Мета роботи: ознайомитися зі структурою об'єктів баз даних Microsoft SQL Server 2005 та отримати навики розроблення проекту бази даних за допомогою інструментарію Management Studio.

1. Основні відомості про Microsoft SQL Server

Microsoft SQL Server - комерційна система керування базами даних, яка входить у трійку лідерів на ринку баз даних (судячи по об'єму продаж, IBM Informix займає перше місце, Oracle знаходиться дуже близько, а третє місце посідає Microsoft SQL Server). Базовий код MS SQL Server (до версії 7.0) ґрунтувався на коді Sybase SQL Server, і це дало можливість швидко вийти на ринок баз даних, де вже конкурували IBM, Oracle і, згодом, Sybase. Microsoft, Sybase та Ashton-Tate об' єдналися для створення та випуску першої версії програми, яка фактично була еквівалентом Sybase SQL Server 3.0 для Unix. У 1992 році була випущена **версія 4.2,** яка входила до складу операційної системи Microsoft OS/2. Одночасно з виходом Windows NT 3.1 з' явився реліз **MS SQL Server 4.21** для цієї операційної системи. Версія **MS SQL Server 6.0** створена була виключно для архітектури NT, і є самостійним релізом Microsoft. На цьому етапі компанії Sybase та Microsoft розійшлися, та займалися створенням власних моделей баз даних. **Версія 7.0** була першим сервером баз даних зі справжнім графічним інтерфейсом адміністрування, а весь програмний код був переписаний (для уникнення порушень авторських прав щодо Sybase).

Таблиця 1.1. Версії Microsoft SQL Server

Версія	Рік	Повна назва	Кодове ім'я
1.0 (OS/2)	1989	SQL Server 1.0	-
4.2	1992	SQL Server 4.2	-
4.21 (Win NT)	1993	SQL Server 4.21	-
6.0	1995	SQL Server 6.0	SQL95
6.5	1996	SQL Server 6.5	Hydra
7.0	1998	SQL Server 7.0	Sphinx
–	1999	SQL Server 7.0 OLAP	Plato
8.0	2000	SQL Server 2000 (32-бітна версія)	Shiloh
8.0	2003	SQL Server 2000 (64-бітна версія)	Liberty
9.0	2005	SQL Server 2005	Yukon
10.0	2008	SQL Server 2008	Katmai

Великої популярності набула **MS SQL Server 2000,** як особливо стабільна та надійна система з мінімальними вимогами до ресурсів. До появи чергової версії, вона протрималася на ринку цілих шість років. Ця версія вже підтримувала XML (Extensible Markup Language, розширювана мова розмітки), одночасне функціонування багатьох копій сервера; була тісно інтегрована з Windows 2000, та автоматично реєструвалася в Active Directory; під керуванням Windows 2000 Datacenter могла використовувати до 32 процесорів та 64 Гбайт оперативної пам'яті; передбачено можливості масштабування (розподіл даних по декількох серверах) та покращена швидкодія.

Для наступної версії, **MS SQL Server 2005,** основним завданням ставилося покращити можливості масштабування та підвищення швидкодії (секціонування для таблиць та індексів, відключення індексів, збільшена кількість з 16 до 50 екземплярів SQL Server на одному комп' ютері, відкладене видалення та перебудова великих об'єктів, динамічні представлення й т.д.). Введені нові можливості системи безпеки (вбудовані засоби шифрування даних, розширені можливості роботи з логінами SQL Server, розділення користувачів та схем), системи реплікації, забезпечення безвідмовності (можливість створення, зміни та видалення індексів в оперативному режимі, виділене адміністративне підключення, дзеркальне відображення баз даних, контрольні суми для перевірки цілісності сторінок баз даних), введено підтримку XML як типу даних.

Для **MS SQL Server 2008** було поставлено завдання зробити керування даними самоналаштовуваним, самоорганізованим та самопідтримуваним. Ця версія також включає підтримку структу-рованих і напівструктурованих даних, у тому числі цифрові медіа-формати для зображень, звуків, відео та інших мультимедійних даних. Ключовим нововведенням MS SQL Server 2008 є розвинені засоби керування ресурсами, що дають можливість ефективно керувати та розподіляти робоче навантаження за допомогою відстеження рівня завантаження процесора та об' єму пам'яті. Введено підтримку просторових даних.

Редакції MS SQL Server 2005.

Усього є 5 редакцій MS SQL Server 2005, дві з них постачаються в 32- та 64-розрядних версіях. У всіх редакціях присутні компоненти для встановлення як на сервері, так і на робочій станції. Вибір редакції залежить від апаратних та фінансових можливостей, та потреб замовника.

Таблиця 1.2. Порівняння редакцій Microsoft SQL Server 2005

Редакція	Ціна*
Enterprise Edition (32- та 64- розрядні версії). Найповніша версія продукту з підвищеною продуктивністю та розширеним набором функцій. Здатна підтримувати тисячі підключень та баз даних, що вимірюються в терабайтах. Для досягнення такого рівня продуктивності необхідні й відповідні комп'ютери – як мінімум, з 16-ма двоядерними процесорами, 32 Гбайтами пам'яті та потужними мережевими платами. Встановлюється лише на серверні операційні системи починаючи з Windows 2000 Server.	$ 24 700
Standard Edition (32- та 64- розрядні версії). Має всі основні функції сервера: служби інтеграції та аналізу, Web-служби, дзеркальне відображення баз даних та кластеризацію. Підтримує лише 4 процесори, однак не має обмежень на пам'ять. У такій конфігурації сервер здатний обслуговувати 500 одночасних підключень та терабайтну базу даних. Однак на відміну від Enterprise Edition, ця версія не може здійснювати ряд операцій (з індексами, дзеркальне резервне копіювання, додавання оперативної пам'яті й т.п.) в оперативному режимі, тобто без відключення користувачів чи без відключення сервера. Також відсутнє секціювання таблиць та індексів	$ 6 060
Workgroup Edition (32-розрядна версія). Ця редакція є ще більш спрощеною в порівнянні зі Standard Edition. Підтримує 2 процесори та 3 Гбайти пам'яті, однак не має обмежень на розмір бази даних. У такій конфігурації сервер здатний обслуговувати до 100 активних користувачів. На відміну від Standard Edition, не підтримується кластеризація та дзеркальне відображення баз даних, відсутня служба бізнес-аналізу, служба інтеграції.	$ 3 900
Developer Edition (32- та 64- розрядні версії). Включає всі можливості Enterprise Edition, але ліцензується лише для розроблення програмного забезпечення та тестування. Запускається в операційних системах, призначених для робочих станцій, наприклад, Windows NT Workstation, Windows 2000 Prof. та Windows XP Prof.	$ 50
Express Edition (32-розрядна версія). Повноцінна версія ядра SQL Server, що призначена для використання з певним програмним забезпеченням у якості клієнтської чи основної серверної бази даних. Підтримує 1 процесор, 1 Гбайт пам'яті, обмеження розміру бази даних становить 4 Гбайти.	Безкошт.

* Ціни приблизні та наведені для порівняння вартості ліцензій різних редакцій із розрахунку на один процесор.

2. Проектування бази даних 2.1. Створення бази даних.

Основним графічним інструментом проектування баз даних в MS SQL Server 2005 є компонент Management Studio (Express). При запуску цієї програми відобразиться діалогове вікно **Connect to Server** (з'єднання із сервером) (рис. 2.1).

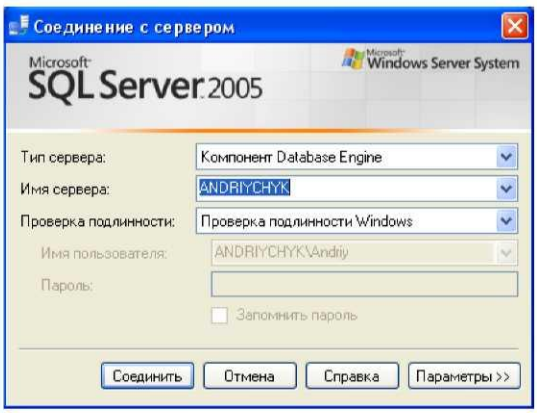


Рис. 2.1. Діалогове вікно Connect to Server

Поле **Server type** (тип сервера) дає можливість вибрати одну з декількох підсистем SQL Server (ядро бази даних чи відповідна служба), у яку повинен зайти користувач.

У полі **Server name** (ім'я сервера) відображається ім'я сервера, до якого необхідно підключитися. Як правило, з'єднання встановлюється з екземпляром за замовчуванням. Якщо необхідно з'єднатися з іншим екземпляром, тоді слід вибрати його у випадаючому списку. Якщо ж підключення до даного екземпляра здійснюється вперше, тоді в цьому випадаючому списку треба вибрати опцію <Browse for more> (огляд далі), та в наступному діалоговому вікні вибрати необхідний екземпляр.

У полі **Authentication** (перевірка автентичності) може бути вибраний один із двох варіантів перевірки автентичності користувача -**Windows Authentication** (перевірка автентичності Windows) та **SQL Server Authentication** (перевірка автентичності SQL Server). Перший варіант передбачає, що облікові записи користувачів Windows відображаються на облікові записи користувачів SQL Server. При спробі користувача зареєструватися в SQL Server, інформація про нього перевіряється в домені Windows та відображається на ролі, відповідно до облікового запису, а ролі вказують, які дозволяється користувачу виконувати дії. При використанні варіанту організації захисту на основі параметру SQL Server Authentication повністю ігноруються права, надані користувачу в мережі, а беруться до уваги лише ті, що явно задані в системі SQL Server. Для цього користувач повинен задати ім' я облікового запису та пароль, що відносяться до SQL Server.

СТВОРЕННЯ З'ЄДНАННЯ:

1. У полі **Server type** (тип сервера) вибираємо значення Database Engine (рис. 2.1).
2. У полі **Server name** (ім'я сервера) залишаємо ім'я сервера за замовчуванням.
3. У полі **Authentication** (перевірка автентичності) вибираємо варіант Windows Authentication або SQL Server Authentication (вводимо ім'я користувача **sa** та його пароль).
4. Натискаємо кнопку **Connect** (з'єднати).

Після запуску програми Management Studio та розгортання вузлів Оглядача об'єктів вигляд буде схожим як на рис. 2.2.

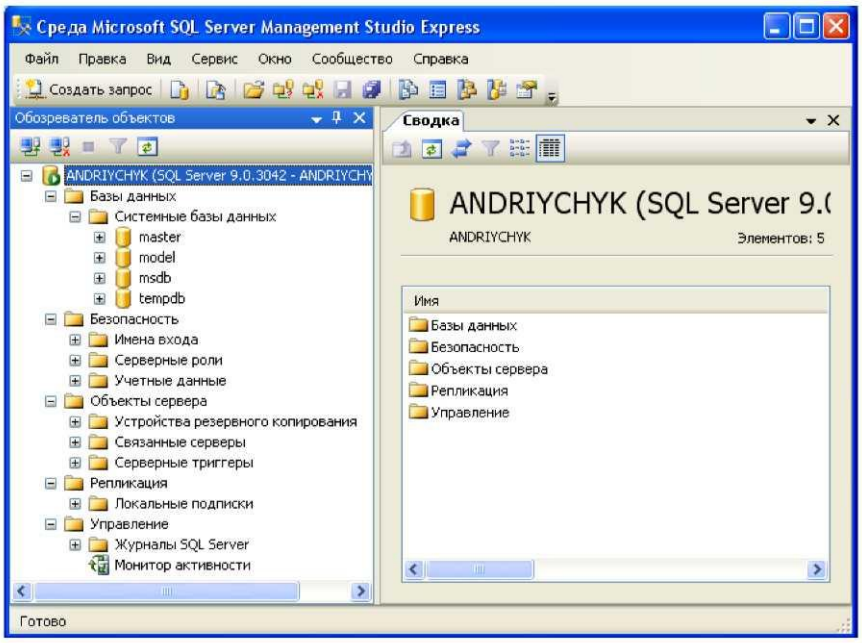


Рис. 2.2. Загальний вигляд Management Studio

Як бачимо, у вікні **Object Explorer** (оглядач об'єктів) у вузлі **Databases** (бази даних) присутня папка з набором системних баз даних: master - основна службова база даних усього сервера, в якій зберігаються налаштування його роботи, список баз даних на сервері з їхніми налаштуваннями, інформація про облікові записи користувачів для підключення до сервера і т.п.; model - шаблонна база даних для створення нових баз даних в SQL Server; msdb - база даних для зберігання службової інформації (служби SQL Server Agent, історії резервного копіювання тощо); tempdb - база даних для зберігання тимчасових таблиць та збережених процедур, проміжних даних при перебудові індексів. Також існують ще й додаткові системні бази даних, які є прихованими для очей користувачів та адміністраторів.

СТВОРЕННЯ БАЗИ ДАНИХ:

1. Натиснути правою кнопкою миші на вузлі **Databases** (бази даних) (рис. 2.2) та в контекстному меню вибрати команду New Database… (створити базу даних…).
2. У відкритому діалоговому вікні ввести ім'я для нової бази даних (рис. 2.3).
3. Вказати шлях на диску для розміщення файлів бази даних та журналу.
4. Натиснути кнопку **ОК**.

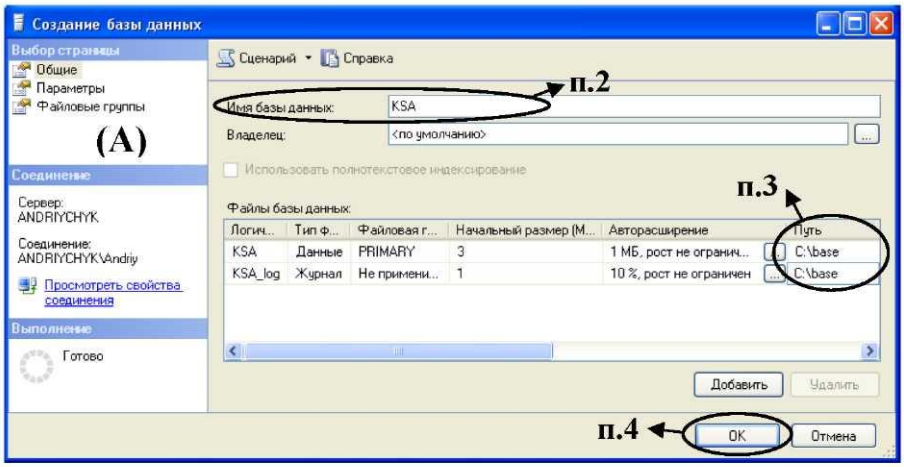


Рис. 2.3. Діалогове вікно для створення нової бази даних

При створенні бази даних, створюються файли самої бази даних та файли журналів транзакцій. Файл журналу транзакцій містить послідовний запис усіх змін, що вносяться в базу даних. Мінімальний набір файлів для будь-якої бази містить один файл для самої бази даних та один файл для журналу транзакцій. У кожній базі обов'язково є один основний файл, з розширенням *.mdf за замовчуванням. Вторинні файли бази даних мають розширення *.ndf, а для журналів транзакцій використовуються розширення *.ldf.

Наступний момент при створенні бази даних пов'язаний із вибором *розміру файлів бази даних та журналів транзакцій.* Звичайно, розмір файлів баз даних залежить від поставленої задачі. Можна одразу створити великі файли баз даних, або налаштувати для них режим автоматичного приросту. Як правило, рекомендують одразу створювати файли максимального розміру (або, принаймні, налаштувати автоматичний приріст великими частинами в декілька гігабайт). Такий підхід знижує фрагментацію файлів баз даних, підвищуючи тим самим продуктивність.

При створенні файлу бази даних можна також вказати, до якої файлової групи він буде відноситися. *Файлова група* - це спосіб організації файлів бази даних. За замовчуванням для будь-якої бази даних створюється файлова група PRIMARY, і всі створювані файли бази даних за замовчуванням будуть відноситися до неї. При необхідності можна створити додаткову файлову групу у вкладці **Filegroups** (файлові групи) (рис. 2.3 - А). Файлові групи використовуються при опти-мізації резервного копіювання. Наприклад, якщо базу даних можна умовно поділити на дві частини: користувацькі таблиці, які є невеликими і постійно змінюються, та таблиці довідника, що змінюються дуже рідко, але є великого розміру. У цьому випадку виконувати резервне копіювання важливо саме для користувацьких таблиць. Для цього створюють додаткову файлову групу, наприклад, USERS, далі створюють новий файл даних, наприклад, users.ndf, та призначають його до цієї групи. Файловій групі можна також призначати користувацькі таблиці та індекси. Тепер можна виконувати резервне копіювання окремих файлових груп із різним розкладом, наприклад, для файлової групи USERS проводити кожного дня, а для групи PRIMARY раз у місяць. Інші ситуації, для яких можна застосувати додаткові

файлові групи - ручний розподіл навантаження на дисковій підсистемі (наприклад, часто використовувані дані розмістити на швидкому диску, а інші на звичайному); розпаралелення запитів у дисковій підсистемі (розмістивши таблицю та її індекси в різних файлових групах) тощо.

Важливим моментом при створенні бази даних є *вибір режиму її відновлення.* Цей параметр вибирається у вкладці **Option** (параметри) (рис. 2.3 - А). Передбачено три режими відновлення бази даних:

• **Full** (режим повного протоколювання) - у журнал записується максимальна кількість операцій. Журнал транзакцій автоматично не обрізується. Цей режим забезпечує максимальні можливості відновлення (за рахунок зниження продуктивності). Лише в цьому режимі можна використовувати дзеркальне відображення баз даних.

• **Bulk-logged** (режим неповного протоколювання) - компромісне рішення між вимогами продуктивності та можливостями відновлення. У цьому режимі практично відключається запис у журнал для операцій масового вставлення, вставлення/зміни великих двійкових даних, операцій по створенню, перебудові та видаленню індексів.

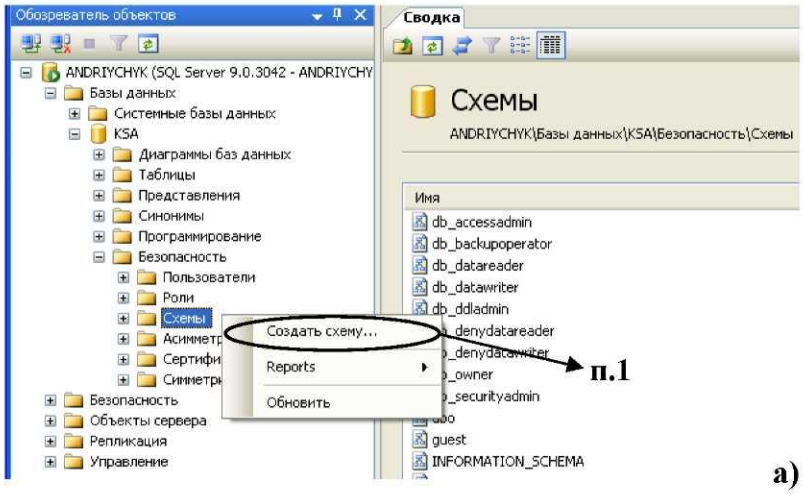
• **Simple** (проста модель відновлення) - максимальний виграш у продуктивності за рахунок можливостей відновлення. Мінімально протоколюються ті ж операції, що й у режимі **Bulk-logged,** а крім того, журнал транзакцій автоматично очищується. Однак, у цьому режимі не можливо використати журнал для відновлення бази даних .

У вкладці **Option** (параметри) (рис. 2.3 - А) знаходиться також ряд *додаткових параметрів,* інформацію про які можна знайти у [2].

