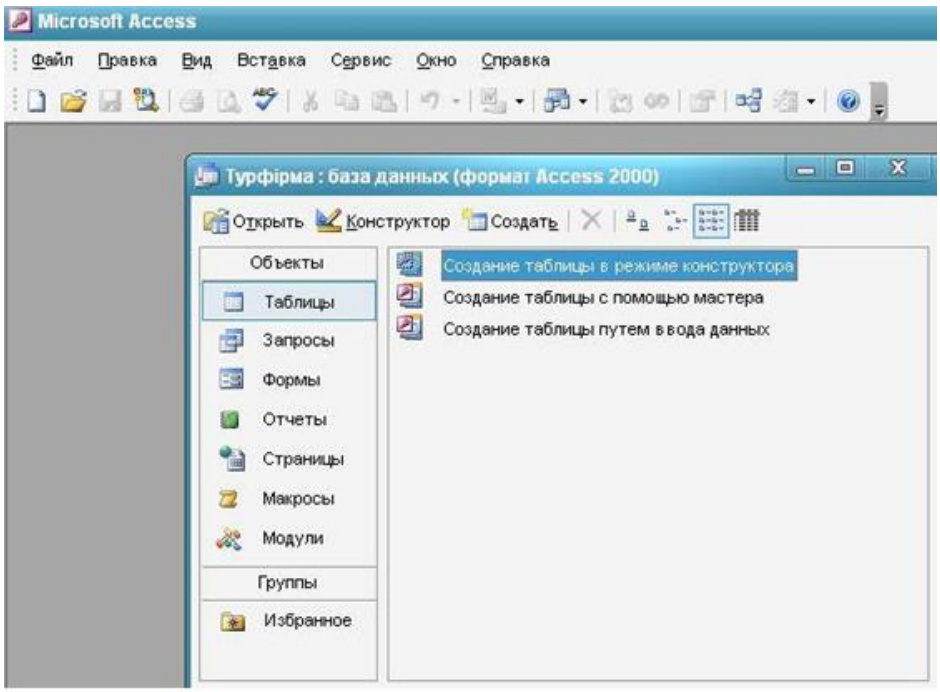
1. Які існують режими створення таблиць в MS Access?
2. Які типи даних підтримуються в MS Access?
3. Який тип даних призначений для введення числа, що автоматично збільшується на одиницю при додаванні до таблиці нового запису?
4. Для чого призначений майстер підстановок?

Основою реляційної бази даних є таблиці. Тому з їх побудови доречно почати створення бази даних.

Режими створення таблиць

Таблиці бази даних описують певні теми. Однак, перед тим як почати створення таблиці, потрібно добре уявити її структуру, тобто склад полів, типи даних і властивості полів.

Для створення таблиці в додатку Access відкрийте вікно бази даних. Якщо воно ще не відкрито, натисніть клавішу F11. Перейдіть у цьому вікні на вкладку Таблиці і клацніть по кнопці Створити.



У наступному діалозі вам буде запропоновано обрати режим створення таблиці.

У Access передбачено кілька таких режимів, які обираються зі списку діалогу **Нова таблиця**:

Режим таблиці - застосовується для заповнення і редагування полів таблиці.

Конструктор - режим для задания структури таблиці, тобто імен полів і типів даних.

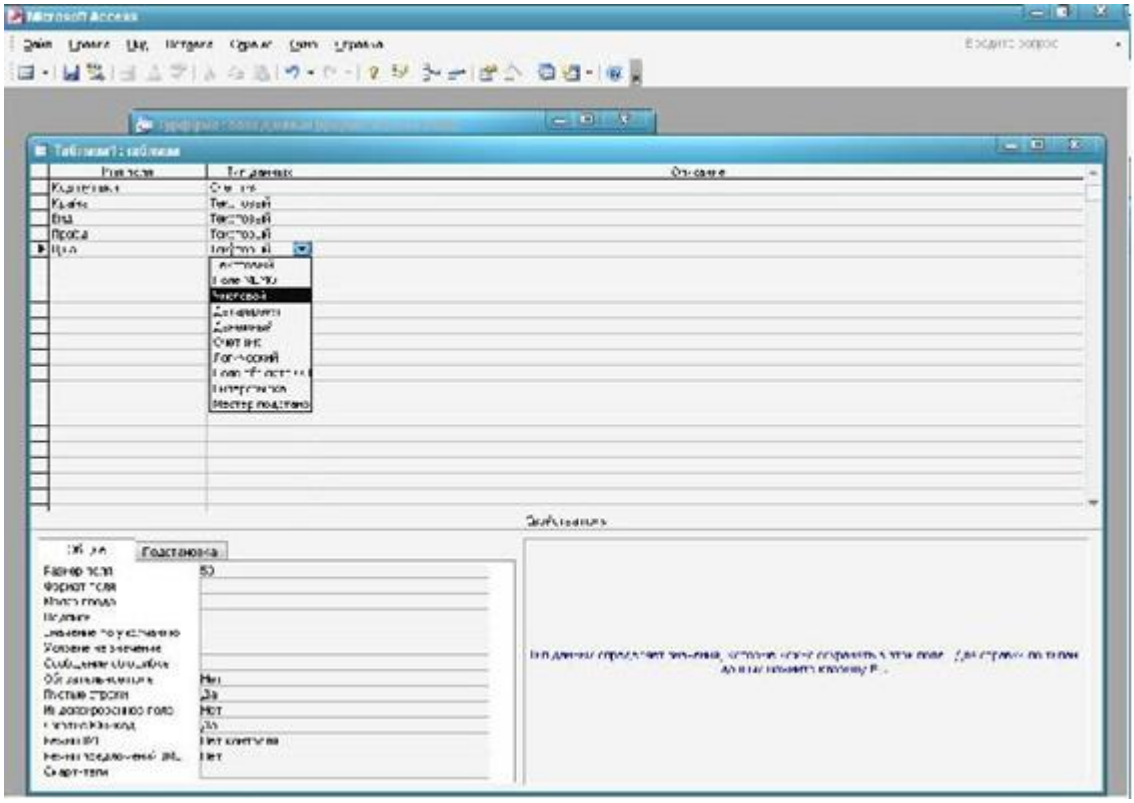
Майстер таблиць - створення таблиці за допомогою програми майстра.

Импорт таблиць - створення таблиці шляхом уведення даних із зовнішнього файла (іншої бази даних, електронних таблиць тощо).

Завдання структури таблиці

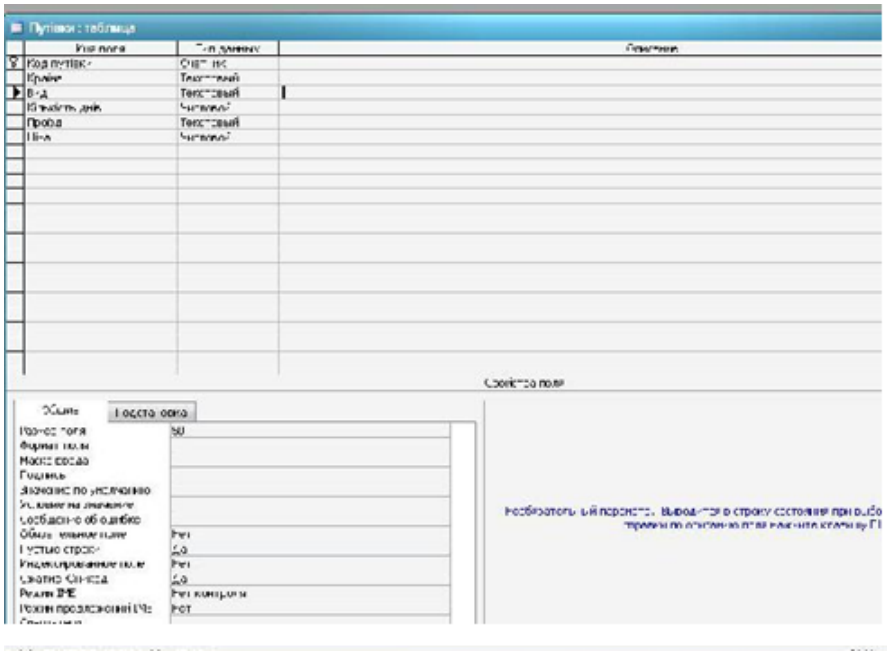
Припустимо, що ви запустили процедуру створення нової таблиці (тобто у вікні бази даних на вкладці **Таблиці** була натиснута кнопка **Створити**). Далі в діалозі **Нова**

таблиця виділіть у списку позицію **Конструктор** і клацніть по кнопці **ОК**, після чого з'явиться вікно конструктора.



У цьому вікні потрібно визначити склад таблиці, заповнивши для кожного поля таблиці три колонки: Ім’я поля, Тип даних і Опис. У першій колонці задається ім'я поля майбутньої таблиці, у другій - вказується його тип, а в третій - вводиться інформація про призначення поля. Кількість записів у вікні конструктора повинно збігатися з кількістю полів у створюваній таблиці.

Складемо таблицю, у якій будуть вміщені відомості про путівки, запропоновані туристичною фірмою «Подорож».



