Міністерство освіти і науки України
Чорноморський національний університет імені Петра Могили

Г. В. Горбань

ОБ'ЄКТНІ БАЗИ ДАНИХ

Методичні вказівки
до виконання лабораторних робіт

Випуск 266

Миколаїв – 2019

УДК 004.65(076)
Г 67

Рекомендовано до друку вченою радою Чорноморського національного університету імені Петра Могили (протокол № 6 від 28 лютого 2019 р.).

Рецензенти:

Литвиненко В. І., доктор технічних наук, професор, завідувач кафедри інформатики і комп'ютерних наук Херсонського національного технічного університету.

Зіновата С. Л., кандидат технічних наук, доцент кафедри системного забезпечення Одеського національного політехнічного університету.

Г 67 Горбань Г. В. Об'єктні бази даних : методичні вказівки до виконання лабораторних робіт / Г. В. Горбань. – Миколаїв : Вид-во ЧНУ ім. Петра Могили, 2019. – 108 с. (Методична серія ; вип. 266).

Методичні вказівки містять опис 10 лабораторних робіт, у яких розглядається постреляційна СКБД Caché, однією з головних особливостей якої є єдина архітектура даних, що дозволяє представити дані у трьох різних моделях. Основну увагу приділено об'єктній моделі даних з точки зору її порівняння з реляційною моделлю даних. Також розглянуто Web-застосунки для роботи з даними СКБД Caché CSP та Zen та доступ до об'єктних даних за допомогою Java. Методичні вказівки призначені для студентів спеціальностей 121 – «Інженерія програмного забезпечення» та 122 – «Комп'ютерні науки», а також можуть бути корисними для студентів інших спеціальностей галузі знань 12 – «Інформаційні технології».

УДК 004.65(076)

ISSN 1811-492X

ЗМІСТ

ВСТУП

На сьогодні основним засобом накопичення та використання структурованої інформації служать бази даних, які є основою будь-якої інформаційної системи (ІС). Найбільш розповсюдженою моделлю даних у системах керування даними (СКБД) поки ще є реляційна. Але зараз настає ера так званих постреляційних СКБД, до яких належать і об'єктні. Деякі спеціалісти розглядають об'єктні системи як серйозного конкурента реляційних систем (у всякому разі SQL-систем), і якщо не повною мірою, то для певних прикладних програм. Зокрема, SQL-системи не мають таких можливостей, які мають мови об'єктно-орієнтованого програмування, основна ідея методології якого полягає в тому, щоб огородити користувача від конструкцій типу поля, рядка, стовпця та інших, тобто підвищення рівня абстракції. Але підвищення «інтелектуальності» баз даних в одних випадках може стати корисним, а в інших – ні. Однак більшість фахівців в ІТ вважають, що об'єктні системи є великим кроком на шляху розвитку технологій керування базами даних.

Однією з небагатьох постреляційних СКБД є Caché, яка підтримує три типи моделей даних: реляційну, об'єктну і багатовимірну. Тому робочою програмою передбачено використання достатньо розповсюдженої СКБД Caché. Її вивчення, крім загальної характеристики на лекціях, передбачає її засвоювання під час виконання лабораторних робіт у комп'ютерних класах, а також під час самостійної роботи, для чого передбачені завдання на створення баз даних та програмних засобів їх ведення із застосуванням СКБД Caché.

Основною метою вивчення дисципліни є формування системи знань із методології й теоретичних основ та умінь з проектування, створення та використання об'єктних баз даних у різних предметних сферах. Засвоєння дисципліни базується на концепціях системного аналізу та сучасних технологій розробки інформаційних систем.

У результаті вивчення дисципліни студенти повинні мати *знання* з:

– концепцій моделювання предметного середовища з розподіленою обробкою інформації;
– етапів і методів проектування об'єктних баз даних;
– організації об'єктних баз даних;
– архітектури і основних блоків сучасних об'єктних СКБД та їх функціональне призначення;
– методів забезпечення, контролю та відновлення цілісності даних;

У результаті вивчення дисципліни студенти повинні *уміти*:

– моделювати предметні середовища з розподіленою обробкою інформації;
– організовувати бази даних у середовищі СКБД Caché;
– програмувати об'єктні бази даних мовою Caché Object Script;
– забезпечувати контроль та відновлення цілісності даних у СКБД Caché;
– експлуатувати бази даних, що створені у СКБД Caché;
– застосовувати інструментальні програмні засоби СКБД Caché.

ЛАБОРАТОРНА РОБОТА № 1

«ЗНАЙОМСТВО ЗІ СКБД CACHÉ. ЇЇ ОСНОВНІ УТИЛІТИ. НАЛАШТУВАННЯ СКБД CACHÉ ДЛЯ СТВОРЕННЯ НОВОЇ БАЗИ ДАНИХ»

Теоретичні відомості

Постреляційна СКБД Caché реалізує на рівні міжнародних стандартів **три моделі даних: реляційну, об'єктну і ієрархічну** і надає можливість їх спільного використання в рамках одного проекту. Оскільки кожна модель має як свої переваги, так і недоліки, їх спільне використання дозволяє агрегувати переваги, одночасно компенсуючи недоліки кожної моделі. Архітектуру СКБД Caché можна порівняти з літаком, що складається з частин (моделей даних), які за своєю природою прагнуть впасти на землю, але, працюючи злагоджено, долають цю тенденцію.

В основі архітектури постреляційної СКБД Caché (Рис. 1.1) лежить високопродуктивне ієрархічне ядро, орієнтоване на роботу з транзакціями. Це ядро є основою М-стандарту і розвивається корпорацією InterSystems з 1978 року. Основними характеристиками ядра Caché є продуктивність і масштабованість.

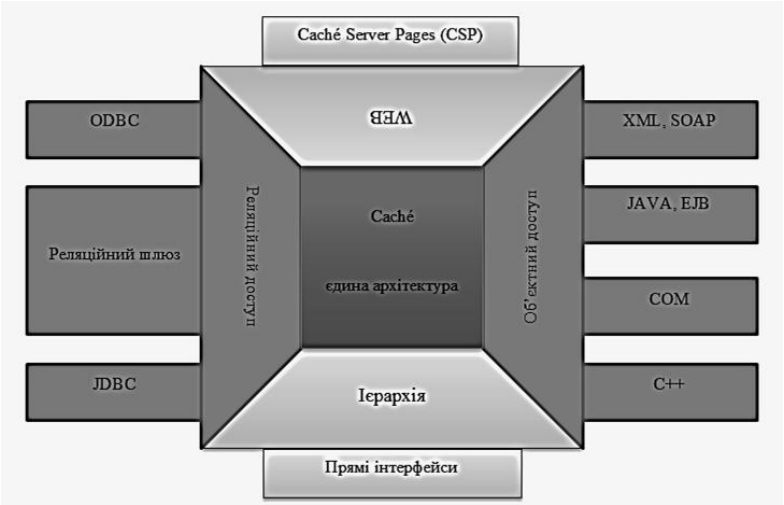


Рис. 1.1. Архітектура Caché

У рамках єдиної архітектури даних Caché багатовимірні масиви даних можуть бути представлені у вигляді класів об'єктів (сервер Caché Objects) і у вигляді реляційних таблиць (сервер Caché SQL). При цьому підтримка об'єктного і реляційного представлень здійснюється в повному обсязі і відповідає основним вимогам міжнародних стандартів (ODMG і SQL92). Варто зазначити, що, на відміну від об'єктно-реляційних або реляційно-об'єктних СКБД, у Caché сервери об'єктного і реляційного представлення даних знаходяться на одному логічному рівні, що забезпечує високу швидкість роботи з даними, представленими як у вигляді класів об'єктів, так і у вигляді реляційних таблиць і дозволяє розширити стандартну об'єктну модель запитами SQL, а механізм SQL – об'єктними розширеннями.

Інструментарій СКБД Caché

Після завершення встановлення Caché, на панелі задач MS Windows з'явиться піктограма Caché-куба. Caché-куб має два стани – сірий куб відповідає вимкненому серверу Caché, синій – активному серверу Caché. Піктограма Caché-куба має сірий колір також у разі встановлення клієнтської версії Caché. Для запуску сервера (сервер Caché може бути запущений тільки на тій машині, де він встановлений) необхідно натиснути на піктограму куба правою кнопкою миші і вибрати пункт меню Start Caché (Запуск Caché) (рис. 1.2).

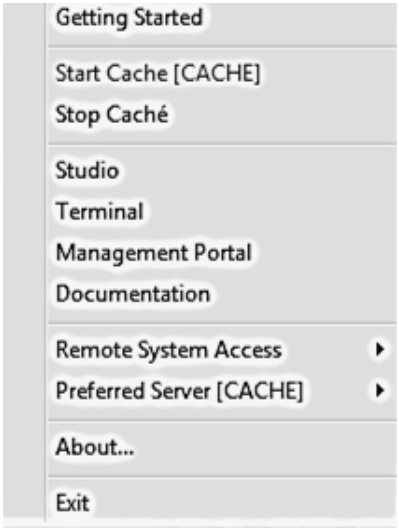


Рис. 1.2. Меню куба Caché

Розглянемо основні утиліти Caché.

1. Caché Studio

Для визначення класів об'єктів, редагування програм чи CSP-сторінок, а також для вирішення багатьох інших завдань розробки проектів Caché надає Caché Studio (Рис. 1.3). Це інтегроване середовище розробки з розвиненим графічним відладником.

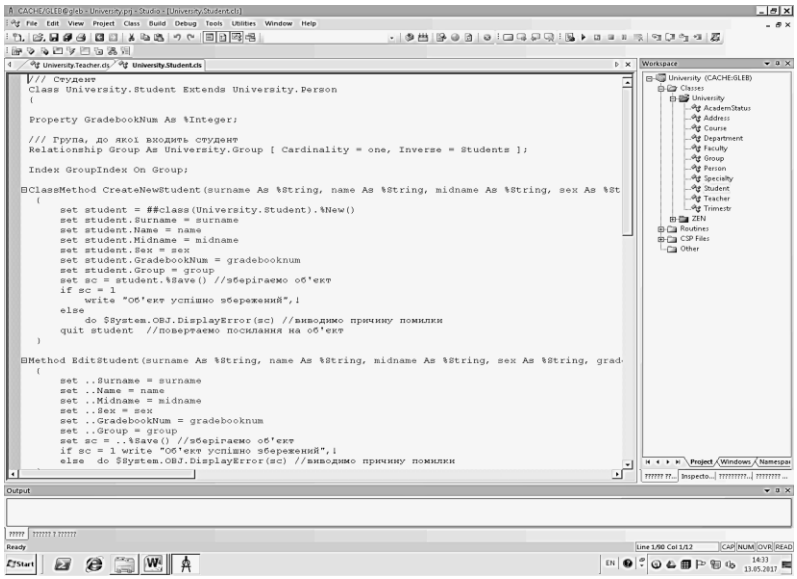


Рис. 1.3. Caché Studio

2. Caché Terminal

Часто для тестування створеного програмного коду буває необхідно змоделювати звернення до бази, перевірити взаємодію класів, працездатність написаних методів. Для виконання таких дій можна скористатися утилітою емуляції ASCII терміналу Caché Terminal (Рис. 1.4). Caché Terminal має ряд обмежень, у числі яких є обмеження на довжину рядка.

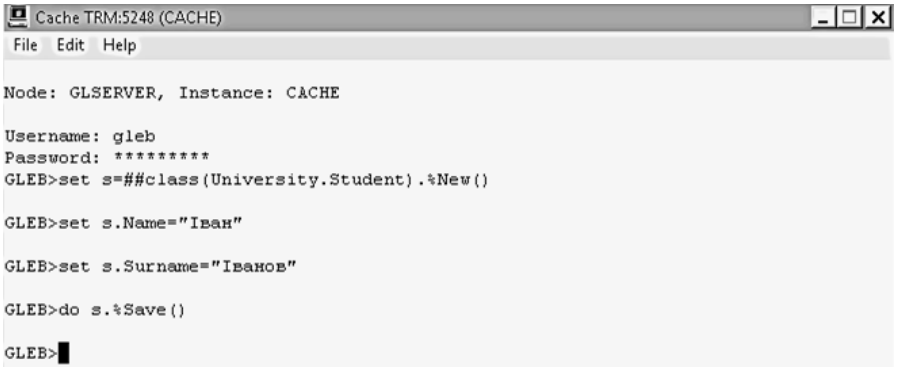


Рис. 1.4. Caché Terminal

3. Портал управління системою

У Порталі управління системою об'єднані утиліти, призначені для адміністрування сервера Caché, утиліти управління даними і засоби для виконання типових операцій адміністрування і налаштування.

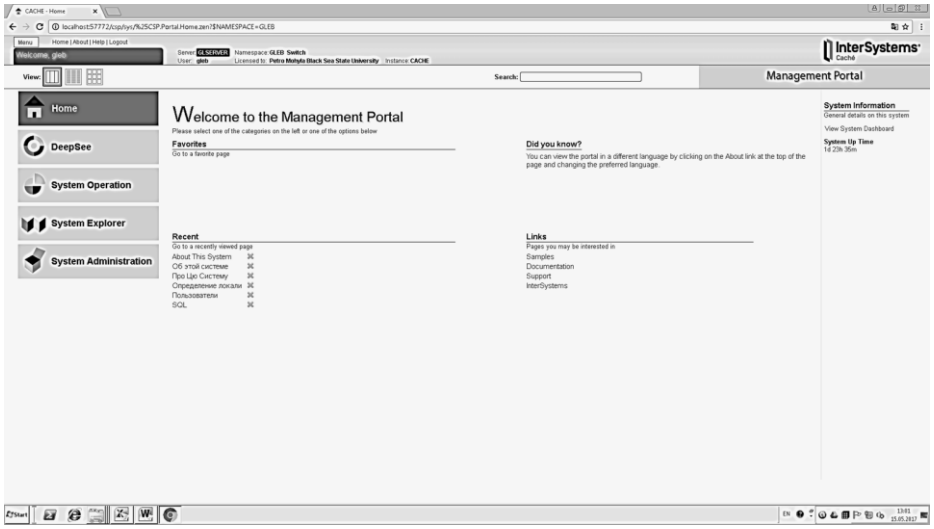


Рис. 1.5. Головна сторінка Порталу управління системою

Створення нової бази даних у СКБД Caché

Уся інформація (дані і програми) в базах даних Caché зберігається в глобалах – багатовимірних масивах даних, що зберігаються.

База даних в Caché – це бінарний файл із фіксованим ім'ям CACHE.DAT, у якому зберігаються глобали. Для кожної нової бази даних необхідно визначити окрему директорію для зберігання файлу CACHE.DAT. Бази даних можуть працювати під управлінням різних серверів Caché. Область (Namespace) у Caché – це логічна карта, на якій вказані імена багатовимірних масивів (глобалів), а також імена БД Caché, у яких вони зберігаються. Під час роботи з глобалами і програмами використовується ім'я області.

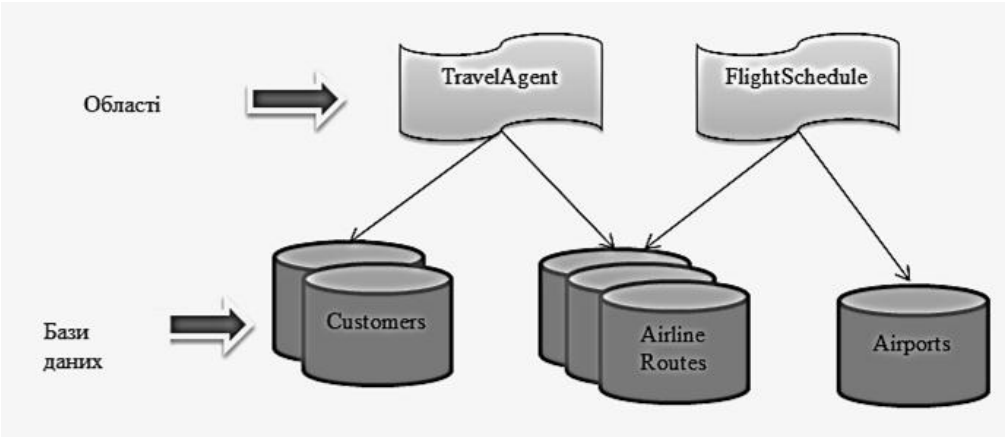


Рис. 1.6. Області даних у СКБД Caché

Створити нову область та базу даних у СКБД Caché можна за допомогою вкладки **System Administration** (Системне адміністрування). Натиснувши на цю вкладку, далі ми йдемо таким шляхом: **Configuration->System Configuration->Namespaces**.

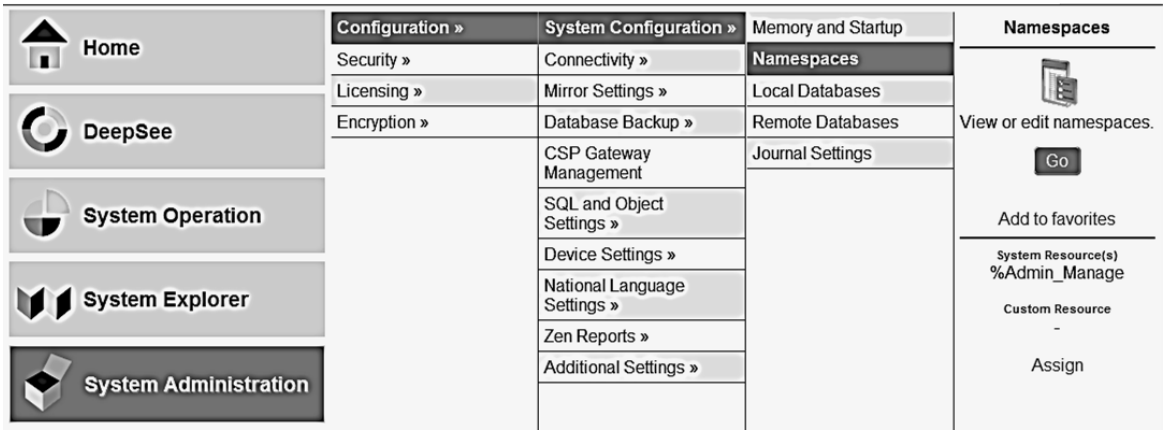


Рис. 1.7. Перехід до вкладки «Області даних» у Порталі управління системою

Далі перед нами буде виведений список областей, уже створених у системі. Щоб створити нову область, натискаємо **Create New Namespace**.

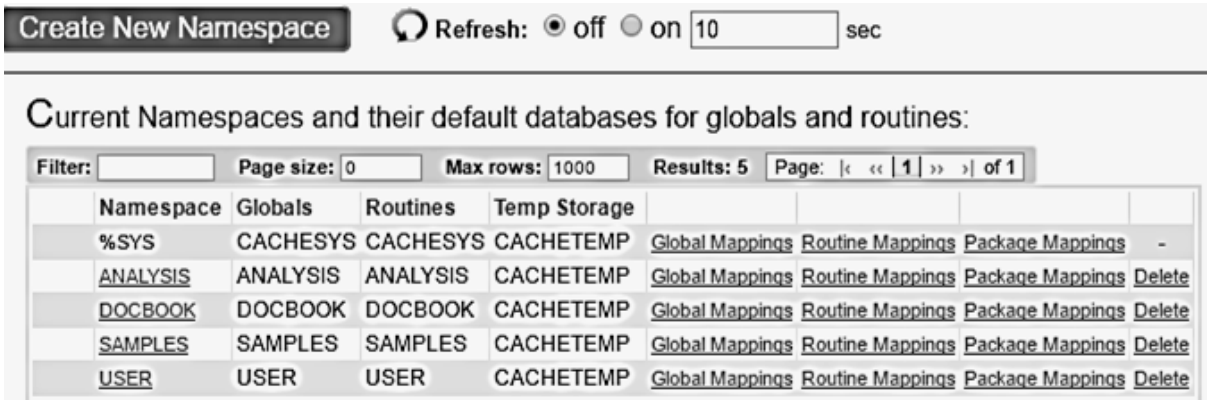


Рис. 1.8. Список областей у системі

Далі показано процес створення нової області даних:

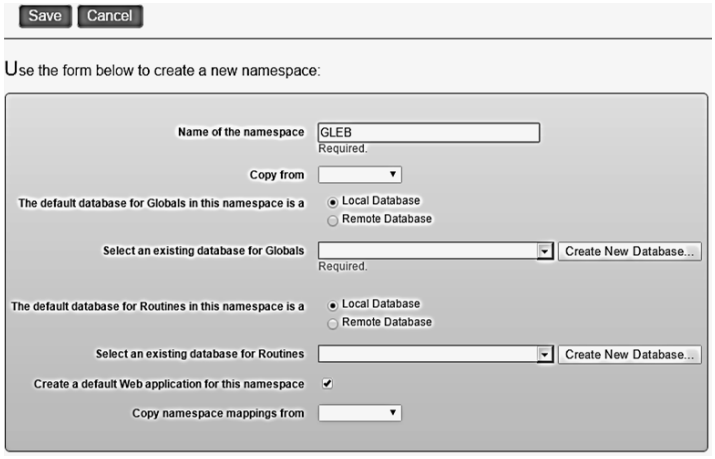


Рис. 1.9. Створення нової області даних

Далі потрібно вказати фізичне розташування нової бази даних на сервері. Для цього після натиснення кнопки **«Create New Database»**, створюючи БД для глобалів або для програм, ми потрапляємо до майстра створення бази даних. Потім створену БД необхідно вказати і в іншому списку, який буде випадати.

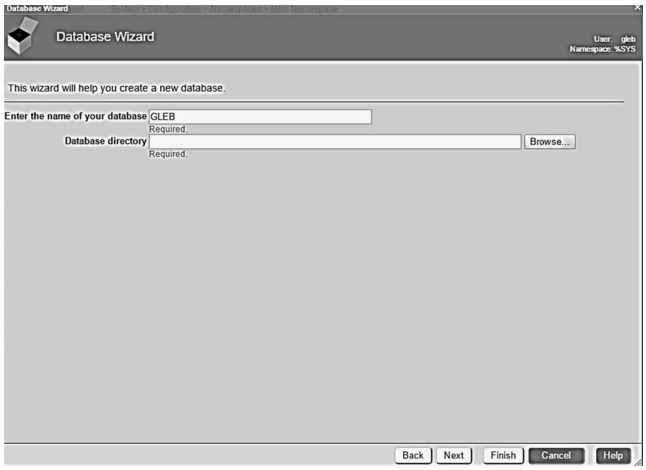


Рис. 1.10. Майстер створення бази даних

Указавши назву бази даних, що буде прив’язана до нової області (це може бути нова база даних або вже існуюча та її назва може як збігатися з назвою області, так і відрізнятись від неї), ми переходимо до наступного етапу, в якому потрібно вказати фізичне розташування бази даних.

Усі бази даних зберігаються на сервері у папці **«<Диск>:\Intersystems\ Cache\mgr\ <Папка бази даних>».** Власне сама БД представляє собою файл CACHE.DAT у відповідній папці. У цій лабораторній роботі створюється нову базу даних, тому необхідно вказати назву нової папки для БД. У процесі створення нової БД папка з вказаною назвою буде створена.

Обрання вже існуючої папки означатиме прив’язку певної існуючої БД до нової області.

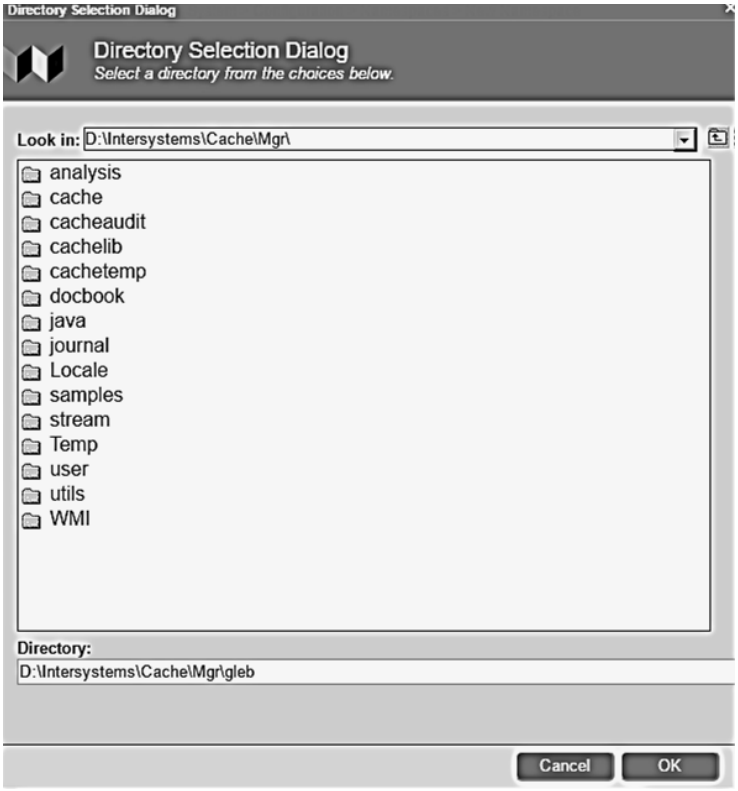


Рис. 1.11. Прив'язка БД до нової області

Після цього, якщо була вказана нова папка, можна побачити таке повідомлення: **Directory does not exist! Please change if you do not wish the directory be created.** Переходимо до наступного етапу.

Після того, як ми вказали директорію зберігання бази даних, майстер запитає її початковий розмір в МБ а також деяку додаткову інформацію, яка тут не розглядається. Натискаємо **Next**.

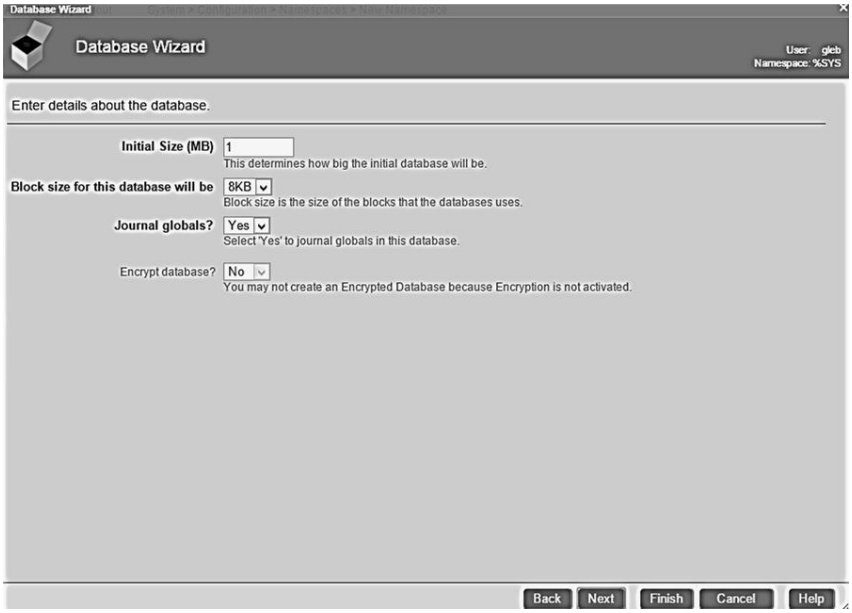


Рис. 1.12. Введення початкового розміру

Далі необхідно вказати ресурс для нової БД. Тут ми все залишаємо за замовчуванням.

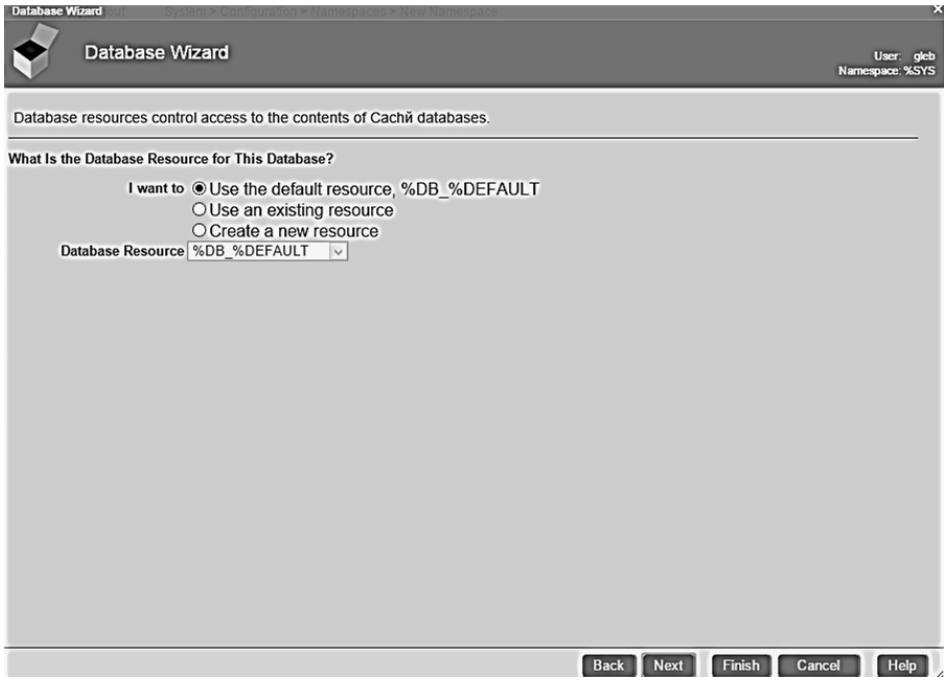


Рис. 1.13. Указання ресурсу для БД

Потім повертаємось до створення нової області даних та зберігаємо її, натиснувши **Save**. У результаті буде створена нова область даних.

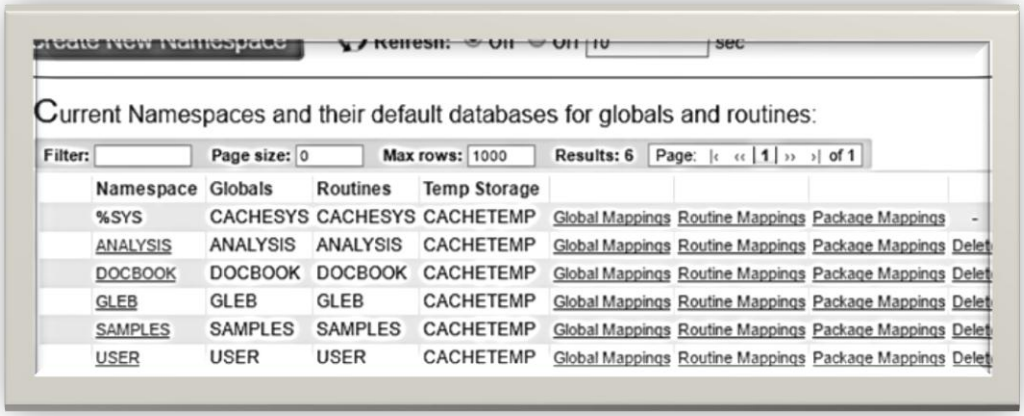
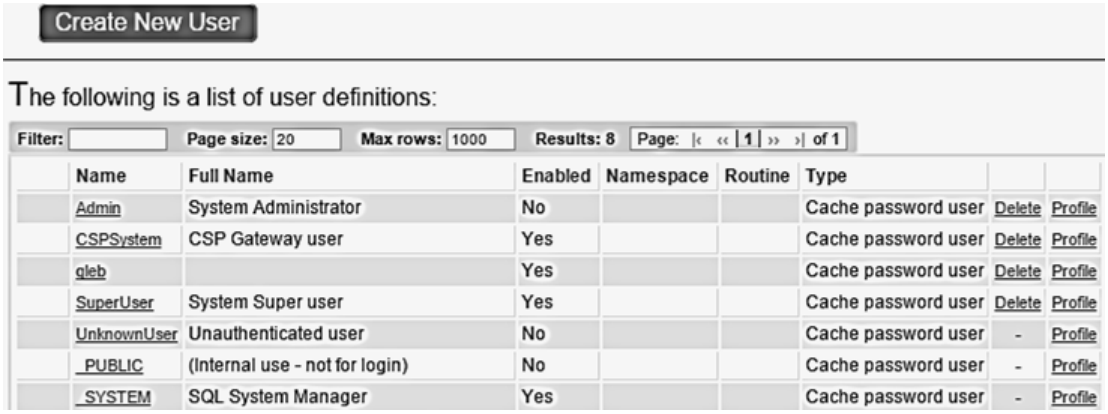


Рис. 1.14. Створення нової області даних

Редагування користувачів

Для редагування інформації про користувачів також використовується вкладка **System Administration**. До речі, до неї є доступ лише у тому випадку, коли у користувача, який користується Порталом управління системою, є права адміністратора системи.

