Міністерство освіти і науки України
Рівненський державний гуманітарний університет

ВОЛОДИМИР СІМАШКО

АДМІНІСТРУВАННЯ ТА ПРОГРАМУВАННЯ В КОРПОРАТИВНИХ БАЗАХ ДАНИХ. ПРАКТИКУМ

Навчальний посібник

Рівне – 2023

УДК 004.65 (075.8); 004.415.2
С 37

Друкується за рішенням Вченої ради Рівненського державного гуманітарного університету РДГУ *(протокол № 2 від 30.03.2023р.).*

Рецензенти: Юськів Б.М., доктор політичних наук, професор кафедри економіки та управління бізнесом РДГУ;
Шпортько О. В., кандидат технічних наук, доцент кафедри інформаційних систем та обчислювальних методів МЕГУ.

С 37 Сімашко В. Адміністрування та програмування в корпоративних базах даних. Практикум: навчальний посібник . – Рівне: РДГУ, 2023. – 84 с.

Навчальний посібник призначений для самостійної підготовки та проведення лабораторних занять, виконання яких сприятиме поглибленому вивченню та засвоєнню теоретичного матеріалу, формуванню навичок самостійної роботи. Кінцевою метою вивченню курсу є набуття здобувачами вищої освіти навичок із адміністрування та програмування корпоративних баз даних. Призначений для здобувачів вищої освіти спеціальностей 051 Економіка, 075 Маркетинг, викладачів та всіх, хто прагне оволодіти технологіями створення та використання баз даних.

Ухвалено кафедрою економіки та управління бізнесом РДГУ (протокол № 1 від 23.01.2023р.).

Зміст

ВВЕДЕННЯ

Завданням даного курсу лабораторних робіт є закріплення студентами теоретичних знань і отримання практичних навичок встановлення і адміністрування сервера РСКБД на прикладі сервера InterBase / FireBird .

РСКБД InterBase була розроблена у США одноіменною фірмою як безкоштовний програмний продукт, та широко використовувалася у вищих навчальних закладах з метою освоєння студентами навичок роботи із реляційними базами даних, та вивчення мови SQL. По причині простоти, безкоштовності, підтримки всіх об'єктів і можливостей стандарту SQL ця БД набула широкого розповсюдження не лише у вищих навчальних закладах США та Європи, а також широко застосовується для розробки прикладних інформаційних систем в СНД та Східній Європі. На момент виходу версії InterBase 4 (1994 рік) ця система була досить потужною, конкурувала навіть із такими системами, як MicroSoft SQL Server та SyBase, і була використана в деяких військових проектах США. Після того, як фірма Borland викупила всі права на InterBase і випустила його платні версії (як окремі, так і інтегровані в середовища візуальної розробки Delphi та C Builder), розробники цієї БД зпочатку відкрито опублікували тексти всіх програм, бібліотек і драйверів останньої безкоштовної версії InterBase, а згодом почали розробку і підтримку РСКБД FireBird – «нащадка» InterBase. Російські програмісти також розвивають свою версію InterBase, яка носить назву Yaffil. На сьогодні FireBird є найбільш вдалим «нащадком» InterBase, оскільки переважає російський Yaffil та InterBase фірми Borland по потужності, надійності і якості підтримки. FireBird – єдина із цих трьох «клонів», яка здатна використовувати можливості багатопроцесорних систем. Всі ці РСКБД (платна InterBase та безкоштовні FireBird і Yaffil) умовно сумісні (файли баз даних досить нескладно переносити з одного сервера на інший). Для учбового процесу дуже важливі такі переваги FireBird: здатність нормально працювати на малопотужних комп'ютерах, підтримка усіх можливостей стандарту ANSI SQL-92, та багатьох додаткових можливостей стандарту SQL:2003 (вкладені SQL-запити, потужний набір SQL-функцій, системні тригери, збережені процедури, тощо). Важливо також, що візуальні компоненти InterBase Express (IBX) та драйвери Borland Database Engine, інтегровані в середовища розробки Borland/Embarcadero (Delphi та C Builder), однаково успішно працюють з усіма трьома «клонами» РСКБД InterBase.

ЗАГАЛЬНІ РЕКОМЕНДАЦІЇ ДО ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ

1. При виконанні лабораторних робіт повинні бути використані знання, набуті при вивченні дисциплін “Математика”, “Бази даних”, “Інформатика” та інших.
2. Перед виконанням кожної лабораторної роботи необхідно заздалегідь, уважно і до кінця ознайомитись з методичними вказівками на її виконання. При наявності запитань – задати їх викладачу перед початком виконання роботи.
3. Всі роботи взаємозв’язані, і кожна наступна робота може бути виконана лише в тому випадку, якщо успішно закінчено всі попередні роботи. По закінченню циклу студент повинен продемонструвати робочу і готову до використання на будь-якому лабораторному комп’ютері систему; розуміти та вміти пояснити сутність технології „клієнт/сервер”; продемонструвати практичні навички адміністрування реляційної бази даних.

СКОРОЧЕННЯ І УМОВНІ ПОЗНАЧЕННЯ

AI	– автоінкрементальне поле/змінна, поле/змінна з автоматичною нумерацією значень.
БД	– база даних.
ПК	– первинний ключ.
РБД	– реляційна база даних.
РСКБД	– реляційна система керування базами даних.
СКБД	– система керування базами даних.
DB	– Database, база даних.
IB	– InterBase.
FB	– FireBird.
PK	– Primary Key, первинний ключ.
SQL	– Structured Query Language, структурована мова запитів.

ЛАБОРАТОРНА РОБОТА № 1 : Інсталяція і початкове налаштування РСКБД. Знайомство із засобами адміністрування.

Тема: Реляційні бази даних.

Мета: Інсталяція сервера РСКБД FireBird, його початкове налаштування, та встановлення засобу адміністрування і програмування.

Теоретичне введення.

Для отримання практичних навичок роботи з реляційною СКБД використовується безкоштовна система керування базами даних FireBird, версія 4.0 та безкоштовна утиліта IBExpert.

Всі описані нижче дії по інсталяції і конфігуруванню програм виконуються адміністратором операційної системи Windows для всіх користувачів комп'ютера/сервера !

Деінсталяція конфліктуючих версій сервера InterBase.

В разі, якщо на комп'ютері вже встановлено або сервер, або клієнт InterBase (це може бути будь-яка версія програм з назвою InterBase / FireBird / Yaffil), необхідно всі ці програми коректно деінсталювати, інакше виникне конфлікт версій. Бажано деінсталювати також драйвери ODBC, JAVA та .NET для даних РСКБД, якщо вони були присутні.

Зокрема, в більшість стандартних поставок середовища програмування Delphi входить або клієнт, або однокористувацький демонстраційний сервер InterBase фірми Borland. Саме на цей випадок нижче наведено приклад деінсталяції InterBase :

1. Через панель керування комп'ютером „Панель управління" → "Адміністрування" → "Служби" (Services) зупинити службу «InterBase Server» (кнопка чи дія " Stop "). При цьому допоміжна служба «InterBase Guardian» зупиниться сама.
2. Через „Панель управління" → "Встановлення/Видалення програм" (або "Програми та компоненти") повністю деінсталювати InterBase .
3. Видалити в меню програмну папку InterBase, яка, можливо, залишилася :
   - або по шляху "…" → "All Users" → "Главное Меню" → "Программы" → "InterBase "
   - або по шляху "…" → "All Users" → "Main Menu" → "Programs" → "InterBase"

   Тут і нижче три крапки "…" позначають папку "C:\Documents and Settings\".

4. Видалити на диску папку “ C: \ Program Files \ Borland \ InterBase \ ” .
5. **Важливо !!!** В папках “C:\WINDOWS\” або “C:\WINNT\SYSTEM32\” не повинно залишитися файла з назвою “ **GDS32.DLL** ” ! Якщо залишився – видалити !
6. Після цього бажано перезавантажити комп’ютер.

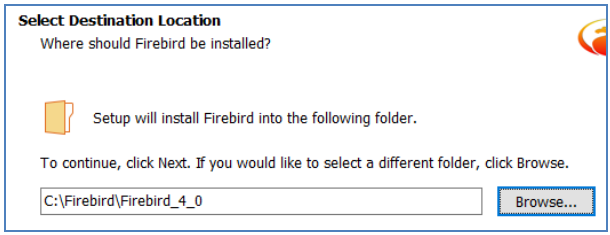
Інсталяція сервера FireBird версії 4.0

1. ***Встановлення сервера FireBird Server версії 4.0*** на платформу Windows:

   1.1. Завантажити необхідні для встановлення FireBird файли потрібно з сайту розробників www.firebirdsql.org, закладка «Завантаження» (DOWNLOADS) розділ «FireBird 4.0». Залежно від розрядності Вашої операційної системи використовуємо 32 або 64-розрядну версію Firebird (клікаємо на посиланні Win32 або Win64). Далі потрібно виконати завантаження інсталяційної програми у описі якого сказано «Windows executable installer, recommended for first-time users»).

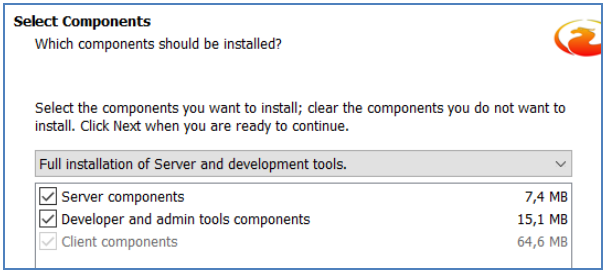
   1.2. Провідником Windows створити вручну каталог “**`C:\Firebird\`**”. Стартувати інсталіційну програму, вибрати мову інсталяції (за замовчуванням English); погодитися з ліцензійною угодою.

   1.3. При встановленні вказати (кнопкою Browse) каталог «C:\Firebird\», при цьому програма встановиться в каталог «C:\Firebird\Firebird_4_0»:

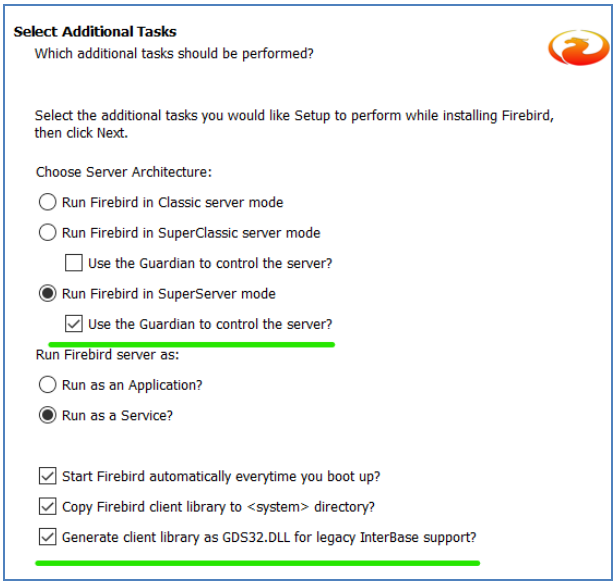


Це потрібно обов’язково зробити, оскільки адміністрування сервера РСКБД FireBird можуть здійснювати особи, які не мають адміністративних прав на операційну систему Windows! Наприклад, це обов’язково слід зробити на лабораторних комп’ютерах і термінальних серверах, оскільки студенти не мають паролів адміністратора Windows на даних комп’ютерах. На власних комп’ютерах студентів *теоретично* цього можна не робити, оскільки їх користувачі (студенти) одночасно являються адміністраторами операційних систем своїх домашніх ПК. Проте слід врахувати те, що у всіх методичних посібниках передбачається встановлення FireBird Server саме в каталог “C:\Firebird\” – тому **для запобігання плутанини рекомендується скрізь дотримуватися цього правила**!

1.4. У вікні вибору компонентів повинен бути обраним тип інсталяції “Повне встановлення сервера та інструментів розробника”:



1.5. У вікні “Оберіть додаткові задачі” необхідно відмітити два додаткові прапорці – для використання додаткового сервісу GUARDIAN, і для сумісності з усіма середовищами розробки:



1.6. У вікні “Створіть пароль адміністратора бази даних” необхідно придумати і двічі ввести пароль користувача SYSDBA (абревіатура від «System Database Administrator»), який здійснює адміністрування сервера Firebird. Цей пароль необхідно запам’ятати, а ще краще - записати. Якщо його втратити – прийдеться переінстальовувати і заново налаштовувати сервер.

1.7. Все інше – за замовчуванням.

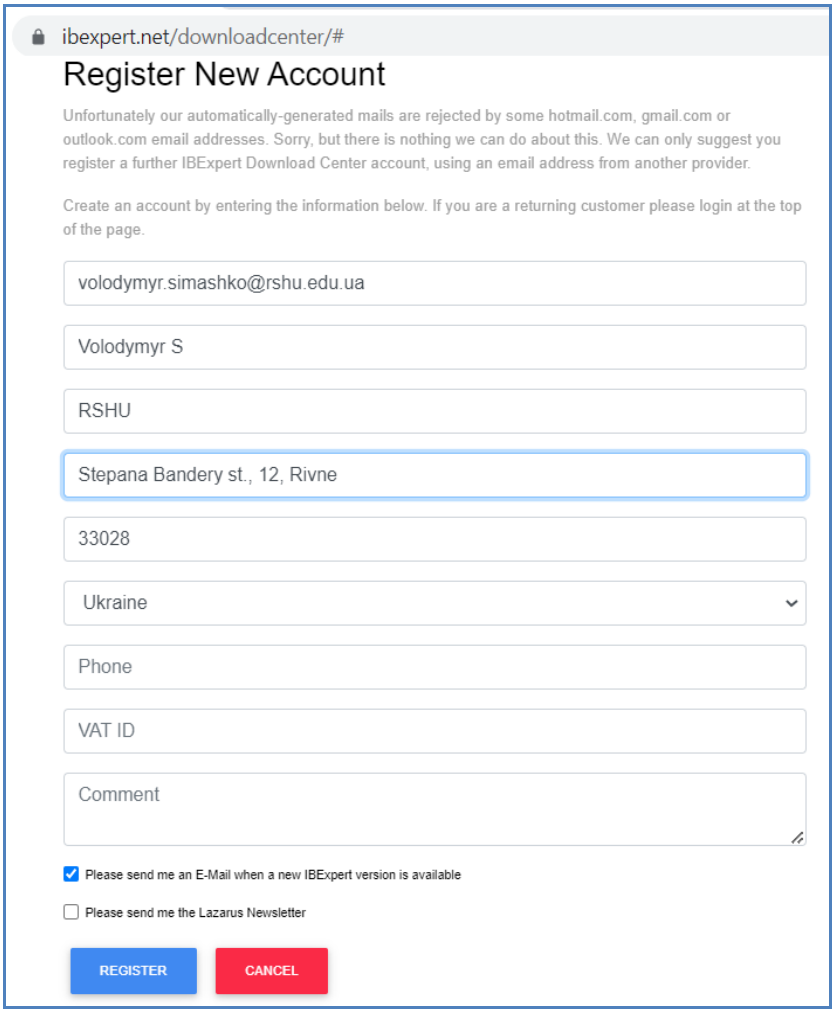
1.8. По завершенню інсталяції необхідно вручну скопіювати файл “**firebird.msg**” з каталогу, в який щойно встановився сервер Firebird

(“**C:\Firebird\Firebird_4_0**”), в каталог “**C:\Windows\System32**” (вважаємо, що ОС Windows встановленио в каталог “**C:\Windows**”). Якщо файл призначення існує – перезаписати його.

Інсталяція і налаштування програми IBExpert

2. ***Встановлення програми IBExpert*** для адміністрування сервера та програмування баз даних Interbase/Firebird/Yaffil:

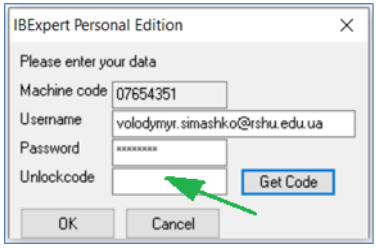
2.1. Зареєструватися на сайті розробника IBExpert www.ibexpert.net:



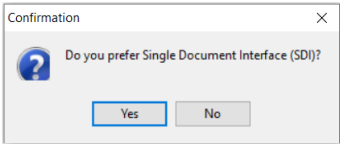
2.2. На вказану при реєстрації електронну адресу прийде логін і пароль, увійти із ними на сайт (https://www.ibexpert.net/downloadcenter/) та завантажити безкоштовну версію програми IBExpert (Personal Edition).

2.3. Встановити IBExpert. При встановленні вказати (кнопкою Browse) каталог «C:\Firebird\», при цьому програма встановиться в каталог «C:\Firebird\HK-Software\». Усе решта – за замовчуванням.

2.4. При першому запуску програми необхідно її активувати. Для цього заповнити Username та Password (ті ж самі, що й при вході на сайт при завантаженні), натиснути кнопку «Get Code», на електронну адресу прийде лист «Your IBExpert Personal Edition unlock code» із кодом розблокування (unlock code) – цей код потрібно внести в поле введення «Unlockcode» (зелена стрілка) і натиснути ОК.



Якщо Ви все зробили вірно – у діалозі підтвердження вигляду інтерфейсу програми за замовчуванням потрібно відповісти Yes:



На випадок необхідності повторної активації програми логін, пароль та код розблокування необхідно запам’ятати, а ще краще - записати.

2.5. Для зручності створіть на робочому столі Windows ярлик на програму «C:\Firebird\HK-Software\IBExpert\IBExpert.exe».

3. ***Створення службової користувацької бази даних IBExpert (далі - User Database)***. Не обов’язково, але бажано створити локальну службову користувацьку базу даних для програми IBExpert. В цій БД зберігатиметься інформація про всі сервери та бази даних, з якими буде з’єднуватися та працювати користувач програми IBExpert, історія виконання ним SQL-запитів, тощо. Ця користувацька база даних створюється для більш стабільної та зручної роботи IBExpert-а, її можна розмістити на будь-якому сервері, де інстальовано і працює FireBird або InterBase. Ми розмістимо її на локальному комп’ютері в каталозі «C:\FireBird\IBExpert_UserDB\»:

3.1. Створити каталог «C:\FireBird\IBExpert_UserDB\».

3.2. Стартувати програму “IBExpert”, відкрити пункт меню «Опції» (Options) → «Налаштування середовища» (Environment Options).

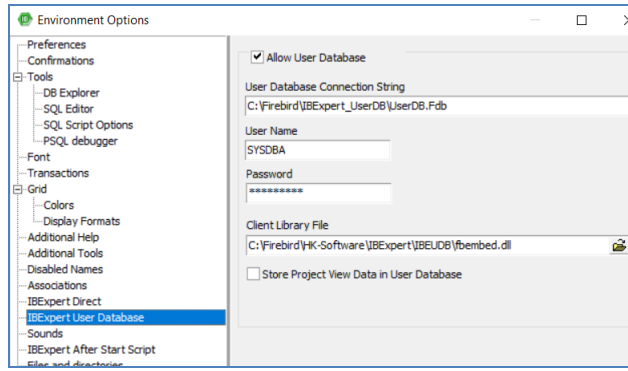
3.3. В гілці “IBExpert User Database” заповнити наступні параметри підключення до службової бази даних (див. малюнок нижче):

3.3.1. Підняти прапорець дозволу використання бази даних користувача (Allow User Database)

3.3.2. Внести рядок з’єднання із користувацькою БД (User Database Connection String) «C:\Firebird\IBExpert_UserDB\UserDB.Fdb». (це означає, що файл службової бази даних UserDB.Fdb буде створено у каталозі C:\FireBird\IBExpert_UserDB\). Розміщення і назва файлу службової бази даних можуть бути довільні, єдина умова – каталог необхідно заздалегідь створити вручну. У даному випадку каталог **C:\FireBird\IBExpert_UserDB** точно існує, тому рекомендується вказати саме цей каталог.

3.3.3. Ім’я користувача «SYSDBA» (логін адміністратора FireBird).

3.3.4. В поле «Пароль» вписується той пароль адміністратора FireBird, який Ви придумали та вказали у процесі інсталяції. **Якщо в подальшому цей пароль змінити – то одразу після цього в даній гілці налаштувань IBExpert-а буде необхідно внести і зберегти новий пароль!**



3.4. Натисути OK, закрити і відкрити IBErpert. При повторному відкритті програма видасть помилку про неможливість з’єднатися з користувацькою БД «Cannot connect to User Database» - це нормально, тому що БД ще не створена. Необхідно зайти в ту ж гілку налаштувань “IBExpert User Database”, переконатися що там з’явилася кнопка створення та ініціалізації користувацької БД «Create and Init User Database». По натисненню цієї кнопки користувацьку БД буде створено – звичайно за умови, що інсталяція і запуск сервера FireBird пройшли успішно. Потрібно впевнитися, що у вказаному каталозі з’явився файл

службової БД. Обов'язково перезапустити IBExpert ще раз, і впевнитися у його нормальному старті.

3.5. На лабораторному сервері ВНЗ користувацька БД може бути вже створена іншим студентом. Тому, якщо робота виконується на лабораторному ПК (а не домашньому) - потрібно перед початком роботи пересвідчитися, що в налаштуваннях IBExpert дозволене використання користувацької БД (піднятий прапорець "Allow User Database") та правильно внесені стрічка з'єднання, логін та пароль. Якщо ні – внести, зберегти і перезапустити IBExpert.

4. ***Підготовка до лабораторних робіт***:

4.1. Створити каталог "C:\Firebird\DATA\", завантажити і розархівувати у цей каталог архів додаткових файлів для виконання лабораторних робіт «FB_Labs_Files.Zip» (зі сторінки курсу в MOODLE, лінк «Файли для лабораторних робіт»). В каталозі повинні з'явитися бази даних «EMPLOYEE» та «SampleDB_1», і файли SQL-скриптів.

Додаткові налаштування для комп'ютерів та сервера у ВНЗ

5. ***Додаткові дії, необхідні лише на термінальних серверах*** (якщо інсталяція проводиться не на домашньому ПК, а у ВНЗ): на лабораторних комп'ютерах і на термінальних серверах, на яких працюватимуть пересічні користувачі Windows (не адміністратори), то слід на каталог "C:\Firebird\", а також на всі вкладені в нього папки і файли, надати повні права усім пересічним користувачам. На власних комп'ютерах студентів цього можна не робити, оскільки їх користувачі (студенти) одночасно являються адміністраторами операційних систем своїх ПК, і навряд чи будуть працювати в режимі з обмеженням прав.

Додаткові налаштування на лабораторних комп'ютерах

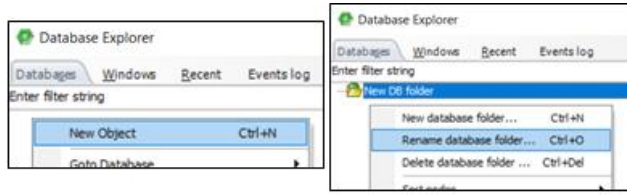
6. ***Додаткові дії, необхідні лише на лабораторних комп'ютерах*** (якщо інсталяція проводиться не на домашньому ПК, а у ВНЗ) виконуються особисто кожним користувачем операційної системи Windows комп'ютера/сервера – персонально для себе. У лабораторії користувацька БД може бути вже створена. Тому на лабораторному ПК потрібно перед початком кожної роботи пересвідчитися, що в налаштуваннях IBExpert дозволене використання користувацької БД (відмічена опція "Allow User Database") та правильно внесені стрічка з'єднання, логін та пароль.

Перевірка роботи інстальованих програм

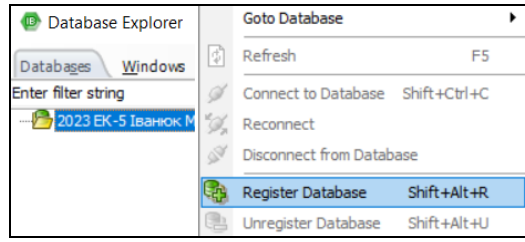
7. ***Для перевірки, чи вдало виконано інсталяцію***:

7.1. Стартувати IBExpert.

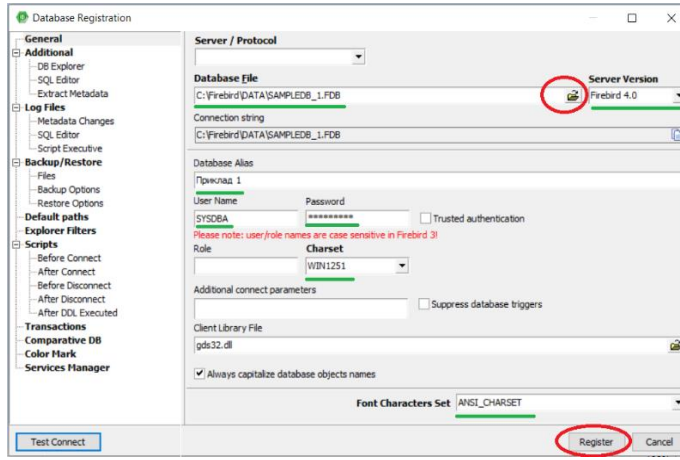
7.2. У вікні провідника баз даних (Database Explorer) через контекстне меню (див. малюнок нижче) створити папку студента, і перейменувати її у формат «*"рік" "група" "піб студента повністю"*» . Наприклад, студент Іванюк Михайло Борисович з академгрупи ЕК-5 у 2023-му навчальному році повинен створити свою папку, та назвати її «2023 ЕК-5 Іванюк Михайло Борисович»:



7.3. У своїй папці (через контекстне меню, як показано на малюнку нижче) зареєструвати тестову базу даних, що знаходиться у файлі «C:\Firebird\DATA\SampleDB_1.Fdb»:



Для цього заповнити поля, як показано на малюнку, файл БД вказувати через кнопку «Огляд».



По натисненню кнопки «Реєструвати» програма IBExpert збереже (в користувацькій БД) параметри з'єднання із базою даних: стрічку

з'єднання, версію сервера БД, логін і пароль користувача, мову бази даних за замовчуванням, опис БД.

7.4. Перезапустити (закрити і відкрити) IBExpert, впевнитися що Ваша реєстраційна інформація збереглася. Підключитися (даблкліком) до бази даних «Тестова БД № 1», впевнитися що вміст бази даних (Домени, Таблиці, Процедури, …) відображається.

Засоби для адміністрування сервера і для роботи з базою даних.