Зверніть увагу, що активне поле (у ньому в даний момент розміщений курсор) позначено зліва індикатором - трикутною стрілкою. Властивості активного поля названі в нижній частині діалогу

Якщо потрібно видалити будь-який рядок з таблиці, клацніть мишею по цьому рядку у вікні конструктора, а потім виконайте команду меню Правка -Знищити. Для вставки нового рядка потрібно активізувати поле нижче рядка, що вставляється, і виконати команду меню Вставка - Рядка.

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 MEMO - 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.

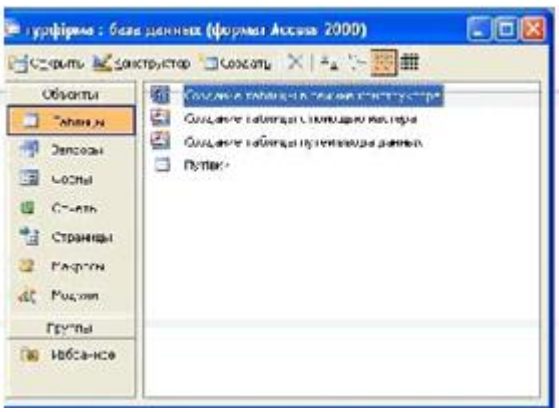
Гиперпосилання - служить для запису до таблиці гіперпосилань (шляху URL)

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

Зберігання таблиці

Отже, структура таблиці задана, і ви можете закрити вікно конструктора таблиці клацанням по кнопці закриття вікна Х, Якщо таблиця нова і раніше не зберігалася, з'явиться запит щодо того, чи варто зберігати структуру таблиці. Клацніть по кнопці Так і в діалозі Збереження введіть ім'я таблиці (воно може містити будь-які символи, крім крапки, знака оклику і кутових дужок), після цього натисніть кнопку ОК.

У розглянутому нами прикладі таблицю збережено під ім'ям «Путівки». Після зберігання у вікні бази даних з'явиться піктограма створеної таблиці.



Тема 3.1.3 Створення форм. Введення даних за допомогою форм.

(2 години)

Мета: знати призначення форм в СУБД MS Access.

Структура заняття:

I. Організаційний момент:
   a. Готовність групи до заняття;
   b. Психоемоційний настрій;
   c. Перевірка присутніх.

II. Повідомлення теми та мети заняття;

III. Виклад нового матеріалу:

План

1. Форма як об’єкт бази даних MS Access.
2. Режими створення форм.

IV. Узагальнення та систематизація знань;

V. Підведення підсумків занять;

VI. Домашнє завдання:
   1. Ознайомитись з теоретичними відомостями теми 3.1.3.
   2. Вивчити основні поняття лекції.
   3. Дати відповіді на контрольні питання.

Контрольні питання:

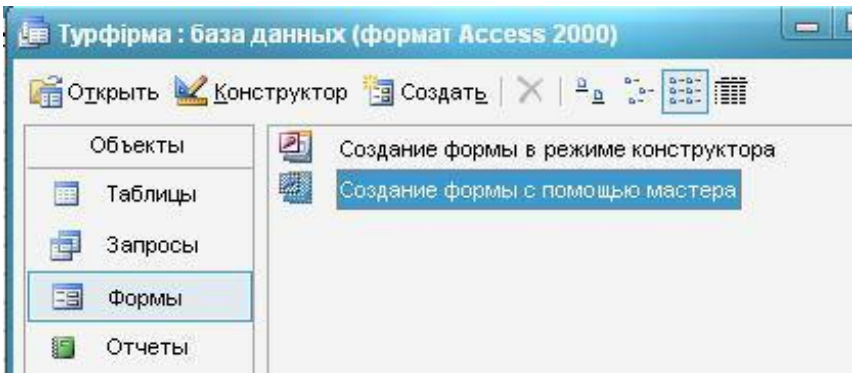
1. Що таке форма?
2. Які є способи створення форм?

До баз даних інформація звичайно вводиться за допомогою форм, а зберігається-у вигляді таблиць.

Форма - це об'єкт бази даних, призначений для введення і відображення інформації. Форма обов'язково містить елементи (поля), до яких користувач уводить дані.

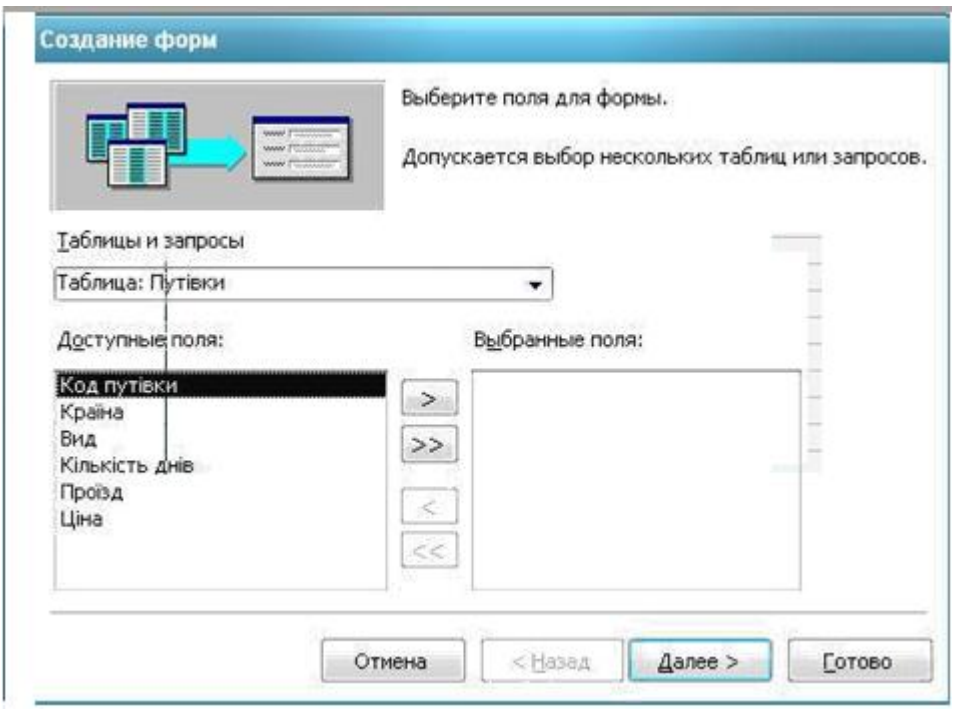
Користувач Access може створити форму самостійно, але краще звернутися до послуг програми-майстра. Це прискорить процес побудови форми, оскільки майстер виконає всю основну роботу.

Відкрийте вікно БД (натисніть клавішу F11). Якщо у вас відкрито вікно таблиці, яка буде основою форми, закрийте його. Перейдіть на вкладку Форми У вікні БД і клацніть по кнопці **Створити.**

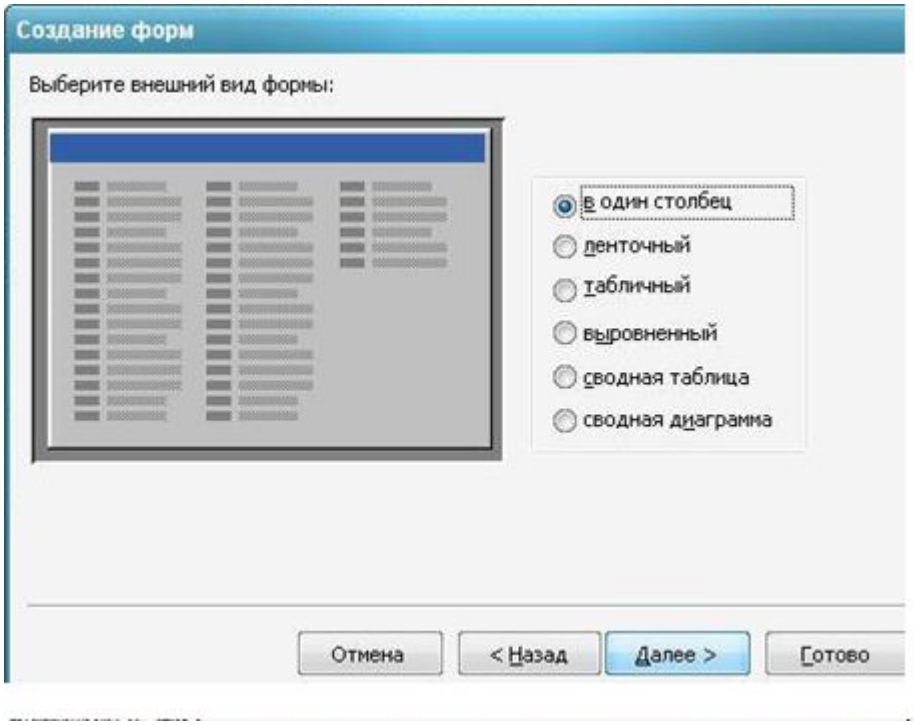


У діалозі **Нова форма** вкажіть режим Мастер форм і У розкривному списку оберіть таблицю, для якої створюватиметься форма.