Для редагування користувачів необхідно відкрити: **System Administration->Security->Users.** Зайшовши сюди, ми побачимо всіх користувачів системи.



Name	Full Name	Enabled	Namespace	Routine	Type		
Admin	System Administrator	No			Cache password user	Delete	Profile
CSPSystem	CSP Gateway user	Yes			Cache password user	Delete	Profile
gleb		Yes			Cache password user	Delete	Profile
SuperUser	System Super user	Yes			Cache password user	Delete	Profile
UnknownUser	Unauthenticated user	No			Cache password user	-	Profile
_PUBLIC	(Internal use - not for login)	No			Cache password user	-	Profile
_SYSTEM	SQL System Manager	Yes			Cache password user	-	Profile

Рис. 1.15. Список користувачів системи

Клацнувши **Edit** для певного користувача, ми переходимо на сторінку його редагування.

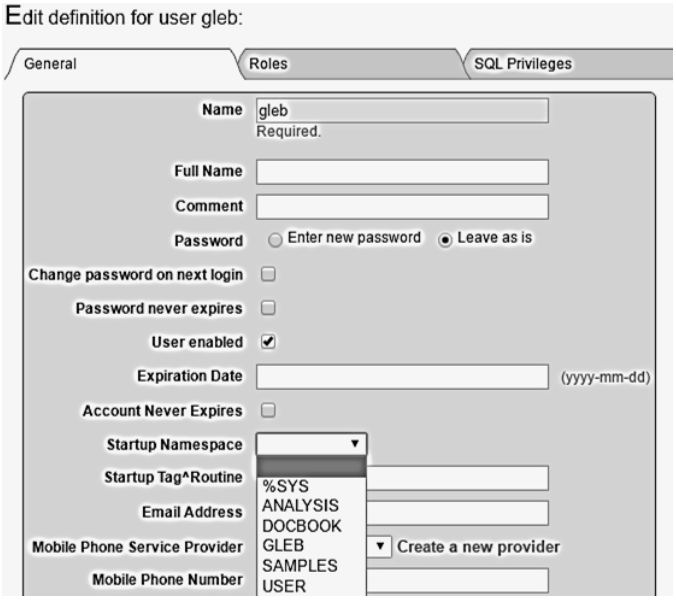


Рис. 1.16. Редагування сторінки користувача

Тут можна змінити таку інформацію, як повне ім'я користувача (**Full Name**), пароль (**Password**), дату закінчення дії акаунта (**Expiration Date**), область даних, з якої користувач буде стартувати в системі (**Startup Namespace**). Якщо змінити цю область, то саме з неї і почнеться робота користувача в Терміналі.

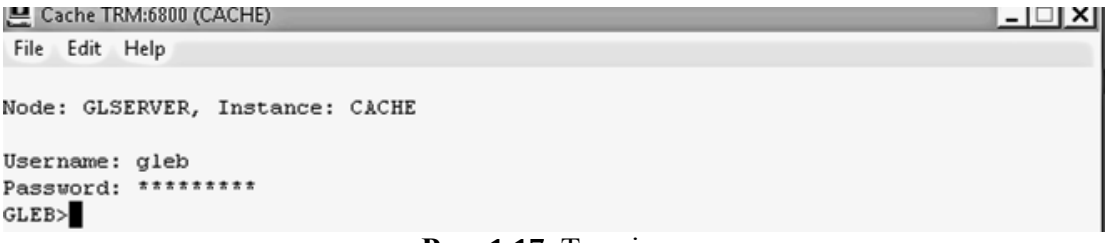


Рис. 1.17. Термінал

Завдання

1. Створити на сервері Caché нову область даних та базу даних, назвою яких буде прізвище відповідного студента.
2. Зробити нову область областю запуску для відповідного користувача.

Контрольні питання

1. Як створити нову область даних у СКБД Cache?
2. Що представляє собою база даних у СКБД Cache?
3. Які основні етапи створення нової бази даних у СКБД Cache?
4. Як змінити область даних, з якої користувач стартує в системі?

ЛАБОРАТОРНА РОБОТА № 2

«ЗНАЙОМСТВО З CACHÉ STUDIO ТА ТЕРМІНАЛОМ. СТВОРЕННЯ НОВОГО КЛАСУ В СКБД CACHÉ. ОСНОВНІ МЕТОДИ МАНІПУЛЮВАННЯ ДАНИМИ: СТВОРЕННЯ, МОДИФІКАЦІЯ ТА ВИДАЛЕННЯ. РІЗНІ СПОСОБИ ДОСТУПУ ДО ДАНИХ У СКБД CACHÉ»

Теоретичні відомості

Caché Studio – інтегроване середовище для розробки моделі БД, написання програм і веб-застосунків, яке:

– дає можливість створювати і редагувати визначення класів, CSP-сторінок, програм на мові Caché ObjectScript (COS), використовуючи єдине середовище;

– підтримує повнотекстове редагування коду з підсвічуванням команд і перевіркою синтаксису мов COS, Basic, Java, SQL, JavaScript, HTML, XML;

– дозволяє виконувати відлагодження;

– організовує код проекту;

– надає майстра для створення класів, методів, властивостей, зв’язків, веб-форм;

– підтримує версійність.

Підключення до сервера Caché

При відкритті Студії потрібно виконати підключення до сервера. Для підключення до певної області даних необхідно:

1. У вікні Server Connection (Рис. 2.1) обрати доступний сервер та натиснути ОК. При виборі сервера у меню куба **Preferred Server** цей етап буде пропущено.

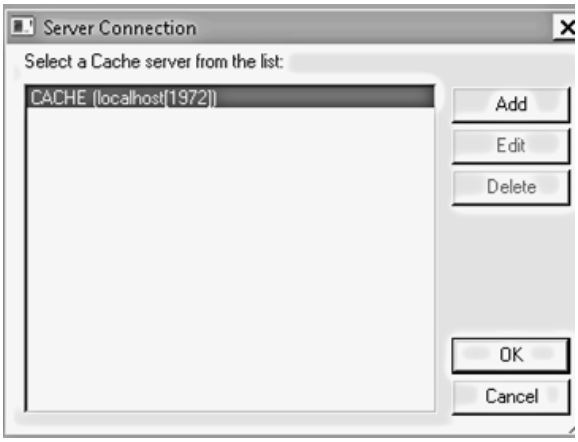


Рис. 2.1. Вікно Server Connection

2. У вікні Log on to <ServerName> ввести логін і пароль (Рис. 2.2).

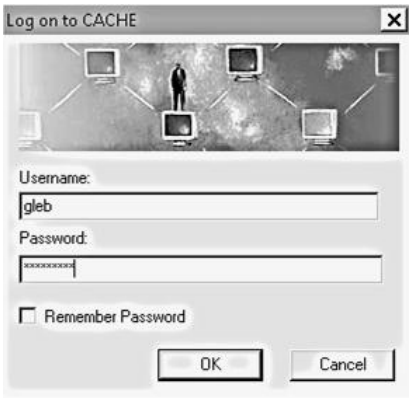


Рис. 2.2. Авторизація користувача

3. У вікні Caché Connection Manager (Рис. 2.3) з'явиться список доступних областей даних на обраному сервері. Зі списку потрібно обрати вашу область даних, створену під час виконання лабораторної роботи № 1, і натиснути OK.

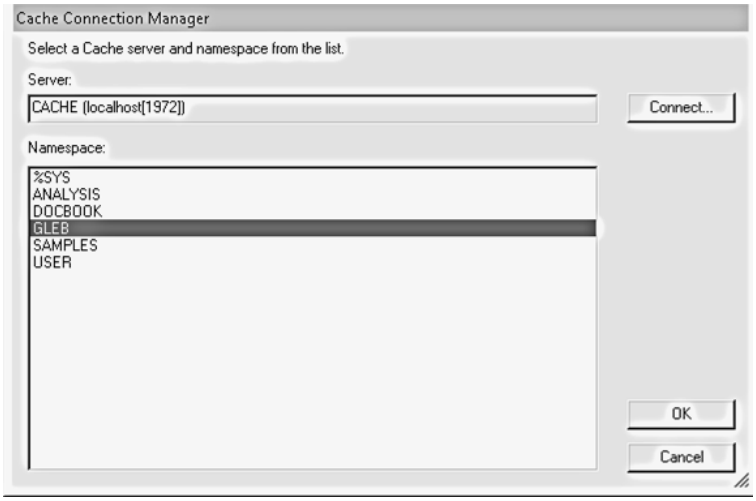


Рис. 2.3. Caché Connection Manager

Увага! Caché Studio запам'ятовує область, з якою працювали протягом останнього сеансу роботи у ній, незалежно від того, чи той самий користувач працював минулого разу чи ні. Тому при авторизації у Caché Studio може виникнути така ситуація: коли програма намагається під'єднатися до області, з якою працював останній користувач, але новий користувач, який авторизувався, не має доступу до цієї області. У цьому випадку відкриється таке вікно.

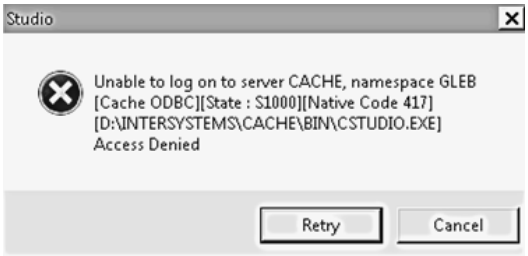


Рис. 2.4. Вікно помилки

У цьому випадку треба змінити область за допомогою пункту меню **File->Change Namespace** або просто натиснувши клавішу **F4**. Тоді відкриється вікно Caché Connection Manager, у якому треба обрати відповідну область даних.

І взагалі потрібно завжди слідкувати за тим, яка область даних відкрита у Студії в даний момент і за необхідності змінити область даних на свою.

Створення нового класу засобами Caché Studio

1. Для початку роботи створимо у Caché Studio новий проект (**File->New Project**).
2. А потім відразу збережемо його у відповідній області даних (**File->Save Project** або **File->Save Project As…**)

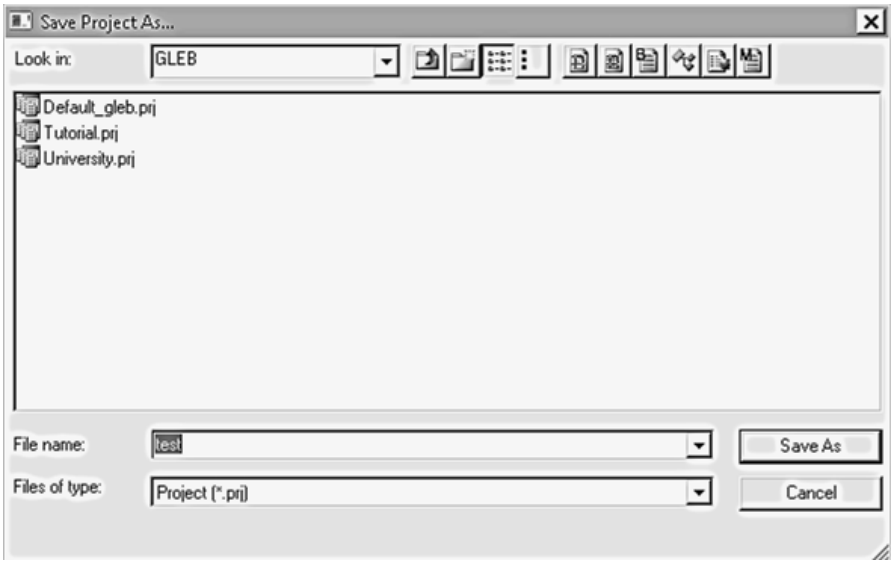


Рис. 2.5. Створення та збереження нового проекту

Для створення нового класу необхідно запустити майстер (**File->New->General->Cache Class Definition**).

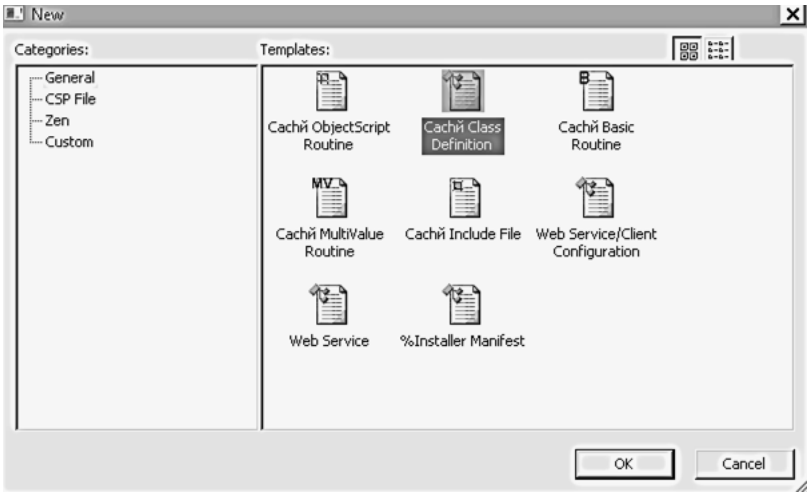


Рис. 2.6. Створення нового класу

Або можна скористатись робочою областю, на якій потрібно буде натиснути правою кнопкою миші **Classes**, а потім у контекстному меню обрати **Create New Class**. Можна додавати у проект і інші класи, що вже існують у поточній області, обравши **Add.**

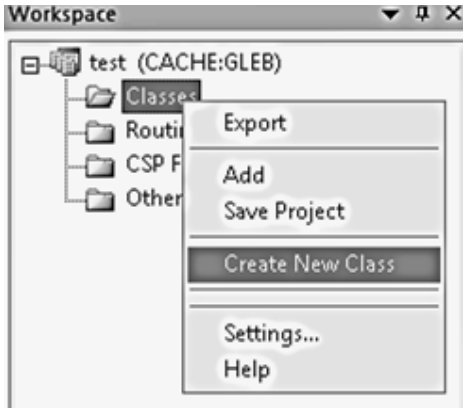


Рис. 2.7. Меню вкладки «Classes»

В обох випадках буде викликаний майстер створення класів. На першому етапі ми вказуємо власне ім'я класу та ім'я пакета (схеми даних), до якого буде належить клас. За замовчанням іменем пакета стане **User.**

Назвемо пакет **Test (**він також буде створений так само як клас, оскільки це буде перший клас, який буде входити до цього пакета). У подальшому ви будете створювати нові класи у вже існуючому пакеті (схемі даних).

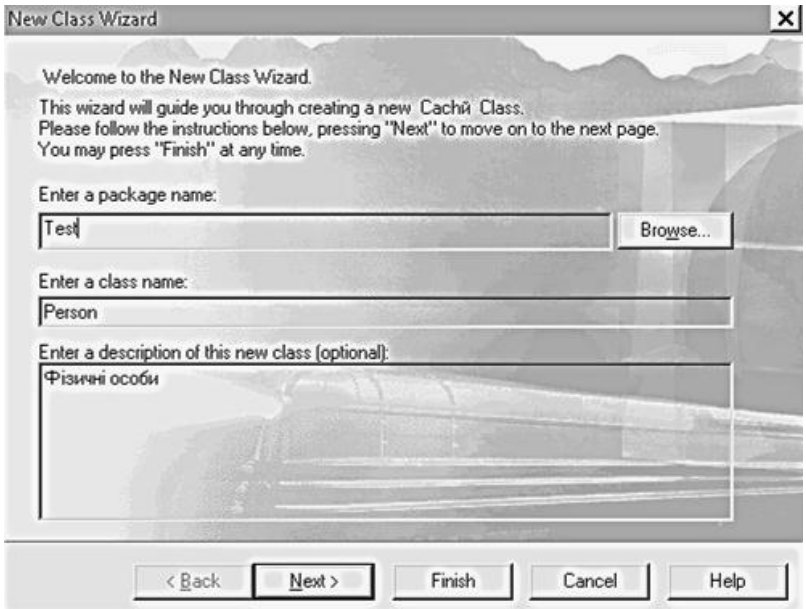


Рис. 2.8. Завдання імені класу та пакета при створенні нового класу

На другому етапі потрібно обрати тип класу. Ми оберемо **Persistent –** той, що зберігається. Це означає, що екземпляри цього класу будуть зберігатись у базі даних.

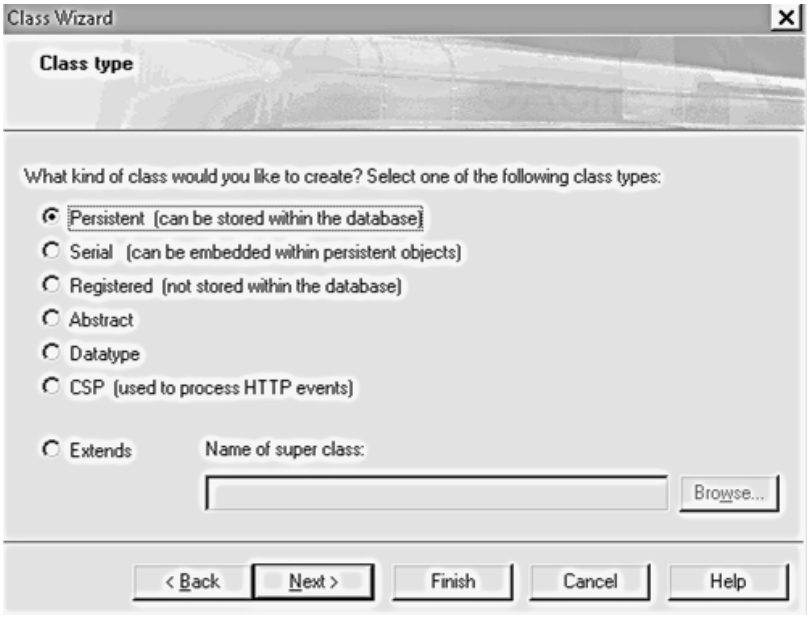


Рис. 2.9. Завдання типу класу

На третьому етапі можна обрати власника класу (**Owner**) та назву таблиці SQL, що буде відповідати класу в реляційному представленні. Ім'я класу та ім'я таблиці можуть бути різними. Обидва параметри не є обов'язковими.

Також на цьому етапі можна вказати додаткові можливості класу: підтримку XML (**XML Enabled**), підтримку моделі даних ZEN (**Zen DataModel**), можливість генерації тестових даних (**Data Population**) та чи дозволено MultiValue (**MultiValue Enabled**). Ми вкажемо підтримку XML.

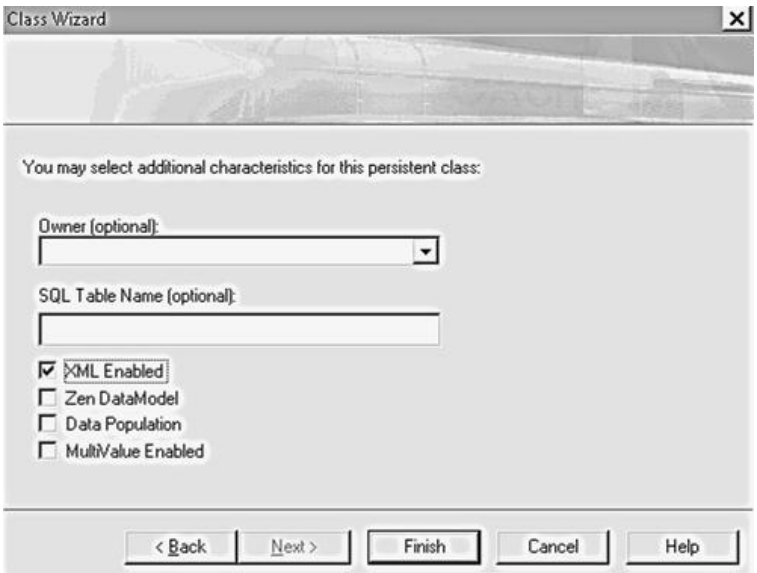


Рис. 2.10. Завдання додаткових можливостей класу

Після цього вже можна натискати Finish, хоча ще є четвертий необов'язковий етап, на якому можна вказати, що ми хочемо перевизначити сетер або гетер для цього класу. Указаний етап тут опущено.

2. Тепер, коли клас створено, визначимо для нього властивості. Для цього натискаємо на панелі «Члени класу»  крайню ліву кнопку або заходимо в меню **Class->Add>Property…** В обох випадках відкриється майстер створення нової властивості.

На першому етапі ми вказуємо назву властивості та її словесний опис (у згенерованому коді він буде відображатись у якості коментаря).

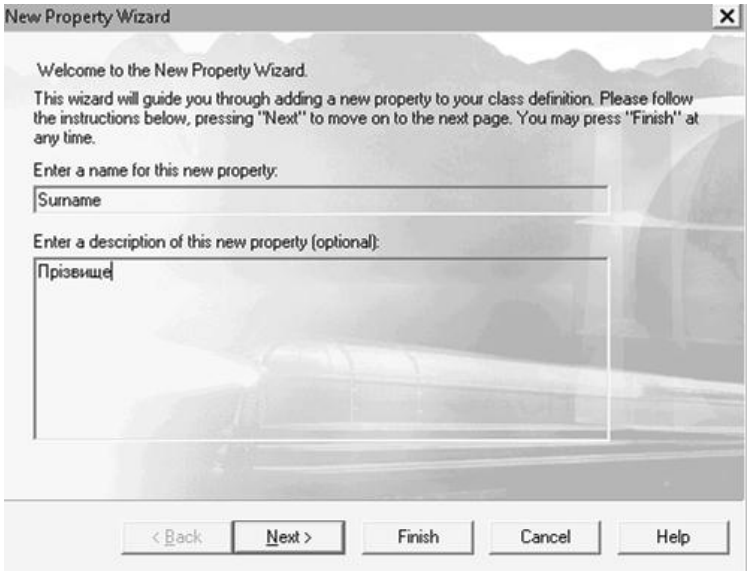


Рис. 2.11. Майстер створення нової властивості

На другому етапі ми вказуємо тип властивості, який можна обрати за допомогою кнопки **Browse…** Для властивості «прізвище» доцільно обрати тип даних *%String*.

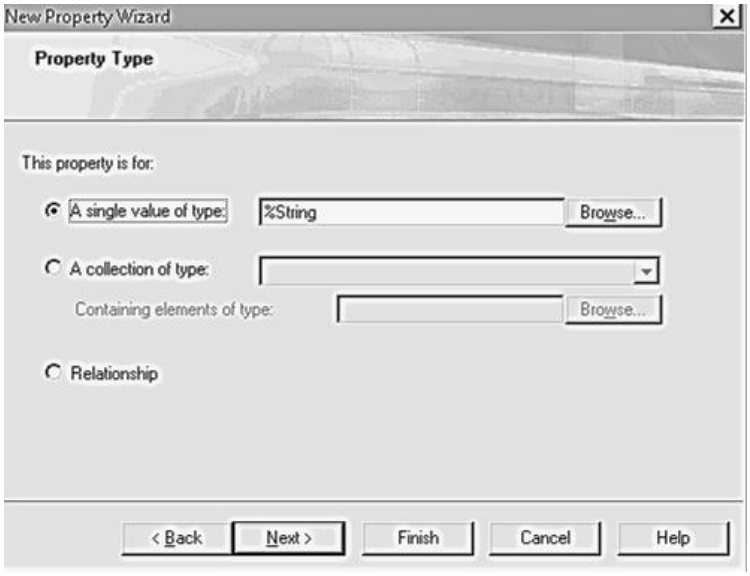


Рис. 2.12. Завдання типу властивості

На третьому етапі можна вказати характеристики властивості, а саме:

- **Required** (Обов’язкова) – властивість не повинна мати значення NULL;
- **Indexed** (Індексована) – для властивості буде створений індекс;

– **Unique** (Унікальна) – значення властивості повинне бути унікальним;
– **Calculated** (Обчислювана) – значення властивості не буде зберігатись у базі даних, а буде обчислене «на льоту».

Для прізвища варто обрати властивість обов'язковою, оскільки просто немає такої людини, у якої б не було прізвища.

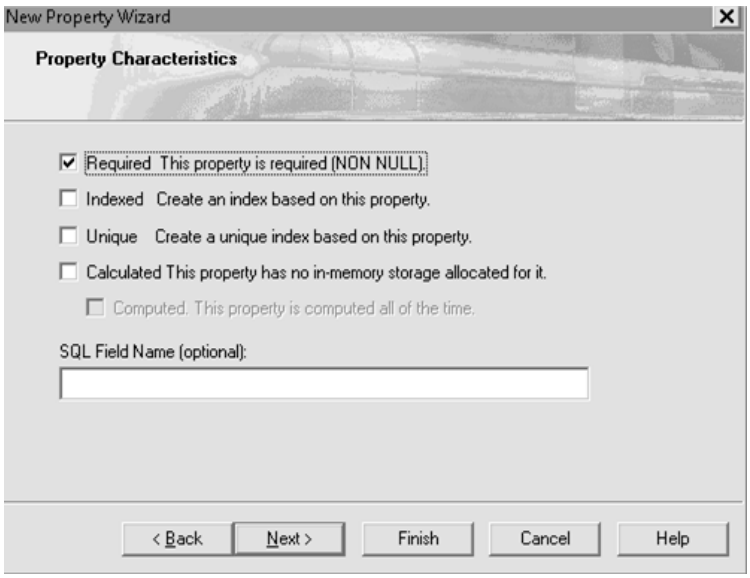


Рис. 2.13. Завдання властивості, який є обов'язковим

На наступному етапі можна вказати значення параметрів властивості. Наприклад, для типу *%String* є такі параметри, як мінімальна та максимальна довжина (MINLEN та MAXLEN).

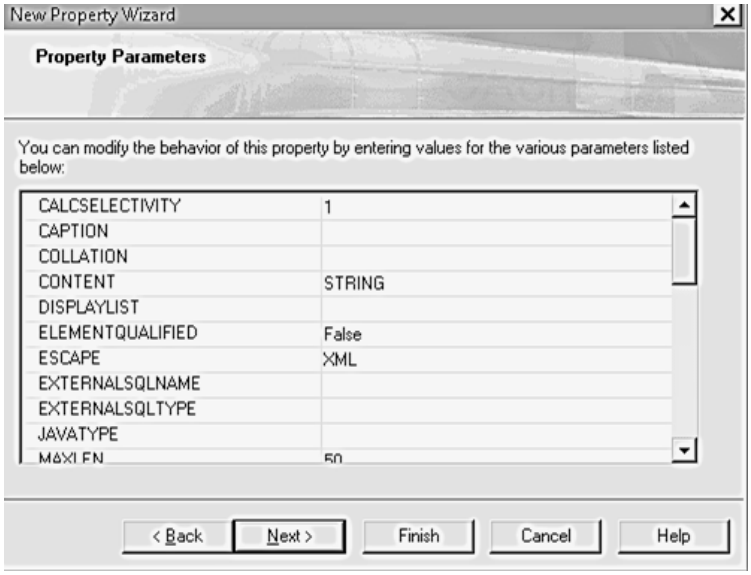


Рис. 2.14. Завдання значень параметрів властивості

3. Аналогічним чином створимо для класу Person ще 2 властивості: Name (Ім'я) типу %String, яке ми вже не будемо вказувати обов'язковим, та DOB (Date Of Birthday – дата народження) типу *%Date* (необов'язкове, а для її параметра FORMAT вкажемо значення 4).

У результаті Caché Studio згенерує такий опис класу.

```
    /// Фізичні особи
Class Test.Person Extends (%Persistent, %XML.Adaptor)
{

/// Прізвище
Property Surname As %String [ Required ];

/// Ім'я
Proerty Name As %String;

/// Дата народження
Property DOB As %Date(FORMAT = 4);

}
```

Тепер потрібно скомпілювати цей клас за допомогою меню **Build->Compile** (або просто натиснувши **Ctrl-F7**), а інакше клас просто не буде доданий у бібліотеку класів. Результат компіляції можна побачити в області виведення.

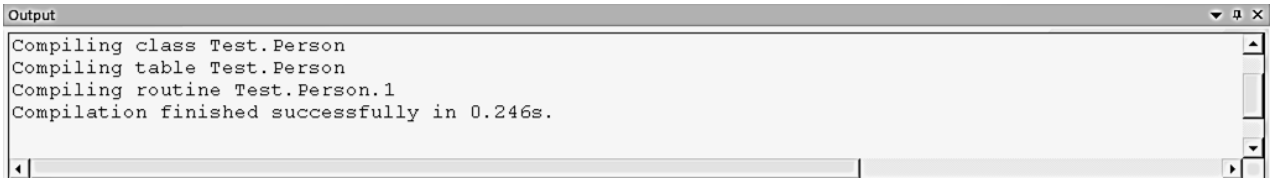


Рис. 2.15. Результат компіляції

Також після успішної компіляції в коді опису класу під блоком його оголошенням можна побачити його згенероване XML-представлення.

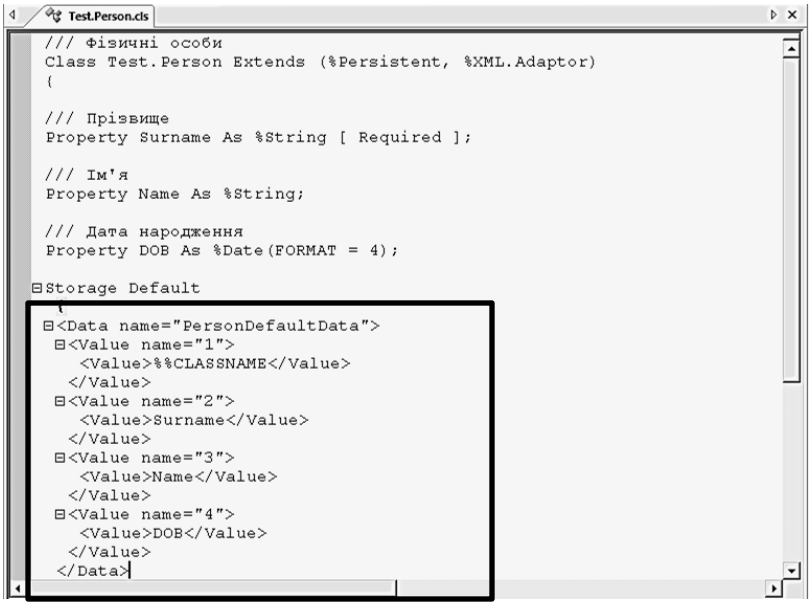


Рис. 2.16. Згенероване XML-представлення

Якщо був відкритий не один клас, а декілька, то доцільно скомпілювати одразу всі класи, що входять в проект за допомогою меню **Build->Rebuild All** (**F7**).

Наповнення даними. Створення екземплярів класу в базі даних

4. Тепер відкриємо браузер класів за допомогою меню **Tools->Class Browser**. Визначимо опцію **Show System Elements** (Показати системні елементи). Оберемо клас Test.Person та натиснемо вкладку **Methods** (Методи). Звернемо увагу на метод *%New()*, що успадковується від класу *%Library.RegisteredObject* та метод *%Save()*, який в свою чергу успадковується від класу *%Library.Persistent*.

Саме метод класу *%New()* і створює новий екземпляр класу, а метод об'єкта *%Save()* зберігає його у базі даних.

5. Для того, щоб викликати метод *%New()*, ми відкриваємо Термінал і записуємо в ньому такий вираз:

```
set p=##class(Test.Person).%New()
```

Такий вираз є виразом на мові програмування Cache Object Script. У ньому *set* – це команда, що присвоює певній змінній значення деякого виразу, *p* – ім'я локальної змінної, *##class()* – макровиклик, який надає можливість завдання класу, *Test.Person* – ім'я класу, *%New()* – метод класу. Детально основні методи маніпулювання даними та синтаксис мови COS буде розглянуто на лекціях.

А в цій лабораторній роботі ми вищезазначеним виразом створили об'єктне посилання в пам'яті, але ще не зберегли об'єкт у базі даних. Поки що ми навіть не заповнили його властивості значеннями. Зробімо це.

```
set p.Name="John"
set p.Surname="Smith"
```

Таким чином ми задали ім'я та прізвище людини. Щоб задати дату народження, ми пишемо такий вираз:

```
set p.DOB=$zdateh("11/05/80")
```

Системна функція $zdateh(date) перевіряє значення рядка і конвертує його з рядка у внутрішнє представлення (date).

Після заповнення властивостей об'єкта вже можна намагатись зберегти його у базі даних. Для цього використовується метод *%Save()*. Його можна викликати у 2 формах: `do p.%Save()` та `set sc=p.%Save()`. У першому варіанті метод виконається, але ми не зможемо побачити його результат. У другому варіанті метод виконається, і результат його виконання буде збережено в локальній змінній, яка буде мати значення 1, якщо об'єкт вдалося зберегти у БД, і 0, якщо при збереженні об'єкта виникли певні помилки і зберегти об'єкт не вдалося. Таким чином, це можна перевірити за допомогою команди `write sc`.

