



Урок 2

План заняття:

1. Модель реляційної бази даних. Первинні ключі
2. Зв'язки
 - 2.1. Поняття зв'язків між таблицями. Зовнішні ключі
 - 2.2. Типи зв'язків
3. Нормалізація таблиць
 - 3.1. Базові основи нормалізації. Нормальні форми
 - 3.2. Аналіз та нормалізація даних засобами СУБД MS Access
4. Забезпечення цілісності даних. Схема даних
5. Форматування даних при вставці в таблицю. Введення даних засобами СУБД
6. Імпорт та зв'язування таблиць
 - 6.1. Імпорт електронних таблиць
 - 6.2. Імпорт текстових файлів
 - 6.3. Зв'язування з таблицями інших баз даних
7. Експорт бази даних MS Access
8. Домашнє завдання

1. Модель реляційної бази даних. Первинні ключі

На минулій парі ми розглядали таке поняття як проектування та побудову баз даних та її основних об'єктів – таблиць. Отже, будувати таблиці Ви вже вмієте, але одного такого вміння недостатньо. Ваша база даних повинна бути ефективною і містити таку кількість таблиць, яка необхідна. Крім того, кожна таблиця повинна бути побудована згідно основних правил побудови таблиць, оскільки тільки в цьому випадку розроблена база даних буде реляційною і продуктивною.

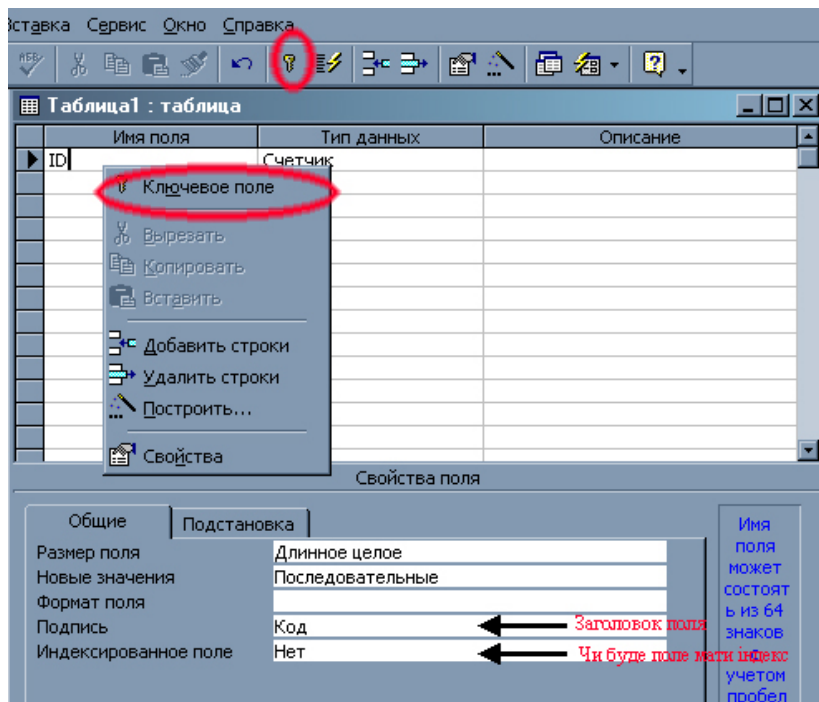
Згідно згаданих вище правил побудови, кожна таблиця в реляційній базі даних повинна складатись з полів, які містять характеристики даних, та записів, що містять безпосередньо самі значення. Кожна таблиця повинна мати **первинний ключ (primary key)** – значення, яке унікально ідентифікує кожен запис в межах таблиці. Первинний ключ, крім того, що вказує на унікальність значень, дозволяє забезпечити швидкий доступ до конкретного запису таблиці.

Значення первинного ключа розміщується в окремому полі або групі полів таблиці і, як правило, в імені міститься префікс або суфікс ID. Наприклад, ProductID, IdGroup тощо. Здебільшого поле, що містить значення первинного ключа розміщують самим першим в списку полів.

Розглянемо практичний приклад побудови таблиці засобами СУБД MS Access з первинним ключем. Наша таблиця буде містити інформацію про працівників підприємства та носитиме назву **Employee**. Враховуючи факт, що першим полем (ключовим полем, тобто полем, що містить первинний ключ) повинен бути унікальний ідентифікатор, її структура набуде наступного вигляду:

| EmployeeId | FName | LName | Position | Department | PhoneDepartment |
|------------|--------|---------|---------------|------------|-----------------|
| 01 | Вася | Пупкін | Інженер | 115 | 21-21-21 |
| 02 | Вова | Петров | Інженер | 126 | 36-32-21 |
| 03 | Оксана | Головач | Адміністратор | 126 | 36-32-21 |

При створенні поля **EmployeeId**, необхідно добитись унікальності значень. Тут можна піти **двома шляхами**: зробити поле числового типу і дозволити користувачу самому, при внесенні нових значень в таблицю, вказувати унікальні значення первинного ключа, або ж дозволити системі генерувати ці значення автоматично. В першому випадку, гарантувати унікальність важко, оскільки користувачі можуть робити помилки та присвоювати одне і те ж значення двом працівникам (особливо в великій системі). Тому, краще покластись на інструменти самої СУБД, які гарантовано дадуть необхідний результат – унікальні значення.



В MS Access для автоматизації процесу генерування унікальних значень в межах таблиці використовується тип «Счетчик». Слід відмітити, що такий тип існує лише в MS Access, в інших СУБД для цих цілей використовуються інші засоби. Щоб знати, які саме, перед створенням таблиці потрібно прочитати відповідну документацію.

Та встановити для поля тип лічильника недостатньо. Щоб зробити його первинним ключем потрібно обрати в контекстному меню даного поля пункт «Ключевое поле» або ж скористатись кнопкою з ключиком на панелі інструментів СУБД.

Щодо закладки «Общие», то на ній ми можемо задати розмір поля, механізм генерації чисел (послідовні, тобто з кроком 1, чи довільні), формат поля, заголовок поля в режимі таблиці (по замовчуванню в якості заголовка використовується ім'я поля в конструкторі), а також вказати на те, чи буде дане поле індексом. Про індекси ми поговоримо дещо пізніше, але,

для інформації, можна зауважити, що вони використовуються для швидкого доступу до даних в базі даних.

Створення інших полів таблиці у Вас не повинно викликати труднощів, оскільки таблиці Ви вже створювали на минулій парі. Наголосимо лише на наступному зауваженні, яке варто взяти собі за правило при побудові таблиць бази даних: **назви полів та самих таблиць потрібно писати латинськими літерами, без пропусків та різноманітних символів (крім символа нижнього підкреслення (_))**. Це правило пов'язано з рекомендацією, щоб при побудові прикладного інтерфейсу до Вашої бази даних на певній мові програмування у Вас не виникало труднощів у створенні запитів за допомогою мови SQL. Якщо поля Ваших таблиць або самі назви таблиць не будуть відповідати вищеописаним критеріям, то при доступі до них, в найкращому випадку, Вам доведеться їх назви брати у кутові дужки або інші управляючі символи. В гіршому - такий спосіб не пройде і доведеться повністю переробляти базу даних (☹).

2. Зв'язки

2.1. Поняття зв'язків між таблицями. Зовнішні ключі

Як Ви вже знаєте, база даних може складатись з багатьох таблиць, які містять певну інформацію. Здебільшого, інформація, яку Ви в ній (базі даних) зберігаєте має один напрямок, наприклад, містить інформацію про роботу підприємства, дані про студентів та викладачів академії тощо. Цю інформацію Ви логічно можете розмістити в різних таблицях, але кожна з цих таблиць буде автономною і не залежати від інших. Як же їх зв'язати?

Як Ви вже знаєте, кожна реляційна таблиця повинна містити первинний ключ, який унікально характеризує кожен її запис. Але, крім даної функції, він дозволяє реалізувати і зв'язки між таблицями, завдяки яким дані з однієї таблиці стають доступні для іншої. В такому випадку, якщо база даних містить кілька таблиць, робота в ній стає більш ефективною, полегшується ввід даних в таблицю, знижується імовірність помилок.

Зв'язок реалізується за рахунок того, що в кожній з таблиць міститься по одному полю, яке має однакове значення. Доречі, ці поля можуть мати різні імена, достатньо лише дотримуватись однозначної відповідності даних. Для однієї з таблиць таким полем є первинний ключ, а для другої таблиці – цим полем буде поле **зовнішнього ключа (foreign key)**.

Отже, **зовнішній ключ** – це поле таблиці, значення якого співпадають з значенням поля, що є первинним ключем іншої таблиці.

Отже, за рахунок пари первинний ключ-зовнішній ключ відбувається зв'язок даних між двома таблицями.

2.2. Типи зв'язків

Зв'язок – це спосіб пояснити СУБД, яким чином слід виконувати вибірку інформації з таблиць баз даних. Між таблицями бази даних існують 4 типи зв'язків:

1. **ОДИН-ДО-ОДНОГО (1:1):** кожному запису таблиці А відповідає лише один запис таблиці Б (або навпаки). Такий тип зв'язків використовується рідко, оскільки фактично всі дані можуть бути розміщені в одній таблиці. Він може бути корисний у випадку, коли, наприклад, доцільно логічно розділити одну громіздку таблицю. Схематично це можна зобразити наступним чином:





Приклад: існують дві таблиці, одна з яких містить дані про працівників підприємства, а друга – професійні відомості про них. В такому випадку можна сказати, що між цими таблицями існує відношення один-до-одного, оскільки для однієї людини (з першої таблиці) може існувати лише один запис, який містить професійні відомості у другій таблиці.

2. **ОДИН-ДО-БАГАТЬОХ (1:Б):** один запис таблиці А зв'язаний з багатьма записами таблиці Б, але одному запису таблиці В не може відповідати декілька записів таблиці А. Схематичне зображення зв'язку:

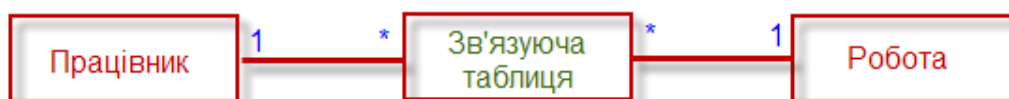


Приклад: квартира може бути пустою, в ній може жити одна або кілька пожильців.

3. **БАГАТО-ДО-ОДНОГО (Б:1)** - зворотній до попереднього. Тип відношення між об'єктами залежить від Вашої точки зору. Якщо розглядати відношення пожильців до квартири, тоді формується зв'язок багато-до-одного.
4. **БАГАТО-ДО-БАГАТЬОХ (Б:Б).** Виникає між двома таблицями у випадках, коли:
- один запис з першої таблиці може бути зв'язаний більше, ніж з одним записом з другої таблиці;
 - один запис з другої таблиці може бути зв'язаний більше, ніж з одним записом з першої таблиці.

Такий зв'язок мало поширений, але якщо він існує, то зазвичай вводить проміжна таблиця (таблиця-зв'язок), яка складається лише з зовнішніх ключів і зв'язує дві таблиці між собою.

Приклад: підприємство, на якому працюють працівники (один або більше). Крім того, цілком імовірна ситуація, що працівники працюють на кількох підприємствах і ця інформація зберігається в базі даних. В такому випадку, між цими таблицями (Робота та Працівники) існує зв'язок багато-до-багатьох.



Існують наступні **правила, яким повинні відповідати типи даних зв'язаних полів:**

- загальні та зв'язані поля повинні бути однакового типу;
- якщо обидва зв'язаних поля мають числовий тип, то вони повинні мати і однакові значення властивості «Розмір поля»;
- поле з типом даних «Счетчик» можна зв'язати з числовим полем, в якого «Розмір поля» має значення «Длинное целое».

3. Нормалізація таблиць

3.1. Базові основи нормалізації. Нормальні форми

На минулій парі ми розглядали з Вами таке поняття як проектування бази даних, але ми не зупинялись на тому, що ж буде відбуватись після того, як Ви її спроектуете. Тобто чи буде вона оптимально спроектованою, мати ефективну структуру, чи не існує збитковості (наприклад, чи в двох таблицях повторюється одна і та ж інформація чи ні), чи забезпечується цілісність даних тощо. Щоб перевірити її на оптимальність не варто ломати голову над придумуванням своїх власних методів аналізу. Для цього достатньо при проектуванні дотримуватись такої процедури як **нормалізація**, яка дозволить покращити структуру Вашої бази даних та зробити її максимально ефективною.

Нормалізація – це покроковий процес заміни однієї таблиці (або набору таблиць) іншими, що мають більш просту структуру. Іншими словами, - це процес розбиття таблиці з складною громіздкою структурою на дві і більше простих. Це призводить до виключення збитковості інформації, її впорядкування, економії пам'яті, збільшення швидкості доступу до даних тощо. На кожному етапі нормалізації таблиці зводяться до певного вигляду, що називають **нормальною формою**.

Щоб краще розібратись з нормалізацією та тим, коли таблиці будуть являтися нормальними формами розглянемо приклад. Припустимо, в нас існує таблиця Employee, яка має наступну структуру та містить певний набір даних:

| EmployeeId | FullName | Position | Salary | Department | PhoneDepartment |
|------------|----------------|---------------|--------|------------|-----------------|
| 1 | Вася Пупкін | Інженер | 4 500 | 115 | 21-21-21 |
| 2 | Вова Петров | Інженер | 5 000 | 127 | 21-17-87 |
| 3 | Оксана Головач | Програміст | 3 000 | 126 | 21-32-21 |
| 4 | Оксана Головач | Адміністратор | 2 000 | 126 | 21-32-21 |
| 5 | Вова Маякін | Програміст | 5 000 | 127 | 21-17-87 |

Проаналізуємо:

1. В даній таблиці дублюється інформація про посади та відділи працівників.



2. Якщо зміниться номер відділу, то необхідно змінювати його у всіх працівників даного відділу. Аналогічна ситуація і по телефонах.
3. Неможливо включити дані про новий відділ, поки не будуть набрані дані про його працівників (парадоксально, але факт).

Тому, згідно правилам нормалізації, потрібно виконати декомпозицію даної таблиці **Employee** і розбити її на менші, які мають оптимальнішу структуру. Кожна з таблиць повинна бути нормальною формою, тобто повинна відповідати трьом основним правилам нормалізації (або трьом нормальним формам). Дані правила застосовуються послідовно, починаючи з першої нормальної форми. Слід також враховувати той факт, що кожна наступна форма, починаючи з першої, являється більш досконалішою за попередню з точки зору збитковості.