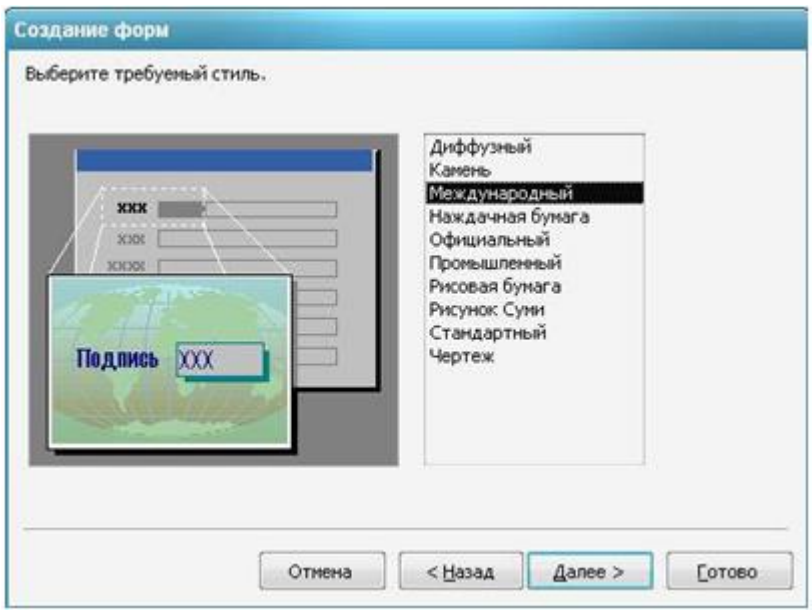
- у першому діалозі майстра створення форм, вкажіть поля, які будуть присутні у формі.



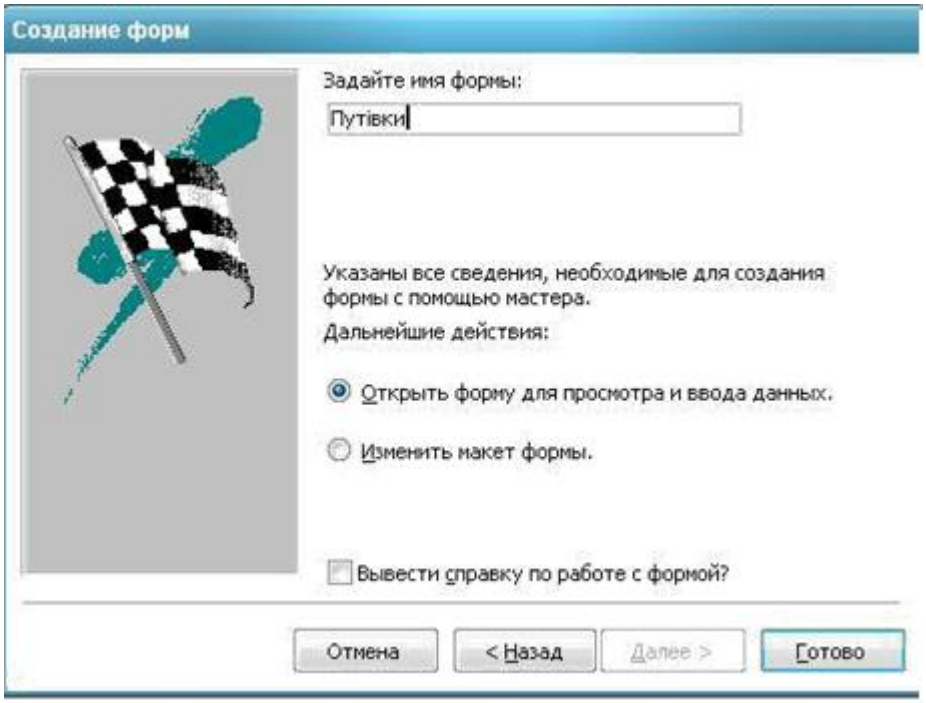
Після натискання на кнопку Далі відкриється наступний діалог для вибору вигляду форми. За умовчанням пропонується форма, в якій поля введення розміщаються в стовпчик. Погодьтеся з цією пропозицією і клацніть по кнопці Далі.



У наступному діалозі вам потрібно вибрати стиль оформлення (фон, кольори полів і написів форми). У поданому списку стилів зазначте потрібний вам і натисніть кнопку Далі.

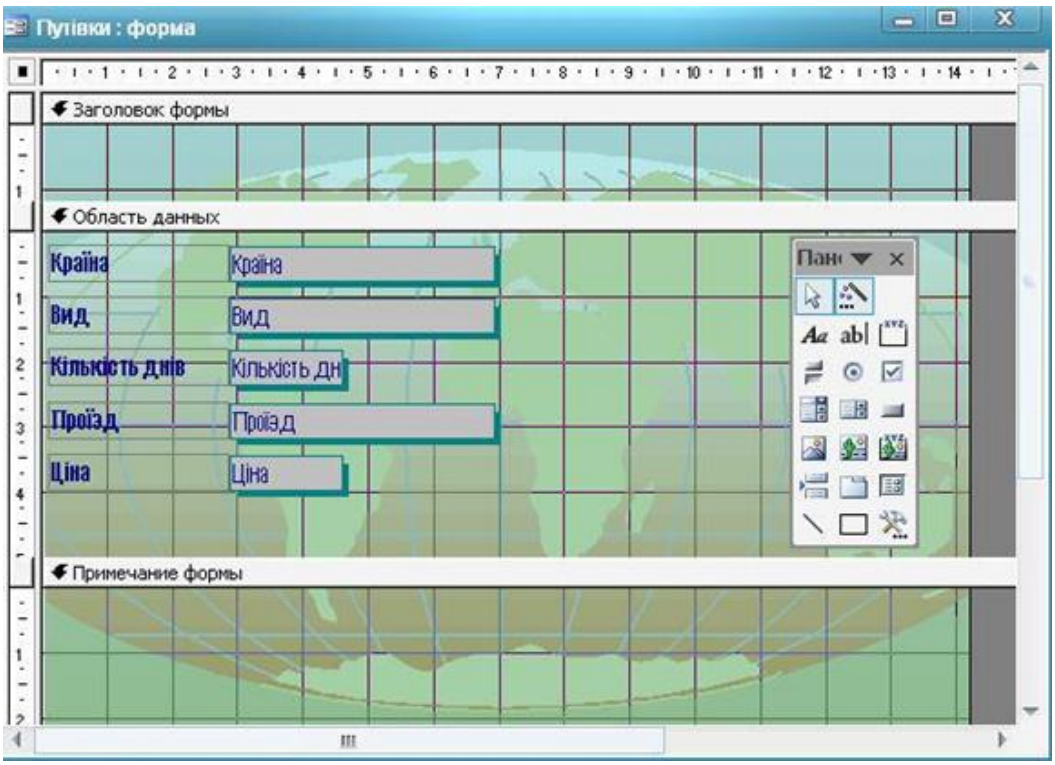


У новому діалозі введіть ім'я форми (за умовчанням пропонується ім'я таблиці-джерела).