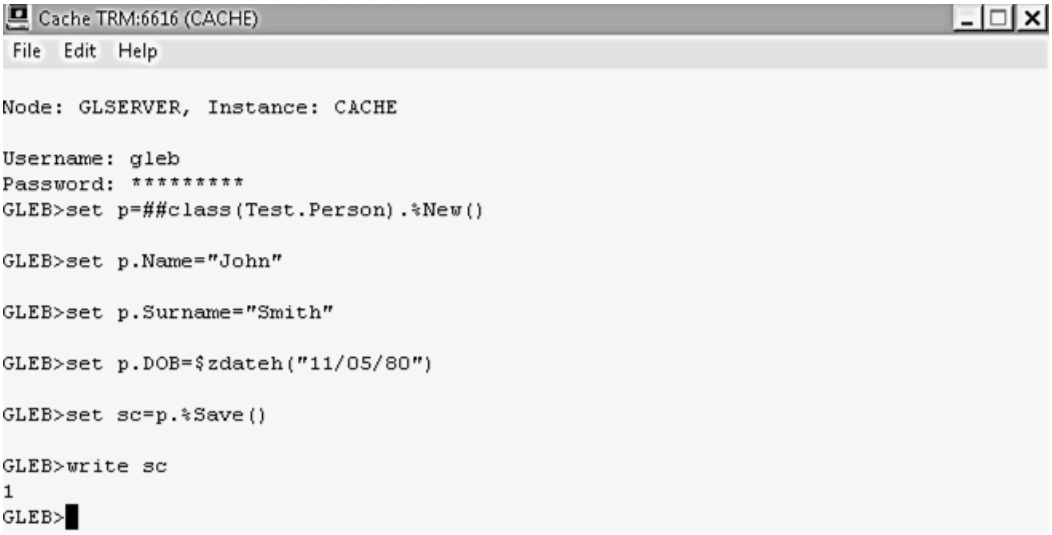


Рис. 2.17. Створення та збереження нового об’єкту класу Test.Person у терміналі

Якщо ж об’єкт чомусь не зберігся, то Cache надає змогу знайти причину помилки і розібратись в ній. Викликавши команду *do $system.OBJ.DisplayError(sc),* нам буде виведено код та пояснення помилки на екран.

Коли об’єкт уже збережено у базі даних, можна звільнити пам’ять від нього. Це можна виконати за допомогою команди *kill p.*

6. Звісно, перший час новачкам у роботі з Cache працювати з Терміналом доволі складно, оскільки він не дає жодної підказки і всі команди доводиться набирати власноруч. Тому є більш гнучкий спосіб створення об’єктів у базі даних – написання програми у Cache Studio.

Cache Studio також дозволяє створювати термінальні програми, що можуть бути використані у відлагодженні системи. Щоб створити таку програму, використовується меню **File->New->General->Cache ObjectScript Routine** або можна на робочій поверхні аналогічно до класу обрати **Routines** та **Create New Routine**.

Тепер пишемо такий код. Необхідно звернути увагу, що кожен рядок коду повинен починатись з проміжку!

```
set p=##class(Test.Person).%New()
set p.Name="Mattew"
set p.Surname="Stones"
set sc=p.%Save()
if (sc=1)
write "Object is saved successfully"
```

Тепер відкомпілюємо програму так само, як і клас, збережемо її (можна зберігати програму як в пакеті, так і окремо) і запустимо на виконання (**Debug->Go** або **Ctrl+F5**). Після цього відкриється вікно, у якому потрібно обрати програму, яку ми хочемо запустити. У вікні Output ми побачимо результат її виконання (Object is saved successfully).

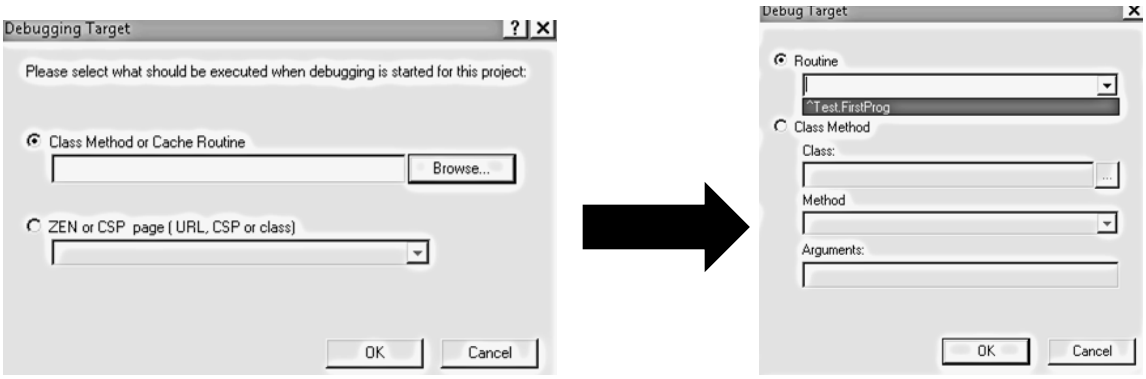


Рис. 2.18. Вибір програми для запуску

7. До речі, програми можна виконувати і за допомогою Терміналу. Для цього в ньому треба виконати команду *do ^<ім'я програми>*. У цьому випадку іменем програми є Test.FirstProg.

Рис. 2.19. Виклик програми в Терміналі

Перегляд створених даних

8. Для перегляду даних використовується Портал управління системою. У ньому заходимо на сторінку **System Explorer->SQL->Browse SQL Schemas.** Тут потрібно обрати відповідну область даних, і тоді відкриється таблиця з існуючими схемами даних у певній області.

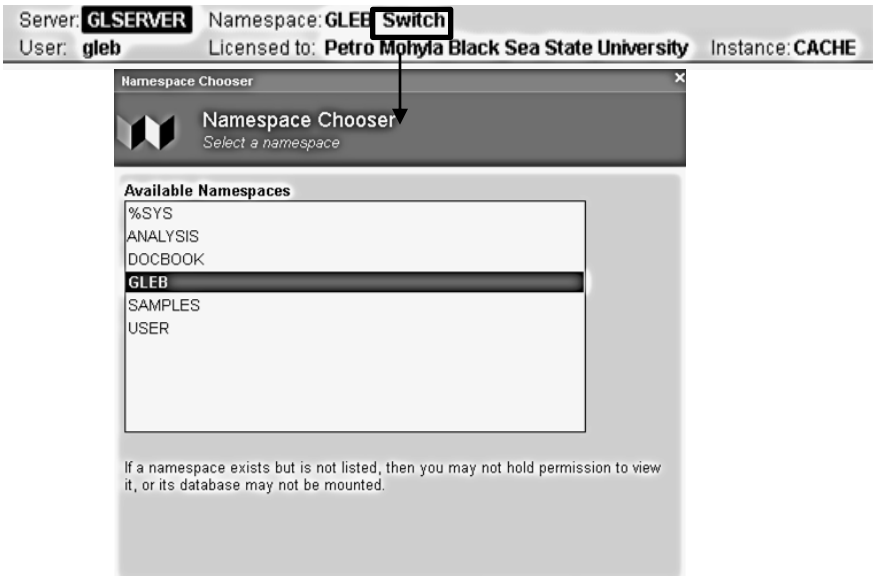


Рис. 2.20. Вибір області даних для перегляду

Клацнувши на посиланні **Tables** напроти відповідної схеми даних, можна перейти до існуючих таблиць у відповідній схемі даних.

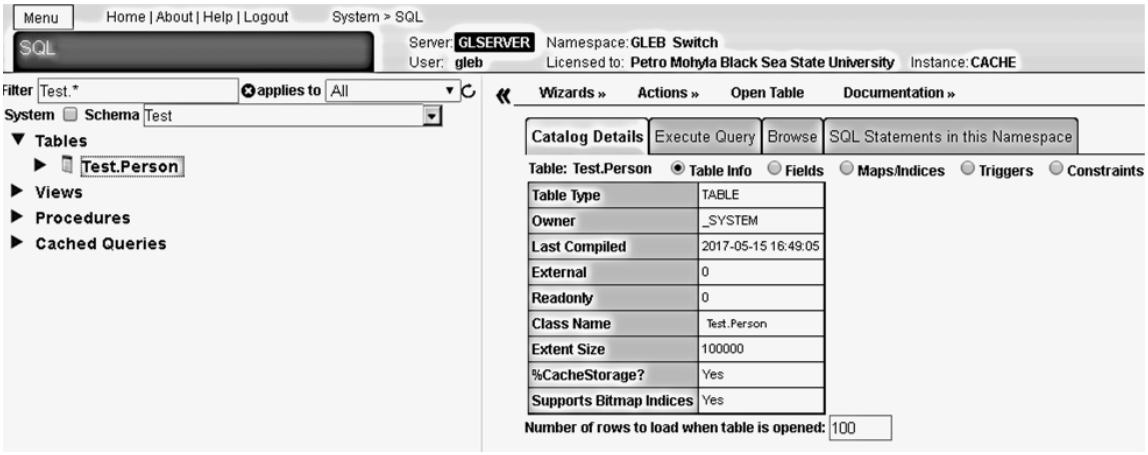


Рис. 2.21. Інформація про існуючі таблиці у відповідній схемі даних

І клацнувши на **Open Table**, можна переглянути збережені дані.

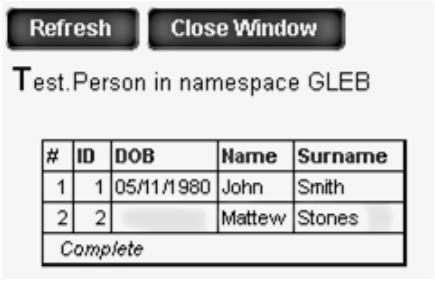


Рис. 2.22. Перегляд збережених даних

Увага! Починаючи з виконання лабораторної роботи № 2, усім студентам надано роль *%Developer*, яка дозволяє використання Cache Studio, маніпулювання даними для баз даних, які мають ресурс *%DB_%DEFAULT* (а це всі БД, створені студентами), але не дозволяє за замовчанням перегляд даних у Порталі управління системою. Студенти зможуть у Порталі зайти у свою область, але не побачать там жодних даних. Тому після створення та компіляції класів у своїх відповідних областях даних ***студентам потрібно назвати своєму викладачу свою область даних для того, щоб викладач надав їм право доступу до перегляду даних***. Це буде стосуватись і виконання подальших лабораторних робіт.

Інші способи маніпулювання даними
Майстер створення Web-форм для CSP

1. Технологія CSP (Caché Server Pages) основний інструмент створення Web-інтерфейсу для інформаційних застосунків, написаних на Caché. Технологія CSP пропонує витончені засоби створення швидкодіючих, добре масштабованих Web-застосунків за короткий час. Вона також спрощує подальший супровід і розвиток таких застосунків.

CSP-сторінки зберігаються в CSP-файлах. При зверненні до CSP-файлу відбувається його трансляція в клас CSP, який потім компілюється за допомогою компілятора Caché Server Pages.

Caché Studio дозволяє створювати сторінки CSP. Для цього треба скористатись меню **File->New->CSP File->Cache Server Page** або в робочій області створити CSP аналогічно до класу та програмі.

Після створення CSP у області редагування Ви зможете побачити шаблон коду Web-сторінки, де зможете знайти відомі Вам з 1 курсу теги <html>,<head>,<title>,<body>.

CSP можна створювати вручну (чим ми і займемось трохи пізніше), а можна скористатись Майстром Web-форм (**Web Form Wizard**). Для цього заходимо у меню **Tools->Templates->Web Form Wizard** і запускаємо його.

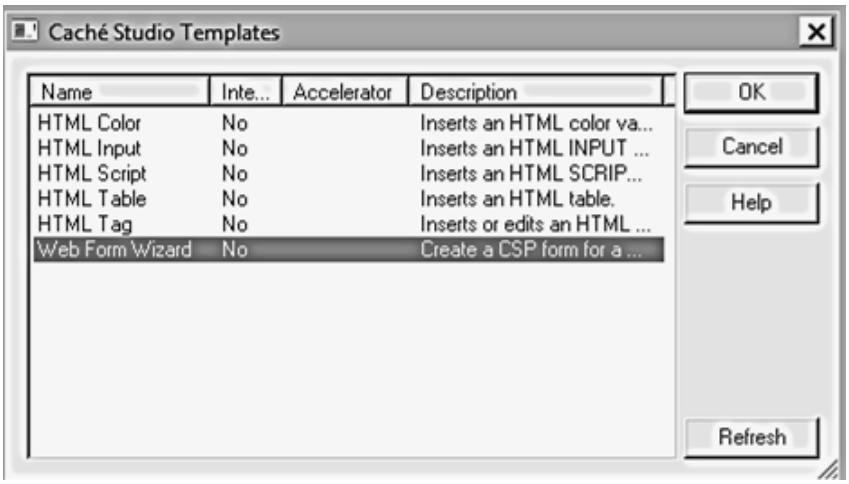


Рис. 2.23. Вибір шаблону Caché Studio

На першому етапі ми обираємо відповідний клас.

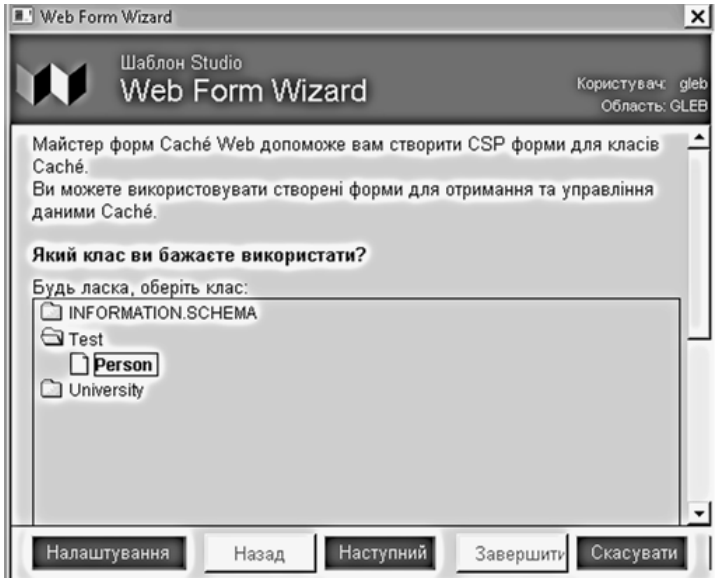


Рис. 2.24. Вибір класу в майстрі Web-форм

На другому етапі обираємо властивості класу для відображення у Web-формі.

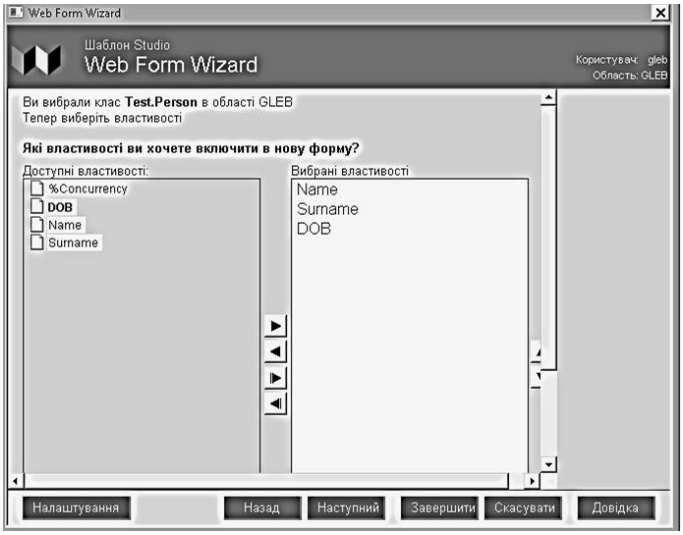


Рис. 2.25. Вибір властивостей класу для відображення на Web-формі

На третьому і останньому етапі можна вказати заголовки до відповідних полів.

Рис. 2.26. Заголовки для відповідних властвостей класів на Web-формі

Після цього, відкомпілювавши Web-сторінку (так само як клас і програму) і натиснувши на робочій області біля відповідної CSP **Show Web Page,** ми отримаємо таку Web-сторінку. При натисненні кнопки **Save** на ній можна також зберегти дані у базі даних.

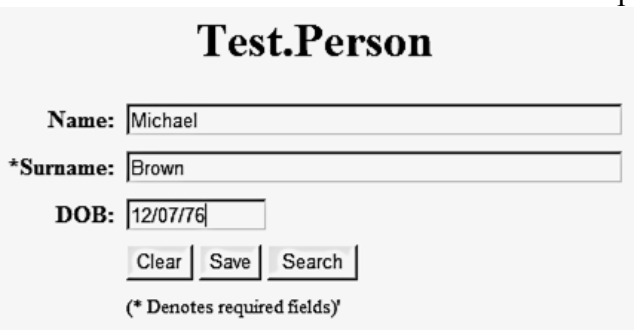


Рис. 2.27. CSP-сторінка, сформована за відповідним шаблоном

Доступ до даних, використовуючи ODBC

Cache, як і будь-яка СКБД, що себе поважає, підтримує ODBC. Під час встановлення сервера або клієнта Cache на локальному комп'ютері автоматично встановиться і ODBC-драйвер для Cache.

Щоб налаштувати ODBC для Cache у Windows 7, потрібно зайти в панель управління та піти таким шляхом: **Control Panel\System and Security\Administrative Tools**. Потім обираємо **Data Sources (ODBC)**.

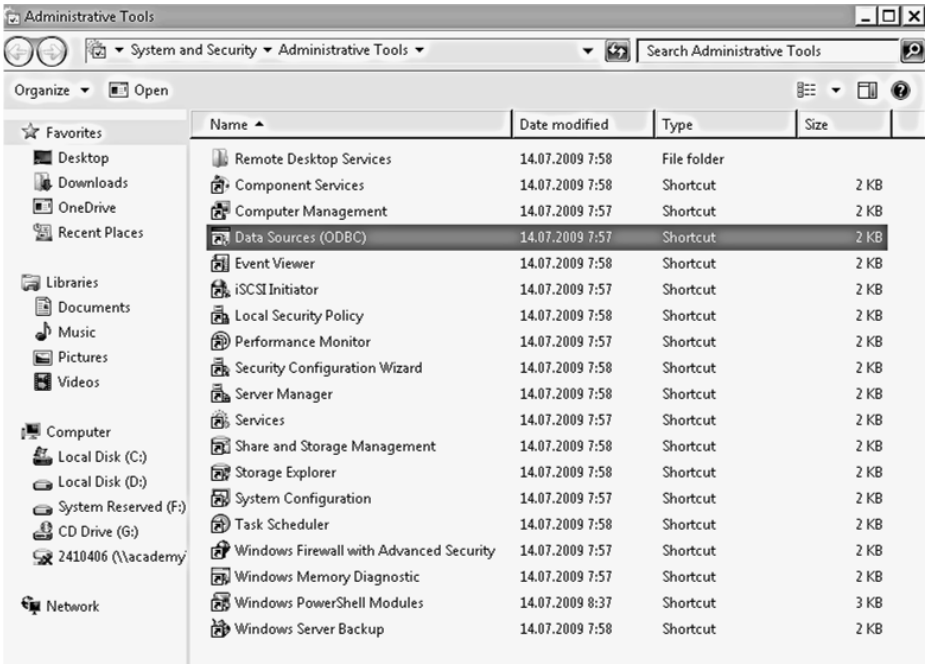


Рис. 2.28. Налаштування ODBC у Windows

Відкриваємо відповідне оснащення і додаємо драйвер Intersystems ODBC 35.

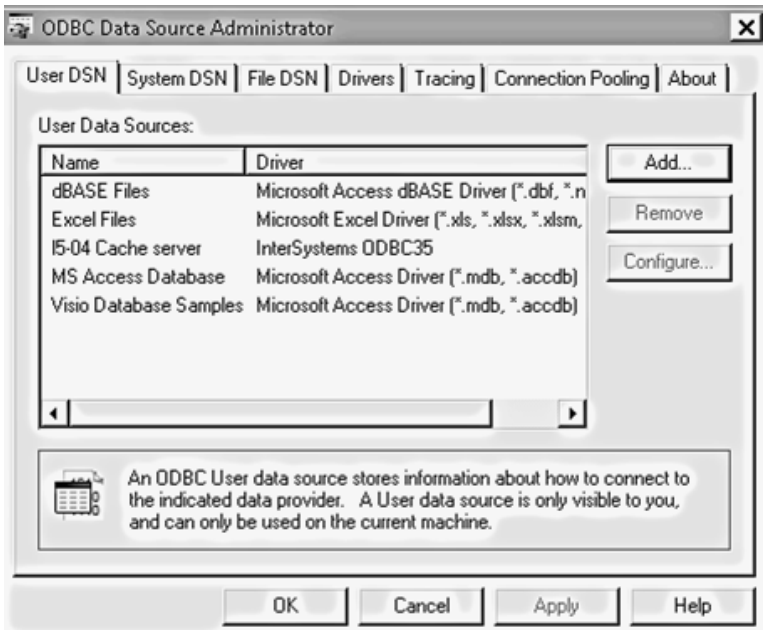


Рис. 2.29. Список існуючих драйверів ODBC

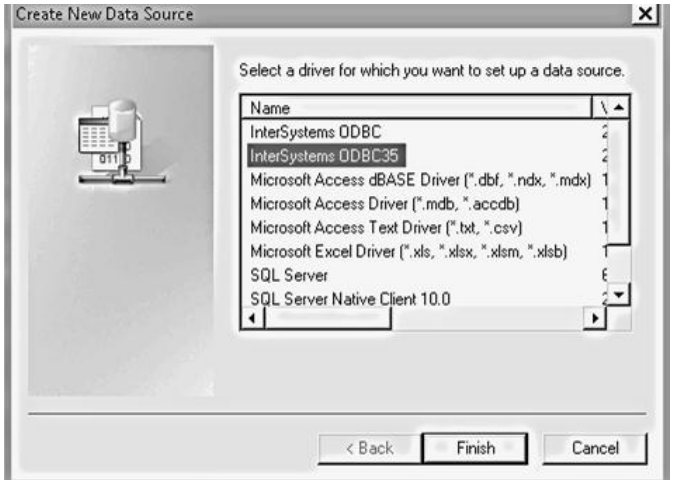


Рис. 2.30. Додавання драйвера Intersystems ODBC 35

Далі налаштовуємо підключення до сервера. Для цього вводимо назву джерела даних (воно може бути довільним), вказуємо IP-адресу сервера (GLSERVER – 192.168.96.179), номер порту – 1972, відповідну область даних, ім’я користувача та пароль.

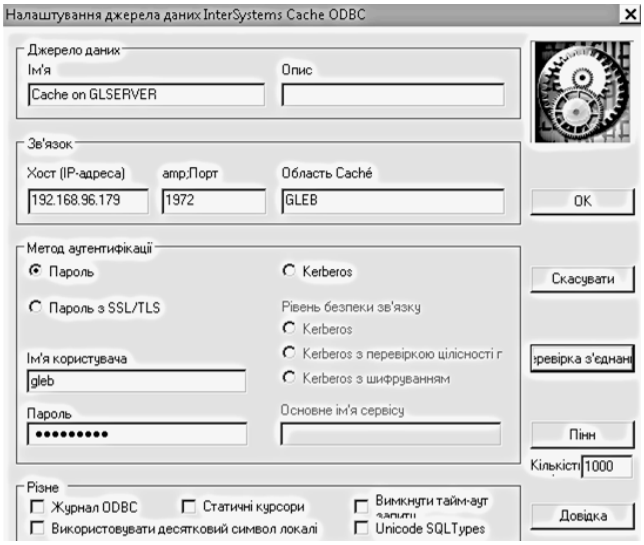


Рис. 2.31. Налаштування підключення до сервера Cache

Потім намагаємось під’єднатись до сервера (**Test Connection**).

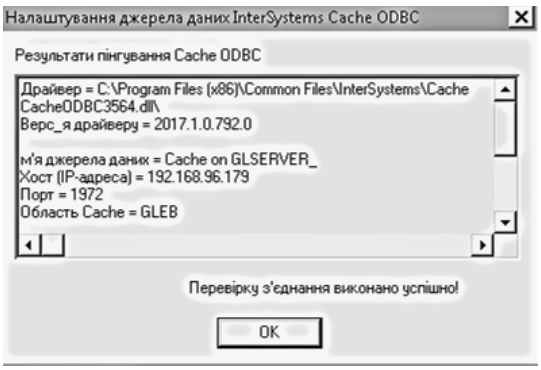


Рис. 2.32. Перевірка з’єднання з СКБД Cache через ODBC

Далі створюємо нову базу даних MS Access. У новій базі даних за допомогою меню **External Data->Import & Link->ODBC Database** створюємо зв'язану таблицю з джерелом даних Cache. Для цього обираємо **Link the data source by creating a linked table**. Обравши відповідний драйвер ODBC, ми отримаємо таблицю, додавши записи у якій, ми додаємо нові дані у БД на сервері Cache.

Table1 | Test_Person

ID	DOB	Name	Surname
1		Ivan	Ivanov
2	05.11.1980	John	Smith
3	03.12.1989	Matthew	Stones
4		Steven	Riley
5		Paul	Berger
6	12.07.1976	Michael	Brown
7	16.02.1982	Eric	Bradley
	22.10.1973	Jeffrey	Morgan

Рис. 2.33. Відображення даних СКБД Cache у СКБД MS Access

Після цього переглядаємо дані у Порталі управління системою на сервері Cache.

Refresh | Close Window

Test.Person in namespace GLEB

#	ID	DOB	Name	Surname
1	1		Ivan	Ivanov
2	2	05/11/1980	John	Smith
3	3	03/12/1989	Matthew	Stones
4	4		Steven	Riley
5	5		Paul	Berger
6	6	12/07/1976	Michael	Brown
7	7	16/02/1982	Eric	Bradley
8	8	22/10/1973	Jeffrey	Morgan

Complete

Рис. 2.34. Перегляд змінених даних у Порталі управління системою на сервері Cache

Завдання

Виконати всі розглянуті вище етапи роботи.

Контрольні питання

1. Які моделі даних підтримує СКБД Cache?
2. Для чого призначена Cache Studio?
3. Для чого призначений Термінал?
4. Для чого призначений Портал управління системою?
5. Як створити екземпляр класу у базі даних?
6. Як присвоїти значення властивостям класу?
7. Як зберегти об'єкт у базі даних (2 варіанти)?
8. Як подивитись розшифровування помилки?
9. Що таке CSP?

ЛАБОРАТОРНА РОБОТА № 3

«ВЛАСТИВОСТІ КЛАСІВ CACHE. ПРОЕКТУВАННЯ КОНЦЕПТУАЛЬНОЇ МОДЕЛІ ПРЕДМЕТНОЇ ОБЛАСТІ. ЕКСПОРТ ТА ІМПОРТ ПРОЕКТУ»

Теоретичні відомості

Під час проходження курсу пропонується створити систему, що автоматизує діяльність у сфері певної предметної області (згідно з варіантом завдання). Для початку потрібно описати постановку задачі та концептуальну модель даних.

Для прикладу візьмемо предметну область університету. У базі даних університету повинна зберігатись інформація про факультети, спеціальності, викладачів; кафедри, на яких працюють викладачі; посади викладачів, студентів та груп, до яких вони належать.

Пропонується наступна концептуальна модель предметної області.

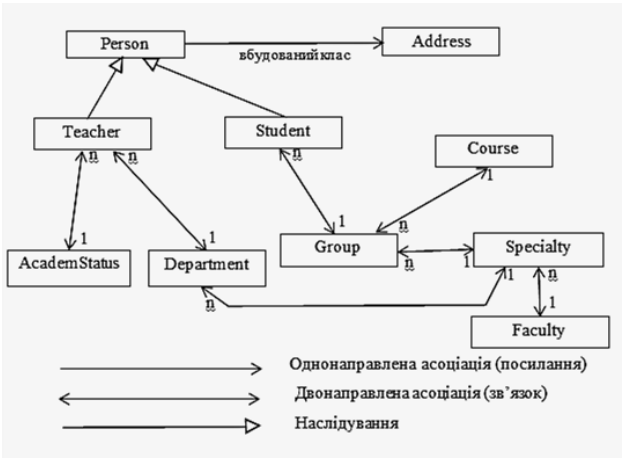


Рис. 3.1. Концептуальна модель предметної області університету

Після виявлення сутностей варто подумати про атрибути, які вони будуть мати, а також про тип атрибутів та можливі обмеження, що будуть накладені на атрибути (їх можна представити за допомогою ключових слів).

Перелік атрибутів сутностей наведено в табл. 3.1.

Таблиця 3.1.

Перелік атрибутів сутностей концептуальної моделі предметної області університету

Ідентифікатор	*Назва*	*Тип*	*Примітка*
University.Person	**Людина**	**Абстрактний клас**	**Abstract, ClassType="persistent"**
Surname	Прізвище	*%String*	Required
Name	Ім'я	*%String*	Required
MidName	По батькові	*%String*	Required
DOB	Дата народження	*%Date*	FORMAT = 5, MAXVAL = +$h
PhoneNumber	Телефон	*%String*	MAXLEN = 12 PATTERN = 3n1"-"3n1"-"4n
University.Teacher	**Викладач**	**Наслідник класу University.Person**	
Degree	Науковий ступінь	*%String*	
Rate	Доля ставки	*%Float*	

Продовження таблиці 3.1.

University.Student	**Студент**	**Наслідник класу University.Person**	
GradebookNum	Номер залікової книжки	*%Integer*	Unique
University.Address	**Адреса**	**Вбудований клас**	
City	Місто	*%String*	
Street	Вулиця	*%String*	
HouseNum	Номер будинку	*%Integer*	
Index	Індекс будинку (якщо він є)	*%String*	
FloatNum	Номер квартири	*%Integer*	
PostalCode	Поштовий індекс	*%String*	PATTERN = 5n.1(1"-"4n)
University.Academstatus	**Посада**	**Клас, що зберігається**	
NameStat	Назва посади	*%String*	Required
Norm	Норма часу	*%Integer*	
Salary	Заробітна плата	*%Float*	
University.Department	**Кафедра**	**Клас, що зберігається**	
CodeDepart	Код кафедри	*%Integer*	Required, Unique
NameDepart	Назва кафедри	*%String*	Required
AudNumber	Номер аудиторії	*%String*	
University.Course	**Курс**	**Клас, що зберігається**	
CourseNum	Номер курсу	*%Integer*	Required
CourseName	Назва курсу (словами)	*%String*	
University.Group	**Група**	**Клас, що зберігається**	
GroupNumber	Номер групи	*%Integer*	Required
University.Specialty	**Спеціальність**	**Клас, що зберігається**	
CodeSpec	Код спеціальності	*%Integer*	Required
NameSpec	Назва спеціальності	*%String*	
University.Faculty	**Факультет**	**Клас, що зберігається**	
FacultyName	Назва факультету	*%String*	Required

Використовуючи вищезазначену таблицю, можна побудувати відповідну діаграму класів, використовуючи Rational Rose, StarUML або ін.

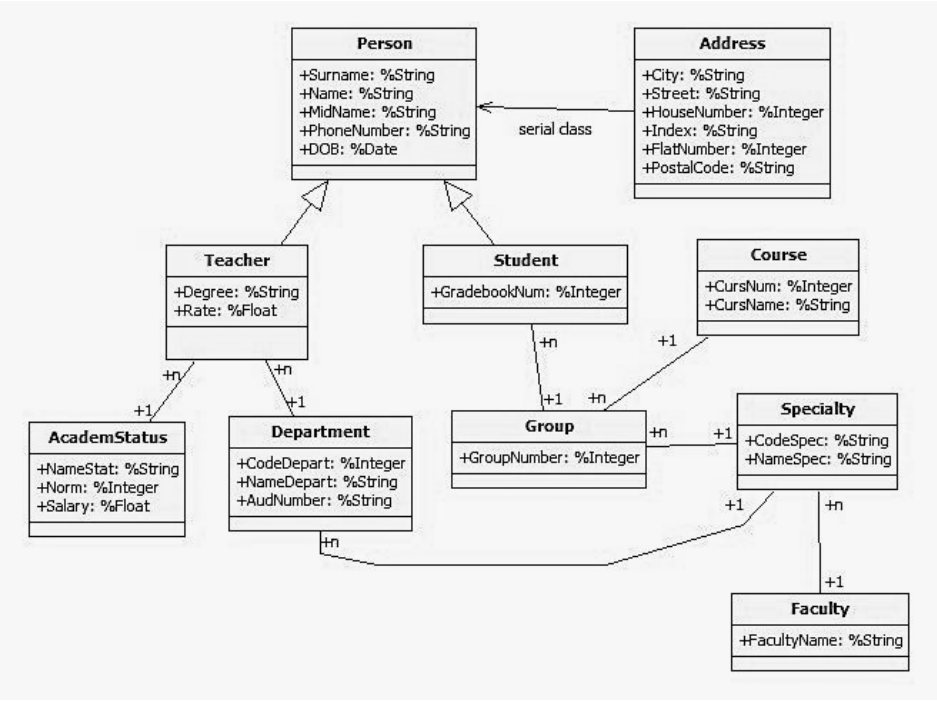


Рис. 3.2. Діаграма класів

Варто зазначити, що ця діаграма не є повною. У ній поки не вказані властивості посилання та зв'язків та методи, які будуть додані в наступних лабораторних роботах.

