## ЗАГАЛЬНІ ПОНЯТТЯ БАЗ ДАНИХ.

*Поняття бази даних. Характеристики баз даних. Поняття СУБД. Функції СУБД. Ієрархічна модель даних. Мережева модель даних. Реляційна модель даних. Рівні моделі даних.*

### 5.1 Бази даних

#### 5.1.1 Поняття бази даних і СУБД

Сприйняття реального миру можна співвіднести з послідовністю разних, хоча іноді і взаємозв'язаних, явищ. З давніх часів люди намагалися описати ці явища (навіть тоді, коли не могли їх зрозуміти). Такий опис називають ***даними***.

Активна діяльність по відшуканню прийнятних способів усуспільнення безперервно зростаючого об'єму інформації привела до створення на початку 60-х років спеціальних програмних комплексів, званих "***Системи управління базами даних" (СУБД)***.

СУБД - це програмна система, що підтримує наповнення і маніпулювання даними, що представляють інтерес для користувачів при вирішенні прикладних завдань. Іншими словами, СУБД є інтерфейсом між базою даних і прикладними завданнями.

Основна особливість СУБД - це наявність процедур для введення і зберігання не тільки самих даних, але і описів їх структури. Файли, забезпечені описом що зберігаються в них даних і СУБД, що знаходяться під управлінням, почали називати банки даних, а потім бази даних (БД).

Існує велика кількість визначень поняття бази даних. Приведемо декілька визначень.

***База даних*** - сукупність взаємозв'язано що зберігаються разом даних за наявності такої мінімальної надмірності, яка допускає їх використання оптимальним чином для одного або декількох застосувань.

***База даних*** - це реалізована за допомогою комп'ютера інформаційна структура (модель), що відображає стан об'єктів і їх відношення.

***База даних (БД)*** - це засіб накопичення і організації великих масивів інформації про об'єкти деякої предметної області (ПО). БД повинна відображати поточні дані про предметну область, накопичувати, зберігати інформацію і надавати різним категоріям користувачів швидкий доступ до даних. Для цього дані в базі мають бути структуровані відповідно до деякої моделі, що відображає основні об'єкти ПО, їх властивості і зв'язки між ними. БД є частиною складної системи, званої банком даних або системою баз даних (СБД).

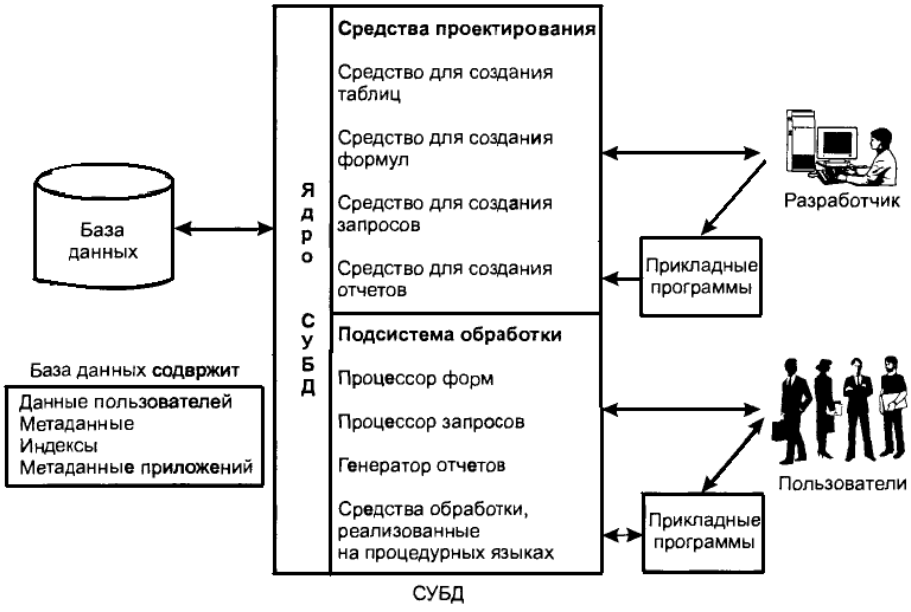



Рис. 5.1.1 Компоненти системи бази даних

Досвід використання баз даних дозволяє виділити загальний набір їх робочих характеристик:

— ***повнота*** - чим повніше база даних, тим ймовірніше, що вона містить потрібну інформацію (проте, не повинно бути надмірної інформації);

— ***правильна організація*** - чим краще структурована база даних, тим легко в ній знайти необхідні відомості;

— ***актуальність*** - будь-яка база даних може бути точною і повною, якщо вона постійно оновлюється, тобто необхідно, щоб база даних в кожен момент часу повністю відповідала стану об'єкту, що відображався нею;

— ***зручність для використання*** - база даних має бути проста і зручна у використанні і мати розвинені методи доступу до будь-якої частини інформації.

Нижче перераховані основні функції СУБД.

1. *Визначення даних* - визначити, яка саме інформація зберігатиметься в базі даних, задасть властивості даних, їх тип (наприклад, число цифр або символів), а також вказати, як ці дані зв'язані між собою. В деяких випадках є можливість задавати формати і критерії перевірки даних.
2. *Обробка даних* - дані можуть оброблятися самими різними способами. Можна вибирати будь-які поля, фільтрувати і сортувати дані. Можна об'єднувати дані з іншою, пов'язаною з ними, інформацією і обчислювати підсумкові значення.
3. *Управління даними* - можна вказати, кому дозволено знайомитися з даними, коректувати їх або додавати нову інформацію. Можна також визначати правила колективного доступу.

Вхідні до складу сучасних СУБД засоби спільно виконують наступні функції:

— опис даних, їх структури (звичайний опис даних і їх структури відбувається при ініціації нової бази даних або додаванні до існуючої бази нових розділів (стосунків); опис даних необхідний для контролю коректності використання даних, для підтримки цілісності бази даних);

— первинне введення, поповнення інформації в базі даних;

— видалення застарілої інформації з бази даних;

— коректування даних для підтримки їх актуальності;

— впорядкування (сортування) даних по деяких ознаках;

— пошук інформації по деяких ознаках (для опису запитів є спеціальна мова запитів, він забезпечує також інтерфейс між базою даних і прикладними програмами користувачів, дозволяє цим програмам використовувати бази даних);

— підготовку і генерацію звітів (засоби підготовки звітів дозволяють створювати і роздруковувати зведення по заданих формах на основі інформації бази даних);

— захист інформації і розмежування доступу користувачів до неї (деякі розділи бази даних можуть бути закриті для користувача зовсім, відкриті тільки для читання або відкриті для зміни; крім того, при многопользовательськом режимі роботи з базою даних необхідно, щоб зміни вносилися коректно; для збереження цілісності даних служить механізм трансакцій при маніпулюванні даними - виконання маніпуляцій невеликими

пакетами, результати кожного з яких у разі виникнення некоректності операцій “відкатуються” і дані повертаються до початкового стану);

— резервне збереження і відновлення бази даних, яке дозволяє відновити втрачену при збоях і аваріях апаратури інформацію бази даних, а також накопичити статистику роботи користувачів з базою даних;

— підтримку інтерфейсу з користувачами, який забезпечується засобами ведення діалогу (у міру розвитку і вдосконалення СУБД цей інтерфейс стає все більш дружнім; дружність існуючих засобів інтерфейсу припускає наявність розвиненої системи допомоги (підказки), до якої у будь-який момент може звернутися користувач, не перериваючи сеансу роботи з комп'ютером і базою даних;

— захист від необдуманих дій, застережливий користувача і запобігаючу втрату інформації у разі поспішних або помилкових команд;

— наявність декількох варіантів виконання одних і тих же дій, з яких користувач може вибрати найбільш зручні для себе, відповідні його підготовці, кваліфікації, звичкам;

— ретельно продуману систему ведення людино-машинного діалогу, відображення інформації на дисплеї, використання клавіш клавіатури).

Розрізняють три типи СУБД:

— ієрархічна;

— мережева;

— реляційна.

**5.1.2 Ієрархічна модель даних**

Найбільш відомим і поширеним представником такої моделі даних є СУБД IMS (Information Management System) компанії IBM. Перша версія системи з'явилася в 1968 р.

Ієрархічна БД складається з впорядкованого набору дерев; точніше, з впорядкованого набору декількох екземплярів одного типу дерева. Тип дерева складається з одного «кореневого» типу запису і впорядкованого набору з нуля або більш за типи піддерев (кожне з яких є деяким типом дерева). Тип дерева в цілому є ієрархічно організованим набором типів запису. Або іншими словами, дані представляються у вигляді дерева з одним кореневим вузлом і з умовами, що кожен вузол нижче кореневого може бути пов'язаний з одним вищестоящим вузлом і з декількома нижчестоячими вузлами.

Розглянемо приклад.


Рис.5.1.2 Приклад типу дерева

У розглянутому прикладі тип запису Відділ є предком для типів запису Керівник і Службовці, а Керівник і Службовці - нащадки типу запису Відділ. Сенс полів типів записів в основному має бути зрозумілий по їх іменах. Поле Рук_отдел типу запису Керівник містить номер відділу, в якому працює службовець, що є даним керівником (передбачається, що він працює не обов'язково в тому ж відділі, яким керує). Між типами запису підтримуються зв'язки.

Один екземпляр дерева приведеного в прикладі мав би наступний вигляд:

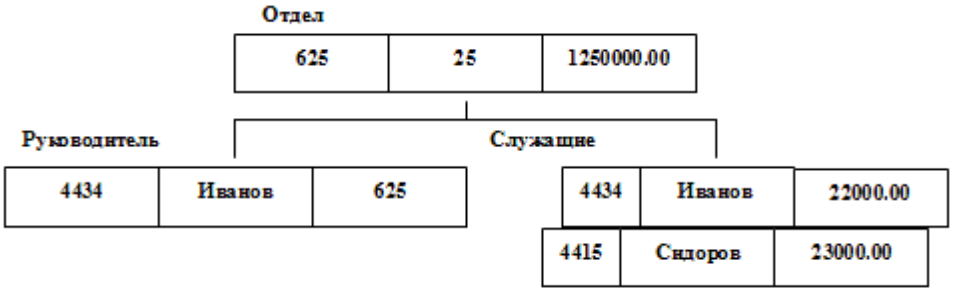



Рис.5.1.3 Приклад ієрархічної бази даних

Всі екземпляри даного типу нащадка із загальним екземпляром типу предка називаються ***близнюками***. Для ієрархічної бази даних визначається повний порядок обходу дерева: зверху-вниз, зліва-направо.

У ієрархічній моделі даних автоматично підтримується цілісність посилань між предками і нащадками. Основне правило: ніякий нащадок не може існувати без свого батька.

Недоліки: якщо дані не мали деревовидної структури, то виникала маса складнощів при побудові ієрархічної моделі і бажанні добитися потрібної продуктивності.

### 5.1.3 Мережева модель даних

Типовим представником систем, заснованих на мережевій моделі даних, є СУБД IDMS (Integrated Database Management System), розроблена компанією Cullinet Software, Inc. і спочатку орієнтована на використання на мейнфреймах компанії IBM. Архітектура системи заснована на пропозиціях Data Base Task Group (DBTG) організації CODASYL (Conference on Data Systems Languages), яка відповідала за визначення мови програмування COBOL. Звіт DBTG був опублікований в 1971 р., і незабаром після цього з'явилося декілька систем, що підтримують архітектуру CODASYL, серед яких присутня і СУБД IDMS. В даний час IDMS належить компанії Computer Associates.

Мережевий підхід до організації даних є розширенням ієрархічного підходу. У ієрархічних структурах запис-нащадок повинен мати в точності одного предка; у мережевій структурі даних у нащадка може бути будь-яке число предків.

Мережева БД складається з набору записів і набору зв'язків між цими записами, а якщо говорити точніше, з набору екземплярів кожного типу із заданого в схемі БД набору типів запису і набору екземплярів кожного типу із заданого набору типів зв'язку.

Тип зв'язку визначається для двох типів запису: предка і нащадка. Екземпляр типу зв'язку складається з одного екземпляра типу запису предка і впорядкованого набору екземплярів типу запису нащадка. Для даного типу зв'язку L з типом запису предка P і типом запису нащадка C повинні виконуватися наступні дві умови:

— кожен екземпляр типу запису P є предком тільки в одному екземплярі типу зв'язку L;

— кожен екземпляр типу запису C є нащадком не більше ніж в одному екземплярі типу зв'язку L.

Розглянемо приклад схеми мережевої БД.

Рис.5.1.4 Приклад схеми мережевої бази даних

На рисунку показано три типи запису: *Відділ, Службовці* і *Керівник* і три типи зв'язку: *Складається із службовців, Має керівника* і *Є таким, що служить*.

У типі зв'язку *Складається із службовців* типом записи-предком є Відділ, а типом записи-потомком - Службовці (екземпляр цього типу зв'язку зв'язує екземпляр типу запису Відділ з багатьма екземплярами типу запису Службовці, відповідними всім службовцем даного відділу).

У типі зв'язку *Має керівника* типом записи-предком є Відділ, а типом записи-потомком - Керівник (екземпляр цього типу зв'язку зв'язує екземпляр типу запису Відділ з одним екземпляром типу запису Керівник, відповідним керівникові даного відділу).

Нарешті, в типі зв'язку *Є служащим* типом записи-предком, є Керівник, а типом записи-потомком - Службовці (екземпляр цього типу зв'язку зв'язує екземпляр типу запису Керівник з одним екземпляром типу запису Службовці, відповідним тому службовцеві, яким є даний керівник).

Недоліки: складність структури.

### 5.1.4 Реляційна модель даних

За минулі десятиліття реляційна модель розвивалася в двох напрямах. Перший напрям заклав знаменитий експериментальний проект компанії IBM System R. У цьому проекті виникла мова SQL, спочатку заснований на ідеях Кодда, але що порушує деякі принципові розпорядження реляційній моделі.

Другий напрям, починаючи з 1990-х рр., очолює Крістофер Дейт, до якого пізніше прилучився Хью Дарвен. Проте в 1990-і рр. Дейт і Дарвен прийшли до висновку, що спотворення реляційної моделі даних, властиві мові SQL, досягли настільки високого рівня, що прийшло час запропонувати альтернативу, що спирається на неспотворені ідеї Едгара Кодда і забезпечує всі можливості як SQL, так і об'єктно-орієнтованого підходу до організації баз даних і СУБД.

Назва “реляційна” (у перекладі з англійського relation - відношення) пов'язана з тим, що кожен запис в таблиці містить інформацію, що відноситься тільки до одного конкретного об'єкту.

Детальніше реляційну модель даних розглянемо пізніше.

### 5.1.5 Рівні моделі даних.

Проектування бази даних треба починати з аналізу наочної області і виявлення вимог до неї окремих користувачів. Проектування зазвичай доручається людині (групі осіб) - ***адміністраторові бази даних (АБД)***. Їм може бути як спеціально виділений співробітник організації, так і майбутній користувач бази даних, досить добре знайомий з машинною обробкою даних.

Виділяють три рівні моделі даних:

— інфологічна;

— даталогична;

— фізична.


Рис.5.1.5 Рівні моделей даних

***Інфологічна модель*** описує предметну область на змістовному рівні. На першому етапі при її розробці здійснюється аналіз предметної області, вирішуваних завдань, запитів користувачів і документів, що відображають події і процеси, що протікають в ПО. Результатом цього аналізу є списки об'єктів предметної області, переліки їх властивостей або атрибутів, визначення зв'язків між об'єктами і опис структури ПО у вигляді діаграми. Для кожного з атрибутів указуються обмеження на їх можливі значення, визначувані властивостями ПО.

Такі обмеження називаються обмеженнями цілісності даних.

Інфологическая модель об'єднує в єдине узагальнене уявлення вимоги окремих користувачів і служить засобом спілкування між ними, тому розробляється без урахування особливостей представлення даних в пам'яті ЕОМ.

***Концептуальна або даталогическая модель*** описує об'єкти і зв'язки ПО на формальному рівні. Її розробка ведеться на другому етапі і грунтується на інфологической моделі, отриманій на першому етапі. В процесі розробки здійснюється вибір типу моделі даних, і визначаються її елементи. Кожна СУБД підтримує тільки одну з моделей. Вибір моделі даних і вибір СУБД тісно взаємозв'язані.

***Внутрішня, або фізична, модель даних*** визначає спосіб розміщення даних безпосередньо на машинному носієві, враховує розподіл даних, методи доступу і способи індексування. У сучасних прикладних програмних засобах цей рівень організації забезпечується автоматично без втручання користувача. Користувач, як правило, оперує в прикладних програмах і універсальних програмних засобах представленнями СУБД на організацію даних.

Таким чином основне завдання проектування полягає в створенні інфологической моделі ПО і концептуальною БД.

**Контрольні питання**:

1. Що таке БД?
2. Що таке СУБД?
3. Які ви знаєте функції СУБД?
4. Які існують типи СУБД? У чому їх відмінності?
5. Дати визначення інфологической моделі БД.
6. Дати визначення концептуальної моделі БД.

### 5.2. Інфологическа модель даних "суть-зв'язок"

*Основні поняття інфологического моделювання. Суть. Атрибут. Ключ. Зв'язок. Основні класи суті. ER-диаграммы і мова інфологического моделювання. Чотири види зв'язків.*

#### 5.2.1 Поняття, використовувані в інфологичному моделюванні.

Модель була запропонована Пітером Ченом (Peter Chen) в 1976 р. Моделювання наочної області базується на використанні графічних діаграм, що включають невелике число різнорідних компонентів.

Мета інфологичного моделювання - забезпечення найбільш природних для людини способів збору і представлення тієї інформації, яку передбачається зберігати в створюваній базі даних. Тому інфологичну модель даних намагаються будувати по аналогії з природною мовою. Основними конструктивними елементами інфологических моделей є суть, зв'язки між ними і їх властивості (атрибути).

***Суть*** - будь-який помітний об'єкт (об'єкт, який ми можемо відрізнити від іншого), інформацію про яке необхідно зберігати в базі даних. Суттю можуть бути люди, місця, літаки, рейси, смак, колір і так далі Необхідно розрізняти такі поняття, як тип суті і екземпляр суті.

Поняття ***тип суті*** відноситься до набору однорідних осіб, предметів, подій або ідей, промовців як ціле. ***Екземпляр суті*** відноситься до конкретної речі в наборі. Наприклад, типом суті може бути МІСТО, а екземпляром - Москва, Київ і так далі

***Атрибут*** - пойменована характеристика суті. Його найменування має бути унікальним для конкретного типу суті, але може бути однаковим для різного типу суті (наприклад, КОЛІР може бути визначений для багатьох суті: СОБАКА, АВТОМОБІЛЬ, ДІМ і так далі). Атрибути використовуються для визначення того, яка інформація має бути зібрана про суть. Прикладами атрибутів для суті АВТОМОБІЛЬ є ТИП, МАРКА, НОМЕРНИЙ ЗНАК, КОЛІР і так далі

Тут також існує відмінність між типом і екземпляром. Тип атрибуту КОЛІР має багато екземплярів або значень: Червоний, Синій, Банановий, Біла ніч і так далі, проте кожному екземпляру суті привласнюється тільки одне значення атрибуту.

***Ключ*** - мінімальний набір атрибутів, по значеннях яких можна однозначно знайти необхідний екземпляр суті. Мінімальність означає, що виключення з набору будь-якого атрибуту не дозволяє ідентифікувати суть по тих, що залишилися. Для суті Розклад ключем є атрибут «Номер рейса» або набір: «Пункт відправлення», «Час вильоту» і «Пункт призначення» (за умови, що з пункту в пункт вилітає в кожен момент часу один літак).