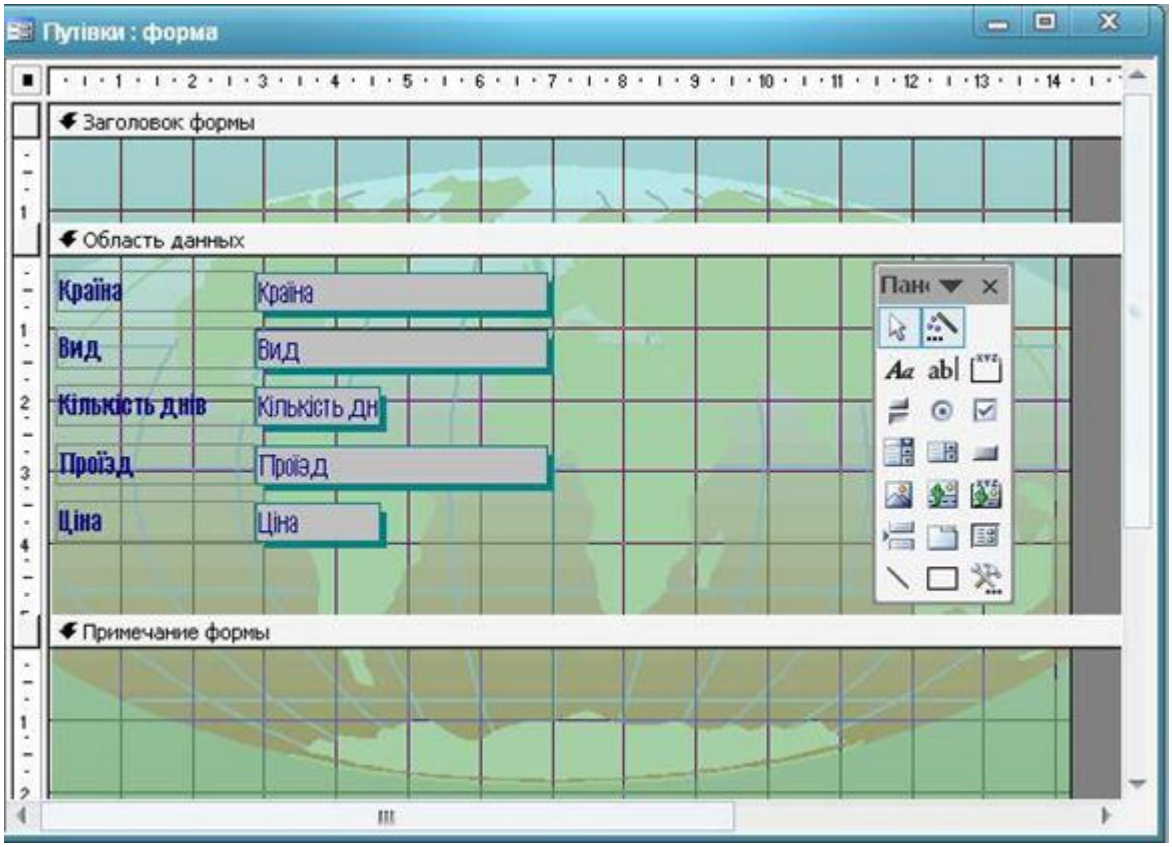


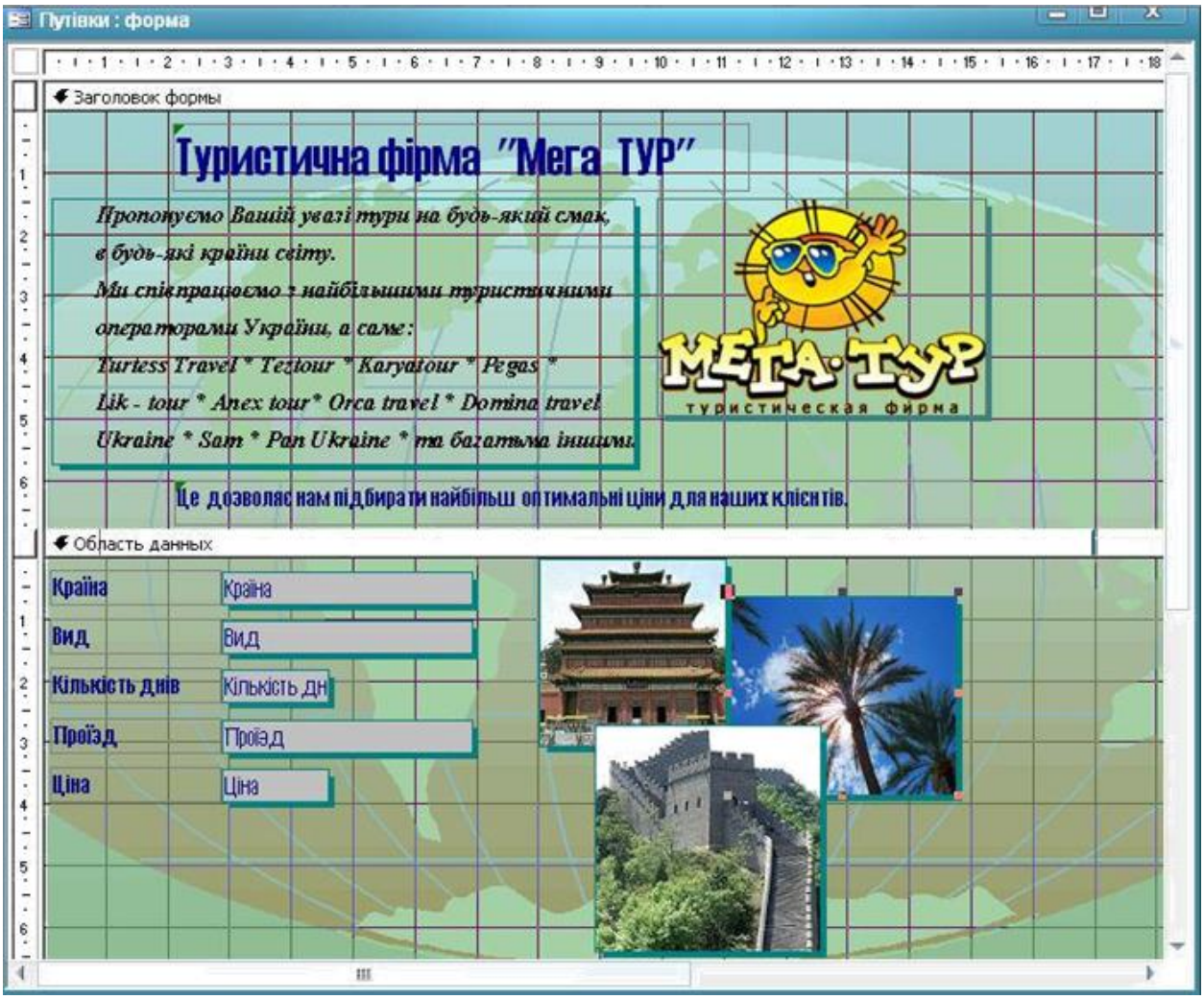
Переконайтеся також, що встановлений перемикач Открытие формы для просмотра или ввода данных і натисніть кнопку Готово.

Внаслідок виконаних дій на екрані з'явиться вікно форми, до якої можна відразу вводити дані.



За допомогою конструктора форм можна удосконалити форму, додати до неї заголовки, примітки, малюнки, управляючі кнопки, тощо.

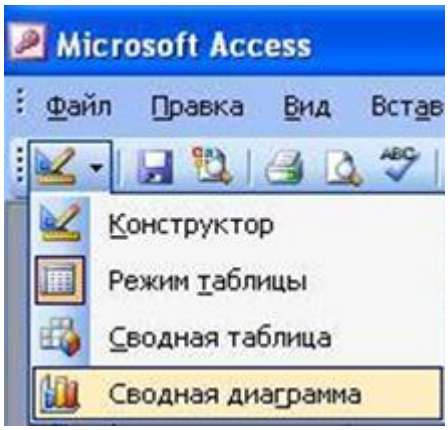

Путівки : форма
Заголовок формы
Область данных
Країна
Країна
Вид
Вид
Кількість днів
Кількість дн
Проїзд
Проїзд
Ціна
Ціна
Примечание формы


Путівки : форма
Заголовок формы
Туристична фірма "Мега ТУР"
Пропонуємо Вашій увазі тури на будь-який смак,
в будь-які країни світу.
Ми співпрацюємо з найбільшими туристичними
операторами України, а саме:
Turiess Travel * Teziour * Karyatour * Pegas *
Lik - tour * Anex tour* Orca travel * Domina travel
Ukraine * Sam * Pan Ukraine * та багатьма іншими.
Це дозволяє нам підбирати найбільш оптимальні ціни для наших клієнтів.
МЕГА-ТУР
туристическая фирма
Область данных
Країна
Країна
Вид
Вид
Кількість днів
Кількість дн
Проїзд
Проїзд
Ціна
Ціна

Введення даних до таблиці

Після задания структури й імені таблиці ви можете безпосередньо ввести дані до неї. Для цього потрібно перейти в Режим таблиці. Цей режим установлюється подвійним клацанням мишею по піктограмі таблиці (або виділенням піктограми таблиці і натисканням кнопки Відкрити після відкриття бази даних. На екрані з'явиться вікно із зображенням рядків таблиці.

Порівняйте вікно із вікном конструктора таблиць. До речі, найзручніше переключатися між режимами конструктора і таблиці за допомогою кнопки Вид на панелі інструментів. Ця кнопка має вигляд Режим таблицы, якщо включений режим конструктора, і вигляд Конструктор, якщо активним є вікно таблиці.



Коли ви заповните перший рядок таблиці, в комірці «Код путівки» з'явиться 1, а напис (Счетчик) автоматично переміститься в наступний рядок.

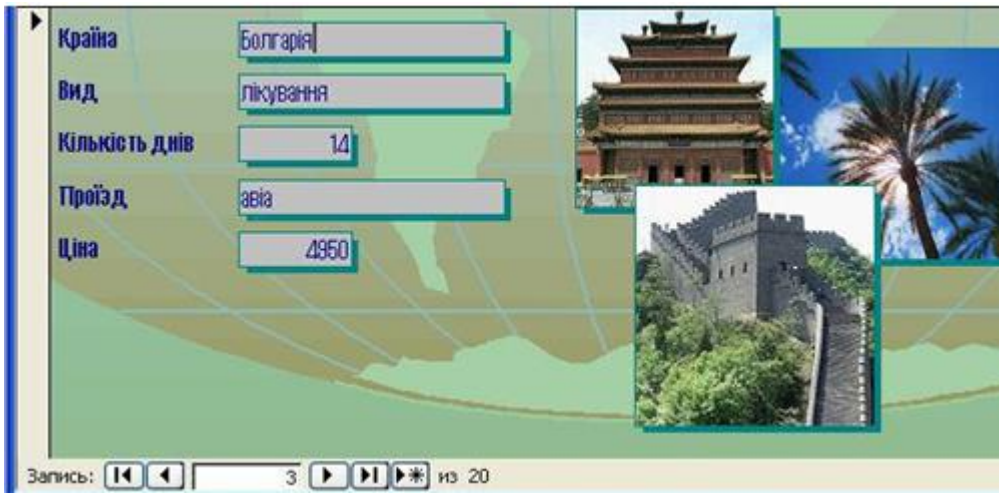
У процесі заповнення таблиці ви можете переміщатися між різними полями і рядками за допомогою клавіш керування курсором, а також клавіші Tab (клавіші Shift+Tab забезпечують переміщення в зворотному напрямку).