Один із найзручніших засобів для адміністрування серверів і для роботи з базами даних FB/IB – безкоштовна програма "IBExpert". Перед тим, як з'єднатися з базою даних FireBird (InterBase, Yaffil), користувач повинен внести дані про цю БД:

- ✓ Server/Protocol: мережевий протокол, у випадку локальної БД не вказується;
- ✓ Server Name: мережева адреса сервера (IP-адреса чи мережеве ім'я), у випадку локальної БД не вказується;
- ✓ Port: номер порту TCP/IP, у випадку локальної БД не вказується;
- ✓ Server Version: версія сервера РСКБД;
- ✓ Database File: **повний шлях до основного файлу** бази даних на сервері (вказується у випадку, якщо псевдоніми БД на даному сервері не використовуються) **або псевдонім** бази даних;
- ✓ Database Alias: довільна назва бази даних для відображення;
- ✓ Charset: кодування у якому зберігаються стрічкові дані;
- ✓ User Name, Password, Role: логін, пароль і назва ролі користувача для з'єднання їх БД; можна не вносити якщо потрібно щоразу при з'єднання вводити їх вручну;

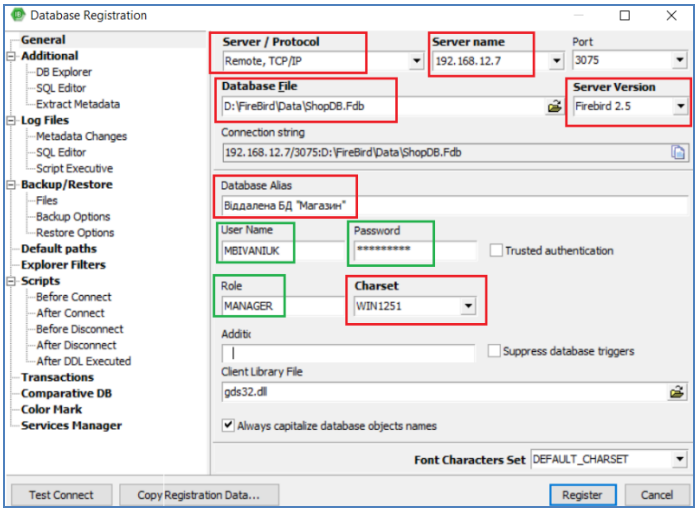
Процедура внесення і збереження цієї інформації називається «реєстрація бази даних (Database registration)». Всі зареєстровані в IBExpert бази даних відображаються у вікні "DataBase Explorer". Реєстрація здійснюється або через пункт меню «База даних → Зареєструвати базу» (Database → Register Database), або через контекстне меню Database Explorer-а.

Програма IBExpert передбачає два способи збереження внесених користувачем даних: або в файлі спеціального формату у профілі користувача операційної системи, або в спеціально створеній службовій користувацькій базі даних ("IBExpert User Database", див. вище). У користувацькій БД IBExpert автоматично запам'ятовуються також налаштування програми та історія виконання SQL-запитів. Перший спосіб – у файлі – за замовчуванням, проте не дуже надійний, і недостатньо гнучкий, тому бажано ним не користуватися. Для використання другого способу, рекомендованого автором програми IBExpert, потрібно одразу після інсталяції IBExpert-а або створити цю службову базу даних (як описано вище), або – підключитися до раніше створеної. Службова користувацька база даних IBExpert-а може знаходитися на будь-якому (мережевому чи локальному) комп'ютері, на якому інстальовано одну з версій

InterBase/FireBird/Yaffil. Якщо службова користувацька БД не відкривається або недоступна – то раніше збережена інформація про сервери, про бази даних, і про налаштування програми - також недоступна. Саме тому найбільш надійно – створювати цю службову БД на локальному комп'ютері.

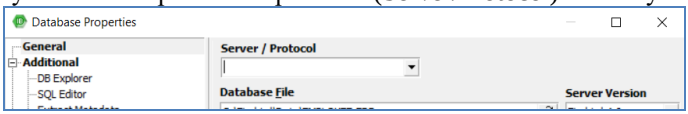
Для з'єднання з сервером РСКБД необхідно зареєструвати хоча б одну базу даних, яка на даному сервері знаходиться.

Нижче наведено приклад заповнення реєстраційної інформації для наступних параметрів сервера і бази даних: Remote (віддалений, тобто мережевий а не локальний) сервер, протокол TCP/IP, адреса сервера 192.168.12.7, порт №3075, сервер РСКБД FireBird версії 2.5, база даних «Магазин», основний файл БД «D:\FireBird\Data\ShopDB.Fdb», кодування текстових даних Win1251 (кирилиця Windows). Поля, які обов'язково повинні бути заповнені при реєстрації, обведено червоними рамками, необов'язкові – зеленими:

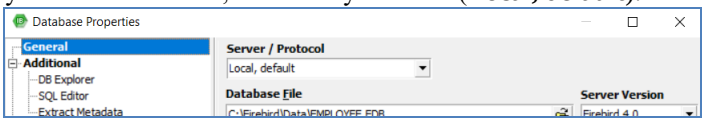


Якщо база даних знаходиться на локальному комп'ютері – існує три способи зареєструвати цю базу даних:

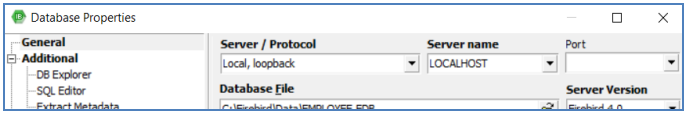
1) за замовчуванням - мережевий протокол (Server/Protocol) не вказується:



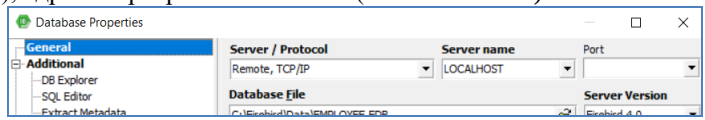
або вказується «Локальний, за замовчуванням» (**Local, default**):



2) мережевий протокол (Server/Protocol) вказується «Локальний, циклічний» (**Local, loopback**), адреса сервера «Локальний» (**LOCALHOST**):



3) мережевий протокол (Server/Protocol) вказується «Віддалений» (**Remote, TCP/IP**), адреса сервера «Локальний» (**LOCALHOST**):

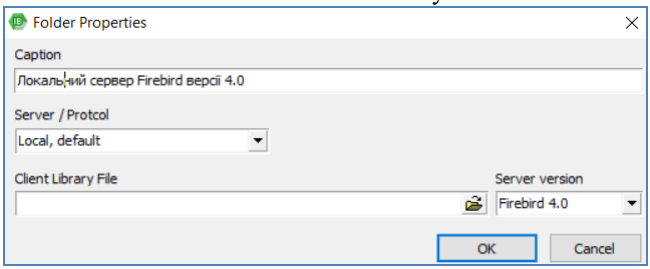


Різниця між цими способами полягає у з'єднанні через різні клієнтські бібліотеки, з точки зору функціональності усі способи рівнозначні. При виникненні проблем при першому способі можна застосувати другий чи третій.

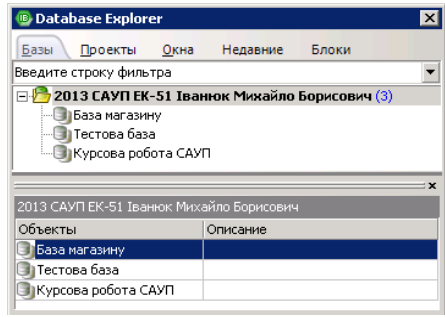
Ім'я користувача і пароль дуже бажано НЕ ВНОСИТИ – в цьому випадку буде необхідно їх вводити щоразу при підключенні до БД.

Редагування або перегляд реєстраційної інформації здійснюється через пункт меню «База даних → Реєстраційна інформація бази» (Database Registration Info…). Створення копії реєстраційної інформації: «База даних → Зклонувати реєстраційну інформація бази» (Clone Registration Info…). Видалення інформації про БД: «База даних → Видалити реєстраційну інформацію» (Unregister Database), при цьому із файлом бази даних ніяких дій не відбувається, просто із програми IBExpert вилучається інформація про з'єднання із цією БД.

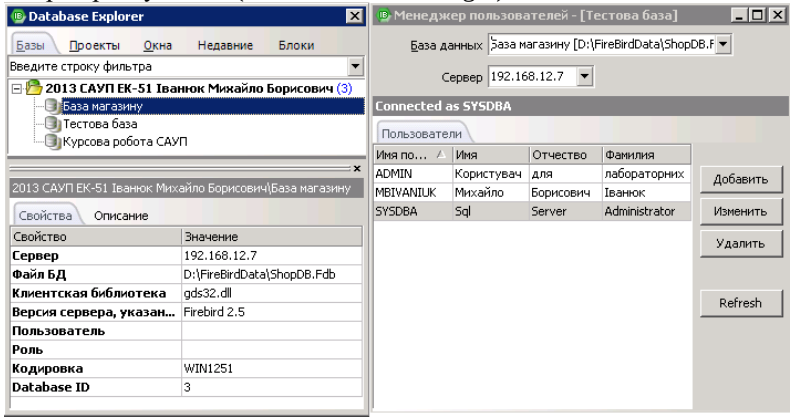
Для випадку, якщо баз даних багато – у вікні "DataBase Explorer" їх можна впорядковувати по папках. Створення нової папки здійснюється через пункт меню «База даних → New DB Object» («Нова папка БД»), редагування або перегляд параметрів папки - через пункт меню «База даних → Перейменувати папку БД», видалення - через пункт меню «База даних → Видалити папку БД». Приклад властивостей папки показано на малюнку:



За звичай в IBExpert-і створюються окремі папки для різних серверів, які адмініструються. На комп'ютерах у ВНЗ окремі папки будуть створюватися для кожного студента. Приклад впорядкування інформації про бази даних у папці студента наведено на малюнку :



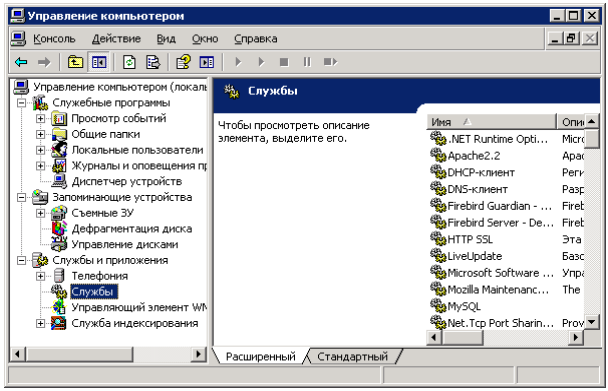
Для роботи з користувачами сервера РСКБД необхідно у вікні DataBase Explorer виділити курсором будь-яку базу даних, яка на цьому сервері знаходиться (див. малюнок нижче – курсором виділено базу даних «Магазин»), та відкрити менеджер користувачів сервера через пункт меню «Інструменти → Менеджер користувачів» (Tools → User Manager):



Після цього IBExpert спробує з'єднатися із сервером РСКБД, вказаним у параметрах виділеної бази даних, для з'єднання необхідно ввести ім'я і пароль адміністратора сервера РСКБД. Вище показано базу даних користувачів після вдалого з'єднання. Для внесення, редагування і видалення користувачів передбачено відповідні дії.

Ручний старт і зупинка сервера FireBird.

Ручне керування роботою сервера InterBase/FireBird/Yaffil виконується через інструментарій Windows «Управління комп'ютером», інструмент «Служби» :



Для зупинки сервера FireBird необхідно зупинити його основну службу з назвою «**Firebird Server** - DefaultInstance».

Для старту сервера FireBird необхідно стартувати його допоміжну службу з назвою «**Firebird Guardian** - DefaultInstance».

Призначення допоміжної служби “FireBird Guardian” – підвищення надійності роботи основної служби “FireBird Server” :

1. Після запуску допоміжна служба “FireBird Guardian” запускає основну службу (процес) “FireBird Server”, та відслідковує її роботу.
2. В разі, якщо “FireBird Server” аварійно зупинився внаслідок критичної помилки, допоміжна служба “FireBird Guardian” здійснює повторний перезапуск основної служби “FireBird Server”.
3. В разі нормальної зупинки сервера РСКБД допоміжна служба “FireBird Guardian” не втручається в його роботу, та зупиняється сама.

Зміна стандартних паролів.

Після інсталяції будь-якої версії InterBase/FireBird/Yaffil в базі даних паролів присутній лише один користувач : ім’я «**SYSDBA**», пароль створюється при встановленні. Це єдиний користувач, якому дозволено вносити зміни в базу даних паролів, тобто єдиний **адміністратор сервера РСКБД Firebird**.

Якщо на особистому комп’ютері студента пароль користувача SYSDBA буде змінено – не забудьте одразу ж змінити пароль в закладці налаштувань IBExpert-а «IBExpert User Database», і перезапустити IBExpert для того, щоб зміни вступили в дію!

Також в учбових цілях в лабораторіях факультету ДКМ одразу після інсталяції вводиться звичайний „пересічний” користувач сервера РСКБД для виконання лабораторних робіт : ім’я «**admin**» , пароль «**admin**». Для простого користувача (не адміністратора сервера !) спеціально обрано ім’я admin для наочної демонстрації того факту, що адміністративні права та спеціально таким чином обране ім’я користувача – абсолютно не зв’язані між собою речі !

Деякі важливі правила кодування імен і паролів користувачів, та назв

об'єктів бази даних.

- При введенні імен користувачів InterBase/FireBird (і інших РСКБД також) регістр клавіатури (великі чи малі літери) ролі не грає. Наприклад: користувачі з іменами «sysdba», «SYSDBA» і «SYSdba» - це один і той же користувач. Саме тому в більшості програм ім"я користувача автоматично переводиться в верхній регістр (перекодовується в великі літери). Це зроблено для запобігання плутанини, яка часто виникає при створенні користувачів у різних регістрах.

  Примітка: у РСКБД Firebird від версії 3.0 та вище імена (логіни) користувачів і при створенні, і при підключенні для БД необхідно вводити лише у верхньому регістрі.
- При введенні паролів InterBase/FireBird (і інших РСКБД також) регістр клавіатури (великі чи малі літери) враховується, і тому треба зважати на індикатор "CapsLock", оскільки пароль за звичай маскується „зірочками". Наприклад: паролі користувачів «abcd», «ABCD» і «ABcd» в усіх РСКБД будуть вважатися різними.
- В процесі редагування облікових записів користувачів пароль користувача можна міняти скільки завгодно разів, а от ім'я залишається незмінним „на все життя".
- Імена, паролі, а також назви об'єктів реляційної бази даних (таблиць, полів, та інших) придумуються довільно, але в їх назвах можна використовувати лише латинські літери, цифри та символ „підкреслення", причому першим символом імені чи назви повинна бути лише літера.
- Назви об'єктів РБД, та імена користувачів повинні бути інформативними.
- Неприпустимо називати користувача абстрактним іменем, наприклад, "User12". Загальноприйнято, що ім'я користувача повинно включати його ініціали (або ім'я) та прізвище в латинській транскрипції, наприклад : MBIVANIUK, OLEGYAKIMCHUK.
- Неприпустимо називати об'єкти бази даних абстрактними іменами, наприклад, таблиця "Table7", чи процедуру "Proc312". Загальноприйнято, що імена об'єктів повинні відповідати даним чи операціям, які містять ці об'єкти, наприклад, таблиця "StudentAdress", процедура "CalculateSaldo".

Завдання студента заключається в :

- Інсталяції сервера FireBird на особистому комп'ютері;
- Встановленні і налаштуванні програми IBExpert;
- Перевірці працездатності сервера FireBird після його встановлення;
- Умінні вручну зупиняти і стартувати сервер FireBird;
- Початковому освоєнні програми IBExpert (провідник баз даних і менеджер користувачів);

Завдання на лабораторну роботу.

Пункти, які нижче виділено жирним шрифтом, можуть бути

виконані лише на особистому комп'ютері студента. На лабораторних комп'ютерах у студента недостатньо прав, тому робота вже виконана !

1. Ознайомитися з теоретичною частиною до лабораторної роботи. При наявності запитань – задати їх викладачу перед початком виконання роботи.
2. **Перевірити наявність конфліктуючих (старих) версій InterBase / FireBird. При їх наявності – деінсталювати.**
3. **Інсталювати сервер FireBird та клієнтську програму IBExpert – відповідно до наведених вище інструкцій та пояснень. Створити (на особистому комп'ютері студента) та налаштувати службову користувацьку базу IBExpert-а.**
4. **Вручну зупинити і стартувати сервер FireBird.**
5. Створити свій особистий каталог для лабораторних робіт. На комп'ютері ВНЗ це припустимо лише на диску D (наприклад: D:\Ivaniuk\Firebird\Labs), на особистому – бажано всередині каталога C:\Firebird (наприклад: C:\Firebird\Labs). Створити у ньому підкаталоги SQL, Files та Data. Завантажити необхідні для лабораторних робіт файли (див. нижче ДОДАТОК 1 в кінці посібника), розархівувати у папку Data.
6. У вікні "Database Explorer" програми IBExpert створити папку з іменем: «*"рік" "група" "піб студента повністю"*». Наприклад, студент Іванюк Михайло Борисович з академгрупи ЕК-5 у 2023-му навчальному році повинен створити папку: «2023 ЕК-5 Іванюк Михайло Борисович».
7. У щойно створеній власній папці "Database Explorer-а" зареєструвати дві тестові бази, що знаходяться на диску "D:" своєму особистому каталозі студента : SampleDB_1.Fdb (із назвою «**Приклад 1**») та EMPLOYEE.Fdb (із назвою «**Службовці**»).
8. З'єднатися з обома базами даних з користувачем sysdba, переглянути таблиці, дані у таблицях, ознайомитися із іншими об'єктами.
9. Відкрити менеджер користувачів IBExpert-а, створити користувача з іменем і паролем admin (якщо такий не існує), та свого власного користувача із англійською транскрипцією своїх ініціалів і прізвища (наприклад, студент Іванюк Михайло Борисович повинен створити користувача MBIVANIUK).
10. З'єднатися з обома базами даних з користувачами admin, та зі своїм власним. Спробувати переглянути дані в таблицях. Пояснити результат.

Контрольні запитання.

1. Як можна перевірити, чи працює сервер РСКБД FireBird? Існує три різних способи, назвіть щонайменше два.
2. Для чого потрібно реєструвати бази даних в програмі IBExpert?
3. Що запам'ятовує програма IBExpert при цій реєстрації, і де саме можуть зберігатися реєстраційні дані? Назвіть два способи.
4. Де в налаштуваннях IBExpert можна перевірити, який із двох способів збереження реєстраційних даних використовується?

5. Якщо під час запуску IBExpert видається повідомлення про помилку “Неможливо підключитися до користувацької БД… Your user name and password are not defined” – що це означає, і як вийти із ситуації?
6. З яким сервером РСКБД намагається встановити з’єднання IBExpert при відкритті «Менеджера користувачів»?
7. В якому випадку спроби редагування облікових записів користувачів в «Менеджері користувачів» будуть вдалими?

ЛАБОРАТОРНА РОБОТА № 2 : Адміністрування сервера РСКБД.

Тема: Реляційні бази даних.
Мета: Отримання навичок адміністрування реляційних баз даних.

Теоретичне введення.

Опис понять „адміністратор сервера”, „адміністратор (власник) бази даних”, „звичайний користувач”.

Адміністратором сервера РСКБД являється той, і лише той користувач, який інсталював дану систему, і знає ім’я і пароль користувача, який при цьому створено або використано. В деяких РСКБД ці ім”я та пароль створюються (вводяться безпосередньо при встановленні сервера РСКБД на комп”ютер). В InterBase/FirBird пароль при інсталяції не вводиться, він описаний в документації (ім’я користувача «sysdba», пароль «masterkey»). Тому кажуть, що адміністратором реляційної СКБД InterBase/FireBird є єдиний користувач - **sysdba**.

Прерогативою адміністратора сервера РСКБД є робота з базою даних паролів (введення і знищення облікових записів користувачів, зміна їх паролів).

Адміністратором бази даних на сервері РСКБД являється той, і лише той користувач, який створив дану базу даних, тобто ім’я і пароль якого введені безпосередньо при створенні БД. Таким користувачем може бути будь-який користувач: як адміністратор сервера, так і звичайний (пересічний). Якщо, наприклад, при створенні бази даних використано обліковий запис звичайного користувача «MBIVANIUK», то кажуть, що адміністратором (власником, OWNER-ом) бази даних є користувач MBIVANIUK.

Прерогативою адміністратора (власника, OWNER) БД є створення / зміна / знищення будь-яких об’єктів (таблиць, ключів,…), і надання на них прав.

Прерогативою пересічних (звичайних) користувачів є робота з даними (читання / запис / зміна / видалення даних, і виконання процедур) – але лише в межах тих прав, які надано цьому користувачу адміністратором БД.

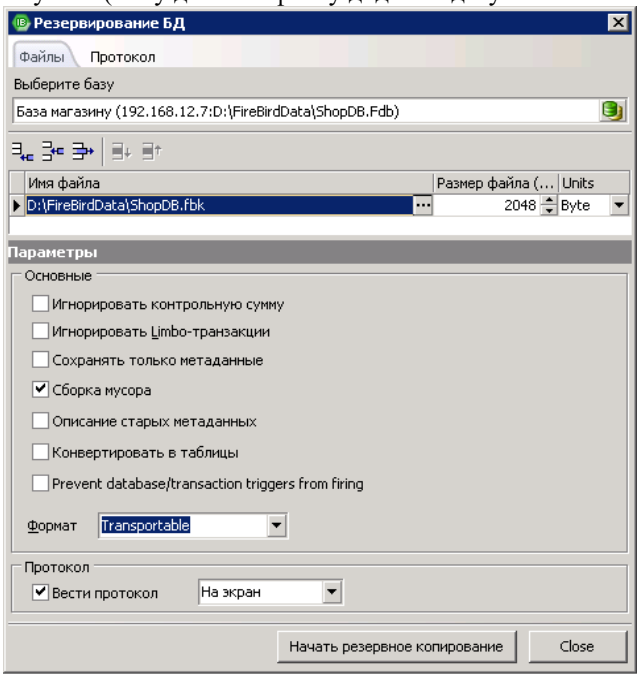
Опис засобів адміністрування.

Один із найзручніших засобів для адміністрування серверів і для роботи з

базами даних FireBird/InterBase/Yaffil – безкоштовна програма “IBExpert” [8, 11]. Можливості цієї програми дуже великі. В даній роботі вивчаються лише основні, які знадобляться для підтримки нормальної роботи сервера та невеликої бази даних (наприклад, при виконанні курсової роботи).

Найголовніші адміністративні процедури виконуються в IBExpert через пункт меню “Служби” (Service):

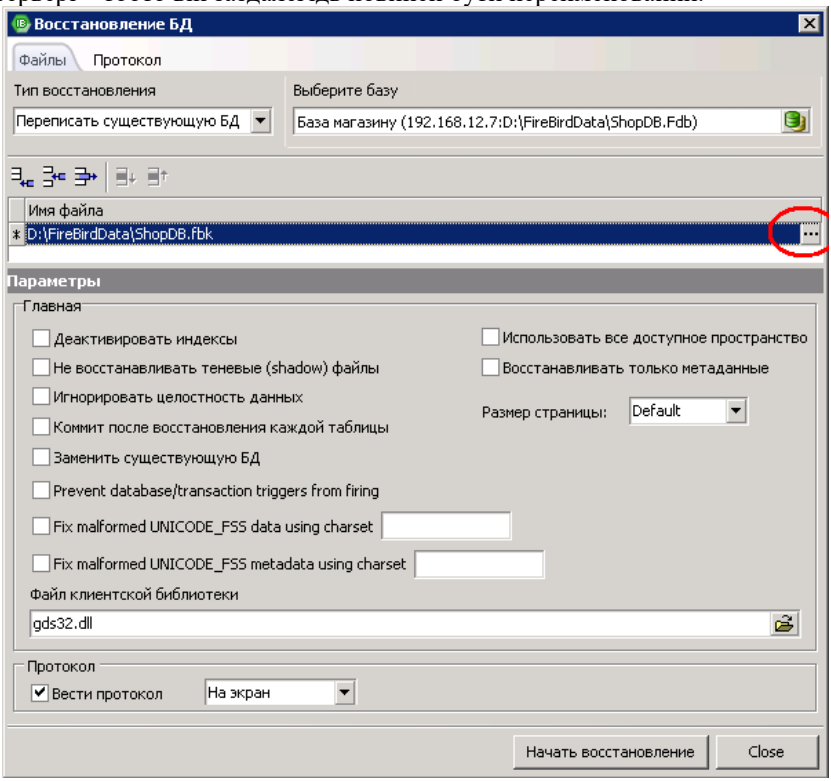
1. “Резервування бази даних” (Backup Database) – повний експорт бази даних в файл резервної копії. На малюнку нижче зображено приклад налаштування рекомендованих параметрів резервування, БД повинна бути закрита від користувачів (виконано Shutdown). Курсором виділено назву файла, в якому буде збережено експортовану базу даних. Важливо розуміти, що файл резервної копії створюється не на локальному комп’ютері, а на тому сервері, де інстальовано РСКБД (в прикладі нижче – на мережевому сервері 192.168.12.7 в каталозі D:\FireBirdData\, назва файлу ShopDB.fbk). За замовчуванням використовується каталог і назва файла бази даних, лише змінюється розширення. Каталог і назву файла можна змінити; вказаний каталог повинен існувати на сервері, а файл резервної копії навпаки – не повинен існувати (тому до імені файлу додають дату і навіть час Backup-у).



Операцію можуть виконати адміністратор сервера РСКБД (sysdba) або власник БД. Слід знати, що всі облікові записи користувачів InterBase/FireBird/Yaffil зберігаються в базі даних паролів в каталозі, в який інстальовано сервер РСКБД. Для версії FireBird 3.0 це файл Security3.Fdb, і якщо сервер інстальовно згідно наведеної вище інструкції, то файл паролів –

це “C:\Firebird\Firebird_3_0\Security3.Fdb”. При резервному копіюванні БД слід копіювати і цей файл – для збереження бази користувачів.

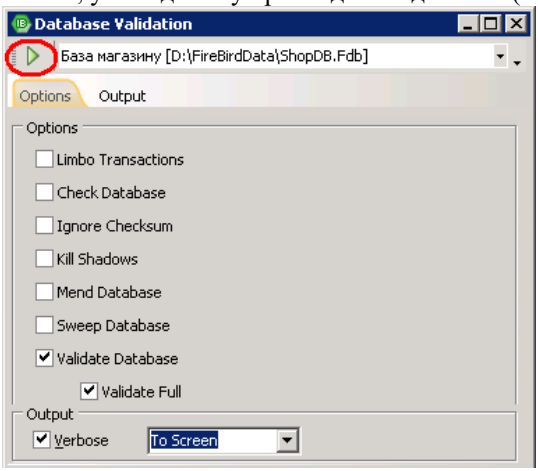
2. “Відновлення бази даних” (Restore Database) – відновлення бази даних із раніше експортованого файлу резервної копії. На малюнку нижче зображено вікно налаштування рекомендованих параметрів відновлення. Курсором виділено назву файла, в якому раніше було збережено експортовану базу даних (див. попередній пункт). Важливо розуміти, що і резервування, і відновлення відбувається на тому сервері, де інстальовано РСКБД (в прикладі нижче – на мережевому сервері 192.168.12.7). Тому кнопка «вибрати файл», яка відкриває діалог (на малюнку нижче обведена червоним), буде працювати лише в випадку, якщо Ви працюєте із локально інстальованим сервером РСКБД. Також при тих налаштуваннях, які показані на малюнку нижче, передбачається, що файл бази даних, яку відновлюють з копії (в даному випадку файл D:\FireBirdData\ShopDB.Fdb) відсутній на сервері – тобто він заздалегідь повинен бути перейменований.