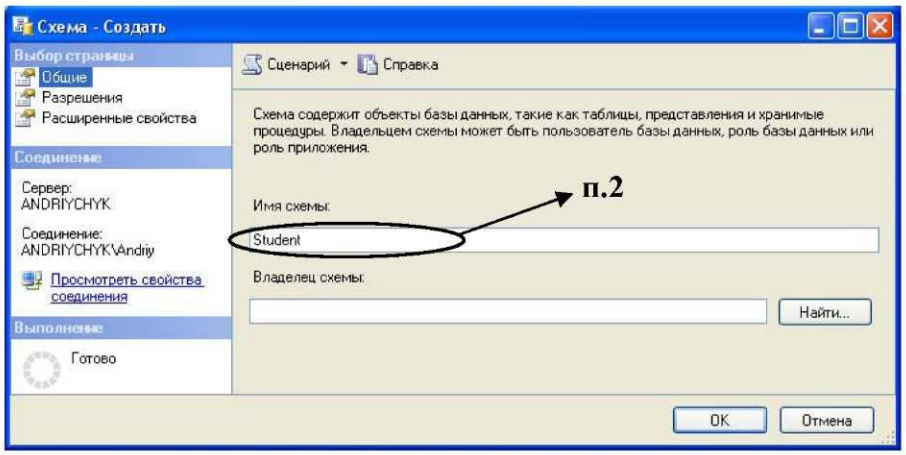
2.2. Створення схем бази даних.

Схеми - це контейнери для об'єктів, або, як їх ще називають, простори імен. Схеми використовують для спрощення керування даними та створення підмножини об' єктів, якими можна керувати як єдиним цілим. Такий підхід є особливо зручним, коли над проектуванням бази даних працюють декілька проектантів, і тоді виключається ситуація зі співпадіннями назв певних об' єктів (об'єкти кожного проектанта можуть бути прив'язані до певної схеми). Для звертання до об' єктів у схемах необхідно вказувати двоскладовий ідентифікатор у вигляді *SchemaName.ObjectName*. Якщо необхідно звертатися до об' єктів іншої бази даних, тоді слід використовувати трискладовий ідентифікатор у вигляді *DatabaseName .SchemaName .ObjectName*.

Якщо для об' єктів явно не вказують приналежність до конкретної схеми, тоді вони за замовчуванням відносяться до схеми **dbo.** Використання схеми за замовчуванням дає можливість звертатися до об' єктів за їхніми назвами (розширення схеми додається автоматично).



а)



б)

Рис. 2.4. Створення нової схеми

СТВОРЕННЯ СХЕМИ:

1. У панелі **Object Explorer** (оглядач об'єктів) для вибраної бази даних розкрити вузли, як на рис. 2.4а, та вибрати в контекстному меню команду New Schema… (створити схему).
2. У діалоговому вікні **Schema – New** (схема - створити) у полі **Schema name** (ім'я схеми) ввести ім'я, а поле **Schema owner** (власник схеми) можемо залишити пустим (рис. 2.4б).
3. Натиснути кнопку **ОК**.

Якщо явно не задати власника схеми, то за замовчуванням власником стане роль **dbo,** яка асоціюється з користувачем, що створив цю базу даних.

2.3. Створення таблиць бази даних.

Найближчим аналогом таблиці бази даних можна вважати лист електронної таблиці Microsoft Excel. При роботі з листом таблиці інформацію вводять у стрічки та стовпці. Також в електронних таблицях, як правило, є заголовки стовпців, згідно яких можна судити про вид інформації в них. Однак, на відміну від електронних таблиць, таблиці баз даних мають певні строгі обмеження відносно даних, які можна вводити в стовпці. В SQL Server така структура даних забезпечується за рахунок використання типів даних та властивостей стовпців. У таблиці 2.1 наводяться основні типи даних та їхні характеристики.

Таблиця 2.1. Вбудовані типи даних Microsoft SQL Server 2005

Тип даних	**Діапазон – опис**	**Розмір у байтах**
Цілочисельні типи		
bit	0, 1 або NULL	1 байт на кожні 8 стовпців
bigint	від -2^{63} до $2^{63}-1$	8
int	від -2^{31} (-2 147 483 648) до $2^{31}-1$ (2 147 483 647)	4
smallint	від -2^{15} (-32 768) до $2^{15}-1$ (32 767)	2
tinyint	від 0 до 255	1
Десяткові типи		
decimal (*p, s*); numeric (*p, s*)	від -10^{38} до $10^{68}-1$ де **p** – точність (макс. к-сть цифр у числі), **s** – степінь (к-сть цифр після коми). Наприклад, для числа 5 123,845: p=7; s=3;	від 5 до 17
Числові типи з плаваючою комою		
float	від $-2.23\cdot10^{308}$ до $2.23\cdot10^{308}$	8
real	від $-3.4\cdot10^{38}$ до $3.4\cdot10^{38}$	4
Фінансові типи		
money	від -922 337 203 685 477,5808 до 922 337 203 685 477,5807	8
smallmoney	від -214 748,3648 до 214 748,3647	4

Типи даних «Дата і час»		
datetime	З 01.01.1753р. до 31.12.9999р. з точністю до 3,33 мсек.	8
smalldatetime	З 01.01.1900р. до 06.06.2079р. з точністю до 1 хв.	4
Символьні / текстові типи		
char (*n*)	**n**= 1 ÷ 8 000 символів ANSI. Розмір фіксований. Недозаповнені поля доповнюються пробілами.	1 для кожного символу
varchar (*n*)	**n**= 1 ÷ 8 000 символів ANSI. Розмір змінний (пробілами не доповнюється).	1 для кожного символу + 2-байтний вказівник
varchar (max)	До 2 147 483 647 символів ANSI. Розмір змінний (до 2 Гбайт).	
text	До 2 147 483 647 символів ANSI. Розмір змінний (до 2 Гбайт).	1 для кожного символу
nchar (*n*)	**n**= 1 ÷ 4 000 символів UNICODE; розмір фіксований; недозаповнені поля доповнюються пробілами.	2 для кожного символу
nvarchar (*n*)	**n**= 1 ÷ 4 000 символів UNICODE; розмір змінний (пробілами не доповнюється)	2 для кожного символу + 2-байтний вказівник
nvarchar (max)	До 1 073 741 823 символів UNICODE; розмір змінний (до 2 Гбайт).	
ntext	До 1 073 741 823 символів UNICODE. Розмір змінний (до 2 Гбайт)	2 для кожного символу
Бінарні типи		
binary (*n*)	**n**= 1 ÷ 8000 байт; бінарні дані фіксованої довжини	1 – 8000
varbinary (*n*)	**n**= 1 ÷ 8000 байт; бінарні дані змінної довжини	1 – 8000 + 2-байтний вказівник
varbinary (max)	Бінарні дані змінної довжини; максимум до 2 147 483 647 байтів	До 2 Гб + 2-байтний вказівник
image	Бінарні дані змінної довжини; максимум до 2 147 483 647 байтів	До 2 Гб

Спеціальні типи		
timestamp	Унікальне значення в межах БД. що вказує на порядок змін; генерується автоматично	8
uniqueidentifer	Глобальний унікальний ідентифікатор (Globally Unique Identifier – GUID)	16
sql_variant	Дає можливість зберігати в одному стовпці дані різних типів (за винятком text, ntext, image, timestamp, xml, varchar(max), varbinary(max), nvarchar(max))	до 8000
xml	Символьне поле, що зберігає XML-дані	до 2 Гб

Числові типи із плаваючою комою **float** і **real** рекомендують використовувати лише тоді, коли вхідні дані виходять за границю діапазонів точних числових типів даних **decimal (numeric).**

Типи даних **money** і **smallmoney** призначені для зберігання грошових значень, однак, через їхнє обмеження в чотири десяткових розряди після коми, вони рідко використовуються у фінансових програмах. Дані програми вимагають виконання розрахунків із точністю до 6, 8, а іноді 12 знаків після коми, і тому, замість фінансових типів, використовують **decimal (numeric).**

Серед символьних типів є, на перший погляд, чимало однотипних, і відмінності між ними, хоча й ледве помітні, але дуже важливі. Тип даних **char** як в ANSI, так і в UNICODE має фіксований розмір. Тому для нього потрібно однаковий об' єм пам' яті при довільній кількості символів, що зберігаються в стовпці. Наприклад, стовпець із типом даних **char(30)** займає 30 байт пам'яті незалежно від того, зберігається в ньому один символ чи тридцять, а невикористана пам' ять доповнюється пробілами. А для зберігання кожного символу в стовпці з типом даних **varchar(30)** необхідно лише один байт.

Типи даних **text** і **ntext** призначені для зберігання великих масивів символьних даних. Однак ці типи мають ряд обмежень: до них неможна застосовувати оператор рівності чи операцію об' єднання; багато системних функцій не можуть працювати із цими типами даних. Через ці обмеження в SQL Server 2005 були введені типи даних **varchar(max)** і **nvarchar(max),** які об'єднують можливості попередніх, можуть зберігати до 2 Гб даних та не мають обмежень щодо їхнього використання з різними операціями та функціями.

Типи даних **binary/varbinary** переважно використовують для збереження групи невеликих файлів розміром 4-6 Кб, що містять різні дані в двійковому форматі. Тип даних **image** дає можливість зберігати

не лише фотографії, але й довільні документи Word, Excel, PDF тощо, довільні файли, розмір яких не перевищує 2 Гб. Для усунення обмежень, що накладаються на тип даних **image** при використанні з різними операціями та функціями, у SQL Server 2005 був введений тип даних **varbinary(max),** що теж

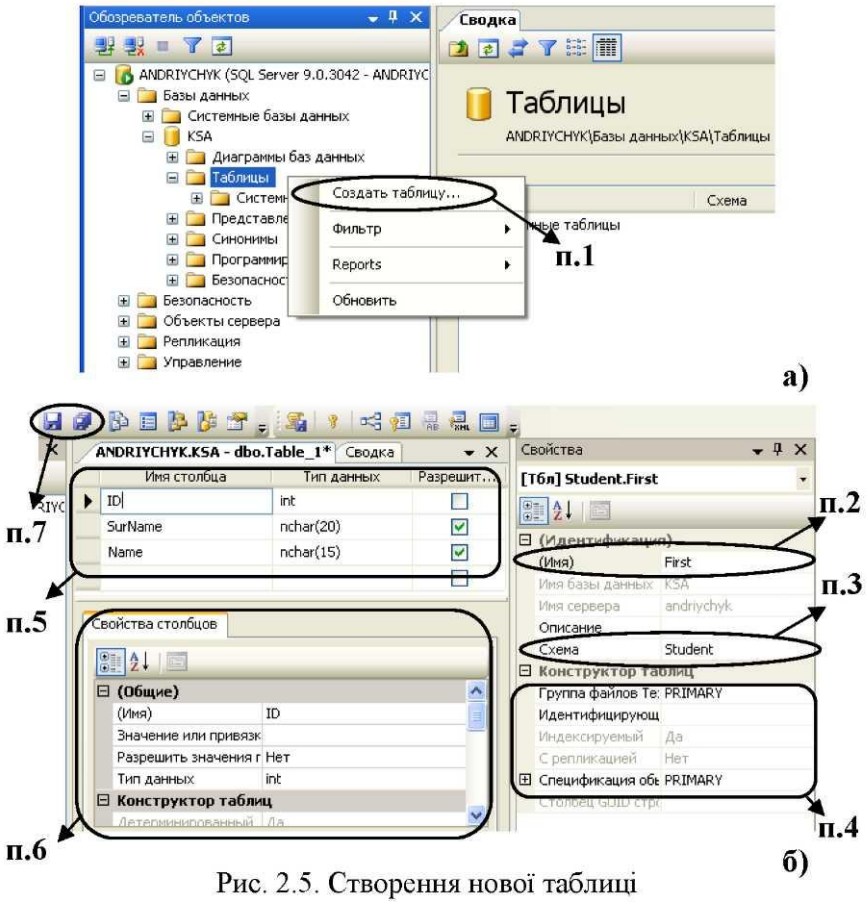


Рис. 2.5. Створення нової таблиці

СТВОРЕННЯ ТАБЛИЦІ:

1. У панелі **Object Explorer** (оглядач об'єктів) для вибраної бази даних розкрити вузли, як на рис. 2.5а, та в контекстному меню вузла **Tables** (таблиці) вибрати команду New Tables… (створити таблицю), після чого відкриється вікно **Table Designer** (конструктор таблиць) (2.5б).

дає можливість зберігати об'єм даних до 2 Гб.

2. Вписати назву таблиці в поле **Name** (ім'я), що розміщене на панелі **Properties** (Властивості).
3. Вибрати приналежність таблиці до певної схеми з випадаючого списку поля **Schema** (схема), що розміщене на панелі **Properties** (Властивості).
4. Вибрати файлові групи, у яких зберігатимуться дані цієї таблиці.
5. Сформувати стовпці таблиці, задаючи ім'я стовпця, його тип даних та дозвіл на NULL.
6. За допомогою вкладки **Column Properties** (властивості стовпців) задати значення властивостей стовпців таблиці.
7. Зберегти таблицю.

Приналежність до заданих файлових груп визначається окремо для звичайних даних і окремо для великих об' єктів:

• Для визначення місця зберігання звичайних даних розкрийте вузол розділу **Regular Data Space Specification** (специфікація звичайного простору даних) (рис. 2.5б, п.4) і у випадаючому списку **File or Partition Scheme Name** (ім'я групи файлів чи схеми розділів) виберіть необхідну *групу файлів.*

• Для визначення місця зберігання даних великих об' єктів виберіть необхідну *групу файлів* із випадаючого списку поля **Text/Image Filegroup** (група файлів Text/Image).

Спеціальна конструкція **null,** що означає «невідомий» чи «відсутній», дає можливість зберігати в стовпці пусті значення. **null** не є значенням, і тому не потребує пам'яті для свого збереження. Якщо ж дозвіл на **null** не встановлений, то це вимагає від користувача обов' язкового введення значення в зазначений стовпець.

Набір властивостей вкладки **Column Properties** (властивості стовпців)

1. **Name** (Ім'я) - ім'я стовпця.
2. **Allow Nulls** (Дозволити значення null) - дозволяє або забороняє значення **null.**
3. **Length** (Довжина) - вказує число символів, дозволених для символьних типів даних.
4. **Default Value or Binding** (значення або прив'язування за замовчуванням) - виводить значення за замовчуванням, яке встановлюється, коли в стовпець не було нічого введено. Значення за замовчуванням може бути введено у вигляді тексту або ж вибране з випадаючого списку, де відображаються всі глобальні значення за замовчуванням, що визначені в базі даних.
5. **Precision** (точність) - вказує максимальну кількість цифр у значеннях стовпця з типом **decimal.**
6. **Scale** (масштаб) - вказує максимальну кількість цифр справа від десяткової коми в значеннях стовпця з типом **decimal.**
7. **Computed Column Specification** (специфікація обчислюваного стовпця) - виводить інформацію про стовпець, що містить результат обчислень із використанням даних інших стовпців таблиці. У полі **(Formula)** [(формула)] задається вираз, згідно якого обчислюється значення для стовпця. В якості цього виразу може бути довільне коректне поєднання операторів, функцій, констант та значень інших стовпців цієї таблиці (оператори та функції наводяться в табл. 3.1). Поле **Is Persisted** (є збережува-ним у базі даних) вказує чи зберігаються результати обчислень у базі даних (значення **Yes** (так)), чи в базі зберігається лише формула (значення **No** (ні)), а самі значення обчислюються лише при звертанні до цього стовпця.
8. **Identity Specification** (специфікація ідентифікуючого стовпця) - виводить інформацію про стан стовпця в режимі автоматичного генерування унікальних числових ідентифікаторів стрічок. Для включення стовпця в режим ідентифікації необхідно встановити в його полі **(Is Identity)** [(є ідентифікуючим стовпцем)] значення **Yes** (так), у полі **Identity Increment** (початкове значення)

задати вихідне значення, а в **Identity Seed** (приріст) - значення інкременту. Вихідне значення - це початкове значення, що встановлюється для найпершої стрічки таблиці, а інкремент визначає, наскільки SQL Server збільшує (зменшує) цю початкову величину при генеруванні кожного наступного значення. Ця властивість доступна лише для цілочисельних типів, а для використання з типами **decimal** та **numeric** необхідно встановити нуль розрядів після коми.

9. **Collation** (таблиця сортування) - задає порядок сортування стовпця, який використовується SQL Server при сортуванні стрічок результатів запиту.

10.**RowGuid** (ідентифікатор GUID стрічки) - вказує на наявність глобальних унікальних ідентифікаторів стрічок; може бути застосованим лише для стовпців із типом даних **uniqueidentifer.** GUID формується на основі номера мережевої плати, унікальність якого гарантується виробниками в найближчі 100 років.

2.4. Створення ключів та реалізація обмежень бази даних.

Є декілька різних типів ключів:

- **Первинний** - містить первинний вказівник на стрічку в таблиці;
- **Альтернативний** - містить додаткові вказівники на стрічку в таблиці, довільні унікальні умови, що представляють один чи більшу кількість стовпців таблиці.
- **Зовнішній** - містить вказівники на ключі в інших таблицях. Первинні та альтернативні ключі - гібридні об' єкти: частково

індекси, а частково обмеження. Обмеження оголошують, що для об' єкта повинен бути істинним певний фактор, а для ключів це означає, що значення в таблиці повинні бути унікальними.

Індекси дають можливість організувати швидкий доступ до даних без пошуку по всій базі даних. Індекси керуються та зберігаються окремо від таблиць. Для побудови індексів SQL Server використовує структуру *B-дерева* (B-tree, Balanced-tree), що складається з кореня, проміжних вузлів та кінцевих вузлів (листків). Деревовидна структура дає можливість організувати швидкий та ефективний пошук, в іншому ж випадку, серверу довелось би почергово зчитувати кожну сторінку даних таблиці в пошуку потрібного запису. Проіндексованим може бути як окремий стовпець, так і сукупність вибраних стовпців.

За своєю структурою індекси поділяються:

- **Кластерний** індекс зберігає сторінки даних таблиці на рівні листків *B-дерева,* при цьому дані фізично впорядковані згідно ключа. *Для кожної таблиці можна визначити лише один кластерний індекс* (**!!!**). При створенні такого індексу відбувається фізичне сортування даних у відповідності до індексу, та перебудова всіх некластерних індексів. Як правило, кластерні індекси створюють для первинних ключів, хоча можуть бути створені й для довільного стовпця. Оптимальним варіантом вважають такий, коли індексовані значення

для кластерного ключа унікальні. Якщо ж значення не унікальні, тоді SQL Server створює додаткові ключі сортування для стрічок, що мають дублікати для основних ключів сортування.

- **Некластерний** індекс на рівні листків B-дерева містить вказівник на стрічку даних, що відповідає ключу в індексі. Якщо таблиця вже має кластерний індекс, тоді вказівник вказує на його ключ, а не на дані. Якщо ж кластерний індекс відсутній, тоді вказівник вказує на реальну стрічку даних. При створенні некластерного індексу SQL Server створює необхідні сторінки індексу, але не змінює фізичного розташування табличних даних, і не видаляє інші індекси таблиці. Кожна таблиця може мати до 249 некластерних індексів.

Вибір стовпців для індексування. Варто пам'ятати, що застосування індексів не тільки пришвидшує пошук необхідної інформації, але й, у свою чергу, вимагає накладних витрат: оперативної пам' яті, місця на диску, машинного часу. Кожного разу, коли відбуваються довільні зміни в індексованих стовпцях, також змінюється й індекс. При одиночній зміні цей час незначний, однак чим активніша система, тим більше це зачіпає продуктивність. У табл. 2.2 наводяться рекомендації щодо вибору стовпців для

Таблиця 2.2. Рекомендації по створенню індексів

Індексувати	Не індексувати
Таблиці з великою кількістю стрічок	Таблиці з невеликою кількістю стрічок
Стовпці, що часто використовуються в запитах	Стовпці, що рідко використовуються в запитах
Стовпці, що зберігають широкий діапазон значень і мають високу ймовірність бути вибраними в типовому запиті	Стовпці, що зберігають широкий діапазон значень і мають низьку ймовірність бути вибраними в типовому запиті
Стовпці, що використовуються в реченнях GROUP BY	Стовпці, що мають великий розмір у байтах
Стовпці, що використовуються в реченнях ORDER BY	Таблиці, де дані часто змінюються, але рідко зчитуються
Стовпці, що використовуються у з'єднанні таблиць	

Таблиця 2.3. Рекомендації по кластеризації індексів

Кластеризувати індекс для	Не кластеризувати індекс для
Первинних ключів, що часто використовуються при пошуку, наприклад, номера рахунків	Первинних ключів, що зберігають послідовні значення ідентифікаторів, наприклад, ідентифікаційних стовпців
Запитів, що повертають численні результуючі набори	

індексації, а у табл. 2.3 - відносно використання кластерних чи некластерних індексів.