Заповнюючи таблиці (а також форми), ви можете використовувати звичайні прийоми редагування, відомі вам по роботі в програмах Блокнот і Word (прийоми вставки і видалення символів, використання буфера обміну тощо).

Введення даних за допомогою форми

Найзручнішим способом введення записів у базу даних є заповнення форм. Припустимо, що ви вже створили форму «Путівки», як було описано в пункті «Форми і їхнє створення» попереднього параграфа. Відкрийте вікно бази даних і перейдіть на вкладку Форми. Двічі клацніть мишею по піктограмі «Путівки», після чого відкриється вікно форми.

Зверніть вагу, що у нижній частині форми розміщені кнопки панелі переходу, які дозволяють переміщатися по записах.



На панелі переходу індикатор записів відображає номер поточного запису. Кнопки на цій панелі дозволяють переходити до наступного або попереднього записів, а також у кінець або на Початок набору записів. У полі індикатора записів можна також зазначити Номер запису, до якого бажаєте перейти.

Щоб додати новий запис, потрібно клацнути по кнопці Новий запис Панелі переходу і потім ввести дані в поля форми.

Введені дані будуть розміщені в таблиці «Путівки» після закриття вікна Форми.

Тема 3.1.4 Поняття запиту. Види запитів

(2 години)

Мета: знати що таке «запит», які види запитів бувають та призначення кожного з цих видів.

Структура заняття:

I. Організаційний момент:
   a. Готовність групи до заняття;
   b. Психоемоційний настрій;
   c. Перевірка присутніх.

II. Повідомлення теми та мети заняття;

III. Виклад нового матеріалу:

План

1. Запит як об’єкт бази даних СУБД MS Access.
2. Способи створення запитів.
3. Види запитів.
4. Вирази в запитах.

IV. Узагальнення та систематизація знань;

V. Підведення підсумків занять;

VI. Домашнє завдання:

1. Ознайомитись з теоретичними відомостями теми 3.1.4.
2. Вивчити основні поняття лекції.
3. Дати відповіді на контрольні питання.

Контрольні питання:

1. Що таке запит?

2. Якими способами можна створити запити?

3. Які існують види запитів?

Таблиці «Клієнти» і «Замовлення»

Для ефективної роботи нашої бази даних необхідно побудувати ще дві таблиці бази даних «Фірма «Подорож». В одній із них будуть відомості про постійних клієнтів фірми, які купують путівки (таблиця «Клієнти»), а в другій - відомості про зроблені замовлення (таблиця «Замовлення»).

Доступ до інформації в базі даних забезпечується таким інструментом, як *запити.* Запити дозволяють відібрати дані, що містяться в різних таблицях бази, а також виконати відбір відповідно до заданих умов (наприклад, список товарів, не дорожчих заданої ціни, дані про клієнтів у певному регіоні тощо). Тобто запити нагадують розглянуті раніше фільтри, однак запити - гнучкіший інструмент доступу до інформації. Так, за допомогою запитів можна не лише добути інформацію з БД, а й формувати нові поля, яких немає у

первинних таблицях. У запитах можна обробляти початкові дані (знаходження середнього, максимального значення, підсумовування тощо).

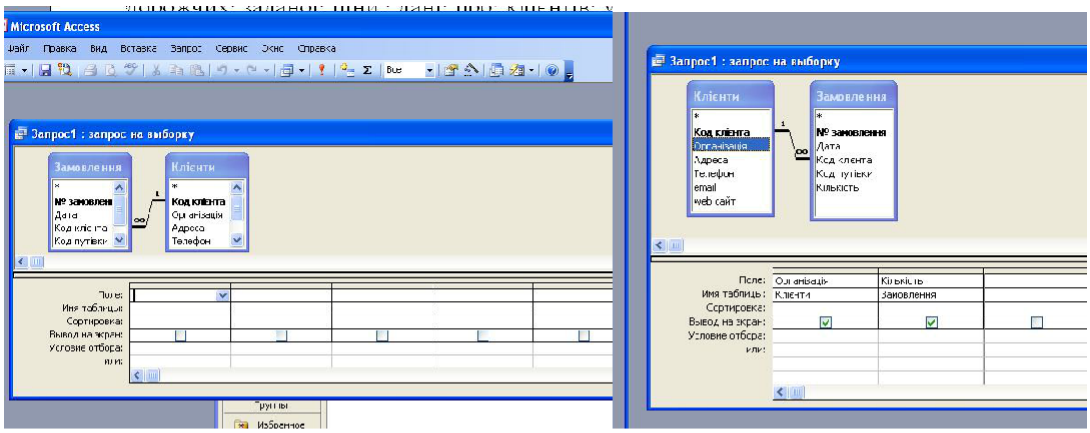
Простий запит

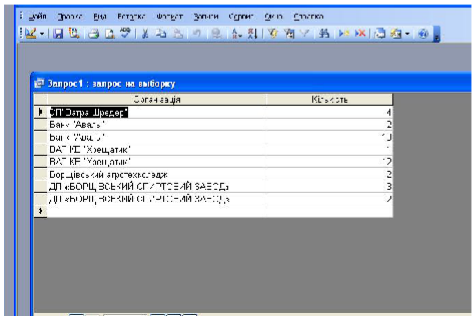
Створення запиту в Access (як і створення інших об'єктів) можливо здійснити за допомогою майстра або в режимі конструктора. Під час ознайомлення технологією запитів ми використовуватимемо майстер простих запитів.

Втримуючись його інструкцій, ви зможете обрати потрібну таблицю і поля них, переглянути результати відбору на екрані.

Наприклад, вас цікавить формація щодо того, які організації й у якій кількості замовляли путівки. Очевидно, що такий запит має бути зроблений на основі таблиць «Замовлення» і «Клієнти». Вважатимемо, що відповідні таблиці побудовані і що між ними встановлено зв'язок.

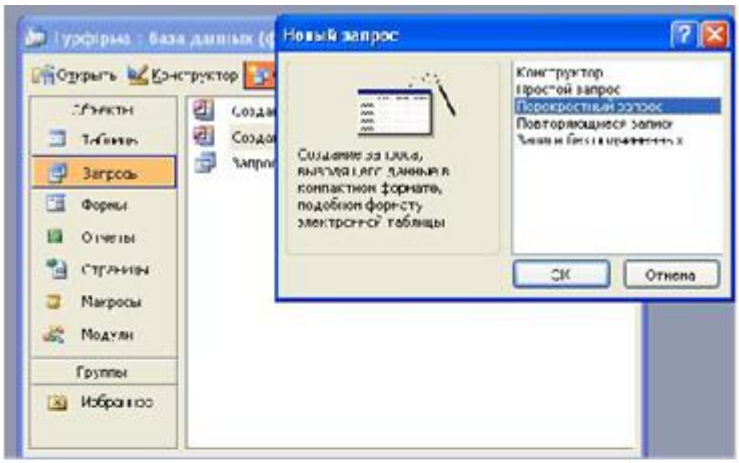
Відкрийте вікно бази даних, перейдіть на вкладку Запити і натисніть кнопку Створити. У діалозі Новий запит. У першому вікні майстра простих запитів зазначте, що ви створюєте запит на основі таблиці «Замовлення». Для цього в списку Таблиці/запити виділіть опцію «Таблиця: Замовлення». Потім у списку Доступні поля клацніть по позиції «Код клієнта» і натисніть кнопку із символом . Зазначена вами позиція переміститься до списку Выбранные поля.