***Зв'язок*** - асоціювання два або більш за суть. Якби призначенням бази даних було тільки зберігання окремих, не зв'язаних між собою даних, то її структура могла б бути дуже простій. Проте одна з основних вимог до організації бази даних - це забезпечення можливості відшукання однієї суті по значеннях інших, для чого необхідно встановити між ними певні зв'язки. А оскільки в реальних базах даних нерідко містяться сотні або навіть тисячі суті, то теоретично між ними може бути встановлене більше мільйона зв'язків. Наявність такої безлічі зв'язків і визначає складність інфологическьких моделей.

### 5.2.2 Основні класи суті.

Існують три основні класи суті: стрижньові, асоціативні і характеристичні, а також підклас асоціативної суті - позначення.

Стрижньова суть (стрижень) - це незалежна суть. Наприклад стрижнями є: "Студент", "Квартира", "Чоловіки", "Лікар", "Брак".

Асоціативна суть (асоціація) - це зв'язок виду "многие-ко-многим" між двома або більш суттю. Асоціації розглядаються як повноправна суть: вони можуть брати участь в інших асоціаціях і позначеннях точно так, як і стрижньова суть; можуть володіти властивостями, тобто мати не тільки набір ключових атрибутів, необхідних для вказівки зв'язків, але і будь-яке число інших атрибутів, що характеризують зв'язок. Наприклад, асоціація "Брак" містять ключові атрибути "Код_м", "Код_ж" і "Табельний номер чоловіка", "Табельний номер дружини", а також уточнюючі атрибути "Номер свідоцтва", "Дата реєстрації", "Место_регистрациі", "Номер запису в книгу ЗАГС" і так далі

Характеристична суть (характеристика) - це зв'язок виду "многие-к-одной" або "одна-к-одной" між двома суттю (окремий випадок асоціації). Єдина мета характеристики в рамках даної наочної області полягає в описі або уточненні деякій іншій суті.

Позначаюча суть або позначення - це зв'язок виду "многие-к-одной" або "одна-к-одной" між двома суттю і відрізняється від характеристики тим, що не залежить від суті, що позначається.

Позначення і характеристики не є повністю незалежною суттю, оскільки вони припускають наявність деякій іншій суті, яка "позначатиметься" або "характеризуватиметься". Проте вони все ж таки є окремими випадками суті і можуть, звичайно, мати властивості, можуть брати участь в асоціаціях, позначеннях і мати свої власні (нижчого рівня) характеристики. Підкреслимо також, що всі екземпляри характеристики мають бути обов'язково пов'язані з яким-небудь екземпляром суті, що характеризується. Проте допускається, щоб деякі екземпляри суті, що характеризується, не мали зв'язків.

Перевизначимо тепер стрижньову суть як суть, яка не є ні асоціацією, ні позначенням, ні характеристикою. Така суть має незалежне існування.

На закінчення розглянемо приклад побудови інфологическьой моделі бази даних "Живлення", де повинна зберігатися інформація про блюда, їх щоденне споживання, продукти, з яких готуються ці блюда, і постачальників цих продуктів. Інформація використовуватиметься кухарем і керівником невеликого підприємства громадського харчування, а також його відвідувачами.

За допомогою вказаних користувачів виділені наступні об'єкти і характеристики проектованої бази:

— Блюда, для опису яких потрібні дані, що входять в їх кулінарні рецепти: номер блюда (наприклад, з книги кулінарних рецептів), назва блюда, вид блюда (закуска, суп, гарячіше і тому подібне), рецепт (технологія приготування блюда), вихід (вага порції), назва, калорійність і вага кожного продукту, що входить в блюдо.

— Для кожного постачальника продуктів: найменування, адреса, назва продукту, що поставляється, дата постачання і ціна на момент постачання.

— Щоденне споживання блюд (витрата): блюдо, кількість порцій, дата.

Аналіз об'єктів дозволяє виділити:

— *Стрижні*: Блюда, Продукти і Міста;

— *Асоціації*: Склад (пов'язує Блюда з Продуктами) і Постачання (пов'язує Постачальників з Продуктами);

— *Позначення*: Постачальники;

— *Характеристики:* Рецепти і Витрата.

### 5.2.3 ER- діаграми і мова інфологическього моделювання (ЯІМ)

При побудові інфологическьих моделей можна використовувати мову ER-диаграмм.

У них суть зображається поміченими прямокутниками, асоціації - поміченими ромбами або шестикутниками, атрибути - поміченими овалами, а зв'язки між ними - ненапрямленими ребрами, над якими може проставлятися ступінь зв'язку (1 або буква, замінююча слово "багато") і необхідне пояснення.


Рис.5. 2.1 Позначення, використовувані в ER-диаграммах.

Мова ER-диаграмм використовується для побудови невеликих моделей і ілюстрації окремих фрагментів великих. Частіше ж застосовується менш наочна, але змістовніша ***мова інфологическього моделювання (ЯІМ)***, в якому суть і асоціації представляються пропозиціями вигляду:

СУТЬ (атрибут 1, атрибут 2 , ..., атрибут n)
АСОЦІАЦІЯ [СУТЬ S1, СУТЬ S2 ...]
(атрибут 1, атрибут 2, ..., атрибут n)
ХАРАКТЕРИСТИКА (атрибут 1, атрибут 2, ...)

{ СПИСОК, ЩО ХАРАКТЕРИЗУЄТ СУТЬ}
ПОЗНАЧЕННЯ (атрибут 1, атрибут 2, ...)

[СПИСОК, ЩО ХАРАКТЕРИЗУЄТ СУТЬ]

де S - ступінь зв'язку, а атрибути, що входять в ключ, мають бути відмічені за допомогою підкреслення.
Для прикладу бази даних "Живлення" модель на мові ЯЇМ має наступний вигляд:
Блюда (БЛ, Блюдо, Вигляд)
Продукти (ПР, Продукт, Калорійність)
Постачальники (ПОС, Місто, Постачальник) [Місто]
Склад [Блюда M, Продукти N] (БЛ, ПР, Вага (г))
Постачання [Постачальники M, Продукти N] (ПОС, ПР, Дата_п, Ціна, Вага (кг))
Міста (Місто, Країна)
Рецепти (БЛ, Рецепт) { Блюда}
Витрата (БЛ, Дата_р, Порцій) { Блюда}

ER-диаграмма моделі бази даних "Живлення" наступний вигляд:

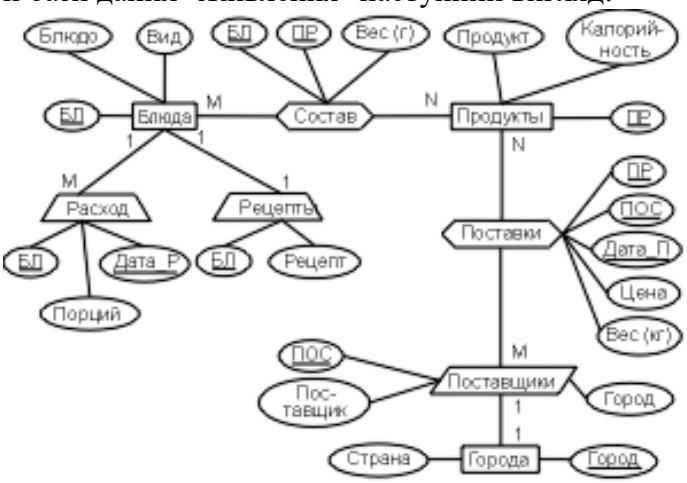
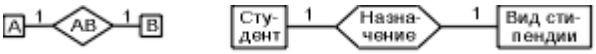



Рис.5.2.2 Інфологическая модель бази даних "Живлення"

**5.2.4 Види зв'язків**

Між двома сутями, наприклад, A і B можливі чотири види зв'язків.

1. зв'язок ***ОДІН-К-ОДНОМУ (1:1)***: у кожен момент часу кожному представникові (екземпляру) суті A відповідає 1 або 0 представників суті B:




Наприклад: студент може не "заробити" стипендію, отримати звичайну або одну з підвищених стипендій.

2. зв'язок ***ОДІН-КО-МНОГИМ (1:м)***: одному представникові суті A відповідають 0, 1 або декілька представників суті B.


Наприклад: квартира може бути порожньою, в ній може жити один або декілька мешканців.

3. ***МНОГИЕ-К-ОДНОМУ (M:1)***
4. ***МНОГИЕ-КО-МНОГИМ (M:N)***.

Приклад. Якщо зв'язок між суттю ЧОЛОВІКА і ЖІНКИ називається БРАК, то існує чотири можливі представлення такого зв'язк:


Характер зв'язків між суттю не обмежується перерахованими. Існують і складніші зв'язки:

— безліч зв'язків між однією і тією ж суттю


(пацієнт, маючи одного лікаря, що лікує, може мати також декілька лікарів-консультантів; лікар може бути лікарем декількох пацієнтів, що лікує, і може одночасно консультувати дещо інших пацієнтів);

— тренарні зв'язки


(лікар може призначити декілька пацієнтів на декілька аналізів, аналіз може бути призначений декількома лікарями декільком пацієнтам і пацієнт може бути призначений на декілька аналізів декількома лікарями);

— зв'язки вищих порядків, семантика (сенс) яких іноді дуже складна.

**Контрольні питання:**

1. Чим відрізняються тип суті та екземпляр суті?
2. Що таке ключ?
3. Які три класи суті ви знаєте?
4. Для чого використовується ER-диаграмм?
5. Чим ER-диаграмм відрізняється від ЯІМ?
6. Які види зв’язків ви знаєте? Приведить приклади?

### 5.3. Реляційна база даних

*Основні поняття. Правила побудови реляційних баз даних. Нормалізація. Процес проектування.*

#### 5.3.1 Основні поняття, використовувані в реляційних базах даних

У 1970 р. Е.Ф. Кодд (E.f. Codd) опублікував свою статтю, в якій він застосував концепції розділу математики, званого реляційною алгеброю, до проблеми зберігання великих об'ємів даних. Стаття Кодда поклала початок руху у сфері проектування баз даних, яке привело декілька років опісля до створення реляційної моделі бази даних. Ця модель є певним способом структуризації і обробки бази даних.

Перевага реляційної моделі полягає в способі зберігання даних, який мінімізує їх дублювання і виключає певні типи помилок обробки, що виникають при інших способах зберігання даних. Дані зберігаються у вигляді таблиць.

Згідно реляційної моделі, не всі види таблиць однаково прийнятні. За допомогою процесу, званого нормалізацією, небажана таблиця може бути перетворена в дві або прийнятніших.

Введемо наступні позначення:

— ***Суть*** - Таблиця (іноді Файл),

— ***Екземпляр суті*** - Рядок (іноді Запис),

— ***Атрибут*** - Стовпець, Поле.

При цьому приймається, що "запис" означає "екземпляр запису", а "поле" означає "ім'я і тип поля".

***Ключ*** або ***можливий ключ*** - це мінімальний набір атрибутів, по значеннях яких можна однозначно знайти необхідний екземпляр суті. Мінімальність означає, що виключення з набору будь-якого атрибуту не дозволяє ідентифікувати суть по тих, що залишилися. Кожна суть володіє хоч би одним можливим ключем. Один з них береться за ***первинний ключ***. При виборі первинного ключа слід віддавати перевагу нескладеним ключам або ключам, складеним з мінімального числа атрибутів. Недоцільно також використовувати ключі з довгими текстовими значеннями (переважно використовувати цілочисельні атрибути). Первинний ключ має бути унікальним.

**Реляційна база даних** - це сукупність стосунків, що містять всю інформацію, яка повинна зберігатися в БД. Проте користувачі можуть сприймати таку базу даних як сукупність таблиць. Так на рис. 3.1 показані таблиці бази даних, побудовані по інфологической моделі бази даних "Живлення" рис. 2.2

Блюда

| БЛ | Блюдо | Вид |
|---|---|---|
| 1 | Лобио | Закуска |
| 2 | Харчо | Суп |
| 3 | Шашлык | Горячее |
| 4 | Кофе | Десерт |

Продукти

| ПР | Продукт | Калор. |
|---|---|---|
| 1 | Фасоль | 3070 |
| 2 | Лук | 450 |
| 3 | Масло | 7420 |
| 4 | Зелень | 180 |
| 5 | Мясо | 1660 |
| 6 | Томаты | 240 |
| 7 | Рис | 3340 |
| 8 | Кофе | 2750 |

Склад

| БЛ | ПР | Вес (г) |
|---|---|---|
| 1 | 1 | 200 |
| 1 | 2 | 40 |
| 1 | 3 | 30 |
| 1 | 4 | 10 |
| 2 | 5 | 80 |
| 2 | 2 | 30 |
| 2 | 6 | 40 |
| 2 | 7 | 50 |
| 2 | 3 | 15 |
| 2 | 4 | 15 |
| 3 | 5 | 180 |
| 3 | 6 | 100 |
| 3 | 2 | 40 |
| 3 | 4 | 20 |

Витрати

| БЛ | Порций | Дата_Р |
|---|---|---|
| 1 | 158 | 1/9/94 |
| 2 | 144 | 1/9/94 |
| 3 | 207 | 1/9/94 |
| 4 | 235 | 1/9/94 |
| ... | ... | ... |

Рецепти

| БЛ | Рецепт |
|---|---|
| 1 | Ломаную очищ |
| ... | ... |

| | | |
|---|---|---|
| 4 | 8 | 8 |

Постачання

| ПОС | ПР | Вес (кг) | Цена | Дата_П |
|---|---|---|---|---|
| 1 | 6 | 120 | 0.45 | 27/8/94 |
| 1 | 3 | 50 | 1.82 | 27/8/94 |
| 1 | 2 | 50 | 0.61 | 27/8/94 |
| 2 | 2 | 100 | 0.52 | 27/8/94 |
| 2 | 5 | 100 | 2.18 | 27/8/94 |
| 2 | 4 | 10 | 0.88 | 27/8/94 |
| 3 | 1 | 250 | 0.37 | 24/8/94 |
| 3 | 7 | 75 | 0.44 | 24/8/94 |
| 3 | 8 | 40 | 2.87 | 24/8/94 |
| 4 | 3 | 70 | 1.56 | 30/8/94 |
| 5 | 5 | 200 | 2.05 | 30/8/94 |
| 6 | 6 | 15 | 0.99 | 30/8/94 |

Постачальники

| ПОС | Поставщик | Город |
|---|---|---|
| 1 | "Полесье" | Киев |
| 2 | "Наталка" | Киев |
| 3 | "Хуанхэ" | Пекин |
| 4 | "Лайма" | Рига |
| 5 | "Юрмала" | Рига |
| 6 | "Даугава" | Рига |

Міста

| Город | Страна |
|---|---|
| Киев | Украина |
| Пекин | Китай |
| Рига | Латвия |

Рис. 5.3.1 База даних "Живлення"

**5.3.2 Правила побудови реляційних баз даних**

Розглянемо основні правила побудови реляційних баз даних:

1. Кожна таблиця складається з однотипних рядків і має унікальне ім'я.

2. Рядки мають фіксоване число полів (стовпців) і значень (множинні поля і групи, що повторюються, недопустимі). Інакше кажучи, в кожній позиції таблиці на перетині рядка і стовпця завжди є в точності одне значення або нічого.

3. Рядки таблиці обов'язково відрізняються один від одного хоч би єдиним значенням, що дозволяє однозначно ідентифікувати будь-який рядок такої таблиці.

4. Стовпцям таблиці однозначно привласнюються імена, і в кожному з них розміщуються однорідні значення даних (дати, прізвища, цілі числа або грошові суми).

5. Повний інформаційний зміст бази даних представляється у вигляді явних значень даних і такий метод уявлення є єдиним. Зокрема, не існує яких-небудь спеціальних "зв'язків" або покажчиків, що сполучають одну таблицю з іншою. Так, зв'язки між рядком з БЛ = 2 таблиці "Блюда" на рис. 3.1 і рядком з ПР = 7 таблиць продукти (для приготування Харчо потрібний Рис), представляється не за допомогою покажчиків, а завдяки існуванню в таблиці "Склад" рядка, в якому номер блюда дорівнює 2, а номер продукту - 7.

6. При виконанні операцій з таблицею її рядка і стовпці можна обробляти у будь-якому порядку безвідносно до їх інформаційного змісту. Цьому сприяє наявність імен

таблиць і їх стовпців, а також можливість виділення будь-якого їх рядка або будь-якого набору рядків з вказаними ознаками (наприклад, рейсів з пунктом призначення "Париж" і часом прибуття до 12 годин).

### 5.3.3 Поняття універсального відношення

Припустимо, що проектування бази даних "Живлення" починається з виявлення атрибутів і підбору даних, зразок яких (частина блюд виготовлених і реалізованих 1/9/94 р.) показаний на рис. 3.2.

Цей варіант таблиці "Живлення" не є відношенням, оскільки більшість її рядків не атомарни. Атомарними є лише значення полів Блюдо, Вигляд, Рецепт (хоча він і великий), Порцій і Дата_р решта полів таблиці рис. 3.2- же множинна. Для додання таким даним форми відношення необхідно реконструювати таблицю. Найпростіше це зробити за допомогою простого процесу вставки, результат якої показаний на рис. 3.3. Проте таке перетворення приводить до виникнення великого об'єму надмірних даних.

Таблиця на рис. 3.3 є екземпляром коректного відношення. Його називають універсальним відношенням проектованої БД. У одне універсальне відношення включаються всі атрибути, що представляють інтерес, і воно може містити всі дані, які передбачається розміщувати в БД в майбутньому. Для малих БД (що включають не більше 15 атрибутів) універсальне відношення може використовуватися як відправна крапка при проектуванні БД.

При використанні універсального відношення виникає декілька проблем:

1. ***Надмірність.*** Дані практично всіх стовпців багато разів повторюються. Повторюються і деякі набори даних (Блюдо-Вид-рецепт, Продукт-Калорійність, Поставщик-Город-страна). Небажане повторення рецептів, деякі з яких набагато більше рецепту "Лобіо". І вже зовсім погано, що всі дані про блюдо (включаючи рецепт) повторюються кожного разу, коли це блюдо включається в меню.

2. ***Потенційна суперечність (аномалії оновлення).*** Унаслідок надмірності можна відновити адресу постачальника в одному рядку, залишаючи його незмінним в інших. Якщо постачальник каві повідомив про свій переїзд до Харбіну і був оновлений рядок з продуктом кави, то у постачальника "Хуанхе" з'являється дві адреси, один з яких не актуальний. Отже, при оновленнях необхідно проглядати всю таблицю для знаходження і зміни всіх відповідних рядків.

3. ***Аномалії включення***. У БД не може бути записаний новий постачальник ("Нярінга", Вільнюс, Литва), якщо продукт (Огірки), що поставляється ним, не використовується ні в одному блюді. Можна, звичайно, помістити невизначені значення в стовпці Блюдо, Вигляд, Порцій і Вес (г) для цього постачальника. Але якщо з'явиться блюдо, в якому використовується цей продукт, чи не забудемо ми видалити рядок з невизначеними значеннями?

По аналогічних причинах не можна ввести і новий продукт (наприклад, Баклажани), який пропонує існуючий постачальник (наприклад, "Полісся"). А як ввести нове блюдо, якщо в нім використовується новий продукт (Краби)?

4. ***Аномалії видалення***. Зворотна проблема виникає при необхідності видалення всіх продуктів, що поставляються даним постачальником або всіх блюд, що використовують ці продукти. При таких видаленнях будуть втрачені відомості про такого постачальника.