Розпочнемо процес. Спочатку потрібно виділити **першу нормальну форму**. Для цього проаналізуємо наші дані на наявність в одному полі даних, які можна розмістити в різних полях, оскільки вони приводять до збитковості. В нашому прикладі, поле FullName містить неоптимальні дані, оскільки, доволі часто необхідно здійснювати пошук даних по прізвищу працівника, а його ім'я невідомо. В такому випадку ці дані варто розмістити в окремих полях. Результатом таких перетворень буде таблиця наступного вигляду:

| EmployeeId | FName | LName | Position | Salary | Department | PhoneDepartment |
|------------|--------|---------|---------------|--------|------------|-----------------|
| 1 | Вася | Пупкін | Інженер | 4 500 | 115 | 21-21-21 |
| 2 | Вова | Петров | Інженер | 5 000 | 127 | 21-17-87 |
| 3 | Оксана | Головач | Програміст | 6 500 | 126 | 21-32-21 |
| 4 | Оксана | Головач | Адміністратор | 6 500 | 126 | 21-32-21 |
| 5 | Вова | Маякін | Програміст | 5 000 | 127 | 21-17-87 |

Тепер поля нашої таблиці містять атомарні значення, тобто такі, які не можна поділити на менші складові. Процес пошуку працівника спрощений. Крім того, дана таблиця знаходиться в першій нормальній формі (1НФ), оскільки **кожен атрибут кожного запису таблиці (тобто кожна комірка таблиці) містить лише атомарні (неподільні) значення**. Іншими словами, при 1НФ кожне поле повинно містити не більше одного значення, яке не можна розділити на частини.

Другий етап нормалізації – це виділення **другої нормальної форми**. Якщо проаналізувати більш детально нашу таблицю, то видно, що значення багатьох полів повторюються. Так, один працівник може працювати на кількох посадах, і одну посаду можуть посідати кілька працівників. При цьому людині, яка буде вносити дані в базу даних доведеться часто вводити однакові значення, що підвищує імовірність помилок при вводі, на пошук та виправлення яких піде час. Щоб цього уникнути та ще більше підвищити пошук даних по базі даних, потрібно розбити таблицю на менші. Слід також не забувати, що кожна з таблиць повинна містити атомарні значення, інакше вони не будуть відповідати 1НФ. В такому разі потрібно для кожної з таблиць повторювати перший етап нормалізації.

Отже, в результаті спрощення ми отримуємо 3 таблиці: **Employee**, **Position** та **Department**. Кожна з цих таблиць має свій власний первинний ключ і вони незалежні одна від одної.

Employee

| EmployeeId | FName | LName | Salary |
|------------|--------|---------|--------|
| 1 | Вася | Пупкін | 4 500 |
| 2 | Вова | Петров | 5 000 |
| 3 | Оксана | Головач | 6 500 |
| 4 | Вова | Маякін | 5 000 |

Position

| PositionId | Name |
|------------|---------------|
| 1 | Інженер |
| 2 | Адміністратор |
| 3 | Програміст |

Department

| DepartmentId | NumberDepartment | PhoneDepartment |
|--------------|------------------|-----------------|
| 1 | 115 | 21-21-21 |
| 2 | 126 | 21-32-21 |
| 3 | 127 | 21-17-87 |

Правило говорить, що **таблиця знаходиться в другій нормальній формі (2НФ), якщо:**

- вона знаходиться в першій нормальній формі;
- всі дані, які можна логічно відділити від інших, виділений в окрему таблицю;
- всі поля, які логічно відносяться до виділених даних, винесені в цю таблицю;
- для кожної таблиці введений унікальний ідентифікатор.



Отже, наші таблиці знаходяться в другій нормальній формі, оскільки тепер по значенню ідентифікатора будь-якої таблиці можна знайти всі значення, які йому відповідають. Наприклад, по значенню поля DepartmentId ми отримаємо інформацію його назву та телефон. Коім того, виділивши окремі таблиці ми можемо з легкістю отримати списки всіх відділів та працівників.

Третій рівень нормалізації – це виділення **третьої нормальної форми**. Таблиця знаходиться в третій нормальній формі (3НФ), якщо вона задовольняє вимоги 2НФ і не одне з її неключових полів не залежить функціонально від інших не ключових полів. Простішими словами, визначені всі зв'язки між таблицями і введені додаткові поля для цього.

Щоб наші таблиці знаходились в третій нормальній формі потрібно їх зв'язати і у випадку необхідності ввести додаткові таблиці. Для зв'язку таблиць між собою вводимо зовнішні ключі в таблицю Employee, адже дві інші з нею пов'язані.

| EmployeeId | FName | LName | Salary | PositionId | DepartmentId |
|------------|--------|---------|--------|------------|--------------|
| 1 | Вася | Пупкін | 4 500 | 1 | 1 |
| 2 | Вова | Петров | 5 000 | 1 | 3 |
| 3 | Оксана | Головач | 6 500 | 3 | 2 |
| 4 | Оксана | Головач | 6 500 | 2 | 2 |
| 5 | Вова | Маякін | 5 000 | 3 | 3 |

Але цього замало, оскільки при встановленні зв'язків в нашій таблиці присутнє дублювання даних, яке стосується працівників, тобто їх імена і прізвища, а це не допустимо. Це пов'язано з тим, що один працівник може працювати на кількох посадах і одночасно з тим на одній посаді може працювати кілька працівників. Фактично, у нас утворився зв'язок БАГАТО-ДО-БАГАТЬОХ, в зв'язку з чим виникає необхідність ввести проміжну таблицю-зв'язок.

Employee

| EmployeeId | FName | LName | Salary | DepartmentId |
|------------|--------|---------|--------|--------------|
| 1 | Вася | Пупкін | 4 500 | 1 |
| 2 | Вова | Петров | 5 000 | 3 |
| 3 | Оксана | Головач | 6 500 | 2 |
| 4 | Вова | Маякін | 5 000 | 3 |

Position

| PositionId | Name |
|------------|---------------|
| 1 | Інженер |
| 2 | Адміністратор |
| 3 | Програміст |

EmployeePosition

| Id | EmployeeId | PositionId |
|----|------------|------------|
| 1 | 1 | 1 |
| 2 | 2 | 1 |
| 3 | 3 | 3 |
| 4 | 3 | 2 |
| 5 | 4 | 3 |

Доречі, первинний ключ в проміжній таблиці не являється обов'язковим.

Після таких перетворень наші таблиці будуть знаходитись в третій нормальній формі, оскільки всі вони містять атомарні значення (вимога 1НФ), дані не дублюються (вимога 2 НФ) і містять зовнішні ключі для зв'язування (вимога 3НФ).

Таким чином ми нормалізували нашу базу даних. Слід відмітити, що всі здійснені нами перетворення хоча і виконуються в суворій відповідності з теорією нормалізації, інтуїтивно зрозумілі і очевидні. З досвідом процес проектування баз даних, вибору необхідної структури таблиць та визначення оптимальних зв'язків між ними вже не буде для Вас настільки загадковим та складним, як на перших етапах освоєння теорії баз даних.

Та наша розповідь була б неповною, якщо б ми не сказали, що насправді третя нормальна форма – це не останній етап нормалізації, існують форми і вищого рівня: 4НФ, 5НФ та 6НФ. Однак ситуації, які потребують форм такого рівня надзвичайно рідкий випадок і тому ми їх розглядати не будемо.

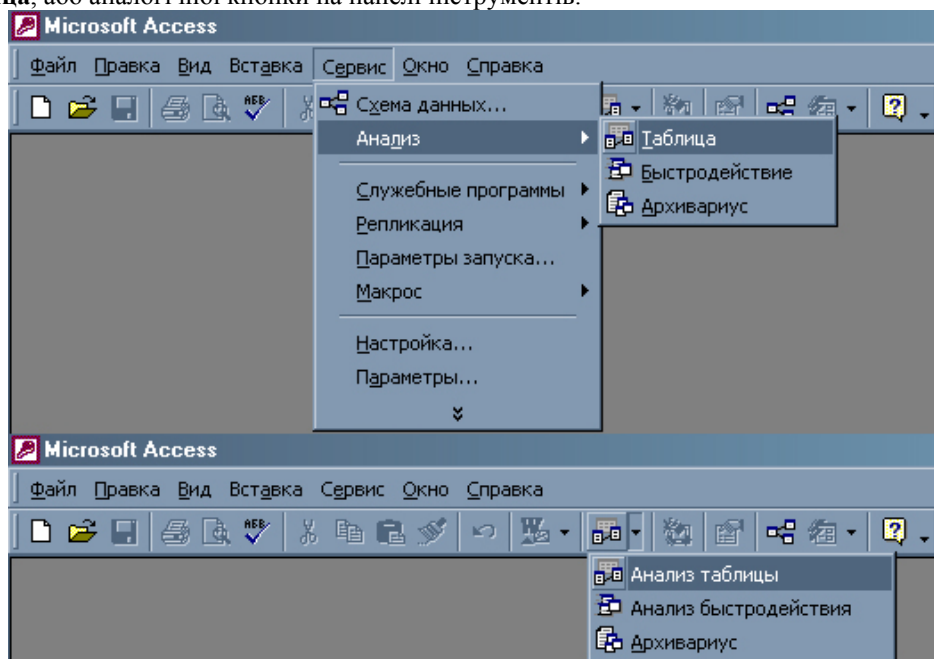
Крім того, теоретики реляційних систем Кодд і Бойс запропонували більш жорстке визначення для 3НФ, яке враховує, що в таблиці може існувати кілька можливих ключів. Таблиця знаходиться в нормальній формі Бойса-Кодда (НФБК), якщо і тільки якщо будь-яка функціональна залежність між її полями зводиться до повної функціональної залежності від можливого ключа.

3.2. Аналіз та нормалізація даних засобами СУБД MS Access

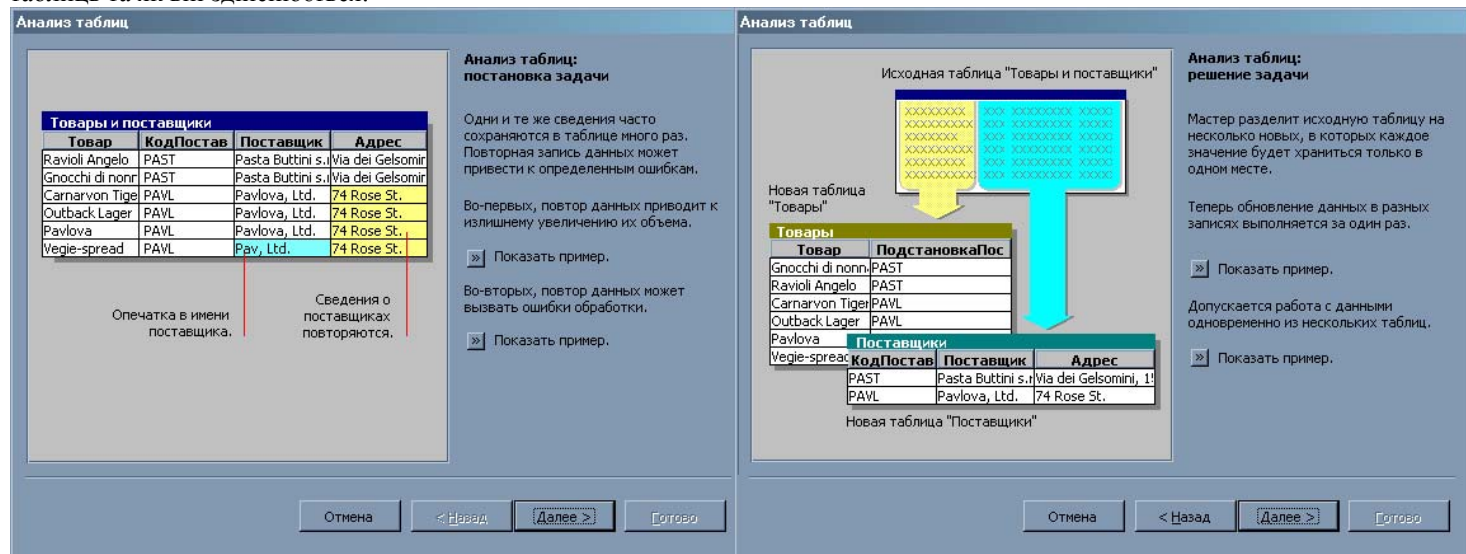
Крім того, що у нас є можливість власноручно організувати нормальні форми, в СУБД MS Access 2000/2003/XP існує інструмент «Мастер анализа таблиць», що дозволяє перевірити структуру таблиці і, у випадку необхідності, розділити її



на нові зв'язані таблиці, що призводить до ефективності зберігання даних. Даний інструмент можна викликати через пункт меню **Анализ->Таблица**, або аналогічної кнопки на панелі інструментів.



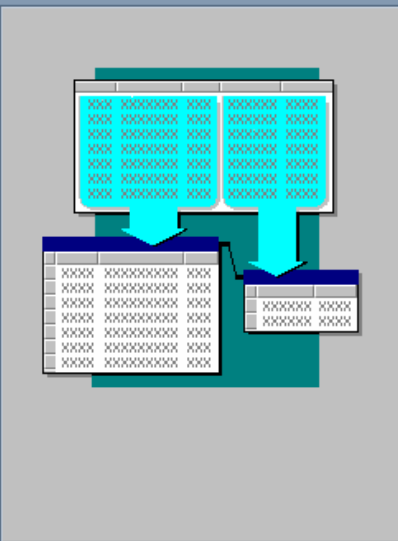
Механізм роботи аналізатора дуже простий. Для прикладу візьмемо нашу ненормалізовану ще первинну таблицю з інформацією про працівників деякої компанії. Обираємо пункт панелі інструментів **“Анализ таблиц”** і в нас автоматично запускається візард аналізатора, на перших двох кроках якого розказують про те, для чого необхідно проводити аналіз таблиць та як він здійснюється:





Потім Вам запропонують обрати таблицю, яку необхідно проаналізувати на відповідність основним принципам нормалізації:

Анализ таблиц



В каких таблицах есть поля, значения которых повторяются много раз?

Данный мастер разделит исходную таблицу на несколько связанных таблиц. При этом сама исходная таблица изменена не будет.

Таблицы:

| |
|----------|
| DataBase |
| Employee |
| Students |
| Таблица1 |

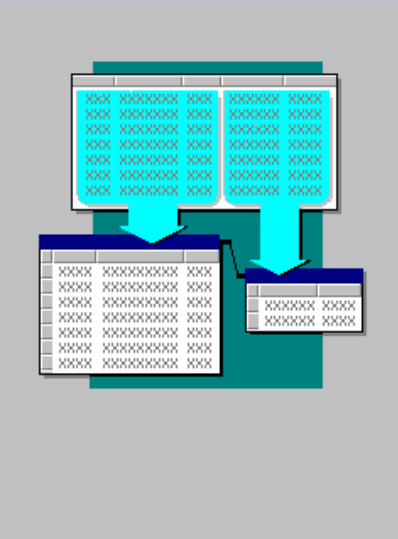
☒ Вывести страницы с пояснениями?

От

Далі потрібно обрати чи поклатися на досвід візарда в плані здійснення нормалізації чи зробити все вручну самому.

У випадку вибору першого варіанту ми маємо наступну ситуацію: Вам запропонують спочатку подивитись на результат розподілу з представленням можливості переіменування таблиці. Після цього запропонують власноруч додати первинні ключі в тих таблицях, які візарду не підвласні, тобто початкових:

Анализ таблиц



Мастер позволяет автоматически разделить поля по таблицам. Воспользоваться этим средством?