Основні методи роботи з об'єктами в СКБД Caché

Кожний клас, що зберігається, наслідується від системного класу *%Persistent*, від якого отримує наступні методи зовнішнього інтерфейсу:

1. `%New()` – конструктор об'єкта. Його завдання – створити екземпляр класу і присвоїти йому OREF;
2. `%Save()` – зберігає екземпляр класу на диску і привласнює йому;
3. `%OID()` – повертає OID екземпляра класу;
4. `%Close()` – у старих версіях зменшував значення лічильника OREF на одиницю і знищував версію об'єкта в пам'яті. Починаючи з версії 5.0, цей метод нічого не робить.
5. `%Open()` – метод класу. У якості першого аргументу отримує OID. Якщо він знаходить об'єкт, існуючий в базі даних, то створює в пам'яті його копію, яка містить значення всіх властивостей, і повертає об'єкт. Якщо об'єкт вже завантажений в пам'ять, просто повертається OREF. Узагалі у методу є три аргументи.
6. `%OpenId()` – відрізняється від попереднього тим, що в якості аргументу одержує не OID, а ID;
7. `%Delete()` – видаляє версію об'єкта, що зберігається на диску, копія в пам'яті при цьому залишається. У якості єдиного параметра отримує OID, цей ідентифікатор більше не використовується згодом (це стосується класів із внутрішньою системою зберігання Cache, в іншому випадку відповідальність за повторне використання OID-ів повністю лягає на плечі розробника).
8. `%DeleteId()` – відрізняється від попереднього тим, що в якості аргументу одержує не OID, а ID.
9. `%IsModified()` – повертає «істинно» (1), якщо значення властивостей об'єкта були змінені, в іншому випадку – 0.

Тепер розглянемо кожен з методів більш детально.

Створення нових об'єктів

Основний синтаксис для створення нових екземплярів об'єктів – це використання методу класу *%New()*:

```
Set oref = ##class(Classname).%New()
```

де oref – посилання OREF на новий об'єкт і Classname – це ім'я класу, чутливе до регістру літер і може містити ім'я пакета.

##class() – це макровиклик, який використовується для виклику методів класу.

Наприклад, для створення нового об'єкта Person використовується наступний синтаксис:

```
Set person = ##class(MyApp.Person).%New()
```

Методом *%New()* можна передати необов'язковий параметр. Якщо він присутній, цей аргумент передається методу-тригеру *%OnNew()* об'єкта. Використання цього аргументу залежить від реалізації конкретного класу.

Відкриття об'єктів

Для доступу до екземпляру об'єкта можна використовувати такі посилання:

1. OID – ідентифікатор екземпляру об'єкта в базі даних
2. ID – ідентифікатор екземпляру всередині класу в базі даних
3. OREF – посилання на екземпляр об'єкта в оперативній пам'яті.

Можна відкрити існуючий об'єкт, що зберігається, за допомогою його ідентифікатора ID, використовуючи наступний синтаксис:

```
Set oref = ##class(Classname).%OpenId(ID)
```

де oref – це змінна, що містить OREF об'єкта, який ми намагаємось відкрити, Classname – це ім'я класу, ID – це ідентифікатор об'єкта. Classname чутливий до регістру букв.

Наприклад, щоб відкрити об'єкт University.Student з ID 22:

```
Set stud = ##class(University.Student).%OpenId(22)
Write stud.Name,!
```

де stud – це нове об'єктне посилання OREF на об'єкт класу Student.

Можна також відкрити об'єкт, використовуючи його повний OID:

```
Set oref = ##class(Classname).%Open(oid)
```

де oref містить посилання OREF, Classname ім'я класу, oid – це повний OID об'єкта.

Повністю заданий OID складається з 2 компонентів: ID-частини, що однозначно ідентифікує екземпляр класу, і імені класу. Більш детально його формування буде розглянуте у наступних лекціях.

Зазвичай застосунки використовують метод *%OpenId()*.

Маючи відкритий об'єкт можна переглядати або змінювати його властивості, використовуючи методи, що описані нижче.

Для визначення повного ідентифікатора об'єкта (OID), що завантажений в пам'ять, служить метод *%Oid()*. Аналогічно можна скористуватись методуми *%Id() і %ClassName()* для отримання ID-частини і імені класу. Таким чином стає відомим і точний клас відкритого об'єкта:

```
Set stud = ##class(University.Student).%OpenId(22)
Set oid = person.%Oid()
Write stud.%Id()
Write stud.%ClassName()
```

Якщо перед відкриттям об'єкта з заданим OID та ID ми захочемо перевірити, чи існує він, можна скористуватись методуми *%Exists()* або *%ExistsId()*.

Іноді буває корсно знати, чи піддавався об'єкт змінам. Дізнатися про це можна за допомогою метода *%IsModified()*.

Доступ до значень властивостей здійснюється за допомогою точкового синтаксису:

```
Write stud.Name

Set stud.Name = "Кузнєцов"
```

Збереження об'єктів

Для збереження об'єкта використовується метод *%Save()* екземпляра:

```
Do oref.%Save()
```

де oref – посилання oref об'єкту, що підлягає збереженню.

Метод *%Save* повертає значення, що дозволяє перевірити успішність виконання методу.

```
Set sc = oref.%Save()
```

де sc – значення, що повертається методом *%Save, oref* – посилання на об'єкт, що підлягає збереженню.

У змінну sc буде записано 1, якщо метод відпрацював успішно, і 0 якщо невдало.

Якщо метод відпрацював невдало, то виклик методу DisplayError() з бібліотеки Caché $system.OBJ роздрукує текст повідомлення про помилку.

Наступний код виконує збереження об'єкта Person з посиланням OREF рівній per і в разі помилки виводить повідомлення:

```
Set sc = per.%Save()
If sc'=1
{
Do $System.OBJ.DisplayError(sc)
}
```

Видалення об'єктів

Можна видалити або один об'єкт, або об'єкти одного екстенту. Під екстентом розуміється набір екземплярів класу і всіх його підкласів.

Для видалення об'єкта з бази даних використовуються 2 методи *%Delete()* і *%DeleteId().*

Перший метод використовує як аргумент повний OID екземпляра, а другий – ID екземпляра. Обидва методи видають інформацію про те, як відпрацював метод.

Синтаксис методу *%Delete()*:

```
Do ##class(Classname).%Delete(oid)
Set sc = ##class(Classname).%Delete(oid)
```

Синтаксис методу *% DeleteId()*:

```
Do ##class(Classname).%DeleteId(id)
Set sc = ##class(Classname).%DeleteId(id)
```

де classname – це ім'я класу;
oid – OID об'єкта видалення;
id – ID екземпляра.

У змінній sc – статус методу, інформація про те, як він відпрацював.

Наприклад, для видалення об'єкта класу Student використовується код:

```
Set sc = ##class(University.Student).%Delete (oid)
```

де oid – це OID об'єкта класу Student, який потрібно видалити.

Для видалення об'єкта класу Student з ID = 24 використовується код:

```
Set sc = ##class(University.Student).%DeleteId(24)
```

Можна видалити всі екземпляри класу і його підкласів, використовуючи метод *%DeleteExtent()*, який видає інформацію про те, як виконалось видалення.

```
Do ##class(Classname).%DeleteExtent()
Set sc = ##class(Classname).%DeleteExtent()
```

де Classname – це ім'я кореневого класу екстента для видалення;
sc – це змінна, яка містить код помилки.

Цей метод видаляє всі екземпляри певного класу, так само як і всі екземпляри його підкласів.

Наприклад, для видалення всіх об'єктів класу Student, включаючи підкласи використовується такий синтаксис:

```
Do ##class(University.Student).%DeleteExtent ()
```

Метод *%DeleteExtent* буде виконувати будь-які методи *%OnDelete*, якщо вони присутні для кожного об'єкта, що видаляється.

Можна також видалити всі екземпляри класу і його підкласів, використовуючи більш швидкий, але й більш небезпечний метод *%KillExtent()*:

```
Do ##class(Classname).%KillExtent()
```

Метод *%KillExtent* негайно видаляє всі дані, асоційовані з екстентом класу, жоден відкат не виконується, і жодні посилання не перевіряються. Його варто використовувати під час розробки систем, що не містять реальних даних.

Експорт та імпорт проекту та окремих класів у XML

Напевно, у кожного з вас вже виникало питання як скопіювати створені вами класи та дані для того, щоб можна було працювати з ними вдома і взагалі чи можна це зробити. Звісно, СКБД Cache дозволяє зробити і це.

Найкращим способом імпорту класів є імпорт цілого проекту повністю. Для цього у Студії йдемо у меню **Tools->Export** і після цього ми побачимо вікно вибору проекту, який ми хочемо експортувати.

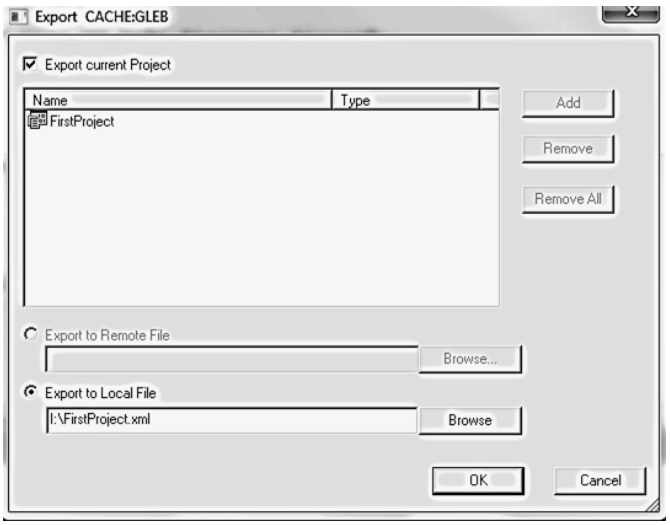


Рис. 3.3. Експорт класів

Щоб зробити обернену дію (імпортувати проект з XML-файлу у вашу область даних на сервері Cache), ми йдемо у меню **Tools->Import Local**, після чого відкриється знайоме вам вікно вибору файлу, у якому потрібно обрати файл XML з експортованим проектом.

Також можна експортувати та імпортувати окремі класи та програми. Для цього у Порталі керування системою йдемо шляхом **System Explorer->Classes**, обираємо свою предметну область та після цього ми побачимо таку сторінку, на якій можна обрати класи, які хочемо експортувати, та після цього натиснути **Export**.

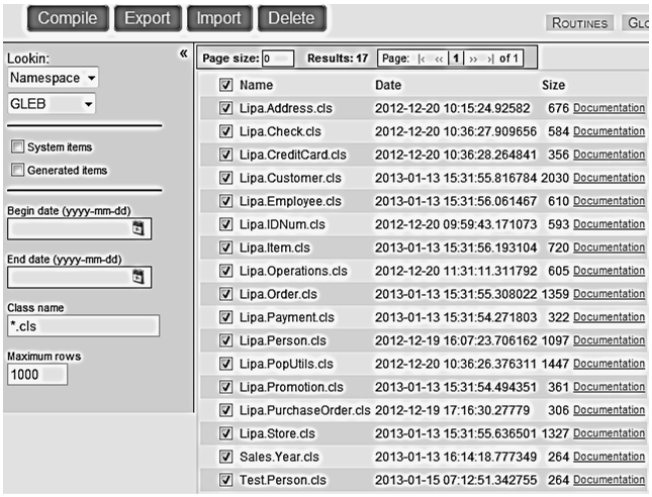


Рис. 3.4. Експорт класів

Завдання

1. Згідно зі своїм варіантом (файл «Варіанти завдань») спроектувати концептуальну модель вашої предметної області.
2. Продумати прості властивості класів вашої предметної області та результати навести в таблицю.
3. Навести діаграму класів вашої предметної області (її зберегти окремо, у наступних лабораторних роботах її вигляд буде змінюватись).
4. За можливістю створити в Студії відповідні класи, додати до них відповідні властивості та скомпілювати їх.
5. Створити об’єктним шляхом мінімум 5 об’єктів кожного класу, задати значення їх властивостей

Контрольні питання

1. Назвіть основні складові об’єктної моделі даних згідно стандарту ODMG.
2. Назвіть основні типи класів об’єктної моделі Cache.
3. Які елементи мають класи в СКБД Cache?
4. Які типи властивостей у СКБД Cache ви знаєте?
5. Що таке параметри класів та властивостей?
6. Як додати новий об’єкт у СКБД Cache?
7. Як присвоїти значення властивостей об’єктів?
8. Як зберегти об’єкт у базі даних?
9. Як видалити об’єкт у базі даних?
10. Як видалити всі об’єкти класу у базі даних?
11. Викладач може поставити питання згідно з вашою предметною областю.

ЛАБОРАТОРНА РОБОТА № 4

«ЗАБЕЗПЕЧЕННЯ ЦІЛІСНОСТІ БАЗИ ДАНИХ У CACHE. СТВОРЕННЯ ЗВ'ЯЗКІВ МІЖ КЛАСАМИ»

Створення зв'язку засобами Cache Studio

У нашій діаграмі класів зв'язки вже визначені. Спробуємо створити перший з них, наприклад, між студентом (University.Student) та групою (University.Group). Нагадаємо, що цей зв'язок буде між 1 об'єктом типу Group з одного боку та багатьма об'єктами типу Students з іншого. Таким чином типом зв'язку буде «One-To-Many».

Щоб створити новий зв'язок у Студії потрібно для певного класу створити новий атрибут, дати йому ім'я, і на наступному етапі, де задається тип, вказати пункт **Relationship** і після цього натиснути **Next**.

Створімо для класу Order властивість і назвемо її Customer.

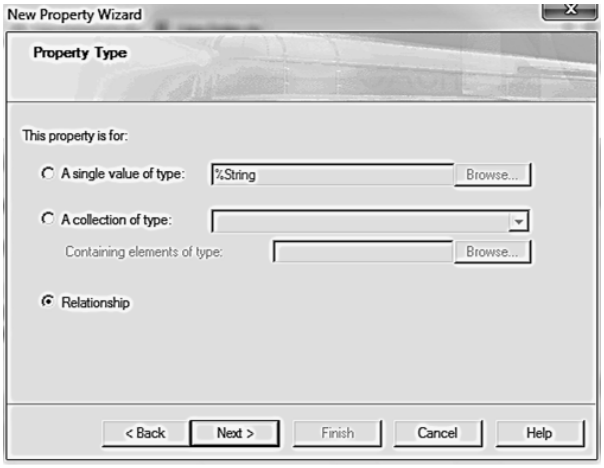


Рис. 4.1. Створення зв'язку як нової властивості класу

На наступному етапі з'явиться вікно, в якому треба вказати тип зв'язку (**The relationship property refers to…**)

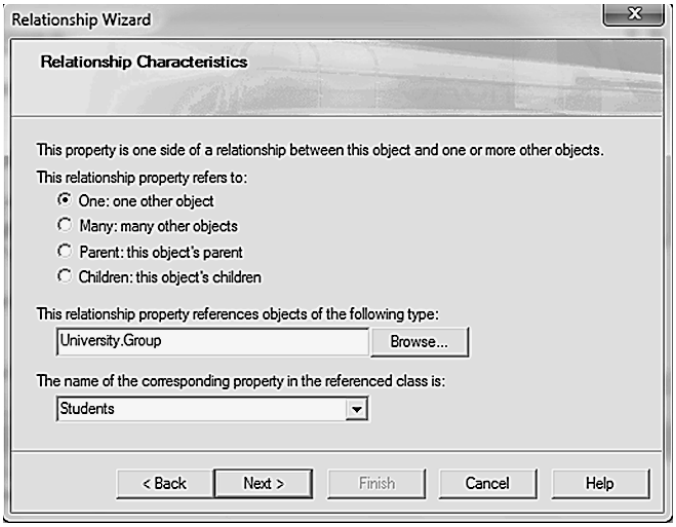


Рис. 4.2. Завдання кардинальності зв'язку

Крім типу зв'язку, треба вказати клас, на об'єкти якого буде вказувати зв'язок (**The relationship property referenced objects of the following type**) та ім'я відповідної властивості зв'язку на іншій стороні класу (**The name of the corresponding property in the referenced class is**).

Під «іншою стороною» тут розуміється клас Group, у якому буде створена властивість Customers і тип зв'язку буде Many. Щоб вказати властивість у зв'язаному класі, можна обрати її зі списку існуючих властивостей у комбобоксі або вручну задати її ім'я (саме так і було зроблено). В останньому випадку така властивість буде створена, про що і буде повідомлено системою на наступному етапі.

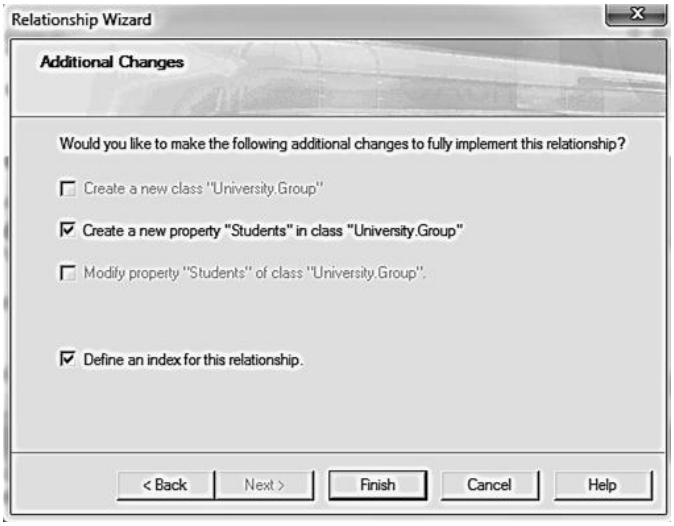


Рис. 4.3. Створення нової властивості зв'язку на зворотному боці зв'язку

Також на цьому етапі вказується, чи створювати для зв'язку індекс.

Зв'язок типу «relationship» під часу опису класу визначається відповідно до наступного синтаксису:

```
Relationship <ім'я атрибута> As <пов'язаний клас> [Inverse = <зворотній
зв'язок>, Cardinality = One|Many|Parent|Children, <інші ключова слова>]
```

де <пов'язаний клас> – ім'я класу, з яким встановлюється зв'язок. Цей клас повинен бути класом, що зберігається;

<зворотній зв'язок> – ім'я атрибута, що забезпечує зв'язок «з іншої сторони»;

<інші ключові слова> – інші ключова слова, що використовуються при декларації атрибутів, такі, як Description, Final, Required, Private і т. д. Вони не є обов'язковими.

Пара Inverse = <зворотній зв'язок> повинна представити ім'я атрибута з <пов'язаний клас>, тобто атрибут, який забезпечує «зворотній» зв'язок.

Значення ключового слова Cardinality визначає: по-перше, яка потужність атрибута, що забезпечує зв'язок «на цій стороні», і, по-друге, чи є зв'язок незалежним або залежним.

Покажемо зв'язки між класами на прикладі класів Student (Студент) та Group (Група). Нехай обидва класи знаходяться в пакеті University (Університет). Ці класи з'єднуються наступним природним зв'язком: студент входить тільки в одну групу, відповідно група складається більш ніж з одного студента. Тому клас Group у зв'язку

знаходиться на стороні «один», а клас – Student – на стороні «багато». Розглянутий зв'язок у Caché буде описаний таким чином:

```
Class Unversity.Student Extends %Persistent
{
    Relationship Group As University.Group [Cardinality = one, Inverse =
Students];
    …
}

Class Unversity.Group Extends %Persistent
{
    Relationship Students As University.Student [Cardinality = many,
Inverse = Group];
    …
}
```

Таким чином наш зв'язок буде представлений 2 властивостями на різних кінцях зв'язку. Клієнт може створити безліч замовлень, а замовлення буде належати до певного клієнта.

На прикладі створення цього зв'язку створімо і інші зв'язки між класами, що вказані у нашій діаграмі класів. Результати опишемо таблицею.

Таблиця 4.1.

Перелік атрибутів, що представляють собою зв'язки між класами

№	Ім'я властивості	Опис властивості	Клас, на який вказує властивість	Тип зв'язку	Відповідна властивість у зв'язаному класі
Клас University.Person					
1	Addr	Домашня адреса	University.Address	вбудований об'єкт	–
Клас University.Teacher					
2	Stat	Посада, яку займає викладач	University.AcademStatus	One	Teachers
3	Depart	Кафедра, на якій працює викладач	University.Department	One	Teachers
Клас University.Student					
4	Group	Група, до якої входить студент	University.Group	One	Students
Клас University.Group					
4	Students	Студенти, які входять у групу	University.Student	Many	Group
5	Course	Курс, до якого відноситься група	University.Course	One	Groups
6	Spec	Спеціальність, до якої відноситься група	University.Specialty	One	Groups
Клас University.Course					
5	Groups	Групи даного курсу	University.Group	Many	Course
Клас University.AcademStatus					
2	Teachers	Викладачі, які займають дану посаду	University.Teacher	Many	Stat

Продовження таблиці 4.1.

Клас University.Department					
3	Teachers	Викладачі, які працюють на даній кафедрі	University.Teacher	Many	Depart
7	Spec	Спеціальність, до якої відноситься дана кафедра	University.Specialty	One	Departs
Клас University.Specialty					
6	Groups	Групи даної спеціальності	University.Group	Many	Spec
7	Departs	Кафедри, які входять до даної спеціальності	University.Department	Many	Spec
8	Faculty	Факультет, до якого відноситься спеціальність	University.Faculty	One	Specialties
Клас University.Faculty					
8	Specialties	Спеціальності, що входять до даного факультету	University.Specialty	Many	Faculty

Таким чином, після опису зв’язків між класами наша діаграма класів буде мати такий вигляд:

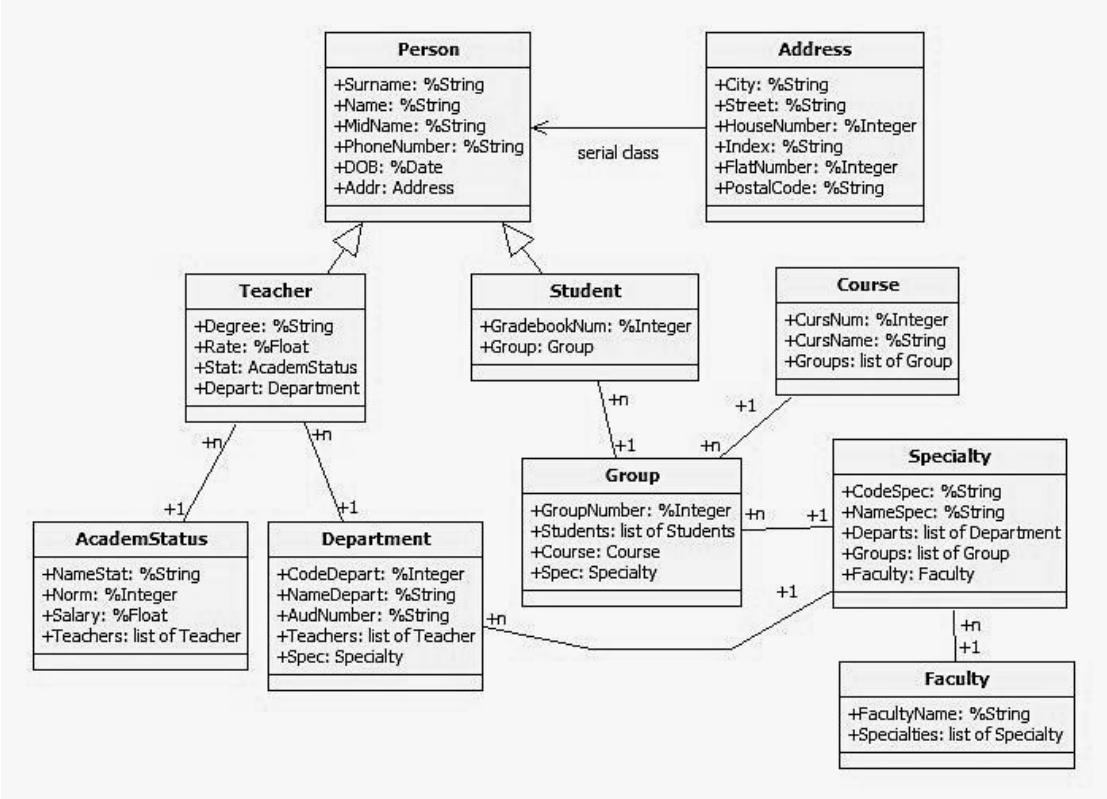


Рис. 4.4. Діаграма класів

Таким чином, описавши всі зв’язки між класами у Caché Studio, ми маємо фізичну модель даних. Залишилось тільки наповнити її коректними даними.

Створення зв'язків між відповідними об'єктами класів

1. Зв'язок ONE-TO-MANY

Для прикладу візьмемо розглянутий зв'язок між студентом (Student) та групою (Group). Приклад спеціально спрощений (у ньому опущені присвоєння властивостей значеннями) для кращого розуміння саме створення зв'язків.

```
Set stud = ##class(University.Student).%New() //створення нового студента
-------------------------------------
Set group = ##class(University.Group).%New() //створення нової групи
-------------------------------------
Set stud.Group = group //встановлення зв'язку між екземплярами класів
Student та Group
```

Таким чином ми працюємо зі зв'язком на стороні «багато». Ми присвоюємо певному об'єкту сторони «багато» (Student) відповідне посилання на об'єкт сторони «один» (Group).

Але також є й інший спосіб встановити зв'язок між цими 2 об'єктами. Як пам'ятаємо, у класі Group такий зв'язок підтримує властивість Students з кардинальністю Many. Така властивість буде вести себе як **колекція-список**. Тому при встановленні зв'язку між цими 2 об'єктами ми також можемо написати такий код.

```
Do group.Students.Insert(stud)
```

У будь-якому випадку зв'язок завжди встановиться на обох сторонах.

При цьому достатньо зберегти зміни в об'єкті на стороні «один». Об'єкти на стороні «багато» збережуться автоматично.

```
Do group.%Save()
```

Тепер наведемо приклад переглядання властивостей зв'язаних об'єктів.

```
Write stud.Group.Number //виводимо номер групи, до якої входить студент stud
Write group.Students.Count() //виводимо кількість студентів у групі
Write group.Students.GetAt(1).Surname //виводимо прізвище першого за
списком студента у групі
```

Щоб видалити об'єкт з бази даних, що знаходиться на стороні «один», спочатку потрібно розірвати його зв'язок між всіма об'єктами на стороні «багато», інакше видалити його ми не зможемо.

Розірвати зв'язок можна таким чином:

```
Set stud.Group = "" //на стороні "багато"
Do group.Students.RemoveAt(1) //на стороні "один"
```

2. Зв'язок PARENT-TO-CHILDREN

Для прикладу візьмемо зв'язок між книгою (Book) та її розділом (Paragraph). Встановлення зв'язку нічим не відрізняється від типу ONE-TO-MANY, тільки самостійно зберегти залежні об'єкти не вдасться окремо від «об'єкта-батька».

А при видаленні «об'єкта-батька» автоматично видаляться і залежні від нього об'єкти зв'язаного класу.

```
Do ##class(Book).%DeleteId(1) //автоматично видаляться і розділи
```

Також слід звернути увагу на те, що ID залежних об'єктів («дітей») формується таким чином – «a||b», де a – ідентифікатор «батька», а b – ідентифікатор «дитини».

Наприклад,

```
Set par = ##class(Paragraph).%OpenId("2||3") //створюємо посилання на
розділ з ID книги 2 та ID розділу 3
```

Завдання

1. Створіть зв'язки між класами відповідно до вашої діаграми класів за допомогою посилань та зв'язків типу relationship (ретельно подумайте, у яких випадках у вашій системі будуть використовуватись зв'язки ONE-TO-MANY, а де PARENT-TO-CHIDLREN. У випадку реалізації зв'язку MANY-TO-MANY створюйте проміжний клас);
2. Установіть зв'язки між об'єктами різних класів у вашій базі даних.
3. Здійсніть розрив зв'язку між певними об'єктами різних класів на ваш розсуд.

Контрольні питання

1. Які функції виконують зв'язки типу «relationship»?
2. Що представляє собою однонаправлена асоціація в Cache?
3. Що представляє собою двонаправлена асоціація в Cache?
4. Які особливості зв'язку ONE-TO-MANY?
5. Які особливості зв'язку PARENT-TO-CHILDREN?
6. Що представляє собою атрибут зв'язку на стороні «один» та «батько»?
7. Що представляє собою атрибут зв'язку на стороні «багато» та «діти»?
8. Як встановити зв'язок між 2 об'єктами?
9. Як змінити зв'язок між 2 об'єктами?
10. Як розірвати зв'язок між 2 об'єктами?

ЛАБОРАТОРНА РОБОТА № 5

«МЕТОДИ КЛАСУ ТА ЕКЗЕМПЛЯРА КЛАСУ В CACHÉ OBJECT SCRIPT. НАПИСАННЯ ПРОСТИХ МЕТОДІВ. CALLBACK-МЕТОДИ. ПОЛІМОРФІЗМ У СКБД CACHÉ»

Створення методів у Cache Studio

Новий метод для класу можна створити вручну або аналогічно до властивості, використовуючи майстер. Для його виклику необхідно обрати або пункт меню **Class->Add->Method** або натиснувши кнопку на панелі Class Members.

Давайте створимо метод, який створює новий об’єкт класу Teacher, встановлює значення його властивостей та зберігає його в базі даних. Цей метод буде методом класу.

На першому етапі ми вказуємо назву майбутнього метода та його опис (у коді він відобразиться як коментар).

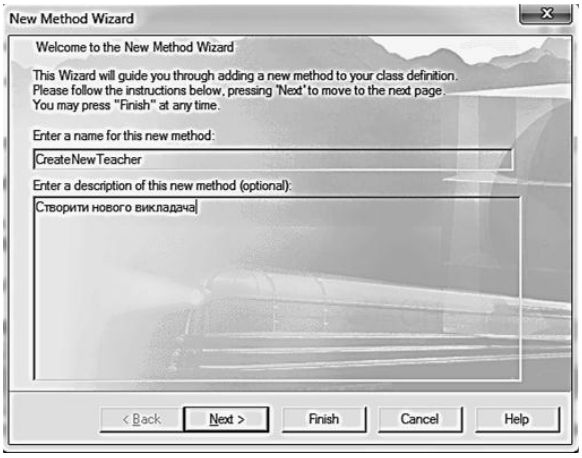


Рис. 5.1. Назва майбутнього методу та його опис

На наступному етапі вказуємо тип, що повертається (якщо метод буде повертати певний тип). Цей метод буде повертати об’єктне посилання на створений об’єкт, тому його типом буде University.Teacher.

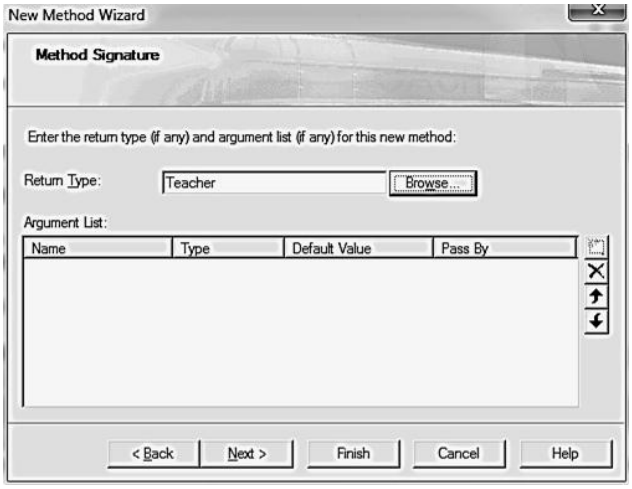


Рис. 5.2. Тип методу

На цьому ж етапі додаємо і аргументи методу (якщо вони є). У нашому випадку аргументами будуть відповідні змінні, значення яких будуть присвоєні відповідним властивостям нового об'єкта.