Стовпців, що використовуються в численних запитах Стовпців із високою селективністю Стовпців, що використовуються в реченнях GROUP BY чи ORDER BY Стовпці, що використовуються у з'єднанні таблиць	Запитів, що повертають невеликі результуючі набори Зовнішніх ключів

Обмеження - це формулювання певних вимог до даних. Обмеження встановлюються на рівні стовпця чи таблиці та забезпечують відповідність даних визначеним правилам забезпечення їхньої цілісності.

Серед методів реалізації обмежень розрізняють такі:

- **PRIMARY KEY** - обмеження первинного ключа.
- **FOREIGN KEY** - обмеження зовнішнього ключа.
- **UNIQUE** - обмеження унікальності, або альтернативного ключа.
- **CHECK** - обмеження перевірки.
- **DEFAULT** - обмеження заданих за замовчуванням значень.
- **Правила.**

Також ще до цих методів відносять задані за замовчуванням значення на рівні таблиці, тригери та збережувані процедури.

Способи іменування ключів та обмежень. Імена ключів та обмежень у системі SQL Server можуть бути довільними, у межах загально дозволеного іменування. Система генерує імена таким чином: скорочена абревіатура у виді двох букв + назва таблиці + унікальне доповнення:

PK_TableName_Definition - первинний ключ;

IX_TableName_Defmition - унікальний ключ / індекс;

FK_TableName_Definition - зовнішній ключ;

CK_TableName_Definition - обмеження перевірки;

Таке позначення є прийнятним і зрозумілим. SQL Server як *Definition* генерує унікальне значення, а в графічній компоненті розробника Management Studio задаються порядкові номери. Проектанти баз даних, як правило, розшифровують його - вказують коротке формулювання його призначення, або вказують ім'я (імена) стовпця (стовпців), на який цей ключ чи обмеження поширюється.

2.4.1. Створення первинного ключа таблиці (PRIMARY KEY).

Первинні ключі являють собою унікальні ідентифікатори для кожної стрічки. SQL Server дає можливість визначити в якості первинного ключа будь-який стовпець чи групу стовпців, які повинні містити унікальні значення (наявність **null**-значень не дозволяється **!!!**). Найбільш імовірними кандидатами на цю роль є, як правило, ідентифікуючі стовпці. Таблиця може мати лише один первинний ключ. Крім того, якщо використовується складовий ключ із декількох стовпців, тоді значення всіх стовпців об' єднуються, для визначення унікальності стрічок.

Вибір первинного ключа - один із найважливіших виборів, що робиться для конкретної таблиці, оскільки первинний ключ буде мігрувати в інші

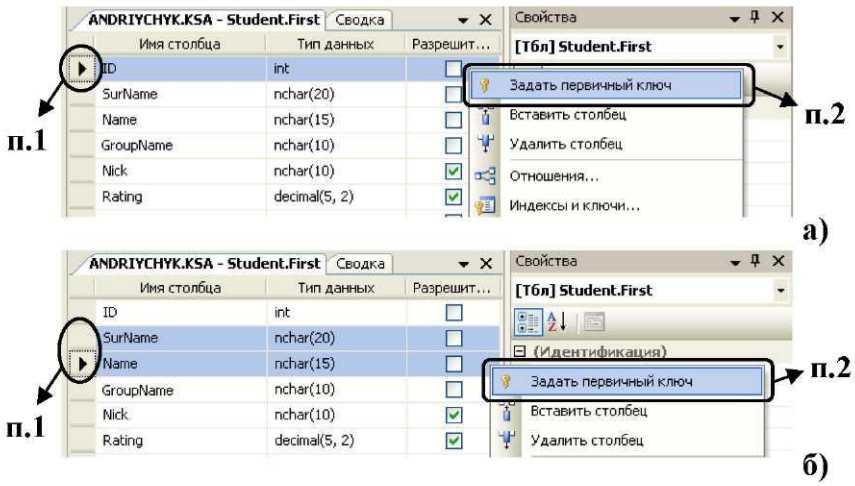


Рис. 2.6. Створення первинного ключа таблиці

СТВОРЕННЯ ПЕРВИННОГО КЛЮЧА:

1. У конструкторові таблиць вибрати стовпець (стовпці), який (які) необхідно використовувати в якості первинного ключа; для цього, утримуючи клавішу Ctrl, вибрати мишею ці стовпці, натиснувши зліва від їхнього імені (рис. 2.6).
2. Натиснути правою клавішею миші по вибраних стовпцях та в контекстному меню вибрати команду Set Primary Key (задати первинний ключ).
3. Зберегти зміни в таблиці.

таблиці як вказівник на конкретне значення.

Примітка: *при створенні первинного ключа SQL Server автоматично створює унікальний кластерний індекс, при умові, що не створено ще жодного іншого кластерного індексу, або тоді некластерний.*

Для первинного ключа іноді необхідно виконати налаштування певних параметрів (рис. 2.7).

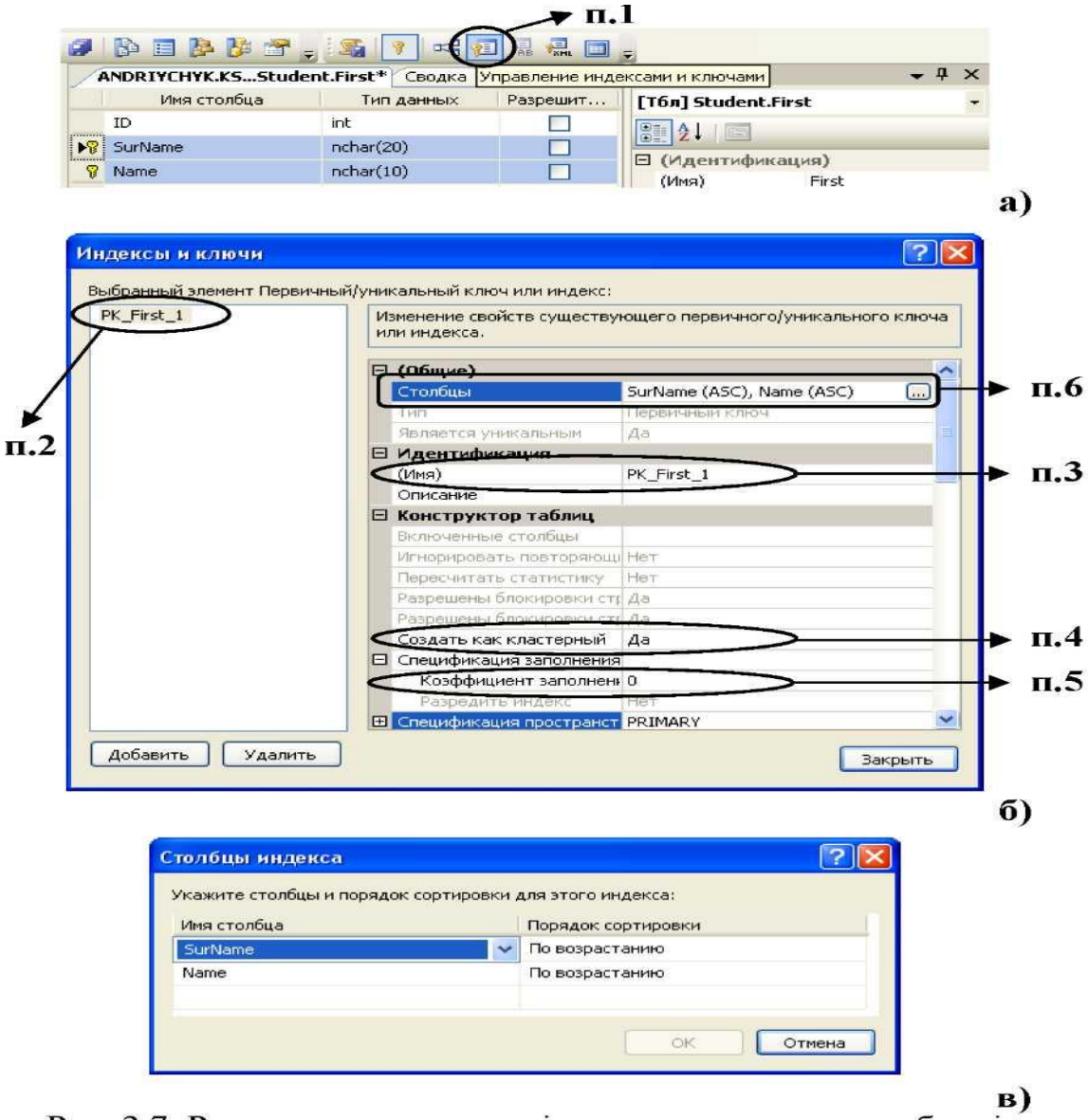


Рис. 2.7. Редагування параметрів первинного ключа таблиці

НАЛАШТУВАННЯ ПАРАМЕТРІВ ПЕРВИННОГО КЛЮЧА:

1. На панелі інструментів (рис. 2.7а) натиснути по іконці Manage Indexes and Keys (керування індексами та ключами).
2. У діалоговому вікні **Indexes/Keys** (індекси та ключі) у лівій колонці (рис. 2.7б) вибрати створений первинний ключ (крім нього можуть бути ще й інші ключі).
3. У правій частині діалогового вікна в полі **(Name)** [(ім'я)] скорегувати назву первинного ключа в більш зрозумілий контекст, наприклад, замість «1» вписати назву стовпця.
4. У полі **Create As Clustered** (створити як кластерний) вибрати необхідний варіант: кластерний чи некластерний.
5. У полі **Fill Factor** (фактор заповнення) при необхідності можна задати значення параметру заповнення.
6. У полі **Columns** (стовпці) вказуються вибрані стовпці для індексування та їхній порядок у індексі. Для внесення змін натиснути кнопку із трикрапкою, розташовану справа від поля; у результаті чого відобразиться діалогове вікно **Index Columns** (стовпці індексу) (рис. 2.7в). У лівій частині вікна за допомогою випадаючих списків поля **Column Name** (ім'я стовпця) можна змінити кількість, вид та порядок входження стовпців у первинний ключ. Для кожного стовпця справа від нього в полі **Sort Order** (порядок сортування) можна задати окремий порядок сортування для індексу як Ascending (по зростанню), або як Descending (по спаданню).
7. Закрити діалогове вікно та зберегти зміни в таблиці.

Фактор заповнення індексів визначає, який відсоток вільного місця на сторінках має зарезервувати система при створенні нового індексу. Якщо сторінки повністю заповненні, то при добавленні нових даних у таблицю системі необхідно їх розщеплювати, що у свою чергу знижує продуктивність. Значення, рівне 100, означає, що сторінки будуть повністю заповненими. У цьому випадку для збереження даних необхідно мінімум простору. Менше значення залишає більше вільного простору на сторінках, і, відповідно, знижується потреба у розбитті сторінок з даними в процесі росту індексів, але при цьому необхідно більше місця для збереження даних. Хоча зарезервований простір з часом теж заповниться, і при вставленні даних серверу все таки доведеться виконувати розщеплення сторінок; при необхідності перерозподілу даних, рекомендують заново створити індекс.

Встановлення фактору заповнення - це певний компроміс для продуктивності: при надто великому значенні, система буде повільніше виконувати операції додавання даних у таблицю, при надто низькому - можливе зниження продуктивності читання даних. Значення 100% рекомендують використовувати для статичних таблиць, що призначенні лише для читання.

За замовчуванням значення фактору заповнення встановлене в 0%, що вказує серверу на автоматичну оптимізацію заповнення індексів, будь-яке інше значення є дійсним відсотком заповнення. Дії системи при значенню 0% в певній мірі аналогічні як при значенню 100: SQL Server створює кластерні індекси із заповненими сторінками даних та некластерні індекси із заповненими сторінками у вершинах індексного дерева. Однак значення 0%, на відміну від 100, дає можливість росту на верхньому рівні дерева індексів. Тому

100% заповнення слід використовувати лише в таблицях, призначених тільки для читання, і в які не планується додавати дані.

Значення поля **Pad Index** (розрідити індекс) вказує на необхідність встановлення долі вільного простору, відповідно до значення фактору заповнення, у проміжних сторінках цього індексу.

2.4.2. Створення вторинного ключа або індексу таблиці (UNIQUE KEY / INDEX).

Під вторинним (альтернативним) ключем мають на увазі так зване обмеження UNIQUE, яке забезпечує унікальність даних. По суті це обмеження майже повністю відповідає обмеженню первинного ключа, оскільки вимагає наявності унікальних значень у всьому вказаному у ньому стовпці (чи комбінації стовпців) таблиці. При додаванні стрічки з дублюючим значенням для стовпця, для якого встановлено обмеження UNIQUE, SQL Server автоматично згенерує помилку та відхилить спробу введення такої стрічки.

Примітка: *при створенні вторинного ключа, аналогічно як і для первинного ключа, SQL Server автоматично створює унікальний некластерний індекс.*

З іншого боку, створення вручну окремого унікального індексу теж забезпечує унікальність даних для визначеного стовпця (стовпців) таблиці. Фактично, для забезпечення унікальності даних немає різниці між створенням обмеження UNIQUE та створенням унікального індексу. Ця відмінність полягає лише в їхній семантиці: унікальний ключ призначений для забезпечення обмежень на дані, а індекси призначені для прискорення доступу до даних.

Основною суттєвою відмінністю унікального ключа від унікального індексу є те, що він усе таки ключ, і на нього можуть посилатися зовнішні ключі (FOREIGN KEY) для забезпечення цілісності даних. Особливістю ж індексів є те, що вони можуть бути як унікальними, так і не унікальними.

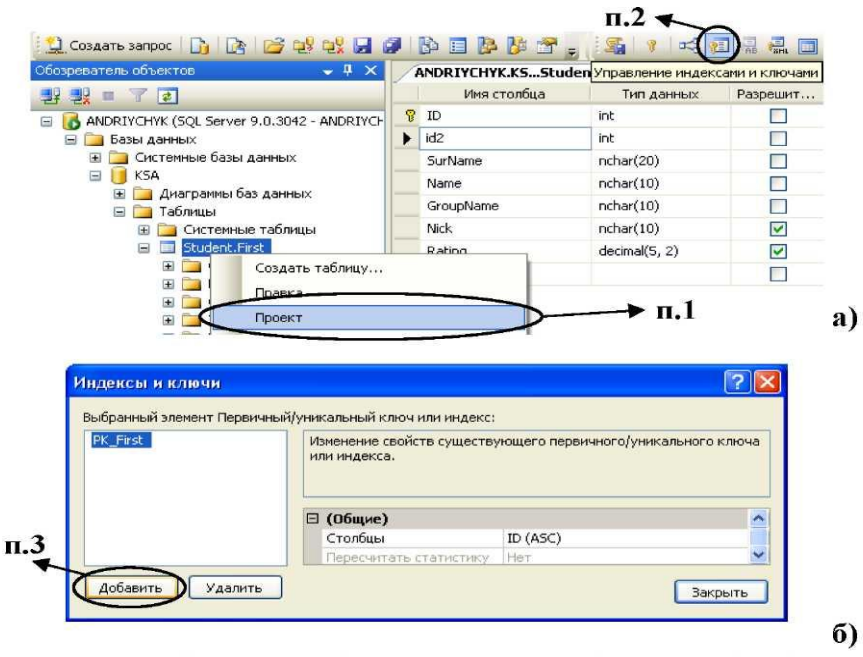


Рис. 2.8. Створення унікального ключа чи індексу таблиці

СТВОРЕННЯ УНІКАЛЬНОГО КЛЮЧА ЧИ ІНДЕКСУ:

1. Для визначеної таблиці зайти в режим конструктора таблиці; для цього в панелі **Object Explorer** (оглядач об'єктів) для вибраної таблиці бази даних розкрити вузли як на рис. 2.7а, та в контекстному меню вузла цієї таблиці вибрати команду Design (проект).
2. На панелі інструментів натиснути по іконці Manage Indexes and Keys (керування індексами та ключами).
3. У діалоговому вікні **Indexes/Keys** (індекси та ключі), у лівій частині вікна (рис. 2.8б), натиснути кнопку **Add** (добавити).
4. Для новоствореного ключа/індексу у полі **Columns** (стовпці) задати стовпець (стовпці) та їхній порядок сортування; для цього слід натиснути кнопку із трикрапкою, розташовану справа від поля (рис. 2.9а); у результаті чого відобразить-

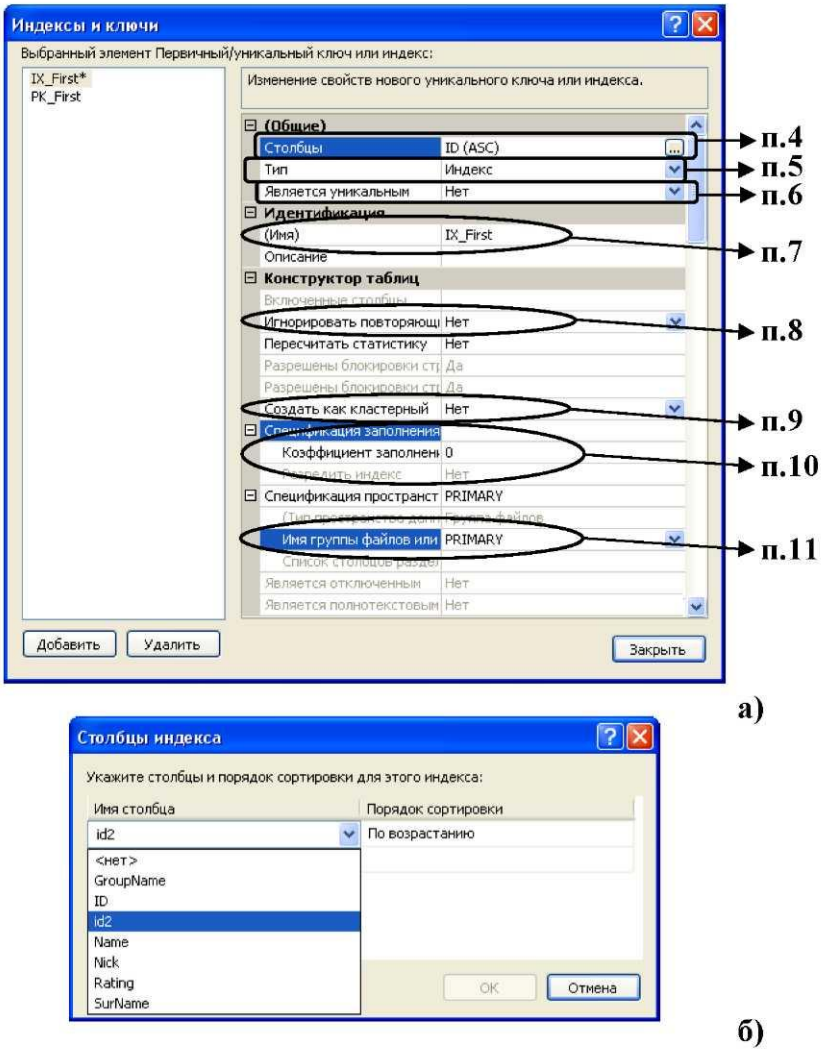


Рис. 2.9. Задання параметрів унікального ключа чи індексу таблиці

ся діалогове вікно **Index Columns** (стовпці індексу). У лівій частині вікна (рис. 2.9б) за допомогою випадаючих списків поля **Column Name** (ім'я стовпця) слід задати кількість, вид та порядок входження стовпців у ключ/індекс. Для кожного стовпця справа від нього в полі **Sort Order** (порядок сортування) можна задати порядок сортування як Ascending (по зростанню), або як Descending (по спаданню).