При автоматическом разделении сохраняется возможность проверки и изменения состава таблиц.

☒ Да, разделение полей выполняется мастером.

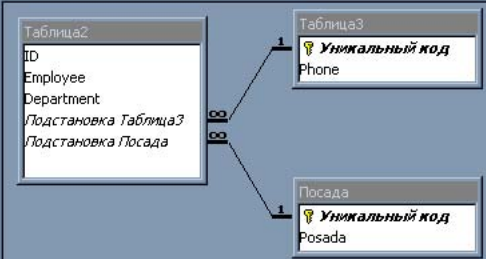
☐ Нет, разделение проводится вручную.

Отмена < Назад Далее > Готово

Анализ таблиц

Правильно ли мастер группирует данные?
Если нет, перетащите поля с помощью мыши в подходящие по смыслу группы.

Какие имена следует присвоить новым таблицам?
Как правило, имя таблицы указывает на хранящиеся в ней данные.



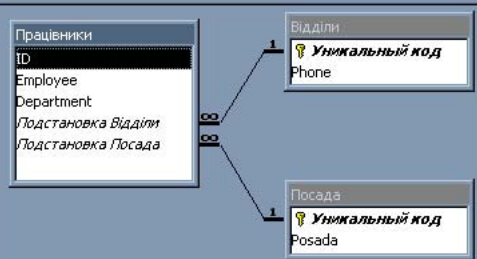
Имя таблицы:

OK Отмена Далее > Готово

Анализ таблиц

Однозначно ли определяют отмеченные поля каждую запись указанной таблицы?

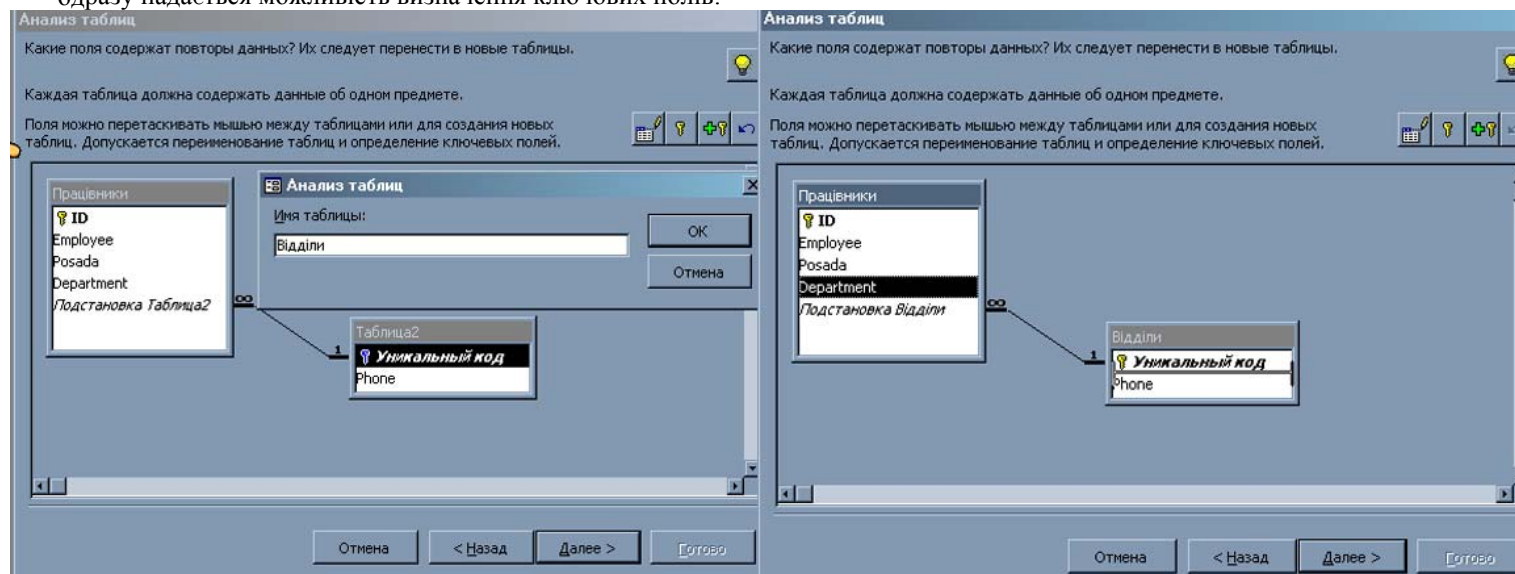
Значение ключевого поля должно быть уникальным для каждой записи таблицы. Если в таблице нет полей без совпадающих значений, мастер позволяет добавить поле автоматически присваиваемого уникального кода записи.



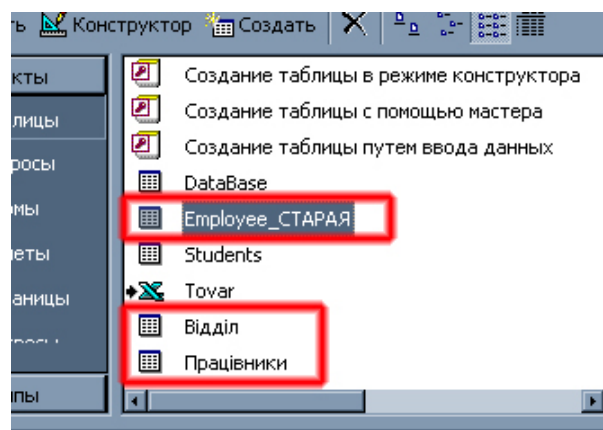
Отмена < Назад Далее > Готово



У випадку вибору другого варіанту відкриється схоже вікно, в якому Ви повинні власноручно створити таблиці простим рухом мишки. Для початку Ви обираєте поле, яке необхідно винести в іншу таблицю, а потім клікаєте на цьому полі і, утримуючи ліву кнопку миші, перетягуєте його на пусту робочу область. Перемістивши поле, відпускаємо кнопку миші. Після цього у Вас утвориться нова таблиця та візард запропонує ввести її нову назву. При бажанні додати до новоствореної таблиці ще одне поле, Ви пророблюєте майже схожі дії, тобто перетягуєте бажане поле за допомогою миші в зарезервоване для нього місце в іншій таблиці. Після завершення всіх дій по створенню оптимальної структури таблиць одразу надається можливість визначення ключових полів:

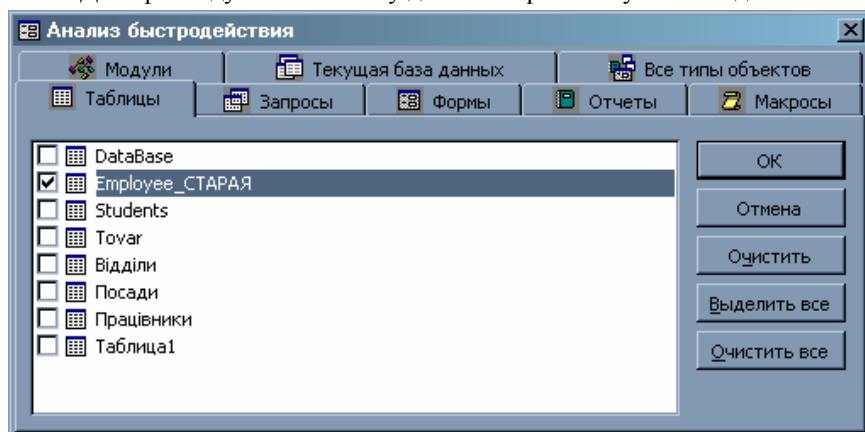


В кінцевому результаті ми отримуємо визначену на етапі аналізу кількість таблиць з вже перенесеними даними:

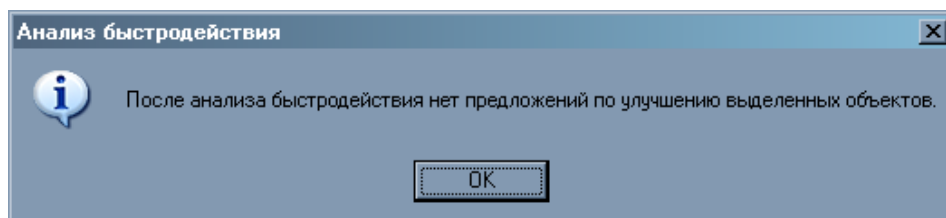


До вбудованих засобів автоматизації слід також віднести і «Анализатор быстрогодействия», що аналізує продуктивність БД і її об'єктів, даючи рекомендації по їх покращенню.

Для прикладу візьмемо базу даних та проаналізуємо її за допомогою даного інструменту.

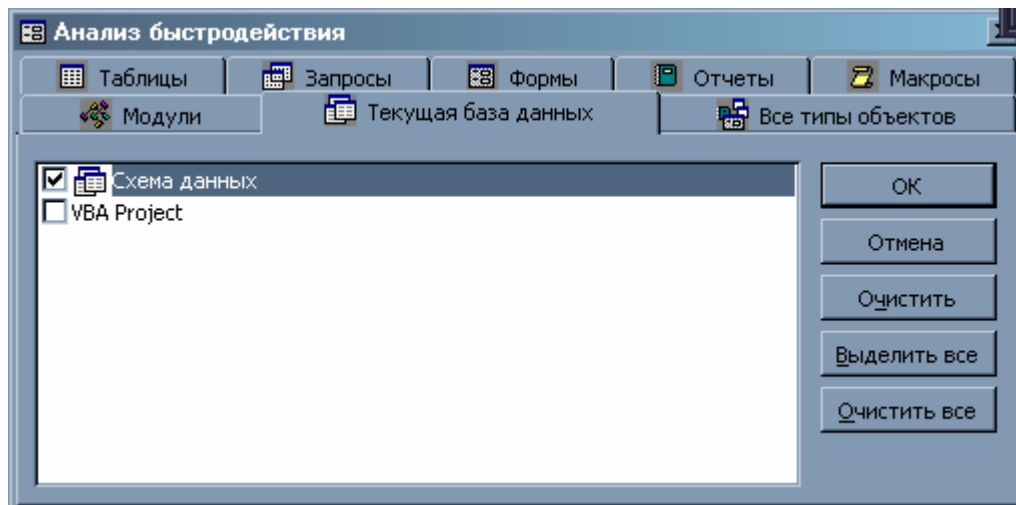


Для цього в вікні аналізатора обираємо ту таблицю, запит, звіт тощо, які ми хочемо проаналізувати на швидкодію. Після натиснення кнопки «ОК», з'явиться вікно з рекомендаціями.

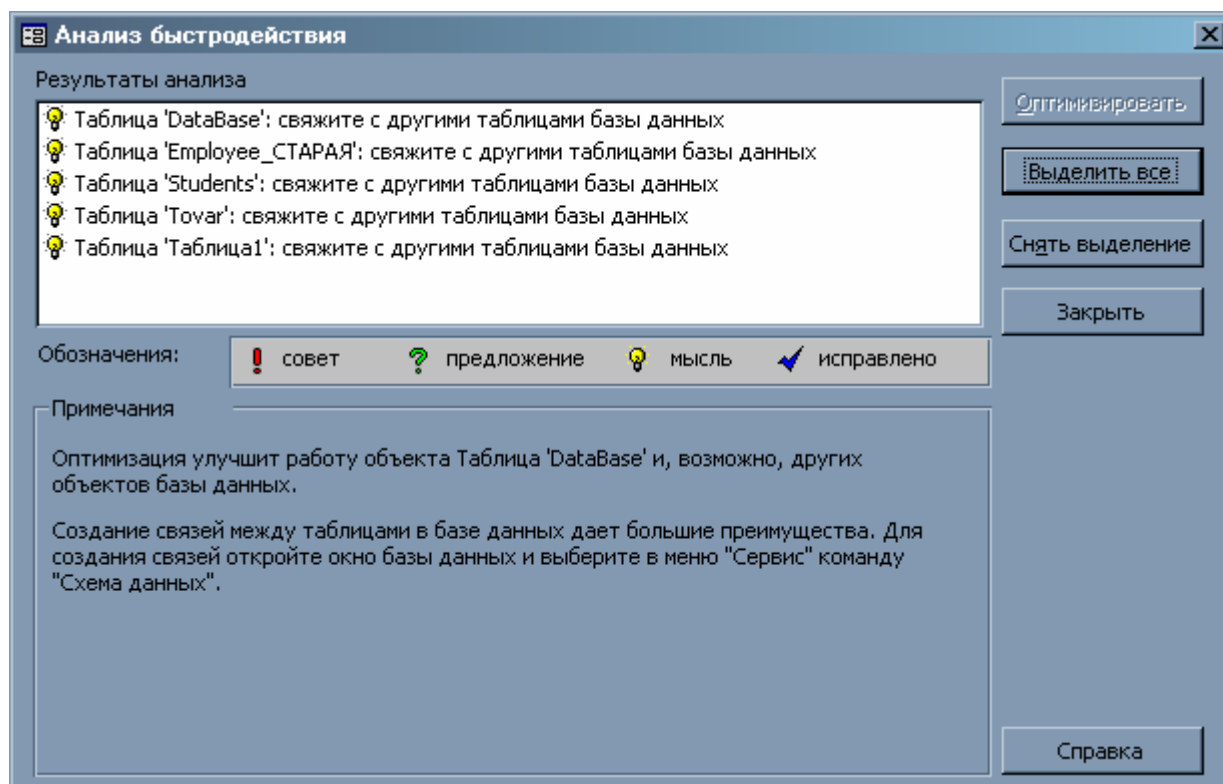




Як бачимо, рекомендацій по нашій таблиці немає, все в нас добре. Підемо далі та піддамо аналізу всю базу даних, а саме схему даних:



А тут не все так добре. В принципі, це і так зрозуміло, оскільки наша база даних експериментальна та більшість таблиць в ній не мають одна до одної ніякого відношення і тому про зв'язки між ними немає сенсу і говорити.



4. Забезпечення цілісності даних. Схема даних

Крім того, що необхідно здійснити нормалізацію даних, виділити нормальні форми, встановити зв'язки між ними, необхідно також забезпечити цілісність даних, які зв'язані. **Цілісність даних** – це система правил, що регулює взаємодію між зв'язаними таблицями і забезпечує коректність зберіганих в таких таблицях даних. База даних являє собою динамічний механізм, в якому постійно вносяться нові дані, видаляються непотрібні і модифікуються вже існуючі. В зв'язку з цим, головна задача засобів забезпечення цілісності даних заключається в тому, щоб БД постійно містила точну та актуальну інформацію.

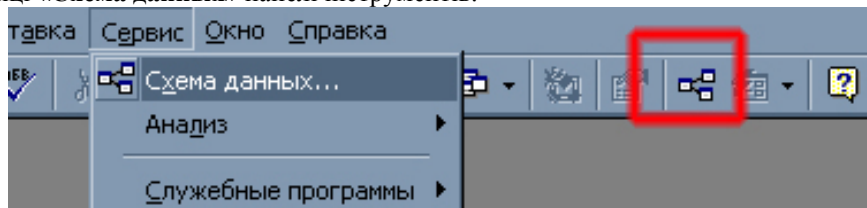
Цілісність даних говорить про те, що дані, які введені в спільне поле двох зв'язаних таблиць, повинні співпадати. З двох зв'язаних таблиць одна називається батьківською, а інша - дочірною, тому коректніше буде сказати, що дані, які вводяться в поле зовнішнього ключа дочірньої таблиці, повинні співпадати з даними, що зберігаються в полі первинного ключа батьківської таблиці. При будь-яких спробах їх модифікації СУБД MS Access автоматично перевіряє їх значення. Якщо така зміна порушує встановлений між таблицями зв'язок, видається повідомлення про порушення цілісності даних.