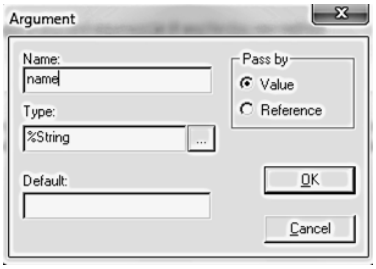


Рис. 5.3. Аргументи методу

Після завдання всіх аргументів методів вікно з аргументами буде мати такий вигляд.

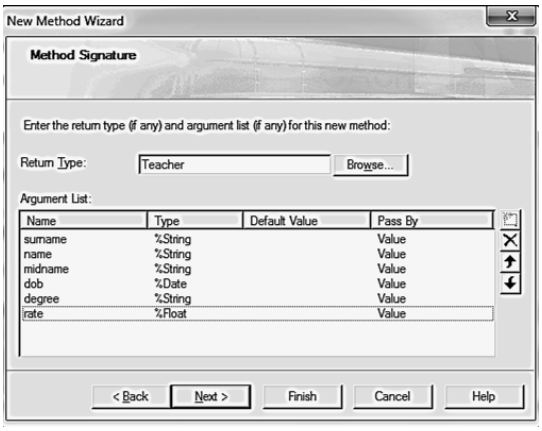


Рис. 5.4. Вікно з аргументами

Після цього етапу вже можна натискати **Finish**. У цьому випадку далі всі ключові слова методу будуть встановленні за замовчанням. При натисненні **Next** можна задати різні ключові слова, які описують метод (закритий метод – **Private**, фінальний – **Final**, метод класу – **ClassMethod** (інакше буде створений метод об'єкта), збережена процедура SQL – **SQL Stored Procedure**). На цьому ж етапі ми вказуємо і мову методу. Ми будемо писати методи на мові Cache Object Script, тому обираємо **cache** у списку **language.**

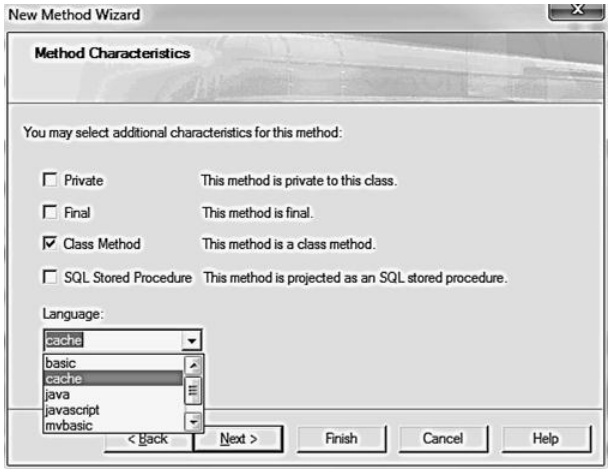


Рис. 5.5. Мова методу

Якщо і на цьому етапі натиснути Next, то ми перейдемо до вікна, у якому можна написати код цього методу. І навпаки при натисненні Finish ми вийдемо з майстра створення методу і повернемось до загального коду визначення класу.

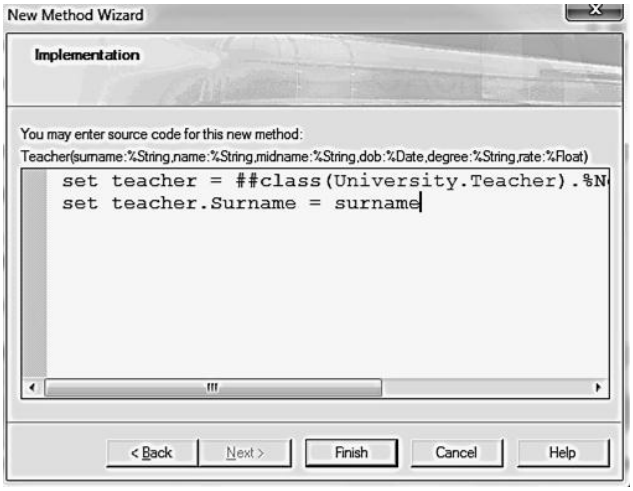


Рис. 5.6. Вихід з майстра створення методу

Код даного методу буде мати такий вигляд:

```
ClassMethod CreateNewTeacher(surname As %String, name As %String,
midname As %String, dob As %Date, degree As %String, rate As %Float)
As University.Teacher
{
     set teacher = ##class(University.Teacher).%New()
     set teacher.Surname = surname
     set teacher.Name = name
     set teacher.Midname = midname
     set teacher.DOB = $zdateh(dob)
     set teacher.Degree = degree
     set teacher.Rate = rate
     set sc = teacher.%Save() //зберігаємо об'єкт
     if sc = 1
      write "Об'єкт успішно збережений",!
     else
      do $System.OBJ.DisplayError(sc) //виводимо причину помилки
     quit teacher  //повертаємо посилання на об'єкт
}
```

Тепер скомпілюємо клас та виконаємо новий створений метод у Терміналі.

```
GORBAN>set t=##class(University.Teacher).CreateNewTeacher("Горбань","Гліб","Вале
нтинович","",1.2)
Об'єкт успішно збережений

GORBAN>
```

Обновить Закрыть окно

University.Teacher в области GORBAN

#	ID	DOB	Degree	Depart	Midname	Name	PhoneNumber	Rate	Stat	Surname
1	1		Доктор технічних наук		Тихонович	Микола		1.12		Фісун
2	2		Доктор педагогічних наук		Павлович	Олександр		1.12		Мещанінов
3	3		Доктор технічних наук		Пантелійович	Юрій		1.12		Кондратенко
4	4		Доктор технічних наук		Павлович	Максим		1.3		Мусієнко
5	5		Кандидат технічних наук		Миколаївна	Ірина		1.2		Журавська
6	6		Кандидат фізико-математичних наук		Василівна	Інеса		1.35		Кулаковська
7	7		Кандидат технічних наук		Вікторович	Євген		1.15		Сіденко
8	8				Олександрович	Євген		1.49		Давиденко
9	9				Валентинович	Гліб		1.2		Горбань

Завершено

Рис. 5.7. Компіляція класу

Тепер давайте створимо метод редагування об’єкта. Він буде мати ті самі аргументи, що й метод створення, але, на відміну від нього, цей метод буде методом об’єкта, тобто його можна викликати тільки якщо є екземпляр класу.

Код методу буде таким:

```
Method EditTeacher(surname As %String, name As %String,
midname As %String, degree As %String, rate As %Float)
{
     set ..Surname = surname
     set ..Name = name
     set ..Midname = midname
     set ..Degree = degree
     set ..Rate = rate
     set sc = ..%Save() //зберігаємо об’єкт
     if sc = 1 write "Об’єкт успішно збережений",!
     else  do $System.OBJ.DisplayError(sc) //виводимо причину помилки
}
```

Виконаємо цей метод у Терміналі, використовуючи команду *do*.

```
GORBAN>set t=##class(University.Teacher).%OpenId(9)

GORBAN>write t.Surname
Горбань
GORBAN>do t.EditTeacher("Бурлаченко","Іван","Сергійович","",1.32)
Об'єкт успішно збережений

GORBAN>
```

Обновить Закрыть окно

University.Teacher в области GORBAN

#	ID	DOB	Degree	Depart	Midname	Name	PhoneNumber	Rate	Stat	Surname
1	1		Доктор технічних наук		Тихонович	Микола		1.12		Фісун
2	2		Доктор педагогічних наук		Павлович	Олександр		1.12		Мещанінов
3	3		Доктор технічних наук		Пантелійович	Юрій		1.12		Кондратенко
4	4		Доктор технічних наук		Павлович	Максим		1.3		Мусієнко
5	5		Кандидат технічних наук		Миколаївна	Ірина		1.2		Журавська
6	6		Кандидат фізико-математичних наук		Василівна	Інеса		1.35		Кулаковська
7	7		Кандидат технічних наук		Вікторович	Євген		1.15		Сіденко
8	8				Олександрович	Євген		1.49		Давиденко
9	9				Сергійович	Іван		1.32		Бурлаченко

Завершено

Рис. 5.8. Використання команди «do» у терміналі

У якості наступного прикладу наведемо метод екземпляра класу University.AcademStatus (посада), який друкує на екрані викладачів, що займають дану посаду.

```
Method PrintTeachers()
{
     set count=..Teachers.Count()
     write "Прізвище",?15,"Ім'я",?25,"По-батькові",!
     for i=1:1:count
     {
      set teacher=..Teachers.GetAt(i)
      write teacher.Surname,?15,teacher.Name,?25,teacher.Midname,!
     }
}
```

Тепер виконаємо метод PrintTeachers() в Терміналі:

```
GORBAN>set stat=##class(University.AcademStatus).%OpenId(1)

GORBAN>write stat.NameStat
професор
GORBAN>do stat.PrintTeachers()
Прізвище        Ім'я       По-батькові
Фісун           Микола     Тихонович
Мещанінов       Олександр  Павлович
Кондратенко     Юрій       Пантелійович
Мусієнко        Максим     Павлович
Кутковецький    Валентин   Якович

GORBAN>
```

Рис. 5.9. Метод PrintTeachers()

Callback-методи

Методи Callback автоматично викликаються як реакція на певні дії або події в системі. Їх безпосередній виклик не передбачений.

У ці методи розробники поміщають код, який повинен спрацьовувати автоматично при появі відповідних подій. Щоб відрізнити методи цього типу від інших, їх імена записуються у форматі *%On<Event>*, де Event – ім'я події, що запускає метод. **Усі Callback-методи повинні мати повертати тип *%Status*.**

Приклад Callback-методу

Клас University.AcademSttus (Посада) має зв'язок з класом University.Teacher (викладач). Якщо ми захочемо видалити деякий об'єкт класу AcademSttus, викликавши метод *%Delete()* або *%DeleteId()*, то у випадку, якщо об'єкт має зв'язані об'єкти класу Teacher, то він видалений не буде.

Щоб усе ж видалити об'єкт класу AcademSttus, треба спочатку розірвати зв'язки з об'єктами класу Teacher, викликавши метод колекції Clear(). Цю операцію і будемо робити в тілі Callback-методу *%OnDelete()*.

Для того, щоб створити метод *%OnDelete()*, його можна описати вручну або за допомогою пункту меню **Class->Override**, оскільки цей метод унаслідується від базового класу *%Persistent*. У якості параметра такий метод приймає повний ідентифікатор об'єкта.

```
ClassMethod %OnDelete(oid As %ObjectIdentity) As %Status [ Private,
ServerOnly = 1 ]
{
     set stat = ##class(University.AcademStatus).%Open(oid)
     do stat.Teachers.Clear()
     set sc = stat.%Save()
     if sc = 1
 {
      Quit $$$OK
     }
     else {
      Quit 0
     }
}
```

Таким чином, при спробі видалення певного об'єкта класу AcademStatus буде викликаний метод *%OnDelete()*, який розірве всі існуючі зв'язки цього об'єкта з об'єктами класу Teacher, після чого цей об'єкт буде видалено.

Реалізація поліморфізму в СКБД Caché

Поліморфізмом називається можливість об'єктів з однаковою специфікацією мати різну реалізацію. Він виводить принцип розділення інтерфейсу від реалізації на якісно новий рівень. Поліморфізм поліпшує структуру програми і робить її більш наглядною, а також дозволяє створювати розширені програми, можливості яких можуть нарощуватись за мірою необхідності.

Завдяки наслідуванню, об'єкт можна інтерпретувати як об'єкт свого або базового типу. Ця можливість є достатньо важливою, оскільки вона дозволяє інтерпретувати різні типи, похідні від базового типу як типи, що належать до нього. Таким чином можна написати фрагмент програми, який зможе працювати з будь-яким із множини типів. Віртуальні функції дозволяють одному типу виразити відмінність від іншого типу – за умови, що вони обидва наслідуються від одного базового типу.

У предметній області Lipa поліморфізм представлений базовим класом Payment (Оплата) та його 3 нащадками: класами PurchaseOrder (Доручення на покупку), Check (Чек) та CreditCard (Кредитна картка).

Оголосимо клас Person абстрактним, але вкажемо тип класу persistent. Цей клас буде мати екземпляри, але кожен з них буде обов'язково належати до одного з нащадків класу Person. Також в SQL-проекції створиться відповідна таблиця.

```
Class University.Person Extends %Persistent [Abstract, ClassType =
persistent]
{
     …
}
```

Тепер приступимо до реалізації поліморфізму. Для того щоб він мав місце, потрібно оголосити у базовому класі метод, але не реалізовувати його. Цей метод буде віртуальним. Але Cache не дозволяє створювати методи без жодного рядка коду, виходом із такої ситуації буде написання в коді методу лише однієї команди – quit.

Наприклад, створимо для класу Person віртуальний метод PrintData, який буде реалізований у класах-нащадках по-різному. Для базового класу в тілі його метода напишемо тільки команду quit.

```
Method PrintData()
{
     quit
}
```

Тепер перевизначимо цей метод у класах-нащадках: Teacher та Student.

У класі Teacher указаний метод буде виводити на екран ПІБ викладача (відповідні властивості успадковані від класу Person), а також інформацію, характерну тільки для викладачів: науковий ступінь, назву посади та назву кафедри, за якою закріплений відповідний викладач.

```
Method PrintData()
{
     write ..Surname,?15,..Name,?25,..Midname,!
     write "Науковий ступінь: ",..Rate,!
     write "Посада: ",..Stat.NameStat,!
     write "Кафедра: ",..Depart.NameDepart,!
}
```

Для класу Student даний метод буде виводити зовсім іншу інформацію, що характерна для студента: номер залікової книжки, номер групи, до якої належить студент, а також назву спеціальності, на якій він навчається.

```
Method PrintData()
{
     write ..Surname,?15,..Name,?25,..Midname,!
     write "№ залікової книжки: ",..GradebookNum,!
     write "Група: ",..Group.GroupNumber,!
     write "Спеціальність: ",..Group.Spec.NameSpec,!
}
```

Виконаємо перевірку. Створимо в пам'яті об'єктне посилання на об'єкт класу Person та почергово відкриємо певні екземпляри класів його нащадків (спочатку викладача, а потім студента), і виконаємо почергово для них метод PrintData().

```
GORBAN>set p=##class(University.Person).%OpenId(1)

GORBAN>do p.PrintData()
Фісун          Микола     Тихонович
Науковий ступінь: 1.12
Посада: професор
Кафедра: Кафедра інтелектуальних інформаційних систем

GORBAN>set p1=##class(University.Person).%OpenId(13)

GORBAN>do p1.PrintData()
Білявський     Ігор       Сергійович
№ залікової книжки: 21410101
Група: 201
Спеціальність: Комп'ютерні науки

GORBAN>
```

Рис. 5.10. Виконання методу PrintData() у Терміналі

Завдання

1. Відповідно до свого варіанту продумати методи для збережених класів та реалізувати їх. Додати методи до діаграми класів з ЛР №4. Загальна кількість методів – на власний розсуд, але не менше 5.
2. З усіх методів реалізувати принаймні один віртуальний метод.
3. Напишіть будь-який callback-метод для первного класу (на Ваш розсуд) та перевірте його роботу.

Контрольні запитання

1. Які типи методів підтримує СКБД Caché та чим вони відрізняються?
2. Наведіть приклад визначення методу класу та методу екземплярів класу?
3. Які види методів Ви знаєте? Наведіть коротку характеристику.
4. Що таке «аргументи методу»?
5. Що таке «відносно точковий синтаксис», для чого він використовується? Наведіть приклад.
6. Які ключові слова методу Вам відомі? Наведіть коротку характеристику.
7. Що таке callback-методи? Чим вони відрізняються від звичайних методів?
8. Як реалізувати поліморфізм у СКБД Caché?

ЛАБОРАТОРНА РОБОТА № 6

«РЕЛЯЦІЙНА МОДЕЛЬ ДАНИХ У СКБД CACHÉ. ВИКОРИСТАННЯ МОВИ SQL»

Визначення запитів в СКБД Caché

Запити (англійською Queries) представляють у розпорядження розробника операції з множинами екземплярів класів. Якщо розглядати всі екземпляри класу як загальну множину, а результат запиту – як його деяку підмножину, то можна вважати, що запити утворюють для об'єктів свого роду фільтр.

За нашим вибором ми можемо формувати запити або безпосередньо на Caché ObjectScript або на Caché Basic, або на SQL. Cache Studio оснащений нескладним у застосуванні майстром запитів (New Query Wizard), за допомогою якого можна з легкістю створювати SQL-запити.

Запустити майстер створення запитів у Caché Studio можна за допомогою меню **Class->Add->Query** або натисканням кнопки на панелі **Class Members**.

На першому етапі задається ім'я нового запиту класу та вказується тип запиту (SQL-запит або запит, визначений користувацьким кодом. Ми будемо створювати запит FindByName для класу Customer (Клієнт).

Рис. 6.1. Запит FindByName для класу Customer

На наступному етапі вказуються параметри запиту подібно до створення нового методу.

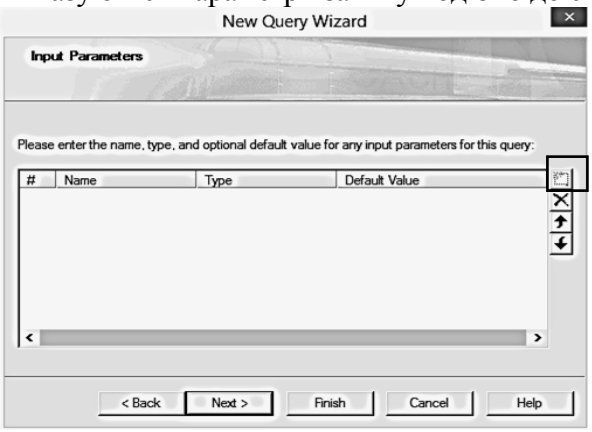


Рис. 6.2. Параметри запиту

Параметром запиту в нашому випадку буде name типу *%String* (також можна зазначити і значення параметра за замовчанням).

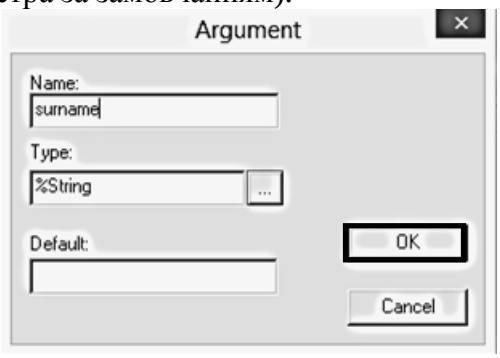


Рис. 6.3. Вибір типу

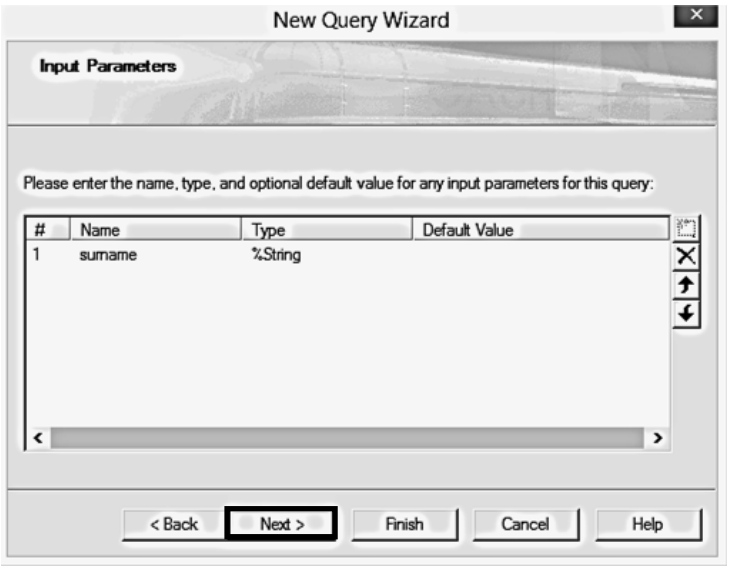


Рис. 6.4. Перехід до наступного етапу

На наступному етапі потрібно вказати властивості класу, які буде повертати запит. У мові SQL вони вказуються після оператора SELECT.

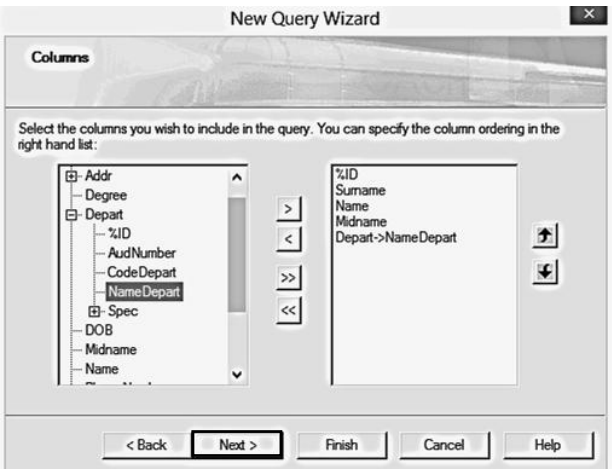


Рис. 6.5. Властивості класу, які буде повертати запит

Наступним етапом є завдання умови запиту. Тут ми вказуємо ім'я властивості, умову та вираз. У виразі можна вказати певне значення або певний параметр запиту, якщо він є, поставивши перед ним двокрапку. Задамо умову, що ім'я клієнта починається з літери, указаної в параметрі :name (умова *%STARTSWITH*).

Рис. 6.6. Завдання умови

Останнім етапом є вказування властивості, за якою буде відбуватись сортування, та порядком сортування (за збільшенням або зменшенням).

Рис. 6.7. Зазначення властивості, за якою буде відбуватись сортування

Після того як ми пройдемо через всі етапи майстра створення запиту – отримаємо в тілі запиту код на мові SQL, який у принципі можна було б написати і вручну. Варто зазначити, що при використанні параметрів запиту мови SQL перед ними ставиться двокрапка.

```
Query FindBySurname(surname As %String) As %SQLQuery
{
SELECT Surname,Name,Midname,Stat->NameStat,Depart->NameDepart FROM Teacher
WHERE (Surname %STARTSWITH :surname)
ORDER BY Surname
}
```

Виконання динамічних запитів, використовуючи клас *%ResultSet*

Опис запиту в певному класу є ще тільки половиною справи, оскільки при цьому він був тільки визначений, але треба знати, як його виконати. Для цього в Cache існує системний клас *%ResultSet* (скорочений напис *%Library.ResultSet*), який надає об'єктно-орієнтований інтерфейс, за допомогою якого у стають доступними запити, що включені в опис класу. Основні елементи класу *%ResultSet* наведені в табл. 6.1.

Таблиця 6.1.

Методи та властивості класу *%ResultSet*

Тип	Ім'я	Пояснення
Методи класу	`%New("Class:Query")`	Створює новий об'єкт ResultSet. У якості параметру отримує рядок символів, що складається з імені класу, двокрапки та імені запиту. Якщо виконується динамічний запит *(%Dynamic Query:SQL*), імена класу та запиту опускаються. У цьому випадку SQL-код повинен бути заданий у методі Prepare()
Методи екземплярів	`QueryIsValid()`	Повертає 1, якщо запит допустимий, в іншому випадку – 0
	`ContainsId()`	Повертає номер відповідного стовпця, якщо кожен рядок, що входить у вибірку, має ID-поле, в іншому випадку повертає 0
	`GetParamCount()`	Кількість параметрів запиту
	`GetParamName(Position)`	Ім'я параметра в позиції Position
	`Prepare(SQL-код)`	SQL-код, що підготовлюється для динамічного запиту. Параметри позначаються знаками питання (?)
	`Execute(P1,P2,P3,…)`	Виконує запит із параметрами P1,P2,P3,…
	`GetColumnCount()`	Кількість стовпців у вибірці
	`GetColumnName(Column)`	Ім'я стовпця Column
	`GetColumnHeader(Column)`	Заголовок стовпця Column
	`Next()`	Перехід на наступний рядок вибірки
	`Get(ColumnName)`	Значення поля з іменем ColumnName поточного рядка
	`GetData(Column)`	Значення поля з номером Column поточного рядка
	`Close()`	Завершує запит
Властивості	`AtEnd`	Містить 1, якщо досягнутий кінець вибірки, в іншому випадку – 0
	`ClassName`	Ім'я класу запиту
	`QueryName`	Ім'я запиту
	`Data(ColumnName)`	Багатомірна властивість, яка містить значення всіх полів. З точки зору ефективності йому варто віддати перевагу перед використанням методів Get() або GetData()

За допомогою ResultSet можна проводити операції з множинами екземплярів об'єктів. При цьому відсутня необхідність утворювати екземпляри кожного з запитуваних об'єктів. Алгоритм роботи з ResultSet полягає у наступному:

1. Створити новий об'єкт ResultSet за допомогою методу класу *%New()*. Перевірити допустимість запиту за допомогою методу QueryIsValid(). Інший метод: задати SQL-код динамічного запиту в методі Prepare() і переконатись в його допустимості, перевіривши значення методу QueryIsValid().

2. Підготувати параметри: дізнатись їх кількість, використовуючи метод GetParamCount(). У свою чергу метод GetParamName(Position) поверне ім'я відповідного параметра. Викликати метод Execute() з заданими параметрами.

3. Методом Next() витягти наступний рядок. На підставі поверненого значення перевірити, чи прочитані дані; в іншому випадку (перший символ коду завершення дорівнює 0) вийти з циклу.

4. Значення кожного поля можна отримати з багатомірної властивості Data(Columname). Виконати необхідну обробку даних, виведення на екран і т. ін.

5. При завершенні обробки закрити запит. Це можна зробити одним з 2 способів:

– Знищити об'єкт ResultSet;

– Викликати метод Close(); у цьому випадку можна повторно виконувати той самий запит (Execute(), Next()) без необхідності його повторної підготовки (тобто, не виконуючи Prepare()).

Приклад. Напишемо метод пошуку клієнтів з використанням визначеного запиту FindByName.

```
ClassMethod LookUp() As %Status
{
     read !, "Введіть частину прізвища: ",name
     write "...пошук... ",!
     set found = 0 //припустимо, що немає даних, які задовольняють умовам
запиту
     set rs =
##class(%Library.ResultSet).%New("University.Teacher:FindBySurname")
   if (rs.QueryIsValid())
   {
      do rs.Execute(name)

      while (rs.Next() '= 0)
       {
            set:(found = 0) found = 1 //як мінімум одне співпадіння
            write rs.Get("ID»), "
",rs.Get("Surname"),?15,rs.Get("Name"),?25,rs.Get("Midname"),?37,rs.Get("N
ameDepart"),!
        }
   }
     do rs.Close() //закрити запит
     quit:(found=0) 0 //немає співпадінь
     quit 1
}
```

Тепер виконаємо даний метод у Терміналі:

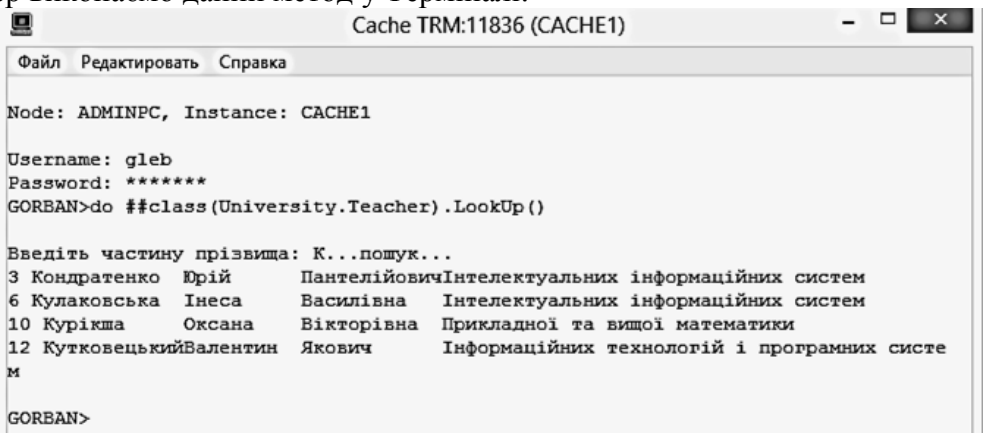


Рис. 6.8. Виконання методу LookUp у Терміналі

Використання вбудованих операторів SQL

Для спрощення розробки прикладних систем баз даних Cache підтримує вбудовування SQL в методи та програми. Тоді кажуть про вбудований SQL або Embedded SQL. Він може використовуватись в рамках Cache Object Script для проведення складних запитів до баз даних та з метою прив'язки результатів до змінних Cache ObjectScript.

Вбудований SQL використовується у визначеннях методу за допомогою препроцесорної функції `&sql()`.

Напишемо інший метод пошуку клієнтів, який буде використовувати вбудований SQL. Назвемо його LookUp2().

```
ClassMethod LookUp2() As %Status
{
     read !, "Введіть частину прізвища: ",surname
     write "...пошук... ",!
     &sql(DECLARE Cur CURSOR FOR
          SELECT %ID,Surname,Name,Midname,Depart-
>NameDepart FROM University.Teacher
            WHERE (Surname %STARTSWITH :surname)
            ORDER BY Surname)  //для запиту визначаємо курсор
   &sql(OPEN Cur)  //відкриваємо курсор
   for
   {
         &sql(FETCH Cur INTO
:id,:surname,:name,:midname,:namedepart) //переміщуємось курсором по
записам вибірки та записуємо значення його полів у відповідні локальні
змінні
         quit:SQLCODE'=0 //якщо досягнутий кінець вибірки, то виходимо з
циклу
         write id, " ",surname,?15,name,?25,midname,?37,namedepart,!
   }
   &sql(CLOSE Cur)  //закриваємо курсор
   quit 1
}
```

У методі використовується системна змінна SQLCODE, яка отримує цифрове значення, що визначає один з наступних станів, що зазачені в таблиці 6.2:

Таблиця 6.2.

Значення системної змінної SQLCODE

SQLCODE	Значення
0	Успішно завершено
100	Успішно завершено, однак (більше) не знайдено жодного запису, який задовольняє умові
<0	Виникла помилка. Повний список значень, що повертаються в SQLCODE, можна знайти в документації по Cache

Виконаємо створений метод класу Customer у Терміналі:

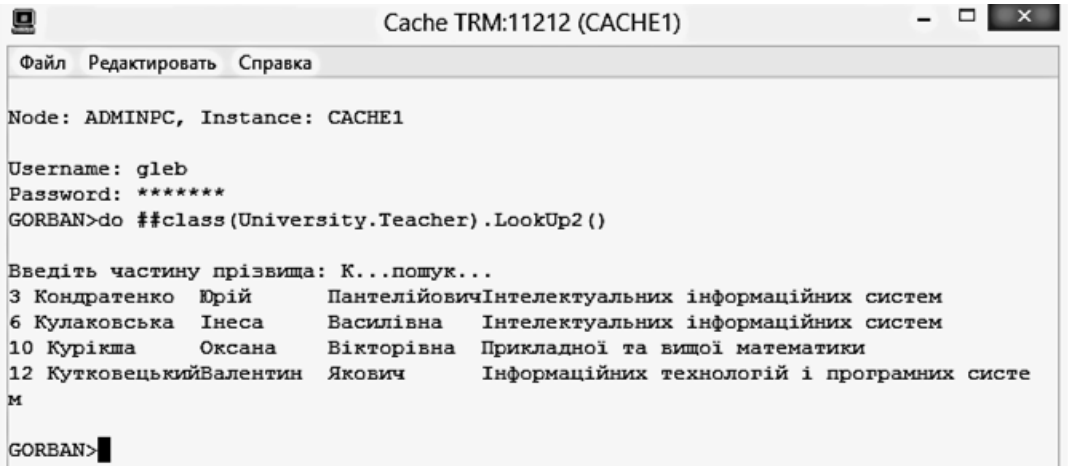


Рис. 6.9. Метод класу Customer у Терміналі

Виконання запитів SQL та створення представлень у Порталі управління системою

У Порталі управління системою також представлена можливість виконання запитів SQL. Для того, щоб виконати запит SQL «на льоту» потрібно зайти на сторінку **System Explorer->SQL->Execute SQL Statements**, для створення представлення (View) та подальшого його зберігання – **System Explorer->SQL->Create SQL Views**.

При виборі виконання запиту Портал набуде такого вигляду:

Рис. 6.10. Портал управління системою

Створення нового класу в СКБД Cache за допомогою запиту SQL

Це стає можливим за допомогою запиту CREATE TABLE у мові SQL. Даний запит можна виконати в Порталі управління системою або за допомогою програмного коду, описаного вище (2 способами).