5. У полі **Type** (тип) вибрати (рис. 2.9а), що необхідно створити: Unique key (унікальний ключ) чи Index (індекс).
6. (лише для індексу) У полі **Is Unique** (є унікальним) визначити чи індекс буде унікальним, чи ні.
7. У полі **(Name)** [(ім'я)] скорегувати назву ключа/індексу в більш зрозумілий контекст, наприклад, *IX_First_id2*.
8. (лише для унікального індексу) У полі **Ignore Duplicate Keys** (ігнорувати повторювані ключі) задати режим генерації повідомлення помилки, у випадку додавання повторюваного значення ключа*.
9. У полі **Create As Clustered** (створити як кластерний), якщо ще не створений для таблиці кластерний індекс, то можна встановити цей ключ/індекс як кластерний.
10. У полі **Fill Factor** (фактор заповнення) при необхідності можна задати значення параметру заповнення (детальніше див. стр. 21).
11. У полі **Filegroup or Partition Scheme** (ім'я групи файлів чи схеми розділів) вибрати групу файлів, у якій буде зберігатися ключ/індекс.
12. Закрити діалогове вікно та зберегти зміни в таблиці.

***** Параметр цього поля визначає дії сервера при операціях додавання даних, що намагаються додати декілька стрічок, у тому числі й із повторюваними значеннями. При значенні Yes (так), стрічки, що не вносять у таблицю повторюваних значень, будуть додані, а дублюючі стрічки будуть відкинуті, і транзакція загалом закінчиться успішно; при значенні No (ні) виводиться повідомлення про помилку та скасовується ввід усього пакету даних.

2.4.3. Створення перевірного обмеження (CHECK).

Обмеження CHECK дає можливість задати діапазон допустимих значень стовпця чи визначити його на основі логічного виразу. Обмеження CHECK завжди повертає логічне значення *True/False*. Перевірний вираз формується, як на основі даних визначеного стовпця, так і даних інших стовпців цієї таблиці; посилатися на стовпці в інших таблицях обмеження не може (**!!!**). SQL Server додає в базу лише ті стрічки з даними, для яких обмеження CHECK повертає значення *True*.

Обмеження CHECK формують на основі тих ж правил, які визначені для логічних операцій конструкції WHERE мови SQL. Деякі приклади цих критеріїв наведені у таблиці 2.4.

Таблиця 2.4. Приклади застосування обмеження CHECK

Логічний вираз	Призначення
Number >= 0	Регламентація використання лише додатних значень
Number **BETWEEN** 1 **AND** 12	Забезпечення значень лише у визначеному діапазонові
Number <= ID	Регламентація значень відносно значень іншого стовпця
Name **IN** ('Петро', 'Вова', 'Василь')	Регламентація допустимих значень
Birthday < getdate()	Регламентація дат днів народжень лише до поточної дати, що повертається системною функцією getdate()
Email **LIKE** '%@%.[a-z][a-z][a-z]'	Забезпечення правильного вводу адресу електронної пошти

У загальному, синтаксис логічних виразів є схожим до аналогічних виразів алгоритмічних мов та складається із логічних, арифметичних, порозрядних, стрічкових операторів, операторів порівняння, системних функцій, числових та символьних даних. Символьні дані повинні бути взятими в одинарні лапки ('). Окрім цього також використовуються у виразах додаткові логічні оператори, наприклад, LIKE, BETWEEN тощо. У таблиці 2.5 та 3.1 дано опис для цих елементів.

Таблиця 2.5. Логічні оператори Microsoft SQL Server 2005

Оператор	Опис
AND	Об'єднує дві умови та повертає значення True, якщо обидві мають значення True. Приклад: Number > 12 **AND** Number < 25 *значення стовпця Number повинні бути більші за 12 та менші за 25*
OR	Об'єднує дві умови та повертає значення True, якщо хоча б одна з них має значення True. Приклад: Number < 12 **OR** Number > 25 *значення стовпця Number повинні бути більші за 12 або менші за 25*
NOT	Інвертує будь-який логічний вираз. Приклад: **NOT** (Number > 12 **AND** Number < 25) *значення стовпця Number НЕ повинні бути більші за 12 та менші за 25*
IS [NOT] NULL	Визначає чи значення у стовпці є **null** (пусте значення). Приклад: Number **IS NOT NULL** *значення стовпця Number не повинні бути пустими, тобто **null***

[NOT] IN	Визначає, чи задане значення співпадає з одним зі значень у списку. Перелік значень наводиться у круглих дужках через кому. Приклад: Number **IN** (1,3.5) *значення стовпця Number повинне бути одним зі значень: 1, 3 або 5* SurName **NOT IN** ('Дзелендзяк', 'Павельчак', 'Самотий') *значення стовпця SurName НЕ повинні містити наступні прізвища: Дзелендзяк, Павельчак, Самотий* **NOT** SurName **IN** ('Дзелендзяк', 'Павельчак', 'Самотий') *умова аналогічна попередній*
[NOT] BETWEEN	Задає діапазон допустимих значень. Для розділення початкових і кінцевих значень використовується оператор **AND**. Приклад: Number **NOT BETWEEN** 1 **AND** 12 *значення стовпця Number НЕ повинно входити у діапазон від 1 до 12, це аналогічно умові* **NOT** (Number > 1 **AND** Number < 12)
[NOT] LIKE	Визначає, чи вказана символьна стрічка співпадає із заданим шаблоном. Шаблон може включати як звичайні символи, так і символи-шаблони (табл. 2.6). Під час порівняння із шаблоном необхідно, щоб його звичайні символи точно співпадали зі символами, що вказані у стрічці. Символи-шаблони можуть співпадати з довільними елементами символьної стрічки.

Таблиця 2.6. Символи-шаблони логічного оператора LIKE

Символ-шаблон	**Опис**
%	Замінює довільну стрічку довжиною від нуля та більше символів. Приклад: SurName **LIKE** '%енко' *поле SurName може містити довільне прізвище, яке закінчується на енко, наприклад, Шевченко, Петренко, Павленко* Note **LIKE** '%студент%' *поле Note може бути довільним, однак воно повинне містити слово студент*
_ (підкреслення)	Замінює довільний одиничний символ. Приклад: Word **LIKE** '_ама' *поле Word може містити довільне слово з 4-х букв із закінченням на ама, наприклад, мама, рама, гама*
[перелік]	Замінює довільний одиничний символ, вказаний у діапазоні, наприклад [a-m] (любий символ від 'a' до 'm' включно), чи наборі, наприклад, [adfm] (будь-який з перелічених символів 'a', 'd', 'f' чи 'm').

	Приклад: SurName **LIKE** '[А-П]%енко' *поле SurName може містити довільне прізвище, яке закінчується на енко та починається на будь-яку букву у проміжку від А до П, наприклад, Арченко, Павленко, Марченко, але не Шевченко (!!!)* Word **LIKE** '[гмр]ама' *поле Word може містити слово із 4-х букв, що починається на одну із букв г, м чи р та закінчується на ама, наприклад, мама, рама, гама, але не дама* PostalCode **LIKE** '[0-9][0-9][0-9][0-9][0-9]' *поле PostalCode повинно містити лише п'ятизначний поштовий індекс* PostalCode **LIKE** '[a-zA-Z][a-zA-Z][a-zA-Z0-9][0-358]' *поле PostalCode повинно містити лише чотиризначний код, формат якого складається з 2-х довільних латинських букв, однієї довільної латинської літери чи числа, та обов'язково закінчується цифрою 0, 1, 2, 3, 5 чи 8*
[^ перелік]	Замінює довільний одиничний символ, що **не входить** у вказаний діапазон чи набір. Приклад: Nick **LIKE** '[^0-9АБ]%' *поле Nick не може містити псевдо, що починається цифрою або буквою А чи Б*

ВАЖЛИВО!

При роботі оператора LIKE з даними у форматі UNICODE (типи даних **nchar** і **nvarchar**) враховуються кінцеві (доповнюючі) пробіли. Тому потрібно це мати на увазі при формуванні шаблону. наприклад, для поля Name типу **nchar(20)** умова Name LIKE '[0-9][0-9]' ніколи не буде задовольнятися. оскільки після 2-х цифр поле містить ще 18 пробілів, що враховуються. Для виправлення некоректної ситуації слід задати вужче поле шириною у 2 символи (**nchar(2)**), або змінити на тип даних ANSI **char(20)**, для якого доповнюючі пробіли не враховуються.

Символи-шаблони (%. _, [,]. ^) **у якості символів.** Є два способи ідентифікації символу-шаблону. як звичайного символу. Перший полягає у тому, що символ-шаблон необхідно помістити у квадратні дужки, наприклад.
Nick **LIKE** '%F[-]15%' *задовольняє довільне входження стрічки 'F-15' у поле.*

Другий спосіб полягає у використанні екрануючих символів та ключового слова ESCAPE (екрануючим може бути довільний символ). наприклад,
Nick **LIKE** '%F!-15%' ESCAPE '!' *результат аналогічний попередньому.*

Якщо у шаблоні LIKE після екрануючого символу немає жодного символу, то шаблон недопустимий. і операція поверне значення False. Якщо символ після екрануючого символу не є символом-шаблоном. тоді екрануючий символ ігнорується і розглядається, як звичайний.

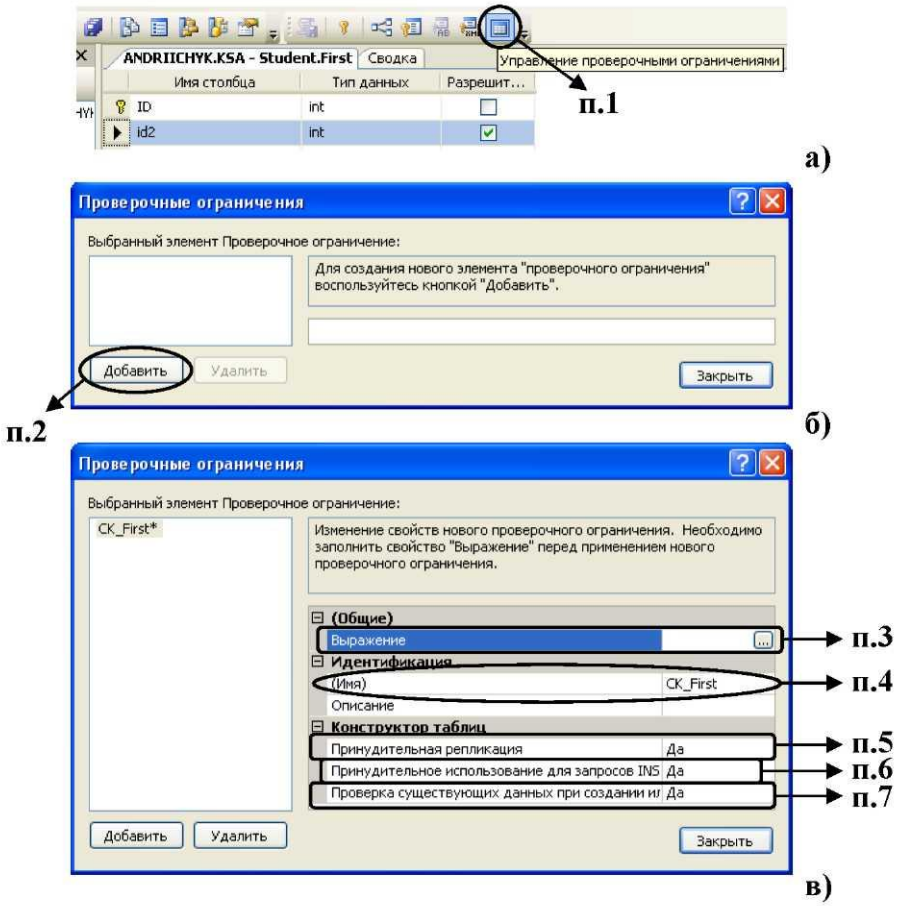


Рис. 2.10. Створення перевірного обмеження CHECK

СТВОРЕННЯ ОБМЕЖЕННЯ **CHECK**:

1. У режимі конструктора робочої таблиці на панелі інструментів натиснути по іконці Manage Check Constraints (керування перевірними обмеженнями) (рис. 2.10а).
2. У діалоговому вікні **Check Constraints** (перевірні обмеження), у лівій частині вікна (рис. 2.10б), натиснути кнопку **Add** (добавити).
3. Для новоствореного обмеження у полі **Expression** (вираз) вписати логічний вираз; для громіздких виразів доцільно скористатися текстовим вікном, що викликається при натисненні кнопки з трикрапкою, розташованої справа від поля.

4. У полі **(Name)** [(ім'я)] скорегувати назву обмеження в більш зрозумілий контекст.
5. У полі **Enforce For Replication** (примусова реплікація) вказати, чи діє обмеження, коли агент реплікації виконує вставлення чи змінення даних у цій таблиці.
6. У полі **Enforce For INSERTs And UPDATEs** (примусове використання для запитів INSERT та UPDATE) вказати, чи діє обмеження при вставці або зміні даних у таблиці.
7. У полі **Check Existing Data On Creation Or Re-Enabling** (перевірення існуючих даних при створенні чи повторному підключенні) вказати, чи перевіряються вже існуючі дані (дані, що вже були у таблиці на момент створення обмеження) на відповідність обмеженню CHECK.
8. Закрити діалогове вікно **Check Constraints** (перевірні обмеження) та зберегти зміни в таблиці.

Синтаксична структура обмежень переважно розрахована на той випадок, коли вони створюються одночасно зі створенням таблиць. Але іноді виникають обставини, коли для вже існуючої таблиці з даними необхідно створити обмеження CHECK, яке б не дозволяло, наприклад, введення від'ємних значень, але вже існуючі стрічки з даними при цьому слід залишити. І тому, щоб ввести в дію нове обмеження, але виключити його застосування до вже існуючих даних, необхідно задати значення false у полі **Check Existing Data On Creation Or Re-Enabling** (п.7).

2.4.4. Створення реляційних зв'язків за допомогою зовнішніх ключів (FOREIGN KEY).

Зовнішні ключі призначені для забезпечення цілісності даних та для створення зв'язків між таблицями. В базі даних зовнішній ключ -це стовпець (чи сукупність стовпців), що співпадає з первинним (унікальним) ключем певної таблиці. Якщо значення зовнішнього ключа відповідає значенню первинного (унікального) ключа, то стає зрозумілим, що між об' єктами бази даних, які представлені співпадаючими стрічками таблиць, існує логічне взаємовідношення.

Основним обмеженням відношення є цілісність посилання. Воно визначає, що кожне значення (не **null)** зовнішнього ключа, повинно посилатися на певне існуюче значення первинного (унікального) ключа. Іншими словами, якщо хтось один посилається на когось іншого, то той «інший» має існувати, інакше система видасть помилку.

Такий спосіб обмеження дає можливість будувати різноманітні відношення між даними у базі даних:

- відношення «один до багатьох»;
- відношення «багато до багатьох»;
- зворотні (рекурсивні) відношення.

Відношення «один до багатьох». Цей найпопулярніший тип відношення зв'язує один запис-предок з декількома записами-нащад-ками. Відношення встановлюється між первинним (унікальним) ключем базової таблиці-предка та

зовнішнім ключем вторинної таблиці-нащадка. Відношення «один до багатьох» насправді потрібно розглядати, як відношення «один до любого числа», та, що охоплює відношення «один до нуля», «один до одного» та «один до багатьох».

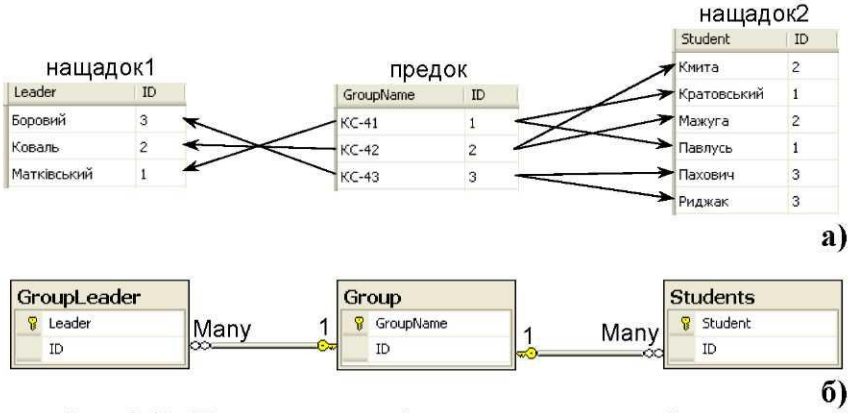


Рис. 2.11. Представлення відношення «один до багатьох»

На рис. 2.11 наведено реалізацію 2-х зв'язків «один до багатьох». У цьому прикладі таблиця-предок містить записи з переліком груп 4-го курсу кафедри КСА. Таблиця-нащадок1 містить записи ; прізвищами старостів груп, а таблиця-нащадок2 - записи з прізвищами студентів усього потоку. Для таблиць встановленні первинні ключі для таких полів: *GroupName, Leader, Student.* Для таблиці-предка задано унікальний ключ по полю *ID,* а для таблиць-нащадків визначені відповідно, зовнішні ключі по власних полях *ID,* відносно цього уні кального ключа. Зазначимо, що зв'язок предок-нащадок 1 відповідає відношенню «один до одного», бо в одній групі може бути лише один староста. Інший зв'язок, предок-нащадок2, повноцінно реалізує відношення «один до багатьох».

Відношення «один до одного» представляє собою фактично доповнення чи продовження основної таблиці. Такий підхід дає можливість перенести конфіденційну інформації в окрему таблицю, доступ до якої SQL Server надає користувачам згідно наданих їм прав.

Розрізняють два класи відношень - обов'язкові та необов'язкові.

Обов'язкове відношення вимагає міграції поля таблиці-предка у екземпляр нащадка. При реалізації цього відношення мігруючий (зовнішній) ключ **не** повинен бути **null** (**!!!**). Тобто кожне поле табли-ці-нащадка повинне обов' язково належати певному полю таблиці-предка, як на рис. 2.11а.

Необов'язкове відношення передбачає, що нащадок не завжди повинен мати значення для мігруючого (зовнішнього) ключа. У цьому випадку поле зовнішнього ключа може приймати значення **null.** На рис. 2.12 показано, що таблиця-нащадок містить прізвища, які не відносяться до жодної з груп 4-го курсу.

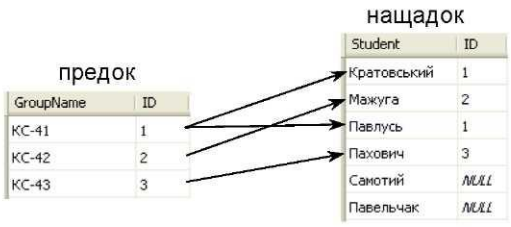


Рис. 2.12. Необов'язкове відношення «один до багатьох»

Відношення «багато до багатьох». У відношеннях цього типу обидві сторони пов'язані з значною кількістю (більше одного) елементів протилежної сторони. Наприклад, є ряд студентів, що мають декілька академзаборгованостей з різних дисциплін (рис. 2.13а). Тобто, кожен зазначений студент має відношення до декількох назв предметів. З іншого боку, кожна навчальна дисципліна має відношення до декількох студентів. І в такий спосіб, між полями двох таблиць утворюються зв' язки типу «багато до багатьох».

Необхідно зазначити, що фактично зреалізувати відношення «багато до багатьох» не є можливим, і тому для логічної розв' язки цього відношення використовують асоціативну таблицю, яку іноді ще називають стикувальною. У результаті чого здійснюється заміна відношення «багато до багатьох» двома відношеннями «один до багатьох» (рис. 2.13б).