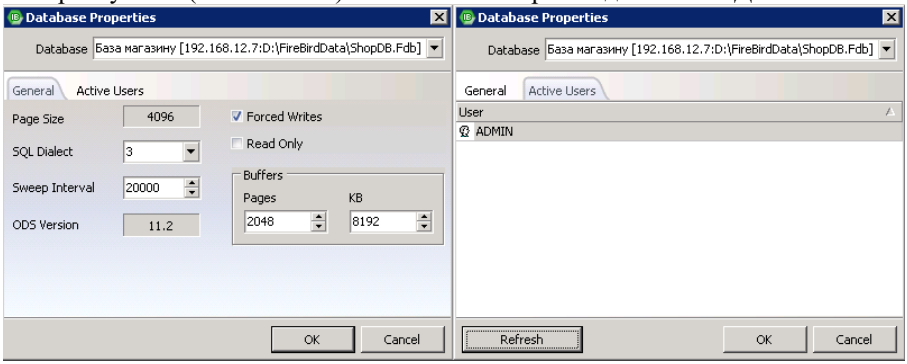
Операцію можуть виконати адміністратор сервера РСКБД (sysdba) або власник БД. Також це може зробити будь-який інший користувач, проте при цьому він автоматично стане власником відновленої бази, але не її об’єктів!

3. “Перевірка бази даних” (Database Validation) – перевірка бази даних на наявність проблем з файлом БД. На малюнку нижче зображено вікно

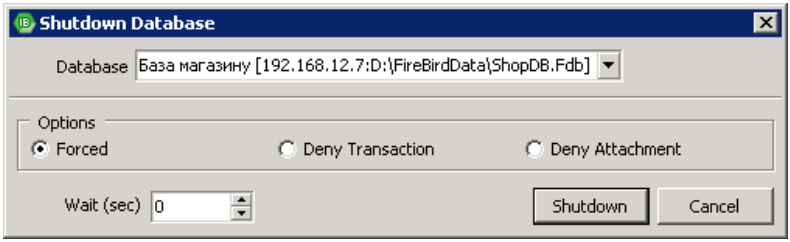
налаштування рекомендованих параметрів для повної перевірки БД. Старт перевірки відбувається по натисненню обведеної червоним кольором кнопки, база даних повинна бути закрита від користувачів (виконано Shutdown). Операцію можуть виконати адміністратор сервера РСКБД (sysdba) або власник БД. В разі наявності проблем/помилок всі вони будуть виведені в протокол, у наведеному прикладі – на дисплей (To Screen).



4. “Параметри бази даних” (Database Properties) – перегляд і зміна найважливіших параметрів РБД. В закладці «Загальне» (General) можна переглянути/змінити важливі параметри БД, в закладці «Активні користувачі» (Active Users) побачити хто зараз з’єднаний із БД.



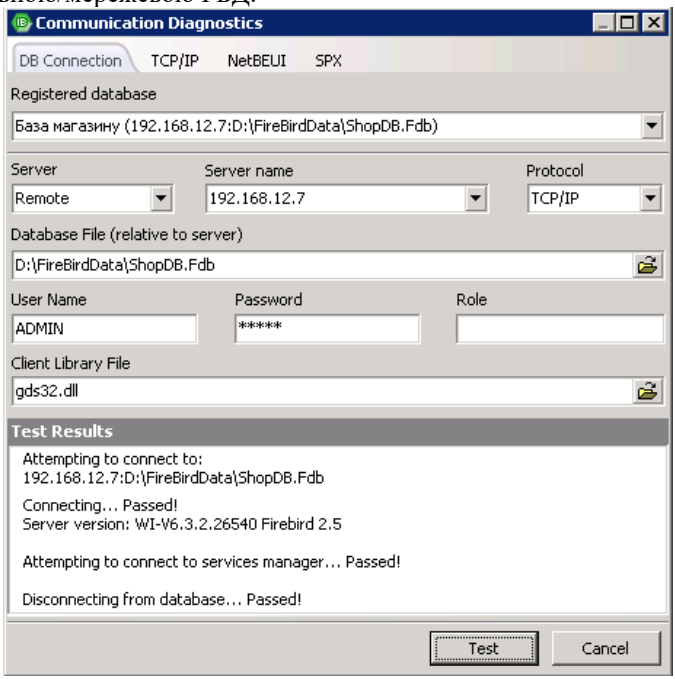
5. “Базу в даун !” (Database Shutdown) – «опускання» РБД (закриття від звичайних користувачів). В вікні налаштувань вибирається тип Shutdown-у та таймаут очікування. Попередньо до БД потрібно підключитися.



По завершенню таймауту при неможливості виконання Shutdown-у генерується помилка (для запобігання безмежного очікування). Якщо БД вже знаходиться у стані “опущено”, а власник БД повторно намагається виконати Shutdown, то виникає помилка “Target shutdown mode is invalid for database”, і нічого не відбувається.

6. “Базу на бочку !” (Database Online) – «підняття» РБД (відкриття для звичайних користувачів).

7. “Діагностика з’єднання (Communication Diagnostics) – діагностика з'єднання з локальною/мережевою РБД.



При виконанні завдання на дану лабораторну роботу слід мати на увазі, що

для коректного виконання всіх адміністративних процедур (пункт меню “Служби” (Service)) їх **слід виконувати лише з обліковим записом власника (OWNER) бази даних**.

Завдання на лабораторну роботу.

1. Відкрити IBExpert, менеджер користувачів, переглянути, які користувачі FireBird зареєстровані на локальному сервері РСКБД.
2. Ввести (створити) обліковий запис нового користувача з одним із таких імен : administrator, superuser, superadmin, administrator1, superuser1, superadmin1, … (обрати будь-яке відсутнє в списку ім’я, крім admin).
3. Закрити менеджер користувачів, з’єднатися з базою даних «Службовці» з іменем щойно введеного нового користувача, відкрити менеджер користувачів, впевнитися що новий користувач з’явився у списку.
4. Спробувати ввести ще одного нового користувача з довільним іменем, пояснити результат.
5. Від’єднатися від бази даних «Службовці» , під’єднатись до неї з іменем „sysdba”, відкрити менеджер користувачів. Знайти обліковий запис щойно введеного нового користувача, та змінити йому пароль.
6. Закрити менеджер користувачів, від’єднатися від бази даних «Службовці», під’єднатись до неї з іменем нового користувача та зміненим паролем.
7. Від’єднатися від бази даних «Службовці» , під’єднатись до неї з іменем „sysdba”, відкрити менеджер користувачів. Знайти обліковий запис щойно введеного нового користувача, та знищити його.
8. Закрити менеджер користувачів, з’єднатися з базою даних «Службовці» з користувачем “admin”, відкрити менеджер користувачів, впевнитися що користувач пропав.
9. Спробувати ввести ще одного нового користувача з довільним іменем, пояснити результат. Закрити менеджер користувачів. Закрити програму IBExpert.
10. Відкрити IBExpert. „Опустити” базу даних «Службовці» (ShutDown).
11. З’єднатися з базою даних «Службовці» з користувачем, який відповідає ініціалам та імені студента (цей користувач був введений у попередній лабораторній роботі; якщо пароль забули – потрібно попередньо змінити його). Пояснити результат.
12. Зробити перевірку бази даних «Службовці» на наявність помилок. Прочитати і перекласти звіт про перевірку, пояснити.
13. „Підняти” базу даних «Службовці» (зробити її ReStart) – тобто відкрити її для простих користувачів.
14. Виконати процедуру резервування і відновлення БД «Службовці» через експорт/імпорт бази даних в бінарний файл (процедури Backup/Restore) :
    14.1. Створити окремий порожній каталог для всіх файлів, які з’являться в результаті виконання процедури Backup/Restore.
    14.2. „Опустити” базу даних (зробити її ShutDown).
    14.3. Перевірити БД на наявність помилок.

14.4. Зробити резервування (Backup) бази даних. **При цьому використати ім'я і пароль користувача – власника бази даних ; це далеко не завжди SYSDBA, як це пропонується за замовчуванням програмою IBExpert !** Файл Backup-у бажано назвати так, щоб було зрозуміло, якої БД це резервна копія, якого числа і о котрій годині вона зроблена (розширення файлу не міняти !).

14.5. Зберегти у текстовий файл протокол Backup-у (або одразу вказати вивід протоколу у файл, або виводити на дисплей, а по завершенню - зберегти). Переглянути протокол, знайти і перекласти заключне повідомлення, яке свідчить про успішне закінчення процедури резервування.

14.6. Впевнитись, що експортований файл резервної копії з'явився.

14.7. Скопіювати файл паролів сервера РСКБД у створений каталог (облікові записи користувачів зберігаються у файлі Security2.Fdb в каталозі, у який інстальовано РСКБД FireBird).

14.8. Змоделювати ситуацію, ніби файл БД зіпсувався: перейменувати файл бази даних. Зробіть це таким чином, щоб було зрозуміло, якого числа і о котрій годині це зроблено (у назву додати дату і час, розширення файлу не міняти). Перемістити перейменований файл БД в каталог для всіх файлів процедури Backup/Restore.

14.9. Зробити відновлення бази даних (Restore) з файла резервної копії. **При цьому використати ім'я і пароль користувача – власника бази даних !**

14.10. Зберегти у текстовий файл протокол Restore (або одразу вказати вивід протоколу у файл, або виводити на дисплей, а по завершенню - зберегти). Переглянути протокол, знайти і перекласти заключне повідомлення, яке свідчить про успішне закінчення відновлення.

14.11. Впевнитись, що файл бази даних з'явився.

14.12. Зробити перевірку бази даних «Службовці» на наявність помилок.

14.13. З'єднатися з базою даних «Службовці» з користувачем, який є власником БД. Переглянути таблиці, впевнитись, що дані збереглися після Restore.

14.14. З'єднатися з базою даних «Службовці» з користувачем, який відповідає ініціалам та імені студента, впевнитися, що БД працює.

14.15. Від'єднатися від БД. Заархівувати всі файли, які з'явилися у каталозі в результаті виконання процедури Backup/Restore, будь-яким архіватором. Файл архіву потрібно назвати таким чином, щоб було зрозуміло, якого числа і о котрій годині проводилася процедура Backup/Restore.

14.16. Скопіювати файл архіву на незалежні та надійні носії інформації.

14.17. Надіслати файл архіву на електронну адресу викладача, у тексті листа вказати «Студент "*ПІБ повністю*", архів файлів процедури Backup/Restore бази даних "*назва БД*"».

15. Виконати процедуру повного «холодного» копіювання БД «Службовці» :

15.1. Створити окремий порожній каталог для всіх файлів, які з'являться в

результаті виконання процедури «холодного» копіювання БД.

15.2. „Опустити” базу даних (зробити її ShutDown).
15.3. Перевірити БД на наявність помилок.
15.4. Від’єднатися від БД, закрити програму IBExpert.
15.5. Зупинити сервер FireBird – **це можливо лише на особистому комп’ютері студента**, тому що потребує адміністративних прав на ОС, і лише у тому випадку, якщо на даному сервері РСКБД FireBird користувачі на даний момент не працюють з іншими базами даних.
15.6. Скопіювати файл паролів сервера РСКБД у створений каталог (облікові записи користувачів зберігаються у файлі Security2.Fdb в каталозі, у який інстальовано РСКБД FireBird).
15.7. Скопіювати файл БД у створений каталог.
15.8. Стартувати сервер FireBird (якщо перед цим вдалося його зупинити).
15.9. Перевірити БД на наявність помилок.
15.10. „Підняти” базу даних.
15.11. З’єднатися з базою даних «Службовці» з користувачем, який відповідає ініціалам та імені студента, впевнитися, що БД працює.
15.12. Від’єднатися від БД. Заархівувати всі файли, які ви скопійували у каталог холодної копії, будь-яким архіватором. Файл архіву потрібно назвати таким чином, щоб було зрозуміло, якого числа і о котрій годині проводилася процедура «холодного» копіювання БД.
15.13. Скопіювати файл архіву на незалежні та надійні носії інформації.
15.14. Надіслати файл архіву на електронну адресу викладача, у тексті листа вказати «Студент ”ПІБ повністю”, архів файлів процедури “холодного” копіювання бази даних “назва БД”».

16. Виконати резервування і відновлення БД «Приклад 1» через експорт/імпорт бази даних в бінарний файл (процедури Backup/Restore).
17. Виконати повне «холодне» копіювання БД «Приклад 1».

Контрольні запитання.

1. Який користувач може вносити зміни в базу даних паролів сервера РСКБД FireBird ? Чи може користувач сервера РСКБД сам витерти свій обліковий запис ?
2. Який пароль адміністратора сервера РСКБД FireBird одразу після інсталяції сервера FireBird ? Чому одразу після інсталяції цей пароль змінюють ?
3. Який пароль адміністратора сервера РСКБД FireBird на лабораторних комп’ютерах ? Чому він такий ? Чому в промислових системах не призначають такий пароль ?
4. Як називається користувач, який створив реляційну БД ?
5. Хто (який користувач РСКБД) має право давати і відміняти дозволи на користування об’єктами реляційних БД іншим користувачам ?
6. Що відбудеться, коли адміністратор сервера РСКБД зітре в базі паролів свій власний обліковий запис ? Як можна вийти з такої ситуації ? Які можуть бути негативні наслідки ?

7. Що відбудеться, коли адміністратор сервера РСКБД забуде свій пароль ? Як можна вийти з такої ситуації ? Які можуть бути негативні наслідки ?
8. Що відбудеться, коли звичайний користувач сервера РСКБД забуде свій пароль ? Як можна вийти з такої ситуації ? Які можуть бути негативні наслідки ?
9. Для чого існує операція «опускання» (shutdown) реляційної БД ? Які види shutdown існують ? Які з видів shutdown не призводять до псування БД ?
10. Для чого існує операція «підняття» (restart, startup, online) реляційної БД ? Які користувачі можуть робити опускання і підняття бази даних ?
11. Що таке експорт (Backup) та імпорт (Restore) бази даних ? В яких випадках роблять експорт та імпорт бази даних ?
12. Для чого роблять резервні копії баз даних ? Які файли слід скопіювати ?
13. Чому перед відновленням бази даних з її резервної копії оригінал бази даних не знищують, а перейменовують або переміщають в інший каталог чи на інший диск ?
14. Для чого в реляційних СКБД використовуються різні облікові записи для різних користувачів, а не один ?
15. Як вийти з ситуації, коли адміністратор сервера РСКБД забув свій пароль ? Які негативні наслідки цього ? Як цьому запобігти ?
16. Як вийти з ситуації, коли адміністратор реляційної БД забув свій пароль ? Які негативні наслідки цього ? Як цьому запобігти ?
17. Як вийти з ситуації, коли звичайний користувач реляційної БД забув свій пароль ? Які негативні наслідки цього ? Як цьому запобігти ?

ЛАБОРАТОРНА РОБОТА № 3 : Типи даних Firebird. Створення РБД, знайомство з засобами її програмування.

Тема: Реляційні бази даних.

Мета: Знайомство з утилітами програмування РБД, типами даних, доменами, типами об'єктів.

Теоретичне введення.

Типи даних.

Для полів таблиць, параметрів і змінних процедур РБД InterBase/FireBird використовують наступні базові типи даних.

Назва	Розмір, байт	Дані (опис, формат, діапазон)
SMALLINT	2	Цілі -32 тис. ... +32 тис.
INTEGER	4	Цілі -2 млн. ...+2 млн.
BIGINT	8	Цілі $-(2^{63}-1)$ ... $+(2^{63}-1)$
FLOAT	4	Плаваючий формат, мантиса 7 знаків
DOUBLE PRECISION	8	Плаваючий формат, мантиса 15

		знаків
NUMERIC (***N,S***)	2,4, або 8	Фіксований формат, ***S*** знаків після коми
DECIMAL (***N,S***)	2,4, або 8	Фіксований формат, ***S*** знаків після коми
DATE	4	Дата
TIME	4	Час
TIMESTAMP	8	Дата + Час
CHAR (***N***)	***N*** байт	Символьні до 32 тис. байт (символів може бути менше, залежить від кодової сторінки)
VARCHAR (***N***)	***N*** байт	Символьні до 32 тис. байт (символів може бути менше, залежить від кодової сторінки)
BLOB	До 64 Кбайт	Файли: (фото, відео, документи ...).

Деякі пояснення на прикладах.

1. Оголошення типу NUMERIC (5,2) та DECIMAL (5,2) майже рівнозначні. Різниця полягає в наступному :

✓ Оголошення типу NUMERIC (***5,2***) означає, що поле (змінна) цього типу гарантовано матиме 5 цифр всього (або менше), з яких точно 2 цифри будуть після крапки. ***Більше 5-ти цифр не запишеться***. Число виглядатиме так : NNN.NN .

✓ Оголошення типу DECIMAL (***5,2***) означає, що поле (змінна) цього типу матиме по крайній мірі 5 цифр (або менше), але можливо й більше цифр всього. „Можливо" залежить від операційної системи і розрядності шини процесора, тобто від фізичної реалізації платформи. Після крапки, як і в попередньому прикладі, буде точно 2 цифри. ***Більше 5-ти цифр може запишеться, а може й ні***. Число виглядатиме так само : NNN.NN . В деяких випадках можливо вдасться записати на 1 цифру більше : NNNN.NN.

На більшості платформ ці типи даних повністю рівнозначні.

2. Оголошення CHAR (5) та VARCHAR (5) майже рівнозначні. Різниця полягає в тому, що при запису рядка менше 5-ти символів в полі CHAR зправа додається стільки пробілів, щоб довжина рядка завжди була 5 символів. В полі VARCHAR нічого не додається.
3. Оголошення типу FLOAT та DOUBLE PRECISION відрізняються лише точністю мантиси (гарантується відповідно 7 та 15 знаків). При запису числа 123.456789012 в поле FLOAT запишеться число 123.45679 (відбудеться округлення після сьомого знаку), а в поле DOUBLE PRECISION запишеться число 123.456789012 (без округлення).
4. Поля типу BLOB призначені для збереження файлів будь-якого типу.
5. Основний формат представлення типу DATE наступний : РРРР-ММ-ДД, справа і зліва взятий в символ „права лапка" (') наприклад : '2018-09-21' .
6. Основний формат представлення типу TIME наступний : ГГ:ХХ:СС, справа

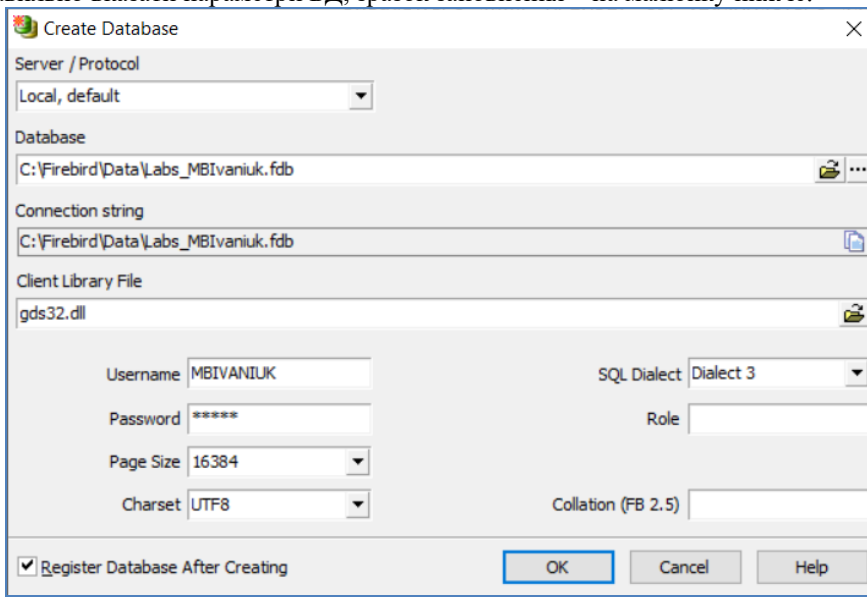
і зліва взятий в символ „права лапка” (') наприклад : '12:40:50'.

7. Основний формат представлення типу TIMESTAMP наступний : дата, плюс пробіл, плюс час, справа і зліва взятий в символ „права лапка” (') наприклад : '2018-09-21 12:40:50'.

Створення РБД.

Існує багато програм, що дозволяють створювати і програмувати реляційні бази даних InterBase/FireBird. Одна із найзручніших – безкоштовна програма “IBExpert” [8, 11]. Можливості цієї програми дуже великі. В даній роботі вивчаються лише основні, які знадобляться для створення і програмування невеликої бази даних (наприклад, при виконанні курсової роботи).

Створення РБД здійснюється через меню “База даних” → “Створити базу”. Вказується реєстраційна інформація – аналогічно, як для існуючої БД. Слід правильно вказати параметри БД, зразок заповнення – на малюнку нижче.



Дуже важливий параметр - кодування стрічок: параметр Charset, “Кодування”.

- Для баз даних із обмеженою кількістю мов вказується параметр із наборів символів ASCII (DOS, WIN, ISO, тощо). Наприклад, БД тільки із з кириличними даними у форматі Windows потрібно вказати Charset = WIN1251 (якщо цього не зробити, це прийдеться робити при створенні кожної таблиці). У цьому випадку додатково бажано уточнити параметр “Collation (FB 2.5)”, “Співставлення”, який впливає на сортування стрічкових даних. Якщо вказати Collation = WIN1251_UA, тоді стрічкові дані будуть коректно сортуватися за

українською абеткою (якщо цього не зробити, то українські літери «і», «ї» будуть найперші, тобто перед «а»).

- Для баз даних сучасних інформаційних систем, які повинні підтримувати усі набори символів (абетки) світу, вказується параметр із наборів символів UNICODE. У цьому випадку можна вказати Charset = UTF8 (одне із найрозповсюдженіших в мережі Інтернет кодування), при цьому параметр Collate залишається порожнім.

Розмір сторінки БД; бажано вказати рівний розміру кластера жорсткого диску (може бути різний, в сучасних дисків за звичай 4096 байт або вище).

Номер діалекту БД; обов'язково вказати «Діалект 3» (якщо цього не зробити, неможливо буде скористатися багатьма новими можливостями SQL).

Розміщення файла БД – вказати існуючу папку і неіснуючий файл;

Внести ім'я користувача – власника БД, і його пароль.

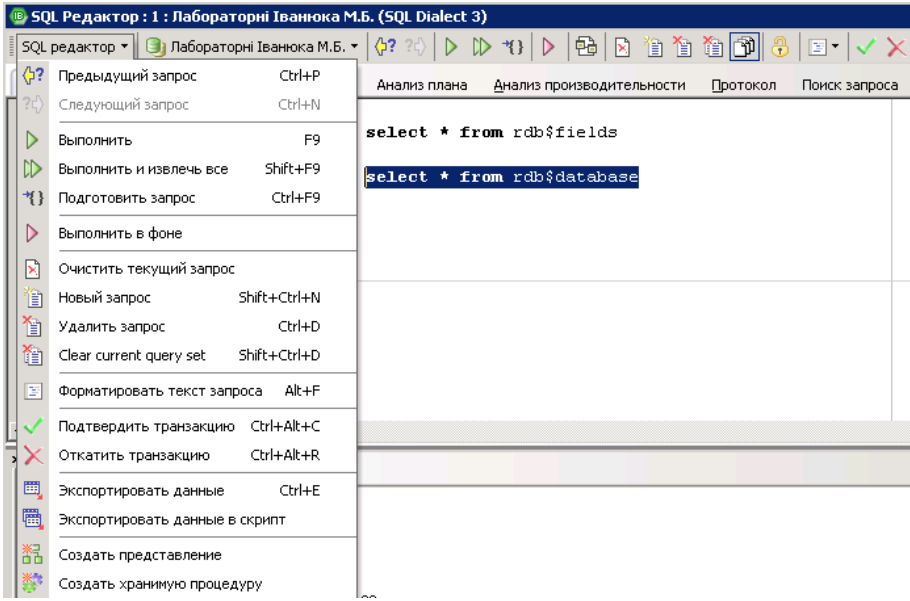
Якщо все внесено вірно, то файл БД з'явиться на диску, і необхідно буде довнести у реєстраційну інформацію лише версію сервера РСКБД та опис БД, і – натиснути кнопку «Зареєструвати». **Пам'ятайте**, що при роботі на ПК у ВНЗ вся реєстраційна інформація зберігається у єдиній для всіх студентів користувацькій базі даних IBExpert-а, тому рекомендується видалити з реєстраційної інформації хоча б пароль власника, а найкраще і ім'я також, щоб сторонні особи не могли підключитися до Вашої БД !

Опис засобів програмування РБД.

Існує два режими виконання SQL-запитів:

- ✓ „інтерактивний" режим: виконується лише один запит, якщо цей запит повертає дані – то вони відображаються на дисплеї;
- ✓ пакетний режим, частіше його називають „виконання SQL-скрипту": виконується кілька запитів підряд; запити між собою розділені символом-розділювачем (або "термінатором", це за звичай „ ; "), якщо якісь запити повертають дані – то вони або зберігаються у вивідний файл, або нікуди не виводяться (пропадають); Текст SQL-скрипта загальноприйнято зберігати у простому текстовому файлі з розширенням *.SQL.

Виконання у інтерактивному режимі відбувається через пункт меню "Інструменти" → "SQL редактор". В редакторі може бути набрано більше одного SQL-запиту, тоді запит, який потрібно виконати, попередньо виділяється, як це показано на малюнку, і для виконання натискається F9 :



Основні дії та «гарячі» кнопки панелі інструментів зрозумілі із малюнка. Окремо слід сказати про два дуже зручні пункти меню:

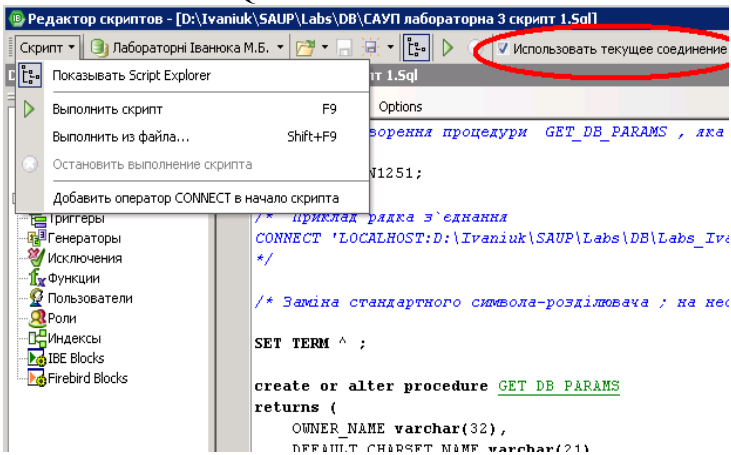
- ✓ «Експортувати дані», дозволяє для запиту, який повертає дані, одразу їх зберегти (експортувати) у файли найбільш поширених форматів;
- ✓ «Експортувати дані у скрипт», дозволяє для запиту, який повертає дані, одразу їх зберегти (експортувати) у SQL-скрипт з готовими командами INSERT, UPDATE, і т.п.; Часто це буває потрібно для масових операцій по перенесенню або редагуванню даних.

Також важливі пункти меню - це «Підтвердити транзакцію» (Commit Work) та «Відкотити транзакцію» (Rollback Work).