Отже, при проектуванні та створенні бази даних необхідно передбачити всі можливі випадки роботи з нею і забезпечити цілісність її даних.



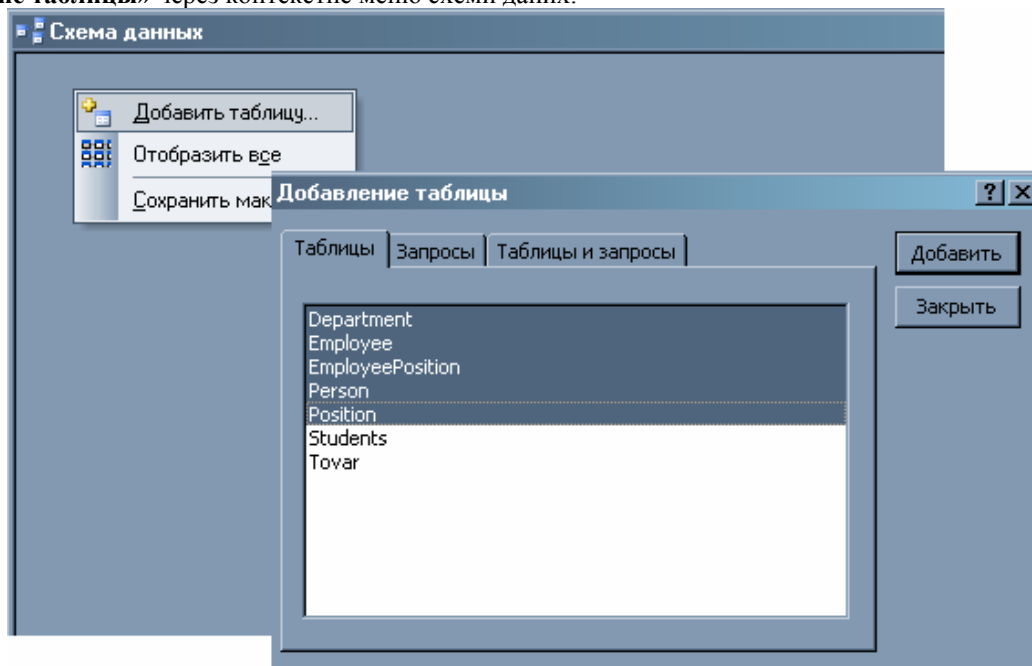
Для цього необхідно здійснити наступні дії:

1. Відкриваємо інструмент «Схема даних». Для цього обираємо команду головного меню **Сервис->Схема данных** або натискаємо на кнопку «Схема данных» панелі інструментів.

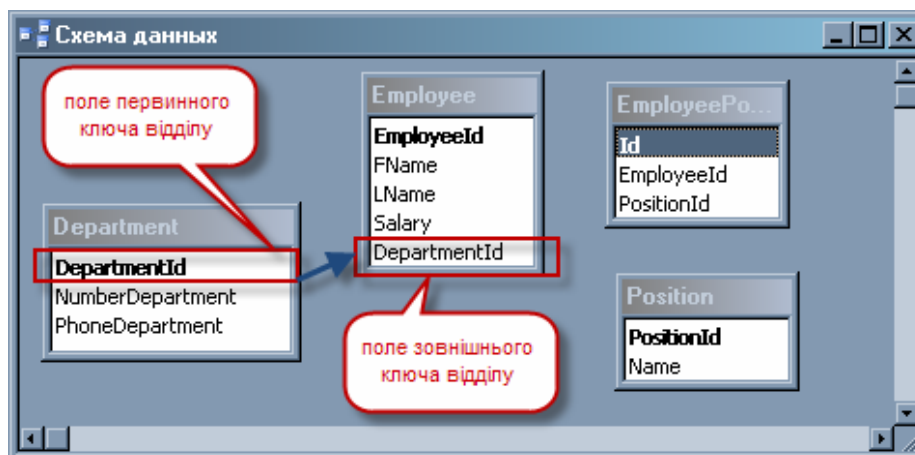


Даний інструмент дозволяє дуже зручно переглядати структуру всіх об'єктів бази даних, визначати тип зв'язків між ними і встановити параметри цілісності даних між таблицями.

2. Якщо в вікно схеми даних ще не присутні таблиці або необхідні нам об'єкти, слід відкрити діалогове вікно «Добавление таблицы» через контекстне меню схеми даних.

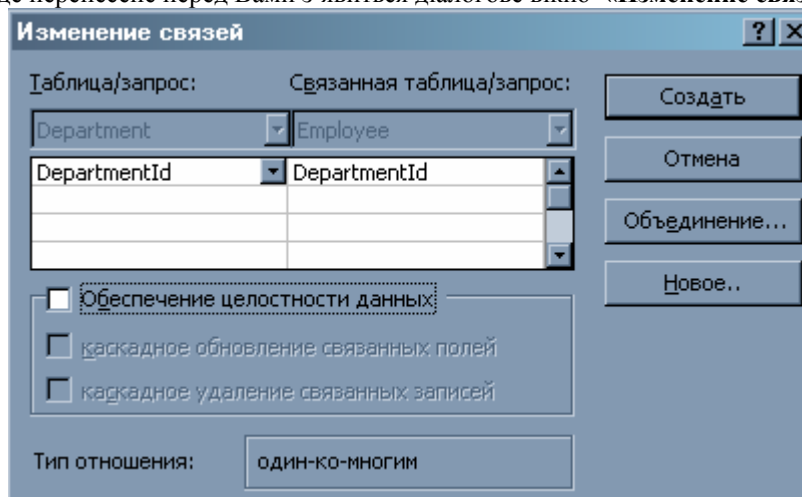


3. Створюємо зв'язи між доданими в схему таблицями. Для цього обираємо таблицю, в якій спільне поле виконує роль первинного ключа (для нас це таблиця **Employee**). Клікаємо на цьому полі і, утримуючи ліву кнопку миші, перетягуємо його на іншу таблицю, помістивши точно над тим полем, яке являється спільним (в таблиці **Department** це поле **DepartmentId**). Перемістивши на поле, відпускаємо кнопку миші. Аналогічні дії пророблюємо і для інших таблиць.

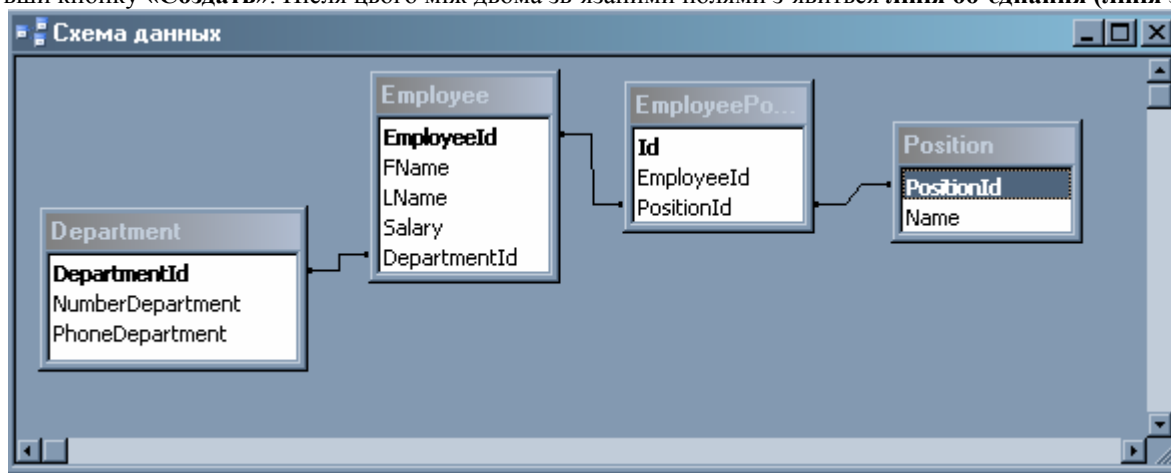




4. Після того як поле буде перенесене перед Вами з'явиться діалогове вікно «Изменение связей».



В цьому вікні представлено декілька параметрів. В верхній частині вікна відображені імена зв'язуємих таблиць і їх спільні поля. Також доступна опція «Обеспечение целостности данных», за допомогою якої забезпечується коректність зв'язку між таблицями. Впевнившись в тому, що параметри задані вірно, встановіть зв'язок між таблицями натиснувши кнопку «Создать». Після цього між двома зв'язаними полями з'явиться **лінія об'єднання (лінія зв'язку)**.



5. Та це ще не все. У Вас утворились прості зв'язки, які просто об'єднують дві таблиці. Подивимось уважно на діалогове вікно «Изменение связей», яке доречі можна повторно викликати за допомогою пункту контекстне меню (або подвійний клік на ньому) «Изменить связь» лінії зв'язку або за допомогою подвійного кліку по зв'язку. Дане вікно мистить ще дві опції, які дозволяють активізувати автоматичне виконання каскадного видалення і каскадного оновлення даних.

Каскадне оновлення зв'язаних полів говорить про те, що при будь-якій зміні даних первинного ключа в батьківській таблиці автоматично будуть оновлюватися значення в відповідному полі зв'язаної таблиці. Якщо ця опція не задана, змінити значення ключового поля первинної таблиці не вдасться. Важливо відмітити, що якщо ключове поле батьківської таблиці – це поле типу «Счетчик», то немає сенсу встановлювати дану опцію, оскільки значення поля лічильника змінити неможливо. Якщо ж поле первинного ключа таблиці приймає участь одразу в кількох зв'язках з іншими таблицями, то для коректної роботи бази даних дану опцію слід встановити для всіх зв'язаних таблиць.

Каскадне видалення зв'язаних записів говорить про те, що при видаленні записів в батьківській таблиці автоматично будуть видалятися всі відповідні записи в дочірній зв'язаній таблиці. Якщо дана опція не встановлена, то MS Access не дозволить видалити записи батьківської таблиці при наявності зв'язаних записів в дочірніх таблицях. Щоб вирішити цю проблему, доведеться спочатку видалити всі записи в дочірніх таблицях і лише потім – запис в головній таблиці.

5. Форматування даних при вставці в таблицю. Форматування даних при вставці в таблицю. Введення даних засобами СУБД

Отже, з принципами побудови таблиць розібрались, переходимо до ознайомлення з детальнішою інформацією про форматування окремих полів, тобто того, як будуть відображатись в них дані для користувача.



Для демонстрації розробимо таблицю **Person**, яка матиме наступну структуру:

| |
|-----------|
| Id |
| Name |
| Age |
| Married |
| Salary |
| E-mail |
| Photo |

Поле Id являється ключовим і створюється по стандартному принципу. Його налаштування ми вже розглядали сьогодні раніше (див. [розділ 1](#)).

Наступним полем буде Name, яке буде зберігати імена осіб та буде мати тип поля – «Текстовий». Як ми вже відзначали, для більш детальнішого налаштування поля використовується закладка «Общие», яка набула для текстового поля наступного вигляду:

| Имя поля | Тип данных | Описание |
|----------|------------|----------|
| ID | Счетчик | |
| Name | Текстовый | |

| Свойства поля | |
|------------------------|---------------------|
| Общие | Подстановка |
| Размер поля | 50 |
| Формат поля | |
| Маска ввода | |
| Подпись | Повне імя |
| Значение по умолчанию | "Невідомий" |
| Условие на значение | |
| Сообщение об ошибке | Невірно введене імя |
| Обязательное поле | Нет |
| Пустые строки | Нет |
| Индексированное поле | Нет |
| Сжатие Юникод | Да |
| Режим IME | Нет контроля |
| Режим предложенный IME | Нет |
| Смарт-теги | |

Red arrows and text in the image indicate: 'максимальна кількість символів' (maximum number of symbols) pointing to the length '50'; 'обов'язково вводити дані в поле?' (must enter data in the field?) pointing to 'Обязательное поле' (Required field) set to 'Нет' (No); 'допускаються пусті (NULL) значення?' (empty (NULL) values are allowed?) pointing to 'Пустые строки' (Empty strings) set to 'Нет' (No); 'зжимати дані в форматі Unicode при збереженні?' (compress data in Unicode format when saving?) pointing to 'Сжатие Юникод' (Unicode compression) set to 'Да' (Yes).

Опис характеристик такого поля продемонстрирован на рисунке, тому зупинимось лише на тих, які потребують детальнішого розгляду.

Таким чином, в полі «Значение поля по умолчанию» ми можемо ввести ті дані, які будуть в полі, якщо користувач його не заповнить. На це значення потрібно зважати, оскільки необов'язково всі Ваші дані в таблицях будуть заповнені.

В полі «Сообщение об ошибке» Ви можете задати текст, який буде розміщений на діалоговому вікні повідомлення про помилку у випадку невірної введення користувачем даних в поле.

Поле «Режим IME». IME (Input Method Editor) – це програма, яка дозволяє розпізнавати символи азійських алфавітів (китайського, корейського і японського), які вводяться за допомогою спеціальних кодів. Дане поле дозволяє вказати режим конвертування, яке відбувається при введенні даних в поле.

Поле «Режим предложенный IME» дозволяє визначити режим додатків IME, які будуть застосовуватись при введенні даних.

Поля «Формат поля», «Маска ввода» та «Условие на значение» пов'язані та схожі між собою, оскільки всі вони призначені для форматування даних при вводі. Відмінності між ними полягають в наступному:

1. **Формат поля** - дані перевіряються після їх вводу на відповідність певному правилу.
2. **Маска вводу** - аналогічно формату поля, але дані перевіряються ще під час введення.
3. **Умова на значення** - дані перевіряються також на етапі введення, але на відповідність певній умові. Наприклад, щоб користувач не ввів від'ємну ціну, або ж дату, яка пізніше поточної тощо.

Для форматування введення даних за допомогою поля «Формат поля» використовуються наступні управляючі символи:

1. **Для текстових і МЕМО-полів** використовуються наступні знаки:
 - ✓ > - все що вводиться, виводитиметься в верхньому регістрі.
 - ✓ < - все що вводиться, виводитиметься в нижньому регістрі.
 - ✓ @ - в поле потрібно обов'язково ввести звичайний символ або символ пропуску (space). Наприклад, при форматі поля (@@) @@-@@-@@ введене значення 0551234567 набуде наступного вигляду: (055) 123-45-67. Слід також враховувати, що заповнення здійснюється з молодшого розряду, інакше вставляється пустий символ. Наприклад, заданий формат поля вигляду "@@-@@-@@". При введенні в дане поле імені "Оля" ми отримаємо наступний результат " Оля".
 - ✓ & - необов'язковий текстовий символ (формат по замовчуванню). Так, як і символ @ відноситься до кожного символа поля.
2. **Числові та грошові поля.** Формат поля може складатися з чотирьох розділів, кожен з яких розділяється крапкою з комою (;):
 - 1-й розділ визначає формат позитивних чисел;
 - 2-й – негативних чисел;
 - 3-й - нулевих значень;
 - 4-й – пустих (NULL) полів.