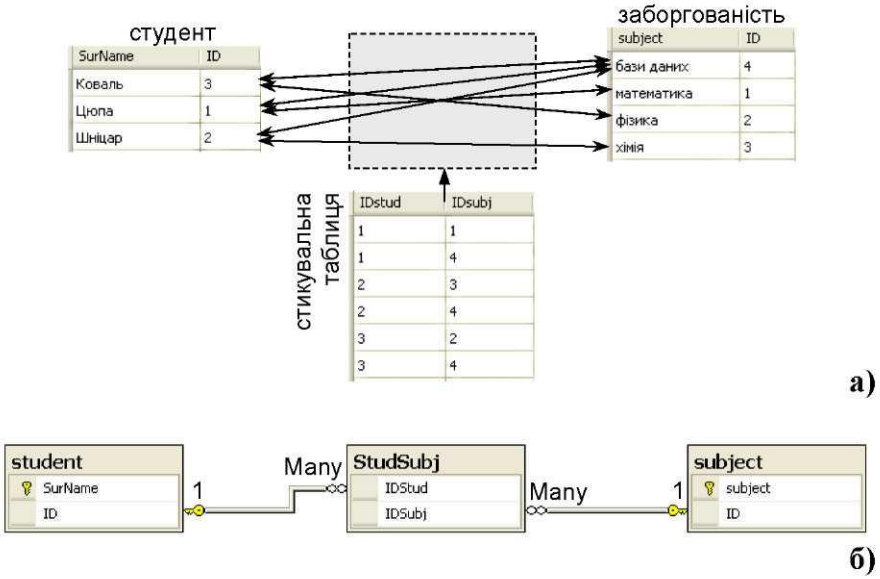


Рис. 2.13. Представлення відношення «багато до багатьох»

Зворотне (рекурсивне) відношення. Це відношення моделюється не відносно іншої таблиці з даними, а в межах тієї самої. Його ще іноді називають ієрархічним, тому що воно дає можливість моделювати деревовидні структури даних (рис. 2.14).

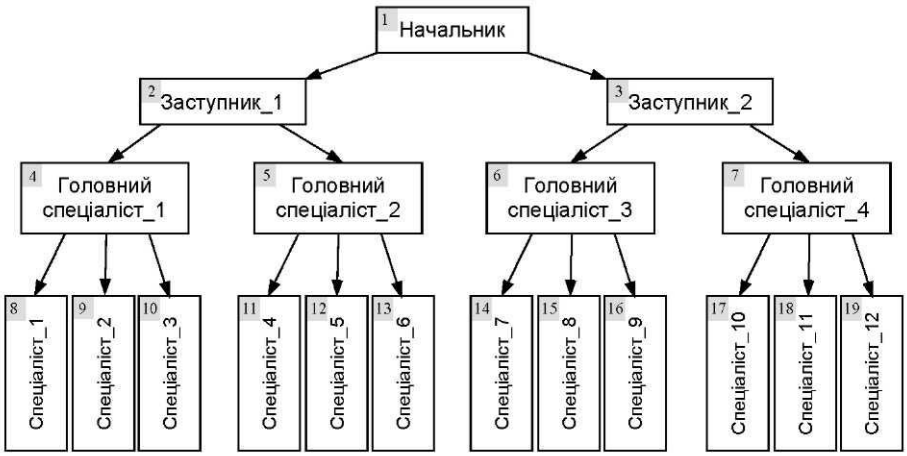
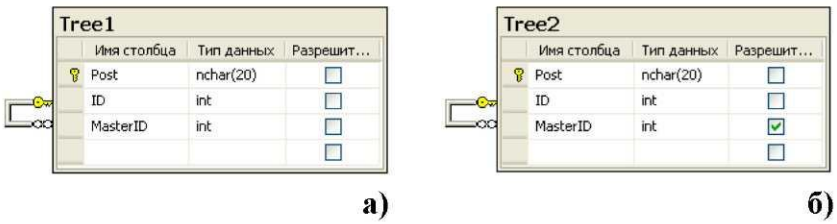


Рис. 2.14. Ієрархічна структура підрозділу

Post	ID	MasterID
Начальник	1	1
Заступник1	2	1
Заступник2	3	1
Гол.Спец.1	4	2
Гол.Спец.2	5	2
Гол.Спец.3	6	3
Гол.Спец.4	7	3
Спеціаліст1	8	4
Спеціаліст2	9	4
Спеціаліст3	10	4
Спеціаліст4	11	5
Спеціаліст5	12	5
Спеціаліст6	13	5
Спеціаліст7	14	6
Спеціаліст8	15	6
Спеціаліст9	16	6
Спеціаліст10	17	7
Спеціаліст11	18	7
Спеціаліст12	19	7

Post	ID	MasterID
Начальник	1	*NULL*
Заступник1	2	1
Заступник2	3	1
Гол.Спец.1	4	2
Гол.Спец.2	5	2
Гол.Спец.3	6	3
Гол.Спец.4	7	3
Спеціаліст1	8	4
Спеціаліст2	9	4
Спеціаліст3	10	4
Спеціаліст4	11	5
Спеціаліст5	12	5
Спеціаліст6	14	5
Спеціаліст7	15	6
Спеціаліст8	16	6
Спеціаліст9	17	6
Спеціаліст10	18	7
Спеціаліст11	19	7
Спеціаліст12	20	7



а) **б)**

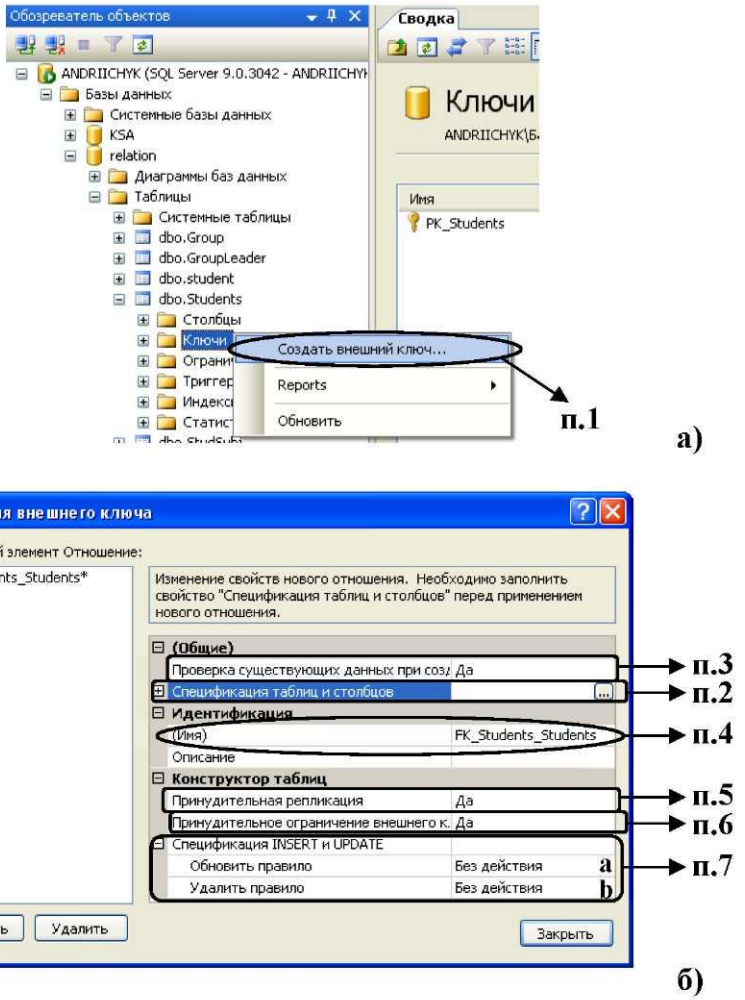
Рис. 2.15. Представлення зворотного відношення

Зворотне відношення по своїй суті є звичайним відношенням «один до багатьох», але лише в межах тієї самої таблиці. При цьому, значення первинного (унікального) ключа мігрує у поле зовнішнього ключа. Для таблиці (рис. 2.15), що реалізує ієрархію певного підрозділу (рис. 2.14), є визначені такі ключі: первинний - для поля *Post*, унікальний - для поля *ID*, зовнішній - для поля *MasterID;* поле *ID є* авто-інкрементним; зворотне відношення тут реалізує зв'язок ID-MasterID.

Інший важливий момент реалізації зворотного (рекурсивного) відношення пов'язаний із внесенням у таблицю першого запису. Оскільки зовнішній ключ вимагає міграції у нього значення з іншого запису, то одразу ж і виникає дилема, де його взяти. Вирішення цієї дилеми може бути таке:

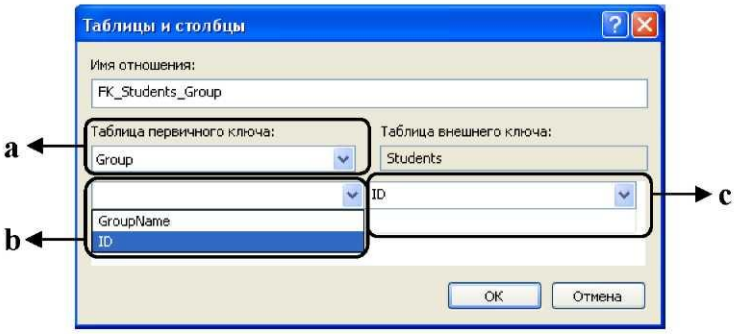
а) Перед створенням зовнішнього ключа необхідно внести у таблицю перший запис, який потім буде посилатися на себе ж самого, або, навпаки, створити зовнішній ключ, а потім відмінити його дію на деякий час, поки не буде внесено перший запис (рис. 2.15а);

б) Стовпець для зовнішнього ключа повинен дозволяти ввід **ІШП1**-значення. Таким чином з'явиться можливість ввести першу стрічку, що має **і**шП-значення у стовпці зовнішнього ключа, і тим сам уникнути необхідності примусового введення першої стрічки (рис. 2.15 б).



продовження рис. 2.16 на наступній сторінці →

← початок рис. 2.16 на попередній сторінці



в)

Рис. 2.16. Створення зовнішнього ключа таблиці

СТВОРЕННЯ ЗОВНІШНЬОГО КЛЮЧА:

1. У панелі **Object Explorer** (оглядач об'єктів) для вибраної таблиці робочої бази даних розкрити вузли як на рис. 2.16а, та в контекстному меню вузла **Keys** (ключі) вибрати команду New Foreign Key… (створити зовнішній ключ), після чого з'явиться діалогове вікно **Foreign Key Relationships** (відношення зовнішнього ключа). Далі слід задати значення для параметрів зовнішнього ключа.
2. Задати механізм зв'язків для зовнішнього ключа в контексті відношення «один до багатьох». Для цього у полі **Tables And Columns Specification** (специфікація таблиць і стовпців) натиснути кнопку з трикрапкою, розташовану справа від поля (рис. 2.16б). В результаті з'явиться діалогове вікно **Tables and Columns** (таблиці та стовпці) (рис. 2.16в). Далі слід вибрати стовпці для зовнішнього та первинного (унікального) ключів:
   a. з випадаючого списку вибрати назву таблиці, де розміщений первинний (унікальний) ключ;
   b. з випадаючого списку вибрати стовпець (стовпці) з первинним (унікальним) ключем;
   c. з випадаючого списку вибрати стовпець (стовпці) для зовнішнього ключа;

   Натиснути кнопку OK.
3. У полі **Check Existing Data On Creation Or Re-Enabling** (перевірення існуючих даних при створенні чи повторному

включенні) вказати, чи буде виконуватися дане перевірення для вже існуючих даних.

4. У полі **(Name)** [(ім'я)] скорегувати назву зовнішнього ключа у більш зрозумілий контекст.
5. Значення поля **Enforce For Replication** (примусова реплікація) вказує, чи використовується дане обмеження, коли агент реплікації виконує у таблиці вставлення, зміну чи видалення.
6. Значення поля **Enforce Foreign Key Constraint** (примусове обмеження зовнішнього ключа) дає можливість відмінити дію зовнішнього ключа щодо забезпечення цілісності даних; для його відміни слід встановити значення поля у false.
7. У категорії **INSERT And UPDATE Specification** (специфікація INSERT і UPDATE) задаються правила для видалення та зміни зв'язку цього відношення*:
   a. у полі **Delete Rule** (видалити правило) вибрати механізм дії SQL Server при спробі користувача видалити поле із записом предка, що мігрує у екземпляр нащадка;
   b. у полі **Update Rule** (обновити правило) вибрати механізм дії SQL Server при спробі користувача обновити поле із записом предка, що мігрує у екземпляр нащадка.
8. Зберегти зміни в таблиці.

* Особливістю зовнішніх ключів, на відміну від інших, є те, що вони двонаправлені: забезпечують, як правильність вводу значень у поля екземпляра нащадка у відповідності до предка, так і забезпечують перевірку зміни значення при діях над записами у таблиці-предка (що запобігає появі у таблиці-нащадка «висячих стрічок», які втратили зв'язок із предком). За замовчуванням SQL Server «захищає» від видалення (чи зміни значення ключа) ті стрічки таблиці-предка, з якими вони мають зв' язки у таблиці-нащадка. У цьому випадку для видалення такого запису, спершу необхідно видалити усі записи, що відповідають йому у таблиці-нащадку. Тому у SQL Server передбачений відповідний механізм, що відповідає за автоматичне виконання подібних операцій видалення чи обновлення, та має назву **процесу каскадного виконання дій.** Його значення особливо суттєве при операціях видалення, що проходять через декілька рівнів залежностей: одна стрічка залежить від другої, друга від третьої і т.д. При операції обновлення значення ключа предка, автоматично виконується обновлення значень і для усіх зв' язаних з ним записів екземпляра нащадка.

Відповідно, поля **Delete Rule** (видалити правило) та **Update Rule** (обновити правило) можуть приймати такі значення:

- **No Action** (без дій) - формує повідомлення про помилку, щодо неможливості операції видалення чи обновлення для стрічок зі значенням ключа, що мігрує у екземпляр нащадка; для операції видалення чи обновлення система виконує відкіт;
- **Cascade** (каскадом) - видаляє (обновлює) всі стрічки, що беруть участь у зв'язках зовнішнього ключа;
- **Set Null** (задати Null) - задає значення **null,** якщо всі стовпці зовнішніх ключів в таблиці можуть бути встановленні в **null;**

• **Set Default** (задати за замовчуванням) - встановлює значення за замовчуванням, що визначене для цього стовпця, при умові, що всі стовпці зовнішнього ключа в таблиці мають значення за замовчуванням.

2.5. Створення діаграм для баз даних.

Діаграма бази даних - це візуальне представлення проекту бази даних. Таке представлення відображає усі таблиці з іменами їх стовпців та усі зв'язки між таблицями. У Management Studio є передбачений відповідний інструментарій для побудови таких діаграм, або як їх ще називають, ER-діаграм.

Перед такими засобами формування ER-діаграм ставлять, як правило, дві задачі:

• підготовлення сценарію для створення бази даних на основі результатів схематичної структури даних;

• зворотне проектування бази даних, тобто підключення до існуючої бази, її сканування та формування діаграми, в якій правильно відображаються усі її об' єкти (таблиці, індекси, ключі тощо). Зазначимо, що інструментарій Management Studio реалізує ці задачі не в повній мірі. Для цього існують окремі інструментальні засоби високо класу (при цьому їх вартість може сягати і до 15 тис. $ за одне робоче місце). Однак, навіть існуючий в Management Studio інструментарій дає можливість, як проектувати з «нуля» базу даних у вигляді ER-діаграми, так і відтворювати ER-діаграму для вже існуючої бази даних.

Оскільки на попередньому етапі ми займалися поетапним створенням об' єктів бази даних безпосередньо у вузлах її дерева, то тепер для цієї бази даних сформуємо діаграму, що відображала б її таблиці та зв' язки між ними.

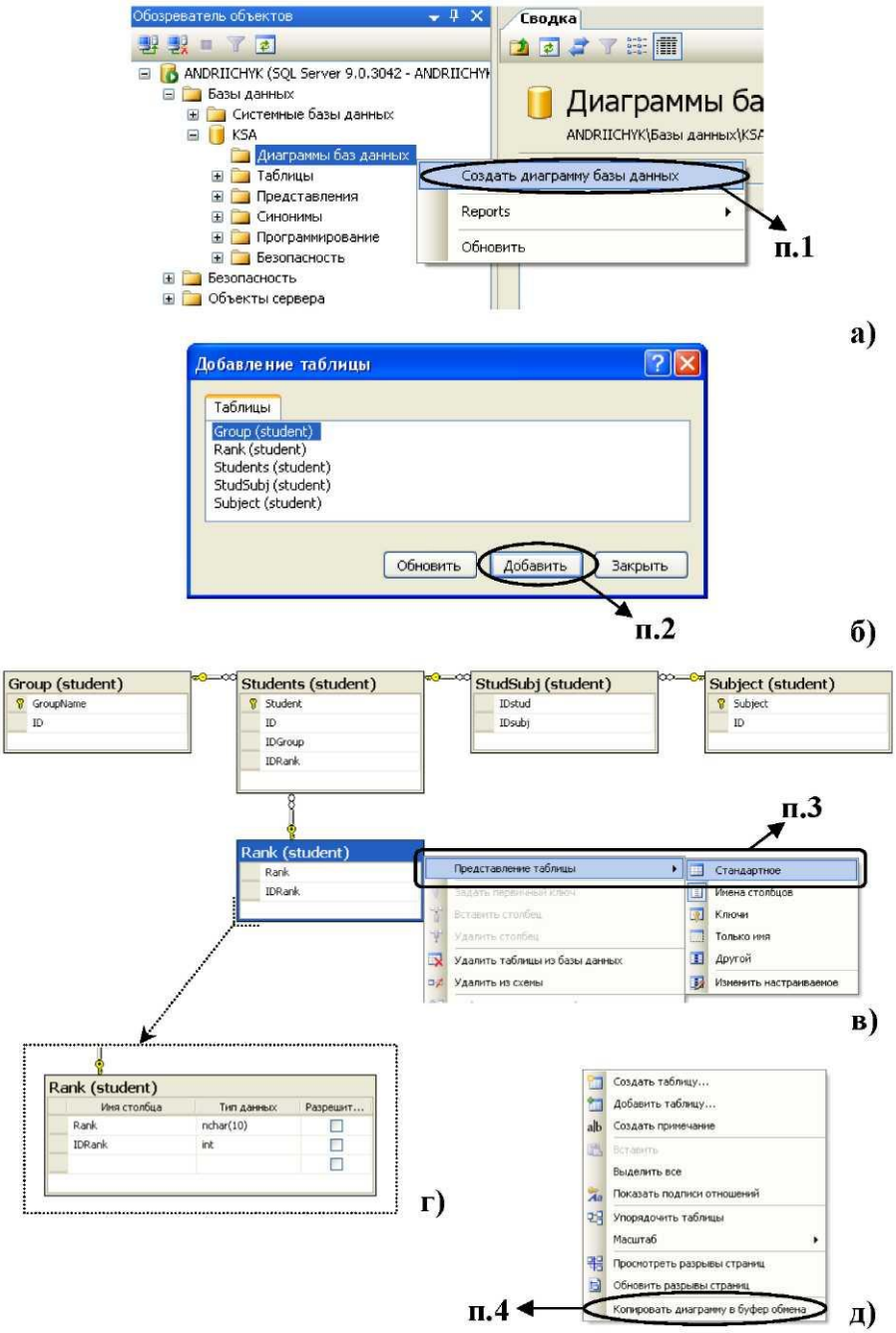


Рис. 2.17. Створення діаграми бази даних

СТВОРЕННЯ ДІАГРАМИ БАЗИ ДАНИХ:

1. У панелі **Object Explorer** (оглядач об'єктів) для вибраної бази даних розкрити вузол (рис. 2.17а) та вибрати пункт **Database Diagrams** (діаграми баз даних). Якщо цей пункт вибирається вперше, то система видасть діалогове вікно щодо відсутності певних об'єктів підтримки, тому слід натиснути кнопку **Yes** (так) для їх створення. Далі натиснути правою клавішею миші по пункту **Database Diagrams** та у контекстному меню вибрати команду New Database Diagram (створити діаграму бази даних), у результаті чого з'явиться діалогове вікно **Add Table** (додавання таблиці).
2. У діалоговому вікні **Add Table** (додавання таблиці) додати усі наявні в базі даних таблиці у діаграму (рис. 2.17б).
3. Після того, як система автоматично розташує усі таблиці зі зв'язками, слід відформатувати вигляд самої діаграми: розташувати таблиці у кращому для сприйняття вигляді та задати для них необхідний формат відображення. Для зміни формату таблиці натиснути правою клавішею миші по вибраній таблиці та у контекстному меню вибрати команду, наприклад, Table View (представлення таблиці) → Standard (стандартне) (рис. 2.17в), після чого вигляд таблиці буде, як на рис. 2.17г.
4. Для перенесення готової діаграми у інші графічно-текстові програми, наприклад, Microsoft Word чи Excel, слід натиснути праву клавішу мишу на чистому полі діаграми та у контекстному меню вибрати команду Copy Diagram to Clipboard (копіювати діаграму в буфер обміну) (рис. 2.17д).
5. При необхідності діаграму зберегти.