Виконання SQL-скриптів відбувається через пункт меню “Інструменти” → “Редактор скриптів”. Основні можливості показано на малюнку нижче. Якщо відмічена опція «Використати поточне з’єднання» (на малюнку обведено), то перед виконанням скрипта попередньо необхідно з’єднатися із базою даних (бажано лише із однією, щоб не переплутати, оскільки в IBExpert одночасно можна працювати з кількома БД), а у скрипті не повинно бути рядка з’єднання (на малюнку рядок з’єднання заключено у символи коментаря /* … */). Можна виконувати скрипти і без активного з’єднання з БД, тоді в скрипті потрібен рядок з’єднання.

За звичай окремі запити у скрипті розділені символом-розділювачем “ ; ” (крапка з комою). Кожен запит виконується по черзі, і в разі помилки вона виводиться в нижній частині вікна, а скрипт виконується далі. Це поведінка IBExpert за замовчуванням, її можна змінити в налаштуваннях (щоб по першій помилці скрипт зупинявся). Якщо зустрічаються SQL-оператори, які вибирають дані, то вони також виконуються, проте вихідні дані втрачаються. Якщо

потрібно, щоб вихідні дані зберігалися у текстовий файл, то скрипт виконують програмою “Win InterBase ISQL”.



Інколи бувають випадки, коли необхідно виконати складний SQL-оператор (що складається із кількох звичайних SQL-операторів), і тому всередині нього зустрічаються символи-розділювачі “ ; ”. Наприклад, збережена процедура, чи тригер створюються складним SQL-оператором, що складається із багатьох SQL-запитів, розділених цим символом. Щоб вийти із цієї ситуації, для програми, яка виконує SQL-скрипти, потрібно ненадовго (а саме – на час виконання такого складного SQL-запиту) тимчасово підмінити символ-розділювач “ ; ” якимось іншим, нестандартним, але таким, який точно не зустрінеться у цьому складному SQL-операторі. Наприклад, замінити розділювач “ ; ” на символ “ ^ ” можна так:

1) Встановити розділювач “ ^ ” замість “ ; ” :
   SET TERM ^ ;
2) Виконати складний SQL-запит :
   create or alter procedure …
   …
   end
   Важливо, щоб всередині цього складного SQL-запиту жодного разу не з’явився символ “ ^ ”.
3) До останнього рядка складного SQL-запиту дописати розділювач:
   end^
4) Встановити стандартний розділювач “ ; ” замість “ ^ ” :
   SET TERM ; ^
5) Продовжити скрипт зі стандартним розділювачем.

Важливо !!! При написання SQL-скриптів не забувайте про Commit Work !

Завдання на лабораторну роботу.

Важливо : в даній роботі студентом самостійно створюється база даних

«Лабораторні студента…», і в наступних роботах ця БД буде поступово наповнюватися даними. Тому в кінці кожної роботи холодну копію даної БД (включно із базою паролів) бажано копіювати на надійні носії інформації, щоб зберегти її до кінця семестру.

1. Під'єднатись до бази даних "Службовці".
2. Переглянути SQL-скрипти створення доменів, оголошених в цій базі даних. Всі об"єкти баз даних відкриваються для перегляду в експлорері БД (DataBase Explorer) подвійним кліком мишки. В закладці метаданих "Скрипт" переглядається SQL-скрипт створення об"єкту.
3. Переглянути SQL-скрипти створення таблиць. Виявити декілька таблиць, при створенні яких було використано домени.
4. Виявити ті таблиці, при створенні яких було використано домени BUDGET, PONUMBER і PROJNO. Визначити кодову сторінку символьних доменів. Описати, які обмеження на значення використано в цих доменах.
5. Під'єднатись до бази даних "Приклад 1".
6. Переглянути SQL-скрипти створення таблиць. Виявити декілька таблиць, при створенні яких не було використано жодного домену, лише базові типи даних.
7. Виявити ті таблиці, при створенні яких було використано домени DOC_NOMER, P_I_B і GROSHI. Визначити кодову сторінку символьних доменів. Описати, які обмеження на значення, і які значення за замовчуванням використано в цих доменах.
8. Створити базу даних студента. Попередньо створити новий окремий каталог жорсткого диску в папці студента, у якому зберігатиметься база даних (до прикладу – "D:\Ivaniuk\FB\Labs\DB\"). Назва файлу БД: префікс "Labs_", суфікс "_Db", і між ними прізвище в латинській транскрипції, наприклад: "Labs_Ivaniuk_Db.Fdb". Користувачем-власником бази даних повинен бути студент, що виконує лабораторну роботу. У попередніх роботах цей користувач повинен бути створений, ім'я за шаблоном «ініціали + прізвище в латинській транскрипції» ! Якщо не був створений – попередньо створіть ! Діалект 3, набір символів UTF8 .
9. Одразу після створення - зареєструвати цю БД (папка Database Explorer-а – це папка студента). Опис БД за шаблоном: «Лабораторні Іванюка М.Б.».
10. Внести опис бази даних : в SQL-редакторі виконати наступний запит:
```
update rdb$database
   set rdb$description=
'БД л.р. Іванюка Михайла Борисовича, ЕК-5, 2023 р.'
```
Звичайно, кожен студент вносить свої дані !!!
11. Не забути підтвердити транзакцію !!!
12. В редакторі скриптів виконати SQL-скрипт "Лабораторна 3 скрипт 1.Sql" – в результаті буде створено процедуру GET_DB_PARAMS. В експлорері БД (Database Explorer) впевнитися, що процедура створилася.
13. В експлорері БД навчитися переглядати опис бази даних: виділити (відмітити мишкою) псевдонім бази даних, а в нижній частині експлорера

знайти щойно внесений опис бази.

14. Там же (в експлорері БД) навчитися переглядати опис об'єктів: виділити (відмітити мишкою) папку із об'єктами (у даному випадку – із процедурами), та знайти опис процедури GET_DB_PARAMS.
15. Там же (в експлорері БД) навчитися переглядати описи параметрів об'єктів: виділити (відмітити мишкою) конкретний об'єкт у папці (у даному випадку – виділити процедуру GET_DB_PARAMS), та знайти опис параметрів процедури GET_DB_PARAMS.
16. Відкрити подвійним кліком об'єкт БД (процедуру GET_DB_PARAMS) – при цьому відкривається редактор об'єктів. Вивчити редактор об'єктів : знайти SQL-скрипт створення об'єкта, права на цей об'єкт, залежності (цього від інших об'єктів і інших від цього), опис. Переглянути вхідні, вихідні параметри і внутрішні змінні процедури GET_DB_PARAMS.
17. На жорсткому диску комп'ютера у каталозі студента створити нову папку для даної роботи (до прикладу – "D:\Ivaniuk\FB\Labs\Лаб3\").
18. В SQL-редакторі виконати наступний запит :
    `Select * from GET_DB_PARAMS`
19. Виконати збереження (експорт) результатів запиту в MS Excel (через меню SQL-редактора «Експортувати дані»). Файл зберегти у щойно створену папку, назвати файл по своєму прізвищу, наприклад "Іванюк.Xls".
20. Виконати збереження (експорт) результатів запиту в XML-таблицю (тип експорту "XML SpreadSheet"). Файл зберегти у щойно створену папку, назвати файл по своєму прізвищу, наприклад "Іванюк.xml".
21. Виконати збереження (експорт) результатів запиту в MS Word. Файл зберегти у щойно створену папку, назвати файл по своєму прізвищу, наприклад Іванюк.Doc .
22. Всі отримані файли переглянути, при необхідності відформатувати і зберегти форматування, закрити.
23. Виконати збереження (експорт) результатів запиту в SQL-скрипт (через меню SQL-редактора «Експортувати дані у скрипт»). Опції «Експортувати в: Файл», «Експортувати як : INSERT оператори». Файл скрипту зберегти у щойно створену папку, назвати файл по своєму прізвищу, наприклад "Іванюк.Sql". Переглянути цей скрипт у редакторі скриптів. Через меню "Скрипт" → "Додати оператор CONNECT у початок скрипта" додати рядок з'єднання із поточною базою даних. Зберегти.
24. В SQL-редакторі виконати наступний запит :
    `Execute procedure GET_DB_PARAMS`
    При необхідності так розширити нижню частину вікна SQL-редактора, щоб було видно всі вихідні параметри процедури. Зробити знімок дисплея, і зберегти зображення у щойно створену папку в форматі JPEG, назвати файл по своєму прізвищу, наприклад "Іванюк.Jpg".
25. Зробити холодну копію БД «Лабораторні…» у цю ж папку.
26. Закрити всі програми, всі файли у створеній папці (сім) заархівувати.
27. Надіслати файл архіву на електронну адресу свого викладача, у листі вказати «Студент "*ПІБ повністю*", лаб.робота номер 3».

28. Холодну копію власної бази даних «Лабораторні…» бажано зкопіювати на власну флешку, оскільки дана БД буде використовуватися і далі – до кінця семестру.

Контрольні запитання.

1. Що таке домен, в яких випадках і для чого його використовують ?
2. Скільки SQL-запитів можна одночасно виконувати в режимі інтерактивного вводу і в режимі скриптів ?
3. Який символ по замовчуванню слугує для розділення SQL-запитів ?
4. Для чого в реляційній БД потрібні таблиці ?
5. Які дані в реляційній БД зберігаються НЕ в таблицях, а якимось іншим чином ?

ЛАБОРАТОРНА РОБОТА № 4 : Базові можливості основних SQL-операторів, створення переглядів.

Тема: Реляційні бази даних.

Мета: Продовження вивчення програми IBExpert, SQL-тренінг (прості запити), знайомство із псевдонімами і переглядами.

Теоретичне введення.

Основні SQL-оператори по роботі із даними в таблицях.

Синтаксис найпростішого оператора SELECT :

```
SELECT {список полів}
  FROM {список таблиць}
  [WHERE {умови зв'язування і відбору записів}]
  [ORDER BY {список полів для сортування} [DESCENDING]]
```

Тут і далі в квадратні дужки взято необов'язкові елементи.

Зв'язування використовують у випадку, якщо запит обробляє кілька таблиць, які логічно зв'язані. **Правило зв'язування**: якщо в списку таблиць після FROM є перелік із N таблиць, то для правильного зв'язування по ключовим полям після WHERE повинно бути як мінімум N-1 правил зв'язування, об'єднаних логічним AND. Якщо правило зв'язування для якоїсь таблиці не написати, то замість логічно зв'язаних даних буде вибрано декартовий добуток цієї таблиці з іншими (декартовий добуток – це всі можливі комбінації записів двох наборів даних за правилом «кожен-з-кожним»).

Сортування використовують для зручного виведення результатів запиту. Якщо зовсім не вказати ORDER BY, то передбачити порядок виведення результатів неможливо (з деяким припущенням можна стверджувати, що порядок записів буде випадковим). Саме тому слід сортування бажано використовувати завжди. **Правило сортування**: після ORDER BY можна

вказувати лише назви тих полів, які присутні перед FROM. За замовчуванням вказані після ORDER BY поля відсортовані у прямому порядку (від меншого до більшого, за зростанням: ASCENDING). Необов'язкова опція DESCENDING (можна писати скорочено DESC) після назви якогось поля означає сортування у зворотному порядку (від більшого до меншого, за зменшенням).

Відбір записів можна зробити з допомогою опцій WHERE. Відбір по WHERE здійснюється в процесі виконання запиту: якщо не виконується умова відбору, вказана у WHERE, запис відкидається. Тому в опції WHERE вказуються умови відбору по значеннях полів (або математичних виразів над полями) у кожному окремо взятому запису (рядку) набору даних. Використання агрегатних функцій в опції WHERE не дозволене. Приклад :

```
SELECT A.F1, A.F2, A.F3, B.F2, C.F2
  FROM A, B, C
  WHERE B.F1 = A.F4 -- Правило зв'язування таблиць B та A
  AND C.F1 = A.F5 -- Правило зв'язування таблиць C та A
  AND C.F3 > 0 -- Правило відбору по числовому полю C.F3
  AND B.F2 LIKE 'Я%' -- Правило відбору по текстовому полю B.F2
  ORDER BY A.F1, A.F2 DESC, A.F3;
```

Синтаксис базового оператора INSERT :

```
INSERT INTO {назва таблиці} [({список назв полів})]
VALUES ({список значень полів})
```

Якщо список назв полів відсутній, тобто після назви таблиці одразу йде слово VALUES, то передбачається, що вказано список значень всіх полів таблиці, причому в тому порядку, в якому вони були впорядковані при створенні таблиці. Це допустимо, проте дуже ненадійно, оскільки після додавання нових полів або зміни порядку існуючих оператор INSERT перестане працювати.

Приклад оператора INSERT:

```
INSERT INTO A (F1,F2,F3) VALUES (7,'Яна','2018-09-22');
```

Синтаксис базового оператора UPDATE :

```
UPDATE {назва таблиці}
  SET {список назва поля = нове значення}
  [WHERE {умови відбору записів}];
```

Приклад оператора UPDATE :

```
UPDATE A
  SET F2='Іван', F3='2018-09-01'
  WHERE F1=7;
```

Синтаксис базового оператора DELETE :

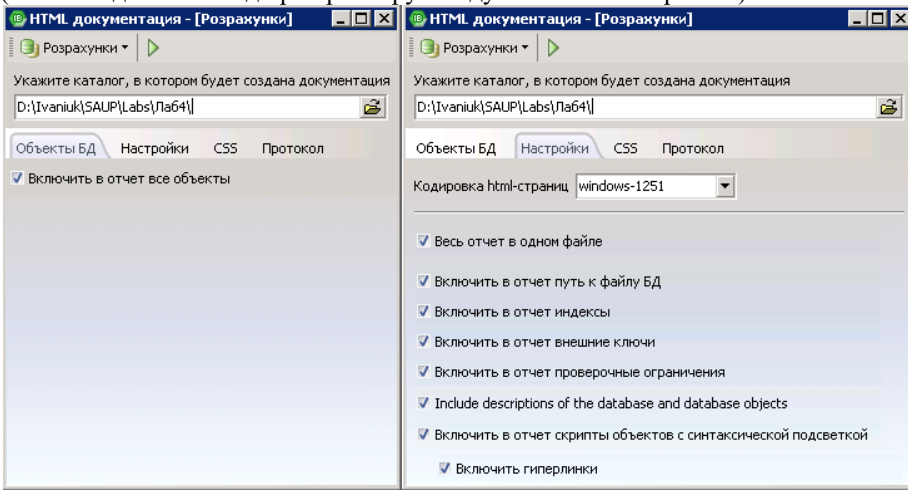
```
DELETE FROM {назва таблиці}
  [WHERE {умови відбору записів}];
```

Приклад оператора DELETE :

```
DELETE FROM A
  WHERE F1<3;
```

Вивчення схеми БД.

Для полегшення розуміння схеми і супроводу БД необхідно вносити в БД описи абсолютно усіх об’єктів, причому робити це негайно (одразу при створенні, при внесені змін). Тоді в IBExpert достатньо легко згенерувати документацію по БД FireBird. Це робиться через пункт меню “Інструменти” → “HTML-документація”. Рекомендовані налаштування показані на малюнку (може знадобитися підбір параметру «Кодування html-сторінок») :



Якщо в БД вносяться, і що не менш важливо – підтримуються в актуальному стані - описи усіх об'єктів, то базу даних значно легше вивчати, розуміти і супроводжувати. Описи основних об’єктів переглядаються в IBExpert прямо у вікні “Експлорер БД” (DataBase Explorer), якшо відмітити папку із цими об’єктами. Проте значно зручніше переглядати описи, відкривши потрібний об’єкт подвійним кліком мишки : тоді в закладці «Опис» переглядають та змінюють опис самого об’єкта, а в інших закладках переглядають та змінюють описи його складових частин (поля таблиць, параметри і змінні процедур, тощо).

Завдання на лабораторну роботу.

Важливо : в даній роботі студентом із приготованого викладачем SQL-скрипта створюється база даних «Розрахунки», в багатьох роботах ця БД буде використовуватися для SQL-тренінгу. Тому в кінці кожної роботи холодну копію даної БД (включно із базою паролів) бажано копіювати на надійні носії інформації, щоб зберегти її до кінця семестру.

1. База даних “Розрахунки” була створена раніше викладачем, власник – користувач MBIVANIUK. Потім база була експортована програмою IBExpert в текстовий SQL-скрипт "Лабораторна 4 скрипт 1.Sql". Новим власником повинен стати студент, що виконує лабораторну роботу. У

попередніх роботах цей користувач повинен бути створений, ім'я за шаблоном «ініціали + прізвище в латинській транскрипції» ! Якщо не був створений – попередньо створіть ! Якщо був – пригадайте пароль.

2. В каталозі студента створити нову папку для даної роботи (наприклад "D:\Ivaniuk\FB\Labs\Лаб4\"). Перенести в цей новий каталог SQL-скрипт "Лабораторна 4 скрипт 1.Sql" (див. Додаток 1).
3. В програмі IBExpert відкрити Редактор скриптів. Не під'єднуючись до жодної БД, відкрити SQL-скрипт "Лабораторна 4 скрипт 1.Sql". Це SQL-скрипт створення і заповнення бази даних "Розрахунки". В цьому скрипті потрібно зробити заміни :
   2.1. Всі входження логіну MBIVANIUK замінити на логін даного студента, що виконує лабораторну роботу (у IBExpert пункт меню "Редактор" → "Заміна" → "Замінити ВСЕ" → мінімум 20 входжень). Зберегти зміни (у IBExpert пункт меню "Редактор" → "Зберегти в файл").
   2.2. В операторі "CREATE DATABASE …" замінити пароль "1" на пароль користувача FireBird – даного студента, що виконує лабораторну роботу. Зберегти.
   2.3. В операторі "CREATE DATABASE …" замінити шлях до файла БД, який буде створюватися скриптом, на той же каталог студента, в якому знаходяться БД «Службовці» і «Приклад 1». Назву файлу БД "Billing.Fdb" не змінювати. Зберегти. Впевнитися, що вказана папка існує, а вказаний файл – не існує.
4. Виконати SQL-скрипт. Якщо під час виконання скрипта були помилки:
   4.1. Подвійний клік на кожній помилці переведе курсор на оператор - джерело помилки.
   4.2. Усунути причину помилки (редагувати SQL-оператор), зберегти.
   4.3. Якщо файл БД з'явився – знищити, і повторно виконати скрипт.
5. Після безпомилкового виконання - впевнитися, що файл БД з'явився. Зареєструвати цю БД (папка Database Explorer-а – це папка студента). Опис (назва) БД : «Розрахунки».
6. З'єднатися із БД. Згенерувати HTML-документацію (каталог для збереження вказати щойно створений).
7. Користуючись цією документацією, а також відкриваючи у експлорері БД (DataBase Explorer) всі таблиці, вивчаючи їх описи та дані, - зрозуміти схему БД (що і як в БД зберігається, як взаємозв'язані таблиці).
8. Вивчити домени бази даних «Розрахунки». Розібратися, які обмеження накладаються на значення доменів, особливо – стрічкових, оскільки це знадобиться при внесенні даних в таблицю.
9. Відкрити в IBExpert одночасно два вікна :
   - ✓ Відкрити SQL-Редактор (через меню Інструменти);
   - ✓ У експлорері БД відкрити папку «Скрипти/Блоки», і подвійним кліком відкрити скрипт «Лабораторна № 4». Він порожній. У подальшому виконанні роботи всі SQL-запити інтерактивно будуть виконуватися у SQL-Редакторі. Кожен вдало виконаний SQL-запит копіюванням/вставкою потрібно переносити у скрипт «Лабораторна

№ 4», вносити послідовно вниз один за одним (з коментарями), і зберігати цей скрипт. Збереження в базу даних робиться не через меню IBExpert, а лише єдиним способом – натисканням в панелі інструментів «Редактора скриптів» піктограми «Дискетка» : “Зберегти SQL скрипт” (“Save SQL script”). Якщо цей же скрипт потрібно зберегти у файл на диск, це робиться черех меню IBExpert “Редактор” → “Зберегти як”.

По закінченню виконання роботи база даних передається Вашому викладачу на електронну пошту, - на перевірку.

10. SQL-Редактор: Оператор SELECT : переглянути всі поля таблиці CITY; потім STREET. Перенести вдало виконані запити у скрипт (додати у кінці скрипта), зберегти.
11. SQL-Редактор: Оператор INSERT : внести в таблицю CITY три нових довільних населених пункти, і в таблицю STREET три нові довільні вулиці. Дані вводити коректні (реальні пункти і вулиці !). Підтвердити транзакцію. Перенести вдало виконані запити у скрипт, зберегти скрипт.
12. SQL-Редактор: Оператор SELECT : переглянути всі поля таблиці CITY, потім таблиці STREET, обидва запити відсортувати по ключовому (першому) полю; потім по назві (населеного пункту; вулиці). Перенести вдалі запити у скрипт, зберегти скрипт.
13. SQL-Редактор: Оператор UPDATE : назву населеного пункту «Острог» змінити на «Остріг» в обох полях; аналогічно змінити «Кузецовськ» на «Вараш» ; коротку назву вулиці «Академіка Грушевського» змінити на «Ак. Грушевського». **Підтвердити транзакцію**. Перенести вдало виконані запити у скрипт, зберегти скрипт.
14. SQL-Редактор: Оператор DELETE : видалити перед-останній внесений населений пункт із тих що вносили самостійно; видалити першу із тих трьох вулиць які вносили самостійно. **Відкотити транзакцію**. Перенести вдало виконані запити у скрипт, зберегти скрипт.
15. SQL-Редактор: Оператор SELECT : вивести не всі поля, а лише ключове (перше) поле та повну назву із таблиці CITY ; потім ключове поле і скорочену назву із таблиці STREET – обидва запити відсортувати по назві. Перенести вдалі запити у скрипт.
16. SQL-Редактор: Оператор SELECT : вивести не не всі поля, а лише ключове (перше) поле та повну назву із таблиці CITY, відібрати не всі записи таблиці, а лише ті, де назва населеного пункту починається на літеру “Р”, відсортувати по назві. Аналогічно з таблиці STREET ключове поле і повну назву вулиць, які починаються на “М”, відсортувати по назві. Перенести вдалі запити у скрипт, зберегти скрипт.
17. SELECT : переглянути всі поля таблиці CLIENT. Запит – у скрипт.
18. SELECT : зв’язати таблиці CLIENT, CLIENTTYPE, CITY, STREET у один запит, вивести всі поля всіх таблиць, відсортувати по особовому рахунку (VACCOUNT). Якщо записів вибралося стільки ж, скільки є у окремо взятій таблиці CLIENT – значить умови зв’язування таблиць написано вірно. Запит – у скрипт.

19. SELECT : зробити те ж саме, що в попередньому пункті (вивести всі поля всіх чотирьох таблиць, відсортувати по особовому рахунку), але запит написати заново, і по-іншому : кожній таблиці присвоїти короткий псевдонім, наприклад так : … from CLIENT c, CLIENTTYPE t, CITY y, STREET s where … . Умови зв'язування і сортування набирати заново, використовуючи ці псевдоніми. Відчути переваги підсказок, які дає IBExpert, та переваги які дає однотипне іменування полів різних таблиць. Запит – у скрипт.
20. SELECT : зробити те ж саме, що в попередньому пункті, проте вивести не всі поля, а лише такі :
    - ✓ Ідентифікатор (ID) клієнта
    - ✓ Особовий рахунок клієнта
    - ✓ ПІБ особи/назва організації повна
    - ✓ Назва типу клієнта повна
    - ✓ Назва населеного пункту повна
    - ✓ Назва вулиці повна
    - ✓ Номер будинку
    - ✓ Номер квартири
    - ✓ Контактний номер телефону клієнта

    Відсортувати по особовому рахунку, запит – у скрипт.
21. На основі SELECT-у з попереднього пункту створити перегляд (VIEW) із назвою CLIENT_DATA, запит CREATE VIEW – у скрипт.
22. INSERT : внести в таблицю CLIENT одного нового клієнта. Особовий рахунок його повинен бути "20000nn", де "nn" – номер студента у списку групи, спереду доповнений нулем при потребі; ПІБ повний і скорочений – ПІБ студента; тип клієнта – фізична особа ; жити повинен клієнт у тому населеному пункті і на тій вулиці, які студент вводив самостійно ; будинок і квартира довільні ; контактний телефон – довільний номер мобілки. Підтвердити транзакцію. Запит – у скрипт.
23. INSERT : внести в таблицю CLIENT ще одного нового клієнта – юридичну особу. Підтвердити транзакцію. Запит – у скрипт.
24. Виконати SELECT із створеного щойно перегляду CLIENT_DATA, відсортувати по ідентифікатору (ID) клієнта, переконатися що два нових клієнти з'явилися. Запит – у скрипт.
25. Відкрити подвійним кліком редактор таблиці CLIENT. Внести ще два клієнти (фізичну і юридичну особу), проте зробити це не через INSERT, а в закладці редактора «Дані», користуючись можливостями IBExpert з випадаючими списками. **Дані вводити коректні і без граматичних помилок!!!** Підтвердити транзакцію в редакторі даних, і закрити його. Повернутися в SQL-редактор, знову виконати запит із перегляду CLIENT_DATA, переконатися що два нових клієнти з'явилися.
26. SELECT : переглянути всі дані перегляду CLIENT_DATA, відсортувати по зростанню ID клієнта, потім – по зменшенню ID. Обидва запити – у скрипт.
27. UPDATE : самому першому у списку клієнту-фізичній особі (з найменшим

ID) одним SQL-запитом змінити тип на юридичну особу; перед повною і короткою назвою спереду додати рядок “ФОП ”; до особового рахунку ззаду додати “/2”. Підтвердити транзакцію, Запит – у скрипт.

28. SELECT : переглянути всі дані таблиці OPERATION. Запит – у скрипт.
29. DELETE : видалити всі дані (тобто без умови WHERE) з таблиці OPERATION. Транзакцію НЕ ПІДТВЕРДЖУВАТИ і НЕ ВІДМІНЯТИ ! Запит – у скрипт.
30. SELECT : переглянути всі поля таблиці OPERATION, впевнитися що вона порожня. ВІДМІНИТИ транзакцію, виконати той же SELECT, впевнитися що дані є.
31. Зберегти скрипт у базу даних, і для підстраховки – у файл (меню “Редактор” → “Зберегти у файл”) - в папку, створену для роботи № 4.
32. Від’єднатися від БД, зробити холодну копію БД «Розрахунки» у цю ж папку.
33. Закрити всі програми, всі файли у створеній папці заархівувати.
34. Надіслати файл архіву на електронну адресу свого викладача, у листі вказати «Студент «ПІБ повністю”, лаб.робота номер 4».
35. Базу даних «Розрахунки» бажано скопіювати на власну флешку, оскільки дана БД буде використовуватися і далі – до кінця семестру.

Контрольні запитання.