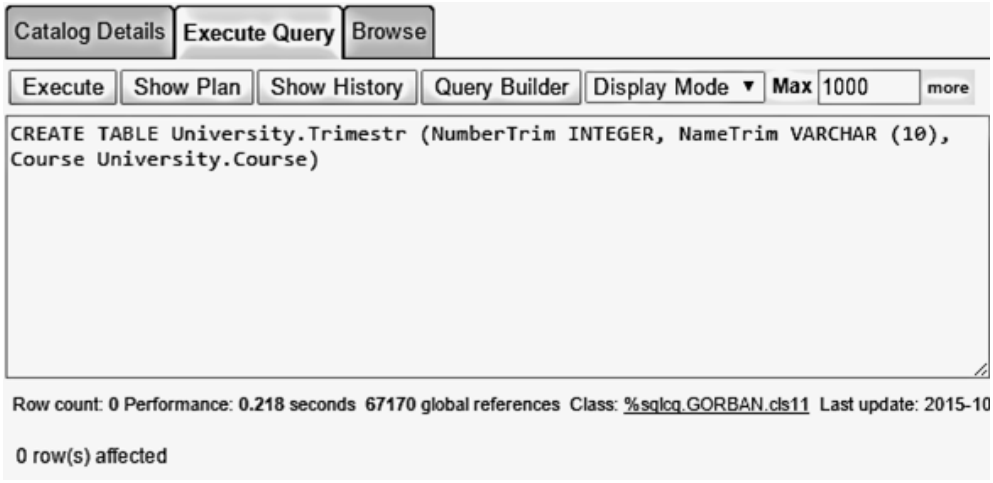


Рис. 6.11. Запит CREATE TABLE у мові SQL

```
set sql = "CREATE TABLE University.Trimestr "_
          " (NumberTrim INTEGER, NameTrim VARCHAR (10), "_
          "Course University.Course) "
set rs = ##class(%Library.ResultSet).%New("%Library.DynamicQuery:SQL")
do rs.Prepare(sql)
do rs.Execute()
do rs.Close()
```

В обох випадках ми побачимо у Студії новий клас Lipa.Promotion і, відкривши його, подивимось його опис мовою Cache Object Script, згенерований Cache.

```
///
Class University.Trimestr Extends %Persistent [ ClassType = persistent,
DdlAllowed, Owner = gleb, ProcedureBlock, SqlRowIdPrivate, SqlTableName =
Trimestr, StorageStrategy = Default ]

{

Property NumberTrim As %Library.Integer(MAXVAL = 2147483647, MINVAL = -
2147483648) [ SqlColumnNumber = 2 ];

Property NameTrim As %Library.String(MAXLEN = 10) [ SqlColumnNumber = 3 ];

Property Course As University.Course [ SqlColumnNumber = 4 ];
```

Завдання

1. Згідно зі своїм варіантом для предметної області виконати запит SQL (див. завдання SQL у файлі «Варіанти завдань.docx») 2 способами:
   - з використанням об’єкта ResultSet;
   - за допомогою вбудованого оператора SQL та курсору.
2. Виконати цей запит у Порталі керування системою;
3. Засобами SQL (будь-яким способом на Ваш розсуд) створити новий клас у СКБД Cache.

Контрольні питання

1. Знати, що представляє собою відповідне об'єктне поняття в реляційній проекції.
2. Для чого призначений клас *%ResultSet*?
3. Назвіть основні методи класу *%ResultSet*.
4. Як пройтись по всій вибірці запиту, використовуючи *%ResultSet*?
5. Як оголошуються локальні змінні Cache в запитах SQL?
6. Який вигляд має вбудований оператор SQL в Cache?
7. Як пройтись по всій вибірці запиту, використовуючи вбудований оператор SQL?
8. Як створити новий клас в Cache засобами SQL?

ЛАБОРАТОРНА РОБОТА № 7

«ІЄРАРХІЧНА МОДЕЛЬ ДАНИХ У СКБД САСНЁ. ЗБЕРІГАННЯ ДАНИХ У ВИГЛЯДІ ГЛОБАЛІВ»

Прямий доступ до даних

Змінні як локальні, так і глобальні можуть існувати у вигляді простих або індексованих структур. Глобальні змінні або глобали, будучи даними, що зберігаються, створюють основу так званого прямого доступу. Багатомірність даних реалізована через індекси, тому говорять про індексовані змінні. Таблиця 7.1 демонструє відмінності між змінними:

Таблиця 7.1.

Відмінності між скалярними та багатомірними змінними

Тип змінної	Локальна	Глобальна
Скалярна	Name=Іванов	^Name=Іванов
Багатомірна	Book(Nomer)=Солярис|Лемм С.	^Book(Nomer)=Солярис|Лемм С.

Змінна Name – скалярна, вона не зберігається в базі даних. Змінна ^Name – глобальна, вона зберігається в базі даних.

Змінні Book і ^Book – це індексовані змінні, на відміну від змінних Name і ^Name. Змінна ^Book є глобальною, тобто зберігається в базі даних.

Глобали – це структури даних, як правило, багатомірні, які зберігаються в базі даних і можуть оброблятися в розрахованому на багато користувачів середовищі різними процесами. Обробка глобалів в Caché достатньо проста: за допомогою команди Set глобалу привласнюється значення, яке може бути або простим або представляти собою список значень. В останньому випадку треба скористатись функцією створення списку $listbuild().

Caché зберігає глобальні дані у B*-деревах. Дерево, яке має рівне число рівнів у кожному своєму піддереві, називається збалансованим. Характерною ознакою B-дерева є мінімізація середньої кількості звертань до дискових блоків при пошуку потрібного запису даних. B-дерево, у якого кожний ключ вказає на блок даних, називається B*-деревом. Воно робить можливим інтеграцію області вказівників та області даних. B*-дерева складаються з різних блоків: блоку каталогу на вершині, одного або деяких блоків вказівників, а також блоків даних, що знаходяться на нижньому рівні, які містять збережені дані.

Створення глобалу

Опишемо приблизну структуру університету у вигляді глобалу. Нехай у його корені буде зберігатись назва університету.

На першому рівні буде міститись загальна інформація про кожен з факультетів, а саме прізвище та ініціали декана відповідного факультету та номер телефону деканату. Таку інформацію доцільно зберігати у списках, що створюються за допомогою функції $listbuild(). При цьому сама назва факультету гратиме роль індексу першого рівня для отримання відповідних даних.

На другому рівні дерева зберігатиметься інформація про групи студентів. Індексом на даному рівні є номер групи, а значенням – прізвище та ініціали куратора групи.

Наостанок, на третьому рівні зберігатиметься інформація про студентів, що входять до відповідної групи. Індексом третього рівня дерева є номер залікової книжки студента, а значенням – прізвище та ініціали відповідного студента.

Зобразимо структуру глобалу схематично.

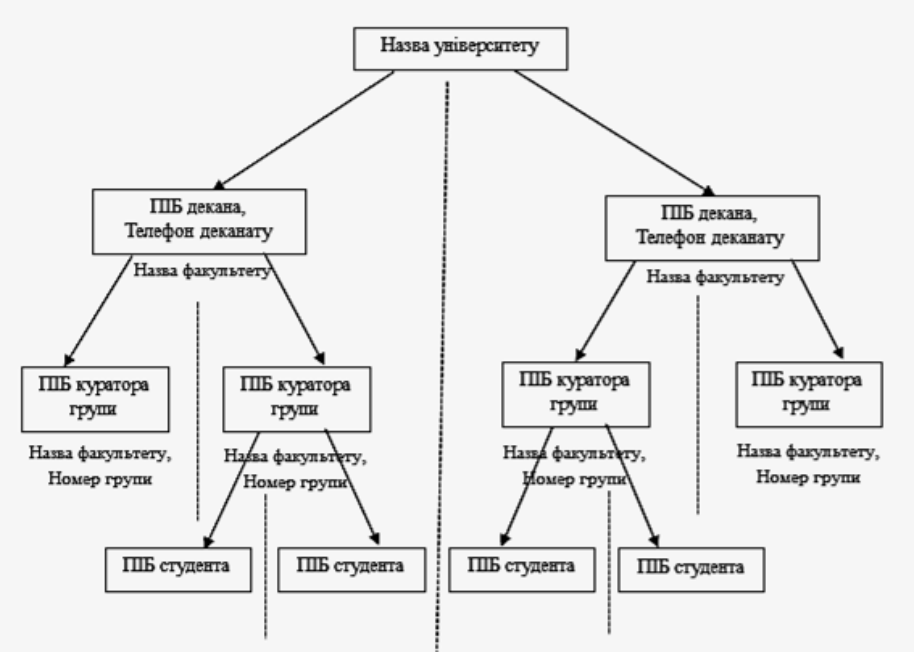


Рис. 7.1. Схематична структура глобалу

Створимо глобал відповідно до описаної вище структури шляхом написання підпрограми CreateUniversityGlobal у термінальній програмі University.Global. Ця підпрограма вхідних параметрів не має, її назва вказується за допомогою мітки, що описується, починаючи з першого символу рядка. Нагадаємо, що для написання програмного коду в термінальних програмах рядок повинен починатись з проміжку, окрім того рядка, в якому описується мітка.

```
CreateUniversityGlobal
set ^Univer = "Чорноморський державний університет імені Петра Могили"
set ^Univer("Факультет комп'ютерних наук") = $listbuild("Мусієнко М.П. ", "76-55-74")
set ^Univer("Факультет економічних наук") = $listbuild("Іщенко Н.М. ", "46-41-26")
set ^Univer("Факультет політичних наук") = $listbuild("Шевчук О.В. ", "76-55-53")
set ^Univer("Інститут філології") = $listbuild("Пронкевич О.В. ","76-92-60")
set ^Univer("Юридичний факультет") = $listbuild("Січко Д.С. ","76-55-65")
set ^Univer("Факультет соціології") = $listbuild("Ляпіна Л.А.","76-55-52")
set ^Univer("Медичний інститут") = $listbuild("Лебідь С.М.","76-55-54")

set ^Univer("Факультет комп'ютерних наук",101) = "Бурлаченко І.С. "
set ^Univer("Факультет комп'ютерних наук",102) = "Давиденко Є.О. "
set ^Univer("Факультет комп'ютерних наук",103) = "Нездолій Ю.О. "
set ^Univer("Факультет комп'ютерних наук",105) = "Крайник Я.М. "
set ^Univer("Факультет комп'ютерних наук",106) = "Корецька О.С. "
set ^Univer("Факультет комп'ютерних наук",107) = "Кулаковська І.В. "
set ^Univer("Факультет комп'ютерних наук",201) = "Боровльова С.Ю. "
set ^Univer("Факультет комп'ютерних наук",202) = "Швед А.В. "
set ^Univer("Факультет комп'ютерних наук",203) = "Солобуто Л.В. "
set ^Univer("Факультет комп'ютерних наук",205) = "Салтовський Б.Г. "
```

```
set ^Univer("Факультет комп’ютерних наук",206) = "Горбань Г.В. "

set ^Univer("Факультет комп’ютерних наук",101,1510101) = "Антонюк В.А. "
set ^Univer("Факультет комп’ютерних наук",101,1510102) = "Бондаренко У.А. "
set ^Univer("Факультет комп’ютерних наук",101,1510103) = "Головатий В.Р. "
set ^Univer("Факультет комп’ютерних наук",101,1510104) = "Доробанський М.Ю."
set ^Univer("Факультет комп’ютерних наук",101,1510105) = "Задорожна О.А. "
set ^Univer("Факультет комп’ютерних наук",101,1510106) = "Іващенко С.В. "
set ^Univer("Факультет комп’ютерних наук",101,1510107) = "Каланжова А.С. "
set ^Univer("Факультет комп’ютерних наук",101,1510108) = "Кліменко Д.О. "
set ^Univer("Факультет комп’ютерних наук",101,1510109) = "Корбут В.В. "
set ^Univer("Факультет комп’ютерних наук",101,1510110) = "Лепетинський Е.Р. "
set ^Univer("Факультет комп’ютерних наук",101,1510111) = "Місюк Т.О."
set ^Univer("Факультет комп’ютерних наук",101,1510112) = "Олейніченко Є.Є."
set ^Univer("Факультет комп’ютерних наук",101,1510113) = "Осадчий А.О."
```

Для того, щоб виконати цю підпрограму потрібно у Терміналі написати наступну команду:

```
do CreateUniversityGlobal^University.Global,
```

де CreateUniversityGlobal – назва підпрограми, University.Global – назва загальної програми, у якій указана підпрограма описана та реалізована.

Перегляд даних глобалів

Подібно до перегляду реляційного представлення даних у Порталі управління системою можливий перегляд даних, що зберігаються у глобалах. Для цього потрібно скористатись шляхом **System Explorer -> Глобали**.

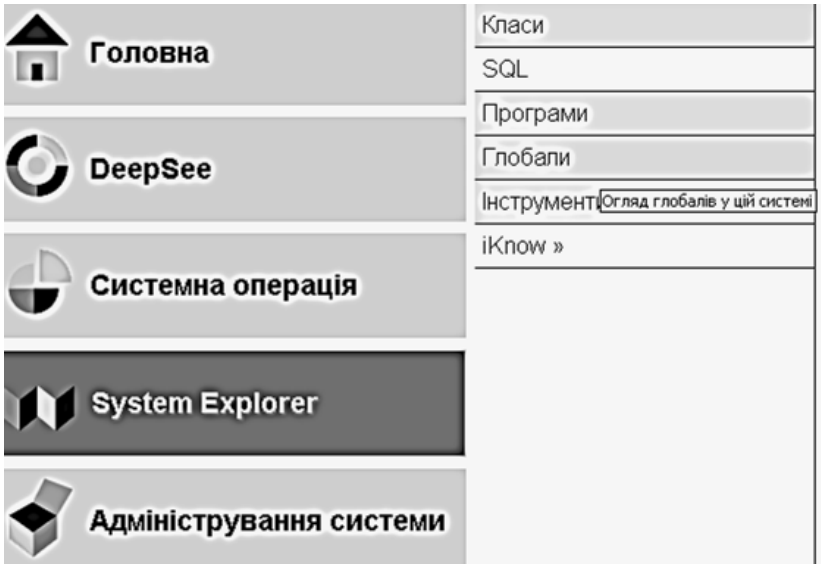
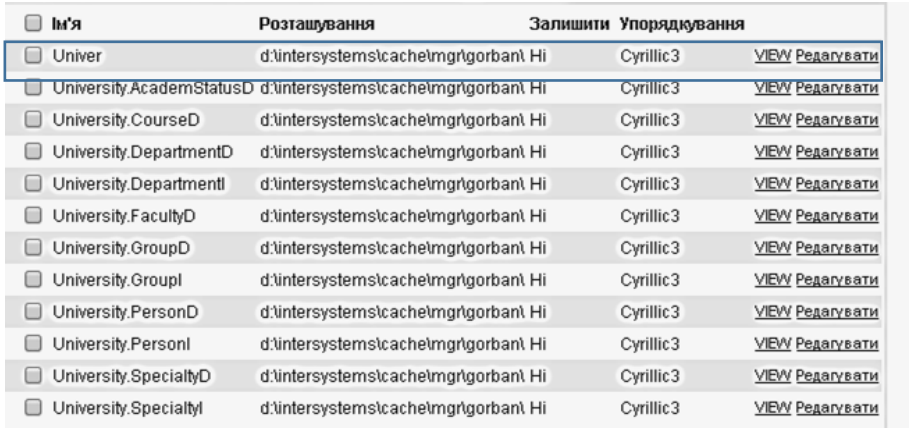


Рис. 7.2. Перегляд даних глобалів

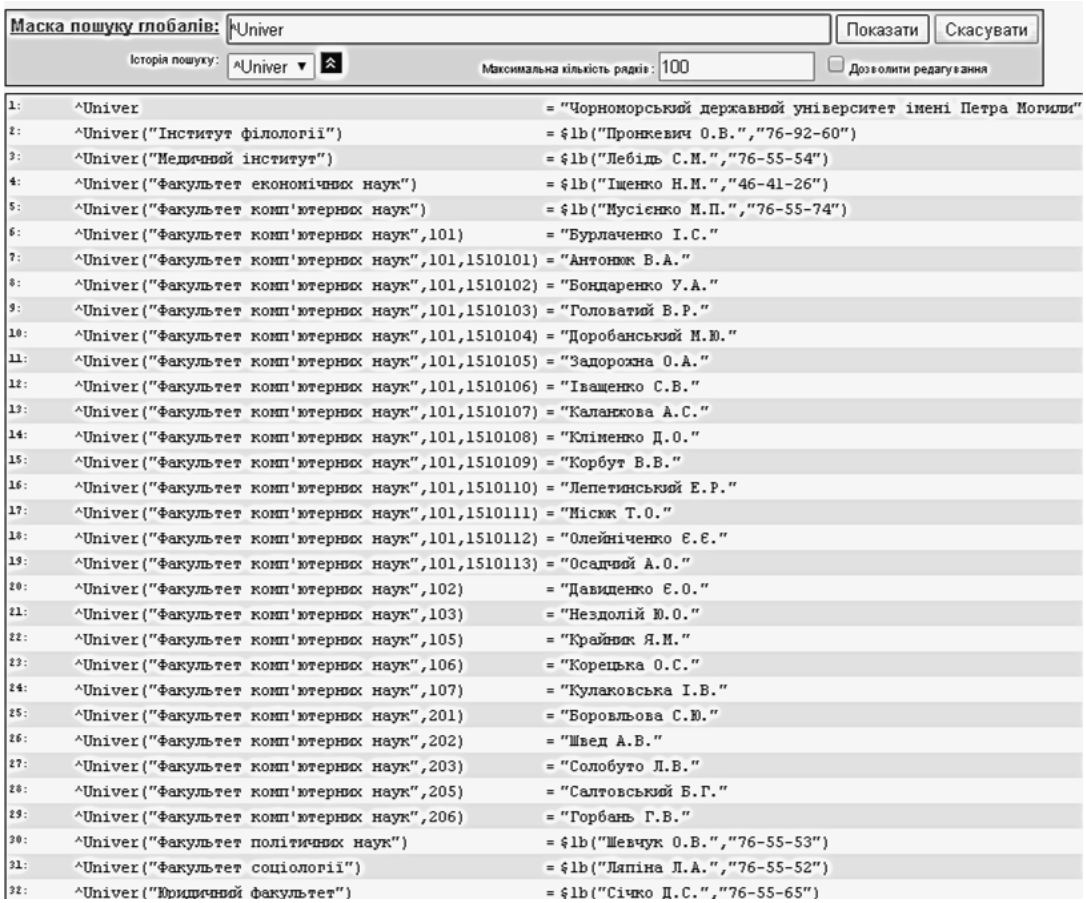
При переході за даним пунктом меню буде відображено список, що збережений у відповідній області даних (нагадаємо, що у кожного студента вона своя). Тут ми побачимо і створений глобал ^Univer. Його вміст можна переглянути, здійснивши перехід за відповідним посиланням **VIEW** для нього.



Ім'я	Розташування	Залишити	Упорядкування	
Univer	d:\intersystems\cache\mgr\gorban\	Hi	Cyrillic3	VIEW Редагувати
University.AcademStatusD	d:\intersystems\cache\mgr\gorban\	Hi	Cyrillic3	VIEW Редагувати
University.CourseD	d:\intersystems\cache\mgr\gorban\	Hi	Cyrillic3	VIEW Редагувати
University.DepartmentD	d:\intersystems\cache\mgr\gorban\	Hi	Cyrillic3	VIEW Редагувати
University.DepartmentI	d:\intersystems\cache\mgr\gorban\	Hi	Cyrillic3	VIEW Редагувати
University.FacultyD	d:\intersystems\cache\mgr\gorban\	Hi	Cyrillic3	VIEW Редагувати
University.GroupD	d:\intersystems\cache\mgr\gorban\	Hi	Cyrillic3	VIEW Редагувати
University.GroupI	d:\intersystems\cache\mgr\gorban\	Hi	Cyrillic3	VIEW Редагувати
University.PersonD	d:\intersystems\cache\mgr\gorban\	Hi	Cyrillic3	VIEW Редагувати
University.PersonI	d:\intersystems\cache\mgr\gorban\	Hi	Cyrillic3	VIEW Редагувати
University.SpecialtyD	d:\intersystems\cache\mgr\gorban\	Hi	Cyrillic3	VIEW Редагувати
University.SpecialtyI	d:\intersystems\cache\mgr\gorban\	Hi	Cyrillic3	VIEW Редагувати

Рис. 7.3. Список збережений у відповідній області даних

Після цього будуть відображені дані глобалу ^Univer, що був створений шляхом виконання описаної вище підпрограми CreateUniversityGlobal.



Маска пошуку глобалів: ^Univer | Показати | Скасувати

Історія пошуку: ^Univer | Максимальна кількість рядків: 100 | Дозволити редагування

```
1:  ^Univer                                          = "Чорноморський державний університет імені Петра Могили"
2:  ^Univer("Інститут філології")                    = $lb("Пронкевич О.В.","76-92-60")
3:  ^Univer("Медичний інститут")                     = $lb("Лебідь С.М.","76-55-54")
4:  ^Univer("Факультет економічних наук")            = $lb("Іщенко Н.М.","46-41-26")
5:  ^Univer("Факультет комп'ютерних наук")           = $lb("Мусієнко М.П.","76-55-74")
6:  ^Univer("Факультет комп'ютерних наук",101)       = "Бурлаченко І.С."
7:  ^Univer("Факультет комп'ютерних наук",101,1510101) = "Антонюк В.А."
8:  ^Univer("Факультет комп'ютерних наук",101,1510102) = "Бондаренко У.А."
9:  ^Univer("Факультет комп'ютерних наук",101,1510103) = "Головатий В.Р."
10: ^Univer("Факультет комп'ютерних наук",101,1510104) = "Доробанський М.Ю."
11: ^Univer("Факультет комп'ютерних наук",101,1510105) = "Задорожна О.А."
12: ^Univer("Факультет комп'ютерних наук",101,1510106) = "Іващенко С.В."
13: ^Univer("Факультет комп'ютерних наук",101,1510107) = "Каланкова А.С."
14: ^Univer("Факультет комп'ютерних наук",101,1510108) = "Кліменко Д.О."
15: ^Univer("Факультет комп'ютерних наук",101,1510109) = "Корбут В.В."
16: ^Univer("Факультет комп'ютерних наук",101,1510110) = "Лепетинський Е.Р."
17: ^Univer("Факультет комп'ютерних наук",101,1510111) = "Місюк Т.О."
18: ^Univer("Факультет комп'ютерних наук",101,1510112) = "Олейніченко Є.Є."
19: ^Univer("Факультет комп'ютерних наук",101,1510113) = "Осадчий А.О."
20: ^Univer("Факультет комп'ютерних наук",102)       = "Давиденко Є.О."
21: ^Univer("Факультет комп'ютерних наук",103)       = "Нездолій Ю.О."
22: ^Univer("Факультет комп'ютерних наук",105)       = "Крайник Я.М."
23: ^Univer("Факультет комп'ютерних наук",106)       = "Корецька О.С."
24: ^Univer("Факультет комп'ютерних наук",107)       = "Кулаковська І.В."
25: ^Univer("Факультет комп'ютерних наук",201)       = "Боровльова С.Ю."
26: ^Univer("Факультет комп'ютерних наук",202)       = "Швед А.В."
27: ^Univer("Факультет комп'ютерних наук",203)       = "Солобуто Л.В."
28: ^Univer("Факультет комп'ютерних наук",205)       = "Салтовський Б.Г."
29: ^Univer("Факультет комп'ютерних наук",206)       = "Горбань Г.В."
30: ^Univer("Факультет політичних наук")             = $lb("Шевчук О.В.","76-55-53")
31: ^Univer("Факультет соціології")                  = $lb("Ляпіна Л.А.","76-55-52")
32: ^Univer("Юридичний факультет")                   = $lb("Січко Д.С.","76-55-65")
```

Рис. 7.4. Дані глобалу ^Univer

Обробка даних глобалів

Мова Caché Object Script має достатньо гнучкий засіб обробки даних глобалів за допомогою вбудованих функцій $data, $get, $order та $query (див. лекцію № 8). Наведемо деякі приклади роботи з даними глобалів.

У вже створеній термінальній програмі University.Global.mac створимо іншу підпрограму під назвою GetStudents, призначення якої полягає в отриманні списку студентів, що належать до певної групи. Оскільки номер групи є індексом глобалу другого рівня, а індексом першого рівня є назва факультету, задамо для даної підпрограми відповідні аргументи: faculty та group. Програмний код підпрограми GetStudents наведений нижче.

```
GetStudents(faculty,group)
set x = ""
for
{
  set x=$order(^Univer(faculty,group,x))
  quit:x=""
  write x    //виведення на екран номеру залікової книжки студента у якості
індексу 3 рівня глобалу
  write " ",^Univer(faculty,group,x),! //виведення на екран значення
глобалу, яке зберігає у собі прізвище студента
}
```

У коді підпрограми використовується вбудована функція $order, що дозволяє отримати дані глобалу певного рівня за кожним з його індексів. У цьому випадку перші два індекси глобалу `^Univer` мають фіксовані значення, що відповідно вказані в аргументах `faculty` та group. Третім індексом глобалу `^Univer` є номер залікової книжки студента, а його значеннями на третьому рівні будуть прізвище та ініціали відповідного студента.

Почергово у циклі `for` (який до того ж оголошений як нескінченний) у змінну `x` будуть витягнуті всі індекси третього рівня, а саме номери залікових книжок студентів, після чого на екран буде виведено відповідне значення глобалу за отриманою сукупністю індексів.

Викликавши підпрограму GetStudents у Терміналі з параметрами «Факультет комп'ютерних наук» та 101 група у якості її аргументів, отримаємо такий результат.

Рис. 7.5. Виклик підпрограми GetStudents у Терміналі

Якщо глобал на певному рівні має більш ніж одне значення, воно представляється у вигляді списків. Для отримання таких значень треба скористатись вбудованими функціями роботи зі списками (див. лекцію № 6).

У якості прикладу напишемо підпрограму з назвою GetFacultiesInformation, яка виводить на екран інформацію про факультет: прізвище та ініціали його декана та телефон деканату. Нагадаємо, що зазначена інформація зберігається у глобалі ^Univer на першому рівні вкладеності. Індексом першого рівня вкладеності є назва факультету, а значення

складається зі списку, першим елементом якого є прізвище та ініціали декана, а другим – телефон деканату. При витягненні цих значень використовується вбудована функція $list.

Нижче наведено програмний код відповідної підпрограми.

```
GetFacultiesInformation
write "Факультет",?30,"Декан",?45,"Телефон",!
set x = ""
for
{
      set x = $order(^Univer(x))
      quit:x=""
      set inf = ^Univer(x)
      set dean = $listget(inf,1) //витяг ПІБ декана факультету
      set phone = $listget(inf,2) //витяг телефону деканату факультету
      write x     //виведення на екран назви факультету у якості індексу 1
рівня глобалу
      write ?30,dean,?45,phone,! //виведення інформації на екран
}
```

При виконанні цієї підпрограми у Терміналі буде отримано такий результат.

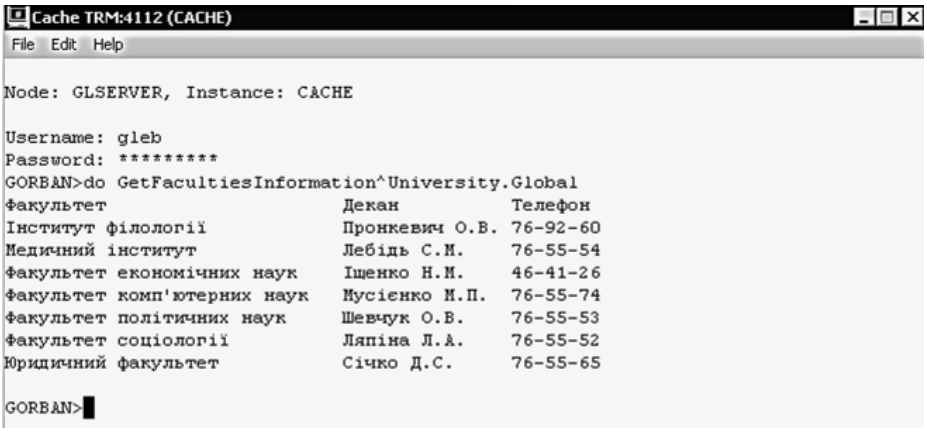


Рис. 7.6. Результат підпрограми

Збереження даних класів Caché у вигляді глобалів

Кожен збережений клас може зберігати свої екземпляри всередині бази даних, використовуючи один або кілька вузлів багатомірних даних (глобалів). Під час створення класу компілятором класу автоматично створюються визначення двох глобалів: глобали даних та індексні глобали. При цьому справедливим є таке:

– Дані зберігаються в глобалі, ім'я якого починається з повного імені класу, тобто пакет.клас, при цьому праворуч до імені глобала даних додається буква «D», до імені глобала індексу додається буква «I»;

– Дані кожного екземпляра зберігаються всередині єдиного вузла глобала даних, при цьому значення властивостей розміщаються в спискових структурах;

– Кожен вузол глобала даних має об'єктний ідентифікатор ID. За замовчуванням, значення ID це цілі числа, значення яких забезпечуються викликом функції $Increment;

При цьому для роботи з об'єктами можна використовувати або об'єктний доступ, або надзвичайно ефективний прямий доступ за допомогою глобалів.

Дані глобалів, що відповідають збереженим класам, також відображуються у вкладці **«Глобали»** у порталі управління системою. Розглянемо, наприклад, глобал класу University.Department, який представляє дані про кафедри університету.

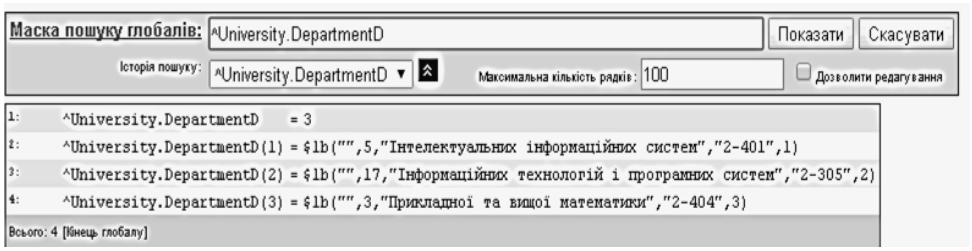


Рис. 7.7. Глобал класу University.Department

Якщо певний збережений клас унаслідується від іншого збереженого класу, який не є абстрактним взагалі, або є абстрактним, але тип класу вказаний як Persistent, то глобал даних та індексний глобал для класу-нащадка не створюються. Відповідні класи будуть створені тільки для батьківського класу.

Нижче представлені дані глобалу класу University.Person. Нагадаємо, що означений клас є абстрактним, але типом класу є persistent, тобто у реляційному відображенні буде створена його таблиця. При цьому власних екземплярів указаний клас не має, а у відповідній таблиці будуть відображені його нащадки: екземпляри класу University.Student та University.Teacher.

Рис. 7.8. Екземпляри класу University.Student та University.Teacher

Розглянемо детально дані глобалу `^University.PersonD`. У його корені зберігається ID останнього об’єкта, що був збережений у базі даних. При цьому нумерація ID є наскрізною і не залежить від того, до якого з нащадків класу Person відноситься той чи інший його

об'єкт. При виклику методів класу `%DeleteExtent` або `%KillExtent` у випадку повного знищення всього екстенту класу також будуть знищені його глобали: D та I.

Індексом глобалу на першому рівні є ID відповідного об'єкта, а його значенням – список значень відповідних властивостей, описаних у **базовому класі**. При цьому першим елементом цього списку буде повне ім'я класу-нащадка, до якого належить даний об'єкт. Усі інші елементи списку представляють собою значення простих властивостей. Якщо серед властивостей класу є вбудований клас, то у глобалі певним елементом списку значень буде вкладений список зі значеннями вбудованого об'єкта.

На другому рівні глобалу також у списку зберігаються дані властивостей, що були описані у класі-нащадку (наприклад, вчене звання для класу Teacher та номер залікової книжки для класу Student), а індексом глобалу на цьому рівні є назва класу-нащадка, до якого належить відповідний об'єкт. При цьому буде вказане не повне, а скорочене ім'я, тобто без назви схеми даних (пакета).

Пошук даних без використання SQL-запитів

Використання глобалів робить можливим пошук даних у класах, не використовуючи запити мовою SQL. Для цього потрібно здійснити проходження за всім екстентом класу. Для цього потрібно знати ID останнього збереженого об'єкта. Відповідно, він буде зберігатись у корені глобалу даних відповідного класу. У свою чергу, існування об'єктів у екстенті класу взагалі можна визначити, використавши вбудовану функцію $Data.