Необхідно зазначити, що починати створення бази даних можна і на чистому аркуші діаграми, поступово додаючи таблиці, ключі, індекси, зв'язки і т.п., та по ходу заповнюючи їхні поля з властивостями.

2.6. Транспортування бази даних.

Коли база даних є вже повністю спроектованою, постає питання про її перенесення замовнику чи, скажімо навіть, на інший комп'ютер. Це можна здійснити двома шляхами:

А. Від'єднати готову базу даних від підключення до SQL Server, скопіювати на носій та розмістити на жорсткому диску іншого ком-п' ютера, після чого знову приєднати її до сервера SQL Server.

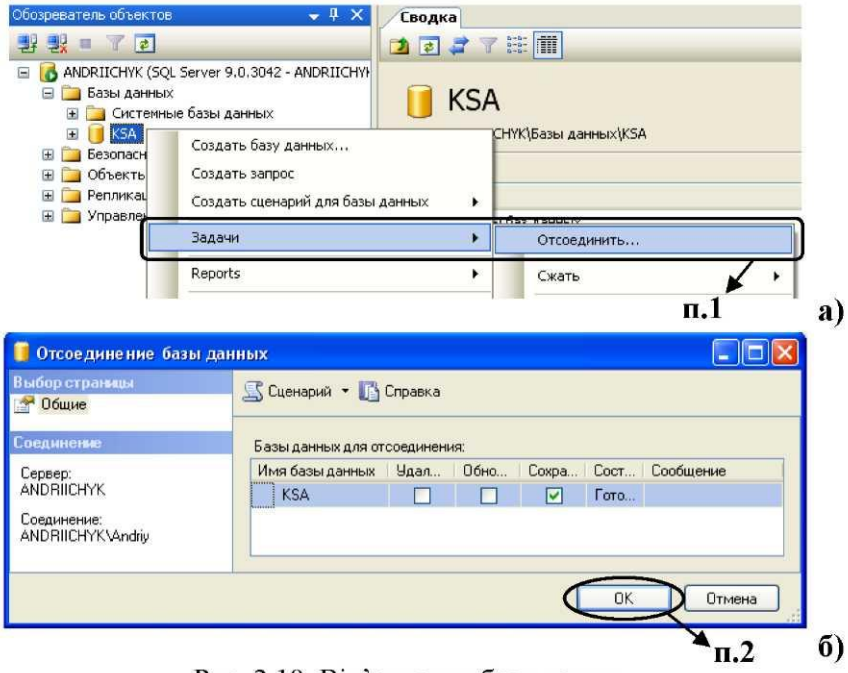
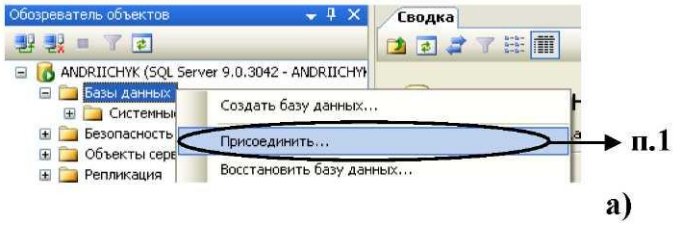


Рис. 2.18. Від'єднання бази даних

ВІД'ЄДНАННЯ БАЗИ ДАНИХ:

1. У панелі **Object Explorer** (оглядач об'єктів) вибрати назву бази даних та правою кнопкою миші натиснути по ній, після чого вибрати у контекстному меню команду Tasks (задачі) → Detach… (від'єднати…) (рис. 2.18а), у результаті чого з'явиться діалогове вікно.
2. У діалоговому вікні **Detach Database** (від'єднання бази даних) слід натиснути кнопку підтвердження **OK** (рис. 2.18б).



продовження рис. 2.19 на наступній сторінці →

← початок рис. 2.19 на попередній сторінці

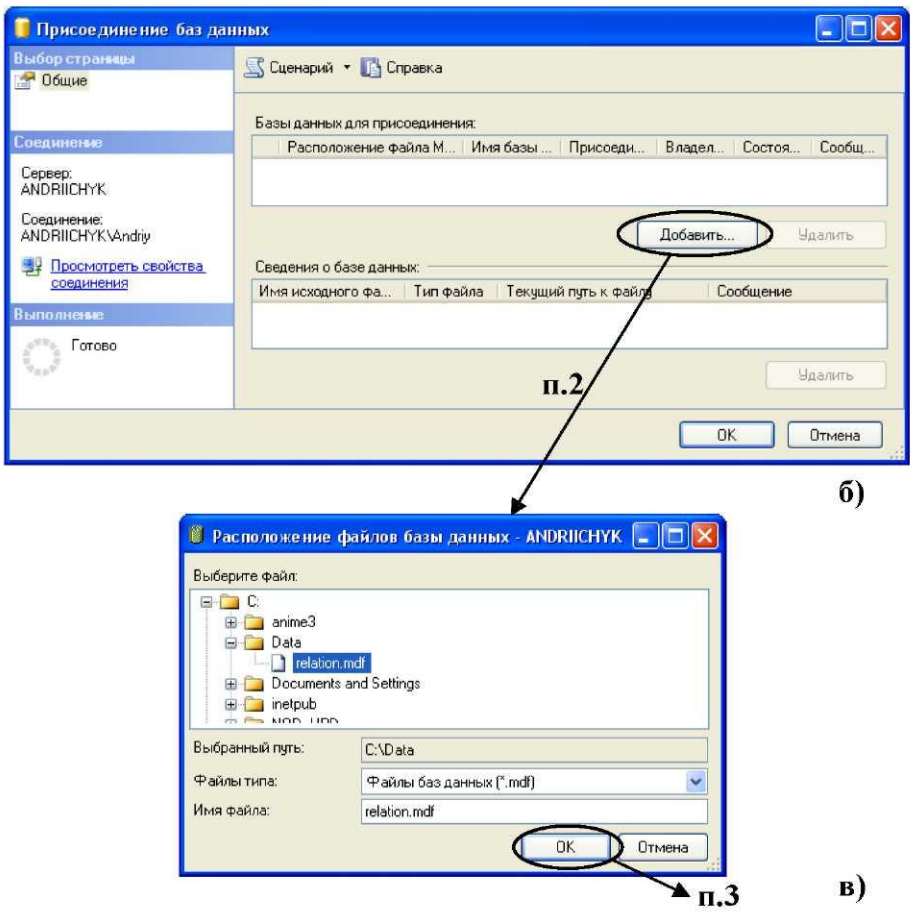


Рис. 2.19. Приєднання бази даних

ПРИЄДНАННЯ БАЗИ ДАНИХ:

1. У панелі **Object Explorer** (оглядач об'єктів) натиснути правою клавішею миші по вузлу **Databases** (бази даних) та у контекстному меню вибрати команду Attach… (приєднати…) (рис. 2.19а).
2. У діалоговому вікні **Attach Databases** (приєднання баз даних) натиснути кнопку **Add…** (додати…) (рис. 2.19б), в результаті чого з'явиться вікно з провідником каталогів **Locate Database Files** (розташування файлів баз даних).

3. У провіднику каталогів (рис. 2.19в) вказати шлях до розміщення файлу бази даних та підтвердити свій вибір натиском кнопки **OK**.
4. Натиснути кнопку **OK** у вікні **Attach Databases** (приєднання баз даних) для підтвердження свого загального вибору.

Б. Для спроектованої бази даних за допомогою автоматизованого побудовувача сценаріїв Management Studio згенерувати SQL-код, як для створення самої бази даних, так і для створення кожного її об'єкту (схеми, таблиці, первинного ключа, унікального ключа, індексу, зовнішнього ключа

3. Додаткові відомості

Таблиця 3.1. Деякі оператори та функції Microsoft SQL Server 2005

Оператор / функція	Опис
Арифметичні оператори	
+ (додавання)	Виконує операцію додавання двох чисел
+ (плюс)	Повертає додатне значення числового виразу
+ (зчеплення стрічок)	Зчіплює один чи декілька символів, чи комбінацію стрічок в один вираз
– (віднімання)	Виконує операцію віднімання одного числа від іншого
– (заперечення)	Повертає від'ємне значення числового виразу
* (множення)	Виконує операцію множення двох чисел
/ (ділення)	Виконує операцію ділення одного числа на інше
% (модуль)	Повертає залишок від ділення двох цілих чисел
Порозрядні оператори	
& (порозрядне І)	Виконує операцію AND над розрядами 2-х цілих чисел
| (порозрядне АБО)	Виконує операцію OR над розрядами 2-х цілих чисел
^ (порозрядне виключне АБО)	Виконує операцію XOR над розрядами 2-х цілих чисел
~ (порозрядне НЕ)	Виконує операцію NOT над розрядами цілого числа
Оператори порівняння	
= (рівне)	Виконує операцію порівняння на предмет рівності значень двох виразів
<> != (не рівне)	Виконує операцію порівняння на предмет нерівності значень двох виразів
< (менше за)	Виконує операцію порівняння, яка визначає, чи значення зліва є меншим за значення справа
> (більше за)	Виконує операцію порівняння, яка визначає, чи значення зліва є більшим за значення справа

тощо).

<= (менше або рівне)	Виконує операцію порівняння, яка визначає, чи значення зліва є меншим чи рівним значенню справа
>= (більше або рівне)	Виконує операцію порівняння, яка визначає, чи значення зліва є більшим чи рівним значенню справа
Математичні функції	
ABS(*expr*)	Повертає додатне, абсолютне значення виразу.
ACOS(*expr*)	Повертає значення кута в радіанах, косинус якого є заданий *expr* (значення виразу повинне знаходитися у межах [-1; 1]).
ASIN(*expr*)	Повертає значення кута в радіанах, сінус якого є заданий *expr* (значення виразу повинне знаходитися у межах [-1; 1]).
ATAN(*expr*)	Повертає значення кута в радіанах, тангенс якого є заданий *expr* (значення виразу повинне знаходитися у межах [-1; 1]).
CEILING(*expr*)	Повертає найближче ціле число, що є більшим чи рівним значенню *expr*, тобто виконує заокруглення вгору.
COS(*expr*)	Повертає значення косинуса кута, заданого *expr* у радіанах.
COT(*expr*)	Повертає значення котангенса кута, заданого *expr* у радіанах.
DEGREES(*expr*)	Перетворює значення кута *expr* з радіан у градуси.
EXP(*expr*)	Повертає експоненту значення *expr*.
FLOOR(*expr*)	Повертає найближче ціле число, що є меншим чи рівним значенню *expr*, тобто виконує заокруглення вниз.
LOG(*expr*)	Повертає натуральний логарифм для значення *expr*.
LOG10(*expr*)	Повертає десятковий логарифм для значення *expr*.
PI()	Повертає значення константи π.
POWER(*expr*, *power*)	Здійснює піднесення *expr* у степінь *power*.
RADIANS(*expr*)	Перетворює значення кута *expr* з градусів у радіани.
RAND([*seed*])	Повертає випадкове число з плаваючою комою в інтервалі від 0 до 1, може містити необов'язковий цілочисельний параметр *seed* для рандомізації генератора випадкових чисел, якщо число не вказано, тоді це значення формується системою на основі системного часу.
ROUND(*expr*, *length* [, *function*])	Повертає значення для *expr*, заокруглене до вказаної довжини чи точності параметром *length*. Якщо *length* додатне число, то *expr* заокруглюється до кількості цифр після коми, що вказане *length*. Якщо від'ємне – заокруглення *expr* здійснюється зліва від крапки. Якщо значення *function* відсутнє чи є рівним 0, *expr* заокруглюється, якщо ж відмінне від нуля, тоді *expr* усікається. Приклади: ROUND(123.5782, 2) → 123.5800

	ROUND(123,5782. -2) → 100,0000 ROUND(123.5782. 2. 1) → 123,5700
SING(*expr*)	Повертає знак числа *expr*. Якщо значення додатне → повертає +1, нульове → 0, від'ємне → -1.
SIN(*expr*)	Повертає значення сінуса кута, заданого *expr* у радіанах.
SQRT(*expr*)	Повертає квадратний корінь *expr*.
SQUARE(*expr*)	Підносить у квадрат *expr*.
TAN(*expr*)	Повертає значення тангенса кута, заданого *expr* у радіанах.
Стрічкові функції	
ASCII(*expr*)	Повертає значення ASCII-коду для крайнього лівого символу *expr*.
CHAR(*expr*)	Повертає символ згідно значення ASCII-коду *expr*.
CHARINDEX(*expr*, *string* [, *start*])	Виконує пошук підстрічки *expr* у стрічці *string*, починаючи зі символу *start*, та повертає початкову позицію. Якщо *start* відсутнє (чи недодатне число), то пошук розпочинається з початку стрічки *string*.
DIFFERENCE(expr1, expr2)	Повертає число у діапазоні 0-4, згідно якого можна судити про співпадіння звучання двох стрічок.
LEFT(*expr*, *int*)	Повертає *int* символів з початку стрічки *expr*.
LEN(*expr*)	Повертає довжину (к-сть символів) стрічки *expr*.
LOWER(*expr*)	Перетворює всі символи стрічки *expr* у нижній регістр.
LTRIM(*expr*)	Видаляє зі стрічки *expr* усі початкові пробіли.
NCHAR(*int*)	Повертає символ UNICODE згідно значення *int*.
PATINDEX(*'%pat%'*, *expr*)	Виконує у стрічці пошук підстрічки, що відповідає заданому шаблону *'%pat%'*. Повертає початкову позицію знайденої підстрічки.
QUOTENAME(*'ch str'* [, *'quote character'*])	Повертає стрічку UNICODE з доданням розділювачів, перетворюючи *'ch str'* у правильний ідентифікатор з розділювачем Microsoft SQL Server 2005.
REPLACE(*expr1*, *expr2*. *expr3*)	Замінює усі входження *expr2* у *expr1* на *expr3*.
REPLICATE(*expr*, *int*)	Повторює стрічку *expr* вказане числом *int* раз.
REVERSE(*expr*)	Повертає стрічку, записану навпаки до *expr*.
RIGHT(*expr*, *int*)	Повертає *int* символів, розташованих справа у *expr*.
RTRIM(*expr*)	Видаляє зі стрічки *expr* усі кінцеві пробіли.
SOUNDEX(*expr*)	Перетворює *expr* у чотирисимвольний код, що використовується для знаходження подібно звучних слів чи імен. Перший символ коду є першим символом *expr*, а 2-4 символи – цифрами. Голосні в *expr* ігноруються, якщо тільки не є першими буквами *expr*.
SPACE(*int*)	Повертає стрічку, що складається з *int* пробілів.

STR(*num* [, *length*, *decimal*])	Перетворює вираз з десятковою комою *num* у стрічку. Параметр *length* задає загальну довжину, включаючи десяткову кому, цифри та пробіли. Параметр *decimal* визначає кількість знаків справа від десяткової коми.
STUFF(*expr1*, *start*, *length*, *expr2*)	Видаляє у стрічці *expr1*, починаючи з позиції *start*, число *length* символів, а натомість вставляє стрічку *expr2*.
SUBSTRING(*expr*, *start*, *length*)	Повертає частину стрічки *expr* довжиною *length*, починаючи з позиції *start*.
UNICODE(*expr*)	Повертає значення UNICODE-коду для крайнього лівого символу *expr*.
UPPER(*expr*)	Перетворює всі символи стрічки *expr* у верхній регістр.
Функції для роботи з датою та часом	
DATEADD(*datepart*, *number*, *date*)	Додає до вказаної дати *date* певне число *number* днів, годин, хвилин тощо. За допомогою аргументу *datepart* вказується, яку частину дати необхідно змінити.
DATEDIFF(*datepart*, *startdate*, *enddate*)	Повертає різницю між двома датами *startdate* та *enddate*, у заданих аргументом *datepart* одиницях виміру часу.
DATENAME(*datepart*, *date*)	Повертає значення *date* у стрічковому форматі, згідно зі специфікацією аргументу *datepart*.
DATEPART(*datepart*, *date*)	Виділяє з дати *date* певну частину у відповідності до аргументу *datepart*, та повертає у числовому форматі.
DAY(*date*)	Повертає числове значення дня місяця.
GETDATE()	Повертає поточну системну дату та час.
GETUTCDATE()	Повертає поточне значення дати та часу за Грінвічем, який виводиться з поточного місцевого часу з врахуванням часового поясу в операційній системі комп`ютера.
MONTH(*date*)	Повертає числове значення номера місяця у році.
YEAR(*date*)	Повертає числове значення року.

Значення аргументу *datepart*:

Опис	значення	скорочене значення
рік	year	yy, yyyy
квартал	quarter	qq, q
місяць	month	mm, m
номер дня у році	dayofyear	dy, y
день місяця	day	dd, d
тиждень	week	wk, ww
день тижня	weekday	dw, w
година	hour	hh
хвилина	minute	mi, n
секунда	second	ss, s
мілісекунда	millisecond	ms

4. Порядок виконання роботи

1. Вдома детально вивчити поданий у інструкції довідковий теоретичний матеріал до лабораторної роботи.

2. Згідно варіанту (порядкового номера в журналі викладача) завдання (таблиця 4.1), вдома розробити детальний проект зазначеної бази даних, а у лабораторії реалізувати його за допомогою графічного інструменту Management Studio та перевірити на працездатність.

3. Завершений проект бази даних на комп' ютері продемонструвати викладачу.

4. За результатами виконаної роботи оформити звіт та здати його.

Таблиця 4.1. Завдання до лабораторної роботи

№ п/п	Завдання
	1. Створити базу даних (БД). 2. У цій БД створити схему з назвою, що відповідає прізвищу студента. Усі новостворювані об`єкти повинні належати цій схемі. 3. Побудувати усі необхідні об`єкти (таблиці, первинні ключі, вторинні ключі, зовнішні ключі, індекси, перевірні обмеження) для вказаної БД. 4. Внести у кожну таблицю БД як мінімум по 10 абстрактних записів (якщо кількість звісно не обмежується логікою). 5. Створити діаграму для повністю спроєктованої БД. Таблиці повинні відображатися на діаграмі у стандартному режимі (ім`я стовпця, тип даних, дозволити знач. null) (рис. 2.17г). Зв`язки між таблицями відкоригувати так, щоб вказували між якими саме стовпцями вони встановлюють зв`язок. Позначення: **P.K.** – первинний ключ; **U.I.** – унікальний індекс; **I.** – неунікальний індекс; **CHECK** – перевірне обмеження.
1	БД електронного ресурсу з книгами (рис. 4.1). **P.K.** – Користувачі→Логін; Книги→Назва. **I.** – Користувачі→Місце народження; Користувачі→Прізвище + Ім`я + По-батькові; Користувачі→Рейтинг. **CHECK** – Посилання→Електронний адрес книги повинен розпочинатися з 'http://www.' ; Значення Користувачі→Рейтинг повинне бути у межах |0; 5|, при цьому тип даних допускає лише два розряди після коми; для Книги→УДК забезпечити формат:

	2 довільні букви + '.' + 3 цифри. **Формула** – для таблиці Користувачі створити поле, що об`єднує в собі Прізвище, Ім`я та По-батькові; для таблиці Користувачі створити поле, що виводило б кількість років користувачу, як різницю між поточною системною датою та датою народження.
2	БД родинного дерева (рис. 4.2). **Р.К.** – Фамільне дерево→Прізвище + Ім`я; Сімейні супутники→ Прізвище + Ім'я; Фамільні цінності→Назва цінності. **І.** – Сімейні супутники →Дата народження; Фамільне дерево→ Дата смерті. **U.I.** – Фамільне дерево→Номер кредитної картки. **CHECK** – Сімейні супутники →Прізвище не може закінчуватися на 'ов' чи 'ова' ; Сімейні супутники →Дата народження та Сімейні супутники →Дата смерті не можуть бути більшими за поточну дату; для Фамільні цінності→Код у каталозі цінностей забезпечити формат: *1 буква: A, M чи Z + 5 цифр + 2 довільні букви.* **Формула** – для таблиці Фамільні цінності створити поле, що міститиме певний коефіцієнт, який розраховується за такою формулою: SIN(*мінімальна вартість*) + COS(*максимальна вартість*); для таблиці Фамільне дерево створити поле, що об`єднує в собі Прізвище та Ім'я.
3	БД студентів, що навчаються на кафедрі КСА (рис. 4.3). **Р.К.** – Студенти→Прізвище + Ім`я + По-батькові, Місто→Місто; Заборгованості→Назва предмету. **І.** – Студенти→Загальний рейтинг, Закінчений заклад середньої освіти→П.І.Б директора школи. **CHECK** – для Студенти→Номер студентського квитка забезпечити формат: 'A' *+ 8 цифр + довільна буква, окрім S* ; Студенти→Електронний адрес повинен обов`язково містити символ '@' ; різниця між Студенти→Дата поступлення та Студенти→Дата народження повинна бути не меншою за 16 років. **Формула** – для таблиці Студенти створити поле, що виводило б значення Номер студентського квитка + '-' + *рік поступлення* ; для таблиці Студенти створити поле, що об`єднує в собі Прізвище, Ім`я та По-батькові.