Наприклад, +0,0;-0,0;0,0;"Не заповнене"

Формат даних полів може задаватись наступними символами:

| Символ | Опис |
|--------------------|---|
| . та , | Десятковий роздільник |
| , та " " (пропуск) | Роздільник груп розрядів |
| #, 0 | Прототип розряду: # - необов'язковий, 0 - обов'язковий |
| \$, грн., р.р. | Текстова константа. Наприклад, \$ ## ###,00 або ## ###,00 грн. |
| % | Відсотковий формат. Всі введені числа в дане поле будуть домножуватись на 100 та дописуватись символ відсотка (%). Наприклад, при форматі поля #0,00% вводиться значення 0,34567. В результаті ми отримаємо в полі значення вигляду 34,57% |
| E- або e- | Експоненційна нотація, в якій додатня степінь відображається без знака, а від'ємна з знаком. Наприклад, при форматі поля #,####E-00 та введенні числа 3456,7 Ви отримаєте значення вигляду: 3,4567E03 |
| E+ або e+ | Експоненційна нотація, в якій і додатня, і від'ємна степінь відображається з знаком. Наприклад, при форматі поля #,####E+00 та введенні числа 3456,7 Ви отримаєте значення вигляду: 3,4567E+03 |

3. Поля дати та часу. Для їх форматування використовуються наступні символи:

| | |
|--------------|---|
| (:) | - роздільник компонентів часу |
| (.), (/) | - роздільники компонентів дати |
| d | - день місяця (1-31) |
| dd | - день місяця (01-31) |
| ddd | - скорочена назва дня тижня (Пн. - Нд.) |
| dddd | - повна назва дня тижня (Понеділок - Неділя) |
| dddddd | - вбудований короткий формат дати |
| dddddd | - вбудований довгий формат дати |
| w | - порядковий номер дня тижня (1-7) |
| ww | - порядковий номер дня тижня в році (1-53) |
| m | - порядковий номер місяця (1-12) |
| mm | - порядковий номер місяця (01-12) |
| mmm | - перші три літери місяця (Янв - Дек) |
| mmmm | - повна назва місяця (Январь - Декабрь) |
| q | - порядковий номер квартала в році |
| y | - номер дня в році (1-366) |
| yy | - рік без століття (01-99) |
| yyyy | - рік з століттям (1999) |
| h, hh | - години (1-24 / 01-24) |
| n, nn | - хвилини (1-59 / 01-59) |
| s, ss | - секунди (1-59 / 01-59) |
| AM/PM | - 12-годинний формат часу з позначенням часу до півдня літерами AM та після півдня – PM |
| am/pm | - аналогічно попередньому, але позначення буде прописними літерами: am, pm |

4. Логічний тип. Формат поля складається з трьох розділів, кожен з яких розділяється крапкою з комою (;):

- 1-й розділ є пустим;
- 2-й розділ призначений для відображення рядкового значення, що заміняє true;
- 3-й розділ призначений для відображення рядкового значення, що заміняє false.

Формат даних полів може задаватись наступними символами:

| Символ | Опис |
|------------|--|
| (Пропуск) | Виводить пропуск як текстову константу |
| \, "текст" | Всі символи в лапках або після символа (\) являються текстовою константою |
| ! | Вирівнювання вмісту по лівому краю |
| * | Вирівнювання вмісту по правому краю. Причому, символ після зірочки виступає в якості заповнювача |
| [колір] | Задає колір тексту: Красный, Зеленый, Синий, Черный, Белый, Пурпурный, Желтый, Бирюзовый |

Маска вводу може містити до трьох розділів, які також розділяються крапкою з комою:

1. Безпосередньо сама маска вводу.



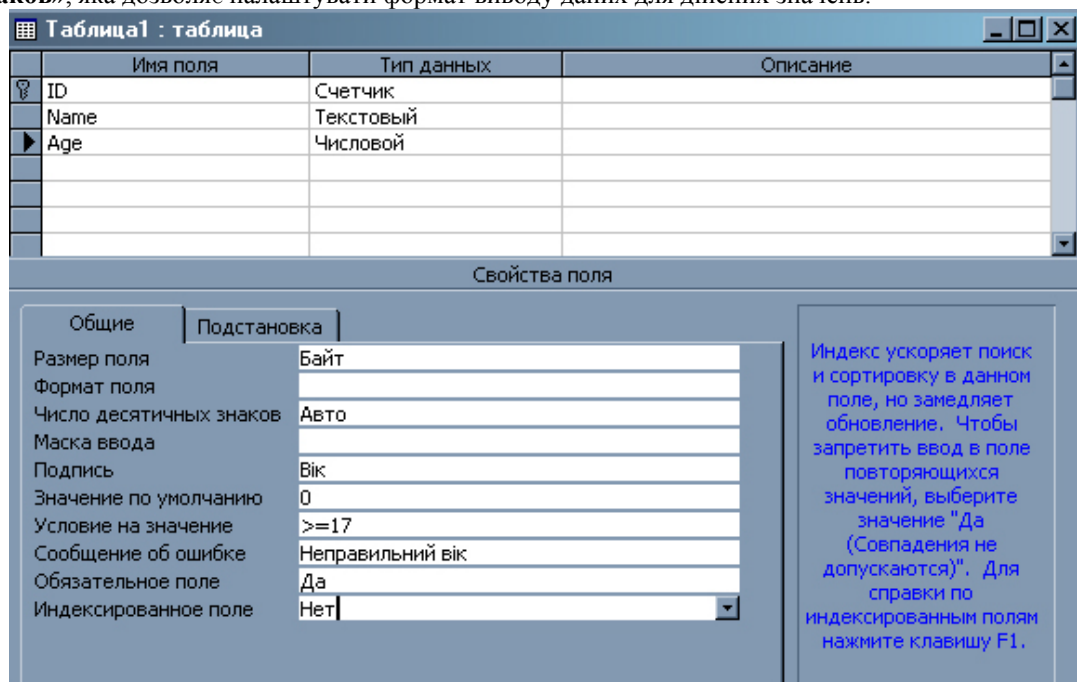
2. Розділ, який вказує на те чи буде СУБД MS Access зберігати при введенні проміжні символи в таблиці чи ні. Значення 0 (нуль) вказує на те, що текстові константи (наприклад, дефіси чи дужки в масці вводу телефонних номерів) будуть зберігатись разом з даними; значення 1 або пусте значення розряду вказує на те, що зберігаються лише дані.
3. Розділ, який визначає символ, який використовується для відображення пустих позицій в масці вводу, тобто куди повинні поміщатись дані, введені користувачем. По замовчуванню використовується символ нижнього підкреслення (_).

Символи, що можуть використовуватись для створення маски вводу:

- ✓ 0 – обов'язкова цифра;
- ✓ 9 - необов'язкова цифра;
- ✓ # - необов'язкова цифра або символ;
- ✓ L – обов'язкова літера;
- ✓ ? – необов'язкова літера;
- ✓ A – обов'язкова цифра або літера;
- ✓ a – необов'язкова цифра або літера;
- ✓ & - будь-що і необов'язкове;
- ✓ C - будь-що і обов'язкове;
- ✓ ! - заповнення даних справа наліво;
- ✓ \, "текст" - всі символи в лапках або після символа (\) являються текстовою константою.

Щоб переглянути результат, якого вдалось досягнути шляхом налаштування даного поля, потрібно ввести в нього дані. Для цього потрібно перейти в режим таблиці за допомогою кнопки на панелі інструментів «Вид» (див. попередній урок).

Наступним додаємо поле, яке зберігатиме вік особи – **Age**. Вставимо для нього числовий тип поля. Закладка «Общие» набула дещо змінилась: деякі характеристики зникли, оскільки потреба в них відпала, і додалась додаткова – «Число десятичних знаків», яка дозволяє налаштувати формат виводу даних для дійсних значень.



| Имя поля | Тип данных | Описание |
|----------|------------|----------|
| ID | Счетчик | |
| Name | Текстовый | |
| Age | Числовой | |

| Свойства поля | |
|-------------------------|------------------|
| Общие | Подстановка |
| Размер поля | Байт |
| Формат поля | |
| Число десятичных знаков | Авто |
| Маска ввода | |
| Подпись | Вік |
| Значение по умолчанию | 0 |
| Условие на значение | >=17 |
| Сообщение об ошибке | Неправильний вік |
| Обязательное поле | Да |
| Индексированное поле | Нет |

Индекс ускоряет поиск и сортировку в данном поле, но замедляет обновление. Чтобы запретить ввод в поле повторяющихся значений, выберите значение "Да (Совпадения не допускаются)". Для справки по индексированным полям нажмите клавишу F1.



Поле, що містить інформацію про сімейний стан (**Married**) доречно зробити логічного типу, оскільки воно може приймати лише одне з двох значень: так чи ні, одружений чи неодружений.

| Имя поля | Тип данных | Описание |
|----------|------------|----------|
| ID | Счетчик | |
| Name | Текстовый | |
| Age | Числовой | |
| Married | Логический | |

Свойства поля

Общие Подстановка

Формат поля: Да/Нет

Подпись: Семейный стан

Значение по умолчанию: Нет

Условие на значение:

Сообщение об ошибке:

Обязательное поле: Нет

Индексированное поле: Нет

Формат вывода значений данного поля. Выберите стандартный формат или создайте новый. Для справки по форматам нажмите клавишу F1.

Якщо після налаштування даного поля перейти в режим таблиці, то в даному полі з'явився перемикач (checkbox), який контролює ввід необхідних даних (одружений чи ні).

| Код | Повне ім'я | Вік | Married |
|-----------|---------------|-----|-------------------------------------|
| 1 | Ольга Павлова | 18 | <input type="checkbox"/> |
| 2 | Иван Приступа | 27 | <input checked="" type="checkbox"/> |
| (Счетчик) | Невідомий | 0 | <input type="checkbox"/> |

Запись: 2 из 2

Але, якщо Вас не влаштовує поле з перемикачем, це можна змінити. Наприклад, надати користувачу можливість обирати необхідні дані з випадючого списку або ж вводити з клавіатури. Для того, щоб це зробити потрібно знову перейти в режим конструктора і перейти на закладку «Параметры».

На цій закладці є лише одне поле з характеристикою, що визначає тип елемента управління, який буде використовуватись для представлення даних, що містяться в даному полі.

| Имя поля | Тип данных | Описание |
|----------|------------|----------|
| ID | Счетчик | |
| Name | Текстовый | |
| Age | Числовой | |
| Married | Логический | |

Свойства поля

Общие Подстановка

Тип элемента управления: Флажок

Флажок

Поле

Поле со списком

Тип элемента управления для вывода этого поля в формах

Хоча варіантів дій – три, але для логічного типу даних будуть працювати лише 2 перших. Для встановлення своїх значень потрібно налаштувати «Формат поля», а при введенні даних вводити значення -1 (для істинного значення - true) та 0 (для false).



Person : таблица

| Имя поля | Тип данных | Описание |
|----------|------------|----------|
| Name | Текстовый | |
| Age | Числовой | |
| Married | Логический | |

Свойства поля

Общие Подстановка

Формат поля ;"так"[Зеленый];"ні"[Красный]

Подпись

Значение по умолчанию

Условие на значение

Сообщение об ошибке

Обязательное поле Нет

Индексированное поле Нет

Якщо Ви все ж хочете використати поле зі списком, тоді доведеться вдатись до хитрощів. По-перше, змінити тип поля на «Текстовый». Потім на закладці «Общие» встановити умову на значення наступного змісту: **"так" Or "ні"**. На закладці «Подстановка» при виборі елемента управління «Поле со списком» з'являться додаткові поля, за допомогою яких можна його налаштувати.

Общие Подстановка

Тип элемента управления Поле со списком

Тип источника строк Список значений

Источник строк так;ні

Присоединенный столбец 1

Число столбцов 1

Заглавия столбцов Нет

Ширина столбцов

Число строк списка 8

Ширина списка Авто

Ограничиться списком Нет

звідки брати дані для списку
самі дані для списку

Тип данных определяет значения, которые можно с
данных нажмите н

Самими основными характеристиками являются:

- **«Тип источника строк»**, в якому вказується тип джерела даних: інша таблиця чи запит, список полів чи список значень.
- **«Источник строк»** дозволяє конкретизувати дану інформацію: написати запит на вибірку даних або ж вказати набір значень, які будуть в випадковому списку.

Всі інші - допоміжні, тобто дозволяють власне налаштувати виведення даних: вказати кількість елементів списку, його ширину тощо.



Наступне поле - поле **DateOfBirth** (Дата народження) з типом поля **Дата/Час**. Нового в списку характеристик закладки «Общие» нічого немає, новизна заключається лише в зміні формату поля.

| Имя поля | Тип данных | Описание |
|-------------|------------|----------|
| Name | Текстовый | |
| Age | Числовой | |
| Married | Логический | |
| DateOfBirth | Дата/время | |

Свойства поля

Общие Подстановка

Формат поля: Краткий формат даты

Маска ввода:

Подпись: Дата народження

Значение по умолчанию: <Date()

Условие на значение: Невірно введена дата

Сообщение об ошибке: Нет

Обязательное поле: Нет

Индексированное поле: Нет

Тип данных определяет значения, которые можно сопоставить в этом

Формат поля: Краткий формат даты

Маска ввода:

Подпись:

Значение по умолчанию:

Условие на значение:

Сообщение об ошибке:

Обязательное поле:

Индексированное поле:

Краткий формат даты: 6/19/1994 17:34:2

Полный формат даты: Sunday, June 19,

Длинный формат даты: 19-Jun-94

Средний формат даты: 6/19/1994

Краткий формат даты: 17:34:23

Длинный формат времени: 5:34 PM

Средний формат времени: 17:34

Краткий формат времени:

Поле **Salary** (Оклад, заробітна плата) зробимо «Денежным» та від форматуємо його.

| Имя поля | Тип данных | Описание |
|-------------|------------|----------|
| Name | Текстовый | |
| Age | Числовой | |
| Married | Логический | |
| DateOfBirth | Дата/время | |
| Salary | Денежный | |

Свойства поля

Общие Подстановка

Формат поля: ## ##0,00"грн.";-## ##0,00"грн."

Число десятичных знаков: 2

Маска ввода:

Подпись: Оклад

Значение по умолчанию: 0

Условие на значение:

Сообщение об ошибке:

Обязательное поле: Да

Индексированное поле: Нет

Формат поля: ## ##0,00"грн.";-## ##0,00"грн."