Далі намагаємось відкрити по черзі об'єкти з ID, що приймає значення від 1 до значення, що збережене у корені глобалу даних класу. При цьому потрібно перевірити, чи існує об'єкт з поточним ID взагалі. Якщо об'єкт з відповідним ID існує (посилання до нього не NULL), можна здійснювати перевірку цього об'єкта за критеріями, вказаними у запиті.

Якщо здійснюється пошук у класі, який є нащадком іншого класу, потрібно використати глобал даних базового класу і при проходженні всіх об'єктів ще перевіряти ім'я класу, до якого належить поточний об'єкт.

Нижче наведені приклади методів пошуку викладачів, що займають певну посаду та студентів, що належать до певної групи. Ці методи є членами класів Teacher та Student відповідно.

Пошук викладачів

```
ClassMethod FindTeacher() As %Status
{
 read !, "Введіть посаду викладача: ",academstat
 write "...пошук...",!
 if ($Data(^University.PersonD)) //якщо існують дані у глобалі класу
Person
 {
   set total = ^University.PersonD //у корні глобалу класу Person
зберігається ID об'єкта, що був збережений у БД останнім
   for i=1:1:total  //цикл за екстентом класу Person
   {
     set p = ##class(University.Person).%OpenId(i)
     if (p'="") //якщо об'єкт з даним ID існує і не був видалений раніше,
тобто посилання не порожнє
     {
```

```
        set classname = p.%ClassName() //знаходимо ім'я класу відкритого
об'єкту (ним є один з класів-нащадків: Student або Teacher)
        if (classname="Teacher") //якщо об'єкт належить до класу Teacher
        {
      if (p.Stat.NameStat=academstat)
write p.Surname,?15,p.Name,?25,p.Midname,?37,p.Depart.NameDepart,!
}
      }
   }
 }
}
```

Виклик методу:

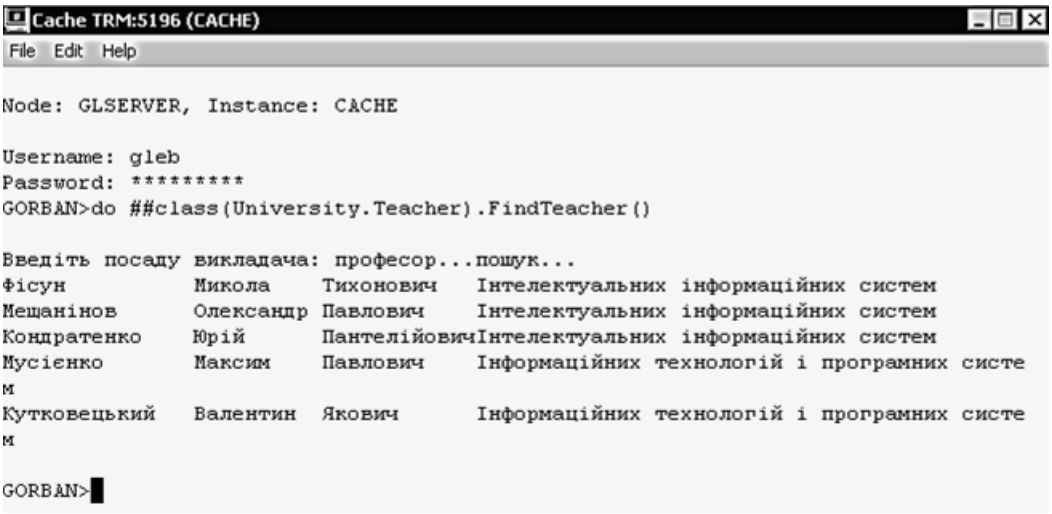


Рис. 7.9. Виклик методу FindTeacher

Пошук студентів

```
ClassMethod FindStudent() As %Status
{
 read !, "Введіть номер групи студента:", groupnum
 write "...пошук... ",!
 if ($Data(^University.PersonD)) //якщо існують дані у глобалі класу
Person
 {
   set total = ^University.PersonD //у корні глобалу класу Person
зберігається ID об'єкта, що був збережений у БД останнім
   for i=1:1:total  //цикл за екстентом класу Person
   {
     set p = ##class(University.Person).%OpenId(i)
     if (p'="") //якщо об'єкт з даним ID існує і не був видалений раніше,
тобто посилання не порожнє
     {
        set classname = p.%ClassName() //знаходимо ім'я класу відкритого
об'єкту (ним є один з класів-нащадків: Student або Teacher)
        if (classname="Student") //якщо об'єкт належить до класу Student
```

```
        {
       if (p.Group.GroupNumber=groupnum)
write p.Surname,?15,p.Name,?25,p.Midname,?37,p.Group.GroupNumber,!
        }
       }
    }
 }
}
```

Виклик методу:

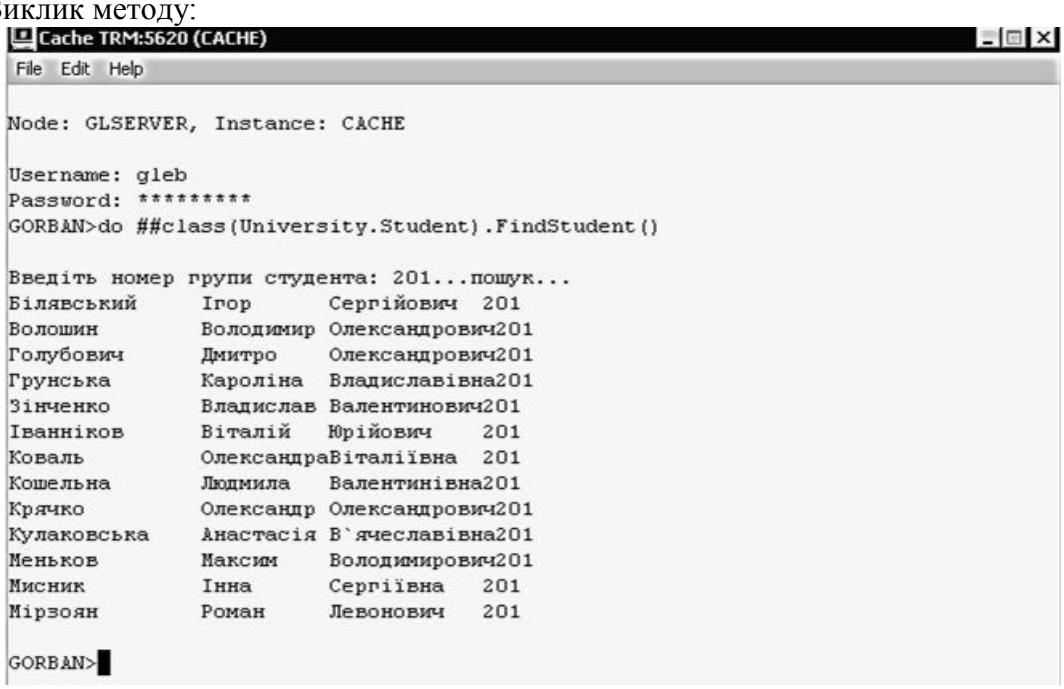


Рис. 7.10. Виклик методу FindStudent

Завдання

1. Відповідно до предметної області спроектувати та створити власний глобал більш ніж з одним рівнем, обґрунтувати кількість рівнів у ньому. Здійнити пошук даних у створеному глобалі.
2. Здійснити запит з лабораторної роботи № 5 без використання SQL.

Контрольні питання

1. У чому полягає ієрархічна модель даних в СКБД Caché?
2. Що таке глобал?
3. Як зберігається глобал у базі даних?
4. Які дії виконують функції $Data та $Get?
5. Які дії виконують функції $Order та $Query?
6. Як скопіювати багатомірну змінну?
7. Як представляються класи, їх підкласи та індекси у вигляді глобалів?
8. Як здійснити пошук об'єктів за критерієм без використання запитів SQL?

ЛАБОРАТОРНА РОБОТА № 8

«ВІДОБРАЖЕННЯ КЛАСІВ CACHÉ У JAVA. СТВОРЕННЯ ЗАСТОСУНКУ JAVA ДЛЯ РОБОТИ З ДАНИМИ В СКБД CACHÉ»

Взаємодія Java з СКБД Caché

Java – поширена об'єктно-орієнтована мова програмування, але, внаслідок свого основоположного принципу «write once, deploy anywhere», вона не специфікує, як треба записувати дані в БД і зчитувати їх з неї. Коли потрібно збереження стану даних, розробники застосунків мовою Java повинні вибрати і реалізувати методи доступу до даних.

Caché підтримує декілька способів звертання Java-застосунків до бази даних Caché:

– застосування JDBC (Java DataBase Connectivity) – платформо незалежного промислового стандарту взаємодії застосунків на Java з різними СУБД – забезпечує високоефективний SQL-доступ з використанням повністю заснованого на Java механізму з'єднання;

– будь-який клас Caché може бути відображений на Java-клас, тому можна отримувати доступ до властивостей і методів як до Java-об'єктів;

– технологія InterSystems Jalapeño™ створює класи Caché з класів Java. Caché автоматично надає методи зберігання і вибірки об'єктів з бази даних, не зачіпаючи класи з класів розробника застосунка;

– технологія Caché eXTreme for Java надає доступ до даних Caché з Java через Java Native Interface (JNI);

– використання технологій, що реалізують специфікацію Java Persistent API (JPA), у тому числі Hibernate і EclipseLink.

Створення Java-проекції класу Caché

Перш ніж клас Caché можна буде використовувати в Java, у його визначення варто додати відображення. Робиться це в Caché Studio подібно тому, як створюються нові методи й властивості: або в меню, що відкривається правою кнопкою миші в контексті вихідного коду класу, або в меню **Class** вибирається опція **Add** і далі **Projection**.

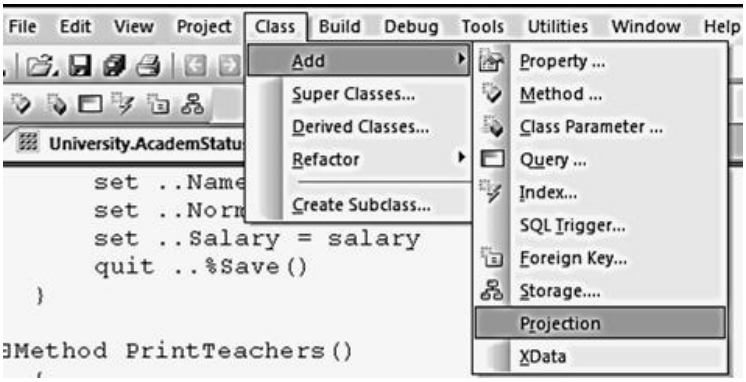


Рис. 8.1. Створення Java-проекції класу Caché

На наступному кроці потрібно вказати назву проекції.

Наприклад:

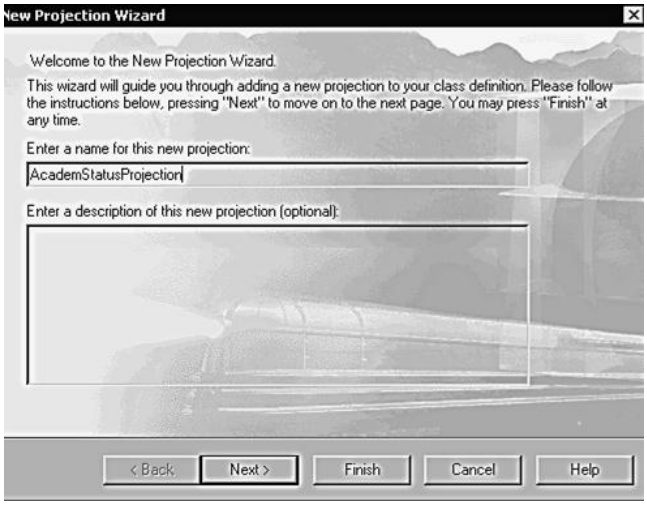


Рис. 8.2. Вікно створення Java-проекції

Класи відображення, як правило, мають параметри, що впливають на процес відображення. Параметру `ROOTDIR` класу типу `%Projection.Java` повинне бути привласнене ім'я каталогу, у якому будуть створюватися файли Java. Класи Java, що генеруються, запозичують ім'я пакета від відображуваного класу Caché. Для зручності треба помістити згенеровані файли Java у папці, де зберігається відповідна база даних: `D:\Intersystems\Cache\mgr\<папка з назвою області даних>`.

Для прикладу, представленого в цій лабораторній роботі, згенеровані файли вихідного коду згенерованих класів Java будуть міститись у папці `D:\Intersystems\Cache\mgr\gorban`. Відповідно, кожному зі студентів потрібно буде розмістити класи Java у папці, де зберігається власна база даних (папка має ідентичну назву з областю даних).

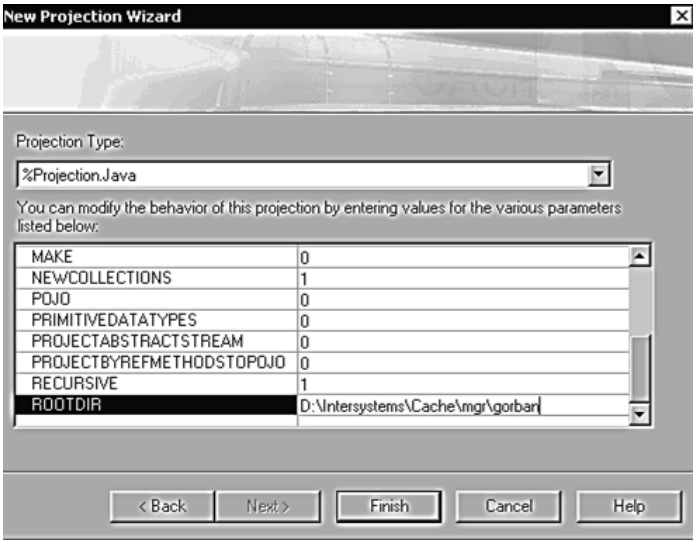


Рис. 8.3. Завдання папки, де будуть розміщені файли Java-проекцій

Варто також нагадати про те, що у даному випадку шлях до відповідних файлів Java `D:\Intersystems\Cache\mgr\<папка з назвою області даних>` буде означати, що вони будуть розташовані на диску D комп'ютера, на якому встановлений сервер Caché, тобто GLSERVER, який має IP-адресу 192.168.96.179.

Дістатися до згенерованих файлів можна за допомогою мережевої адреси `glserver\\Cache\mgr\<папка з назвою області даних>`, оскільки у кожного студента є доступ до папки, де встановлено сервер Caché. Крім того, у відповідній папці буде створена папка з відповідним пакетом, що відповідає схемі даних (у цьому прикладі University), у якій у свою чергу і буть розташовуватись необхідні файли з кодом Java.

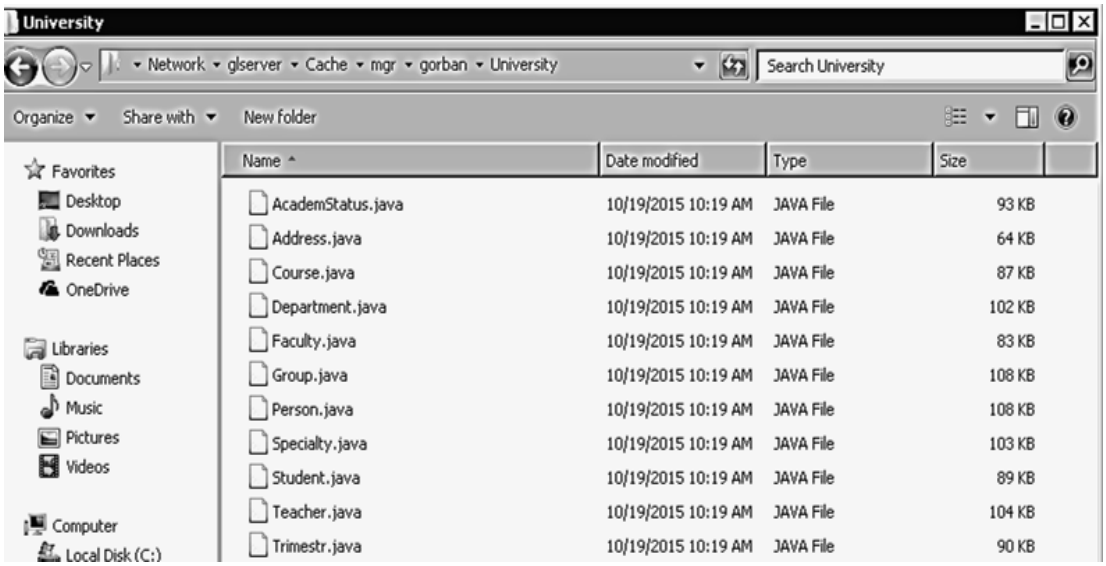


Рис. 8.4. Папка з відповідним пакетом

Разом з Caché поставляється набір службових класів Java, що працюють спільно зі згенерованими Java-класами. Вони забезпечують прозорий доступ до об'єктів, збережених на сервері Caché.

Службові класи Java упаковані в архів `cachedb.jar`. Унікальність засобів взаємодії Caché з Java проявляється в тому, що вони дозволяють поєднувати об'єктний доступ до методів і властивостей об'єктів з реляційним доступом через JDBC. Із цієї причини класи, що обслуговують об'єктну взаємодію, тісно пов'язані із класами стандартизованої архітектури.

Файл `cachedb.jar` можна знайти у папці `glserver\\Cache\dev\java\lib\JDK17`, у якій також можна знайти і інші jar-файли для роботи зі СКБД Caché, зокрема файл `cachejdbc.jar`, класи якого призначені для реляційного доступу до даних Caché, використовуючи JDBC.

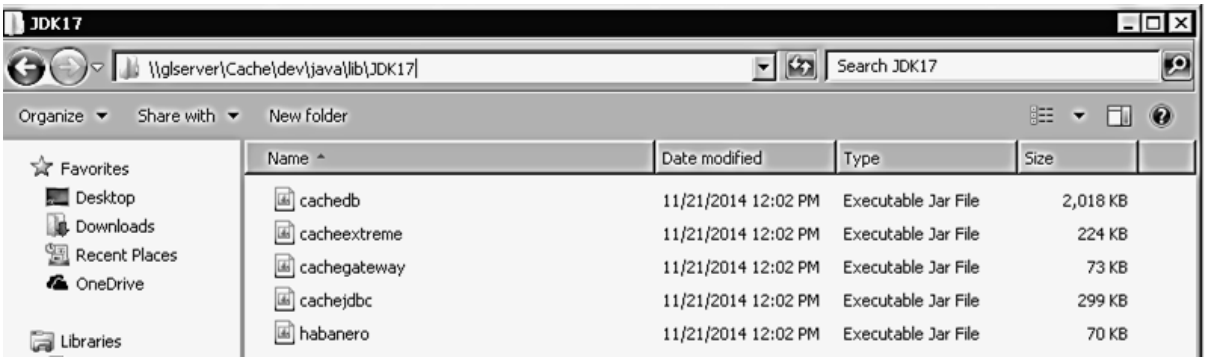


Рис. 8.5. Файл `cachedb.jar`

Для виконання лабораторної роботи потрібно створити новий проект Java (достатньо буде створити звичайний проект Java SE) та додати до нього згенеровані Java-проекції класів Caché та до бібліотек проекту додати файли `cachedb.jar` та `cachejdbc.jar`, без якого не можна буде працювати зі згенерованими класами Java.

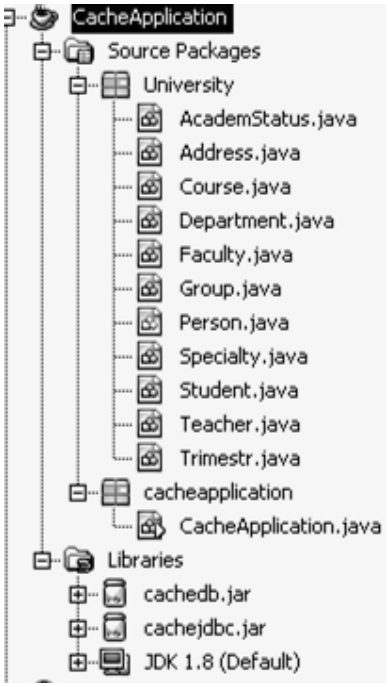


Рис. 8.6. Створення нового проекту Java

Далі потрібно додати до проекту новий клас, у якому буде метод `public static void main(String[] args)` та додати у створений клас наступні імпорти:

```
import University.*;
import com.intersys.objects.CacheDatabase;
import com.intersys.objects.CacheException;
import com.intersys.objects.CacheQuery;
import com.intersys.objects.Database;
import com.intersys.objects.Id;
import java.sql.ResultSet;
import java.sql.SQLException;
```

Таким чином, ми імпортували до проекту всі згенеровані класи Java-проекцій класів Caché з пакета University, а також наступні службові класи:

– `com.intersys.objects.Database` – клас, що визначає методи інтерфейсу з базою даних Caché;

– `com.intersys.objects.CacheDatabase` – клас, що створює нові об'єкти, які пов'язані зі з'єднанням із сервером Caché;

– `com.intersys.objects.CacheException` – оброблювач виняткових ситуацій, що реагує на помилки взаємодії із базою даних Caché;

– `com.intersys.objects.Id` – клас, що відповідає унікальному *Id* об'єкта класу, що зберігається;

– `com.intersys.objects.CacheQuery` – клас, що визначає методи інтерфейсу із запитами, що визначені у класах Caché.

Останній описаний клас також буде пов'язаний з класами `java.sql.ResultSet` та `java.sql.SQLException`, які також були імпортовані до нового проекту.

З'єднання з базою даних Caché

Для створення нового з'єднання з БД додамо до головного класу наступну властивість:

```
static Database db;
```

Після чого створюємо метод createConnection(), який повертає екземпляр класу Database.

```
public static Database createConnection() throws CacheException {
    //формуємо URL до бази даних Cache, що складається з IP-адоеси
сервера, порту (1972)
    //та назви відповідної області даних
    String url = "jdbc:Cache://192.168.96.179:1972/GORBAN";
    //вказуємо ім'я користувача та пароль
    String username = "gleb";
    String password = "_________";
    //створюємо з'єднання
    Database db = CacheDatabase.getDatabase(url,username,password);
    return db;
}

public static void main(String[] args) {
   // TODO code application logic here
   try {
      db = createConnection();
     //тут буде виклик методів, описаних нижче
   }
    catch (CacheException e) {
       System.out.println(e.getMessage());
    }
}
```

Робота з об'єктами Caché, що зберігаються у БД

Крім властивостей і методів, визначених користувачем, у згенерованих класах присутні методи, що реалізують читання, запис та видалення об'єкта:

1) метод класу `_open(Database db, Id id)` – повертає посилання на об'єкт відповідного класу;

2) метод екземпляра класу `_save()` – зберігає об'єкт в БД і повертає статус;

3) метод класу `_deleteId(Database db, Id id)` – видаляє об'єкт з відповідним ID із БД.

Відкриття та перегляд даних існуючого у БД об'єкта

Створимо метод, який виводить інформацію про певного викладача. Зазначений метод має аргумент – id відповідного викладача у БД.

У методі виводимо інформацію про прізвище, ім'я та по батькові викладача за допомогою стандартних гетерів відповідних властивостей: `getSurname()`, `getName()` та `getMidname()`.

Також при генерації відповідних Java-проекцій класів Caché створюються і гетери властивостей, що представляють собою посилання та зв'язки з об'єктами, що належать до інших класів. Так, для отримання об'єкта посади викладача (AcademStatus) та кафедри, до якої викладач належить (Department), існують гетери `getStat()` та `getDepart()` відповідно, оскільки клас Teacher має властивості Stat та Depart.

У свою чергу для об'єктів класів AcademStatus та Department існують гетери їх властивостей.

```
public static void getTeacherInformation(int id) {
  try {
     //відкриваємо з БД об'єкт відповідного до id викладача, приводимо
об'єкт до типу Teacher
Teacher teacher = (Teacher)(Teacher._open(db, new Id(id)));

    System.out.println("Прізвище: " + teacher.getSurname() + "\n" +
          "Ім'я: " + teacher.getName() + "\n" +
          "По-батькові: " + teacher.getMidname() + "\n" +
          "Посада: " + teacher.getStat().getNameStat() + "\n" +
          "Кафедра: " + teacher.getDepart().getNameDepart() + "\n");
    //закриваємо відповідний об'єкт класу Teacher у базі даних
    db.closeObject(teacher.getOref());
  }
  catch (CacheException e){
     System.out.println(e.getMessage());
  }
}
```

Викличемо даний метод у методі `main`, вказавши id існуючого у БД об'єкта:

```
getTeacherInformation(1);
```

та запустимо проект. Ми отримаємо такий результат:

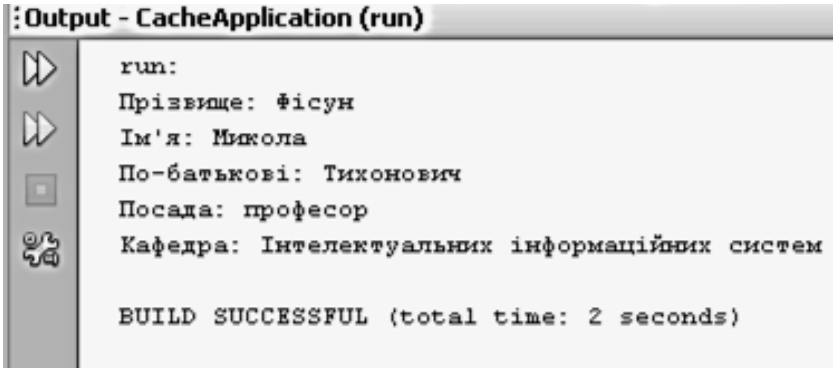


Рис. 8.7. Метод `getTeacherInformation`

Створення та збереження нового об'єкта у БД

Створення об'єкта і наповнення його властивостей виконується за допомогою стандартного синтаксису Java (POJO). У конструктор об'єкта передається посилання на підключення до БД.

Напишемо метод, що створює новий об'єкт класу Teacher, заповнює його властивості значеннями, задає його зв'язки з об'єктами класів, а потім зберігає його у БД. Для заповнення властивостей об'єкта викликаються стандартні сетери: setSurname(), setName() і т. ін. Також існують сетери для властивостей посилань та зв'язків, однак їх аргументи представляють собою посилання на об'єкти. Так ми відкриємо існуючі об'єкти класів AcademStatus та Department за допомогою їх id, заданих, як параметри методу, що розглядаються.

```
public static void createNewTeacher(String surname,String name, String
midname,int idStat,int idDepart) {
  Teacher newTeacher = null;

  try {
    newTeacher = new Teacher(db);
    //встановлення значень простих властивостей
    newTeacher.setSurname(surname);
    newTeacher.setName(name);
    newTeacher.setMidname(midname);
     //відкриття об'єктів класів AcademStatus та Department
    AcademStatus status = (AcademStatus)(AcademStatus._open(db, new
Id(idStat)));
    Department department = (Department)(Department._open(db, new
Id(idDepart)));
     //встановлення значень властивостей посилань
    newTeacher.setDepart(department);
    newTeacher.setStat(status);

    //спроба зберегти об'єкт у БД
    if (newTeacher._save()!=1)
    {
       System.out.println("При спробі збереження об'єкта виникла
проблема!");
    }
    else
    {
       System.out.println("Об'єкт успішно збережений!");
    }
    db.closeObject(newTeacher.getOref());
 }
 catch (CacheException e){
    System.out.println(e.getMessage());
 }
}
```

Викличемо даний метод у методі `main`

```
createNewTeacher("Горбань","ГЛІБ","Валентинович",5,1);
```

та запустимо проект. Ми отримаємо такий результат:

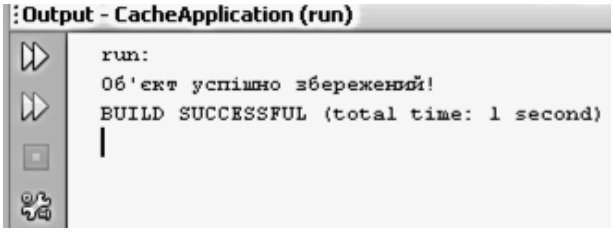
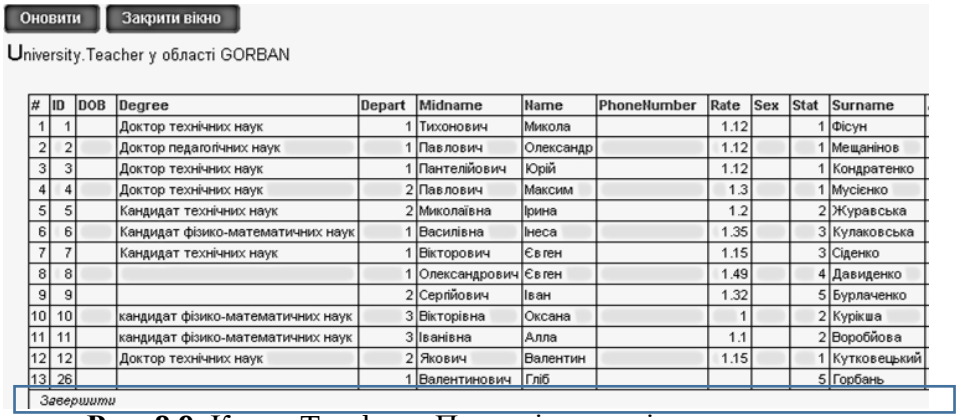


Рис. 8.8. Метод `createNewTeacher`

Відкривши дані у реляційному відображенні класу Teacher у Порталі управління системою, ще раз можна переконатись, що новий об'єкт був створений.



#	ID	DOB	Degree	Depart	Midname	Name	PhoneNumber	Rate	Sex	Stat	Surname
1	1		Доктор технічних наук	1	Тихонович	Микола		1.12		1	Фісун
2	2		Доктор педагогічних наук	1	Павлович	Олександр		1.12		1	Мещанінов
3	3		Доктор технічних наук	1	Пантелійович	Юрій		1.12		1	Кондратенко
4	4		Доктор технічних наук	2	Павлович	Максим		1.3		1	Мусієнко
5	5		Кандидат технічних наук	2	Миколаївна	Ірина		1.2		2	Журавська
6	6		Кандидат фізико-математичних наук	1	Василівна	Інеса		1.35		3	Кулаковська
7	7		Кандидат технічних наук	1	Вікторович	Євген		1.15		3	Сіденко
8	8			1	Олександрович	Євген		1.49		4	Давиденко
9	9			2	Сергійович	Іван		1.32		5	Бурлаченко
10	10		кандидат фізико-математичних наук	3	Вікторівна	Оксана		1		2	Курікша
11	11		кандидат фізико-математичних наук	3	Іванівна	Алла		1.1		2	Воробйова
12	12		Доктор технічних наук	2	Якович	Валентин		1.15		1	Кутковецький
13	26			1	Валентинович	Гліб				5	Горбань

Рис. 8.9. Класу Teacher у Порталі управління системою

Виконання запиту, описаного у класі Caché, що зберігається

Caché автоматично генерує методи для кожного методу-запиту в класі і додає їх до проекції. Для роботи з такими методуми використовується клас `com.intersys.objects.CacheQuery`. Результат виконання запиту повертається в об'єкт стандартного класу `java.sql.ResultSet`, який може генерувати виняткову ситуацію `SQLException`.

Напишемо метод, який виконує описаний у класі Teacher запит FindBySurname (див. ЛР № 5). Але для того, щоб його можна було виконати у Java, сам запит потрібно позначити ключовим словом SqlProc. Тим самим ми дозволимо його виконання засобами JDBC.

```
Query FindBySurname(surname As %String) As %SQLQuery [SqlProc]
{
SELECT %ID,Surname,Name,Midname,Depart->NameDepart FROM Teacher
WHERE (Surname %STARTSWITH :surname)
ORDER BY Surname
}
```

Далі наводиться код методу, який оброблює цей запит. Його параметром, як і для запиту, є прізвище викладача (або його частина).

```
public static void findTeacher(String surname) {
 try {
    CacheQuery query = new
CacheQuery(db, "University.Teacher","FindBySurname");
    ResultSet rs = query.execute(surname);
```

```
    try {
      while (rs.next()) {
        for (int i=0; i<rs.getMetaData().getColumnCount(); i++)
{
           System.out.print(rs.getString(i+1) + " ");
        }
        System.out.println();
      }
      rs.close();
      query.close();
    }
    catch (SQLException e) {
        System.out.println(e.getMessage());
    }
 }
 catch (CacheException e){
    System.out.println(e.getMessage());
 }
}
```

Викличемо цей метод у методі `main`

```
findTeacher("К");
```

та запустимо проект.