4	БД співробітників кафедри КСА (рис. 4.4). **P.K.** – Співробітники→Прізвище + Ім`я + По-батькові; Дисципліни→Назва дисципліни. **I.** – Співробітники→Трудовий стаж; **U.I.** – Співробітники→Серія та номер паспорту; Дисципліни→Код. **CHECK** – значення Співробітники→Трудовий стаж не може бути більшим за різницю між *поточною датою* та *датою народження*; значення Дисципліни→Номер семестру повинно бути у межах [1; 10]; **Формула** – для таблиці Співробітники створити поле, що об`єднує в собі Прізвище, Ім`я та По-батькові; для таблиці Співробітники створити поле, що містить певний код, який обчислюється за такою формулою: SQRT(*рік народження*) + COS(*число місяця*) - EXP(*№ місяця*).
5	БД аптечних установ м.Новосілки (рис. 4.5). **P.K.** – Співробітники→ Прізвище + Ім`я + По-батькові; Аптечна установа→Назва; Перелік лікарств→Назва. **I.** – Співробітники→Серія та номер паспорту. **U.I.** – Співробітники→Ідентифікаційний номер; Перелік лікарств→Код міністерства. **CHECK** –для Співробітники→ Ідентифікаційний номер забезпечити формат: '10 цифр' ; Перша буква Перелік лікарств→ Код міністерства повинна співпадати з першою буквою Перелік лікарств→Назва. **Формула** – для таблиці Співробітники створити поле, що об`єднує в собі Прізвище, Ім`я та По-батькові; для таблиці Співробітники створити поле, що складатиметься з: прізвища + 2-х останніх цифр ідентифікаційного номера; для таблиці Аптечна установа створити поле, що міститиме значення поля Адрес вебсторінки, але з усіченими на початку та на кінці випадковими пробілами.
6	БД електронного ресурсу з книгами (рис. 4.1). **P.K.** – Користувачі→Прізвище+Ім`я+Дата народження; Книги→Назва. **U.I.** – Користувачі→Логін. **I.** – Користувачі→Місце народження; Користувачі→Місце

	проживання. **CHECK** – Користувачі →Прізвище не може закінчуватися на 'ін' чи 'іна' ; у полі Користувачі→Місце проживання допускається ввід лише таких значень: 'Львів', 'Київ', 'Донецьк' та 'Одеса' . **Формула** – для таблиці Книги створити поле, що об`єднує в собі Авторів та Назву; для таблиці Користувачі створити поле, що виводило б певне число з плаваючою комою на основі Дати народження, та яке формується за такою формулою: COS(*рік народження*) + SIN(*номер місяця у році*) + TAN(*день місяця*), причому *рік*, *місяць* та *день* відображають значення у градусах.
7	БД родинного дерева (рис. 4.2). **P.K.** – Фамільне дерево→Прізвище + Ім`я; Сімейні супутники → Прізвище + Ім`я; Фамільні цінності→Назва цінності. **I.** – Сімейні супутники →Місце народження; Фамільне дерево→ Номер кредитної картки. **U.I.** – Фамільні цінності →Код у каталозі цінностей. **CHECK –** для Фамільні дерево →Номер кредитної картки забезпечити формат: *4 цифри + пробіл + 4 цифри + пробіл + 4 цифри + пробіл + 4 цифри* ; значення Фамільні цінності→Максимальна вартість не може бути меншою за значення Фамільні цінності→ Мінімальна вартість, а значення Фамільні цінності→Орієнтовна вартість повинно знаходитися, відповідно, між максимальною та мінімальною вартостями. **Формула** – для таблиці Фамільне дерево створити поле, що відображатиме значення у такому форматі: Ім'я + *пробіл* + Прізвище + ' народився ' + *номер дня у році* + ' дня ' + *рік* + ' року Божого ' ; для таблиці Фамільні цінності створити поле, що міститиме чотирисимвольний код співзвучності назв для поля Назва цінності (функція SOUNDEX()).
8	БД студентів, що навчаються на кафедрі КСА (рис. 4.3). **P.K.** – Студенти→Номер студентського квитка; Закінчений заклад середньої освіти→Назва закладу; Заборгованості→Назва предмету. **U.I.** – Область →Код області. **I.** – Студенти→Загальний рейтинг, Закінчений заклад середньої освіти→телефон. **CHECK** – для Область →Код області забезпечити формат вводу:

	перша буква поля Область + 2 цифри ; Студенти→Номер студентського не повинен розпочинатися з '99'. **Формула** – для таблиці Студенти створити поле, що відображало б у скільки років студент поступив на кафедру (як різницю між роком поступлення та роком народження); для таблиці Закінчений заклад середньої освіти створити поле, що відображатиме значення у такому форматі: 'Директор' + *пробіл* + Назва закладу + *пробіл* + П.І.Б. директора закладу.
9	БД співробітників кафедри КСА (рис. 4.4). **Р.К.** – Співробітники→Серія та номер паспорту; Дисципліни→Назва дисципліни. **І.** – Дисципліни→Номер семестру. **U.I.** – Співробітники→Прізвище + Ім'я + По-батькові; Дисципліни→Код. **CHECK** – для Співробітники→Серія та номер паспорту забезпечити формат вводу: *2 букви + пробіл + 6 цифр* ; у полі Співробітники →Ім`я допускається ввід лише таких імен: 'Андрій', 'Оля', 'Володимир' та 'Оксана'. **Формула** – для таблиці Співробітники створити поле, що виводить лише *номер паспорту*; для таблиці Співробітники створити поле, що виводить *рік* у якому співробітник влаштувався на роботу, як різницю між *поточним роком* та значенням трудового стажу.
10	БД аптечних установ м.Новосілки (рис. 4.5). **Р.К.** – Співробітники→ Ідентифікаційний номер; Аптечна установа→Назва; Перелік лікарств→Код міністерства. **І.** – Співробітники→Дата народження. **U.I.** – Співробітники→Прізвище + Ім'я + По-батькові; Перелік лікарств→Назва. **CHECK** – для Співробітники→Серія та номер паспорту забезпечити формат вводу: *2 букви + пробіл + 6 цифр* ; у полі Співробітники →Ім`я допускається ввід лише таких імен: 'Василь', 'Іван', 'Галина' та 'Олександра'. **Формула** – для таблиці Співробітники створити поле, що виводить лише *серію паспорту* ; для таблиці Співробітники створити поле, що складатиметься з: Прізвища та Імені + *2-х перших цифр ідентифікаційного номера* .

11	БД електронного ресурсу з книгами (рис. 4.1). **Р.К.** – Користувачі→Логін; Книги→Назва. **І.** – Користувачі→Місце проживання; Користувачі→Прізвище + Ім`я + По-батькові; Книги→УДК. **CHECK** – Користувачі →Логін не повинен розпочинатися з цифри та бути меншим за 6 символів; значення Користувачі→ Рейтинг повинно знаходитися у межах |1; 10|. **Формула** – для таблиці Книги створити поле, що об`єднує в собі Авторів та Назву; для таблиці Користувачі створити поле, що виводить підряд перші букви полів Прізвище, Ім`я, По-батькові, а потім виводить символ '-' та значення Рейтинг.
12	БД родинного дерева (рис. 4.2). **Р.К.** – Фамільне дерево→Прізвище + Ім`я; Сімейні супутники → Прізвище + Ім`я; Фамільні цінності→Код у каталозі цінностей. **І.** – Сімейні супутники →Місце народження; Фамільне дерево→ Місце народження. **U.I.** – Фамільні цінності →Назва цінності. **CHECK** – у полі Фамільне дерево →Місце смерті допускається ввід лише таких значень: 'с. Стрілки', 'с. Підкопанс', 'с. Ярів' ; поле Фамільне дерево→Дата народження не може бути більшим за поточну дату. **Формула** – для таблиці Фамільні цінності створити поле, що виводитиме таке значення: 'Назва цінності: ' + Назва цінності; для таблиці Фамільне дерево створити поле, що виводитиме числовий код згідно такого арифметичного виразу: *Рік народження + номер місяця народження + день місяця народження.*
13	БД студентів, що навчаються на кафедрі КСА (рис. 4.3). **Р.К.** – Групи→Назва групи + Номер групи; Місто→Місто; Студенти→Номер студентського квитка. **U.I.** – Студенти→Прізвище + Ім`я + По-батькові. **І.** – Область→Код області; Закінчений заклад середньої освіти→П.І.Б директора школи. **CHECK** – для Закінчений заклад середньої освіти→Телефон забезпечити формат (код міста + телефон): '(' + *3 цифри* + ') ' + *3 цифри* + '-' + *2 цифри* + '-' + *2 цифри* ; у полі Студенти →Ім`я допускається ввід лише таких імен: 'Світлана', 'Петро', 'Оля', 'Тарас', 'Василь', 'Антон' .

	Формула – для таблиці Студенти створити поле, що виводить підряд перші букви полів Прізвище, Ім`я, По-батькові, а потім виводить символ '-' та значення Дата поступлення ; для таблиці Закінчений заклад середньої освіти створити поле, яке виводитиме лише сам номер телефону (7 цифр), без розділових рисок та коду області.
14	БД співробітників кафедри КСА (рис. 4.4). **Р.К.** – Співробітники→Прізвище + Ім`я + По-батькові; Посада→Назва посади. **І.** – Співробітники→Дата народження; Дисципліни→ Номер дисципліни. **U.I.** – Співробітники→Серія та номер паспорту. **CHECK** – для Співробітники→Серія та номер паспорту забезпечити формат вводу: *2 букви + пробіл + 6 цифр* ; Співробітники →Прізвище не може закінчуватися на 'вий' чи 'ва' . **Формула** – для таблиці Співробітники створити поле, що виводить суму таких значень: Трудовий стаж + *рік народження*; для таблиці Співробітники створити поле, що виводить модифікацію значення Прізвище, тобто останню букву прізвища замінює на сусідній символ, згідно кодової таблиці (ASCII чи UNICODE).
15	БД аптечних установ м.Новосілки (рис. 4.5). **Р.К.** – Співробітники→ Серія та номер паспорту; Вулиця → Назва вулиці; Зона впливу→Назва. **І.** – Перелік лікарств→Код міністерства. **U.I.** – Співробітники→Прізвище + Ім`я + По-батькові; Співробітники→Ідентифікаційний номер. **CHECK** – Співробітники→ Ідентифікаційний номер не може закінчувати двома нулями; для Перелік лікарств→ Код міністерства забезпечити формат вводу: *2 довільні букви, окрім М і П* + '-' + *3 цифри* + '-' + *2 цифри* . **Формула** – для таблиці Співробітники створити поле, що виводить таке значення: 'Ідент. №' + Ідентифікаційний номер; для таблиці Співробітники створити поле, що виводило б певне число з плаваючою комою на основі Дати народження, та яке формується за такою формулою: SQRT(*рік народження*) + TAN(*номер місяця у році*) + COS (*день місяця*), причому *місяць* та *день* відображають значення у радіанах.

16	БД електронного ресурсу з книгами (рис. 4.1). **Р.К.** – Користувачі→Логін; Книги→Назва. **І.** – Користувачі→Місце народження; Користувачі→Прізвище + Ім`я + По-батькові; Користувачі→Рейтинг. **CHECK** – Посилання→Електронний адрес книги повинен обов`язково містити стрічку 'www.' та '.ua' ; Значення Користувачі→Рейтинг повинне бути у межах |1; 10|, при цьому тип даних допускає лише один розряд після коми; для Книги→ УДК забезпечити формат: *1 кирилична буква + 2 цифри* + '.' + *3 цифри*. **Формула** – для таблиці Користувачі створити поле, що об`єднує в собі Прізвище, Ім`я та По-батькові; для таблиці Користувачі створити поле, що виводило б кількість років користувачу, як різницю між поточною системною датою та датою народження.
17	БД родинного дерева (рис. 4.2). **Р.К.** – Фамільне дерево→Номер Кредитної Картки; Сімейні супутники→ Прізвище + Ім`я; Фамільні цінності→Назва цінності. **І.** – Сімейні супутники →Місце народження; Фамільне дерево→ Дата смерті. **U.I.** – Фамільне дерево→Прізвище + Ім`я. **CHECK** – Сімейні супутники →Прізвище не може починатися на букву 'А' та закінчуватися на 'ін' чи 'іна' ; Сімейні супутники → Дата народження та Сімейні супутники→Дата смерті не можуть бути більшими за поточну дату; для Фамільні цінності→Код у каталозі цінностей забезпечити формат: *1 довільна буква, окрім I'* + *3 цифри* + '/' + *2 довільні букви*. **Формула** – для таблиці Фамільні цінності створити поле, що міститиме певний коефіцієнт, який розраховується за такою формулою: COS(*мінімальна вартість*) + SIN(*максимальна вартість*); для таблиці Сімейні супутники створити поле, що об`єднує в собі Прізвище та Ім`я.
18	БД студентів, що навчаються на кафедрі КСА (рис. 4.3). **Р.К.** – Студенти→Прізвище + Ім`я + По-батькові; Місто→Місто; Область→Область. **І.** – Студенти→Загальний рейтинг; Закінчений заклад середньої освіти→П.І.Б директора школи.

	CHECK – для Студенти→Номер студентського квитка забезпечити формат: *довільна буква, окрім S, E, L + 8 цифр* (набір цифр не може містити цифру 7) ; Студенти→Електронний адрес повинен бути розміщеним на українському ресурсі, тобто містити підстрічку '.ua' ; обмежити поле Студенти→Дата народження так, щоб вік теперішнього студента був не меншим за 18 років. **Формула –** для таблиці Студенти створити поле, що виводило б значення: Прізвище + '-' + *кількість років* + 'р.' ; для таблиці Студенти створити поле, що об'єднує в собі Прізвище, Ім'я та По-батькові.
19	БД співробітників кафедри КСА (рис. 4.4). **P.K.** – Співробітники→Прізвище + Ім'я + По-батькові; Дисципліни→Назва дисципліни. **I.** – Співробітники→Дата народження. **U.I.** – Співробітники→Серія та номер паспорту; Дисципліни→Код. **CHECK** – значення Співробітники→Трудовий стаж не може бути більшим за різницю між *поточною датою* та *датою народження + 15 років*; для Дисципліни→ Код забезпечити формат вводу: *буква Б, М або С* + '.' + *3 цифри* ; **Формула –** для таблиці Співробітники створити поле, що об'єднує в собі Прізвище, Ім'я та По-батькові; для таблиці Співробітники створити поле, що містить певний код, який обчислюється за такою формулою: SIN(*рік народження*) + TAN(*число місяця*) – COS(*№ місяця*).
20	БД аптечних установ м.Новосілки (рис. 4.5). **P.K. –** Співробітники→ Прізвище + Ім'я + По-батькові; Аптечна установа→Назва; Перелік лікарств→Назва. **I.** – Співробітники→ Ідентифікаційний номер. **U.I.** – Співробітники→ Серія та номер паспорту; Перелік лікарств→Код міністерства. **CHECK –**для Співробітники→ Ідентифікаційний номер забезпечити формат: '10 цифр' ; для Перелік лікарств→ Код міністерства забезпечити формат вводу: *перша буква назви лікарства + 3 цифри* . **Формула** – для таблиці Співробітники створити поле, що об'єднує в собі Прізвище, Ім'я та По-батькові; для таблиці

	Співробітники створити поле, що складатиметься з: Прізвище + *2-ві перші цифри ідентифікаційного номера*; для таблиці Аптечна установа створити поле, що міститиме значення поля: 'web: ' + Адрес вебсторінки з усіченими на початку та на кінці випадковими пробілами.
21	БД електронного ресурсу з книгами (рис. 4.1). **Р.К.** – Користувачі→Прізвище + Ім`я + Дата народження; Книги → Назва. **U.I.** – Користувачі→Логін. **І. –** Користувачі→Місце народження; Користувачі→Місце проживання. **CHECK** – Користувачі →Прізвище не може починатися на букву 'Ю' чи 'Я' ; у полі Користувачі→Місце народження допускається ввід лише таких значень: 'Харків', 'Вінниця', 'Дніпропетровськ' та 'Ужгород' . **Формула** – для таблиці Книги створити поле, що об`єднує в собі Авторів та Назву; для таблиці Користувачі створити поле, що виводило б певне число з плаваючою комою на основі Дати народження, та яке формується за такою формулою: SQRT(*рік народження*) + COS(*номер місяця у році*) + SIN(*день місяця*), причому *місяць* та *день* відображають значення у радіанах.
22	БД родинного дерева (рис. 4.2). **Р.К.** – Фамільне дерево→Прізвище + Ім`я; Сімейні супутники → Прізвище + Ім`я; Фамільні цінності→Назва цінності. **І.** – Сімейні супутники →Місце народження; Фамільне дерево→ Номер кредитної картки. **U.I. –** Фамільні цінності →Код у каталозі цінностей. **CHECK** – для Фамільні дерево →Номер кредитної картки забезпечити формат: *4 цифри* + '-'+ *4 цифри* + '-'+ *4 цифри* + '-'+ *4 цифри* ; значення Фамільні цінності→Максимальна вартість не може бути меншою за значення Фамільні цінності→ Мінімальна вартість. **Формула** – для таблиці Сімейні супутники створити поле, що відображатиме значення у такому форматі: Ім`я + *пробіл* + Прізвище + ' народ. ' + *номер дня у році* + ' дня ' + *рік* + 'р. від різдва Христового' ; для таблиці Фамільні цінності створити поле, що міститиме чотирисимвольний код

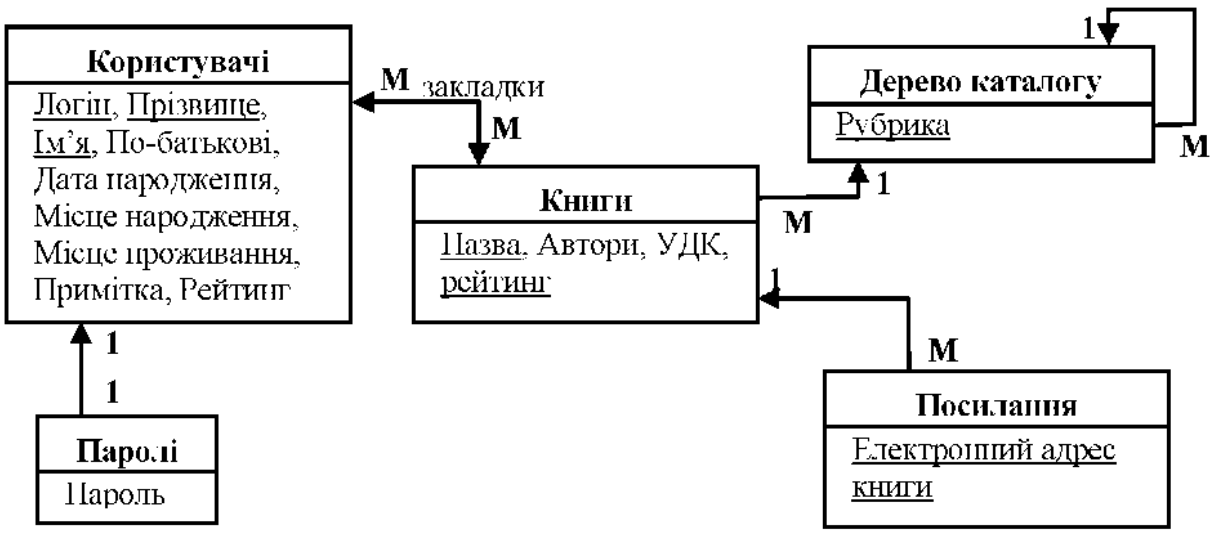
	співзвучності назв для поля Назва цінності (функція SOUNDEX()).
23	БД студентів, що навчаються на кафедрі КСА (рис. 4.3). **Р.К.** – Студенти→Номер студентського квитка; Закінчений заклад середньої освіти→Назва закладу; Заборгованості→Назва предмету. **U.I.** – Область →Код області. **I.** – Студенти→Загальний рейтинг; Закінчений заклад середньої освіти→телефон. **CHECK** – для Область →Код області забезпечити формат вводу: *перші дві букви поля Область* + *3 цифри* ; Студенти→Номер студентського не повинен закінчуватися на '00'. **Формула** – для таблиці Студенти створити поле, що відображало б у скільки років студент поступив на кафедру (як різницю між роком поступлення та роком народження); для таблиці Студенти створити поле, що відображатиме значення у такому форматі: 'Студент' + *пробіл* + Прізвище + *пробіл* + Ім'я.
24	БД співробітників кафедри КСА (рис. 4.4). **Р.К.** – Співробітники→Серія та номер паспорту; Дисципліни→Назва дисципліни. **I.** – Дисципліни→Номер семестру. **U.I.** – Співробітники→Прізвище + Ім`я + По-батькові; Дисципліни→Код. **CHECK** – для Співробітники→Серія та номер паспорту забезпечити формат вводу: *2 великі кириличні букви* + '-' + *6 цифр* ; у полі Співробітники →Ім'я допускається ввід лише таких імен: 'Василь', 'Анна', 'Ірина', 'Андрій' та 'Юрій'. **Формула** – для таблиці Співробітники створити поле, що виводить лише *серію паспорту*; для таблиці Співробітники створити поле, що виводить *рік* у якому співробітник влаштувався на роботу, як різницю між *поточним роком* та значенням трудового стажу.
25	БД аптечних установ м.Новосілки (рис. 4.5). **Р.К.** – Співробітники→ Ідентифікаційний номер; Аптечна установа→Назва; Перелік лікарств→Код міністерства. **I.** – Співробітники→Дата народження.