1. Напишіть базовий синтаксис оператора SELECT.
2. Напишіть базовий синтаксис оператора INSERT.
3. Напишіть базовий синтаксис оператора UPDATE.
4. Напишіть базовий синтаксис оператора DELETE.
5. Напишіть приклад оператора SELECT із зв’язуванням трьох таблиць та сортуванням по кількох полях без групування.
6. Напишіть приклад оператора SELECT із однієї таблиці: розрахунок суми по одному з полів таблиці, та сортування по двох інших полях.
7. Напишіть приклад оператора SELECT із зв’язуванням двох таблиць, пошуком максимального значення по полю однієї таблиці, та зворотнім сортуванням по двом іншим полям.
8. Напишіть два приклади оператора INSERT – з переліком назв полів і без. Який варіант більш надійний, і чому ?
9. Напишіть приклад оператора UPDATE без WHERE. Які записи таблиці будуть відредаговані ?
10. Напишіть приклад оператора UPDATE, в якому редагуються лише ті записи, у яких ключове поле більше 7-ми і менше 12-ти.
11. Напишіть приклад оператора UPDATE, в якому редагуються лише ті записи, у яких одне із полів починається на велику літеру ’Я’.
12. Напишіть приклад оператора DELETE без WHERE. Які записи таблиці будуть видалені ?
13. Напишіть приклад оператора DELETE, в якому видаляються лише ті записи, у яких ключове поле менше 7-ми або більше 12-ти.

14. Напишіть приклад оператора DELETE, в якому видаляються лише ті записи, у яких одне із полів починається на велику літеру ’Ю’ або на малу ’ю’.

ЛАБОРАТОРНА РОБОТА № 5 : Поглиблене вивчення основних SQL-операторів.

Тема: Реляційні бази даних.

Мета: Продовження SQL-тренінгу: складні SQL-запити (групування, вкладені запити, фільтрація) та робота зі збереженими процедурами.

Теоретичне введення.

Групування в SQL-операторі SELECT.

Синтаксис базового (без розширеного синтаксиса) оператора SELECT :

```
SELECT {список полів}
  FROM {список таблиць}
  [WHERE {умови зв’язування і відбору записів}]
  [GROUP BY {список полів для групування}]
  [HAVING {умови відбору записів після групування}]
  [ORDER BY {список полів для сортування} [DESCENDING]]
```

Тут і далі в квадратні дужки взято необов’язкові елементи.

Групування в операторі SELECT застосовують, коли потрібно використати агрегатні функції. Термін “агрегатні” означає обробку набору значень, тобто агрегацію значення поля із різних записів таблиці. Ось деякі агрегатні функції :

- ✓ сума значень поля або виразу багатьох записів: SUM(поле/вираз);
- ✓ максимальне значення поля або виразу: MAX(поле/вираз);
- ✓ кількість записів: COUNT(поле/*);
- ✓ і т.п.

В круглих дужках може бути назва поля або математичний вираз за участю полів одного запису набору даних. Спочатку вибирається значення поля (або обчислюється вираз) у кожному записі (рядку), потім функція обробляє набір записів. **Правило групування**: після GROUP BY вказуються всі ті поля, які після SELECT не включені в агрегатні функції, а вибираються окремо. Єдиний випадок, коли при використанні агрегатних функцій не потрібне групування – якщо після SELECT відсутні прості поля (всі поля запиту агрегатні), наприклад:

```
SELECT MAX(BUDGET) "Макс.бюджет",
COUNT(DEPT_NO) "К-сть відділів"
FROM DEPARTMENT WHERE LOCATION LIKE '%Boston%';
```

Якщо застосовано групування, то крім опції WHERE відбір (фільтрацію) результату виконання SELECT можна зробити також з допомогою HAVING.

Пригадайте: відбір по WHERE здійснюється в процесі виконання запиту: якщо не виконується умова відбору, вказана у WHERE, запис відкидається

одразу. Тому в опції WHERE вказуються умови відбору по значеннях полів (або математичних виразів над полями) у одному запису (рядку) набору даних. Використання агрегатних функцій в опції WHERE не дозволене.

Відбір в опції HAVING здійснюється вже після закінчення виконання запиту, на основі вже відібраних даних. Тому HAVING використовують лише у випадку потреби відбору записів по результатах групування, і лише по значеннях агрегатних функцій (або операцій над агрегатними функціями). Приклад використання обох способів :

```
SELECT A.F1, A.F2, A.F3, B.F2, C.F2, SUM(A.F6), SUM(C.F3)
  FROM A, B, C
  WHERE B.F1 = A.F4 -- Правило зв'язування таблиць B та A
  AND C.F1 = A.F5 -- Правило зв'язування таблиць C та A
  AND C.F3 > 0 -- Правило відбору по числовому полю C.F3
  AND B.F2 LIKE 'Я%' -- Правило відбору по текстовому полю B.F2
  GROUP BY A.F1, A.F2, A.F3, B.F2, C.F2
  HAVING SUM(A.F6)>100 -- Правило відбору по агрегатній функції
  ORDER BY A.F1, A.F2 DESC, A.F3;
```

Основні SQL-оператори по роботі зі збереженими процедурами.

Збережені процедури можуть не мати вихідних параметрів (змінних), тоді вони просто виконують якісь дії в базі даних. Якщо збережені процедури мають вихідні параметри (змінні), то крім виконання якихось дій вони повертають у цих змінних якісь значення.

Процедури, що мають вихідні параметри, поділяються на два види: селективні (selectable) і прості (не селективні, not selectable). Не селективна процедура повертає значення вихідних параметрів – аналогічно до процедур чи функцій у звичайних мовах програмування. Селективна процедура повертає набір даних – щось на зразок таблиці чи перегляду бази даних. У наборі вихідних даних селективної процедури заздалегідь невідома кількість записів (рядків): нуль (немає жодного запису), один, або більше записів (рядків) значень вихідних параметрів. Процедури, які взагалі не мають вихідних параметрів (змінних), усі прості – вони не можуть бути селективними.

Збережену процедуру можна виконати двома способами: або через оператор прямого виклику EXECUTE (в деяких РСКБД це оператор CALL), або в розширеному синтаксисі SQL-оператора SELECT.

Синтаксис оператора EXECUTE :

```
EXECUTE PROCEDURE {назва процедури}
  ( {список значень всіх вхідних параметрів} );
```

Якщо процедура має вихідні параметри, то при такому виклику може бути повернуте лише одне значення. Якщо процедура селективна, то при такому виклику завжди буде повернуто один запис (рядок) значень вихідних параметрів, а всі наступні рядки (навіть якщо вони є) будуть втрачені.

Приклад оператора EXECUTE :

```
EXECUTE PROCEDURE PROC_1 (7,'Яна','2018-09-22');
```
Тут процедура з назвою PROC_1 має три вхідних параметри: чисельний, стрічковий, і дату. Якщо ця процедура має чотири вихідних параметри з назвами P1, P2, P3 та P4 (відповідно цілочисельний, чисельний з фіксованою крапкою, стрічковий, та дата+час), то при інтерактивному виконанні процедури на дисплеї може бути відображено, наприклад, таке :
```
P1 = 12
P2 = 7.24
P3 = 'Нарахування'
P4 = '2018-09-16 12:45:00'
```
Розширений синтаксис оператора SELECT передбачає виклик селективних збережених процедур. При цьому перед FROM та після ORDER BY можна вказати назви вихідних параметрів процедури, а після FROM назва процедури та в дужках – список значень всіх вхідних параметрів.

Приклад виконання селективної процедури :
```
SELECT P1, P2, P4 FROM PROC_1 (7,'Яна','2018-09-22')
ORDER BY P1;
```
Приклад показує, що вхідні параметри в дужках після назви процедури потрібно вказати всі (інакше виникне помилка «невірна кількість/тип вхідних параметрів»), а от вихідні параметри можна вказувати не всі, а лише потрібні. При інтерактивному виконанні такого запиту на дисплеї може бути відображено, наприклад, таке :
```
      P1    P2          P4
      ==    ==          ==
      12    7.24        2018-09-16 12:45:00
      14    -0.25       2017-31-12 00:00:00
      21    3.14        2016-11-16 10:10:00
```
Із наведеного прикладу видно, що селективна процедура розглядається як набір даних, подібно до таблиці або перегляду (VIEW). Тому селективні процедури можна комбінувати в операторі SELECT з іншими наборами даних (таблицями, переглядами - із певними обмеженнями). Також зі сказаного вище ясно, що селективні процедури можна викликати двома можливими способами – і в операторі EXECUTE, і в операторі SELECT. Прості (не селективні) процедури цієї можливості не мають.

Селективною процедуру робить спеціальна SQL-функція «SUSPEND;» («відкласти»). В деяких РСКБД це функція «PIPE;» («труба»). Щоразу, коли зустрічається ця функція, поточні значення вихідних параметрів «відкладаються» у вихідний набір даних (або «виштовхуються» у вихідний набір даних ніби в трубу).

Розширений синтаксис SQL – вкладені запити.

Вкладений запит – це будь-який коректний SQL-запит SELECT, який заключається в круглі дужки, і поміщається всередину іншого SQL-запиту (тобто один запит «вкладається» в інший).

Є три варіанти вкладених запитів :

1. Після SELECT перед FROM. У цьому випадку вкладений запит повинен мати лише одне поле, а в цьому полі повертати не набір даних, а тільки одне значення (не менше і не більше !) або NULL. Такому вкладеному запиту обов'язково присвоюється псевдонім – назва поля. Приклади :

   1.1. Читання у вкладеному запиті даних із таблиці, яка не використовується у зовнішньому запиті :

```
SELECT A.ID, A.NAME,
       ( SELECT SUM(B.OPLATA) FROM B
           WHERE B.ID=A.ID ) as SUMA_OPLAT
FROM A ORDER BY A.ID;
```

   У цьому прикладі вкладений запит обчислює суму оплат по таблиці B, такий запит гарантовано поверне лише одне число, а не набір даних. Зверніть увагу, що вкладений запит використовує поле зовнішнього запиту A.ID, як параметр для прив'язки до потрібного клієнта.

   1.2. Виклик у вкладеному запиті селективної збереженої процедури, яка повертає лише одне значення :

```
SELECT A.ID, A.NAME,
       ( SELECT G.ADDRESS FROM
           GET_CLIENT_ADDRESS(A.ID) G) as ADRESA
FROM A ORDER BY A.ID;
```

   У цьому прикладі вкладений запит викликає збережену процедуру, яка визначає адресу клієнта по заданому ID, тобто також гарантовано поверне лише один рядок, а не набір даних. Зверніть увагу, що вкладений запит використовує поле зовнішнього запиту A.ID, як вхідний параметр процедури GET_CLIENT_ADDRESS - для прив'язки до потрібного клієнта.

2. Після FROM перед WHERE. У цьому випадку вкладений запит може мати довільну кількість полів, та повертати довільну кількість значень (в тому числі і жодного). Такий вкладений запит розглядається як ще одна таблиця. У цьому випадку вкладеному запиту обов'язково присвоюється псевдонім – назва таблиці. Приклад :

   2.1. Читання у вкладеному запиті даних із таблиці, яка не використовується у зовнішньому запиті :

```
SELECT A.ID, A.NAME, C.DATE, C.OPLATA
FROM A,
     ( SELECT B.ID, B.DATE, B.OPLATA FROM B
       WHERE B.DATE<'2018-09-01' ) as C
WHERE C.ID=A.ID
ORDER BY A.ID;
```

   У цьому прикладі вкладений запит - це набір даних із трьох полів. Зверніть увагу, що в цьому випадку прив'язка вкладеного запиту до зовнішнього відбувається аналогічно до зв'язування звичайних таблиць.

3. В умові WHERE. У цьому випадку вкладений запит розглядається як додаткова умова пошуку, він повинен мати лише одне поле (не менше і не

більше !). В цьому полі можна повертати або набір даних (повинна бути використана тільки умова пошуку IN), або тільки одне значення (може бути використана будь-яка умова пошуку: IN, =, >, <, …). Такому вкладеному запиту псевдонім не потрібен. Приклади :

3.1. Пошук платежів по клієнтах типу ’ФОП’:

```
SELECT * FROM PAYMENTS
       WHERE CLIENT_ID IN
      (SELECT ID FROM CLIENTS WHERE TYPE='ФОП')
```

У цьому прикладі вкладений запит - це набір даних із ID усіх клієнтів, обраних за типом ’ФОП’.

3.2. Пошук платежів по певному клієнту:

```
SELECT * FROM PAYMENTS
       WHERE CLIENT_ID =
      (SELECT ID FROM CLIENTS WHERE NAME='Верес')
```

У цьому прикладі вкладений запит - це ID одного клієнта, обраного за назвою.

Завдання на лабораторну роботу.

1. В каталозі студента створити нову папку для даної роботи (наприклад “D:\Ivaniuk\FB\Labs\Лаб5\”).
2. В програмі IBExpert з’єднатися із базою даних «Розрахунки». Відкрити Редактор скриптів. Відкрити SQL-скрипт "Лабораторна 5 скрипт 1.Sql". Це SQL-скрипт створення процедури. Виконати скрипт на БД «Розрахунки» (тобто із опцією «Використати поточне з’єднання»). Впевнитися, що процедура GET_CLIENT_STATE створилася.
3. В базі даних «Розрахунки» створити новий порожній скрипт з назвою «Лабораторна № 5». Для цього у Експлорері БД по натисненню правої кнопки мишки на папці «Скрипти/Блоки» вибрати пункт меню «Новий скрипт/блок/ВЕБ-форма», і обравши «SQL срипт» натиснути ОК. В полі «Назва скрипта» (Script name) змінити назву «Лабораторна № 5» і зберегти скрипт у базу даних. Збереження в базу даних робиться натисканням в панелі інструментів «Редактора скриптів» піктограми «Дискетка» : “Зберегти SQL скрипт” (“Save SQL script”). Якщо цей же скрипт потрібно зберегти у файл на диск, це робиться черех меню IBExpert “Редактор” → “Зберегти як”. У подальшому всі вдало виконані SQL-запити студент переносить у цей скрипт, і зберігає в БД – для подальшої перевірки викладачем.
4. В редакторі SQL написати запит, який виводить ID та повну назву населеного пункту, ID та особовий рахунок клієнта. Сортування: по назві населеного пункту, і по особовому рахунку. Зберегти запит у скрипті. Результат виконання запиту експортувати у щойно створений каталог в файл типу «XML-таблиця» (“XML SpreadSheet”) з іменем “001_1.Xml”.
5. Переписати цей запит так, щоб виводилися ID та повна назва населеного пункту, і кількість клієнтів із кожного населеного пункту, сортування по

назві населеного пункту. Зберегти запит. Результат виконання запиту експортувати у щойно створений каталог в файл типу «XML-таблиця» (“XML SpreadSheet”) з іменем “002_1.Xml”.

6. Переписати цей запит так, щоб виводилися ID та повна назва населеного пункту, скорочена назва типу клієнта (фізична особа, юридична особа), і кількість клієнтів із кожного населеного пункту, сортування по назві населеного пункту і по назві типу клієнта. Зберегти запит. Результат виконання запиту експортувати в файл з іменем “003_1.Xml”.
7. Переписати цей запит так, щоб виводилися не усі дані, а лише ті населені пункти, де проживає більше одного клієнта. Зберегти запит. Результат виконання запиту експортувати в файл з іменем “004_1.Xml”.
8. Через редактор таблиці CLIENT внести у таблицю два нових клієнти, які проживають у місті Рівне: одну фізичну особу, одну юридичну особу. Підтвердити транзакцію. Виконати всі чотири SQL-запити, впевнитися що вони працюють коректно. Результати виконання запитів експортувати в файли з іменами “001_2.Xml”, “002_2.Xml”, “003_2.Xml”, “004_2.Xml”.
9. В редакторі SQL написати запит, який відображає таблицю OPERSTATE (Стани операцій з коштами на особових рахунках); сортування: по первинному ключу. Зберегти запит. Результат виконання запиту експортувати в файл з іменем “005.Xml”. Вивчити, у яких станах можуть бути операції з коштами, знайти (по опису полів) в таблиці OPERATION відповідне поле.
10. В редакторі SQL написати запит, який відображає таблицю OPERTYPE (Типи операцій з коштами на особових рахунках); сортування: по первинному ключу. Зберегти запит. Результат виконання запиту експортувати в файл з іменем “006.Xml”. Вивчити, які типи операції з коштами застосовуються до особових рахунків клієнтів, знайти (по опису полів) в таблиці OPERATION відповідне поле.
11. Відкрити редактор таблиці OPERATION. Переглянути всі обмеження цілісності даної таблиці. Знайти первинний ключ, скопіювати його назву і назву ключового поля у скрипт (написати коментар «Первинний ключ … таблиці Операції, поле …»). Знайти два вторинні (зовнішні) ключі, що зв’язують цю таблицю з довідниками OPERSTATE і OPERTYPE, скопіювати їх назви у скрипт, дописати відповідні коментарі.
12. В редакторі SQL написати запит, який виводить ID, особовий рахунок та коротку назву клієнта; суму і дату здійснення операції ; повну назву типу операції ; ID та коротку назву стану операції. Сортування: по особовому рахунку ; по даті здійснення операції ; по назві операції ; по назві стану . До полів запиту написати зрозумілі псевдоніми українською. Зберегти запит у скрипті. Результат виконання запиту експортувати у XML-таблицю “007_1.Xml”.
13. Переписати цей запит так, щоб виводилися не усі дані, а лише ті операції, які здійснені у серпні 2018 року. Зберегти запит у скрипті. Результат виконання запиту експортувати у XML-таблицю “007_2.Xml”.
14. Переписати цей запит так, щоб не виводилися видалені операції (умова

«лише ті операції, які здійснені у серпні 2018 року» залишається). Зберегти запит у скрипті. Результат виконання запиту експортувати у XML-таблицю “007_3.Xml”.

15. Переписати цей запит так, щоб виводилися : ID, особовий рахунок та коротка назва клієнта; загальна сума та кількість всіх операцій клієнта у серпні 2018 року (крім видалених) ; мінімальна і максимальна дати здійснення операції. Запит - у скрипт. Результат виконання запиту - у XML-таблицю “007_4.Xml”.
16. Закоментувати умову «лише ті операції, які здійснені у серпні 2018 року». Результат виконання запиту - у XML-таблицю “007_5.Xml”.
17. У редакторі таблиці OPERATION додати до таблиці непорожнє (NOT NULL) текстове поле DESCRIPTION довжиною 200 символів. Додати опис цього поля : «Підстава операції (назва і номер документу)».
18. Написати оператор UPDATE, який заповнить значення поля DESCRIPTION порожньою стрічкою (стрічка без жодного символа `''`) у всіх записах таблиці. Запит - у скрипт.
19. Відкрити процедуру GET_CLIENT_ADDRESS. Вивчити її опис, описи параметрів процедури. В редакторі SQL виконати процедуру двома способами (EXECUTE та SELECT), послідовно задати ID різних клієнтів (1, 2 та 4). Запити - у скрипт.
20. Виконати процедуру GET_CLIENT_ADDRESS в редакторі SQL так, щоб запит повертав не усі вихідні параметри, а лише скорочену адресу клієнта, задавши ID клієнта 3. Запит - у скрипт.
21. Відкрити процедуру GET_CLIENT_STATE в редакторі. Вивчити її опис, описи параметрів процедури. В редакторі SQL виконати процедуру двома способами (EXECUTE та SELECT), задавши ID клієнта 1, 2 та 4 ; а дату – невизначену (NULL). Запити - у скрипт.
22. Скопіювати в редактор SQL останній запит з групуванням по таблиці OPERATION (той, у якому була закоментована умова «лише ті операції, які здійснені у серпні 2018 року»). Додати до цього запиту вкладений запит перед FROM : виклик процедури GET_CLIENT_ADDRESS для визначення скороченої адреси клієнта. Запит - у скрипт.
23. Відкрити Редактор скриптів. Відкрити SQL-скрипт "Лабораторна 5 скрипт 2.Sql". Це SQL-скрипт активування тригерів. Виконати скрипт на БД «Розрахунки» (тобто із опцією «Використати поточне з’єднання»).
24. Зберегти скрипт у базу даних, і для підстраховки – у файл (меню “Редактор” → “Зберегти у файл”) - в папку, створену для роботи № 5.
25. Від’єднатися від БД, зробити холодну копію БД «Розрахунки» у цю ж папку.
26. Закрити всі програми, всі файли у створеній папці заархівувати.
27. Надіслати файл архіву на електронну адресу свого викладача, у листі вказати «Студент «ПІБ повністю”, лаб.робота номер 5».
28. Бази даних «Розрахунки» та «Лабораторні студента …» скопіювати на власну флешку, оскільки вони використовуються в подальших роботах.

Контрольні запитання.

1. Напишіть базовий синтаксис оператора EXECUTE .
2. Поясніть, в яких випадках в операторі SELECT використовується опція GROUP BY.
3. Поясніть, що таке агрегатна функція, наведіть приклади.
4. Поясніть, які поля слід вказувати в операторі SELECT після GROUP BY.
5. Поясніть, в яких випадках в операторі SELECT використовується опція HAVING.
6. Поясніть, які умови відбору записів можна вказувати в операторі SELECT після HAVING.
7. Поясніть різницю у відборі записів після WHERE та після HAVING.
8. Поясніть, що таке збережена процедура, і які операції вона може виконувати.
9. Поясніть, що таке селективна і не селективна збережена процедура.
10. Яким чином можна викликати не селективні процедури.
11. Яким чином можна викликати селективні процедури.
12. Що відбувається із набором вихідних даних селективної процедури, якщо її виконати з допомогою EXECUTE ?

ЛАБОРАТОРНА РОБОТА № 6 : Створення основних об'єктів бази даних.

Тема: Реляційні бази даних.
Мета: Вивчення об'єктів реляційних баз даних : таблиці, перегляди, унікальні індекси, первинні і вторинні ключі.

Теоретичне введення.

Правила іменування об'єктів РБД.

Всі об'єкти в базі даних іменовані, об'єктів без імен не буває. Якщо при створенні об'єкта не вказати його ім'я (синтаксис оператора CREATE деяких об'єктів це допускає), то сервер РСКБД самостійно присвоїть випадкове ім'я із набору літер і цифр. Такого бажано уникати: для полегшення супроводу, адміністрування, та експлуатації РБД потрібно давати всім об'єктам інформативні імена. В назві об'єкта можуть бути латинські літери, цифри, і знак підкреслення, перший символ – завжди літера.

Змінити ім'я об'єкта в БД дуже тяжко, часто неможливо. Тому імена об'єктів слід добре продумувати заздалегідь – до створення.

В базі даних завжди існують зв'язані, або залежні об'єкти. Наприклад, для автоматичної нумерації ключового поля таблиці створюють генератор і тригер. Загальноприйнято всі залежні об'єкти іменувати так, щоб в їх назві в якості основної частини (кореня) була назва головного об'єкта. В даному прикладі

назву таблиці включають як спільний корінь в назви генератора і тригера. Це дозволяє менше звертатися до документації та інтуїтивно розуміти логіку БД.

За звичай розробники БД для утворення імен залежних об'єктів використовують префікси та суфікси. Ім'я генератора (чи як називають генератори в найновіших версіях РБД – "послідовність") утворюють, додаючи до імені поля, яке нумерується з допомогою цього генератора, префіксу GEN_ або SEQ_ (наприклад, генератор GEN_STREET_ID для автонумерації поля NSTREET_ID). Імена тригерів утворюють додаванням суфіксу _TR до назви таблиці, наприклад: тригер з іменем STREET_TR_BI – це тригер до таблиці STREET, який спрацьовує "перед вставкою" ("B(efore) I(nsert)"). Ім'я домена можна утворити додаванням суфіксу _DOM до назви сутності, яку цей домен описує (наприклад, домен PRICE_DOM для опису сутності "Ціна").

Бажано, щоб назви таблиць, полів, та інших об'єктів, іменувалися по-англійськи, а не транслітом. Наприклад, при створенні таблиці "Склад" перевагу надають назвам STORAGE, або STOCKROOM, використання назви SKLAD вважається "дилетантським", особливо в серйозних проектах. Занадто довгі назви для зручності написання інколи скорочують, наприклад, OPER(ation), DESC(ription), STOCK(room).

При створенні таблиць загальноприйнято дотримуватись таких правил :

- ✓ Назва таблиці - це англійський переклад назви основної сутності, яка в ній зберігається, в однині.
- ✓ До назв полів можна спереду додавати одну літеру, яка вказуватиме на тип даних поля: N(umber) для чисельних, D(ate) для дат, T(imestamp) для дати з часом, V(archar) для символьних, і т.п. У подальшому це дозволить менше звертатися до опису схеми БД для з'ясування типів полів;
- ✓ Перше (ключове) поле – завжди цілочисельне, утворюється додаванням до назви таблиці спереду літери N, ззаду суфікса "_ID".
- ✓ Якщо в інших таблицях є посилання (зв'язки) на це ключове поле, то в цих інших таблицях перша частина назви лишається, суфікс "_ID" вилучається.

Приклад :

- ✓ таблицю, в якій зберігаються назви вулиць, назвемо STREET ;
- ✓ ключове поле – ID вулиці – назвемо NSTREET_ID ;
- ✓ у інших таблицях, де використовується ID вулиці, відповідні поля називатимемо NSTREET .

При іменуванні об'єктів не можна використовувати зарезервовані слова (назви типів даних, назви стандартних функцій, тощо). Наприклад, для іменування таблиці «Користувачі БД» замість назви стандартної функції USER слід використати ідентифікатор DbUser, або UserDb, або Users. Для іменування поля «Дата документа» замість назви типу даних DATE слід використати ідентифікатор dDocDate, або dDateDoc, або dDate. Список всіх зарезервованих слів (заборонених ідентифікаторів) можна прочитати в [15] (розділ "KeyWords", стор.184) та в [16] (розділах "New Reserved Words", стор.50).

Дотримання цих правил значно полегшує розуміння і супровід бази даних командою програмістів. У лабораторних і курсових роботах такі вимоги до БД хоча і не обов'язкові, проте дуже бажані, оскільки привчають студента до

загальноприйнятих норм, та полегшують викладачу перевірку роботи студента.

Опис поняття „таблиця”.

Таблиця – основний об’єкт РБД. Поза таблицями в РБД не зберігається нічого, і це є одна із основних умов реляційної моделі даних.

В системних таблицях (які автоматично створюються при створенні “порожньої” БД) зберігається так званий “словник бази даних”: це опис типів даних, об'єктів, обмежень цілісності, процедур, значення параметрів БД, параметри всіх сесій (сеансів) і транзакцій користувачів, права на об’єкти.

В користувацьких таблицях (які створюються власником БД або тими користувачами, яким власник надав таке право) зберігається те, заради чого БД створювалася: власне інформація, яку вносять користувачі БД.

Приклади створення двох таблиць :

```
CREATE TABLE OPERTYPE (
       NOPERTYPE_ID SMALLINT NOT NULL,
       VNAME        VARCHAR(32) NOT NULL,
       VNAMESHORT   VARCHAR(12) NOT NULL,
       NSIGN        SMALLINT NOT NULL CHECK (NSIGN IN (-
      1,0,1) ) );
CREATE TABLE OPERATION (
       NOPER_ID    INTEGER NOT NULL,
       NCLIENT     INTEGER NOT NULL,
       NOPERTYPE   SMALLINT NOT NULL,
       NSUMMA      NUMERIC(15,2) NOT NULL,
       DDATE       DATE NOT NULL,
       NOPERSTATE  SMALLINT NOT NULL,
       TEDIT       TIMESTAMP NOT NULL,
       NUSER       SMALLINT NOT NULL,
       DESCRIPTION  VARCHAR(200) NOT NULL );
```

При створенні таблиць бажано, щоб всі поля мали обмеження цілісності «не порожні значення» (NOT NULL) – крім тих випадків, коли бізнес-логіка не дозволяє уникнути значення NULL у даному полі. Наприклад, у стрічкових полях уникнути значення NULL просто: з допомогою порожньої стрічки: ’’. У полях типу «Дата» це зробити складно (наприклад, у полі «Дата звільнення з роботи» таблиці «Працівники» доведеться дозволити порожні значення NULL).