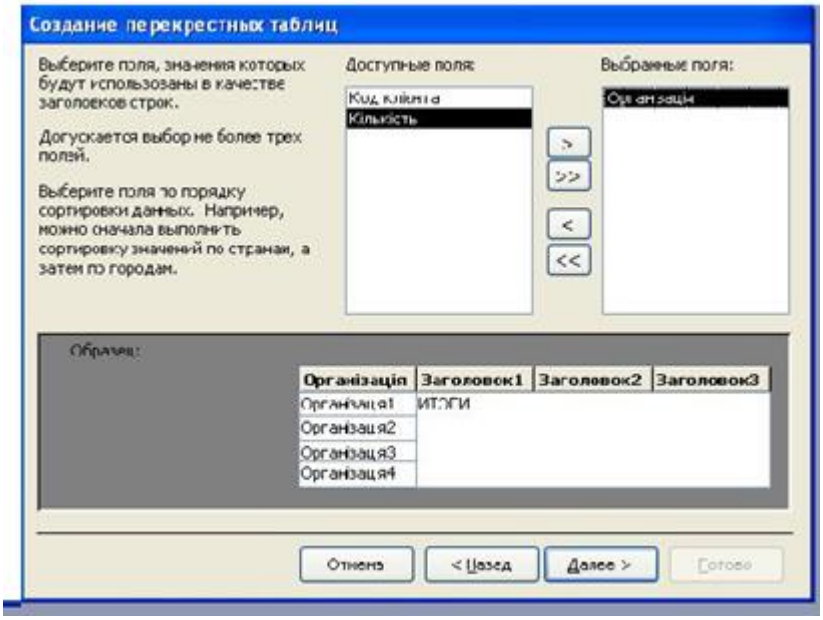


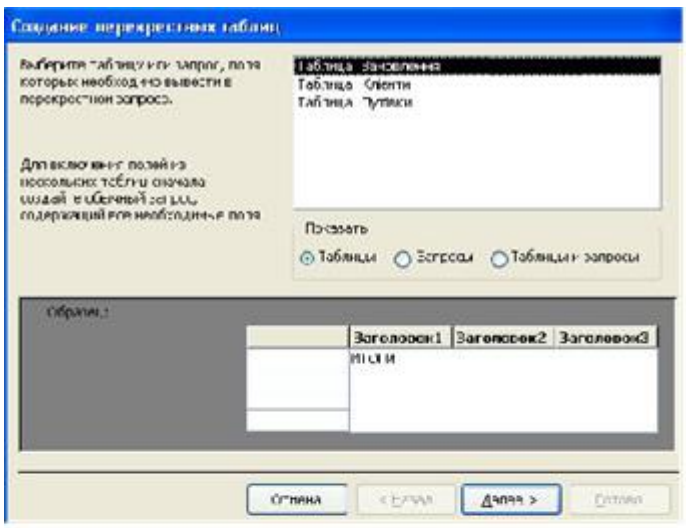


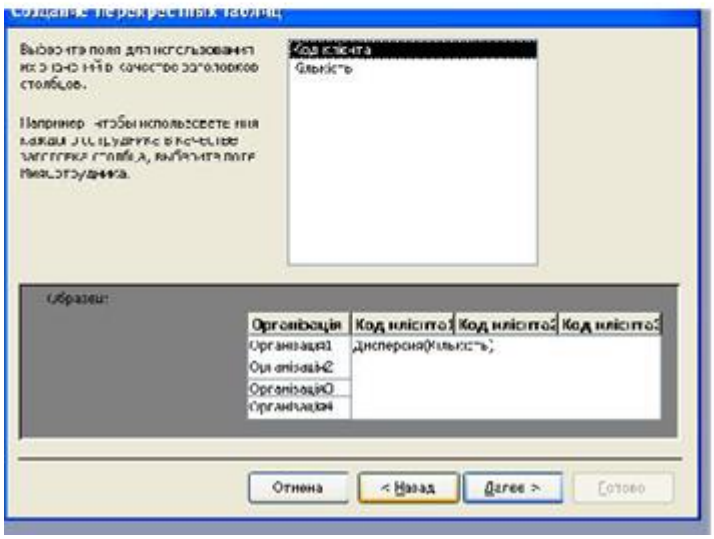
Перехресний запит

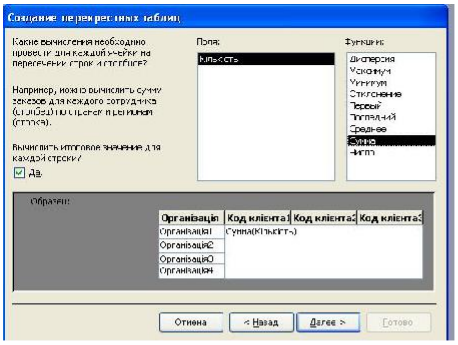
Ефективним засобом аалізу даних є перехресний запит. Цей запит дозволяє згруповувати дані рядків або стовпців і виводити підсумкові значення до окремого стовпця.

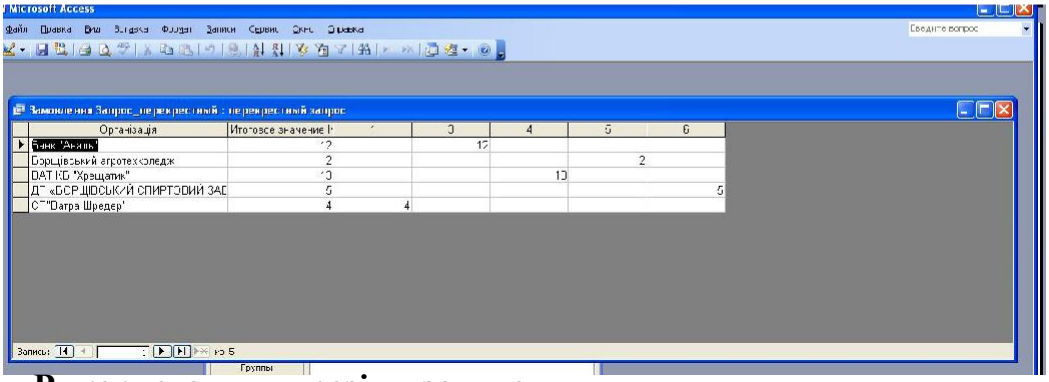



Создание перекрестных таблиц
Выберите поля, значения которых будут использованы в качестве заголовков строк.
Допускается выбор не более трех полей.
Выберите поля по порядку сортировки данных. Например, можно сначала выполнить сортировку значений по странам, а затем по городам.
Доступные поля:
Код клиента
Кількість
Выбранные поля:
Организация
>
>>
<
<<
Образец:
Організація
Заголовок1
Заголовок2
Заголовок3
Организация1
ИТОГИ
Организация2
Организация3
Организация4
Отмена
< Назад
Далее >
Готово


Создание перекрестных таблиц
Таблица: Заявления
Таблица: Клиенти
Таблица: Путяки
Показать
Таблицы
Запросы
Таблицы и запросы
Образец:
Заголовок1
Заголовок2
Заголовок3
ИТОГИ
Отмена
Далее >


Код клиента
Кількість
Образец:
Організація
Код клієнта1
Код клієнта2
Код клієнта3
Организация1
Дисперсия(Кількість)
Организация2
Организация3
Организация4
Отмена
< Назад
Далее >
Готово





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.

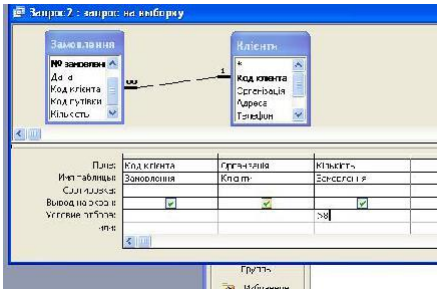
Надалі ми використовуватимемо вирази при заданні умов відбору і під час формування поля, що обчислюється в запиті.

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.

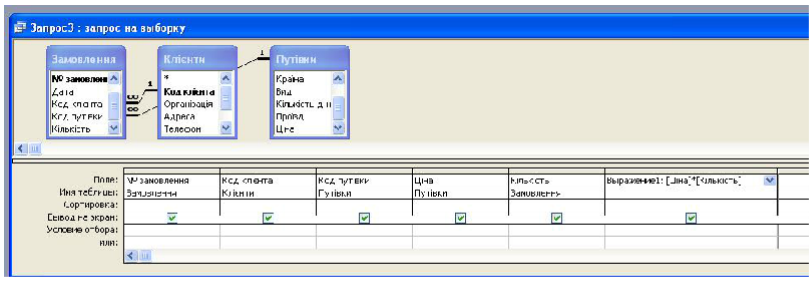
Розрахунки в запиті

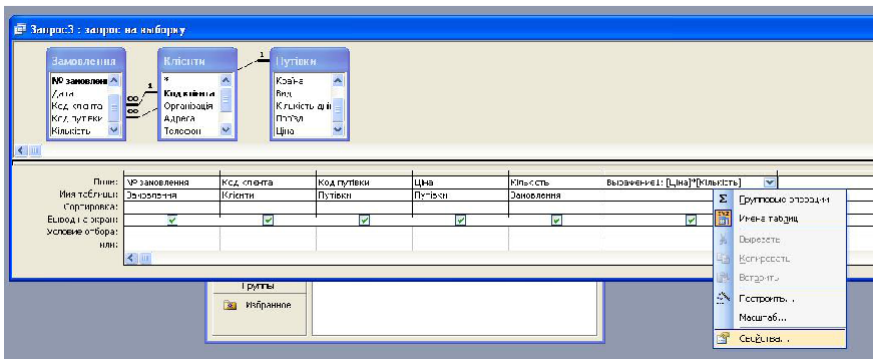
Проілюструємо виконання розрахунків на прикладі запиту, сформованого на основі таблиць «Замовлення» і «Путівки». Нас цікавитиме сума кожного замовлення, що обчислюється як добуток ціни путівки та кількості путівок: [Ціна]*[Кількість]. Виконується подібний запит таким чином.

Спочатку сформуйте запит, показаний на рисунку.