	U.I. – Співробітники→Прізвище + Ім'я + По-батькові; Перелік лікарств→Назва. **CHECK** – для Співробітники→Серія та номер паспорту забезпечити формат вводу: *2 довільні букви + 2 пробіли + 6 цифр* ; у полі Співробітники →Ім`я допускається ввід лише таких імен: 'Галина', 'Олександр', 'Андріян' та 'Ілона'. **Формула** – для таблиці Співробітники створити поле, що виводить лише *номер паспорту* ; для таблиці Співробітники створити поле, що складатиметься з: Прізвища та Імені + 2-х останніх цифр ідентифікаційного номера.
26	БД електронного ресурсу з книгами (рис. 4.1). **P.K.** – Користувачі→Логін; Книги→Назва. **I.** – Користувачі→Місце проживання; Користувачі→Прізвище + Ім'я + По-батькові; Книги→УДК. **CHECK** – Користувачі →Логін не повинен розпочинатися з цифри чи літери F та бути меншим за 8 символів; значення Користувачі→ Рейтинг повинно знаходитися у межах |1; 5|. **Формула** – для таблиці Книги створити поле, що об'єднує в собі Авторів та Назву; для таблиці Користувачі створити поле, що виводить підряд перші букви полів Прізвище, Ім'я, По-батькові, з крапками на кінці букв, а потім виводить символи '--' та значення Рейтинг.
27	БД родинного дерева (рис. 4.2). **P.K.** – Фамільне дерево→Прізвище + Ім`я; Сімейні супутники → Прізвище + Ім`я; Фамільні цінності→Код у каталозі цінностей. **I.** – Сімейні супутники →Місце народження; Фамільне дерево→ Місце народження. **U.I.** – Фамільні цінності →Назва цінності. **CHECK** – у полі Фамільне дерево →Місце народження допускається ввід лише таких значень: 'м. Львів', 'с. Зашків', 'с. Зарудці' ; поле Сімейні супутники→Дата народження не може бути більшим за поточну дату. **Формула** – для таблиці Фамільні цінності створити поле, що виводитиме таке значення: 'Вартість: ' + Орієнтовна вартість; для таблиці Сімейні супутники створити поле, що виводитиме числовий код згідно такого арифметичного виразу: *Рік народження – номер місяця народження – день місяця народження.*

28	БД студентів, що навчаються на кафедрі КСА (рис. 4.3). **P.K.** – Групи→Назва групи + Номер групи, Місто→Місто; Студенти→Номер студентського квитка. **U.I.** – Студенти→Прізвище + Ім`я + По-батькові. **I.** – Область→Код області; Закінчений заклад середньої освіти→П.І.Б директора школи. **CHECK** – для Закінчений заклад середньої освіти→Телефон забезпечити формат: '8' + пробіл + *3 цифри* + '-' + *3 цифри* + '-' + *2 цифри* + '-' + *2 цифри* ; у полі Студенти →Ім`я допускається ввід лише таких імен: 'Анатолій', 'Гаврило', 'Юля', 'Іван', 'Віталій', 'Уляна' . **Формула** – для таблиці Студенти створити поле, що виводить підряд перші букви полів Прізвище, Ім`я, По-батькові, а потім виводить символ '-' та значення *року народження* ; для таблиці Закінчений заклад середньої освіти створити поле, яке виводитиме номер телефону (11 цифр) без розділових знаків.
29	БД співробітників кафедри КСА (рис. 4.4). **P.K.** – Співробітники→Прізвище + Ім`я + По-батькові; Посада→Назва посади. **I.** – Співробітники→Дата народження; Дисципліни→ Номер дисципліни. **U.I.** – Співробітники→Серія та номер паспорту. **CHECK** – для Співробітники→Серія та номер паспорту забезпечити формат вводу: *2 букви* + '-' + *6 цифр* ; Співробітники →Прізвище не може починатися на 'Ма' чи 'Па' . **Формула** – для таблиці Співробітники створити поле, що виводить суму таких значень: Трудовий стаж + *рік народження* −20; для таблиці Співробітники створити поле, що виводить модифікацію значення Прізвище, тобто першу букву прізвища замінює на сусідній символ, згідно кодової таблиці (ASCII чи UNICODE).
30	БД аптечних установ м.Новосілки (рис. 4.5). **P.K.** – Співробітники→ Серія та номер паспорту; Вулиця → Назва вулиці; Зона впливу→Назва. **I.** – Перелік лікарств→Код міністерства. **U.I.** – Співробітники→Прізвище + Ім`я + По-батькові; Співробітники→Ідентифікаційний номер.

	CHECK – Співробітники→ Ідентифікаційний номер не може починатися двома дев'ятками; для Перелік лікарств→ Код міністерства забезпечити формат вводу: *2 цифри, окрім п'ятірок* + '-' + *3 латинські букви* + '.' + *3 цифри* . **Формула** – для таблиці Співробітники створити поле, що виводить таке значення: 'Ідент. №' + Ідентифікаційний номер; для таблиці Співробітники створити поле, що виводило б певне число з плаваючою комою на основі Дати народження, та яке формується за такою формулою: TAN(*рік народження*) + SIN(*номер місяця у році*) + COS (*день місяця*), причому *рік*, *місяць* та *день* відображають значення у градусах.

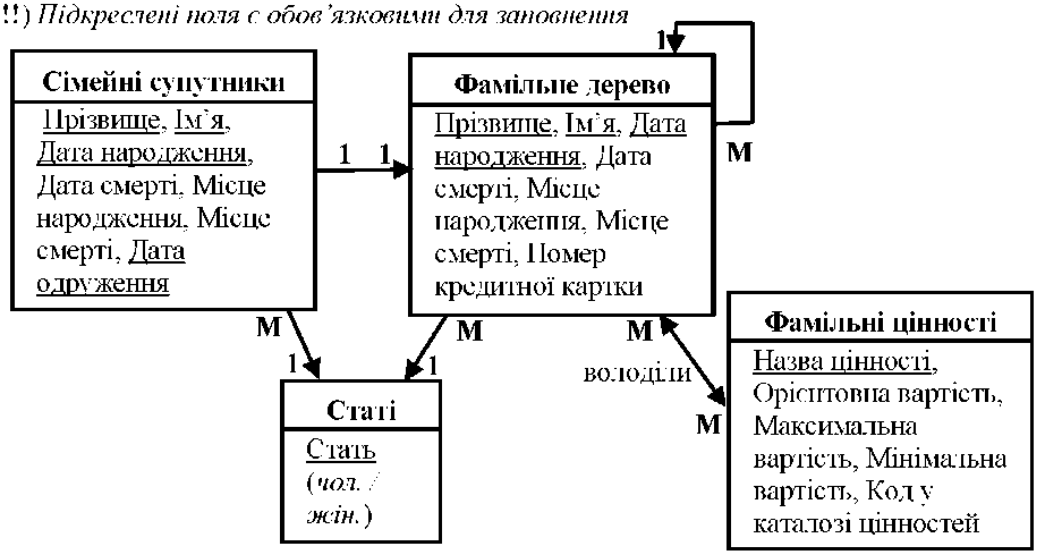
(!!!) *Підкреслені поля є обов'язковими для заповнення*



Опис: Дерево каталогу представляє собою логічне дерево з розкриваючими у глибину пунктами (уявіть дерево з плюсами/мінусами біля узагальнюючих пунктів для розгортання/згортання листків). Кожен користувач каталогу може мати закладки на потрібні книжки. Паролі користувачів зберігаються в окремій таблиці з метою конфіденційності. Книги мають посилання на різні місця їх збереження.

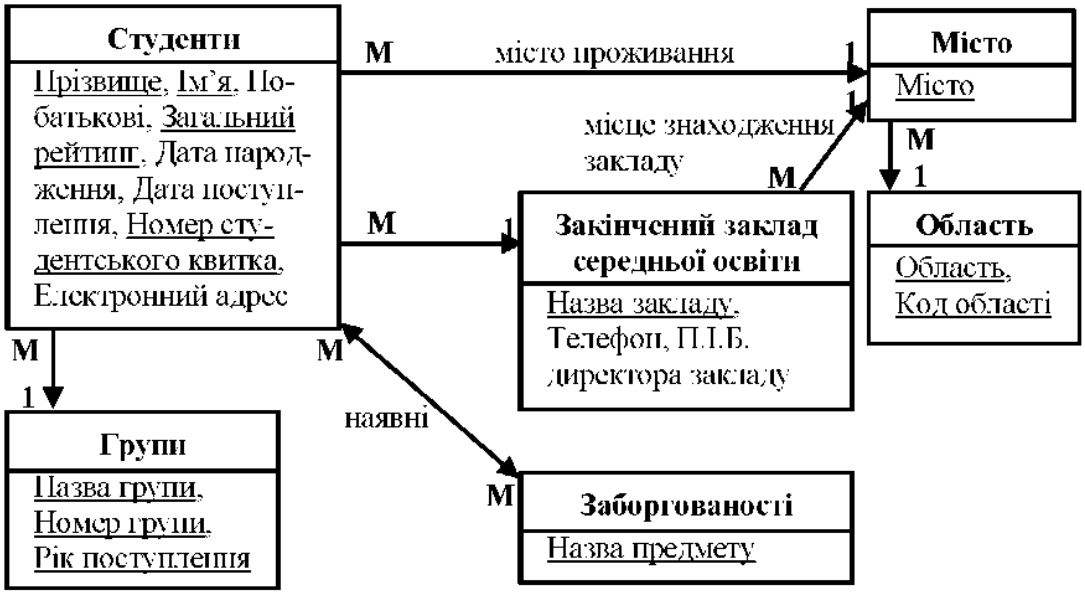
Рис. 4.1. База даних електронного ресурсу з книгами

(!!!) *Підкреслені поля є обов'язковими для заповнення*



Опис: Фамільне дерево представляє собою логічне дерево з розкриваючими у глибину пунктами (уявіть дерево з плюсами/мінусами біля узагальнюючих пунктів для розгортання/згортання листків). Кожен член роду має може

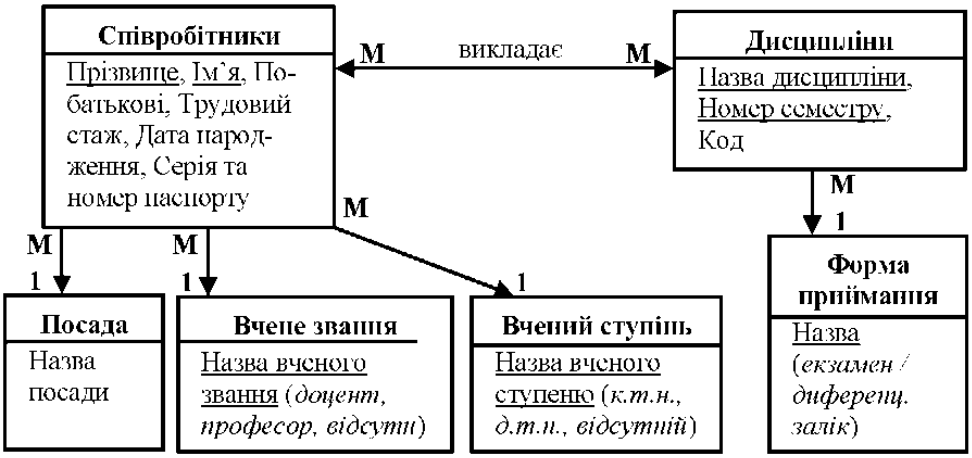
(!!!) *Підкреслені поля є обов'язковими для заповнення*



Опис: Студенти приїхали навчатися на кафедрі КСА з різних міст України. При цьому вони отримали середню освіту вдома (у рідному місті), але не обов'язково. Студенти розбиті по групах. Для кожного студента ведеться статистика наявних академзаборогованостей по різних предметах.

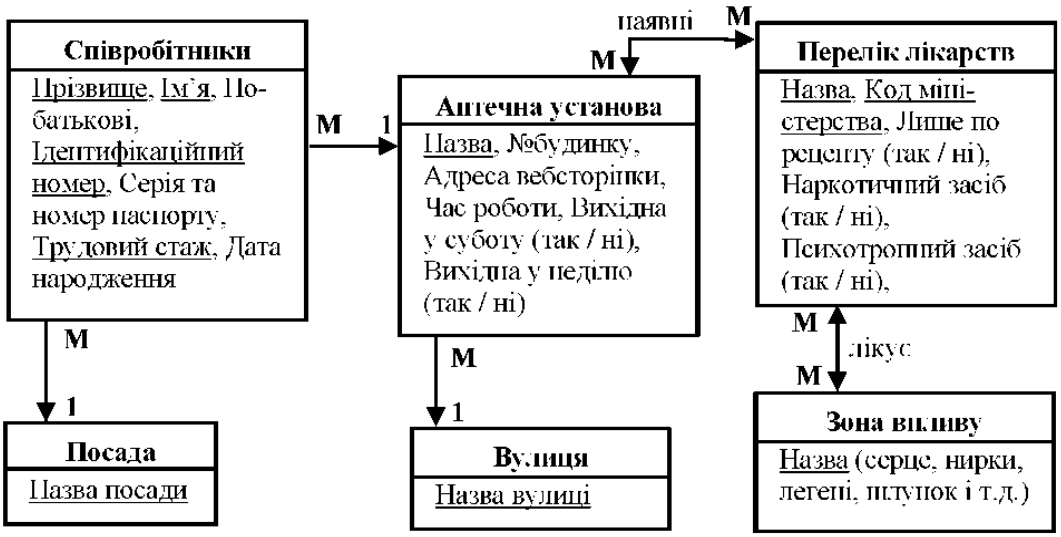
Рис. 4.3. База даних студентів, що навчаються на кафедрі КСА

(!!!) *Підкреслені поля є обов'язковими для заповнення*



Опис: Співробітники кафедри (професор, доцент, ст.викладач, асистент, інженер, лаборант і т.д) можуть викладати студентам різні дисципліни. При цьому той самий предмет (лекції, практичні, лабораторні) можуть викладати різні викладачі.

(!!!) *Підкреслені поля є обов'язковими для заповнення*



Опис: для полегшення пошуку в місті Новосілки аптеки групуються за вулицями. Кожна аптека має у наявності певну частину лікарських засобів, а для кожного лікарського засобу, відповідно, можна переглянути у яких аптеках він є у наявності. Кожен засіб може використовуватися для лікування різних людських органів.

Рис. 4.5. База даних аптечних установ м. Новосілки

5. Зміст звіту

1. Номер і назва лабораторної роботи, із зазначенням її виконавця.
2. Мета роботи.
3. Завдання до лабораторної роботи.
4. Короткі теоретичні відомості, що необхідні для виконання лабораторної роботи (не більше 3 сторінок).
5. Логічні вирази для усіх заданих у БД перевірних обмежень CHECK та значення формул для обчислювальних полів.
6. Діаграма спроектованої бази даних.
7. Висновки.

6. Контрольні запитання

1. Що представляє собою продукт Microsoft SQL Server 2005?
2. Основні відмінності між різними редакціями MS SQL Server 2005.
3. Назвіть два режими перевірення автентичності користувача при підключенні до MS SQL Server 2005, та які є відмінності між ними.
4. Які існують основні системні бази даних MS SQL Server 2005; для чого вони потрібні та їх короткий опис?
5. Що таке файлові групи у базах даних MS SQL Server 2005?
6. Назвіть основні режими відновлення баз даних та їх відмінності.
7. Що таке схеми у базах даних MS SQL Server 2005?
8. Що представляє собою таблиця бази даних?
9. Назвіть основні типи даних та їхні характеристики.
10. Що таке автоінкрементні поля?
11. Що представляють собою обчислювальні стовпці та яку інформацію вони можуть відображати?
12. Що таке GUID-ідентифікатори?
13. Які існують типи ключів?
14. Яка існує відмінність між унікальними ключами та унікальними індексами?
15. Яка відмінність між кластеризованими та некластеризованими індексами?
16. Згідно яких критеріїв вибирають стовпці для індексування?
17. Згідно яких критерії роблять індекси кластерними?
18. Які існують типи обмежень?
19. Чи потрібно для стовпця з первинним ключем створювати додатково індекс?
20. Чи потрібно для стовпця з унікальним ключем створювати індекс?
21. Що представляє собою перевірне обмеження CHECK?
22. Назвіть символи-шаблони логічного оператора LIKE та наведіть приклади їх застосування?
23. Для чого призначені зовнішні ключі?
24. Як реалізується відношення «один до одного»?
25. Як реалізується відношення «один до багатьох»?
26. Як реалізується відношення «багато до багатьох»?

27.Як реалізується зворотне (рекурсивне) відношення?

28.Що таке діаграми баз даних?

7. Список літератури

1. Роберт Виейра. Программирование баз данных Microsoft SQL Server 2005. Базовый курс.: Пер. с англ. - М.: ООО «И.Д. Вильямс». - 2007. - 832 с.

2. Ростислав Михеев. MS SQL Server 2005 для администраторов. -СПб.: БХВ-Петербург. - 2007. - 544 с.

3. Пол Нильсен. Microsoft SQL Server 2005. Библия пользователя.: Пер. с англ. - М.: ООО «И.Д. Вильямс». - 2008. - 1232 с.

4. Microsoft SQL Server 2005. Реализация и обслуживание. Учебный курс Microsoft / Пер. с англ. - М.: «Русская Редакция», СПб.:

«Питер». - 2007. - 768 с.

5. Уильям Р. Станек. Microsoft SQL Server 2005. Справочник администратора / Пер. с англ. - М.: «Русская Редакция». - 2006. - 544 с.