Багато проблем цього прикладу зникнуть, якщо виділити в окремі таблиці відомості про блюда, рецепти, витрату блюд, продукти і їх постачальників, а також створити таблиці, що пов'язують, "Склад" і "Постачання".

| Блюдо | Вид | Рецепт | Порций | Дата Р | Продукт | Калор. | Вес (г) | Поставщик | Город | Страна | Вес (кг) | Цена ($) | Дата П |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Лобио | Закуска | Лом. | 158 | 1/9/94 | Фасоль | 3070 | 200 | "Хуанхэ" | Пекин | Китай | 250 | 0.37 | 24/8/94 |
| | | | | | Лук | 450 | 40 | "Наталка" | Киев | Украина | 100 | 0.52 | 27/8/94 |
| | | | | | Масло | 7420 | 30 | "Лайма" | Рига | Латвия | 70 | 1.55 | 30/8/94 |
| | | | | | Зелень | 180 | 10 | "Даугава" | Рига | Латвия | 15 | 0.99 | 30/8/94 |
| Харчо | Суп | ... | 144 | 1/9/94 | Мясо | 1660 | 80 | "Наталка" | Киев | Украина | 100 | 2.18 | 27/8/94 |
| | | | | | Лук | 450 | 30 | "Наталка" | Киев | Украина | 100 | 0.52 | 27/8/94 |
| | | | | | Томаты | 240 | 40 | "Полесье" | Киев | Украина | 120 | 0.45 | 27/8/94 |
| | | | | | Рис | 3340 | 50 | "Хуанхэ" | Пекин | Китай | 75 | 0.44 | 24/8/94 |
| | | | | | Масло | 7420 | 15 | "Полесье" | Киев | Украина | 50 | 1.62 | 27/8/94 |
| | | | | | Зелень | 180 | 15 | "Наталка" | Киев | Украина | 10 | 0.88 | 27/8/94 |
| Шашлык | Горячее | ... | 207 | 1/9/94 | Мясо | 1660 | 180 | "Юрмала" | Рига | Латвия | 200 | 2.05 | 30/8/94 |
| | | | | | Лук | 450 | 40 | "Полесье" | Киев | Украина | 50 | 0.61 | 27/8/94 |
| | | | | | Томаты | 240 | 100 | "Полесье" | Киев | Украина | 120 | 0.45 | 27/8/94 |
| | | | | | Зелень | 180 | 20 | "Даугава" | Рига | Латвия | 15 | 0.99 | 30/8/94 |
| Кофе | Десерт | ... | 235 | 1/9/94 | Кофе | 2750 | 8 | "Хуанхэ" | Пекин | Китай | 40 | 2.87 | 24/8/94 |

Рис. 3.2 Дані, необхідні для створення бази даних "Живлення"

| Блюдо | Вид | Рецепт | Порций | Дата Р | Продукт | Калор. | Вес (г) | Поставщик | Город | Страна | Вес (кг) | Цена ($) | Дата П |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Лобио | Закуска | Лом. | 158 | 1/9/94 | Фасоль | 3070 | 200 | "Хуанхэ" | Пекин | Китай | 250 | 0.37 | 24/8/94 |
| Лобио | Закуска | Лом | 158 | 1/9/94 | Лук | 450 | 40 | "Наталка" | Киев | Украина | 100 | 0.52 | 27/8/94 |
| Лобио | Закуска | Лом | 158 | 1/9/94 | Масло | 7420 | 30 | "Лайма" | Рига | Латвия | 70 | 1.55 | 30/8/94 |
| Лобио | Закуска | Лом | 158 | 1/9/94 | Зелень | 180 | 10 | "Даугава" | Рига | Латвия | 15 | 0.99 | 30/8/94 |
| Харчо | Суп | Лом | 144 | 1/9/94 | Мясо | 1660 | 80 | "Наталка" | Киев | Украина | 100 | 2.18 | 27/8/94 |
| Харчо | Суп | Лом | 144 | 1/9/94 | Лук | 450 | 30 | "Наталка" | Киев | Украина | 100 | 0.52 | 27/8/94 |
| Харчо | Суп | Лом | 144 | 1/9/94 | Томаты | 240 | 40 | "Полесье" | Киев | Украина | 120 | 0.45 | 27/8/94 |
| Харчо | Суп | Лом | 144 | 1/9/94 | Рис | 3340 | 50 | "Хуанхэ" | Пекин | Китай | 75 | 0.44 | 24/8/94 |
| Харчо | Суп | Лом | 144 | 1/9/94 | Масло | 7420 | 15 | "Полесье" | Киев | Украина | 50 | 1.62 | 27/8/94 |
| Харчо | Суп | Лом | 144 | 1/9/94 | Зелень | 180 | 15 | "Наталка" | Киев | Украина | 10 | 0.88 | 27/8/94 |
| Шашлык | Горячее | Лом | 207 | 1/9/94 | Мясо | 1660 | 180 | "Юрмала" | Рига | Латвия | 200 | 2.05 | 30/8/94 |
| Шашлык | Горячее | Лом | 207 | 1/9/94 | Лук | 450 | 40 | "Полесье" | Киев | Украина | 50 | 0.61 | 27/8/94 |
| Шашлык | Горячее | Лом | 207 | 1/9/94 | Томаты | 240 | 100 | "Полесье" | Киев | Украина | 120 | 0.45 | 27/8/94 |
| Шашлык | Горячее | Лом | 207 | 1/9/94 | Зелень | 180 | 20 | "Даугава" | Рига | Латвия | 15 | 0.99 | 30/8/94 |
| Кофе | Десерт | Лом | 235 | 1/9/94 | Кофе | 2750 | 8 | "Хуанхэ" | Пекин | Китай | 40 | 2.87 | 24/8/94 |

Рис.5.3.3 Універсальне відношення "Живлення"

### 5.3.4 Нормалізація

***Нормалізація*** - це розбиття таблиці на дві або більш, що володіють кращими властивостями при включенні, зміні і видаленні даних. Остаточна мета нормалізації зводиться до отримання такого проекту бази даних, в якому кожен факт з'являється лише в одному місці, тобто виключена надмірність інформації. Це робиться не стільки з метою економії пам'яті, скільки для виключення можливої суперечності даних, що зберігаються.

Існують наступні нормальні форми:

Таблиця знаходиться в ***першій нормальній формі (1нф)*** тоді і тільки тоді, коли жодна з її рядків не містить в будь-якому своєму полі більш за одне значення і жодне з її ключових полів не порожньо.

Таблиця знаходиться в другій ***нормальній формі (2нф),*** якщо вона задовольняє визначенню 1нф і всі її поля, що не входять в первинний ключ, зв'язані повною функціональною залежністю з первинним ключем.

Таблиця знаходиться в ***третій нормальній формі (3нф)***, якщо вона задовольняє визначенню 2нф і не одне з її не ключових полів не залежить функціонально від будь-якого іншого не ключового поля.

### 5.3.5 Процедура проектування

Процес проектування інформаційних систем є достатньо складним завданням. Він починається з побудови інфологической моделі даних (п. 2), тобто ідентифікації суті. Потім необхідно виконати наступні кроки процедури проектування даталогической моделі.

1. Представити кожен стрижень (незалежну суть) таблицею бази даних (базовою таблицею) і специфікувати первинний ключ цієї базової таблиці.
2. Представити кожну асоціацію (зв'язок виду "многие-ко-многим" або "многие-ко-многим-ко-многим" і так далі між суттю) як базову таблицю. Використовувати в цій таблиці зовнішні ключі для ідентифікації учасників асоціації і специфікувати обмеження, пов'язані з кожним з цих зовнішніх ключів.
3. Представити кожну характеристику як базову таблицю із зовнішнім ключем, що ідентифікує суть, що описується цією характеристикою. Специфікувати обмеження на зовнішній ключ цієї таблиці і її первинний ключ - ймовірно, комбінації цього зовнішнього ключа і властивості, яка гарантує "унікальність в рамках описуваної суті".
4. Представити кожне позначення, яке не розглядалося в попередньому пункті, як базову таблицю із зовнішнім ключем, що ідентифікує суть, що позначається. Специфікувати пов'язані з кожним таким зовнішнім ключем обмеження.
5. Представити кожну властивість як поле в базовій таблиці, що представляє суть, яка безпосередньо описується цією властивістю.
6. Для того, щоб виключити в проекті ненавмисні порушення яких-небудь принципів нормалізації, виконати процедуру нормалізації.
7. Якщо в процесі нормалізації було проведено розділення яких-небудь таблиць, то слід модифікувати інфологическую модель бази даних і повторити перераховані кроки.
8. Вказати обмеження цілісності проектованої бази даних і дати (якщо це необхідно) короткий опис отриманих таблиць і їх полів.

**Контрольні питання:**

1. Які основні поняття реляційних баз даних ви знаєте?
2. Які основні правила створення реляційної бази даних?

3. Що таке універсальне відношення?
4. Для чого використовується універсальне відношення?
5. Які нормальні форми ви знаєте?
6. Яка процедура проектування реляційної бази даних?

## 5.4. Microsoft Access. Создание таблиц

*Запуск програми Microsoft Access. Об'єкти бази даних Access. Створення таблиць в режимі таблиць. Створення таблиць в режимі конструктора. Створення таблиці за допомогою майстра.*

### 5.4.1 Запуск программы Microsoft Access

Розглядатимемо програму Microsoft Access версії 2003. Запускаючий файл зазвичай знаходиться по наступному шлях:

C:\Program Files\Microsoft Office\Office\MSAccess.exe

Всі можливості Microsoft Access зведені в два меню і одну інструментальну панель. Першим меню є системне, розташоване у верхній частині робочого вікна Microsoft Access. Це Файл, Правка, Вигляд, Вставка, Сервіс, Вікно і Допомога. Другим призначеним для користувача меню є контекстно-залежне, таке, що викликається натисненням правою клавішею миші. Паралельно з меню існує панель інструментів. В результаті роботи програми Microsoft Access створюється файл з розширенням *.mdb.

Вся початкова інформація зберігається в чітко певних таблицях. Під чітким визначенням мається на увазі така структура таблиці, в якій кожен рядок має унікальний ідентифікатор (наприклад, номер рядка), а дані представлені стовпцями. Таким чином, будь-яка таблиця є одновимірним набором записів. Неодмінним правилом створення таблиці в СУБД є строге визначення вмісту самої таблиці. У її осередках може зберігатися тільки фактична, і лише незмінна інформація.

Після запуску програми з'являється вікно наступного вигляду

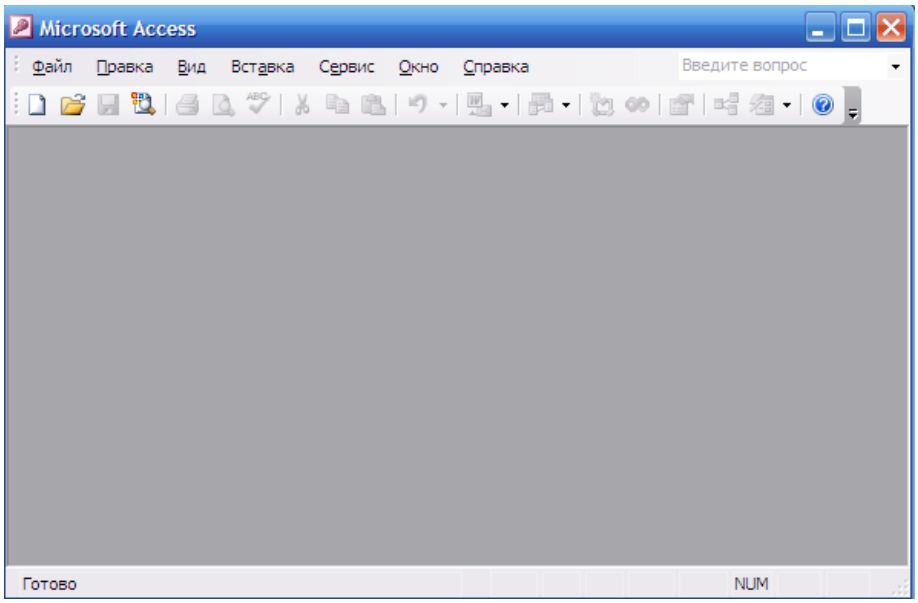



Рис.5.4.1 Вікно запуску Access

Для початку роботи необхідно відкрити існуючу базу даних або створити нову. Для цього необхідно вибрати в меню Файл → Створити ( ) або Файл → Відкрити ( ).

Після вибору одного або іншого способу роботи з базою з'явиться вікно бази даних, що складається з набору панелей з вкладками, кожна з яких відповідає одному з 6 типів об'єктів бази даних Access: **таблиці, запити, форми, звіти, макроси і модулі і сторінки**.

Схема взаємодії об'єктів бази даних зображена на рис. 5.4.2. Основною одиницею зберігання даних тут є таблиця.

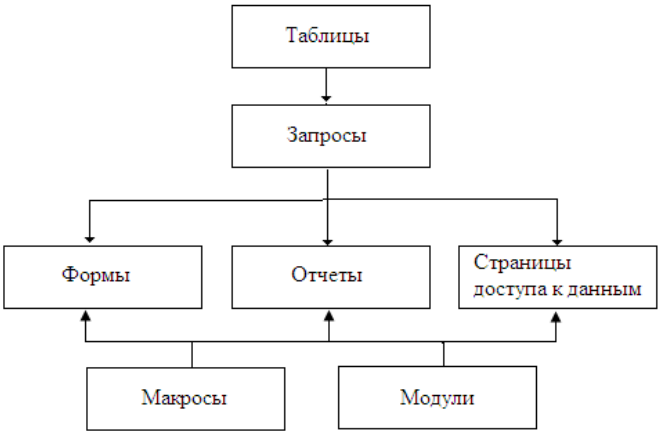



Рис.5.4.2 Схема взаємодії об'єктів бази даних

Розглянемо основне призначення об'єктів бази даних Access.

***Таблиця*** - це об'єкт, визначуваний для зберігання даних. Кожна таблиця включає інформацію про об'єкт реального миру, наприклад про клієнтів фірми. Таблиця складається із заголовка і тіла. Заголовок включає імена атрибутів об'єкту (стовпців) і їх властивості, наприклад прізвище, телефон і адресу клієнта. Тіло містить кортежі (рядки), кожен рядок представляє безліч значень стовпців, в яких зберігаються дані про конкретний екземпляр об'єкту.

***Запит*** - це об'єкт, який дозволяє користувачеві отримати потрібні дані з однієї або декількох базових таблиць, і інших запитів. У запиті можна вказати умови, яким повинні задовольняти дані. Завдяки цьому запит дозволяє з великого масиву інформації, що зберігається в БД, витягувати тільки потрібні дані. Для створення запиту використовують запит за зразком *(QBE)* або інструкції *SQL*.

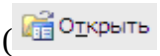
***Форма*** - це об'єкт, призначений в основному для введення даних, відображення їх на екрані або управління роботою додатку. Форми використовуються для того, щоб реалізувати вимоги замовника до представлення даних з таблиць і запитів. Форми можна роздрукувати. За допомогою форми можна у відповідь на деяку подію запустити *макрос* або *процедуру,* що виконують певну обробку даних.

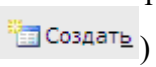
***Звіт*** - це об'єкт, призначений для створення документа, який згодом може бути роздрукований або включений в документ іншого застосування. Перш ніж виводити звіт на принтер, його можна проглянути на екрані.

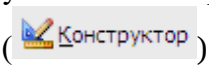
***Макрос*** - це об'єкт, що є структурованим описом одного або декількох дій, які виконуватимуться у відповідь на певну подію. Наприклад, можна визначити макрос, який у відповідь на вибір деякого елементу в основній формі відкриває іншу форму.

***Модуль*** - це об'єкт, що містить програми на Microsoft Access Visual Basic, які можуть розроблятися користувачем для реалізації нестандартних процедур при створенні додатку.

Розглянемо зовнішній вигляд вікна програми (див. рис. 5.4.3).

Кожна область робочого екрану містить вгорі три кнопки: Відкрити ( 

),Створити ( 

) и Конструктор ( 

).

Кнопка ***Відкрити*** призначена для активізації виділеного елементу з наявних в даному проекті. Відкриваючись, ці елементи предстають в тому вигляді, який використовується для його перегляду. Таблиці і запити відкриваються у вигляді таблиці. Форми і звіти - в тому вигляді, в якому вони повинні представати перед користувачем. Лише макроси і модулі відкриваються у вигляді для редагування.

Всі об'єкти в Access можуть бути створені користувачем за допомогою ***конструктора*** або за допомогою різних ***майстрів.*** *Майстри* допомагають користувачеві в режимі діалогу створювати об'єкти, дають підказки, пропонують свої рішення, що полегшує роботу початкуючим і непрофесійним користувачам.

Також для створення нових елементів того або іншого вигляду можна використовувати кнопку ***Створити***.

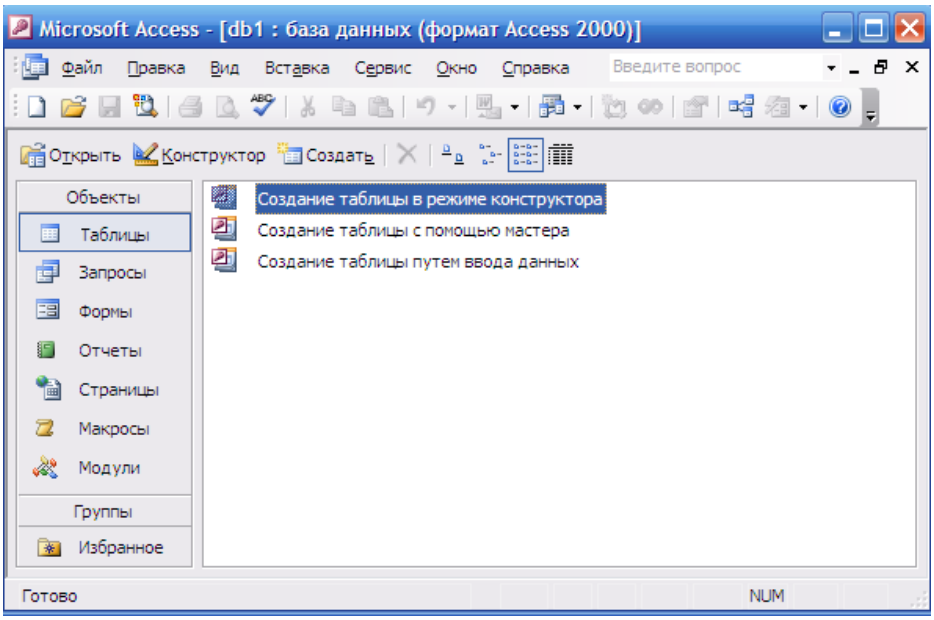



Рис. 5.4.3 Вікно для роботи з базою даних

### 5.4.2 Створення таблиць

При роботі з Access таблиці є одним з основних об'єктів, на їх базі здійснюється побудова всіх інших елементів, таких, як форми, запити і звіти. У таблиці збираються дані по конкретній темі, наприклад вся інформація про клієнтуру фірми.

БД Access може складатися з декількох таблиць, в кожній з яких зберігається інформація на одну тему. У одній таблиці можуть зберігатися відомості про клієнтів, в іншій - про всі торгові угоди, які поміщені з тими або іншими клієнтами, в третій - інформація про витрати, податки і витрати на розвиток фірми, в четвертій - інвентаризаційний список складського фонду, в п'ятій - терміни проведення виставок і презентацій і так далі

У вікні БД клацніть на кнопці Таблиці і натисніть кнопку Створити**,** щоб приступити до проектування нової таблиці. Access відкриє діалогове вікно (див. рис. 5.4.4), в якому буде запропоновано скористатися одним з наступних способів створення таблиці.