Число десятичных знаков: Основной 3456.789

Маска ввода: Денежный \$3,456.79

Подпись: Евро €3,456.79

Значение по умолчанию: Фиксированный 3456.79

Условие на значение: С разделителями разрядов 3,456.79

Сообщение об ошибке: Процентный 123.00%

Обязательное поле: Экспоненциальный 3.46E+03

Индексированное поле:

Як видно з рисунка необхідного формату даних (для вказання величини грошового окладу в гривнях) в нас не знайшлося, але ми вже познайомились з тим як створити свій власний формат, отож створимо його власноруч.

Для поля, що містить **E-mail** адресу особи, якщо така в неї така є, краще зробити типу «Гіперссылка». Поэкспериментуємо з даним типом поля.

| Имя поля | Тип данных | Описание |
|-------------|-------------|----------|
| Age | Числовой | |
| Married | Логический | |
| DateOfBirth | Дата/время | |
| Salary | Денежный | |
| E-mail | Гиперссылка | |

Свойства поля

Общие Подстановка

Формат поля:

Подпись: E-mail

Значение по умолчанию:

Условие на значение:

Сообщение об ошибке:

Обязательное поле: Нет

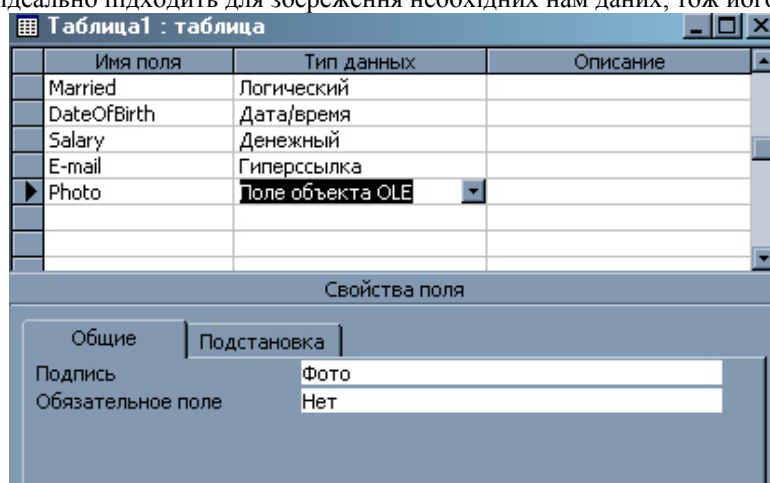
Пустые строки: Нет

Сжатие Юникод: Да

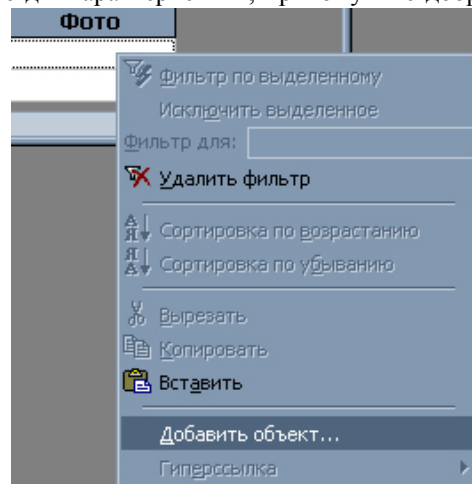
Имя поля может состоять из 64 знаков с учетом пробелов. Для справки по именам полей нажмите клавишу F1.



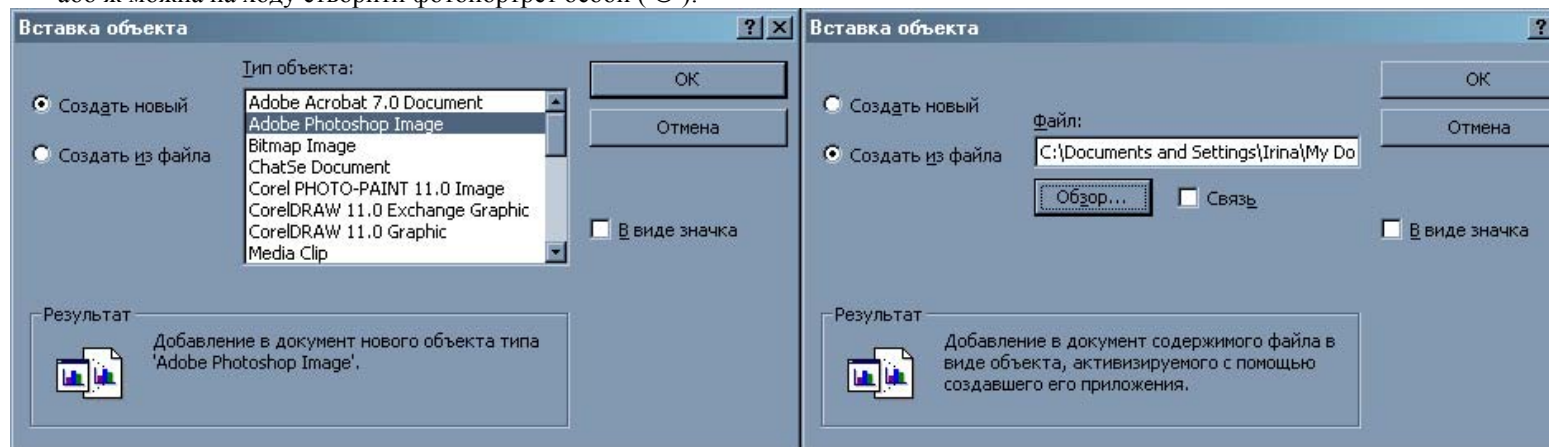
Поле Photo, що містить фото працівника. В нас залишилось лише два простих типи полів, які ми ще не розглядали детально: **поле MEMO** (характеристики якого нічим не відрізняються від характеристик звичайного текстового поля; єдина маленька відмінність – це максимальна кількість допустимих символів, про що вже говорилось на першому уроці) та **поле об'єкта OLE**. Останній ідеально підходить для збереження необхідних нам даних, тож його і оберемо.



Закладка «Общие» для такого типу полів не набула різноманітності, - тут лише дві характеристики, причому вже добре Вам знайомих. Після цього перейдемо в режим таблиці і поглянемо на дане поле з боку користувача: як в таке поле вводити дані? Для цього потрібно в контекстному меню даного поля обрати пункт «Добавить объект».



Після цього з'явиться діалогове вікно з можливістю вибору фотокартки або ж можна на ходу створити фотопортрет особи (☺):



Ви обираєте те, що Вам найбільше підходить та закриваєте дане вікно.

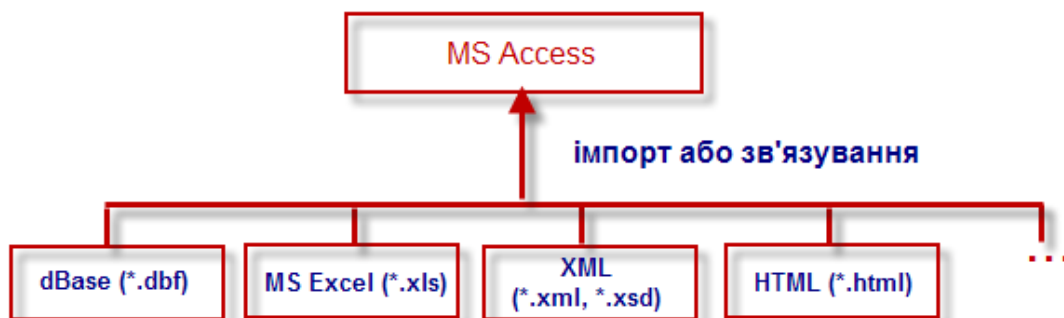
6. Імпорт та зв'язування таблиць

6.1. Імпорт електронних таблиць

Виникають різноманітні ситуації в житті і цілком імовірно, що виникне необхідність імпортувати дані з інших форматів в формат бази даних MS Access. Дуже часто така ситуація виникає, якщо створити інтерфейс до бази даних іншого формату. Зокрема підтримується імпортування даних з формату електронних таблиць та текстових документів. В цих випадках дані повинні бути вірно оформлені, інакше імпорт буде неможливий.



Схематично таку взаємодію можна зобразити наступним чином:



Для кращої наочності розглянемо імпортування таблиць з даних кількох форматів, а почнемо з отримання даних в форматі **MS Excel**. Для цього створимо невеличку БД з таблицями в MS Excel оформлених у відповідності до наступних вимог:

- в таблиці повинні бути тільки дані;
- кожна таблиця повинна розміщуватись на окремому листі;
- формули не копіюються, тому слід пам'ятати, що поля з ними будуть після імпорту пустими. Але для того, щоб такої ситуації не трапилось перед імпортом Ви можете перевести формули в числа.

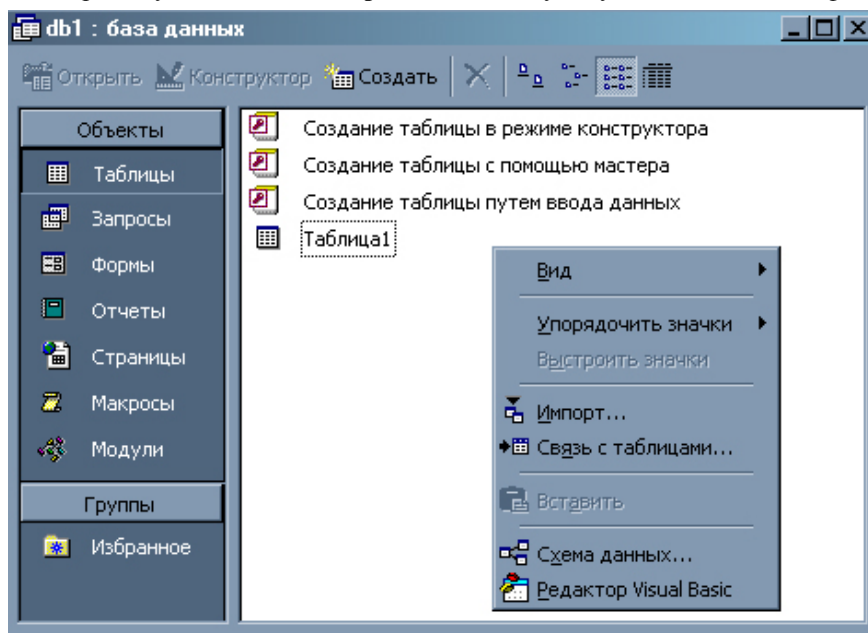
В результаті наші електронні таблиці для імпорту будуть мати наступний вигляд:

| | A | B | C | D |
|---|-----|---------------|-----|---|
| 1 | Код | Ім'я студента | Вік | |
| 2 | 1 | Вася | 17 | |
| 3 | 2 | Коля | 20 | |
| 4 | 3 | Оля | 18 | |
| 5 | 4 | Діма | 19 | |
| 6 | 5 | Інна | 20 | |
| 7 | | | | |
| 8 | | | | |
| 9 | | | | |

| | A | B | C | D | E | F |
|---|-----|------------------|-------|-----------|----------|---|
| 1 | Код | Назва товару | Ціна | Кількість | Вартість | |
| 2 | 1 | Молоко | 2.00 | 5 | 10.00 | |
| 3 | 2 | Хліб | 1.50 | 10 | 15.00 | |
| 4 | 3 | Шоколад 'Світоч' | 5.00 | 10 | 50.00 | |
| 5 | 4 | Торт 'Київський' | 17.00 | 6 | 102.00 | |
| 6 | 5 | Буряк | 1.00 | 78 | 78.00 | |
| 7 | 6 | Чіпси 'Lays' | 5.00 | 45 | 225.00 | |
| 8 | | | | | | |
| 9 | | | | | | |

Як видно на зображенні, наша електронна база даних містити два листки: на першому розміщується інформація про студентів, а на другому про товар: його назва, кількість, ціна за одиницю або кілограм та загальна вартість наявного товару тієї чи іншої категорії, обчислена за формулою.

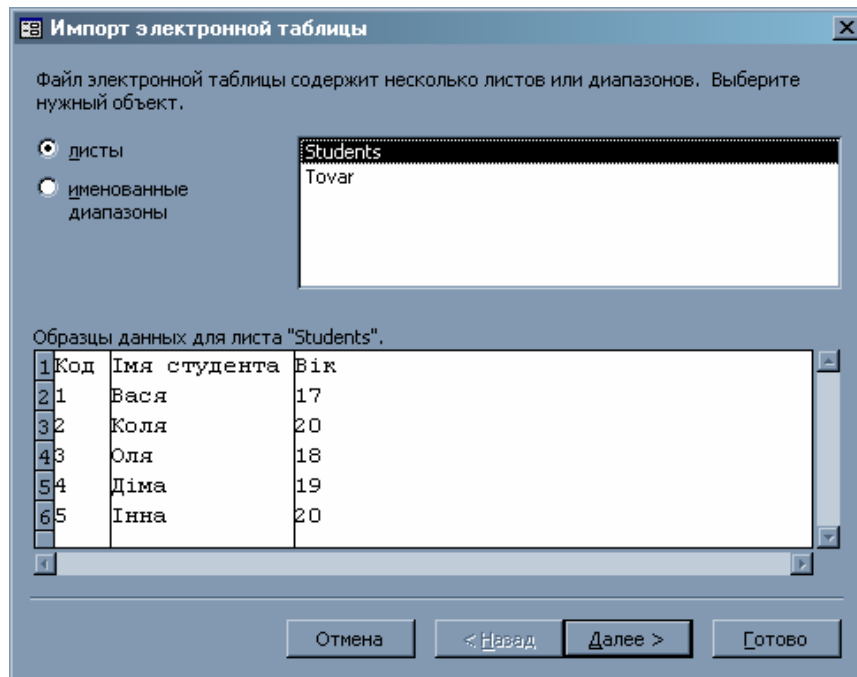
Імпортуємо її в базу даних MS Access. Для цього потрібно викликати контекстне меню в вікні об'єкта «Таблицы» та обрати пункт меню **«Імпорт»** та необхідну базу даних, тобто наш файл в форматі *.xls.