Ключі, індекси.

Приклади створення первинних ключів (PRIMARY KEY) з назвами PK_OPERTYPE та PK_OPERATION до таблиць OPERTYPE і OPERATION :

```
ALTER TABLE OPERTYPE ADD CONSTRAINT PK_OPERTYPE
   PRIMARY KEY (NOPERTYPE_ID);
ALTER TABLE OPERATION ADD CONSTRAINT PK_OPERATION
   PRIMARY KEY (NOPER_ID);
```

Первинний ключ гарантує унікальність значень ключового поля, і тому значення цього поля є ніби "ключем" для пошуку потрібного запису (рядка).

Приклад створення вторинного (зовнішнього) ключа (FOREIGN KEY) з назвою FK_OPERATION_TYPE для поля NOPERTYPE таблиці OPERATION :

```
ALTER TABLE OPERATION ADD CONSTRAINT FK_OPERATION_TYPE
   FOREIGN KEY (NOPERTYPE)
   REFERENCES OPERTYPE (NOPERTYPE_ID);
```

Вторинний (зовнішній) ключ реалізує зв'язок "один до багатьох" і гарантує, що в тому полі таблиці, для якого він створений, будуть присутні лише ті значення, які є у тому полі зовнішньої таблиці, на яку є посилання. У наведеному прикладі: в поле NOPERTYPE таблиці OPERATION, для якого створений вторинний (зовнішній) ключ з назвою FK_OPERATION_TYPE, можливо внести лише ті значення, які є в полі NOPERTYPE_ID зовнішньої таблиці OPERTYPE.

Створення та редагування об'єктів у програмі IBExpert.

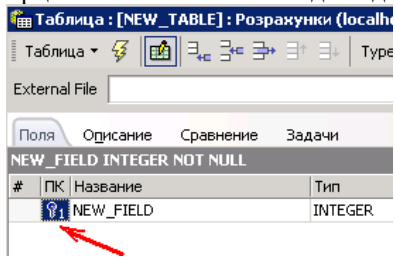
В програмі IBExpert можна створювати і змінювати об'єкти в редакторі SQL, написавши запит "CREATE …", "ALTER …".

Проте значно зрічніше це робити в редакторі об'єктів. Це дозволяє одразу заповнити опис об'єкта, описи основних елементів об'єкта (наприклад, полів таблиць), створювати обмеження цілісності, швидко і зручно реалізувати автонумерування поля (через генератор і тригер), і т.п.

Створення об'єкта слід починати з Експлорера бази даних (DataBase Explorer) – помітити необхідну папку (наприклад, «Таблиці»), і через контекстне меню (по натисненню правої клавіші мишки) перейти до створення потрібного об'єкта (наприклад, вибрати «Нова таблиця»). Автоматично відкриється редактор обраного типу об'єкта (наприклад, редактор таблиць), і IBExpert запропонує ім'я цього об'єкту (наприклад, NEW_TABLE).

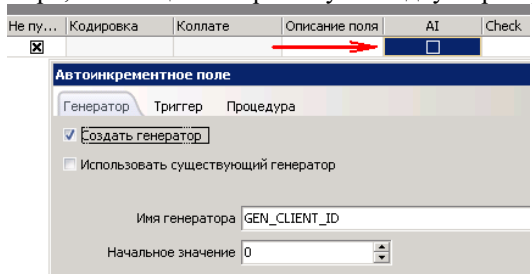
Першим кроком (після відкриття редактора об'єкту) завжди потрібно змінити запропоноване програмою ім'я об'єкта на своє, заздалегідь придумане (наприклад, витерти назву NEW_TABLE і замість неї ввести інформативну назву). Якщо цього не зробити, то всі зв'язані об'єкти (наприклад, для таблиці це - генератори і тригери - будуть у своїй назві містити корінь NEW_TABLE, що призведе до плутанини і до необхідності змінювати ці назви у подальшому).

Первинний ключ створюється в програмі IBExpert при створенні таблиці подвійним кліком в комірці «ПК» властивостей відповідного поля :

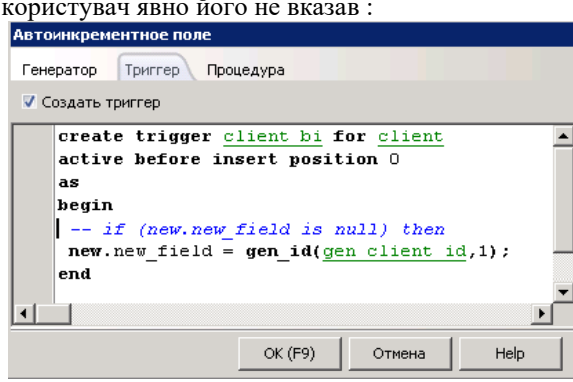


Автонумерація значень поля найкраще реалізується з допомогою

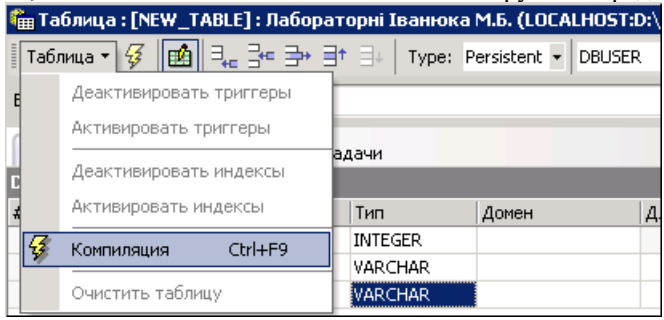
генератора та тригера. Подвійний клік в комірці «AI» (автоінкремент) властивостей поля викликає діалог, у якому необхідно відмітити опцію «Створити генератор», і після цього перейти у закладку «Тригер»:



У закладці «Тригер» спочатку відмічається опція «Створити тригер», потім оператор IF … заключається в символ коментаря, як показано на малюнку нижче – для того, щоб нове значення поля генерувалося щоразу, а не лише у випадку, коли користувач явно його не вказав :



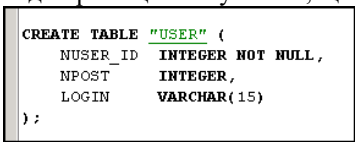
Збереження об’єкта в базу даних здійснюється через меню редактора «Компіляція», або кнопкою «Блискавка» на панелі інструментів редактора :



При створенні/модифікації кожного об’єкту в обов’язковому порядку заповнюється/редагується **опис об’єкта, та описи усіх його елементів** (наприклад, полів таблиць, всіх параметрів процедур, тощо).

Зміна у програмі IBExpert назви таблиці і назв залежних від неї об'єктів.

Одразу після створення (тобто після **першого** збереження в БД) таблиці в програмі IBExpert слід уважно переглянути її SQL-скрипт (в закладці «Скрипт» редактора таблиці). Якщо ім'я новоствореної таблиці взято у лапки – значить воно співпало із зарезервованим словом (див. малюнок), і при написанні SQL-запитів потрібно буде завжди брати це ім'я у лапки, що дуже незручно.



```
CREATE TABLE "USER" (
    NUSER_ID  INTEGER NOT NULL,
    NPOST     INTEGER,
    LOGIN     VARCHAR(15)
);
```

В цьому випадку потрібно створити заново ідентичну таблицю з іншим іменем, а після цього непотрібну таблицю – видалити разом із залежними від неї об'єктами. Заново створити ідентичну таблицю можна вручну. Проте швидше і зручніше скористатися існуючим скриптом невірно названої таблиці, а саме :

1) Виділити весь SQL-скрипт існуючої таблиці (CTRL+A), скопіювати його в буфер обміну Windows (CTRL+C), відкрити новий редактор скриптів, і вставити в нього цілий скрипт існуючої таблиці (CTRL+V);
2) Замінити в тексті скрипта ім'я таблиці там, де воно зустрічається цілим словом, тобто : пошуком і заміною в редакторі скриптів (CTRL+R) замінити взяте в лапки небажане ім'я таблиці на нове бажане (в даному прикладі заміна була б така : `"USER"` → `DBUSER`);
3) Замінити в тексті скрипта ім'я таблиці там, де воно зустрічається як корінь у назві інших об'єктів, тобто : пошуком і заміною в редакторі скриптів (CTRL+R) замінити небажане ім'я таблиці без лапок на нове бажане (в даному прикладі заміна була б така : `USER` → `DBUSER`);
4) Перевірити, що замінено всі входження старого імені таблиці (наприклад ім'я ключового поля NUSER_ID повинно змінитися на NDBUSER_ID); і перевірити, що у скрипті не залишилося імен об'єктів у лапках.
5) Виконати утворений SQL-скрипт.

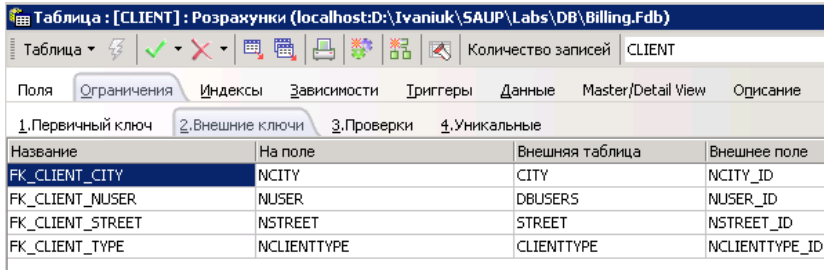
Якщо скрипт виконався без помилок – можна знищити непотрібні об'єкти у такій послідовності: тригер на таблицю, генератор, таблиця.

За такою технологією: копіювання (і, можливо, подальше редагування) SQL-скрипта об'єкта, - можна переіменовувати і/або переносити у іншу БД не лише таблиці, але й інші типи об'єктів РБД.

Обмеження цілісності таблиці.

Деякі обмеження цілісності таблиці можна додати до неї як у процесі створення таблиці, так і потім – під час редагування таблиці: це первинні ключі, перевірки, обмеження NOT NULL (див. вище), домени.

Інші обмеження цілісності таблиці можна додати до неї тільки після вдалого створення - в закладці «Обмеження», як показано на малюнку :



До таких обмежень належать вторинні ключі, унікальні індекси.

Більш детально обмеження цілісності розглядатимуться у наступній роботі.

Внесення даних в таблиці у програмі IBExpert.

Дані в таблицю можна вносити і редагувати дані, написавши запит “INSERT …”, “UPDATE …”. Проте в програмі IBExpert значно зручніше і швидше це робити в редакторі таблиці, у закладці «Дані».

При заповненні таблиць (внесенні даних) слід вказувати правдоподобні, логічні дані, без граматичних помилок. За неграмотні і нечитабельні дані знижуються бали.

Завдання на лабораторну роботу.

1. В каталозі студента створити нову папку для даної роботи (наприклад “D:\Ivaniuk\FB\Labs\Лаб6\”) (в нього будуть зберігатися деякі файли, про які буде сказано в завданні).
2. В програмі IBExpert з’єднатися із базою даних «Розрахунки». Відкрити Редактор скриптів. Відкрити SQL-скрипт "Лабораторна 6 скрипт 1.Sql". Це SQL-скрипт створення процедур, які реалізують переведення коштів з одного особового рахунку на інший. Виконати скрипт на БД «Розрахунки» (тобто із опцією «Використати поточне з’єднання»). Впевнитися, що процедури CALC_TRANSFER_TAX та TRANSFER_MONEY_FROM_A_TO_B створилася.
3. Оскільки скрипт модифікує поле DESCRIPTION таблиці OPERATION, це потребує негайного Backup/Restore бази даних «Розрахунки». Потрібно створити в папці ЛР ще одну окрему папку для Backup/Restore; зробити Backup/Restore, і при цьому не забути, що власником БД є сам студент – потрібно використовувати лише логін/пароль власника. Два протоколи виконання Backup та Restore зберегти у окремі текстові файли, прослідкувати, щоб обидва протоколи збереглися повні. Після виконання Backup/Restore виконати перевірку БД на помилки, і перевірити модифіковану таблицю : select * from OPERATION, від’єднатися від БД «Розрахунки».
4. Повторити створення бази даних «Лабораторні студента ...» - див. лист Сімашко В.Й. від 4 жовтня 2017 р. (прочитати лист, прочитати

«Лабораторна робота № 3, Теоретична частина, розділ "**Створення РБД.**"», повторити лише пункти «Лабораторна робота № 3, Завдання на роботу, пункти **8, 9, 10, 11, 12 та 24**.», файл JPG зберегти в каталог для лабораторної). При виконанні пункту 12 слід використати нову версію SQL-скрипта "Лабораторна 3 скрипт 1.Sql" – та, яка у новій версії архіву "**FB_Labs_Files.Zip**", див. додаток 1.

5. З'єднатися (власником) із базою «Лабораторні студента ...». У базі даних створити новий порожній скрипт з назвою «Лабораторна № 6» (в нього будуть зберігатися деякі запити, про які буде сказано в завданні).
6. Створити таблицю «Посади в магазині» з полями:
   - ✓ ID посади: цілочисельне (Integer), PK (Primary Key, Первинний ключ), AI (Auto Increment, автонумерація значень з допомогою генератора і тригера);
   - ✓ Назва посади повна: стрічкове 50 символів, унікальне.
   - ✓ Назва посади скорочена: стрічкове 10 символів, унікальне.
7. Внести в таблицю «Посади в магазині» наступні посади:
   - ✓ Директор;
   - ✓ Системний адміністратор - програміст.
   - ✓ Бухгалтер.
   - ✓ Менеджер - комірник.
   - ✓ Старший продавець.
   - ✓ Продавець.
   - ✓ Прибиральник.

   При заповненні таблиць (внесенні даних) у всіх таблицях слід вказувати правдоподобні, логічні дані, без граматичних помилок.
8. Створити таблицю «Працівники магазину» з полями:
   - ✓ ID працівника: цілочисельне, PK, AI;
   - ✓ ПІБ працівника повністю: стрічкове 70 символів.
   - ✓ ПІБ працівника скорочено: стрічкове 30 символів.
   - ✓ ID посади: FK (вторинний ключ) на таблицю «Посади в магазині».
   - ✓ Адреса працівника: стрічкове 70 символів (в промислових базах таке ведення адрес неприпустиме, адреси реалізуються тільки через довідники, див. БД «Розрахунки»).
   - ✓ Контактний телефон працівника.
   - ✓ Дата влаштування на роботу.
   - ✓ Дата звільнення з роботи: допускаються порожні (NULL) значення.
9. Внести в таблицю «Працівники магазину» стільки працівників, скільки на думку студента достатньо для нормальної роботи магазину із трьох відділів. На кожній посаді повинно бути хоча б по одному працівнику. В якості системного адміністратора – програміста студент вказує сам себе.
10. Створити таблицю «Користувачі бази даних» з полями:
    - ✓ ID користувача: цілочисельне, PK, AI;
    - ✓ Ім'я (логін) користувача в БД: стрічкове 20 символів, унікальне.
    - ✓ ID працівника : FK (вторинний ключ) на таблицю «Працівники магазину».

- ✓ Цілочисельне поле «Користувач заблокований»: допускаються лише два можливих значення: 0 – дозвіл для роботи в БД відкрито; 1 – користувач заблокований (значення за замовчуванням).

11. Внести в таблицю «Користувачі бази даних» лише тих працівників, які на думку студента повинні працювати із базою даних згідно своїх посадових обов'язків. Логін системного адміністратора – програміста повинен співпадати із логіном власника БД «Лабораторні студента ...». Інші логіни повинні бути внесені за правилами: ініціали + прізвище (або ініціал імені і прізвище) в латинській транскрипції.
12. З'єднатися із базою користувачів сервера РСКБД, і внести всіх користувачів, яких щойно внесли в таблицю «Користувачі бази даних». Пароль у всіх «1». Від'єднатися від менеджера користувачів.
13. З'єднатися (**власником !!!**) із базою «Лабораторні студента ...». На базі таблиць БД створити перегляд (VIEW) «Працівники магазину» з такими полями:
    - ✓ ID працівника;
    - ✓ ПІБ працівника повністю.
    - ✓ ПІБ працівника скорочено.
    - ✓ Назва посади працівника повністю.
    - ✓ Назва посади працівника скорочено.
    - ✓ Адреса працівника.
    - ✓ Контактний телефон працівника.
    - ✓ Дата влаштування на роботу.
    - ✓ Дата звільнення з роботи.
14. Вибрати всі дані із цього перегляду (сортовані по ID працівника), результат експортувати в XML-таблицю, зберегти в папку лабораторної роботи у файл "Працівники магазину.XML". Два останні SQL-запити: 1) створення перегляду «Працівники магазину» та 2) вибір даних з цього перегляду – зберегти в БД у скрипті «Лабораторна № 6».
15. На базі перегляду «Працівники магазину» і таблиці «Користувачі бази даних» створити перегляд (VIEW) «Користувачі бази даних» з такими полями:
    - ✓ ID користувача БД;
    - ✓ Ім'я (логін) користувача в БД.
    - ✓ Цілочисельне поле «Користувач заблокований».
    - ✓ ID працівника;
    - ✓ ПІБ працівника повністю.
    - ✓ ПІБ працівника скорочено.
    - ✓ Назва посади працівника повністю.
    - ✓ Назва посади працівника скорочено.
    - ✓ Адреса працівника.
    - ✓ Контактний телефон працівника.
    - ✓ Дата влаштування працівника на роботу.
    - ✓ Дата звільнення працівника з роботи.
16. Вибрати всі дані із цього перегляду (сортовані по ID користувача),

результат експортувати в XML-таблицю, зберегти в папку лабораторної роботи у файл “Користувачі бази даних.XML”. Два останні SQL-запити: 1) створення перегляду «Користувачі бази даних» та 2) вибір даних з цього перегляду – зберегти в БД у скрипті «Лабораторна № 6».

17. Створити таблицю «Одиниці виміру» з полями:
    - ✓ ID одиниці виміру: Цілочисельне (Integer), PK (Primary Key, Первинний ключ), AI (Auto Increment, автонумерація значень з допомогою генератора і тригера);
    - ✓ Назва одиниці виміру повна: стрічкове 20 символів, унікальне.
    - ✓ Назва одиниці виміру скорочена: стрічкове 5 символів, унікальне.
18. Внести в таблицю «Одиниці виміру» : кілограми; літри; штуки; метри.
19. Створити таблицю «Відділи магазину» з полями:
    - ✓ ID відділу: Цілочисельне (Integer), PK , AI ;
    - ✓ Назва відділу повна: стрічкове 30 символів, унікальне.
    - ✓ Назва відділу скорочена: стрічкове 12 символів, унікальне.
20. Внести в таблицю «Відділи магазину» мінімум три відділи (можна більше за бажанням).
21. Створити таблицю «Товари» з полями:
    - ✓ ID товару: Цілочисельне (Integer), PK (Primary Key, Первинний ключ), AI (Auto Increment, автонумерація значень з допомогою генератора і тригера);
    - ✓ Назва товару повна: стрічкове 50 символів, унікальне.
    - ✓ Назва товару скорочена: стрічкове 30 символів, унікальне.
    - ✓ ID одиниці виміру товару: FK на таблицю «Одиниці виміру».
    - ✓ ID відділу, у якому продається цей товар: FK на таблицю «Відділи магазину».
22. Внести в таблицю «Товари» мінімум по чотири товари у кожен відділ (можна більше за бажанням). При цьому особливу увагу звернути на назви товарів. Якщо товар ваговий/наливний/довжинний (вимірюється на кілограми/літри/метри), то в назві бажано щоб був виробник (наприклад «Шоколад Світоч чорний ваговий», «Олія Радема розливна», «Дріт ПХВ-0.5х3 Одесакабель»). Якщо товар штучний (вимірюється на штуки), то в назві повинен бути і виробник і кількість товару в одній штуці (наприклад «Шоколад Світоч молочний 90 г.», «Олія Радема ПЕТ 1 л.»).
23. На базі таблиць БД створити перегляд (VIEW) «Номенклатура товарів магазину» з такими полями:
    - ✓ ID товару;
    - ✓ Назва товару повна.
    - ✓ Назва товару скорочена.
    - ✓ Назва одиниці виміру повна.
    - ✓ Назва одиниці виміру скорочена.
    - ✓ Назва товару повна і через пробіл - назва одиниці виміру повна.
    - ✓ Назва товару скорочена і через пробіл - назва одиниці виміру скорочена.
    - ✓ Назва відділу повна.

- ✓ Назва відділу скорочена.

24. Вибрати всі дані із цього перегляду (сортовані по ID товару), результат експортувати в XML-таблицю, зберегти в папку лабораторної роботи у файл “Номенклатура товарів магазину.XML”. Два останні SQL-запити: 1) створення перегляду «Номенклатура товарів магазину» та 2) вибір даних з цього перегляду – зберегти в БД у скрипті «Лабораторна № 6».
25. Створити таблицю «Статус документу» з полями:
    - ✓ ID статусу: Цілочисельне (Integer), PK , AI ;
    - ✓ Назва статусу повна: стрічкове 30 символів, унікальне.
    - ✓ Назва статусу скорочена: стрічкове 12 символів, унікальне.
26. Внести в таблицю «Статус документу» три записи :
    1) 1 = Оформлення = Оформл.
    2) 2 = Виконано = Вик.
    3) 3 = Анульовано = Анул.
27. Створити таблицю «Постачальники» з полями:
    - ✓ ID постачальника: цілочисельне, PK, AI;
    - ✓ Назва постачальника повна: стрічкове 70 символів, унікальне.
    - ✓ Назва постачальника скорочена: стрічкове 30 символів, унікальне.
    - ✓ Адреса постачальника : стрічкове 70 символів (в промислових базах таке ведення адрес неприпустиме, адреси реалізуються тільки через довідники, див. БД «Розрахунки»).
    - ✓ Контактний телефон постачальника.
    - ✓ Номер договору із постачальником: стрічкове 30 символів, унікальне.
    - ✓ Дата укладання договору із постачальником.
    - ✓ Дата розірвання договору із постачальником: допускаються порожні (NULL) значення.
28. Внести в таблицю «Постачальники» мінімум трьох постачальників (можна більше за бажанням).
29. Створити таблицю «Приходні накладні» з полями:
    - ✓ ID приходної накладної: цілочисельне, PK , AI ;
    - ✓ Номер приходної накладної: стрічкове 12 символів.
    - ✓ Дата приходної накладної: дата.
    - ✓ Загальна сума по приходній накладній: чисельне (15,2).
    - ✓ ID статусу накладної : FK на таблицю «Статус документу», значення за замовчуванням 1.
    - ✓ ID працівника магазину, який прийняв товари по накладній: FK на таблицю «Працівники магазину».
    - ✓ ID постачальника : FK на таблицю «Постачальники».
    - ✓ Дата+Час внесення/останнього редагування накладної.
    - ✓ ID користувача БД, який вніс/останнім редагував накладну: FK на таблицю «Користувачі БД».
30. Створити таблицю «Надходження товарів у магазин» з полями:
    - ✓ ID надходження : цілочисельне PK AI ;
    - ✓ ID приходної накладної: FK на таблицю «Приходні накладні».
    - ✓ ID товару : FK на таблицю «Товари».

- ✓ Кількість товару, що надійшов, у одиницях виміру : чисельне (15,3).
- ✓ Собівартість одиниці виміру товару, що надійшов : чисельне (15,2).
- ✓ Номер партії від виробника товару: стрічкове 12 символів.
- ✓ Дата виготовлення даної партії товару виробником.
- ✓ Кінцева дата терміну придатності від виробника даної партії товару.

31. Внести в таблиці «Приходні накладні» та «Надходження товарів у магазин» мінімум три приходних накладних, кожна накладна зі статусом «Виконано» (можна більше за бажанням), і в кожній приходній накладній повинно бути мінімум по чотири позиції різних товарів (можна більше за бажанням). Прослідкувати, щоб товари потрапили у кожен з відділів.
32. Внести в таблиці «Приходні накладні» та «Надходження товарів у магазин» ще одну накладну зі статусом «Анульовано» (можна більше за бажанням), і в цій приходній накладній повинно бути по одному товару в кожен із відділів магазину (можна більше за бажанням).
33. Внести в таблиці «Приходні накладні» та «Надходження товарів у магазин» ще одну накладну зі статусом «Оформлення» (можна більше за бажанням), і в цій приходній накладній повинно бути по одному товару в кожен із відділів магазину (можна більше за бажанням).
34. На базі таблиць БД створити перегляд (VIEW) «Технологічний перегляд: всі дані із таблиці надходження товарів у магазин незалежно від статусу» з такими полями:
    - ✓ ID надходження;
    - ✓ Номер приходної накладної.
    - ✓ Дата приходної накладної.
    - ✓ ID статусу приходної накладної.
    - ✓ Назва статусу приходної накладної повна.
    - ✓ Назва статусу приходної накладної скорочена.
    - ✓ ПІБ працівника магазину, який прийняв товар по накладній, повне.
    - ✓ ПІБ працівника магазину, який прийняв товар по накладній, скорочне.
    - ✓ Назва постачальника повна.
    - ✓ Назва постачальника скорочена.
    - ✓ Назва товару повна;
    - ✓ Назва товару скорочена.
    - ✓ Кількість товару у одиницях виміру.
    - ✓ Назва одиниці виміру товару повна.
    - ✓ Назва одиниці виміру товару скорочена.
    - ✓ Собівартість одиниці виміру товару.
    - ✓ Загальна собівартість даної позиції надходження товару.
    - ✓ Номер партії товару.
    - ✓ Дата виготовлення даної партії товару.
    - ✓ Кінцева дата терміну придатності даної партії товару.
    - ✓ Дата+Час внесення/останнього редагування накладної.
    - ✓ ПІБ скорочено користувача БД, який вніс/останнім редагував накладну.
35. На базі технологічного перегляду «Технологічний перегляд: всі дані із таблиці надходження товарів у магазин незалежно від статусу» створити

ще один перегляд (VIEW) «Виконані надходження товарів у магазин» з такими ж полями, як у технологічному перегляді, але накласти додаткову умову відбору: відібрати лише накладні зі статусом «Виконано».