***Режим таблиці*** - створення таблиці в табличному уявленні (проектування таблиці відбувається в ході її заповнення по аналізу даних, що вводяться).

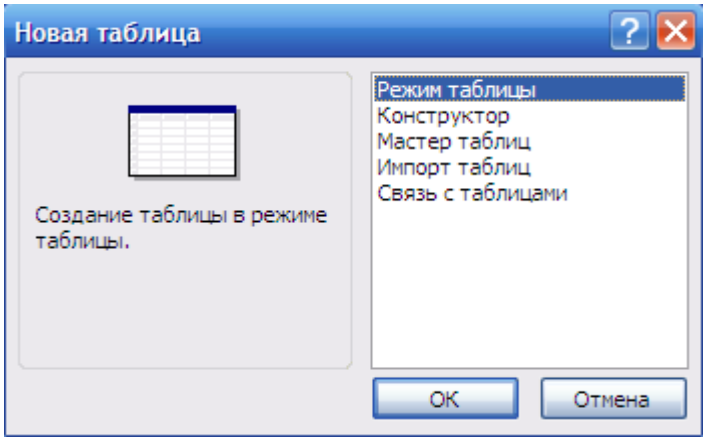



Рис. 5.4.4 Засоби для створення таблиць

***Конструктор*** - створення таблиці за допомогою конструктора таблиць.

***Майстер таблиць*** - створення таблиці за допомогою майстра таблиць на основі колекції таблиць і полів.

***Імпорт таблиць*** - створення таблиці шляхом імпорту даних із зовнішнього файлу або з іншої БД.

***Зв'язок з таблицями*** - приєднання зовнішнього файлу або таблиці іншої БД.

Розглянемо докладніше кожен засіб створення таблиць.

### 5.4.2.1 Режим таблиць

У цьому режимі дані представлені у вигляді таблиці. У цьому режимі можна вводити або редагувати дані, а також перевизначати саму таблицю. Кожен рядок відповідає запису, а кожен стовпець - полю. Останній рядок таблиці відмічений * позначає місце для нового запису.

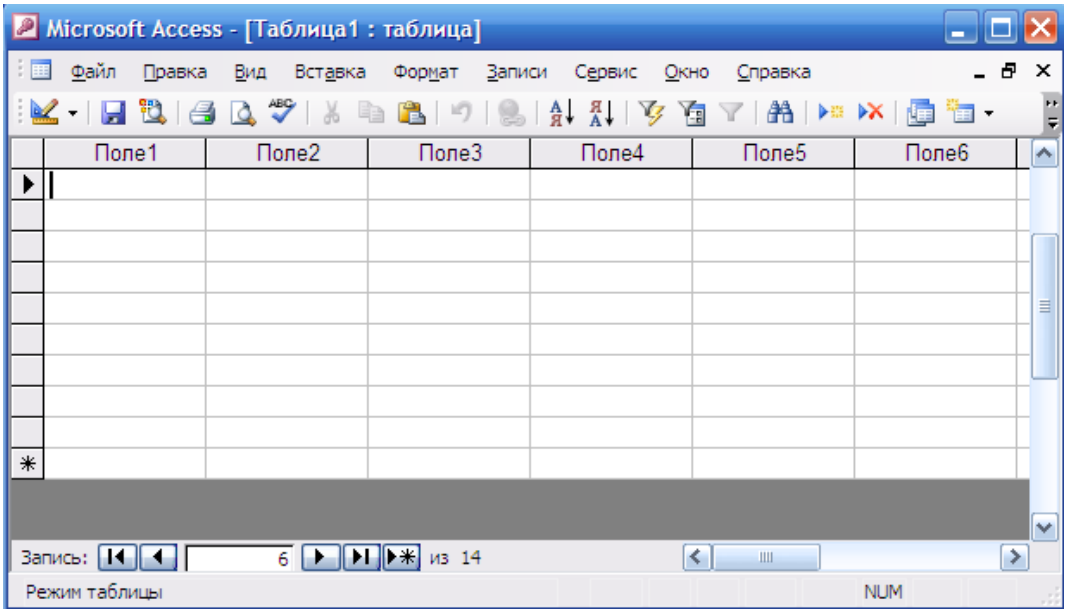



Рис. 5.4.5 Нова таблиця в режимі створення таблиць

Нижній рядок вікна режиму таблиць призначений для швидкого переміщення по записах. Елемент  відображає номер поточного запису. Щоб перейти до заданого

запису, необхідно ввести потрібний номер запису в це поле. Елементи  та  дозволяють пересуватися по записах, крайні - пересування в самий початок і в самий кінець соотв., внутрішні - на один запис назад або вперед. Елемент  відповідає за переміщення в полі введення нового запису.

Управління структурою і видом таблиці:

— Щоб вставити новий запис в середині таблиці, необхідно клацнути на заголовку того рядка, в який ви хочете її ввести і вибрати ***Новий запис***.

— Для переміщення стовпця мишею, необхідно клацнути на його назві і не відпускаючи кнопку миші і перетягнути стовпець на нове місце.

— Щоб змінити ім'я стовпця, для цього необхідно двічі натиснути на заголовок стовпця і ввести нове ім'я.

— Щоб змінити розмір стовпця, необхідно підвести покажчик миші до правої межі його заголовка і перетягнути межу до потрібної ширини. Розмір рядка змінюється для всіх рядків одночасно.

— Щоб деякі стовпці завжди залишалися видимими при горизонтальній прокрутці вікна, необхідно встановити покажчик на заголовку стовпців, виділити їх, клацнути правою кнопкою миші і вибрати ***Закріпити стовпці***. Закріплені стовпці переміщаються до лівого краю вікна.

— Можна ськрить/отобразіть стовпці для цього натиснути праву клавішу миші і вибрати ***Приховати стовпці***. Для відображення необхідно встановити покажчик миші на вільну від стовпців і рядків частина вікна, натиснути праву клавішу миші і вибрати ***Відобразити стовпці*** і у вікні, що з'явилося, галочками позначить які стовпці ви хочете відобразити.

**5.4.2.2 Режим конструктора**

У режимі Конструктора ми можемо створити нову таблицю або відредагувати що існує (рис. 5.4.6).

Щоб створити нове поле, необхідно провести наступні дії:

- Ввести його назву в ім'я ***поля***.
- У стовпці ***Тип даних*** вибрати відповідний тип із списку, що розкривається.
- При необхідності додати опис поля в колонці ***Опис***.

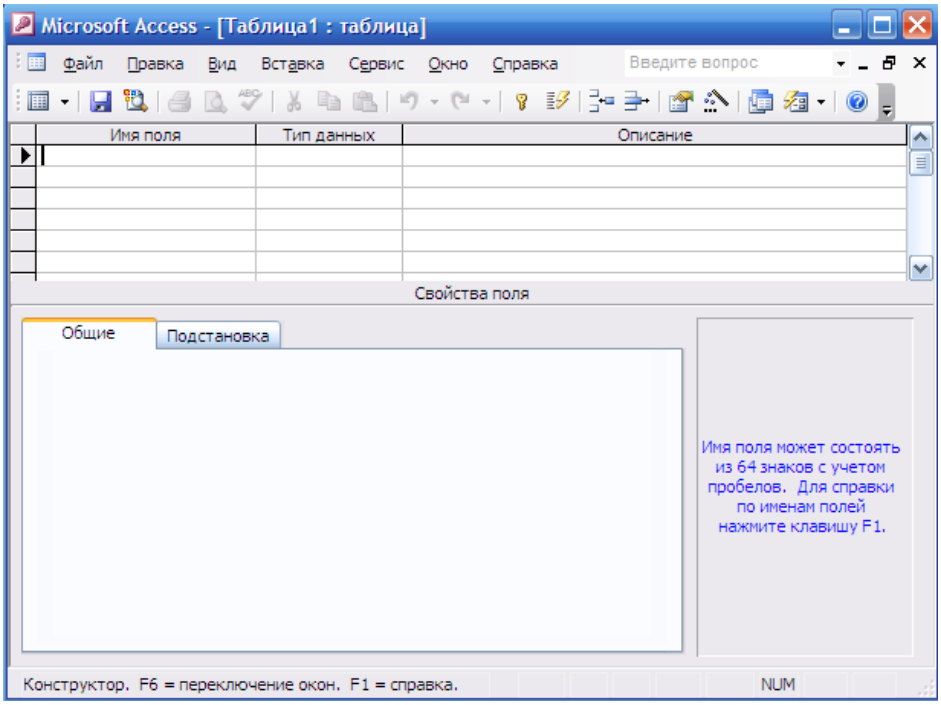



Рис. 5.4.6 Створення таблиць в режимі конструктора

Властивість ***Ім'я поля (Fieldname)*** визначає ім'я поля в таблиці. Введіть ім'я, що задовольняє угодам про імена об'єктів Microsoft Access. Це ім'я не повинне збігатися з ім'ям іншого поля в цій таблиці.

Угоди про імена - це набір правив, що обмежують допустимі імена об'єктів Microsoft Access. Імена об'єктів повинні містити не більше 64 символів і можуть включати будь-які комбінації букв, цифр пропусків і спеціальних символів за винятком крапки (.), знаку (!) оклику, надрядкового символу (`) і прямих дужок ([ ]). Відзначимо, що ім'я не повинне починатися з пропуску і містити символи, що управляють (з кодами ASCII 00 - 31).

Прагніть не включати в імена об'єктів пропуски, особливо, якщо передбачається часто використовувати посилання на ці імена у виразах. Уникайте дуже довгих імен: такі імена важко запам'ятовувати і на них незручно посилатися.

Властивість ***Тип даних (Datatype)*** визначає тип даних, що зберігаються в полі таблиці. У кожне поле допускається введення даних тільки одного типу.

Існують наступні типи даних:

| Тип | Вміст поля | Розмір |
|---|---|---|
| Текстовый | (Значення за умовчанням). Текст або числа, що не вимагають проведення розрахунків, наприклад, номера телефонів | Число символів, мінімальне, що не перевищує, з двох значень: 255 або значення властивості Розмір поля |
| Поле МЕМО | Довгий текст або комбінація тексту і чисел | До 65535 символів |
| Числовий | Числові дані, використовувані для проведення розрахунків | 1, 2, 4 або 8 байт |
| Дата/час | Дати і час, що відносяться до років з 100 по 9999, включно | 8 байт |
| Грошовий | Грошові значення і числові дані, використовувані в математичних розрахунках, що проводяться з точністю до 15 знаків в цілій | 8 байт |

| | | |
|---|---|---|
| | і до 4 знаків в дробовій частині | |
| Счетчик | Унікальні послідовно зростаючі (на 1) або випадкові числа, що автоматично вводяться при додаванні кожному новому запису в таблицю. Значення полів типу счетчик оновлювати не можна | 4 байт |
| Логічний | Логічні значення, а також поля, які можуть містити одне з двох можливих значень (True/False, Да/Ні) | 1 бит |
| Поле об'єкту OLE | Об'єкт (наприклад, електронна таблиця Microsoft Excel, документ Microsoft Word, малюнок, звукозапис або інші дані в двійковому форматі), зв'язаний або упроваджений в таблицю Microsoft Access | До 1 Гбайт (обмежується об'ємом диска) |
| Гіперпосилання | Рядок, що складається з букв і цифр, і що представляє адресу гіперпосилання | до 2048 символів |
| Майстер підстановок | Створює поле, в якому пропонується вибір значень із списку, або з поля із списком, що містить набір постійних значень або значень з іншої таблиці. Вибір цього параметра в списку в осередку запускає майстра підстановок, який визначає тип поля | 4 байт |

Зміна типу поля після введення даних в таблицю викличе те, що займає достатньо довгий час перетворення даних при збереженні таблиці. Несумісність існуючих даних з новим значенням властивості Тип даних (Datatype) може привести до втрати даних або видачі помилки.

Властивість ***Опис (Description)*** визначає текст, що містить опис окремих полів таблиці. Максимальна довжина опису складає 255 символів.

Область ***Властивість поля*** (див. рис. 5.4.6) містить список властивостей, доступних для типу даних вибраного в даний момент поля. Закладка Загальні цій області містить наступні властивості:

- ***Розмір поля*** - визначає максимальний розмір даних, які можуть зберігатися в полях з типом даних Текстовий, Числовий або Счетчик.
- ***Формат поля*** - дозволяє вказати формати виведення тексту, чисел, дат і значень часу на екран і на друк.
- ***Маска введення*** - задає маску введення, що полегшує введення даних в полі. Наприклад, зручно створити наступну маску введення для поля «Телефон», що дозволяє вводити тільки цифри і що автоматично додає проміжні символи: (___) ___-____. Значення даної властивості визначається автоматично при використанні майстра по створенню масок введення.
- ***Підпис*** - визначає текст, який виводиться в підписах об'єктів в різних режимах. Якщо користувач не указує текст підпису поля таблиці, то як текст підпису елементу управління і заголовка стовпця в режимі таблиці використовується значення властивості Ім'я поля.
- ***Значення за умовчанням*** - дозволяє вказати значення, що автоматично вводиться в поле при створенні нового запису. Наприклад, в таблиці «Адреси» може виявитися зручним вказати автоматичне введення значення «Москва» в полі «Місто». При заповненні таблиці користувачі зможуть залишити в цьому полі стандартне значення або, при необхідності, вказати інше місто.

— ***Умова на значення*** - визначає вимоги до даних, що вводяться в запис, в полі або в елемент управління. Максимальна довжина значення складає 2048 символів. Для елементів управління може бути вказане будь-який правильний вираз. Вираз, вказаний як умова для поля, не повинен містити посилання на інші поля.

— ***Повідомлення про помилку*** - дозволяє вказати текст повідомлення, що виводиться на екран, якщо введені дані порушують умову, визначену у властивості Умова на значення. Максимальна довжина значення властивості Повідомлення про помилку складає 255 символів.

— ***Обов'язкове поле*** - указує, чи вимагає поле обов'язкового введення значення. Якщо ця властивість має значення «Так», то при введенні нового запису необхідно ввести значення в це поле.

У режимі Конструктора можна виконувати наступні дії:

— Щоб перемістити поле в списку, клацніть на заголовку цього поля, відпустите кнопку миші і перетягнете поле в нове місце.

— Щоб вставити або видалити поля (рядки), встановите покажчик на необхідному рядку, клацніть правою кнопкою миші, і виберіть відповідну команду контекстного меню.

— Щоб додати поле з наданих таблиць Access, встановите покажчик в тому місці списку, де ви хочете вставити нове поле, клацніть правій миші і виберіть команду контекстного меню ***Побудувати***. У вікні, що з'явилося, вибрати зразок таблиці, а потім зразок поля.

— Для позначення поля як ключового, необхідно виділити потрібний рядок або групу рядків і вибрати піктограму  або Правка□□ Ключове поле.

— Щоб ввести конкретні значення в таблицю, що вийшла, необхідно зберегти таблицю, а потім вибрати Вигляд□□ Режим таблиці або натиснути піктограму  .

### 5.4.2.3 Майстер таблиць

Дозволяє створити таблицю, використовуючи зразки таблиць, що поставляються разом з Access. Створення таблиці поділене на кроки. На кожному кроці майстра проводяться різні налаштування для майбутньої таблиці. Перехід до наступного кроку здійснюється натисненням кнопки ***Далі***.

На першому кроці необхідно вибрати зразок таблиці і полів. Зразки таблиць діляться на ділове застосування і особисте застосування. Вікно для вибору потрібного зразка і полів з цього зразка виглядає, як показано на рисуку 5.4.7. Для вибору потрібного поля використовуються кнопки:  - для вибору одного поля,  - для вибору всіх полів зразка таблиці,  - для видалення одного вибраного поля,  - для видалення всіх вибраних полів. На цьому ж кроці вибраним полям можна привласнити інші імена. Для цього необхідно вибрати поле і натиснути кнопку ***Перейменувати поле*** і ввести нове ім'я.

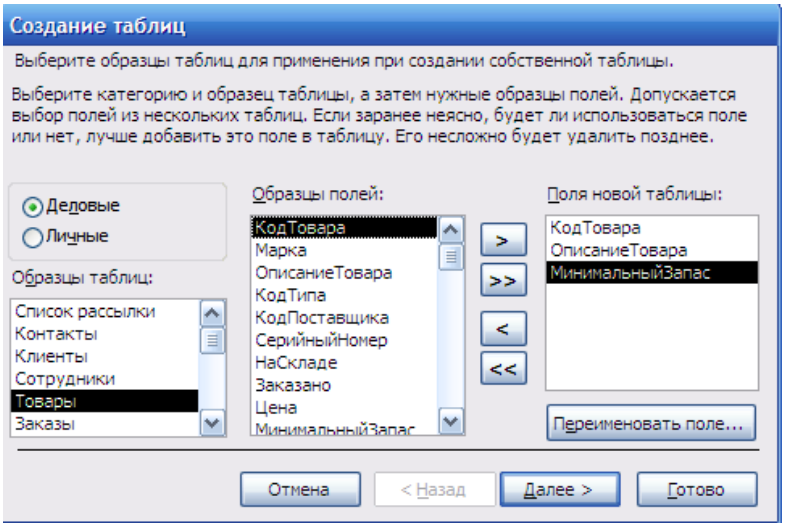



Рис. 5.4.7 Майстер таблиць. Вікно для вибору полів.

Після вибору полів для своєї таблиці, переходиться до наступного кроку, в якому можна ввести ім'я для нової таблиці і вибрати чи будете ви самостійно вибирати ключове поле або надасте цю операцію Access (див. рис. 5.4.8).

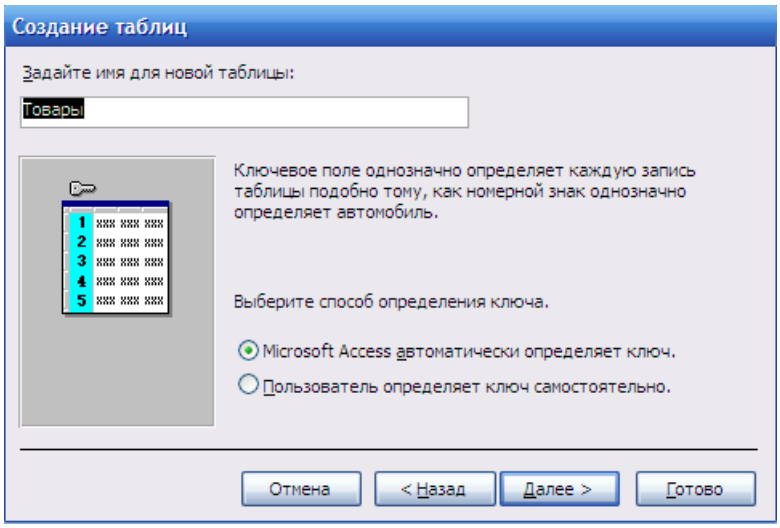



Рис. 5.4.8 Майстер таблиць. Вікно для вибору способу визначення ключа.