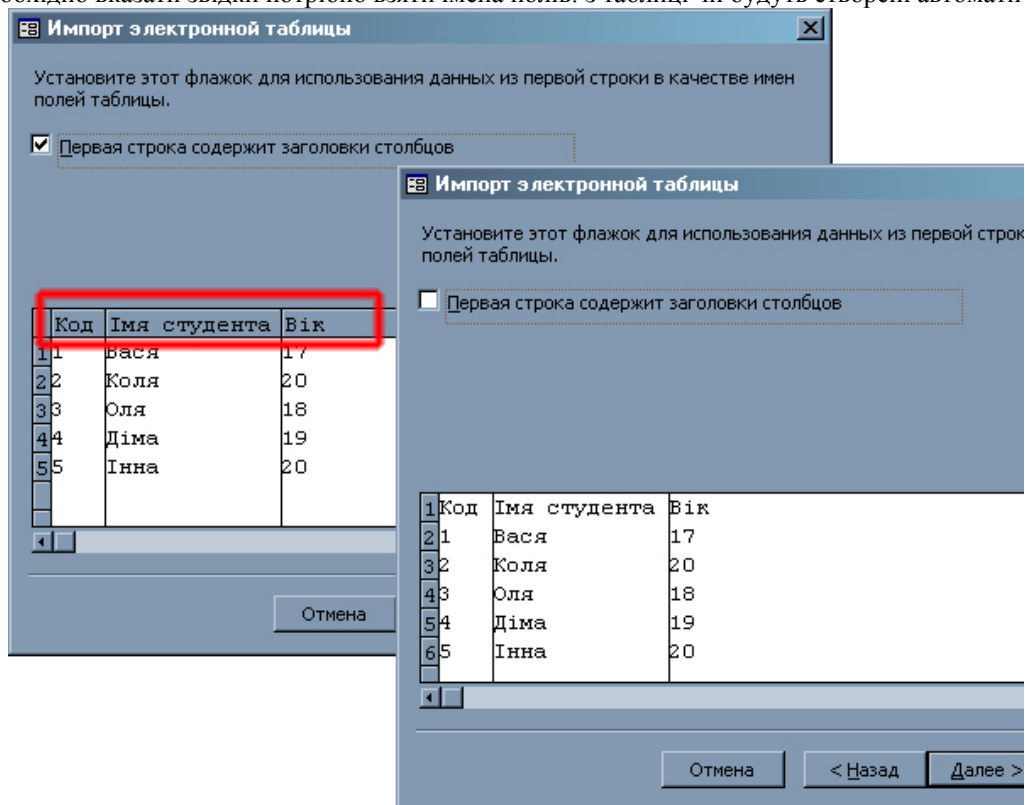
Варто вадмітити, що при імпорті таблиця поміщається в базу даних назавжди, тобто всі зміни залишаться лише в MDB-файлі, в файлі MS Excel вони відображатись не будуть.

Після обрання необхідного документу для імпорту MS Access запустить візард, який дозволить налаштувати імпортовані дані.

Оскільки, MS Access розуміє кожен листок нашого файлу як окрему таблицю БД, а ім'я листа нашого файлу як окрему таблицю БД, а ім'я листа таблиці є іменовані діапазони, то вони також будуть сприйняті як окремі таблиці.



Після цього необхідно вказати звідки потрібно взяти імена полів: з таблиці чи будуть створені автоматично.



На цьому обов'язкові кроки завершуються і Ви можете натискати на кнопку «Готово», але оскільки існує ще ряд кроків, то варто все ж їх розглянути на всяк випадок.



Отже, наступний крок дозволяє визначитись з тим, куди додавати імпортуємі дані: в вже існуючу таблицю (якщо звичайно формат є сумісним) чи створити нову.

Импорт электронной таблицы

Сохранение данных допускается в новой или в существующей таблице.

Данные необходимо сохранить:

☒ в новой таблице

☐ в существующей таблице: Таблица1

| | Код | Имя студента | Вик |
|---|-----|--------------|-----|
| 1 | 1 | Вася | 17 |
| 2 | 2 | Коля | 20 |
| 3 | 3 | Оля | 18 |
| 4 | 4 | Дима | 19 |
| 5 | 5 | Инна | 20 |

Отмена < Назад Далее > Готово

При выборе второго варианта и переходе до нового шага дані будуть імпортовані в обрану з списку таблицю. При выборе ж первого, - необходимо по шагам создать новую.

Импорт электронной таблицы

Имеется возможность описать каждое поле импорта. Выберите поле в нижней части окна и измените сведения в области "Описание поля".

Описание поля

имя поля: Код ☐ не импортировать (пропустить) поле

тип данных: Двойное с плавающей точкой

индекс: Да (Допускаются совпадения)

| | Код | Имя студента | Вик |
|---|-----|--------------|-----|
| 1 | 1 | Вася | 17 |
| 2 | 2 | Коля | 20 |
| 3 | 3 | Оля | 18 |
| 4 | 4 | Дима | 19 |
| 5 | 5 | Инна | 20 |

Отмена < Назад Далее > Готово



Далі потрібно описати кожне полі таблиці, після чого вказати її первинний ключ, тобто яке поле буде ключовим.

Импорт электронной таблицы

Рекомендуется задать ключевое поле в новой таблице. Ключ используется для однозначного определения каждой записи таблицы и позволяет ускорить обработку данных.

☐ автоматически создать ключ
☒ определить ключ:
☐ не создавать ключ

Код
Код
Имя студента
Вік

| Код | Имя студента | Вік |
|-----|--------------|-----|
| 1 | Вася | 17 |
| 2 | Коля | 20 |
| 3 | Оля | 18 |
| 4 | Діма | 19 |
| 5 | Інна | 20 |

Отмена < Назад Далее > Готово

В самом конце вводимую новую название таблицы, если Вы не соглашаетесь с предложенной, и с уверенностью завершаете работу мастера.

Импорт электронной таблицы

Указаны все сведения, необходимые для импорта данных.

Импорт в таблицу:
students

☐ Проанализировать таблицу после импорта данных.
☐ Вывести справку после завершения работы мастера.

Отмена < Назад Далее > Готово

Работу завершено. Таблица импортирована.

6.2. Импорт текстовых файлов

Наступным разглянемо импорт текстовых файлов, поскольку такие возможности даёт много программ. При импорте текстовых файлов выделяют несколько понятий:

1. **Сепаратор (раздільник)** - это символ, который отделяет одно поле от другого.
2. **Обмежувач** - это символ, в середине которого находятся поля.

Сам импорт (и экспорт также) может осуществляться в одном из **двух** вариантов:

1. Фиксированной (статической) ширины;
2. С сепараторами

Отже, начнём. Создадим новую таблицу с значениями в текстовом файле. Для этого воспользуемся многофункциональным текстовым редактором «Блокнот» (☺).

DataBase.txt - Notepad

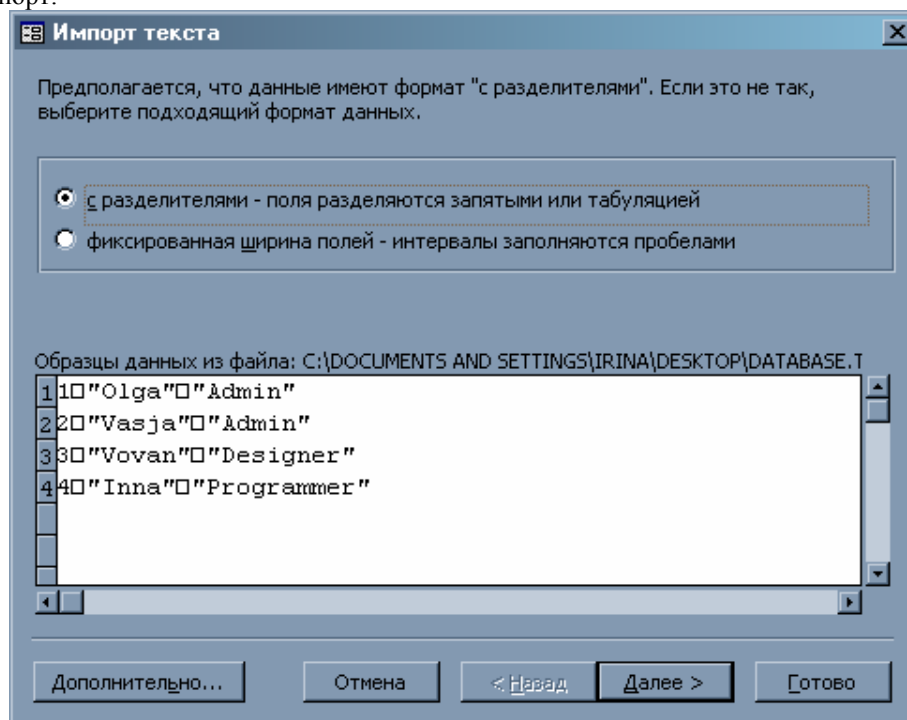
File Edit Format View Help

```

1      "Olga"  "Admin"
2      "Vasja" "Admin"
3      "Vovan" "Designer"
4      "Inna"  "Programmer"
  
```

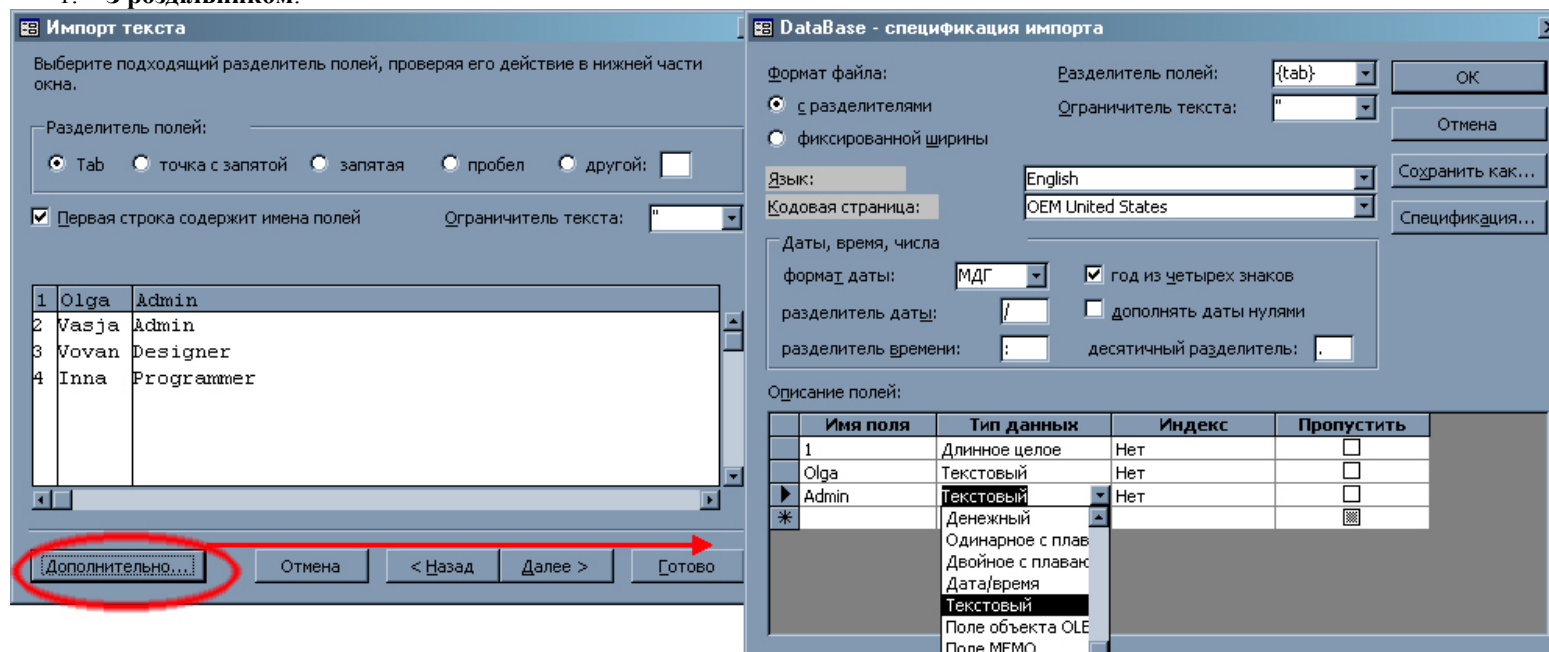


Після цього імпортуємо створений файл, обравши пункт меню «Імпорт». На першому кроці необхідно вказати як необхідно здійснити імпорт:



Різниця в наступних діях буде видна лише на наступних кроках, тож спробуємо і один, і другий спосіб:

1. З роздільником:





2. З фіксованою шириною полів:

Импорт текста

Предлагается следующее разделение полей с данными. Если предложение мастера не соответствует данным, настройте разделение полей вручную.

Линии со стрелками отмечают места разделения полей.

Чтобы создать разделитель, щелкните нужное положение.

Чтобы удалить разделитель, дважды щелкните его.

Чтобы переместить разделитель, перетащите его в нужное положение.

10 20 30 40 50

1 "Olga" "Admin"

2 "Vasja" "Admin"

3 "Vovan" "Designer"

4 "Inna" "Programmer"

Дополнительно... Отмена < Назад Далее > Готово

DataBase - спецификация импорта

Формат файла: Разделитель полей: {tab} OK

☐ с разделителями

☒ фиксированной ширины

Ограничитель текста: " "

Отмена

Сохранить как...

Спецификация...

Язык: English

Кодовая страница: OEM United States

Даты, время, числа

формат даты: МДГ ☒ год из четырех знаков

разделитель даты: / ☐ дополнять даты нулями

разделитель времени: : десятичный разделитель: .

Описание полей:

| Имя поля | Тип данных | Начало | Ширина | Индекс | Пропу |
|----------|------------|--------|--------|--------|--------------------------|
| Поле1 | Текстовый | 1 | 22 | Нет | <input type="checkbox"/> |

Зупинимось на створенні полів з роздільником та йдемо далі. А далі нам пропонують один з варіантів збереження імпортуємої інформації: чи створити нову таблицю чи помістити нові дані в вже існуючу. Обираємо для збереження нову таблицю, оскільки поки що в нашій БД немає таблиці з схожою структурою:

Импорт текста

Сохранение данных допускается в новой или в существующей таблице.

Данные необходимо сохранить:

☒ в новой таблице

☐ в существующей таблице:

Наступні кроки Вам вже знайомі: спочатку описуємо кожне поле імпорту, потім задаємо первинний ключ і завершуємо роботу над імпортом наших даних:

Импорт текста

Имеется возможность описать каждое поле импорта. Выберите поле в нижней области и измените сведения в области "Описание поля".

Описание поля

имя поля: Поле1 ☐ не импортировать (пропустить)

тип данных: Длинное целое

индекс: Да (Совпадения не допускаются)

Поле2 Поле3

1 Olga Admin

2 Vasja Admin

3 Vovan Designer

4 Inna Programmer

Дополнительно... Отмена < Назад Далее > Готово

Импорт текста

Рекомендуется задать ключевое поле в новой таблице. Ключ используется для однозначного определения каждой записи таблицы и позволяет ускорить обработку данных.