36. Вибрати всі дані із цього перегляду (сортовані по ID надходження), результат експортувати в XML-таблицю, зберегти в папку лабораторної роботи у файл “Виконані надходження товарів у магазин.XML”. Три останні SQL-запити: 1) створення технологічного перегляду «Всі надходження» , 2) створення перегляду «Виконані надходження» та 3) вибір даних з перегляду «Виконані надходження» – зберегти в БД у скрипті «Лабораторна № 6».
37. На базі технологічного перегляду «Технологічний перегляд: всі дані із таблиці надходження товарів у магазин незалежно від статусу» створити ще один перегляд (VIEW) «Надходження товарів у магазин, які оформляються» з такими ж полями, як у технологічному перегляді, але накласти додаткову умову відбору: відібрати лише накладні зі статусом «Оформлення».
38. Вибрати всі дані із цього перегляду (сортовані по ID надходження), результат експортувати в XML-таблицю, зберегти в папку лабораторної роботи у файл “Надходження які оформляються.XML”. Два останні SQL-запити: 1) створення перегляду «Надходження які оформляються» та 2) вибір даних з перегляду «Надходження які оформляються» – зберегти в БД у скрипті «Лабораторна № 6».
39. На базі технологічного перегляду «Технологічний перегляд: всі дані із таблиці надходження товарів у магазин незалежно від статусу» створити ще один перегляд (VIEW) «Анульовані надходження товарів у магазин» з такими ж полями, як у технологічному перегляді, але накласти додаткову умову відбору: відібрати лише накладні зі статусом «Анульовано».
40. Вибрати всі дані із цього перегляду (сортовані по ID надходження), результат експортувати в XML-таблицю, зберегти в папку лабораторної роботи у файл “Анульовані надходження.XML”. Два останні SQL-запити: 1) створення перегляду «Анульовані надходження» та 2) вибір даних з перегляду «Анульовані надходження» – зберегти в БД у скрипті «Лабораторна № 6».
41. Створити таблицю «Журнал обліку реалізації товарів» з полями :
    - ✓ ID облікового запису в журналі обліку: цілочисельне, PK , AI ;
    - ✓ Номер облікового запису: стрічкове 12 символів.
    - ✓ Дата початку облікового запису: дата початку періоду, за який вносяться в БД реалізації товарів.
    - ✓ Дата закінчення облікового запису: дата закінчення періоду, за який вносяться в БД реалізації товарів.
    - ✓ Загальна сума по всіх позиціях облікового запису: чисельне (15,2).
    - ✓ ID статусу облікового запису: FK на таблицю «Статус документу», значення за замовчуванням 1.
    - ✓ ID працівника магазину, який реалізував товари: FK на таблицю «Працівники магазину».

- ✓ Дата+Час внесення/останнього редагування облікового запису.
- ✓ ID користувача БД, який вніс/останнім редагував обліковий запис: FK на таблицю «Користувачі БД».

42. Створити таблицю «Позиції реалізованих товарів» з полями:
    - ✓ ID позиції реалізації: цілочисельне PK AI ;
    - ✓ ID облікового запису в журналі обліку: FK на таблицю «Журнал обліку реалізації товарів».
    - ✓ ID товару : FK на таблицю «Товари».
    - ✓ Кількість реалізованого (проданого) товару у одиницях виміру : чисельне (15,3).
    - ✓ Ціна реалізації (продажі) одиниці виміру товару: чисельне (15,2).
43. Внести в таблиці «Журнал обліку реалізації товарів» та «Позиції реалізованих товарів» мінімум три записи у журналі обліку реалізації товарів, кожен запис зі статусом «Виконано» (можна більше за бажанням), і в кожному запису в журналі повинно бути мінімум по чотири позиції різних товарів (можна більше за бажанням). Прослідкувати, щоб реалізовані товари потрапили у кожен з відділів.
44. Внести в таблиці «Журнал обліку реалізації товарів» та «Позиції реалізованих товарів» ще один запис у журналі обліку реалізації товарів зі статусом «Анульовано» (можна більше), в цьому записі в журналі повинно бути по одній позиції товару в кожен відділ (можна більше за бажанням).
45. Внести в таблиці «Журнал обліку реалізації товарів» та «Позиції реалізованих товарів» ще один запис у журналі обліку реалізації товарів зі статусом «Оформлення» (можна більше), в цьому записі в журналі повинно бути по одній позиції товару в кожен відділ (можна більше).
46. На базі таблиць БД створити перегляд (VIEW) «Технологічний перегляд: всі дані із таблиці позиції реалізованих товарів незалежно від статусу» з такими полями:
    - ✓ ID позиції реалізації;
    - ✓ Номер облікового запису в журналі реалізації.
    - ✓ Дата початку облікового запису в журналі реалізації.
    - ✓ Дата закінчення облікового запису в журналі реалізації.
    - ✓ ID статусу облікового запису в журналі.
    - ✓ Назва статусу облікового запису в журналі повна.
    - ✓ Назва статусу облікового запису в журналі скорочена.
    - ✓ ПІБ працівника магазину, який реалізував товар, повне.
    - ✓ ПІБ працівника магазину, який реалізував товар, скорочене.
    - ✓ Кількість проданого товару у одиницях виміру.
    - ✓ Назва одиниці виміру товару повна.
    - ✓ Назва одиниці виміру товару скорочена.
    - ✓ Ціна реалізації (продажі) одиниці виміру товару.
    - ✓ Загальна вартість реалізації (продажі) даної позиції товару.
    - ✓ Дата+Час внесення/останнього редагування облікового запису.
    - ✓ ПІБ скорочено користувача БД, який вніс/останнім редагував обліковий запис.

47. На базі технологічного перегляду «Технологічний перегляд: всі дані із таблиці позиції реалізованих товарів незалежно від статусу» створити ще один перегляд (VIEW) «Виконані позиції реалізованих товарів» з такими ж полями, як у технологічному перегляді, але накласти додаткову умову відбору: відібрати лише записи в журналі реалізації зі статусом «Виконано».
48. Вибрати всі дані із цього перегляду (сортовані по ID позиції реалізації), результат експортувати в XML-таблицю, зберегти в папку лабораторної роботи у файл “Виконані позиції реалізованих товарів.XML”. Три останні SQL-запити: 1) створення технологічного перегляду «Всі реалізації» , 2) створення перегляду «Виконані реалізації» та 3) вибір даних з перегляду «Виконані реалізації» – зберегти в БД у скрипті «Лабораторна № 6».
49. На базі технологічного перегляду «Технологічний перегляд: всі дані із таблиці позиції реалізованих товарів незалежно від статусу» створити ще один перегляд (VIEW) «Реалізації товарів які оформляються» з такими ж полями, як у технологічному перегляді, але накласти додаткову умову відбору: відібрати лише записи в журналі реалізації зі статусом «Оформлення».
50. Вибрати всі дані із цього перегляду (сортовані по ID надходження), результат експортувати в XML-таблицю, зберегти в папку лабораторної роботи у файл “Реалізації які оформляються.XML”. Два останні SQL-запити: 1) створення перегляду «Реалізації які оформляються» та 2) вибір даних з перегляду «Реалізації які оформляються» – зберегти в БД у скрипті «Лабораторна № 6».
51. На базі технологічного перегляду «Технологічний перегляд: всі дані із таблиці позиції реалізованих товарів незалежно від статусу» створити ще один перегляд (VIEW) «Анульовані реалізації товарів» з такими ж полями, як у технологічному перегляді, але накласти додаткову умову відбору: відібрати лише записи в журналі реалізації зі статусом «Анульовано».
52. Вибрати всі дані із цього перегляду (сортовані по ID надходження), результат експортувати в XML-таблицю, зберегти в папку лабораторної роботи у файл “Анульовані реалізації.XML”. Два останні SQL-запити: 1) створення перегляду «Анульовані реалізації» та 2) вибір даних з перегляду «Анульовані реалізації» – зберегти в БД у скрипті «Лабораторна № 6».
53. Від’єднатися від усіх БД, зробити холодні копії БД «Розрахунки» та «Лабораторні студента ...» у створену для даної лабораторної роботи папку (**не забувайте, що в копію БД входить і файл паролів !**!!).
54. Закрити всі програми, всі файли у створеній папці заархівувати.
55. Надіслати файл архіву на електронну адресу свого викладача, у листі вказати «Студент «ПІБ повністю”, лаб.робота номер 6».
56. Бази даних «Розрахунки» та «Лабораторні студента …» скопіювати на власну флешку, оскільки вони використовуються в подальших роботах.

Контрольні запитання.

1. Чи можливі SQL-оператори CREATE… створення об'єкта РБД, у яких не вказується назва створюваного об'єкта ? Чи існують в РБД об'єкти без імен ?
2. Чому для іменування усіх об'єктів потрібно дотримуватися однотипних домовленостей ?
3. Чому об'єкти в РБД потрібно називати інформативними іменами, які б відображали сутність або призначення цього об'єкта ? Наведіть приклад.
4. Чому зв'язані об'єкти в РБД бажано іменувати таким чином, щоб у їх назвах був спільний корінь ? Наведіть приклад.
5. Напишіть базовий синтаксис оператора CREATE TABLE та приклад створення довільної таблиці.
6. Чому при створенні таблиць слід уникати полів, у яких допускаються невизначені (порожні) значення NULL ?
7. Поясніть, в яких випадках в таблицях використовуються поля, у яких допускаються невизначені (порожні) значення NULL ?.
8. Поясніть, що таке первинний ключ таблиці, поясніть на прикладі.
9. Чому на Вашу думку в назві «первинний ключ таблиці» використане слово «ключ» ? Поясніть на прикладі.
10. Поясніть, що таке вторинний (зовнішній) ключ таблиці, поясніть на прикладі.

ЛАБОРАТОРНА РОБОТА № 7 : Створення об'єктів бази даних (початок).

Тема: Реляційні бази даних.

Мета: Вивчення об'єктів реляційних баз даних : домени, перевірки, прості індекси, процедури і тригери.

Теоретичне введення.

Опис поняття „домен".

Якщо в базі даних часто використовується однакова сутність (тобто в кількох таблицях, процедурах, ...), цю сутність можна описати в так званому домені, а в таблицях, процедурах, і тощо використовувати цю назву домена. Спрощений синтаксис створення домена такий :

CREATE DOMAIN {ім'я домену} AS {тип даних} [додаткові обмеження цілісності (допустимі значення домену)] ;

Наприклад, якщо в базі даних в кількох таблицях будуть зберігатися прізвища людей, можна створити домен з назвою VSURNAME_DOM :

```
CREATE DOMAIN VSURNAME_DOM AS
  VARCHAR(50) CHARACTER SET WIN1251 NOT NULL
  CHECK (VALUE SIMILAR TO '[А-Я]([А-Я]|[а-я]|-|")*')
  COLLATE WIN1251_UA;
COMMENT ON DOMAIN VSURNAME_DOM IS
```

```
'або Прізвище, або Ім`я, або По-батькові';
```
а потім скрізь використовувати назву домена VSURNAME_DOM.

Приклад 1 : в даному прикладі використано такі обмеження на тип і значення домену:

- ✓ `VARCHAR(50)` – символьний тип даних не більше 50-ти символів;
- ✓ `CHARACTER SET WIN1251` – набір символів WIN1251 (латиниця + кирилиця);
- ✓ `NOT NULL` – значення не може бути невизначене;
- ✓ `CHECK (VALUE …)` – обмеження на значення домену (ключове слово VALUE (дослівно "значення") буде замінене на назву того поля/змінної, до якої буде застосовано домен VSURNAME_DOM);
- ✓ `VALUE SIMILAR TO 'маска'` – визначає якій масці повинне відповідати символьне значення (дослівно "значення подібне до …");
- ✓ … `'[А-Я]([А-Я]|[а-я]|-|″)*'` – маска, що визначає яким може бути значення : першим символом обов'язково одна велика літера від А до Я (**`'[А-Я]`**), потім те що в круглих дужках може повторюватися будь-яку кількість разів (**`(…)*'`**), те що в круглих дужках, це може бути одна велика літера від А до Я (**`([А-Я]`**), або одна мала літера від а до я (**`[а-я]`**), або символ "дефіс" (**`-`**), або символ "подвійні лапки" (**`″`**). Символ "вертикальна риска" (**`|`**) означає "або";
- ✓ `COLLATE WIN1251_UA` – сортування за українським алфавітом;
- ✓ `COMMENT ON DOMAIN` – опис домена, теж може зберігатися в базі даних для полегшення її розуміння і супроводу.

Такі стрічки символів, як :

- ✓ `'Іванюк'`
- ✓ `'Сидорчук-Б″янко'`

відповідають обмеженням домена VSURNAME_DOM. Такі стрічки, як :

```
'ФОП Стасевич'
'Інфосистема-2'
```

не відповідають обмеженням домена VSURNAME_DOM (тому що містять пробіл, цифру), тому цей домен не можна застосувати для назв юридичних осіб.

У операторі SIMILAR TO для позначення дозволеної кількості повторів використовуються такі зарезервовані символи повторюваності :

- ✓ **`?`** – те, що розміщене у масці перед символом повторюваності '**`?`**' : або одиночний символ, або перелік символів у квадратних дужках **`[]`**, або більш складна група символів в круглих дужках **`()`**, - може бути присутнє у стрічці **нуль чи один раз**;
- ✓ **`*`** – те, що розміщене у масці перед символом повторюваності '**`*`**' : або одиночний символ, або перелік символів у квадратних дужках **`[]`**, або більш складна група символів в круглих дужках **`()`**, - може бути присутнє у стрічці **нуль чи більше разів**;
- ✓ **`+`** – те, що розміщене у масці перед символом повторюваності '**`+`**' : або одиночний символ, або перелік символів у квадратних дужках **`[]`**, або

більш складна група символів в круглих дужках **()**, - може бути присутнє у **стрічці один чи більше разів**;

Детальніше про оператор SIMILAR TO можна почитати в [10, 13, 12]).

Приклад 2 : домен з назвою NSIGN_DOM :

```
CREATE DOMAIN NSIGN_DOM AS SMALLINT NOT NULL
  CHECK (VALUE IN (-1,0,1));
COMMENT ON DOMAIN NSIGN_DOM IS
  'Знак суми операції:
  -1="мінус або нуль", дебет ;
   0="будь-який", дебет і кредит ;
   1="плюс або нуль", кредит .';
```

Це цілочисельний домен, що може приймати одне із трьох значень: -1, 0 або 1, і не може мати невизначеного значення.

Приклад 3 : домен з назвою NSUMMA_DOM :

```
CREATE DOMAIN NSUMMA_DOM AS NUMERIC(15,2);
COMMENT ON DOMAIN NSUMMA_DOM IS
  'Сума в гривнях, після крапки лише дві цифри';
```

Це чисельний домен з фіксованою крапкою, не більше 15 цифр, в т.ч. дві після крапки, може мати невизначене значення.

Приклади створення двох таблиць з використанням доменів :

```
CREATE TABLE OPERTYPE (
      NOPERTYPE_ID  SMALLINT NOT NULL,
      VNAME         VARCHAR(32) NOT NULL,
      VNAMESHORT    VARCHAR(12) NOT NULL,
      NSIGN         NSIGN_DOM NOT NULL );
CREATE TABLE OPERATION (
      NOPER_ID    INTEGER NOT NULL,
      NCLIENT     INTEGER NOT NULL,
      NOPERTYPE   SMALLINT NOT NULL,
      NSUMMA      NSUMMA_DOM NOT NULL,
      DDATE       DATE NOT NULL,
      NOPERSTATE  SMALLINT NOT NULL,
      TEDIT       TIMESTAMP NOT NULL,
      NUSER       SMALLINT NOT NULL,
      DESCRIPTION  VARCHAR(200) NOT NULL );
```

Курсивом виділено домени, використані при створенні полів таблиці (порівняйте із прикладом створення таблиць в попередній лабораторній роботі).

Якщо таблиця вже створена, і якесь із полів необхідно змінити, це можна зробити SQL-запитом. Наприклад, зміна поля NSIGN таблиці OPERTYPE із простого типу SMALLINT на домен NSIGN_DOM виконується так :

```
update RDB$RELATION_FIELDS set
  RDB$FIELD_SOURCE = 'NSIGN_DOM'
  where (RDB$FIELD_NAME = 'NSIGN') and
  (RDB$RELATION_NAME = 'OPERTYPE');
```

Проте значно зручніше і наглядніше це робиться в редакторі таблиць програми IBExpert: в закладці «Поля» через контекстне меню (по натисненню правої клавіші мишки) : вибрати «Змінити поле».

Приклади створення процедур та тригерів.

Приклад 1: процедура для перевірки коректності номера накладної :

```
CREATE OR ALTER PROCEDURE INVOISE_IS_CORRECT
  ( INVOICE_NUMBER VARCHAR(15) not null )
  returns ( IS_CORRECT smallint )
  as
  begin
    IS_CORRECT = 0;
    if (:INVOICE_NUMBER SIMILAR TO
    '[0-9]+/([0-9]|[А-Я])+') then IS_CORRECT = 1;
    suspend;
  end
```

Процедура для перевірки використовує оператор SIMILAR TO. В масці, з якою оператор порівнює змінну INVOICE_NUMBER, вказано такі правила:

1) **[0-9]+** – з початку номеру накладної повинні бути цифри (у квадратних дужках вказано діапазон символів "**0-9**"), причому цифр може бути одна або більше (після квадратних дужок вказано символ повторюваності "**+**");
2) **/** – потім повинен бути символ коса риска "**/**" , і оскільки після цього символу немає жодного із символів повторюваності (**?*+**), це означає, що коса риска повинна обов'язково бути присутня;
3) **([0-9]|[А-Я])+** – через вертикальну риску "**|**" , яка означає "або", вказано що може бути <будь-яка цифра від 0 до 9> або <будь –яка літера кирилиці від А до Я>. Оскільки все це заключено в дужки і після дужок стоїть символ повторюваності "**+**" , це означає, що цифра чи літера може повторятися один або більше разів;

Якщо змінна INVOICE_NUMBER відповідає масці, то процедура повертає у вихідній змінній IS_CORRECT значення 1, якщо не відповідає – 0.

Запити:

```
SELECT IS_CORRECT FROM INVOISE_IS_CORRECT('00/1Ґ');
SELECT IS_CORRECT FROM INVOISE_IS_CORRECT('00/1ґ');
```

повернуть значення IS_CORRECT=1, оскільки номери накладної '00/1Ґ' та '00/1ґ' відповідають такій масці.

Запити:

```
SELECT IS_CORRECT FROM INVOISE_IS_CORRECT('00/1W');
SELECT IS_CORRECT FROM INVOISE_IS_CORRECT('001Г');
```

повернуть значення IS_CORRECT=0, оскільки номери накладних '00/1W' та '001Г' не відповідають такій масці (у першій присутній символ не кирилиці, у другій відсутня обов'язкова коса риска).

Приклад 2: тригер INVOISE_BI перед внесенням нового запису у таблицю

INVOISE «Накладні» :

```
CREATE OR ALTER TRIGGER INVOISE_BI
  FOR INVOICE ACTIVE BEFORE INSERT
  as
   declare variable nIS_CORRECT SMALLINT;
  begin
   -- Відсікання пробілів, перевід у верхній регістр
   New.INVOISE_NUMBER = Trim(Upper(:New.INVOISE_NUMBER));
   -- Перевірка коректності
   SELECT IS_CORRECT
     FROM INVOISE_IS_CORRECT( :New.INVOISE_NUMBER )
     INTO :nIS_CORRECT;
   IF (:nIS_CORRECT <> 1) THEN EXCEPTION INVOICE_BAD_EX
     'Невірний номер накладної '||:New.INVOISE_NUMBER;
   -- Автонумерація поля ID накладної
   New.nINVOICE_id = Gen_Id(gen_INVOICE_id,1);
  end
```

Тригер INVOISE_BI працює так :

1) Переводить внесене користувачем значення поля INVOISE_NUMBER «Номер накладної» (це значення доступне для читання і для редагування у тригері через змінну NEW.INVOISE_NUMBER) у верхній регістр (функція Upper()), і вилучає з початку і з кінця усі пробіли (функція Trim());
2) Викликає заздалегідь створену (див. попередній приклад) процедуру INVOISE_IS_CORRECT, яка перевіряє нове значення поля INVOISE_NUMBER «Номер накладної» перед вставкою запису у таблицю INVOICE на відповідність масці;
3) Якщо номер накладної не відповідає масці, у внутрішню змінну тригера nIS_CORRECT буде записано значення, не рівне 1, у цьому випадку тригер перериває операцію вставки із допомогою заздалегідь створеного виключення INVOICE_BAD_EX, причому текст помилки формується у тригері, щоб користувач побачив невірний номер накладної;
4) З допомогою заздалегідь створеного генератора gen_INVOICE_id автоматично нумерується поле nINVOICE_id «ID накладної». Зверніть увагу, що у тригері значення генераторів змінюються (викликами функції Gen_Id()) у кінці тригера, тобто вже після виконання усіх перевірок. Це робиться для того, щоб якщо в даних допущено помилку, дані не пройшли перевірку, і тригер генерує виключення і перериває операцію – щоб значення генераторів дарма не мінялися.

Приклад 3: тригер INVOISE_BU перед редагуванням існуючого запису у таблиці INVOISE «Накладні» :

```
CREATE OR ALTER TRIGGER INVOISE_BU
  FOR INVOICE ACTIVE BEFORE UPDATE
  as
   declare variable nIS_CORRECT SMALLINT;
```

```
begin
 -- Відсікання пробілів, перевід у верхній регістр
 New.INVOISE_NUMBER = Trim(Upper(:New.INVOISE_NUMBER));
 -- Перевірка коректності
 SELECT IS_CORRECT
   FROM INVOISE_IS_CORRECT( :New.INVOISE_NUMBER )
   INTO :nIS_CORRECT;
 IF (:nIS_CORRECT <> 1) THEN EXCEPTION INVOICE_BAD_EX
   'Невірний номер накладної '||:New.INVOISE_NUMBER;
 -- Блокування зміни поля ID накладної
 New.nINVOICE_id = :Old.nINVOICE_id;
end
```

Тригер INVOISE_BU працює аналогічно до тригера INVOISE_BU , за винятком лише одного: замість автонумерації поля nINVOICE_id «ID накладної» цьому полю примусово залишається старе (яке було до редагування) значення.

Приклад 4: тригер INVOISE_BD перед видаленням існуючого запису у таблиці INVOISE «Накладні» :

```
CREATE OR ALTER TRIGGER INVOISE_BD
  FOR INVOICE ACTIVE BEFORE DELETE
  as
  begin
   EXCEPTION INVOICE_BAD_EX 'Видалення накладної '||
      :Old.INVOISE_NUMBER||' заборонене, '||
      'скористайтесь операцією "Анулювати накладну"';
  end
```

Тригер INVOISE_BD без всяких умов IF (тобто завжди !) перериває операцію видалення із допомогою заздалегідь створеного виключення INVOICE_BAD_EX, причому текст помилки формується у тригері, щоб користувач побачив той номер накладної, яку він хотів би видалити.

Приклад 5: тригер DB_TRIG_ON_CONNECT після з'єднання користувача із базою даних :

```
CREATE OR ALTER TRIGGER DB_TRIG_ON_CONNECT
  ACTIVE ON CONNECT
  as
  begin
   IF ( (CURRENT_USER='SEMEN') OR
        (CURRENT_USER='ADMIN')    )
      THEN EXCEPTION USER_BAD_EX
           'Вам не дозволено роботу із БД';
  end
```

Тригер DB_TRIG_ON_CONNECT використовує заздалегідь створене виключення **USER_BAD_EX** для блокування з'єднання із базою даних двох користувачів з логінами 'SEMEN' та 'ADMIN'. Текст помилки формується у

тригері. За звичай у таких тригерах реєструють всі з'єднання користувачів із БД – шляхом вставки запису в спеціально створену для цього таблицю.

Завдання на лабораторну роботу.

1. Відкрити конспект лекцій, теми «Стандартні SQL та PSQL оператори» та «Стандартні SQL та PSQL функції», також відкрити посилання [10] – для вивчення і використання усіх можливостей SQL та PSQL.
2. В каталозі студента створити нову папку для даної роботи (наприклад "D:\Ivaniuk\FB\Labs\Лаб7\") (в нього будуть зберігатися деякі файли, про які буде сказано в завданні).
3. В програмі IBExpert з'єднатися із базою даних «Лабораторні студента…». У базі даних створити новий порожній скрипт з назвою «Лабораторна № 7» (в нього будуть зберігатися деякі запити, про які буде сказано в завданні).
4. Створити домен «Кількість товару» - чисельне (15,3), не порожнє. Застосувати цей домен для :
   - ✓ Поля «Кількість товару, що надійшов» Таблиці «Надходження товарів у магазин»;
   - ✓ Поля «Кількість реалізованого (проданого) товару» таблиці «Позиції реалізованих товарів»;
5. Створити домен «Вартість товару» - чисельне (15,2), не порожнє. Застосувати цей домен для :
   - ✓ Поля «Загальна сума по приходній накладній» Таблиці «Приходні накладні»;
   - ✓ Поля «Собівартість одиниці виміру товару, що надійшов» Таблиці «Надходження товарів у магазин»;
   - ✓ Поля «Загальна сума по всіх позиціях облікового запису» Таблиці «Журнал обліку реалізації товарів»;
   - ✓ Поля «Ціна реалізації (продажі) одиниці виміру товару» таблиці «Позиції реалізованих товарів»;
6. До таблиці «Працівники магазину» додати перевірку: або дата звільнення з роботи невизначена (NULL), або більша за дату влаштування на роботу.
7. До таблиці «Постачальники» додати перевірку: або дата розірвання договору невизначена (NULL), або більша за дату заключення договору.
8. До таблиці «Надходження товарів у магазин» додати перевірку: дата «Кінцева дата терміну придатності» більша за дату «Дата виготовлення» мінімум на 2 дні та максимум на один рік.
9. До таблиці «Надходження товарів у магазин» додати дві перевірки: кількість і собівартість товару додатні.
10. До таблиці «Позиції реалізованих товарів» додати дві перевірки: кількість і ціна товару додатні.
11. Створити індекси для прискорення пошуку :
    - ✓ По номеру приходної накладної;
    - ✓ По даті надходження товару в магазин;

- ✓ По номеру облікового запису в журналі реалізації (продажу) товару;
- ✓ По даті реалізації (продажу) товару;
- ✓ По даті виготовлення товару;
- ✓ По кінцевій даті придатності товару;

12. Перевести всі раніше внесені у таблицю «Користувачі бази даних» логіни користувачів (поле «Ім'я (логін) користувача в БД») у верхній регістр. SQL-запит, яким це зроблено, записати у скрипт «Лабораторна № 7». Після цього вибрати всі дані таблиці «Користувачі бази даних», експортувати в XML-таблицю, зберегти в папку лабораторної роботи у файл "Користувачі бази даних.XML".
13. Створити виключення «Некоректний користувач» - для подальшого використання в тригерах і процедурах, пов'язаних із таблицею «Користувачі бази даних».
14. Створити процедуру «Перевірка поточного користувача», яка б перевіряла користувача, з'єднаного із БД, по таблиці «Користувачі бази даних». Процедура без вхідних параметрів, і один вихідний параметр «ID користувача БД». У разі, якщо користувач не зареєстрований в таблиці – процедура повинна генерувати виключення з відповідним текстом помилки. У разі, якщо користувач зареєстрований в таблиці, але його обліковий запис заблоковано (цілочисельне поле «Користувач заблокований» не рівне нулю) – процедура також повинна генерувати виключення з відповідним текстом помилки. В разі якщо все ґаразд – процедура повинна повертати у вихідний параметр ID користувача БД із таблиці «Користувачі бази даних».
15. Створити тригер на подію з'єднання із БД, який повинен використати процедуру «Перевірка поточного користувача» для блокування не зареєстрованих та заблокованих користувачів. Цей тригер повинен спрацьовувати для усіх користувачів, крім SYSDBA .
16. Створити домен «Номер документу» - символьне, не порожнє, довжиною до 14-ти символів. Застосувати цей домен для :
    - ✓ Поля «Номер приходної накладної» Таблиці «Приходні накладні»;
    - ✓ Поля «Номер облікового запису» Таблиці «Журнал обліку реалізації товарів»;
17. Створити процедуру «Перевірка номеру приходної накладної», яка б перевіряла вхідний стрічковий параметр «Номер накладної» на відповідність такій масці: спочатку велика кирилична літера "Н" (від "Накладна"), потім одна чи більше цифр, потім коса риска, потім один чи більше символів "цифра або кирилична літера". Тип вхідного стрічкового параметру «Номер накладної» повинен бути доменом «Номер документу».
18. Написати запит, який вибирає поля «ID накладної» і «Номер накладної» з таблиці «Приходні накладні» - всі існуючі записи. З допомогою вкладеного запиту (після SELECT перед FROM) вставити виклик процедури «Перевірка номеру приходної накладної», і таким чином виявити ті записи, які не відповідають масці. Даний SQL-запит записати у скрипт «Лабораторна № 7». Вручну відредагувати ті номери накладних,

які не відповідають масці. Після цього результат виконання запиту експортувати в XML-таблицю, зберегти в папку лабораторної роботи у файл “Приходні накладні.XML”.