Якщо ви вибрали самостійне визначення ключового поля, то на наступному кроці, необхідно вибрати поле, яке буде ключовим, і які дані міститимуться в ключовому полі:

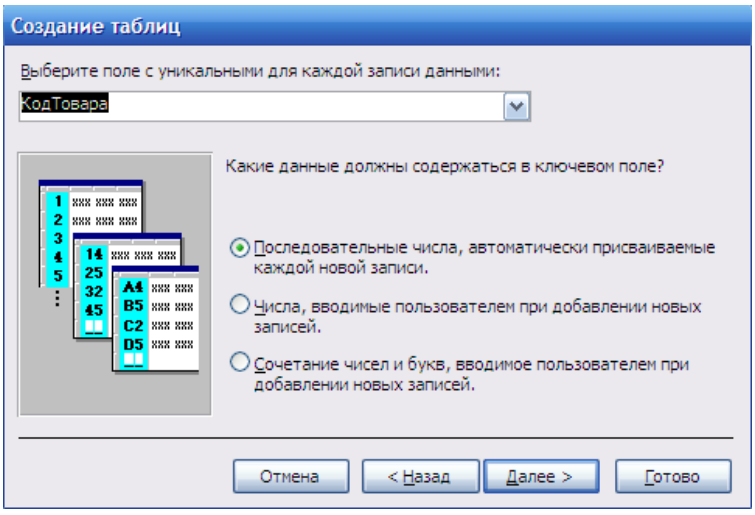


Рис. 5.4.9 Майстер таблиць. Вікно для вибору ключового поля

На завершальному кроці ви вибираєте подальші дії, які ви хочете виконати після створення таблиці ( див. рис. 5.4.10). До таких дій відносяться: зміна структури таблиці, тобто відкриття створеної таблиці в режимі конструктора; введення даних, тобто відкриття таблиці в режимі таблиці; введення даних за допомогою форми, тобто створення макету форми для створеної таблиці.

У завершенні роботи майстра необхідно натиснути кнопку ***Готово***.

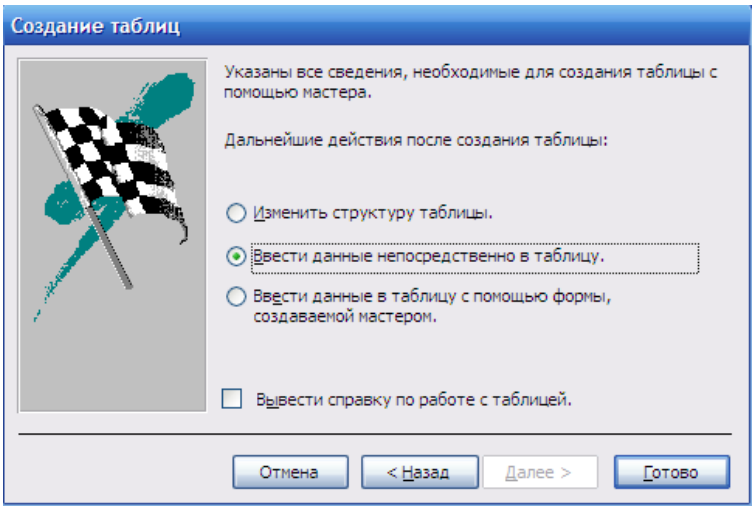



Рис.5.4.10 Майстер таблиць.
Вікно для вибору подальших дій після створення таблиць

### 5.4.3. Види представлення таблиці

Після створення таблиці ми ще не раз можемо звернутися до таблиці, щоб проглянути, обробити, змінити або проаналізувати дані. Залежно від конкретних цілей ми можемо представляти таблицю в тому або іншому вигляді, найбільш відповідному нам.

Для відкриття створеної таблиці необхідно виконати подвійне клацання на її назві. За умовчанням таблиця відкриється у вигляді ***Таблиця***. Всього існує 4 види представлення таблиць, перемикати їх можна або кнопкою панелі інструментів, або за допомогою вибору відповідних пунктів меню Вигляд*:*

- ***Таблиця*** - вигляд, найбільш оптимальний для представлення табличних даних, він відкриває природну структуру їх уявлення.
- ***Конструктор*** - вид представлення таблиці, що відображає вичерпну інформацію про поля таблиці і їх властивості, - так звані *Метадани таблиці*.
- ***Звідна таблиця*** - вид представлення великих об'ємів даних, що робить їх зручними для аналізу і обробки.
- ***Звідна діаграма*** - ще зручніший вид представлення великих об'ємів даних, представлення даних в цьому вигляді наочніше, але менш інформативно, але в той же час з використанням звідних діаграм аналізувати дані значно простіше.

### 5.4.4. Редагування проекту таблиці

Скористаємося можливістю внесення змін в структуру таблиці. Зміни вноситимемо в режимі конструктора. Хай в існуючу таблицю, наприклад Товар, необхідно додати нове поле Вид товару, яке матиме тільки два значення: штучний або ваговий. Для додавання такого поля необхідно вказати позицію для його вставки, зайти в меню Вставка ] Поле підстановок. Після вибору ***Поле підстановок*** активізується ***майстер підстановок,*** який візьме управління на себе.

На першому кроці майстра підстановок вимагає визначити вид джерела даних для підстановки. Наприклад, для нашого завдання, виберемо Фіксований набір значень (див. рис. 5.4.11)

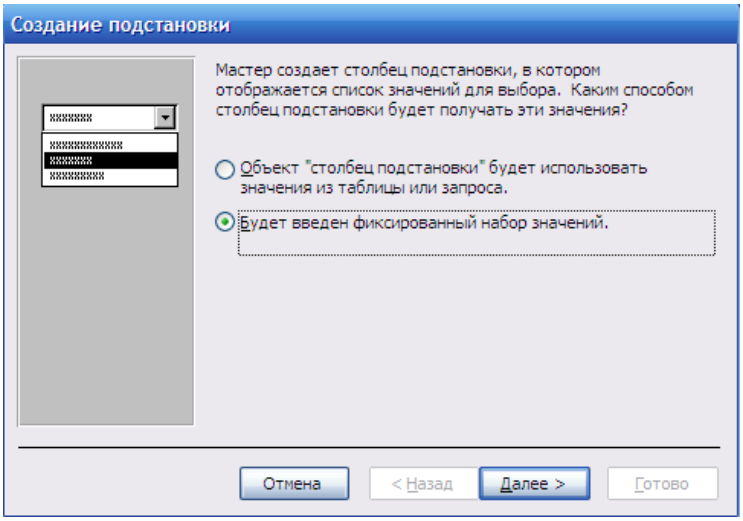



Рис.5.4.11 Перший крок майстра підстановок

На другому кроці вимагає визначити список значень для підстановки. Список значень вводиться користувачем в стовпець з ім'ям Столбец1 (див. рис. 5.4.12).

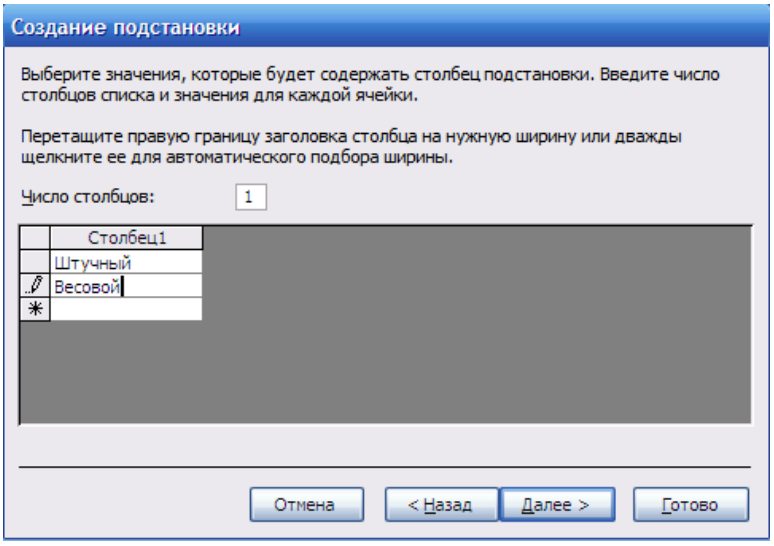


Рис. 5.4.12 Другий крок майстра підстановок

На третьому кроці необхідно ввести підпис для створюваного поля (див. рис. 5.4.13).

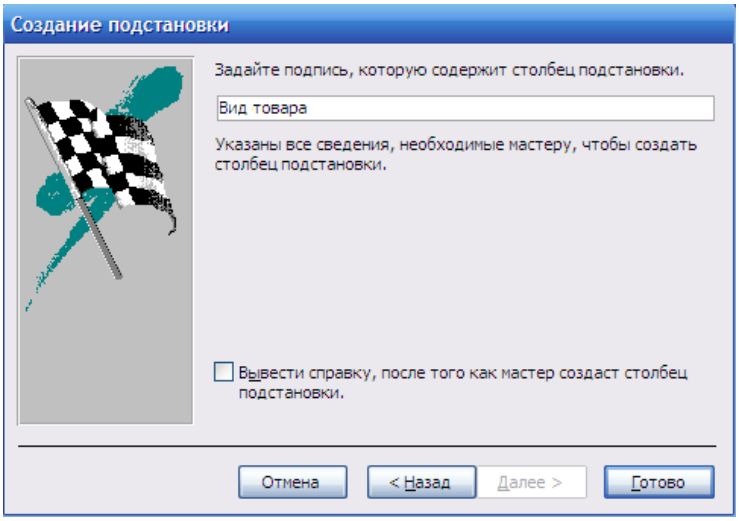



Рис. 5.4.13 Завершальний крок майстра підстановок

Налаштування підстановки здійснюється в розділі Властивості поля, закладка Підстановка.

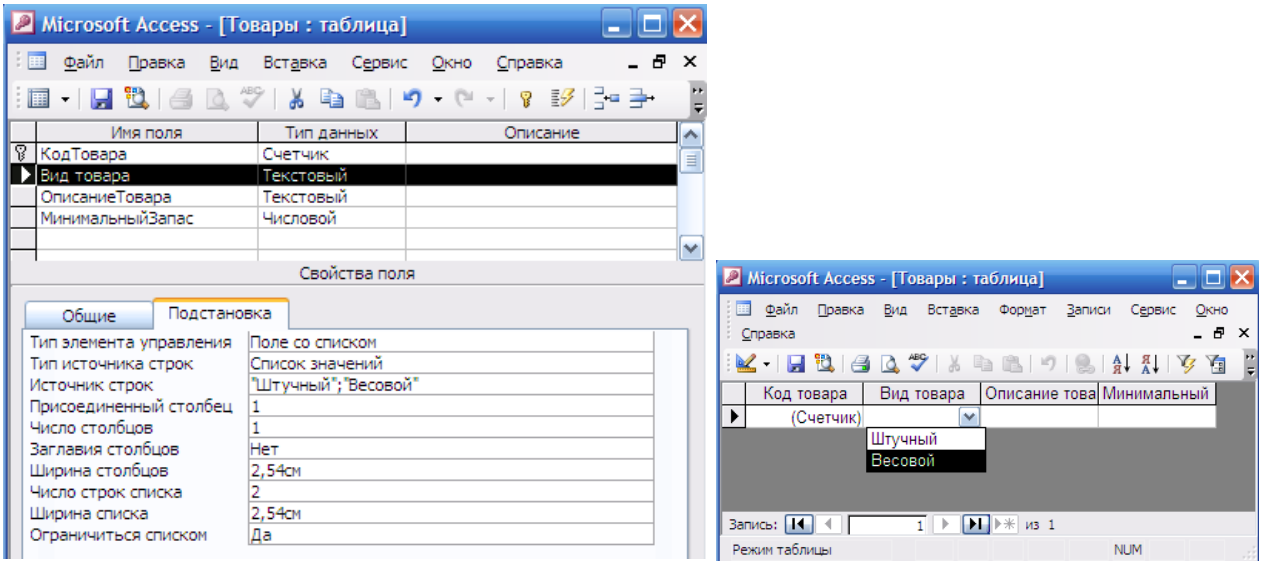



Рис. 5.4.14 Результат роботи майстра підстановок

Розглянемо налаштування підстановки (див. рис. 5.4.14):

— Тип елементу управління - Визначає вид управління, що виводиться для поля елементу (в даному випадку це *Поле із списком)*.

— Тип джерела рядків - Визначає тип джерела даних. Оскільки дані вводилися безпосередньо користувачем, то ця властивість має значення *Список значень*

— Джерело рядків - Визначає джерело даних для поля.

— Приєднаний стовпець - Номер стовпця, значення якого виводитиметься в полі.

— Число стовпців - Визначає кількість стовпців для показу.

— Заголовки стовпців - Визначає, чи слід виводити для поля із списком заголовок.

— Ширіна стовпців - Визначає ширину стовпців в списку.

— Число рядків списку - Визначає кількість елементів в списку.

— Ширіна списку - Визначає ширину списку, що виводиться.

— Обмежитися списком - Визначає вибір значень для даного поля: *Так* - значення можуть належати списку, що тільки виводиться

**Контрольні питання:**

1. Які 6 типів об'єктів бази даних Access ви знаєте?
2. Скільки способів ви знаєте для створення талиць?
3. Чим відрізняються ці способи створення таблиць?
4. Які типи даних використовуються в Access?
5. Що таке поле підстановок? Для чого використовується? Як його зробити?

### 5.5. Визначення зв'язків між таблицями

*Встановлення зв'язків між таблицями. Зміна зв'язків. Забезпечення цілісності даних.*

Після створення різних таблиць, що містять дані, бази даних, що відносяться до різних аспектів, розробник повинен продумати, яким чином Microsoft Access об'єднуватиме ці дані при їх витяганні з бази даних. Першим кроком при цьому є визначення зв'язків між таблицями.

Зв'язок між таблицями встановлює стосунки між співпадаючими значеннями в ключових полях, зазвичай між полями різних таблиць, що мають однакові імена. В більшості випадків з ключовим полем однієї таблиці, унікальним ідентифікатором кожного запису, що є, зв'язується зовнішній ключ іншої таблиці.

Для того, щоб визначити зв'язок між таблицями, слід вибрати Сервіс  Схема даних або клацнути на панелі інструментів на піктограму  . Якщо зв'язки ще не були встановлені, то з'явиться вікно в якому необхідно вибрати таблиці і/або запити між якими буде встановлений зв'язок (див. рис. 5.5.1).

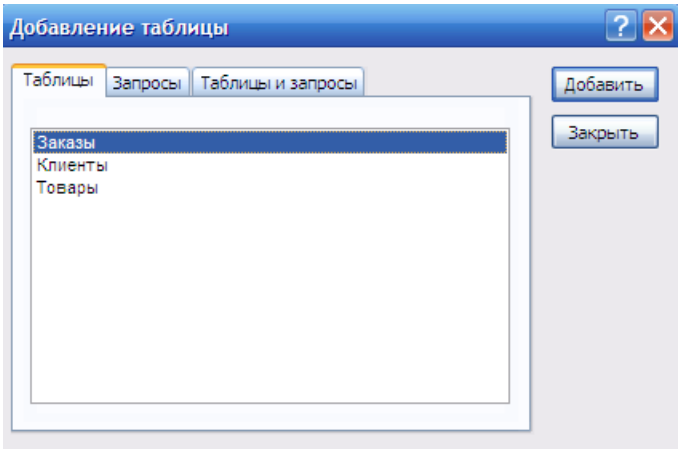



Рис. 5.5.1 Вікно для додавання таблиць для встановлення зв'язків

Також вікно для додавання таблиць можна викликати з панелі інструментів натиснувши на кнопку Відобразити таблицю ( )або вибравши меню Зв'язки → Додати таблицю.

Після додавання потрібних таблиць, Access автоматично виставить зв'язки, грунтуючись на однакових іменах полів (див. рис. 5.5.2).

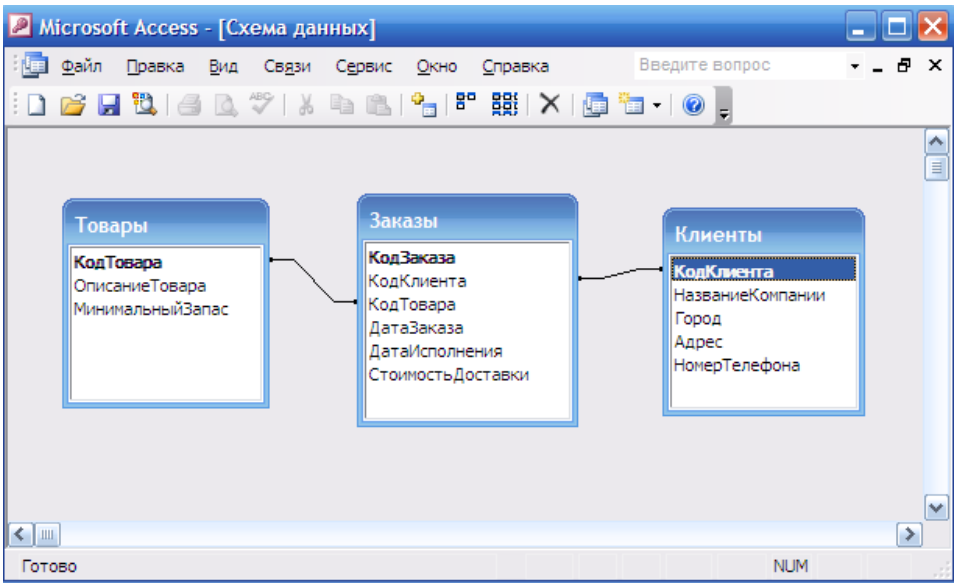



Рис. 5.5.2 Вікно Схеми даних.

Для ручного встановлення зв'язків необхідно за допомогою миші перенесіть поле, яке слід використовувати для установки зв'язку, із списку однієї таблиці до відповідного поля іншої таблиці. На екрані з'явиться діалогове вікно ***Зміна зв'язків*** (див. рис. 5.3).

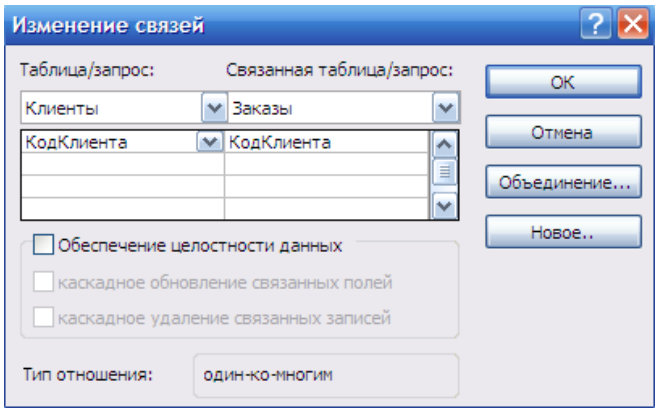



Рис. 5.5.3 Вікно для установки зв'язків між полями двох таблиць

Встановлення опції перевірки посилальної цілісності ***Забезпечення цілісності даних*** забезпечує перевірку посилальної цілісності зв'язку між обома таблицями. Ця перевірка дозволяє уникнути ряду помилок, що допускаються при видаленні записів з первинної таблиці і введенні інформації в зв'язану таблицю. Завдяки перевірці посилальної цілісності можна уникнути наступних помилкових ситуацій:

- додавання в зв'язану таблицю записів, для яких відсутній відповідний запис в первинній таблиці;
- здійснення змін в головній таблиці, які приведуть до появи «усиротілих» записів в зв'язаній таблиці;
- видалення записів в головній таблиці, на які посилаються записи із зв'язаної таблиці.

**Контрольні питання:**

1. Для чого встановлюють зв’язки між таблицями?
2. Які кроки необхідно зробити, щоб встановити зв'язок між талицями?
3. Для чого використовується опція Забезпечення цілісності даних?

### 5.6. Створення запитів

*Поняття запиту. Види запитів. Способи створення запитів. Запити в режимі конструктора. Запити на мові SQL. Приклади запитів.*

***Запити*** - засіб відбору і сортування даних -в Access є проміжною ланкою між, таблицями і формами. Запроси- це своєрідні віртуальні таблиці, які, володіючи всіма властивостями таблиць, насправді сховищем даних не є.