☐ автоматически создать ключ

☒ определить ключ: Поле1

☐ не создавать ключ

Поле1

Поле2 Поле3

1 Olga Admin

2 Vasja Admin

3 Vovan Designer

4 Inna Programmer

Дополнительно... Отмена < Назад Далее > Готово

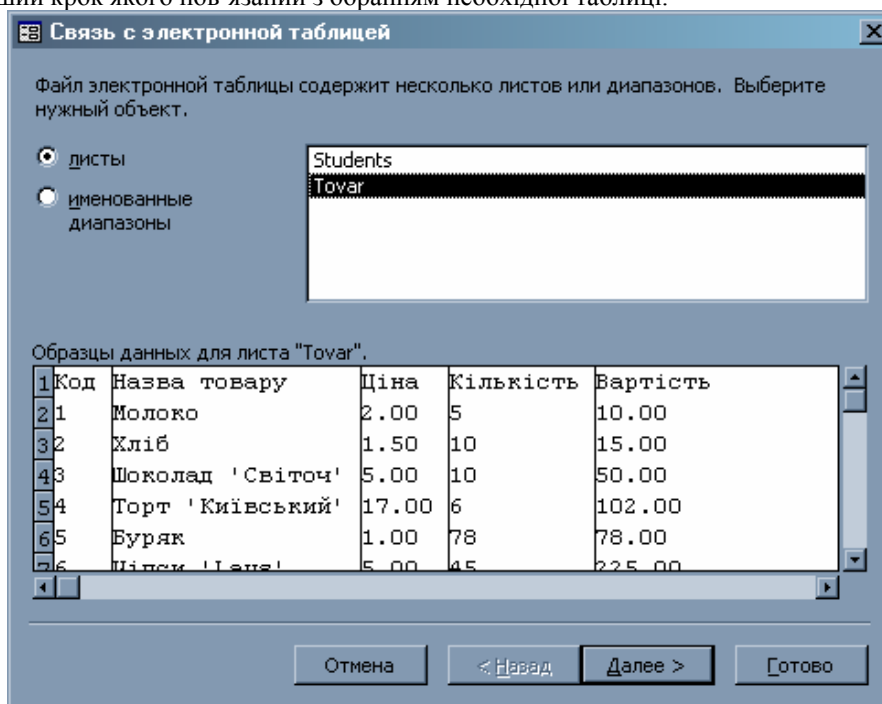
6.3. Зв'язування з таблицями інших баз даних

Крім імпортування існує і інша можливість роботи з базами даних, організованих в інший формат (відмінний від MS Access), - це зв'язок з таблицями. При зв'язуванні таблиці зміни, які Ви будете здійснювати в MDB-файлі будуть

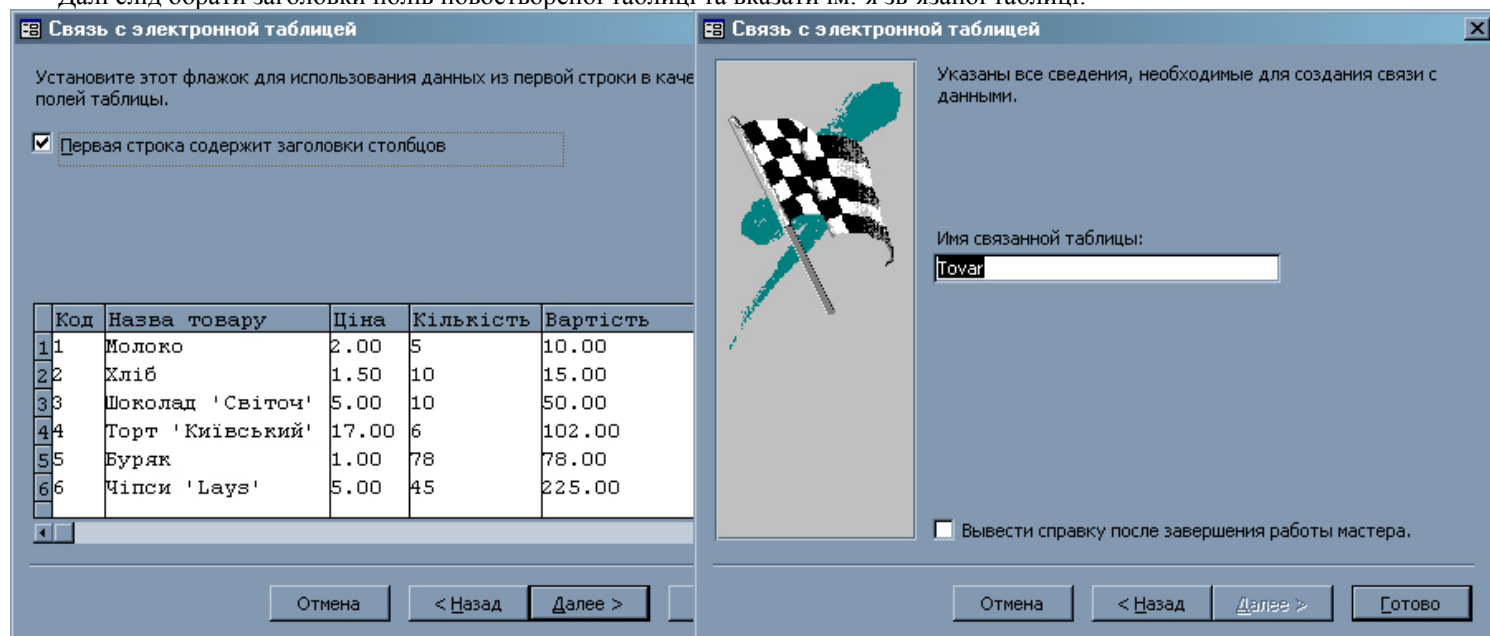


відображатись і в зв'язаній базі, та навпаки. Цей варіант поширеніший від імпорту, оскільки MS Access неефективно працює з даними великого об'єму, але дозволяє створити для них зручний користувацький інтерфейс, не знаючи при цьому мови програмування.

Зв'язування на етапі реалізації нічим суттєво не відрізняється від імпортування, різниця лише в кінцевому результаті. Для демонстрації зв'язування візьмемо другий лист бази даних формату MS Excel, створеної раніше. Отже, для зв'язування в контекстному меню вікна бази даних обираємо пункт «Связь с таблицей» і необхідний документ. Після цього запускається візард, перший крок якого пов'язаний з обранням необхідної таблиці.

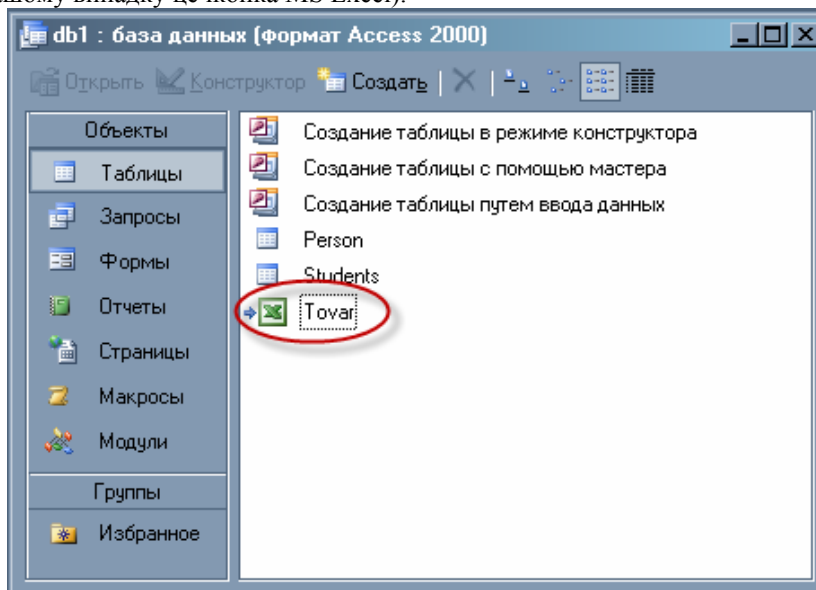


Далі слід обрати заголовки полів новоствореної таблиці та вказати ім.'я зв'язаної таблиці:





Перейдемо в вікно нашої бази даних. Як видно з зображення, зв'язана таблиця відображається з новою спеціальною піктограмою з стрілкою, яка розміщується зліва від назви об'єкта бази даних. Саме ця стрілка в піктограмі дозволяє відрізнити зв'язану таблицю в базі даних MS Access від звичайної статичної таблиці. Крім того, піктограма також вказує на тип зв'язуваних даних (в нашому випадку це іконка MS Excel).



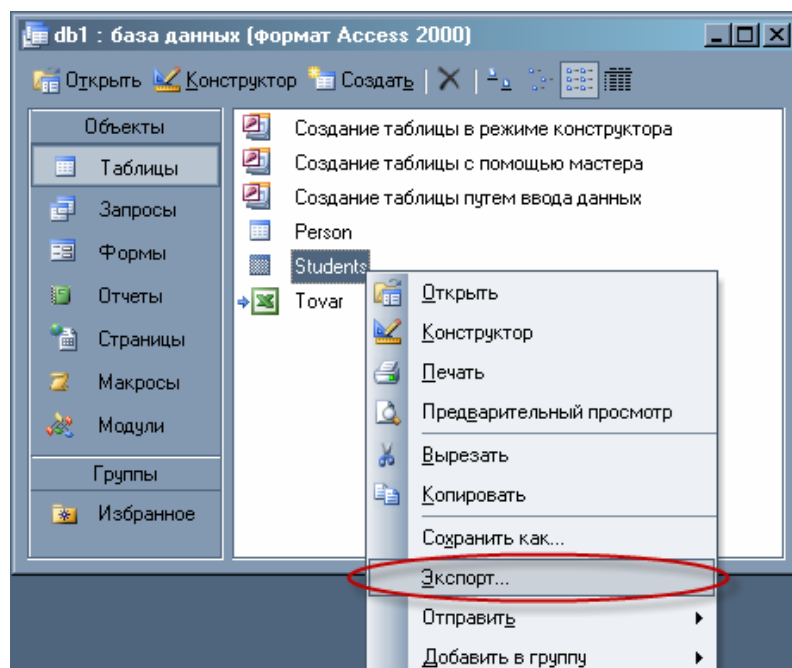
Та це ще не все. Існує ряд особливостей роботи з зв'язаними даними. Наприклад, структура зовнішньої таблиці, визначеної в іншій програмі, залишається незмінною (неможливо перевпорядкувати, видалити або додати нові поля). Однак доступні для налаштувань властивості таблиці, які визначають спосіб відображення даних на екрані, оскільки на вхідну таблицю такі операції ніяк не впливають.

Підсумовуючи відмітимо, **коли ж саме віддати перевагу зв'язуванню**:

- ✓ Коли існує розподілена база даних, яка розміщується на сервері і необхідно написати до неї користувацький інтерфейс через MS Access. В такому випадку Ваша база даних буде розділена на дві частини: **серверну**, де зберігаються всі таблиці, та **клієнтську**, де розміщуються форми, звіти тощо. Копія клієнтської БД надається кожному користувачу і зберігається в його локальній системі, а доступ забезпечується завдяки зв'язуванню.
- ✓ В випадку, коли розмір бази даних, що зберігається на сервері дуже великий.

7. Экспорт базы данных MS Access

Крім імпортування даних з баз даних, збережених в інших форматах MS Access підтримує можливість експорту даних в іншу базу даних типу MS Access або ж в інші формати. Але, слід пам'ятати, що експортувати можна лише окремі об'єкти, наприклад, таблиці. Експорт цілої бази даних в MS Access не передбачений.

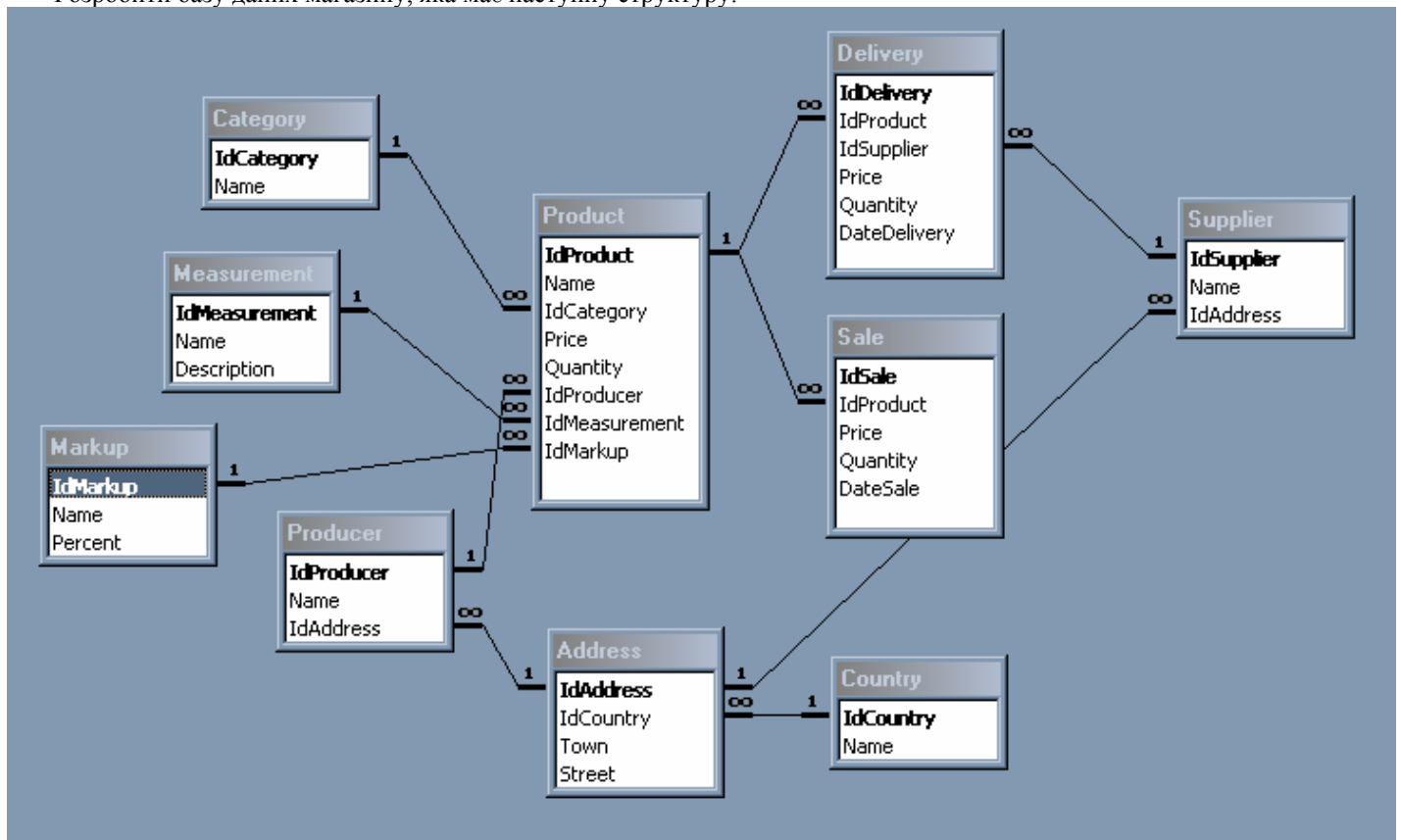


Для того, щоб здійснити експорт даних, наприклад, в формат електронних таблиць MS Excel достатньо обрати в контекстному меню необхідної таблиці пункт **“Експорт”** та вказати необхідний формат для експорту.



8. Домашнє завдання

Розробити базу даних магазину, яка має наступну структуру:



Коротка характеристика таблиць:

- **Product** – містить інформацію про продукт;
- **Delivery** – поставка товарів;
- **Supplier** – інформація про постачальника;
- **Sale** – продаж товарів;
- **Address** – повна адреса;
- **Country** – список країн;
- **Producer** – інформація про виробників товарів;
- **Markup** – інформація про знижки на продукцію: їх назва та відсоток. Наприклад, сезонна знижка, акція тощо;
- **Measurement** – одиниці виміру: скорочена назва та опис;
- **Category** – категорія товару.