19. Створити виключення «Некоректна приходна накладна» - для подальшого використання в тригерах і процедурах, пов’язаних із таблицею «Приходні накладні».
20. Створити процедуру «Зміна номеру приходної накладної до стандартного вигляду», яка повинна мати один вхідний параметр «Номер накладної» (тип параметру є доменом «Номер документу») та один вихідний параметр «Стандартний номер накладної» (тип параметру також є доменом «Номер документу»). Процедура працює за такою логікою:
    - ✓ Вхідний номер накладної перетворюється у верхній регістр, пробіли зліва і справа видаляються;
    - ✓ Якщо першим символом в номер накладної не внесена велика кирилична літера ”Н” – додати її примусово (скористатися функцією `SUBSTR()`, яка копіює частину стрічки);
    - ✓ Перевірити утворений номер накладної процедурою «Перевірка номеру приходної накладної», якщо перевірка невдала – згенерувати виключення; повідомлення про помилку повинно містити адекватний текст та невірний номер накладної.
    - ✓ Якщо перевірка вдала – у вихідний параметр повертається перетворений номер накладної.
21. Переписати існуючий тригер, що спрацьовує перед подією вставки запису в таблицю «Приходні накладні», на наступну бізнес-логіку :
    - ✓ Викликати процедуру «Стандартний номер накладної», у вхідний параметр якої передати нове значення номеру накладної. Якщо у процедурі виникає виключення - операцію вставки буде заблоковано. Якщо не виникне – то потрібно примусово встановити те значення, яке поверне процедура, в поле «Номер приходної накладної»;
    - ✓ Перевірити, щоб дата накладної не була більша від поточної дати, якщо перевищує - заблокувати операцію вставки, повідомлення про помилку повинно містити адекватний текст та відобразити некоректно задану дату накладної (скористатися функцією `CAST()`, яка перетворить тип ”дата” у тип ”рядок”).
    - ✓ Примусово встановити значення ”0” в поле «Загальна сума»;
    - ✓ Примусово встановити значення ”1” в поле «ID статусу накладної» (статус 1 = «Внесення»);
    - ✓ Примусово встановити значення ”Поточна дата та час на сервері” в поле «Дата+Час внесення»;
    - ✓ Примусово встановити значення ”ID поточного користувача” в поле «ID користувача БД, який вніс накладну», використати при цьому раніше написану процедуру «Перевірка поточного користувача»;
    - ✓ Автонумерація поля «ID приходної накладної».
22. Створити новий тригер, що спрацьовує перед подією редагування запису в таблиці «Приходні накладні», на наступну бізнес-логіку :

- ✓ Перевірити старе значення поля «ID статусу накладної». Якщо це значення не рівне 1 (статус 1 = «Внесення») – заблокувати операцію редагування. Текст помилки повинен відображати пояснення, у якому статусі знаходиться дана накладна;
- ✓ Викликати процедуру «Стандартний номер накладної», у вхідний параметр якої передати нове значення номеру накладної. Якщо у процедурі виникає виключення - операцію редагування буде заблоковано. Якщо не виникне – то потрібно примусово встановити те значення, яке поверне процедура, в поле «Новий номер приходної накладної»;
- ✓ Перевірити, щоб нова дата накладної не була більша від поточної дати, якщо перевищує - заблокувати операцію редагування, повідомлення про помилку повинно містити адекватний текст та відобразити некоректно задану дату накладної (скористатися функцією `CAST()`, яка перетворить тип ”дата” у тип ”рядок”).
- ✓ Примусово встановити значення ”Поточна дата та час на сервері” в поле «Дата+Час внесення»;
- ✓ Примусово встановити значення ”ID поточного користувача” в поле «ID користувача БД, який вніс накладну», використати при цьому раніше написану процедуру «Перевірка поточного користувача»;
- ✓ Заблокувати редагування поля «ID приходної накладної» (примусово встановити значення ”Старе ID приходної накладної” в поле «ID приходної накладної»).

23. Створити новий тригер, що спрацьовує перед подією видалення запису в таблиці «Приходні накладні», на наступну бізнес-логіку :
    - ✓ Повністю заблокувати операцію видалення. Текст помилки повинен відображати пояснення, що операція видалення заміняється операцією «Анулювання»;
24. Від’єднатися від усіх БД, зробити холодну копію БД «Лабораторні студента ...» у створену для даної лабораторної роботи папку (не забувайте, що в копію БД входить і файл паролів !!!).
25. Закрити всі програми, всі файли у створеній папці заархівувати.
26. Надіслати файл архіву на електронну адресу свого викладача, у листі вказати «Студент «ПІБ повністю”, лаб.робота номер 7».
27. Базу «Лабораторні студента …» скопіювати на власну флешку, оскільки вона використовується в подальших роботах.

Контрольні запитання.

1. Чому немає необхідності у створенні індексів для пошуку товарів по назві в таблицях «Надходження товарів у магазин» та «Позиції реалізованих товарів» ?
2. Яким чином обчислити різницю у днях між двома датами ?
3. Опишіть технологію, з допомогою якої блокуються небажані операції в базі даних.

4. Опишіть технологію, з допомогою якої в таблиці БД вноситься інформація про дійсні дату+час виконання тієї чи іншої операції, а не те значення, яке вводить користувач в SQL-запит.
5. Опишіть технологію, з допомогою якої в таблиці БД вноситься інформація про дійсного користувача, який виконує ту чи іншу операцію, а не те значення, яке вводить користувач в SQL-запит.

ЛАБОРАТОРНА РОБОТА № 8 : Створення об'єктів бази даних (продовження).

Тема: Реляційні бази даних.
Мета: Продовження вивчення об'єктів реляційних баз даних: процедури і тригери.

Завдання на лабораторну роботу.

1. Відкрити конспект лекцій, теми «Стандартні SQL та PSQL оператори» та «Стандартні SQL та PSQL функції», також відкрити посилання [10] – для вивчення і використання усіх можливостей SQL та PSQL.
2. В каталозі студента створити нову папку для даної роботи (наприклад "D:\Ivaniuk\FB\Labs\Лаб8\") (в нього будуть зберігатися деякі файли, про які буде сказано в завданні).
3. В програмі IBExpert з'єднатися із базою даних «Лабораторні студента…». У базі даних створити новий порожній скрипт з назвою «Лабораторна № 8» (в нього будуть зберігатися деякі запити, про які буде сказано в завданні).
4. Створити виключення «Некоректне надходження товарів у магазин» - для подальшого використання в тригерах і процедурах, пов'язаних із таблицею «Надходження товарів у магазин».
5. Створити процедуру «Перевірка статусу накладної» із одним вхідним параметром «ID приходної накладної». Процедура повинна :
   - ✓ Визначити і повернути у вихідний параметр ID статусу, в якому ця накладна знаходиться (з поля «ID статусу накладної» таблиці «Приходні накладні»).
   - ✓ Якщо вказане значення «ID приходної накладної» в таблиці «Приходні накладні» відсутнє, повинно генеруватися виключення, в тексті помилки повинно бути вказане неіснуюче значення ID приходної накладної (скористатися функцією CAST(), яка перетворить тип "число" у тип "рядок").
   - ✓ Якщо отриманий з БД «ID статусу накладної» не рівний 1 (Внесення), то повинно генеруватися виключення, в тексті помилки повинно бути вказано, що накладна з таким-то номером знаходиться в такому-то статусі (номер і назву статусу взяти з таблиць БД), і внесення змін в цю накладну заборонене.

6. Створити процедуру «Перевірка дати приходної накладної» із трьома вхідними параметрами:
    1) «ID приходної накладної»;
    2) «Дата виготовлення»;
    3) «Кінцева дата терміну придатності».

   Процедура повинна :
   - ✓ Визначити і повернути у вихідний параметр дату приходної накладної (з поля «Дата приходної накладної» таблиці «Приходні накладні»).
   - ✓ Якщо вказане значення «ID приходної накладної» в таблиці «Приходні накладні» відсутнє, повинно генеруватися виключення, в тексті помилки повинно бути вказане неіснуюче значення ID приходної накладної (скористатися функцією CAST(), яка перетворить тип ”число” у тип ”рядок”).
   - ✓ Якщо «Дата виготовлення» перевищує отриману з БД «Дату приходної накладної», повинно генеруватися виключення, в тексті помилки повинно бути вказано, що введена дата виготовлення товару не може бути більша за дату його надходження в магазин по накладній такій-то (в тексті повинні виводитися обидві дати, для цього скористатися функцією `CAST()`, та номер накладної).
   - ✓ Якщо «Кінцева дата терміну придатності» менша або рівна отриманої з БД «Дати приходної накладної», повинно генеруватися виключення, в тексті помилки повинно бути вказано, що введена кінцева дата терміну придатності товару не може бути менша або рівна за дату його надходження в магазин по накладній такій-то (в тексті повинні виводитися обидві дати, для цього скористатися функцією `CAST()`, та номер накладної).
7. Переписати існуючий тригер, що спрацьовує перед подією вставки запису в таблицю «Надходження товарів у магазин», на наступну бізнес-логіку :
   - ✓ Викликати процедуру «Перевірка статусу накладної», у вхідний параметр якої передати **нове** значення ID приходної накладної. Якщо у процедурі виникає виключення - операцію вставки буде заблоковано (це повинно відбутися, якщо ID статусу накладної» не рівний 1 ”Внесення”);
   - ✓ Викликати процедуру «Перевірка дати приходної накладної», у вхідні параметри якої передати нові значення ID приходної накладної, дати виготовлення і дати придатності товару. Якщо у процедурі виникає виключення - операцію вставки буде заблоковано (це повинно відбутися, якщо дати товару некоректні по відношенню до дати накладної);
   - ✓ Автонумерація поля «ID надходження».
8. Створити новий тригер, що спрацьовує перед подією редагування запису в таблиці «Надходження товарів у магазин», на наступну бізнес-логіку :
   - ✓ Викликати процедуру «Перевірка статусу накладної», у вхідний параметр якої передати **старе** значення ID приходної накладної. Якщо у процедурі виникає виключення - операцію редагування буде

заблоковано (це повинно відбутися, якщо старе ID статусу накладної не рівний 1 ”Внесення”);

- ✓ Викликати процедуру «Перевірка статусу накладної», у вхідний параметр якої передати **нове** значення ID приходної накладної. Якщо у процедурі виникає виключення - операцію редагування буде заблоковано (це повинно відбутися, якщо нове ID статусу накладної не рівний 1 ”Внесення”);
- ✓ Викликати процедуру «Перевірка дати приходної накладної», у вхідні параметри якої передати нові значення ID приходної накладної, дати виготовлення і дати придатності товару. Якщо у процедурі виникає виключення - операцію редагування буде заблоковано (це повинно відбутися, якщо дати товару некоректні по відношенню до дати накладної);
- ✓ Заблокувати редагування поля «ID надходження» (примусово встановити значення ”Старе ID надходження” в поле «ID надходження»).

9. Створити новий тригер, що спрацьовує перед подією видалення запису в таблиці «Надходження товарів у магазин», на наступну бізнес-логіку :
   - ✓ Викликати процедуру «Перевірка статусу накладної», у вхідний параметр якої передати **старе** значення ID приходної накладної. Якщо у процедурі виникає виключення - операцію видалення буде заблоковано (це повинно відбутися, якщо ID статусу накладної не рівний 1 ”Внесення”);
10. Створити процедуру «Закриття приходної накладної» із одним вхідним параметром «ID приходної накладної». Процедура повинна :
    - ✓ Викликати процедуру «Перевірка статусу накладної», у вхідний параметр якої передати значення ID приходної накладної. Якщо у процедурі виникає виключення - операцію «Закриття» буде заблоковано (це повинно відбутися, якщо ID статусу накладної не рівний 1 ”Внесення”);
    - ✓ В циклі FOR по всіх записах таблиці «Надходження товарів у магазин», які відносяться до даної накладної, підрахувати і зберегти у внутрішню змінну процедури загальну суму по даній приходній накладній. Внутрішню змінна процедури повинна мати тип домену «Вартість товару». При кожному присвоєнні значення цій змінній необхідно використовувати функцію `CAST()` для приведення числа (результату) до типу даних домена «Вартість товару».
    - ✓ Якщо загальна сума не додатна – заблокувати операцію «Закриття», викликавши виключення з відповідним текстом помилки, в якому вивести номер накладної;
    - ✓ В операторі UPDATE редагувати два поля таблиці «Приходні накладні» :
      1) Примусово встановити щойно обчислене значення загальної суми по даній приходній накладній в поле «Загальна сума по приходній накладній»;

2) Встановити значення 2 (Виконано) в поле «ID статусу накладної»;

11. Створити процедуру «Анулювання приходної накладної» із одним вхідним параметром «ID приходної накладної». Процедура повинна :
    - ✓ Викликати процедуру «Перевірка статусу накладної», у вхідний параметр якої передати значення ID приходної накладної. Якщо у процедурі виникає виключення - операцію «Анулювання» буде заблоковано (це повинно відбутися, якщо ID статусу накладної не рівний 1 ”Внесення”);
    - ✓ В операторі UPDATE редагувати одне поле таблиці «Приходні накладні» :
        1) Встановити значення 3 (Анульовано) в поле «ID статусу накладної»;
12. Провести тестування створених процедур і тригерів на таблиці «Приходні накладні» і «Надходження товарів у магазин» :
    - ✓ Почати нову накладну (внести запис в таблицю «Приходні накладні»), при цьому спеціально вносити невірний номер, повинна виникнути помилка;
    - ✓ Почати нову накладну (внести запис в таблицю «Приходні накладні»), при цьому спеціально вносити невірну дату, повинна виникнути помилка;
    - ✓ Внести правильний запис в таблицю «Приходні накладні»;
    - ✓ Спробувати одразу виконати процедуру «Закриття приходної накладної», повинна виникнути помилка;
    - ✓ Внести кілька товарів в накладну, при цьому спеціально вносити невірні дати виготовлення і придатності, повинні виникати помилки.
    - ✓ Внести кілька коректних записів в дану накладну;
    - ✓ Пробувати редагувати різні поля по надходженням товарів;
    - ✓ Виконати процедуру «Закриття приходної накладної»;
    - ✓ Аналогічно – на ще одній накладній - протестувати процедуру «Анулювання приходної накладної»;
    - ✓ При виявленні невірної роботи процедур і тригерів – вносити в них виправлення.
    - ✓ При завершенні тестування враховувати те, що викладач при перевірці роботи буде також моделювати різні коректні і некоректні ситуації.
13. Вибрати всі дані із перегляду «Виконані надходження товарів у магазин» (сортовані по ID надходження), результат експортувати в XML-таблицю, зберегти в папку лабораторної роботи у файл “Виконані надходження товарів у магазин.XML”.
14. Вибрати всі дані із перегляду «Надходження які оформляються» (сортовані по ID надходження), результат експортувати в XML-таблицю, зберегти в папку лабораторної роботи у файл “Надходження які оформляються.XML”.
15. Вибрати всі дані із перегляду «Анульовані надходження товарів у магазин» (сортовані по ID надходження), результат експортувати в XML-таблицю, зберегти в папку лабораторної роботи у файл “Анульовані

надходження товарів у магазин.XML".

16. Від'єднатися від усіх БД, зробити холодну копію БД «Лабораторні студента ...» у створену для даної лабораторної роботи папку (не забувайте, що в копію БД входить і файл паролів !!!).
17. Закрити всі програми, всі файли у створеній папці заархівувати.
18. Надіслати файл архіву на електронну адресу свого викладача, у листі вказати «Студент «ПІБ повністю", лаб.робота номер 8».
19. Базу «Лабораторні студента …» скопіювати на власну флешку, оскільки вона використовується в подальших роботах.

Контрольні запитання.

1. Виходячи з чого визначаються алгоритми процедур і тригерів, які програмуються в базу даних?
2. Чому в промислових системах кожен тип документу має кілька можливих статусів, і в залежності від цих статусів є різні алгоритми обробки документів ?
3. Яким чином можна перевірити, чи запрограмовані алгоритми працюють коректно ?

ЛАБОРАТОРНА РОБОТА № 9 : Адміністрування прав користувачів.

Тема: Реляційні бази даних.

Мета: Практичне вивчення технології адміністрування прав користувачів по доступу до об'єктів реляційної БД.

Завдання на лабораторну роботу.

1. В каталозі студента створити нову папку для даної роботи (наприклад "D:\Ivaniuk\FB\Labs\Лаб9\") (в нього будуть зберігатися деякі файли, про які буде сказано в завданні).
2. В програмі IBExpert з'єднатися із базою даних «Лабораторні студента…». У базі даних створити новий порожній скрипт з назвою «Лабораторна № 9» (в нього будуть зберігатися деякі запити, про які буде сказано в завданні).
3. Створити таблицю «Ролі користувачів» з полями:
   - ✓ ID ролі: цілочисельне (Integer), PK (Primary Key, Первинний ключ), AI (Auto Increment, автонумерація значень з допомогою генератора і тригера);
   - ✓ Ім'я ролі: стрічкове 20 символів, унікальне, дозволені лише великі латинські літери і знак підкреслення.
   - ✓ Назва ролі: стрічкове 30 символів, унікальне.
   - ✓ Назви посад, які підтримуються даною роллю, через кому: стрічкове 50

символів.

4. Внести в таблицю «Ролі користувачів» наступні ролі:
   - ✓ Ім’я ”R_ADMIN”, Назва «Адміністратор», для посади «Системний адміністратор - програміст»;
   - ✓ Ім’я ”R_DIRECTOR”, Назва «Директор», для посади «Директор»;
   - ✓ Ім’я ”R_BUHGALTER”, Назва «Бухгалтер», для посади «Бухгалтер»;
   - ✓ Ім’я ”R_MANAGER”, Назва «Менеджер», для посади «Менеджер - комірник»;
   - ✓ Ім’я ”R_SELLER”, Назва «Продавець», для посад «Продавець, Старший продавець».
5. До таблиці «Користувачі бази даних» додати поле «ID ролі». Заповнити це поле таким чином, щоб в кожного користувача РБД була своя роль. Після цього для нового поля створити FK (вторинний ключ) на таблицю «Ролі користувачів».
6. Створити в базі даних ті ролі, які внесено в таблицю «Ролі користувачів».
7. Перевірити, чи кожен користувач, який є в таблиці «Користувачі бази даних», зареєстрований в базі даних користувачів. При відсутності – внести (gaроль у всіх «1»).
8. Підключити кожного користувача, який є в таблиці «Користувачі бази даних» до тієї ролі, яку йому щойно було призначено.
9. До створеного раніше перегляду «Користувачі бази даних» додати наступні нові поля :
   - ✓ ID ролі;
   - ✓ Ім’я ролі.
   - ✓ Назва ролі.
   - ✓ Назви посад, які підтримуються даною роллю.
10. Вибрати всі дані із цього перегляду (сортовані по ID користувача), результат експортувати в XML-таблицю, зберегти в папку лабораторної роботи у файл “Користувачі бази даних.XML”. Два останні SQL-запити: 1) створення перегляду «Користувачі бази даних» та 2) вибір даних з цього перегляду – зберегти в БД у скрипті «Лабораторна № 9».
11. Відкрити «Менеджер прав» програми IBExpert, вибрати «Права для: ролі», і для кожної створеної ролі призначити права на об’єкти бази даних. «Менеджер прав» відкривається через меню, або подвійним кліком мишки на назві будь-якої ролі в вікні в експлорері БД (DataBase Explorer).
12. Права на таблиці, і на інші об’єкти бази даних, для кожної ролі призначати, виходячи з посадових обов’язків :
    - ✓ Для ролі «Адміністратор» потрібні всі права на таблиці «Працівники», «Користувачі» і «Ролі», а для всіх інших таблиць – лише перегляд;
    - ✓ Для ролі «Директор» потрібні права на перегляд всіх таблиць;
    - ✓ Для ролі «Бухгалтер» потрібні всі права на ті таблиці, в яких обліковуються надходження і реалізація товарів, а для всіх інших таблиць – лише перегляд;
    - ✓ Для ролі «Менеджер» потрібні всі права на ті таблиці, в яких обліковується номенклатура товарів, постачальники товарів,

надходження і реалізація товарів, а для всіх інших таблиць – лише перегляд;

- ✓ Для ролі «Продавець» потрібні всі права на ті таблиці, в яких обліковується реалізація товарів, а для всіх інших таблиць – лише перегляд;

Права на перегляди (VIEW) надаються лише на читання.

Права на процедури надаються, виходячи із з посадових обов'язків, а також із того, які функції виконують дані процедури.

13. Перевірити надані права. Для цього з'єднатися із базою даних по черзі із кожним простим користувачем, для якого призначено роль. **При з'єднанні вводити ім'я користувача, пароль, та назву тієї ролі яка цьому користувачу призначена.** Відкривати таблиці та перегляди, і перевіряти як діють права доступу (вносити нові дані, редагувати існуючі). Пробувати виконувати процедури, і перевіряти як діють права на їх виконання. При виявленні помилок – виправляти права.
14. Після того, як усі права сконфігуровано **і перевірено**, необхідно зберегти п'ять зображень вікна «Менеджер прав» програми IBExpert, вибравши по черзі кожну із п'яти ролей користувачів. Зображення зберегти в папку лабораторної роботи у файл з відповідним іменем, наприклад : "Права ролі R_ADMIN.Jpg".
15. Від'єднатися від усіх БД, зробити холодну копію БД «Лабораторні студента ...» у створену для даної лабораторної роботи папку (не забувайте, що в копію БД входить і файл паролів !!!).
16. Закрити всі програми, всі файли у створеній папці заархівувати.
17. Надіслати файл архіву на електронну адресу свого викладача, у листі вказати «Студент «ПІБ повністю", лаб.робота номер 9».
18. Базу «Лабораторні студента …» зкопіювати на власну флешку, оскільки вона використовується в подальших роботах.

Контрольні запитання.

1. Які переваги адміністрування прав з допомогою ролей користувачів порівняно із наданням прав безпосередньо користувачам ?
2. Яким чином за звичай визначають перелік тих ролей, які потрібно створити в реляційній БД конкретного підприємства ?
3. Яким чином визначається за звичай визначають перелік тих прав, які належить надати тій чи іншій ролі на конкретному підприємстві ?

ДОДАТОК 1
ФАЙЛИ, НЕОБХІДНІ ДЛЯ ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ

Для виконання даного курсу лабораторних робіт надаються файли, заархівовані в один архів "**FB_Labs_Files.Zip**" :

1. SampleDB_1.Fdb – зразкова база даних FireBird «**Приклад 1**».

2. EMPLOYEE.Fdb – зразкова база даних FireBird «**Службовці**».
3. SQL-скрипт "Лабораторна 3 скрипт 1.Sql".
4. SQL-скрипт "Лабораторна 4 скрипт 1.Sql".
5. SQL-скрипт "Лабораторна 5 скрипт 1.Sql".
6. SQL-скрипт "Лабораторна 5 скрипт 2.Sql".
7. SQL-скрипт "Лабораторна 6 скрипт 1.Sql".

Ці файли повинні бути завантажені зі сторінки курсу в MOODLE (лінк «Файли для лабораторних робіт»), збережені на комп'ютер і розархівовані: на домашньому ПК в каталог C:\Firebird\DATA\, на лабораторному - в особистий каталог студента на диску D.

НАВЧАЛЬНА ЛІТЕРАТУРА

1. Сімашко В. Й. Сучасні технології адміністрування та програмування реляційних баз даних: Курс лекцій з дисципліни "Адміністрування та програмування в корпоративних базах даних" для студентів спеціальності 051 „Економіка" ОПП „Економічна кібернетика". Рівне: РДГУ, 2022. 76 с.
2. Сімашко В. Й. Адміністрування та програмування в корпоративних базах даних: методичний посібник для підготовки курсової роботи з дисципліни студентів спеціальності 051 „Економіка" ОПП „Економічна кібернетика". Рівне: РДГУ, 2018. 40 с.
3. Джеймс Р. Грофф, Пол Н. Вайнберг. SQL : Полное руководство. Второе издание. - Санкт-Петербург : BHV, 2002. 814 с.
4. Борри Х. FireBird: Руководство разработчика баз данных. Санкт-Петербург : "BHV", 2006. 1104 с.
5. Кауффман Д. SQL. Программирование / Д. Кауффман, Б.Матсик, К.Спенсер и др. М.: БИНОМ ЛЗ, 2002. 744 с.
6. Оппель Энди. Изучаем SQL/Серия 'Раскрытие тайн'. К.: НТ-Прес, 2007. 320 с.

ІНФОРМАЦІЙНІ РЕСУРСИ

7. Офіційний сайт розробників FireBird. URL: http://firebirdsql.org/ -
8. SQL FireBird. URL: http://firebirdsql.su/doku.php -
9. Документація по SQL FireBird. URL: http://www.firebirdsql.org/file/documentation/reference_manuals/Firebird-Language-Reference-Russian-11-Nov-2014.pdf -
10. IBExpert. *Вікіпедія*. URL: http://ru.wikipedia.org/wiki/IBExpert -
11. Документація по FireBird від розробників РСКБД, та посилання на інші джерела інформації. URL: http://www.firebirdsql.org/en/documentation/ -
12. Довідник по SQL. URL: http://www.sql-ex.ru/help/ -.
13. Англомовний довідник по базовому SQL РСКБД InterBase. URL: http://ibase.ru/v6/doc/langref.zip.
14. Англомовний довідник по розширенням SQL РСКБД FireBird версії 1.5.

URL: http://ibase.ru/firebird/fb_1_53_releasenotes.pdf -.
15. Посилання для завантаження останньої версії програми IBExper. URL: http://www.ibexpert.com/rus/ibe_sfx.exe -.