Запити призначені для крупних операцій по вибірці інформації, а також для додавання і видалення записів в існуючих таблицях і створення нові.

Існують наступні види запитів:

1. Запит на вибірку даних
2. Параметризований запит
3. Табличний запит
4. Запити-дії

***Запит на вибірку даних*** - основний вид запитів. Він вибирає дані, відповідні вказаному критерію, з однієї або декількох таблиць і поміщає їх в нову таблицю. У запит цього вигляду можна включити різні обчислення і підрахунок підсумкових значень. Значення результуючої таблиці можна редагувати і внесені зміни запам'ятовуються в полях початкових таблиць.

***Параметризований запит*** – це запит на вибірку, критерій якого визначається під час запуску. Параметри критерію вводяться в спеціальному діалоговому вікні, що з'являється при запуску запиту, що параметризується.

Підрахунок середнього, суми або кількості значень, в одному полі, згрупованому по іншому, здійснюється за допомогою ***табличного запиту***.

***Запити-дії*** призначені для одночасної зміни декількох записів. Вони використовуються в тих випадках, коли, наприклад, необхідно підняти ціни по всій таблиці або додати префікс до всіх номерів частин продуктів певної лінії.

Запити можна створювати за допомогою двох режимів:

— режим конструктора мова QBE (Query By Example);

— режим SQL.

Таблична мова запитів QBE (скорочення від Query-by-example або Запитів за зразком), разом з мовою SQL, використовується для створення різних запитів до реляційних БД. Мова QBE є наочнішою і простішою для розуміння в порівнянні з SQL, хоча і більш обмеженою в можливостях.

Для створення нового запиту необхідно натиснути кнопку Створити. В результаті з'явиться вікно з інструментами для створення запитів.

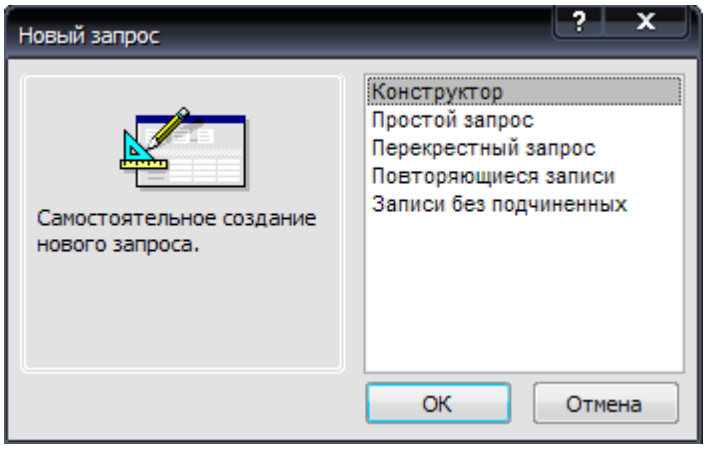



Рис. 5.6.1 Вікно вибору способу створення запиту

Access надає наступні можливості для створення нового запиту:

1. ***Конструктор*** - створення запиту в режимі конструктора.
2. ***Простий запит*** - створення запиту на вибірку з певних полів.
3. ***Перехресний запит*** - створення запиту, що виводить дані в компактному форматі, подібному формату електронної таблиці.
4. ***Записи, що повторюються,*** - створення запитів на пошук записів, що повторюються, в простій таблиці або запиті.
5. ***Записи без підлеглих*** - створення запитів на пошук записів, яким не відповідає жоден запис в підлеглій таблиці.

### 5.6.1 Створення запитів в режимі конструктора

При виборі режиму конструктора або при натисненні кнопки Конструктор з'являється вікно Додавання таблиці. У нім необхідно вибрати таблицю або декілька таблиць, які будуть вам необхідні для побудови нового запиту. Їх додавання відбувається після натиснення кнопки додати. Після додавання потрібних таблиць закрийте це вікно.

Режим конструювання запитів має вид наступного вікна: (див. рис. 5.6.2)

Вікно запиту розбите горизонтально посередині. У верхній половині відображаються списки полів всіх вибраних таблиць, що беруть участь в запиті, а в ніжней - власне специфікація запиту. Таблиці додаються в запит за допомогою кнопки панелі інструментів (Додати таблицю) або за допомогою контекстного меню, викликаного для верхньої половини вікна. Запуск запиту на виконання здійснюється натисненням на кнопку панелі інструментів (Запуск).

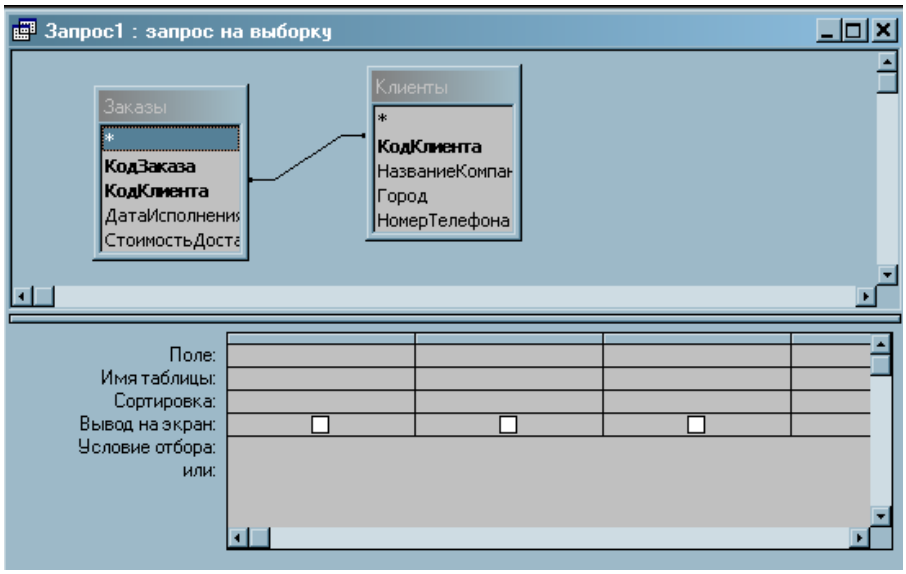



Рис. 5.6.2 Вікно створення запитів в режимі Конструктора

У нижній частині вікна розташована сітка побудови запиту. Кожен стовпець відповідає полю, дані з якого включаються в запит. Сітка складається з наступних рядків:

— ***Поле*** - щоб вставити поле якоїсь таблиці в запит, його потрібно вибрати із списку полів випадного в цьому рядку. Зірочка * у списку полів позначає включення всіх полів з відповідної таблиці.

— ***Ім'я таблиці*** - указується ім'я таблиці, поле якого ми вибрали раніше.

— ***Групова операція*** - використовується для підрахунку ряду обчислень. Якщо такий рядок відсутній в сітці, то вона додається при натисненні кнопки панелі інструментів Σ.

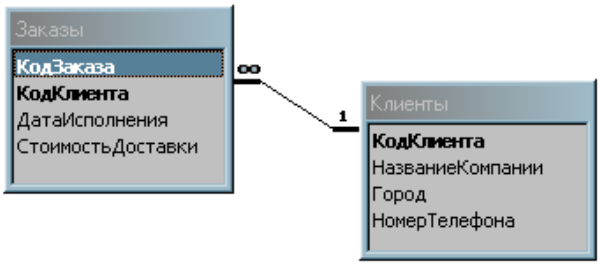
— ***Сортування*** - в цьому рядку визначається по яких полях будуть відсортовані результати виконання запиту. При сортуванні по декількох полях Access сортує дані в порядку появи полів в сітці побудови запиту зліва направо. Положення стовпця поля можна змінити, виділивши його клацанням миші на заголовку і перетягнувши його на нове місце.

— ***Вивід на екран*** -устанавлівається, якщо дані з поля відповідного стовпця мають бути включені в результат запиту.

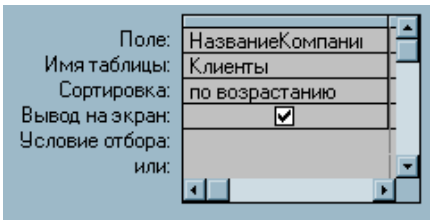
— ***Умова відбору*** - в цьому рядку вводиться критерій даних.

### 5.6.2 Приклади запитів

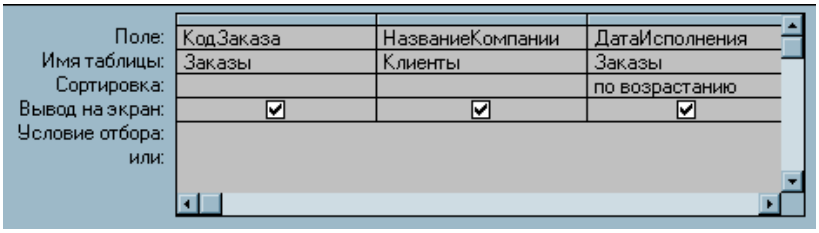
Для розгляду прикладів створення запитів в режимі конструктора, створимо 2 таблиці: Замовлення і Клієнти.



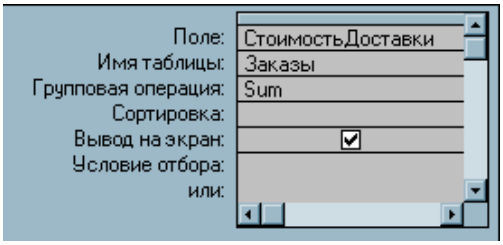

Необхідно відсортувати назви клієнтів за збільшенням. Тоді конструктор запитів матиме наступний вигляд



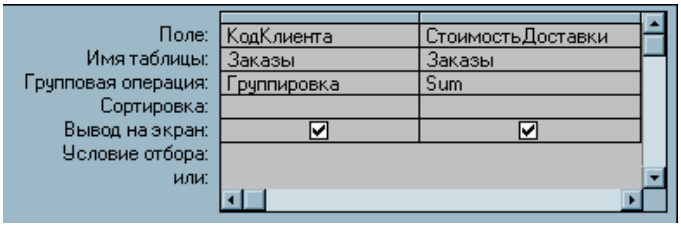

Необхідно вивести поля Кодзаказа і Датаїсполненія з таблиці Замовлення і поле Названієкомпанії з таблиці Клієнти і відсортувати Датуїсполненія за збільшенням. Тоді запит матиме наступний вигляд:



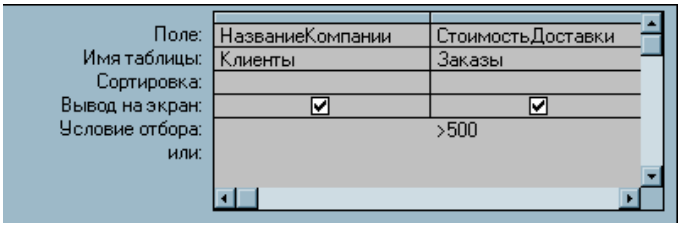

Необхідно підрахувати суму доставок всіх замовлень. Тоді запит має вигляд:



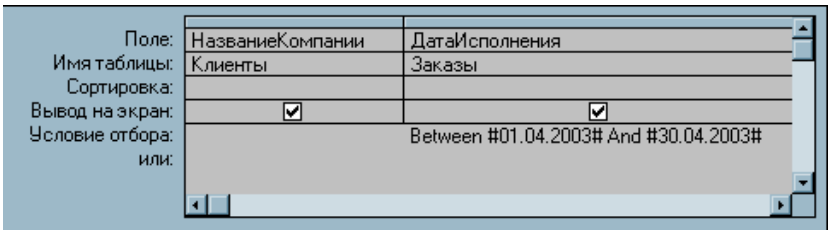

Необхідно підрахувати суму доставки по кожному клієнтові. Тоді запит має вигляд:



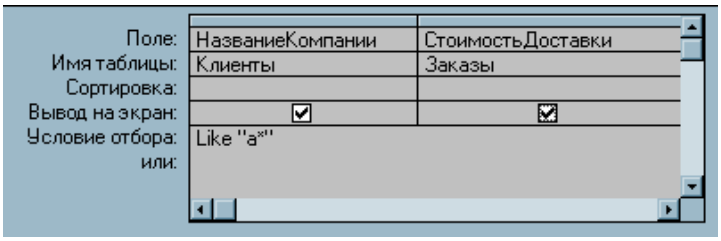

Необхідно вивести Названієкомпанії і Суммудоставки, якщо Суммудоставки > 500. Тоді запит має вигляд:



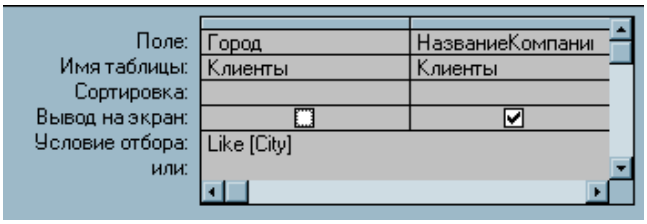

Необхідно вивести Назваєкомпанії і Датуїсполненія, якщо Датуїсполненія знаходиться між 01.04.2003 і 30.04.2003. Тоді запит має вигляд:




Необхідно вивести Назваєкомпанії і Суммудоставки, якщо Назваєкомпанії починається з букви «а». Тоді запит має вигляд:




Вивести Назваєкомпанії з деякого міста. Ім'я міста задавати як параметр. Тоді запит має вигляд:



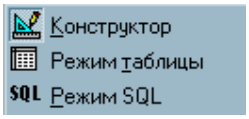

### 5.6.3 Створення запитів в режимі SQL

Мова SQL (Structured Query Language) використовується при створенні запитів, а також для оновлення і управління реляційними базами даних, такими як бази дані Microsoft Access. Коли користувач створює запит в режимі конструктора запиту, Microsoft Access автоматично створює еквівалентну інструкцію SQL. Користувач має можливість переглядати і змінювати інструкції SQL в режимі SQL. Зміни, внесені до запиту в режимі SQL, приведуть до відповідних змін в бланку запиту в режимі конструктора. Деякі запити не можуть бути визначені в бланку запиту конструктора. Для створення таких запитів потрібно ввести інструкцію SQL безпосередньо у вікно запиту в режимі SQL.

Для перегляду і зміни інструкції SQL необхідно виконати наступні кроки:

— створити або відкрити існуючий запит.

— Натисніть на панелі інструментів кнопку  (Вигляд) і виберіть Режим SQL 




### 5.6.4 Інструкції SQL

1. **Виведення даних з однієї таблиці**
   **SELECT <имя поля1>,…, <имя поляN>**
   **FROM <имя таблицы>;**

Наприклад:

— Вивести всі поля таблиці Клієнти

```
SELECT *
FROM Клиенты;
```

— Вивести назву компаній і місто знаходження цих компаній

```
SELECT НазваниеКомпании, Город
FROM Клиенты;
```

2. **Виведення даних з однієї таблиці по деякій умові**

**SELECT <имя поля1>,…, <имя поляN>**
**FROM <имя таблицы>**
**WHERE <имя поля> Условие Значение;**

Наприклад:

— Вивести назву компанії з міста ABC

```
SELECT НазваниеКомпании
FROM Клиенты
WHERE Город = 'ABC';
```

— Вивести код замовлення, код клієнта, вартість доставки, якщо вартість доставки більше 400

```
SELECT КодЗаказа, КодКлиента, СтоимостьДоставки
FROM Заказы
WHERE СтоимостьДоставки>400;
```

— Вивести назву компаній назва "Київ";

```
SELECT НазваниеКомпании
FROM Клиенты
WHERE НазваниеКомпании<"Киев";
```

— Вивести код замовлення і дата виконання замовлення, якщо дата виконання знаходиться між 01.03.2003 і 15.05.2003.

```
SELECT КодЗаказа, ДатаИсполнения
FROM Заказы
WHERE ДатаИсполнения BETWEEN #03/01/03# AND #05/15/03#;
```

— Вивести назву компанії, місто якої починається з букви В

```
SELECT НазваниеКомпании
FROM Клиенты
WHERE Город Like "В*";
```

— Вивести назву компанії, місто якої задається у вигляді параметра

```
SELECT НазваниеКомпании
FROM Клиенты
WHERE Город Like [City];
```

— Вивести код замовлення і вартість доставки, якщо вартість доставки більше параметра, що задається

```
SELECT КодЗаказа, СтоимостьДоставки
FROM Заказы
WHERE СтоимостьДоставки > [Fee];
```

**3. Об'єднання записів з однаковими значеннями**

**SELECT <имя поля1>,…, <имя поляN>**
**FROM <имя таблицы>**
**WHERE <имя поля> Условие Значение**
**GROUP BY <имя группируемого поля>;**

Наприклад:

— Вивести суму доставки по кожному клієнтові

```
SELECT Sum (СтоимостьДоставки) AS Sum
FROM Заказы
GROUP BY КодКлиента;
```

— Вивести код клієнта і кількість замовлень по кожному клієнтові

```
SELECT КодКлиента, Count(КодКлиента) AS [Num of zakaz]
FROM Заказы
GROUP BY КодКлиента;
```

**4. Накладення умов на згруповані записи**

**SELECT <имя поля1>,…, <имя поляN>**
**FROM <имя таблицы>**
**WHERE <имя поля> Условие Значение**

**GROUP BY <имя группируемого поля>**
**HAVING <имя поля> Условие Значение;**

Наприклад:

— Вивести код клієнта і суму доставки по кожному клієнтові, якщо сума доставки перевищує 100.

```
SELECT КодКлиента, Sum(СтоимостьДоставки)
FROM Заказы
GROUP BY КодКлиента
HAVING Sum(СтоимостьДоставки) > 100;
```

**5. Сортування в порядку зростання або убування**

**SELECT <имя поля1>,…, <имя поляN>**
**FROM <имя таблицы>**
**WHERE <имя поля> Условие Значение**
**ORDER BY <имя поля1> [ASC | DESC ][, <имя поля2> [ASC | DESC ];**

За умовчанням здійснюється сортування за збільшенням. **ASC** - за збільшенням, **DESC** - по убуванню.

Наприклад:

— Вивести на екран код замовлення і вартість доставки по всіх замовленнях і відсортувати за збільшенням вартість доставки

```
SELECT КодКлиента, СтоимостьДоставки
FROM Заказы
ORDER BY СтоимостьДоставки;
```

Або

```
SELECT КодКлиента, СтоимостьДоставки
FROM Заказы
ORDER BY СтоимостьДоставки ASC;
```

**Контрольні питання:**

1. Для чого використовуються запити?
2. Які типи запитів ви знаєте?
3. Як створити запит в Access?
4. Які режими створення запитів ви знаєте?
5. Що таке SQL?

## 5.7. ACCESS. СТВОРЕННЯ ФОРМ

*Способи створення форм. Мастре форм. Створення форм в режимі конструктора*.

Розробник, працюючи над створенням бази даних, ретельно продумує структуру всіх таблиць і запитів, які в неї входитимуть. Але подальшим інформаційним заповненням цієї бази, він вже займатися не буде. Це робитимуть спеціальні користувачі БД. Дуже часто вони можуть виявитися людьми з кваліфікацією недостатньою для роботи з базою даних в

чистому вигляді. Ось саме для цих людей і для максимального спрощення їх праці - створюються ***форми***.

Сама форма є електронним бланком, в якому є ряд пойменованих (підписаних) полів для введення інформації. Користувач заповнює ці поля інформацією, і вона автоматично заноситься у відповідні таблиці бази даних.

Після натиснення на кнопку Створити, розміщену на закладці Форми, головного вікна бази даних з'явиться вікно з вибором способів створення форм.