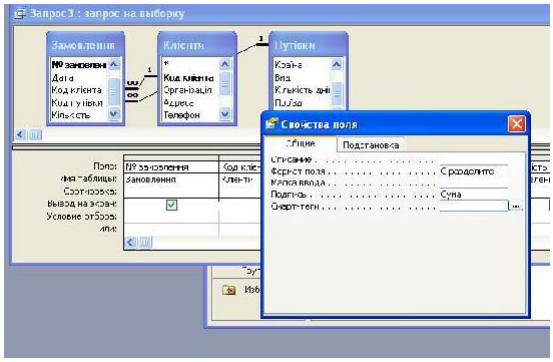
Передемо в режим конструктора запитів, клацнвши на кнопці Вид н панелі інструментів.

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

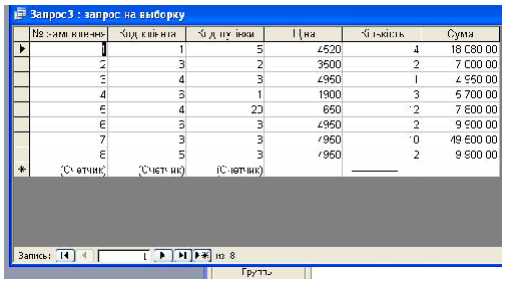




Клацніть правою кнопкою миші в зоні поля з виразом і оберіть у контекстному меню команду Свойства. У діалозі Свойства поля задайте формат поля С разделителем (два десяткові знаки після коми) і назву поля «Сума».



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.

Блок змістових модулів №4

СУБД в архітектурі «Клієнт - сервер»

Змістовий модуль 4.1 - Архітектура «Клієнт - сервер»

Тема 4.1 1 Архітектура «Клієнт-сервер». Моделі «Клієнт-сервер»

(1 година)

Мета: Знати про архітектуру «Клієнт - сервер» та про моделі архітектури «Клієнт - сервер».

Структура заняття:

I. Організаційний момент:

a. Готовність групи до заняття;

b. Психоемоційний настрій;

c. Перевірка присутніх.

II. Повідомлення теми та мети заняття;

III. Виклад нового матеріалу:

План

1. Архітектура «Клієнт - сервер».

2. Моделі архітектури «Клієнт - сервер».

IV. Узагальнення та систематизація знань;

V. Підведення підсумків занять;

VI. Домашнє завдання:

1. Ознайомитись з теоретичними відомостями теми 4.1.1.
2. Вивчити основні поняття лекції.
3. Дати відповіді на контрольні питання.

Контрольні питання:

1. Перелічіть основні функції стандартного інтерактивного додатка, пов'язаного з обробкою баз даних.

2. Перелічіть основні завдання презентаційної логіки.

3. Що містить у собі бізнес-логіка додатка?

4. Перелічіть варіанти розподілу функцій стандартного інтерактивного додатка в архітектурі клієнт-сервер.

5. Назвіть основні дворівневі моделі архітектури клієнт-сервер.

6. У чому полягають особливості моделі віддаленого доступу до даних?

7. Якими засобами реалізується бізнес-логіка при використанні моделі віддаленої презентації?

1. Моделі клієнт-сервер у технології БД

Обчислювальна модель клієнт-сервер початково пов'язана з парадигмою відкритих систем, яка з'явилася в 90-х роках і швидко розвивалася. Термін клієнт-сервер початково застосовували до архітектури, за якої клієнтський процес запитує деякі послуги, а серверний процес забезпечує їх виконання.

Реалізація архітектури клієнт-сервер, стосовно розробки БД, дає змогу більш повно використовувати ресурси мережі. Навантаження рівномірно розподіляється

між комп'ютером сервером і комп'ютером клієнтом, який, як і сервер, володіє власними ресурсами.

Основний принцип технології клієнт-сервер стосовно технології БД полягає в поділі функцій стандартного інтерактивного додатка на 5 груп, що мають різну природу:

- Функції введення і відображення даних (Presentation Logic).

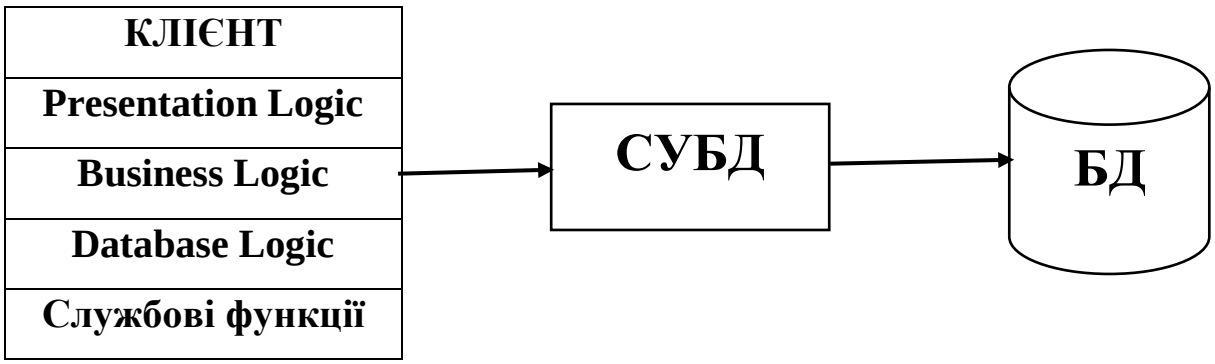
- Прикладні функції, що визначають основні алгоритми розв'язання задач програми (Business Logic).

- Функції опрацювання даних усередині програми (Database Logic).

- Функції управління інформаційними ресурсами (Database Manager System).

- Службові функції, що відіграють роль зв'язок між функціями перших 4-х груп.

Структуру типового інтерактивного додатка, що працює з БД, наведено на рисунку:



Презентаційна логіка - це частина програми, яка визначає те, що користувач бачить на екрані. Сюди відносяться інтерфейсні екранні форми, а також все, що виводиться користувачеві на екран, як результати розв'язання проміжних завдань або довідкова інформація.

Основними завданнями презентаційної логіки є:

- формування екранних зображень;

- читання і запис в екранні форми інформації;

- управління екраном;

- обробка рухів миші та натискання клавіш клавіатури.

Бізнес-логіка або логіка застосунків - це частина коду застосунку, яка визначає власне алгоритми розв'язання задач застосунку. Зазвичай цей код пишеться за допомогою різних мов програмування: C, Cobol, Visual Basic.

Логіка опрацювання даних - це частина коду застосунку, яка пов'язана з опрацюванням даних усередині застосунку. Даними керує власне СУБД. Для забезпечення доступу до даних використовуються мова запитів і засоби маніпулювання даними мови SQL. Процесор управління даними - це власне СУБД, яка забезпечує зберігання та управління БД.

У централізованій архітектурі ці функції розташовуються в єдиному середовищі та комбінуються всередині виконуваної програми. У децентралізованій архітектурі ці завдання можуть бути по-різному розподілені між серверним і клієнтським процесами. Залежно від характеру розподілу можна виділити такі моделі розподілів:

- розподілена презентація; (частина подання на клієнті, частина на сервері, на сервері - усі інші частини)

- віддалена презентація; (уся презентація на клієнті - все інше на сервері)

- розподілена бізнес-логіка; (презентація і частина бізнес-логіки на клієнті)

- розподілене управління даними; (презентація, бізнес-логіка, і частина управління даними на клієнті);

- віддалене управління даними (презентаційна і бізнес-логіка на клієнті, решта на сервері).

- розподілена БД.

Ця класифікація показує, як завдання можуть бути розподілені між серверним і клієнтським процесами.

Дворівневі моделі

Дворівнева модель передбачає розподіл усіх зазначених функцій між 2-ма процесами, які виконуються на 2-х платформах - клієнті та сервері.

У чистому вигляді майже ніяка модель не існує.

2. Модель віддаленого доступу до даних

У цій моделі презентаційна логіка і бізнес-логіка розташовуються на клієнті. На сервері розташовується база даних і ядро СУБД (див. рисунку).

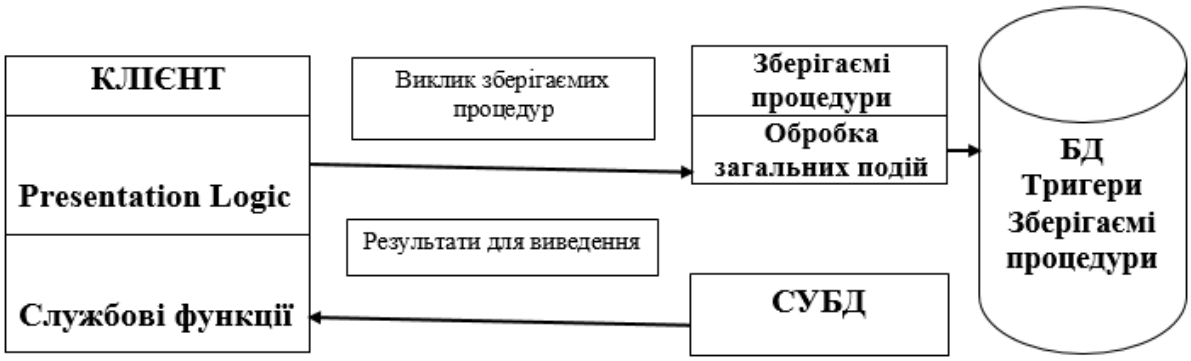
Звернення за сервісом управління даними відбувається за допомогою мови SQL. Перевага моделі - наявність великої кількості готових СУБД, що мають SQL-інтерфейси і набір інструментальних засобів, що забезпечують створення клієнтських додатків. У цій моделі мережею передаються SQL-запити, у відповідь на запити клієнт отримує не блоки файлів, а тільки дані, релевантні запиту. Основна перевага - уніфікація інтерфейсу клієнт-сервер, стандартом при спілкуванні клієнта і сервера стає мова SQL. Недоліки: досить високе завантаження системи передачі даних, внаслідок того, що вся логіка зосереджена в додатку, а оброблювані дані розташовані на віддаленому вузлі.