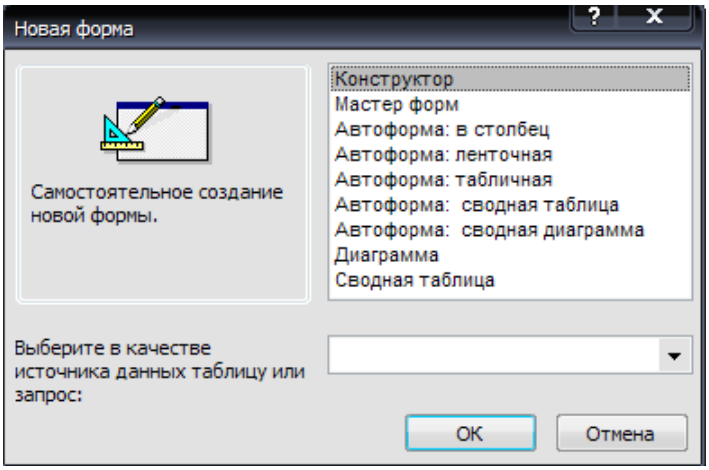



Рис. 5.7.1 Вікно створення форм

Існують наступні способи створення форм:

— ***Конструктор*** - самостійне створення конструктора, тобто уручну.

— ***Майстер форм*** - автоматичне створення форми на основі вибраних полів.

— ***Автоформа в стовпець*** - автоматичне створення форм з полями, розташованими в один стовпець.

— ***Автоформа стрічкова*** - автоматичне створення стрічкових форм.

— ***Автоформа таблична*** - автоматичне створення табличних форм.

— ***Діаграма*** - створення форми з діаграмою.

— ***Звідна таблиця*** - створення форми із звідною таблицею Excel.

### 5.7.1 Майстер форм.

Вікно Нова форма, окрім вибору режиму створення форми, просить вказати джерело даних для форми. Це може бути таблиця або запит. Список вже наявних об'єктів даних типів знаходиться в нижній частині вікна. Якщо зараз не вказати джерело даних і натиснути ОК, то майстер не почне лаятися, а просто почне свою роботу. На наступному етапі його роботи буде можливість знову вибрати джерело даних.

Етапи роботи Майстра:

Шаг 1. полягає у виборі полів таблиць, дані, для заповнення яких можна буде вводити в створюваній формі. Виконується це виділенням відповідного пункту в списку Доступні поля і натисненням на одну з двох кнопок переміщення (одинарний значок переносить виділене поле, а подвійний - відразу всі доступні поля). При цьому назва поля переходить в список Вибрані поля і це означає, що воно буде доступне для заповнення на створюваній формі. Якщо помилково було додано непотрібне поле, то його можна перемістити назад, виконавши аналогічні маніпуляції в списку вибраних полів і скориставшись кнопками із значками направленими в протилежному напрямі.

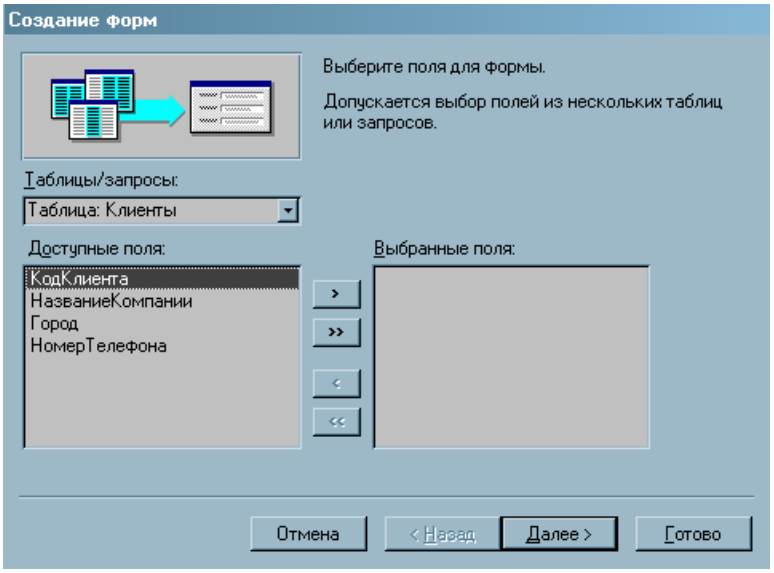
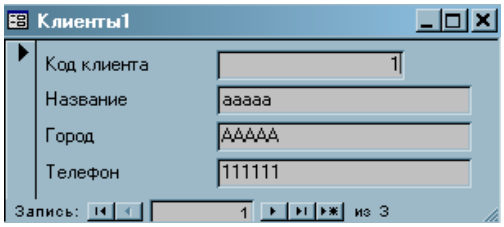


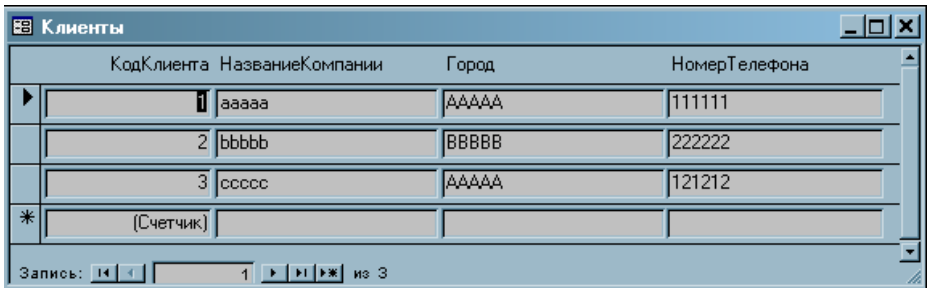

Рис. 5.7.2 Вікно першого кроку Майстра форм

Шаг 2. дозволяє вибрати один з чотирьох можливих видів представлення форми:

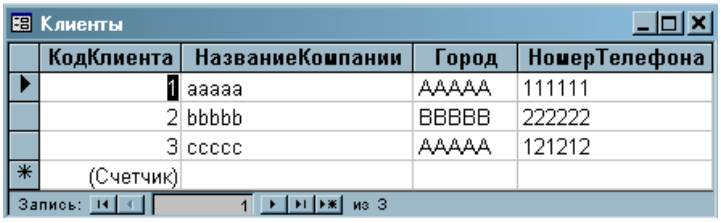
1. ***У один стовпець*** - розміщує поля для введення даних у форму, одне під іншим, утворюючи стовпець. Таким чином, завжди видно тільки одна поточний запис. Перемикання між сусідніми записами проводиться за допомогою кнопок переходу в нижній частині екрану.



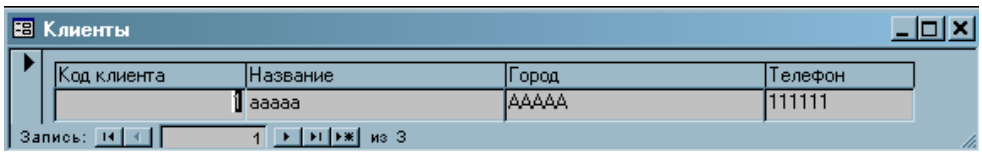

2. ***Стрічковий*** - у формі одночасний відображається ціла група сусідніх записів. Це більш необхідно для роботи із записами зв'язаними між собою. Наприклад, коли для введення наступного значення необхідно знати ряд попередніх.




3. ***Табличний*** - практично не відрізняється на вигляд від звичайної таблиці. Єдина відмінність полягає в тому, що для заповнення відриті тільки вибрані при створенні форми поля. Всі останні залишаються недоступними з даної форми.




4. ***Вирівняний*** - створює форму, поля на якій займають практично все вільне місце і при цьому вирівняні по відношенню один до одного. Іноді корисна для створення форм, що містять велику кількість полів, які необхідно умістити на екрані, без виконання скролінгу (прокрутки).




Шаг 3. На цьому кроці буде запропоновано вибрати стиль оформлення форми. Це фоновий малюнок або набір певних квітів елементів. При виборі будь-якого з можливих варіантів у вікні попереднього перегляду відразу буде створений приклад для наочного відображення стилю.

Шаг 4. Це останній крок, на якому майстер запитає ім'я форми, під яким вона буде збережена в базі, і запропонує два варіанти подальших дій

— Відкриття форми - приведе до запуску форми, після чого з її допомогою можна почати процес додавання нових записів в базу і редагування тих, що вже є.

— Зміна макету - відкриє створену форму в режимі конструктора. Що дасть можливість змінити і відредагувати структуру всіх полів форми.

### 5.7.2 Режим Конструктора

Структура форми в режимі конструктора розділена на три окремі розділи: заголовок форми, область даних і примітка форми.

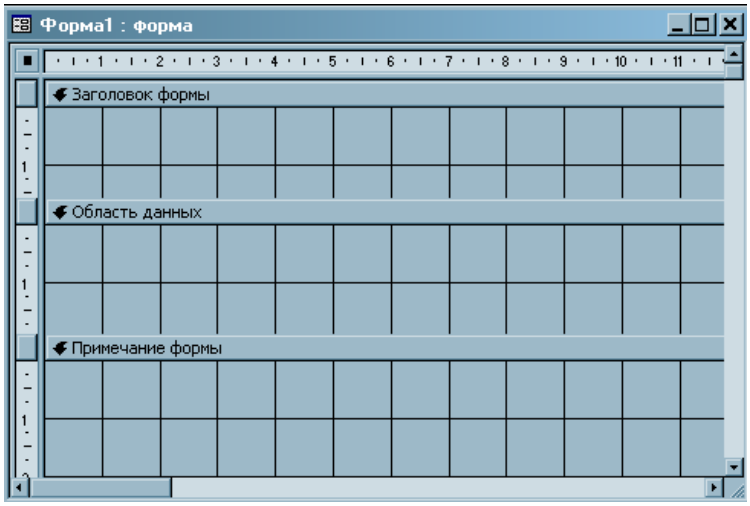



Рис. 5.7.3 Структура форми в режимі Конструктора

***Заголовок форми*** - може містити елементи, необхідні для зовнішнього оформлення або просто для прикраси форми. Вони будуть присутні на формі завжди, незалежно від того, створюється новий запис або редагується та, що вже існує. Це може бути напис, малюнок, логотип фірми або інший графічний елемент.

***Область даних*** - це основний розділ будь-якої форми. Без нього немає форми, оскільки вона містить елементи управління. Тут здійснюються всі динамічні процеси, що відбуваються при роботі форми. У нашому випадку там будуть тільки два типи елементів управління - зв'язане поле і приєднаний напис.

У ***зв'язане поле*** здійснюється введення даних, які синхронно поступають в однойменне поле відповідної таблиці, на основі якої створена форма. Поле форми і поле таблиці зв'язані між собою.

***Приєднаний напис*** це статичний елемент, що містить текстове пояснення. Вона переміщається по полю форми тільки разом зі своїм елементом управління (приєднана до нього).

***Примітка форми*** - схоже по своєму призначенню на заголовок, але знаходиться в нижній частині форми. Дуже часто містить інструкцію по заповненню форми або пам'ятку з вказівкою необхідній довідковій інформації.

Фоновий малюнок, що знаходиться під елементами управління і розграфлений сіткою, показує реальні розміри робочої області форми. Ті, в яких вона виводиться на екран. При наведенні на межі областей, покажчик міняє свою форму, і будь-яка з меж у цей момент може бути переміщена методом перетягання. Розташування елементів теж можна змінити, провівши клацання в полі елементу і не відпускаючи кнопку миші перетянув його на нове місце. Приєднаний напис при цьому теж автоматично переміщатиметься.

Також за допомогою контекстного меню, викликаного до області форми, можуть бути додані верхній і нижній колонтитули.

Існують наступні елементи управління:

1.  – вибір об'єктів.
2.  – майстер, засіб для ставлення питань і на основі відповідей будує відповідний об'єкт.
3.  – напис, призначена для виведення описового тексту.
4.  – поле, використовується для відображення, введення, зміни даних в джерелі записів форми для виведення результатів обчислень, а також для прийому даних, що вводяться користувачем.
5.  – група перемикачів, використовується для розміщення набору прапорців, перемикачів або вимикачів, що представляє набір альтернативних значень.
6.  – вимикач, використовується як окремий елемент управління, що приймає дії користувача в спеціальному діалоговому вікні.
7.  – перемикач, використовується як окремий елемент управління, що приймає дії користувача в спеціальному діалоговому вікні.
8.  – прапорець, використовується як окремий елемент управління, що приймає дії користувача в спеціальному діалоговому вікні.
9.  – поле із списком, складовий елемент управління, об'єднуючий поле і список, що розкривається. У таке поле можна ввести значення або вибрати його із списку.

10.  – список, створює список, що допускає прокрутку. Вибране в списку значення можна ввести в новий запис або використовувати для зміни в існуючому записі.

11.  – кнопка, використовується для виконання дій.

12.  – малюнок, використовується для відображення незмінного малюнка.

13.  – вільна рамка об'єкту, використовується для відображення вільного об'єкту OLE.

14.  – приєднана рамка об'єкту, використовується для відображення об'єктів OLE, таких як набір малюнків. При переході на новий запис вводяться нові об'єкти.

15.  – розриви сторінки, використовується для початку нового екрану у формі.

16.  – набір вкладок, використовується для створення форми з декількома вкладками.

17.  – підлегла форма, використовується для виводу у формі даних з декількох таблиць.

18.  – лінія, використовується для відділення даних.

19.  – прямокутник, використовується для просторового угрупування споріднених даних.

20.  – додаткові елементи.

### 5.7.3 Заповнення Форми в режимі конструктора

Додавання напису в заголовку форми розглянемо на конкретному прикладі. Допустимо, нам необхідно створити форму для заповнення таблиця Клієнти(Кодклієнта, Названіекомпанії, Місто, Номертелефона).

У заголовок форми додамо напис «Додавання нового клієнта».

Для цього необхідно:

1. Збільшити область заголовка. Для цього перетягнете вниз межу розділу заголовка і області даних. Виділите достатньо місця для створення необхідного напису.

2. Знайдіть на панелі елементів об'єкт Напис, призначений для створення текстових написів  (якщо панель елементів відсутня, то її можна включити через Вигляд - Панель елементів, головного меню програми  ).

3. Клацання на елементі Напис, міняє покажчик на заголовну букву **А** і маленький плюс, за допомогою якого можна вказати місце на формі, де розташовуватиметься створюваний напис. При наборі тексту немає необхідності піклується про його форматування, та і первинне місце не має особливого значення. Все це можна змінити після завершення введення і натиснення клавіші Enter.

4. Для форматування об'єкту його необхідно виділити. У такому режимі об'єкт можна перетягнути на інше місце, змінити розміри, змінити формат тексту і т.п.

В результаті цих дій отримаємо заголовок створюваної форми.

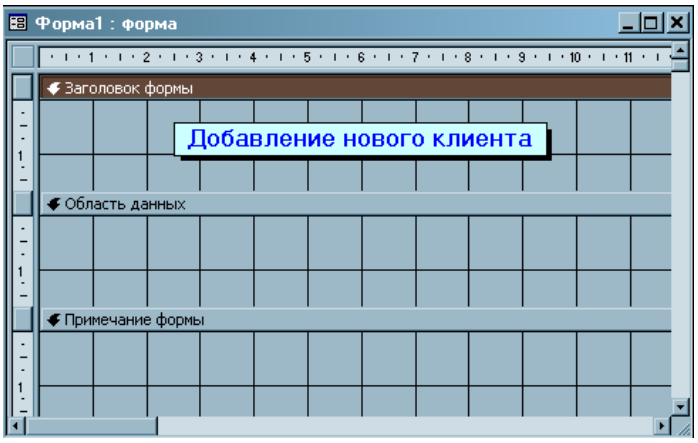



Рис. 5.7.4. Додавання заголовка у форму.

Додаткові елементи форми, такі як заголовок, не пов'язані ні з одним з полів, який або таблиці в базі даних. Тому такий елемент ще прийнято називати вільним полем.

Текст, вказаний в нім при створенні, залишається незмінним незалежно від того, який запис з БД ми проглядаємо, редагуємо або додаємо за допомогою форми в даний момент часу. Це статичний текст.

Форма насамперед призначена для роботи з полями таблиці. Тому на кожній формі обов'язково присутні елементи управління, звані зв'язаними полями. Це інтерфейс між користувачем і формою, і далі між формою і таблицею. Такий зв'язок дуже важливий і має бути строго зафіксована.

Наступний крок - це додавання полів введення або зв'язаних полів в область даних. Для цього необхідно:

1. При натисненні на  (Поле), покажчик міняє свій вигляд на зображення поля введення і редагування з маленьким плюсом, за допомогою якого можна виділити місце для майбутнього елементу на формі. В результаті в області даних форми з'явиться вільний зв'язаний елемент. Слово Вільний означає, що зв'язок для даного елементу поки не визначений.
2. Разом з появою вільного зв'язаного елементу автоматично з'являється додатковий елемент управління, званий ***приєднаний напис***. Вона переміщається формою разом зі своїм зв'язаним полем і тим самим утворює з ним єдине ціле. Поле0 - це його назва з порядковим номером, що служить для іменування елементів на формі.

Розміри кожного з елементів можна змінити, розтягуючи поля за маркери розміру. При необхідності "відірвати" поле від прикріпленого напису, скористайтеся спеціальним маркером, розташованим в лівому верхньому кутку кожного з елементів. При наведенні на нього покажчик міняється на зображення долоні з вказівним пальцем. У цей момент зв'язане поле можна відокремити від приєднаного напису (або навпаки напис від поля) і переміщати за формою самостійного.

Для точнішого позиціонування елементів на формі (з точністю до одного пікселя), служать курсорні клавіші клавіатури, використовувані спільно з CTRL і SHIFT. При утриманні клавіші SHIFT відбувається зміна розмірів виділеного елементу, а при натиснутому CTRL - змінюється положення елементу на формі.

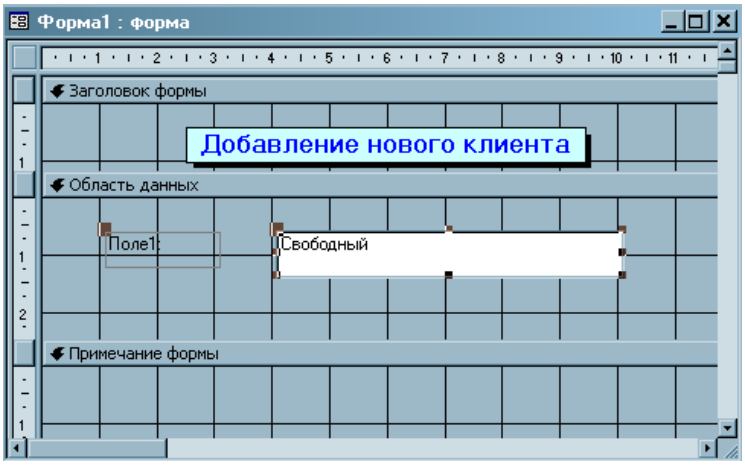



Рис. 5.7.5 Додавання вільно зв'язаного елементу і напису.

Для зміни тексту прикріпленому напису необхідно перемістити покажчик в її центральну область (при цьому він міняється на текстовий курсор) і провести клацання. Поле стане доступним для редагування. Змініть привласнений за умовчанням напис на необхідну і натисніть **Enter**. По правій кнопці миші буде доступне додаткове контекстне меню, що дозволяє вибрати тип оформлення і кольору елементів.

3. Тепер необхідно вказати для зв'язаного поля, з якою таблицею воно працюватиме. Для цього виділяємо його і натискаємо праву кнопку миші. Тепер нас цікавить пункт Властивості і закладка Дані. На ній можна набудувати зв'язок з таблицею. Якщо при створенні форми було вказано джерело, то поля відповідної таблиці будуть доступні для вибору в графі Дані.

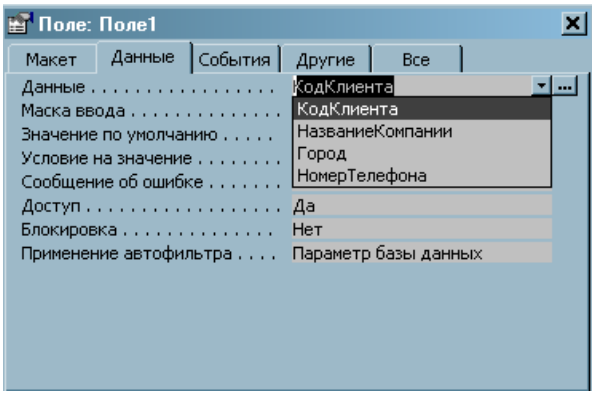



Рис. 5.7.6 Вибір поля таблиці для зв'язаного поля.

Поле Доступ у властивостях поля, показує, чи буде воно активно на формі чи ні. Неактивне поле виводиться у втопленому і затіненому вигляді. Властивість Блокування приводить до заборони або дозволу редагування інформації в полі. За умовчанням редагування і доступ до даних дозволений. Змініть ці властивості на свій розсуд.

4. Якщо при створенні форми джерело не було вказане, то необхідно буде використовувати Будівника виразів, що відкривається за допомогою кнопки із зображенням трикрапки в пункті Властивості, закладка Дані і властивості Дані. У додатковому вікні необхідно уручну вказати шлях до необхідної таблиці і вибрати поле, що цікавить Вас.

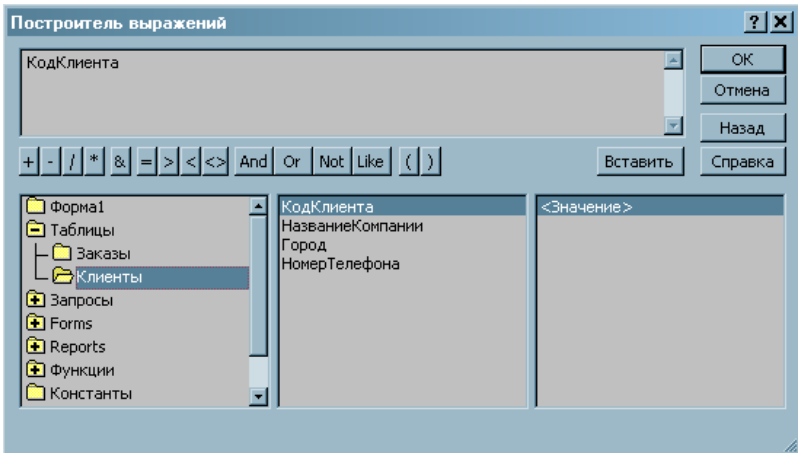



Рис. 5.7.7 Вікно Будівника виразів.

### 5.7.4 Виклик однієї форми з іншої

Хай створена форми для заповнення таблиць Клієнти і Замовлення. Необхідно додати у форму для введення замовлень можливість з неї відкривати форму для заповнення клієнтів. Для виконання цього завдання скористаємося елементом управління Кнопка .

Для цього необхідно виконати наступні дії:

1. Відкрийте форму для замовлень в режимі конструктора.
2. Виберіть в панелі елементів управління властивість Кнопка і помістите її на форму.

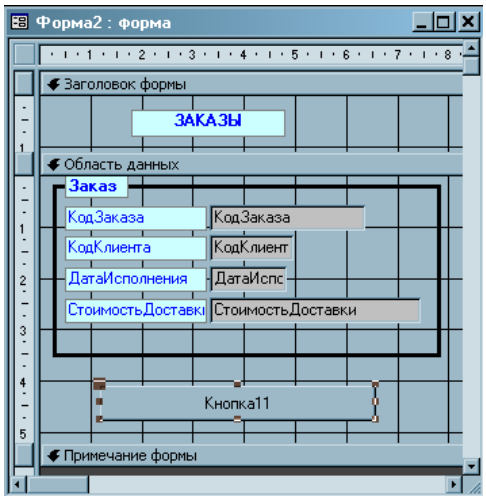



Рис. 5.7.8 Додавання кнопки у форму

3. Виберіть пункт Властивості в контекстному меню для створеної кнопки. Потім у вкладці Події властивість Натиснення кнопки. В результаті з'явиться вікно Будівник з трьома пунктами для вибору: вирази, макроси, програми. Нас цікавитимуть Макроси. В результаті з'явиться вікно з проханням назвати макрос, залишимо пропоноване ім'я Макрос 1. Будівника макросів має наступний вигляд:

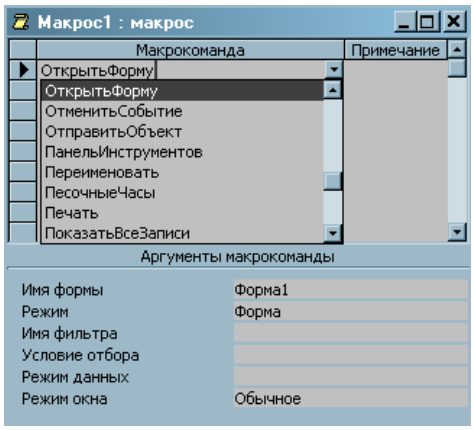



Рис. 5.7.9 Вікно для побудови макросів

4. У полі Макрокоманда з випадного списку вибираємо Відкрити Форму, а у властивостях Аргументи макрокоманди, в ім'я форми вибираємо ім'я необхідної форми. Закриваємо будівника макросів.

5. Збережете форму і перейдіть в режим форми.

Результатом натиснення на кнопку буде виклик форми для введення клієнтів.

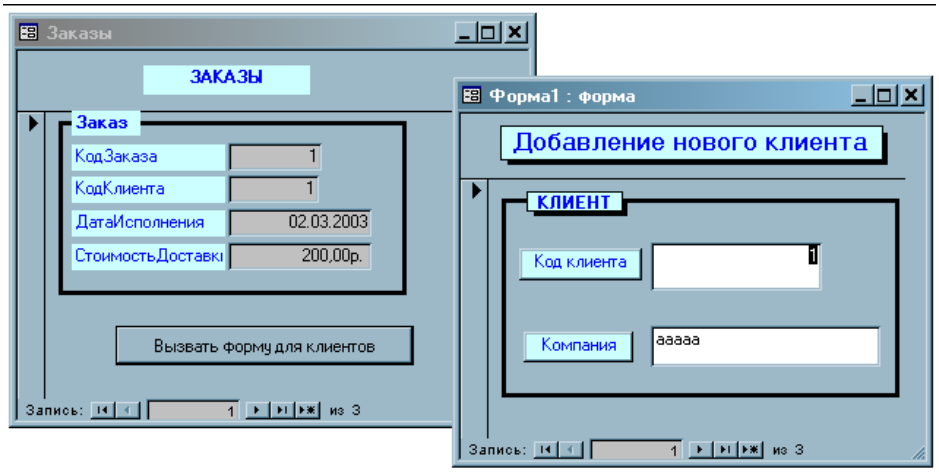



Рис. 5.7.10 Виклик однієї форми з іншої.

### 5.7.5 Додавання в Примітку форми поточної дати і час

Відкрийте форму Замовлення в режимі конструктора. При необхідності збільште область примітки форми. Додайте в цю область елемент Поле. У написі зітріть весь текст і новий не вводите. Зробіть активним елемент з ім'ям Вільний (тобто необхідно клацнути на нім мишкою). По правій клавіші миші викличте його контекстне меню і в нім виберіть пункт Властивості.

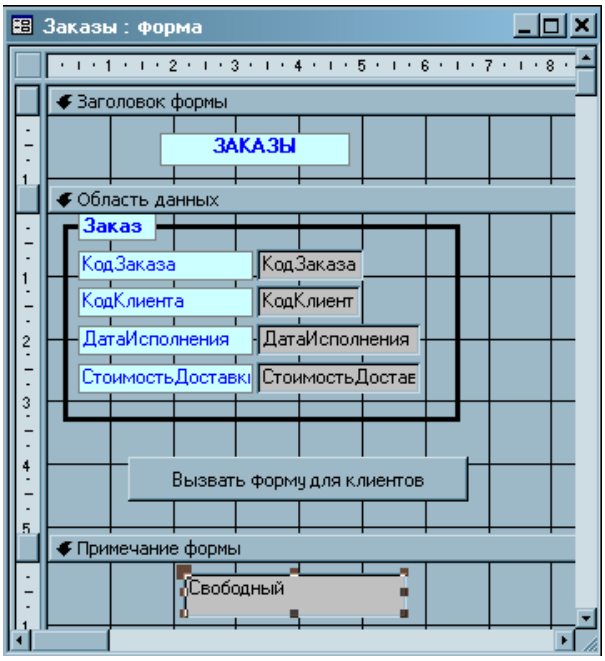



Рис. 5.7.11 Додавання поля для дати і часу в примітку форми

У вікні властивостей виберіть закладку Дані. У ній пункт Доступ зміните нанівець». Потім перейдіть в полі Дані, натисніть на кнопку ... У вікні, що з'явилося, виберіть Вираз. У вікні будівника виразів вибираєте теку Функції, в ній теку вбудовані функції. У сусідньому полі вибираєте пункт Дата/Час. Потім в останньому полі вибираєте функцію ***Now*** і двічі клацаєте по ній. Для завершення роботи з будівником виразів натисніть ОК.

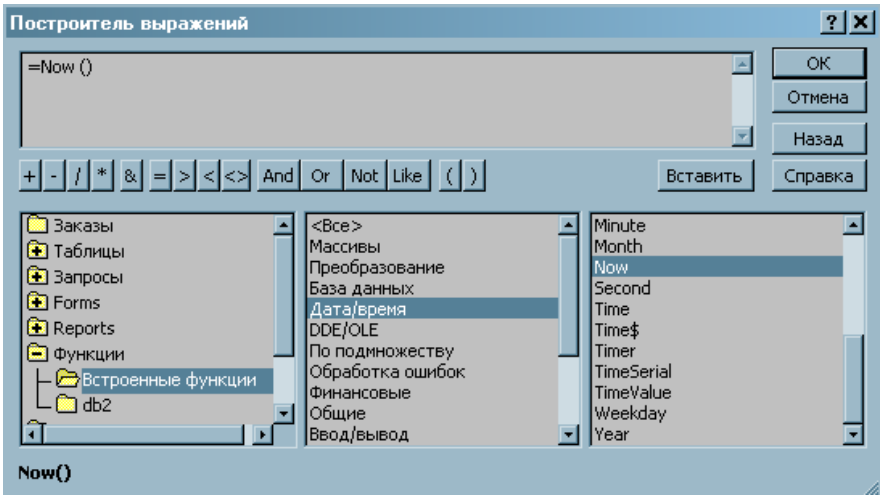



Рис. 5.7.12 Вибір функції дати і часу

В результаті після запуску форми в області примітки висвічуватиметься поточна дата і час.

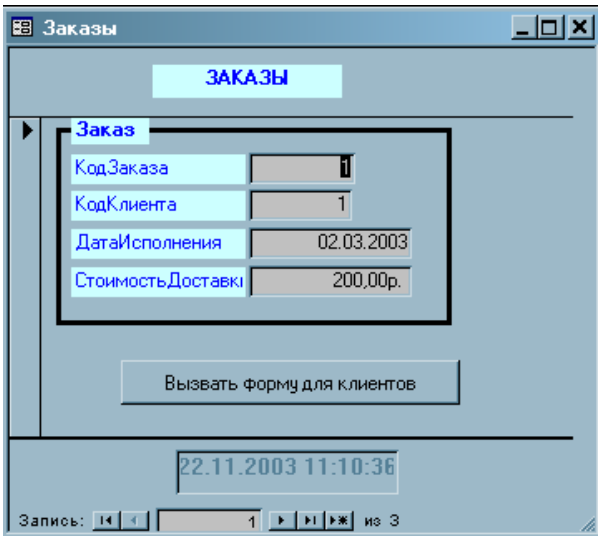



Рис. 5.7.13 Форма з поточною датою і часом.

Якщо немає необхідності виводити даті і час разом, то замість функції Now, можна використовувати наступні функції:

- ***Date()*** – для виведення системної дати
- ***Time()*** – для виведення системного часу

Контрольні питання:

1. Що таке форми? Для чого вони використовуються?
2. Які режими створення форм ви знаєте?
3. Які чотири можливих видів представлення форми ви знаєте?
4. Які області є в конструкторі форм? Для чого вони призначенні?
5. Як додати необхідний елемент управління до форми?
6. Що таке будівник виразів?

## 5.8. ACCESS. СТВОРЕННЯ ЗВІТІВ

*Способи створення звітів. Створення звіту в режимі матсера. Створення звіту в режимі конструктора.*

Звіти багато в чому схожі на форми і теж дозволяють отримати результати роботи запитів в наочній формі, але тільки не на, екрані, а у вигляді роздруку на принтері. Таким чином, в результаті роботи звіту створюється паперовий документ.

За відсутності принтера звіти створювати все-таки можна. Досить виконати програмну установку за допомогою команди операційної системи: Пуск  Налаштування  Принтери  Установка принтера, після чого зареєструвати драйвер принтера, або узявши його з гнучкого диска, або вибравши один з драйверів, що додаються до самої операційної системи.

Велика частина того, що було сказане про форми, відноситься і до звітів. Так звіти можна створити декількома способами

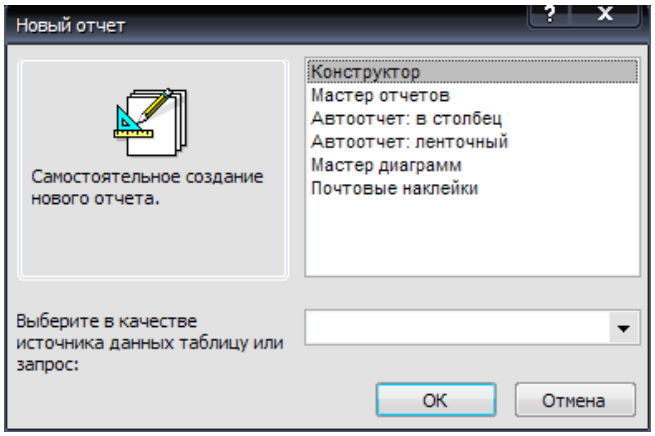



Рис.5.8.1 Способи створення звітів

Режими створення звітів:

- Конструктор.
- Майстер звітів
- Автозвіт: у стовпець
- Автозвіт: стрічковий
- Майстер діаграм
- Поштові наклейки - створення звіту, що відформатував для друку поштових наклейок.

### 5.8.1 Кроки створення звіту в режимі Майстра звітів

¬◊© 1. Виберіть для створення режим Майстер звітів і таблицю для якої будуватиметься звіт.

¬◊© 2. Вибираються поля заданої таблиці, які будуть поміщені в звіт.

¬◊© 3. Указуються поля по яких буде проводиться угрупування

¬◊© 4. Вибір сортування полів

¬◊© 5. Вибирається вид макету для звіту.

¬◊© 6. Вибирається стиль відображення звіту

¬◊© 7. Указується ім'я звіту і подальші дії з ним.

Після проходження всіх кроків Майстра буде створений паперовий макет звіту, готовий до друку. Якщо необхідно відкоректувати його, то вікно з макетом закривається і потім звіт відкривається в режимі Конструктора.

### 5.8.2 Створення звітів в режимі Конструктора

Як і форми, звіти складаються з розділів, а розділи можуть містити елементи управління. Але, на відміну від форм, розділів в звітах стільки ж, а елементів управління, менше.

Структура звіту складається з п'яти розділів: заголовка звіту, верхнього колонтитулу, області даних, нижнього колонтитулу і примітки звіту.

Розділ заголовка служить для друку загального заголовка звіту.

Розділ верхнього колонтитулу можна використовувати для друку підзаголовків, якщо звіт має складну структуру і займає багато сторінок. Тут можна також поміщати і колонцифри (номери сторінок), якщо це не зроблено в нижньому колонтитулі.

В області даних розміщують елементи управління, пов'язані з вмістом полів таблиць бази. У ці елементи управління видаються дані з таблиць для друку на принтері. Порядок розміщення і вирівнювання елементів управління той же, що і при створенні структури форм.

Розділ нижнього колонтитулу використовують для тих же цілей, що і розділ верхнього колонтитулу.

Розділ примітки використовують для розміщення додаткової інформації.

Звіт по таблиці Клієнти в режимі Конструктора виглядатиме таким чином:

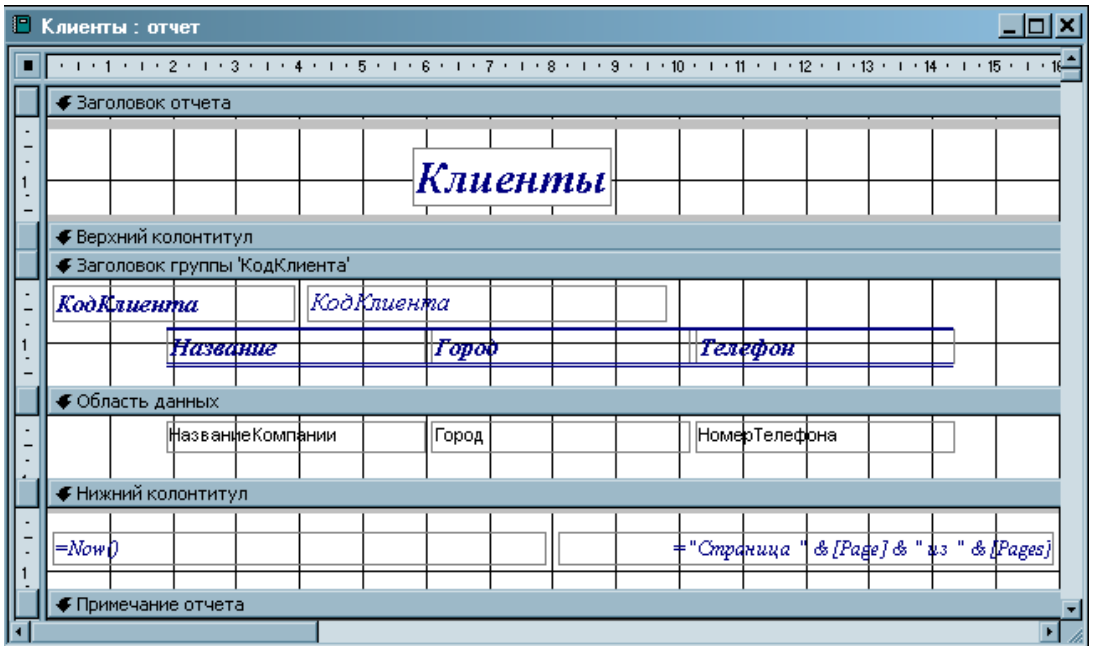



Рис. 5.8.2 Звіт в режимі Конструктора

У нижній (верхній) колонтитул можна додати наступні вирази:

- Виведення номера сторінки – ="Страница" & Page
- Число сторінок – =Pages
- Сторінка N з M – = " Страница " & Page & " из  " & Pages
- Поточна дата – = Date()
- Поточна дата і час – = Now()
- Поточний користувач – = CurrentUser()

**Контрольні питання:**

1. Що таке звіт? Для чого він використовується?

2. Які режими створення звітів ви знаєте?
3. З яких п'яти розділів складається структура звіту у режимі Конструктора?
4. Для чогопризначен кожен розділ?