Ці системи незручні з точки зору модифікації та супроводу. Навіть у разі незначної зміни функцій системи потрібна переробка всієї прикладної частини. Оскільки на клієнті розміщена і презентаційна логіка, і бізнес-логіка застосунку, то в разі повторення аналогічних функцій у різних застосунках код відповідної бізнес-логіки має бути повторено для кожного клієнтського застосунку.

Сервер у цій моделі відіграє пасивну роль, тому функції інформаційного управління мають виконуватися на клієнті. Наприклад, якщо необхідно виконувати контроль страхових запасів на складі, то кожен застосунок, що пов'язаний зі зміною стану складу, після виконання операцій модифікації даних, що імітують продаж або видалення товару зі складу, має виконувати перевірку на обсяг залишку. У разі якщо він менший за страховий запас, необхідно формувати відповідну заявку на поставку необхідного товару. Це може викликати необґрунтоване замовлення додаткових товарів кількома додатками.

3. Віддалена презентація (Модель сервера БД)

Модель сервера БД наведено на рисунку.



Модель сервера БД вирізняється тим, що функції комп'ютера клієнта обмежуються поданням інформації, тоді як прикладні функції забезпечуються додатком, який знаходиться на комп'ютері-сервері. Ця модель є більш технологічною, ніж модель віддаленого доступу.

Для того щоб позбутися недоліків моделі віддаленого доступу, мають бути дотримані такі умови:

- Необхідно, щоб БД у кожен момент відображала поточний стан предметної області.

- БД має відображати деякі правила предметної області, закони, за якими вона функціонує. Наприклад, завод може нормально функціонувати тільки в тому разі, коли є достатній запас деталей певної номенклатури, деталь можна запустити у виробництво тільки в тому разі, якщо на складі є достатньо матеріалу для її виготовлення тощо.

- Необхідний постійний контроль над станом БД, відстеження всіх змін і адекватна реакція на них. Наприклад, у разі зменшення товарного запасу нижче за критичний рівень має бути сформована заявка на поставку відповідного товару.

Таку модель підтримують більшість сучасних СУБД: Informix, Oracle, Sybase, MS SQL Server. Основу цієї моделі становить механізм збережених процедур як засіб програмування SQL сервера, механізм тригерів як механізм відстеження поточного стану інформаційного сховища та механізм обмежень на призначені для користувача типи даних, який називається механізмом підтримки доменної структури. Процедури зазвичай зберігаються в словнику БД і поділяються кількома клієнтами. Збережені процедури можуть виконуватися в режимах інтерпретації та компіляції. Клієнтський додаток звертається до сервера з командою запуску збереженої процедури, а сервер виконує цю процедуру і реєструє всі зміни в БД, які в ній

передбачені. Сервер повертає клієнту дані, що відповідають його запиту, які потрібні або для виведення на екран, або для виконання частини бізнес-логіки, яка розташована на клієнті. Трафік обміну інформацією між клієнтом і сервером помітно зменшується.

Централізований контроль цілісності даних у моделі сервера БД виконується з використанням механізму тригерів. Тригери також є частиною БД. Термін тригер узятий з електроніки і семантично точно відображає механізм відстеження спеціальних подій, які пов'язані зі станом БД. Ядро СУБД проводить моніторинг усіх подій, які спричиняють створені та описані тригери в БД, і при настанні відповідної події сервер запускає відповідний тригер. Тригер являє собою деяку програму, яка виконується над БД. Тригери можуть викликати збережені процедури.

У цій моделі сервер є активним, тому що не тільки клієнт, а й сам сервер, використовуючи механізм тригерів, може бути ініціатором обробки даних у БД. І збережені процедури, і тригери зберігаються в словнику БД, вони можуть бути використані кількома клієнтами, що істотно зменшує дублювання алгоритмів обробки даних у різних клієнтських додатках. Для написання збережених процедур і тригерів використовується розширення стандартної мови SQL.

Переваги моделі - можливість гарного централізованого адміністрування додатків на етапах розробки, супроводу та модифікації, а також ефективне використання обчислювальних і комунікаційних ресурсів. Один із недоліків моделі пов'язаний з обмеженнями засобів розробки збережених процедур. Основне обмеження - сильна прив'язка операторів збережених процедур до використовуваної СУБД. Мова написання збережених процедур, по суті, є процедурним розширенням мови SQL і не може змагатися за функціональними можливостями з традиційними мовами, такими як С або Паскаль. Інший недолік - дуже велике завантаження сервера.

Якщо ми переклали більшу частину бізнес-логіки застосунку на сервер, то вимоги до клієнтів у цій моделі різко зменшуються. Іноді таку модель називають моделлю з тонким клієнтом, на відміну від попередніх, де на клієнта покладалися набагато серйозніші завдання.

Можлива модель розподіленої бізнес-функції, у якій загальну частину бізнес-функцій реалізовано на сервері, а специфічні функції опрацювання інформації

знаходяться на клієнті. Функції загального характеру можуть містити стандартне забезпечення цілісності даних, наприклад, у вигляді збережених процедур, а прикладні функції, що залишилися, реалізують спеціальне прикладне оброблення.

4. Модель розподіленої БД

Ця модель передбачає використання потужного комп'ютера клієнта, причому дані зберігаються і на комп'ютері-клієнті, і на комп'ютері-сервері. Взаємозв'язок обох БД може бути 2-х різновидів:

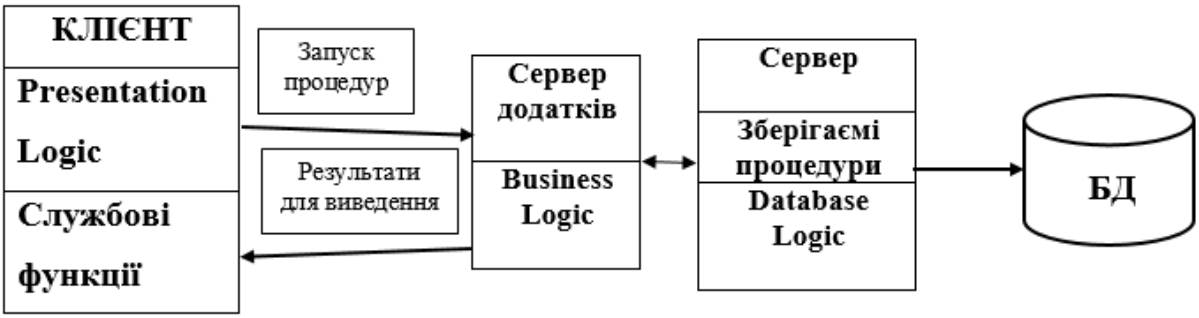
а) у локальній і віддаленій базах зберігаються окремі частини єдиної БД;

б) локальна і віддалена БД є копіями, що синхронізуються одна з одною.

Перевага - гнучкість розроблюваних ІС, що дають змогу комп'ютеру клієнту обробляти і локальні, і віддалені БД. Недолік - високі витрати при виконанні великої кількості однакових додатків на комп'ютерах клієнтах.

5. Модель сервера додатків

Ця модель є розширенням 2-рівневої моделі і в ній вводиться додатковий проміжний рівень між клієнтом і сервером. Цей проміжний рівень містить один або кілька серверів додатків.



У цій моделі компоненти діляться між трьома виконавцями:

6. Клієнт забезпечує логіку подання, включно з графічним користувацьким інтерфейсом; клієнт може запускати локальний код додатка клієнта, що може містити звертання до локальної БД, яка розміщена на комп'ютері-клієнті. Клієнт виконує комунікаційні функції, які забезпечують доступ клієнту в локальну або глобальну мережу.

7. Сервери додатків являють собою новий додатковий рівень архітектури. На сервері додатків реалізується кілька прикладних функцій, кожна з яких оформлена як служба надання послуг усім програмам, які цього потребують. Серверів додатків може бути кілька, причому кожен з них надає свій вид сервісу. Будь-яка програма,

що запитує послугу у сервера додатків, є для нього клієнтом. Запити, що надходять до серверів від клієнтів, поміщаються в чергу.

8. Сервери БД у цій моделі займаються винятково функціями СУБД: забезпечують створення і ведення БД, забезпечують функції сховищ БД, крім того, на них покладаються функції створення резервних копій, відновлення БД після збоїв, управління виконанням транзакцій.

Ця модель має більшу гнучкість, ніж дворівнева модель.