Ми отримаємо наступний результат:

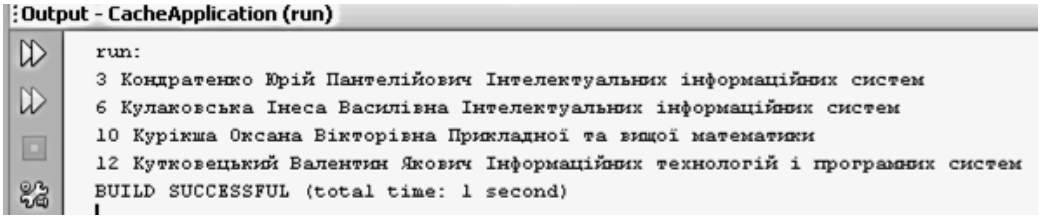


Рис. 8.10. Метод `findTeacher`

Завдання

1. Знегерувати Java-проекції класів Caché з власної бази даних (усіх або деяких класів на власний розсуд). Помістити їх у папку, де зберігається БД.
2. Створити проект Java для роботи з власною БД Caché. Мовою Java написати:

- метод, що виводить інформацію про певний об’єкт, що існує у БД;
- метод, який створює новий об’єкт та зберігає його у БД;
- метод, який виконує існуючий SQL-запит у класі Caché.

Класи з БД обрати на власний розсуд.

Контрольні питання

1. Які компоненти забезпечують взаємодію Caché з Java?
2. За якими правилами клас Caché відображується у Java?
3. Які службові Java-класи для СКБД Caché ви знаєте?
4. Як створити екземпляр класу Caché у Java?
5. Як відображуються множинні властивості класів Caché у Java?

ЛАБОРАТОРНА РОБОТА № 9

«СТВОРЕННЯ WEB-ЗАСТОСУНКІВ ДО СКБД CACHÉ ЗА ДОПОМОГОЮ ТЕХНОЛОГІЇ CSP (CACHÉ SERVER PAGE)»

Теоретичні відомості

Технологія CSP (Cache Server Pages) – основний інструмент створення Web-інтерфейсу для інформаційних застосунків, написаних на Cache. Технологія CSP пропонує витончені засоби створення швидкодійних, добре масштабованих Web-застосунків за короткий час. Вона також спрощує подальший супровід і розвиток таких застосунків.

На фазі виконання, архітектура CSP застосування складається з наступних елементів (рис. 1):

- **HTTP клієнт** (звичайно, web browser);
- **HTTP сервер** (web-сервер, наприклад, Apache або IIS);
- **CSP шлюз;**
- **Caché Server** на якому виконуються запити до застосування.

Коли HTTP клієнт, наприклад, Web-браузер, відправляє повідомлення на сервер, формується HTTP запит. Життєвий цикл цього запиту полягає в наступному (рис. 21):

1. Барузер відправляє HTTP запит (request).
2. Web сервер визначає, що отриманий запит – CSP запит і переправляє його до CSP шлюзу (встановленому на Web сервері).

2.a. Web сервер визначає, що користувачу потрібно відправити статичну сторінку і на цьому обробка запиту припиняється.

3. CSP шлюз перепаковує у внутрішній формат запит і пересилає його на відповідний Caché сервер.
4. Caché сервер оброблює запит і визначає, що потрібно повернути CSP клас. У цьому випадку сервер знаходить необхідний клас і запускає на виконання метод OnPage цього класу.

4.a. Caché сервер оброблює запит і визначає, що користувачу потрібно повернути статичний файл (наприклад, *.html або *.jpg). У цьому випадку сервер знаходить файл у локальній файловій системі і відправляє його назад у шлюз.

5. Результат виконання методу OnPage відправляється у шлюз.
6. У шлюзі формується відповідь (response) та відправляється на Web сервер.
7. Web-сервер формує HTTP відповідь та передає у браузер, який оброблює цю відповідь, і якщо вона задана у форматі html – відображає.

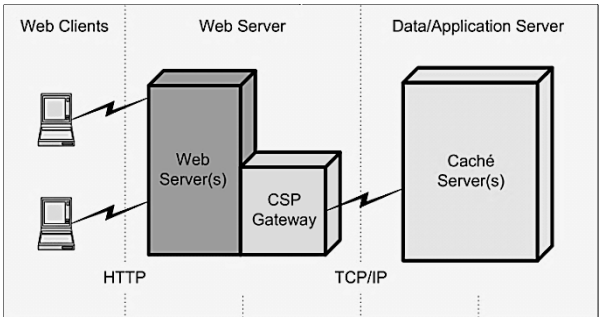


Рис. 9.1. Платформа Cache Server Pages

URL для CSP сторінок формується за правилами:

```
http://<ip-адреса сервера Caché>:<порт Web-сервера>/csp/<назва області даних>/<назва CSP-сторінки>
```

Наприклад: *http://localhost:57772/csp/samples/menu.csp*
де localhost – ім'я сервера,
57772 – порт сервера,
csp – віртуальна директорія, у якій зберігаються сторінки та дані,
samples – ім'я пакета,
menu.csp – ім'я сторінки.

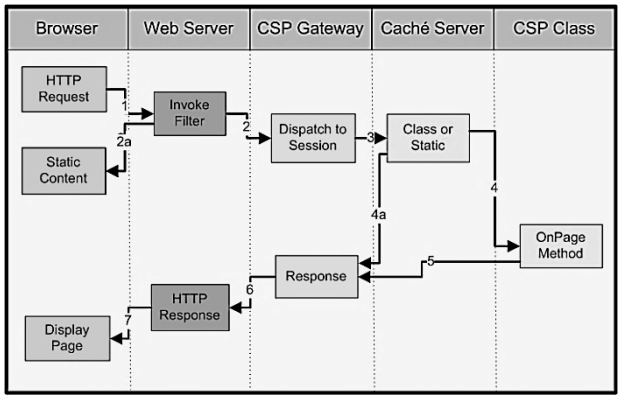


Рис. 9.2. Життєвий цикл http-запиту

Є два шляхи написання CSP сторінок:

1) створення Caché класу, який наслідується від системного класу *%CSP.Page*. Програмування в даному випадку в основному полягає в написанні методів, що наслідуються від *%CSP.Page*. Клас міститься у файлі з розширенням *.cls.

2) створення сторінок, що поєднують мову розмітки HTML та CSP, а також включають сценарії для виконання на мовах COS та CB. CSP-компілятор автоматично транслює таку сторінку у Caché класу перед компіляцією. Сторінка має зберігатися у файлі з розширенням *.csp.

Створення класу, що наслідується від класу *%CSP.Page*

Для створення такого класу необхідно зайти до майстра (File -> New -> ClassDefinition) та вказати його тип – CSP, а потім вказати тип контенту – html (також можливий тип контенту – xml).

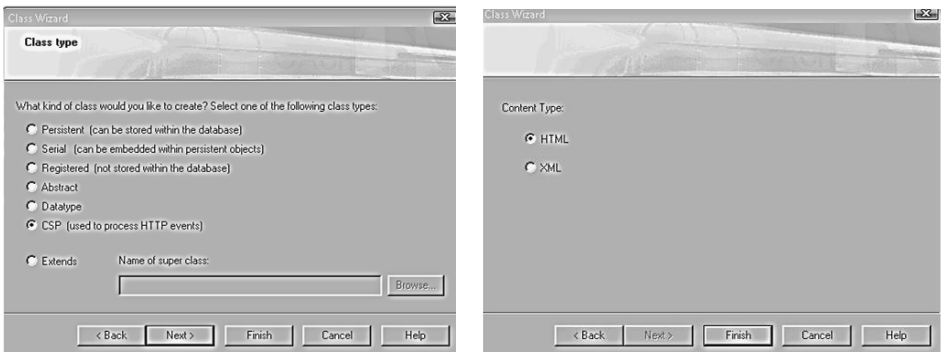


Рис. 9.3. Створення класу, що наслідується від класу *%CSP.Page*

Після створення такого класу ми побачимо наступний код:

```
Class User.NewCsp Extends %CSP.Page
{

ClassMethod OnPage() As %Status
{
     &html<<html>
<head>
</head>
<body>>
     ; To do...
     &html<</body>
</html>>
     Quit $$$OK
}

}
```

Як видно з коду цього класу, він є наслідуваним від базового класу всіх CSP-сторінок, який має назву *%CSP.Page*. Від базового класу новий клас наслідує метод класу OnPage, який відповідає за зовнішній вигляд сторінки.

У тілі методу OnPage можна побачити вбудований оператор html, що має вигляд **&html** та дозволяє описувати теги html. При цьому опис тегів відкривається додатковим символом <, а закривається відповідним додатковим символом >.

Далі у методі пишеться звичайний код мовою Caché Object Script (замість **`; To do...`** в описаному вище прикладі.

Створення нової CSP-сторінки

Однак найбільш поширеним способом є створення власне CSP-сторінки. Найпростіше це можна зробити шляхом виклику контекстного меню **«Create New CSP File»** для пункту **«CSP File»** у робочому середовищі (Workspace).

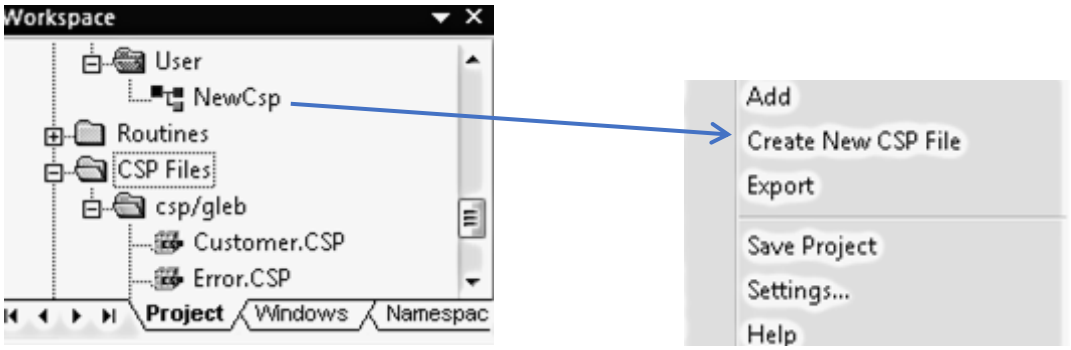


Рис. 9.4. Створення нової CSP-сторінки

Ми отримаємо наступну заготовку для коду, яка вже більш нагадує html-сторінку. Свій код можна писати на місці **`My page body`**.

```
<html>
<head>

<!-- Put your page Title here -->
<title>    Cache Server Page </title>

</head>

<body>

      <!-- Put your page code here -->
      My page body
</body>
</html>
```

Програмування CSP-застосунку

Для бази даних University створимо простий CSP-застосунок, що складається з 3 сторінок.

На першій сторінці, що має назву `TeachersInformation.csp` буде виведена таблиця зі списком викладачів (об'єктів класу Teacher), які збережені у базі даних. При цьому остання колонка таблиці буде містити посилання на сторінку форми редагування відповідного викладача, яка має назву `EditTeacherForm.csp`.

Друга сторінка, крім текстових полів для редагування властивостей, також містить кнопку, при натисненні на яку відбувається оновлення даних та перехід на третю сторінку, яка повідомляє про те, що дані були успішно оновлені або навпаки, не були оновлені, якщо виникла певна помилка.

Скріншот сторінки `TeachersInformation.csp` зображений на рис. 9.4.

Список викладачів

Прізвище	Ім'я	По-батькові	Посада	Назва кафедри	
Фісун	Микола	Тихонович	професор	Інтелектуальних інформаційних систем	Редагувати
Мещанінов	Олександр	Павлович	професор	Інтелектуальних інформаційних систем	Редагувати
Кондратенко	Юрій	Пантелійович	професор	Інтелектуальних інформаційних систем	Редагувати
Мусієнко	Максим	Павлович	професор	Інформаційних технологій і програмних систем	Редагувати
Журавська	Ірина	Миколаївна	доцент	Інформаційних технологій і програмних систем	Редагувати
Кулаковська	Інеса	Василівна	старший викладач (к.т.н.)	Інтелектуальних інформаційних систем	Редагувати
Сіденко	Євген	Вікторович	старший викладач (к.т.н.)	Інтелектуальних інформаційних систем	Редагувати
Давиденко	Євген	Олександрович	старший викладач	Інтелектуальних інформаційних систем	Редагувати
Бурлаченко	Іван	Сергійович	викладач	Інформаційних технологій і програмних систем	Редагувати
Курікша	Оксана	Вікторівна	доцент	Прикладної та вищої математики	Редагувати
Воробйова	Алла	Іванівна	доцент	Прикладної та вищої математики	Редагувати
Кутковецький	Валентин	Якович	професор	Інформаційних технологій і програмних систем	Редагувати
Горбань	Гліб	Валентинович	викладач	Інтелектуальних інформаційних систем	Редагувати

Рис. 9.5. Сторінка `TeachersInformation.csp`

Наведемо програмний код цієї сторінки та пояснимо його.

На самому початку використовуються вже знайомі теги html (<html>,<head>, <title>,<body>).

```
<html>
<head>
<title> List of teachers </title>
</head>
<body>
Тепер виводимо дані про всіх викладачів у таблицю. Для цього
використовуємо скрипт на мові Caché Object Script, який виконується на
боці сервера.
<h1 align="center">Список викладачів</h1>
 <script language="cache" runat="server">
```

Спочатку формуємо шапку таблиці з заголовками певних властивостей класу Teacher (Прізвище, Ім'я, По-батькові, Посада, Назва кафедри). Це буде перший рядок таблиці.

```
write "<table align=""center"" border=""2""><tr><th>Прізвище</th>"
write "<th>Ім'я</th><th>По-батькові</th>"
write "<th>Посада</th><th>Назва кафедри</th><th></th></tr>"
```

Усі інші рядки таблиці містять власне об'єкти. Для їх виведення скористаємось динамічним запитом шляхом створення нового екземпляра класу ResultSet (див. Лекція № 7, ЛР № 5). У якості альтернативного варіанта можна скористатись вбудованим SQL або запитом без використання SQL (див. ЛР. № 6).

```
set rset=##class(%ResultSet).%New()
do rset.Prepare("SELECT %ID,Surname,Name,Midname,Stat->NameStat,Depart-
>NameDepart FROM University.Teacher")
do rset.Execute()
```

Поки існують рядки у запиті, по черзі формуємо нові рядки таблиці, застосувавши тег <tr>. Дві останні колонки у таблиці представляють собою посилання на CSP-сторінки редагування та видалення об'єкта відповідно. При цьому на них буде передаватись один параметр – id об'єкта, значення якого вказується у гіперпосиланні після назви сторінки, на яку здійснюється перехід. Цей параметр буде переданий сторінкам методом get.

```
while rset.Next()
{
      set surname=rset.Data("Surname»)
      write "<td>",surname, "</td>"
      set name=rset.Data("Name")
      write "<td>",name, "</td>"
      set midname=rset.Data("Midname")
      write "<td>",midname,"</td>"
      set nameStat=rset.Data("NameStat")
      write "<td>",nameStat,"</td>"
      set nameDepart=rset.Data("NameDepart")
      write "<td>",nameDepart,"</td>"
      set id=rset.Data("ID")
      write "<td><a
href=""EditTeacherForm.csp?id=",id,""">Редагувати</a></td>"
}
```

Після виведення даних про всі об’єкти, ми відповідно закриваємо запит та таблицю, потім закриваємо скрипт, після чого закриваємо видиму частину сторінки та документ загалом, написавши відповідні закриваючі теги `</body>` та `</html>`.

```
    do rset.Close()
 write "</table>"
 </script>
</body>
</html>
```

Сторінка `EditTeacherForm.csp` містить форму редагування властивостей об’єкта класу Teacher. Її скріншот зображено на рис. 4.

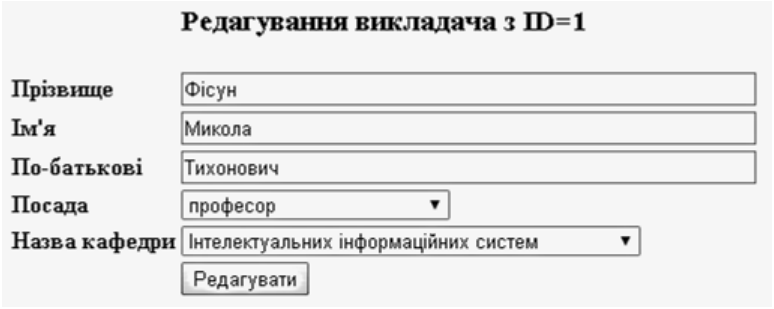


Рис. 9.6. Сторінка EditTeacherForm.csp

Також наведемо код указаної сторінки з поясненнями, де це потрібно.

```
<html>
<head>
<!-- Put your page Title here -->
<title> Edit Teacher Form </title>
</head>
<body>
```

Як вже було зазначено вище, на дану сторінку зі сторінки `TeachersInformation.csp` передається параметр id, який ми зчитаємо за допомогою об’єкта `%request`.

Наступний код означає прив’язку відповідного об’єкта до форми та відповідне оголошення форми.

```
<h3> Редагування викладача з ID=#(%request.Get("id"))#</h3>
<csp:object name="teacher" classname="University.Teacher" OBJID="#(%reques
t.Get("id"))#">
```

Останній рядок означає відкриття існуючого об’єкта класу Teacher у змінну `teacher`. Даний код представляє CSP-аналогію коду:

```
set teacher=##class(University.Teacher).%OpenId(%request.Get("id"))
```

Наступний рядок коду CSP-сторінки означає прив'язку відкритого об'єкта до форми. Далі створюємо поля для редагування форми, у які поміщаємо значення відповідних властивостей об'єкта `teacher`.

```
<form name="editform" cspbind="teacher" action="EditTeacher.csp">
<table align="center">

<tr><td><font color="green"><b>Прізвище</b></td>
   <td><input type="text" name="Surname" cspbind="Surname" size="50" cspre
quired></td>
</tr>

<tr><td><font color="green"><b>Ім'я</b></td>
   <td><input type="text" name="Name" cspbind="Name" size="50" csprequired
></td>
</tr>

<tr><td><font color="green"><b>По-батькові</b></td>
   <td><input type="text" name="Midname" cspbind="Midname" size="50" cspre
quired></td>
</tr>
```

Атрибут `csprequired` означає, що відповідне текстове поле не має бути порожнім.

Далі виводимо на форму назву посади викладача та назву кафедри, до якої він належить. На відміну від вже розглянутих властивостей, ці властивості представляють собою зв'язки об'єкта класу Teacher з об'єктами класів AcademStatus та Department відповідно.

На CSP-сторінку ми виводимо два списки, що випадають значеннями яких будуть усі можливі назви посад та кафедр, однак значеннями, що буде передаватись з форми на іншу сторінку, будуть ID відповідних об'єктів.

Виведемо описані вище значення зв'язків на сторінку за допомогою скрипта мовою Caché Object Script.

```
<script language="cache" runat="server">
 write "<tr><td><font color=""green""><b>Посада</b></td>"
 write "<td><select name=""Stat"">"

 set rset=##class(%ResultSet).%New()
 do rset.Prepare("SELECT * FROM University.AcademStatus")
 do rset.Execute()

 while rset.Next()
 {
        set idstat=rset.Data("ID")
        set namestat=rset.Data("NameStat")
        write idstat, " ",namestat
        write "<option value=""",idstat,""""
        if teacher.Stat.%Id()=idstat write " selected"
        write ">",namestat, "</option>"
 }
```

```
 do rset.Close()
 write "</select>"
 write "</td></tr>"

 write "<tr><td><font color=""green""><b>Назва кафедри</b></td>"
 write "<td><select name=""Depart"">"

 set rset=##class(%ResultSet).%New()
 do rset.Prepare("SELECT * FROM University.Department")
 do rset.Execute()

 while rset.Next()
 {
        set iddepart=rset.Data("ID")
        set namedepart=rset.Data("NameDepart")
        write "<option value=""",iddepart, """"
        if teacher.Depart.%Id()=iddepart write " selected"
        write ">",namedepart, "</option>"
 }
 do rset.Close()

 write "</select>«
 write "</td></tr>«
    write "<input type=""hidden""name=""Id""value=""",teacher.%Id(),""">"
</script>
```

Створюємо кнопку типу «submit», при натисненні на яку ми перейдемо на сторінку EditTeacher.csp, вказану в атрибуті «action» тегу «form».

```
<tr><td></td><td><input align="center" type="submit" name="Edit" value=
"Редагувати"></td></tr>
</table>

</form>
</body>
</html>
```

Третя сторінка, що має назву `EditTeacher.csp`, містить сценарій, який зчитує дані з форми, відкриває відповідний об'єкт класу `Teacher` та присвоює нові значення його властивостям, які відповідно були зчитані з форми. Із цієї сторінки через гіперпосилання можна повернутись до першої сторінки (`TeachersInformation.csp`).

Інформація про викладача з ID=26 успішно збережена.
Повернутись до списку викладачів

Рис. 9.7. Сторінка EditTeacher.csp

```
<html>
<head>
<!-- Put your page Title here -->
<title> Edit teacher </title>
```

```
</head>
<body>
     <script language="cache" runat="server">
     set id=%request.Get("Id")
     set surname=%request.Get("Surname")
     set name=%request.Get("Name")
     set midname=%request.Get("Midname")
     set idstat=%request.Get("Stat")
     set iddepart=%request.Get("Depart")

     set teacher=##class(University.Teacher).%OpenId(id)
     set teacher.Surname=surname
     set teacher.Name=name
     set teacher.Midname=midname
     set status=##class(University.AcademStatus).%OpenId(idstat)
     set department=##class(University.Department).%OpenId(iddepart)

     set teacher.Stat=status
     set teacher.Depart=department
     set sc=teacher.%Save()
     if (sc=1) write "Інформація про викладача з ID=",id," успішно
збережена.",!
     else  write "Помилка! Інформацію про викладача з ID=",id," зберегти не
вдалося. Причина - ",$System.OBJ.DisplayError(sc),!
     </script>
     <br><a href="TeachersInformation.csp">Повернутись до списку
викладачів</a>
</body>
</html>
```

Завдання

Для власної бази даних згідно з варіантом створити CSP-застосунок, за допомогою якого можна редагувати дані. Клас обрати на власний розсуд.

Контрольні питання

1. Що представляє собою технологія CSP?
2. Який механізм роботи технології CSP?
3. Як вбудувати вирази Cache у код HTML?
4. Як зв'язати об'єкт з бази даних з WEB-формою?
5. Як зчитати дані з WEB-форми за допомогою об'єкта *%request*?
6. Яке призначення об'єкта *%session*?

ЛАБОРАТОРНА РОБОТА № 10

«СТВОРЕННЯ WEB-ЗАСТОСУНКУ ДО СКБД САСНÉ ЗА ДОПОМОГОЮ ТЕХНОЛОГІЇ ZEN. НАПИСАННЯ КЛІЄНТСЬКИХ МЕТОДІВ»

Теоретичні відомості

Технологія InterSystems Zen надає простий спосіб швидкого створення складних насичених даними веб-застосунків з візуально привабливим високоінтерактивним інтерфейсом користувача. Zen не є мовою програмування четвертого покоління. Zen – це багата бібліотека попередньо побудованих об'єктних компонентів і засобів розробки на основі технології CSP і об'єктної технології.

Компоненти Zen дозволяють виконувати набагато більше динамічних взаємодій, не обмежуючи розробників лише можливістю передачі значень на сервер із використанням механізму «Submit». Наприклад, використовуючи компонент форми у Zen, можна задати власну процедуру валідації, включаючи негайні виклики сервера без необхідності направлення запиту сторінки і її подальшого перемальовування. Для користувачів такий процес введення даних виглядає більш природним.

У Zen застосовується CSP-механізм управління сеансом, що забезпечує аутентифікацію користувача, шифрування даних і збереження незмінних даних сеансу за запитами сторінок. Усі сполучення між браузером і сервером відбувається за допомогою передачі об'єктів між ними з використанням більш передової версії методу, часто званого AJAX (Asynchronous JavaScript і XML).

Сторінки на базі Zen можуть бути легко скомбіновані зі сторінками, розробленими за допомогою інших методів веб-розробки на основі CSP.

Створення нового застосунку Zen

Застосунок Zen складається з трьох частин:

- з класу застосунка (`%ZEN.application`), що визначає поведінку і стиль;
- з класів сторінок (`%ZEN.Component.page`);
- з класів компонентів (`%ZEN.Component.component`), розміщених на сторінках.

Zen-застосунок описує повну множину дій, можливих при взаємодії сервера і клієнта. Ці дії можуть містити відображення web-сторінки, виконання запитів до бази даних, запис даних на диск і багато іншого. Коли створюється Zen-застосунок, створюється набір класів Zen, що реалізують ці дії.

Базовий клас Zen-застосунку (`%ZEN.application`) і базовий клас Zen-сторінки (`%ZEN.Component.page`) наслідуються від класу `%CSP.page`. Це каже про те, що і Zen-застосунок і Zen-сторінка є сторінками CSP. Таким чином, усі властивості, методи, параметри і змінні (наприклад, `%session`) доступні і Zen-сторінці.

Для того, що створити новий застосунок, необхідно у Caché Studio скористатись меню **File -> New** та обрати вкладку **Zen**, після чого вказати **Zen Application**.

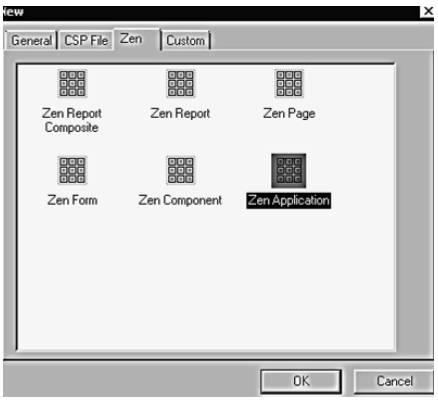


Рис. 10.1. Zen Application

Після цього відкриється майстер створення застосунку Zen, у якому необхідно сказати назву пакета, де буде міститись клас застосунку; власну назву класу застосунку, назву застосунку (звичайно збігається з назвою класу) та опис (необов'язково).

Рис. 10.2. Майстер створення застосунку Zen

Після завершення створення нового застосунку Zen буде створений клас, що наслідує клас `%Zen.application`, що є базовим для всіх класів Zen-застосунків. Згенерований код створеного класу нового Zen-застосунку наведений нижче.

```
/// Zen Application for university database
Class ZEN.UniversityZenApp Extends %ZEN.application
{

/// This is the name of this application.
Parameter APPLICATIONNAME = "UniversityZenApp";

/// This is the URL of the main starting page of this application.
Parameter HOMEPAGE ="";

/// This Style block contains application-wide CSS style definitions.
XData Style
{
<style type="text/css">
</style>
}
}
```

Як видно з програмного коду, у класі є блок, який описує стиль (`XData Style`). У цьому блоці можна задати стиль CSS для певних елементів сторінки. При цьому заданий стиль буде застосовано для всіх сторінок, які будуть входити в застосунок.

Наприклад, опишемо між тегами `<style>` та `</style>` наступний стиль CSS:

```
body {
     background: #F0F0FF;
     font-family: verdana;
}
```

Додавання нової сторінки Zen до застосунку

Для того, щоб створити нову сторінку Zen, потрібно скористатись у Caché Studio тим самим пунктом меню, що й при створенні застосунку Zen, варто тільки обрати **Zen Page** замість **Zen Application**.

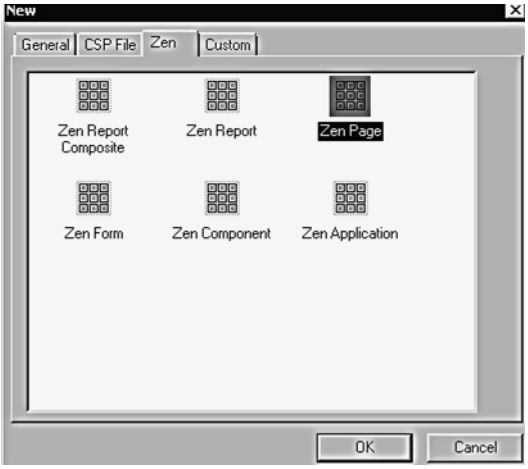


Рис. 10.3. Додавання нової сторінки Zen до застосунку

Після цього буде відкритий майстер створення нової сторінки Zen, на першому етапі якого необхідно задати назву класу сторінки; назву пакета, до якого належатиме клас; а також Zen-застосунок, до якого буде входити сторінка, заголовок таблиці та домен. При цьому обов'язковими є тільки назва класу та назва пакета.

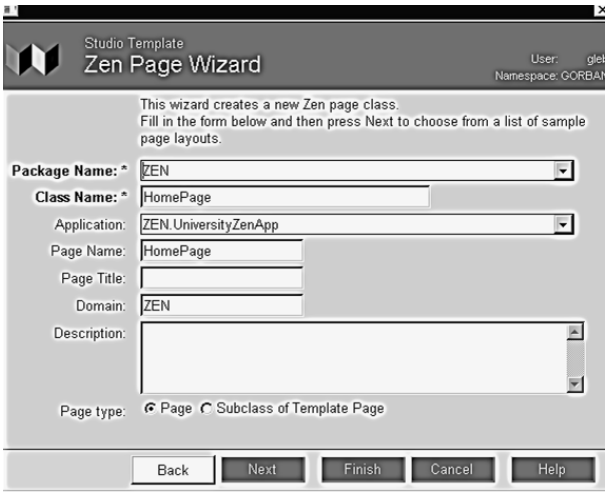


Рис. 10.4. Майстер створення нової сторінки Zen

Якщо натиснути Next, можна перейти до другого етапу майстра, який дозволяє обрати зовнішній вигляд майбутньої сторінки. Обравши на цьому етапі Default та натиснувши Finish, ми отримаємо клас сторінки Zen, який наслідує клас *%ZEN.Component.page*, який є базовим класом для всіх класів, що представляють собою сторінки Zen.

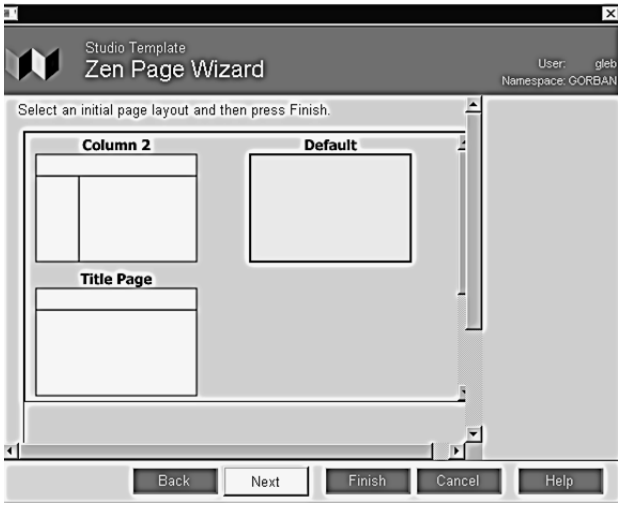


Рис. 10.5. Клас сторінки Zen

У результаті буде створено клас нової Zen-сторінки, знегерований код якого наведено нижче.

```
/// Created using the page template: Default
Class ZEN.HomePage Extends %ZEN.Component.page
{

/// Class name of application this page belongs to.
Parameter APPLICATION = "ZEN.UniversityZenApp";

/// Displayed name of this page.
Parameter PAGENAME = "HomePage";

/// Domain used for localization.
Parameter DOMAIN = "ZEN";

/// This Style block contains page-specific CSS style definitions.
XData Style
{
<style type="text/css">
</style>
}

/// This XML block defines the contents of this page.
XData Contents [ XMLNamespace = "http://www.intersystems.com/zen" ]
{
<page xmlns="http://www.intersystems.com/zen" title="DepartmentPage">
</page>
}

}
```

Складовими Zen-сторінки є блок `XData Style`, що визначає стиль, у якому можна визначити власний стиль CSS для компонентів сторінки, та блок змісту `XData Contents`, що визначає компоненти, що розміщуються на сторінці. У цьому блоці визначаються компоненти, що будуть розміщені на сторінці шляхом його описання на мові XML. Кожен елемент XML відповідає класу компонента з тим же ім'ям. Блок `XData Contents` містить один елемент `<page>`, який виступає в якості контейнеру верхнього рівня для всіх XML-елементів у цьому блоці.

Новостворена сторінка завжди є порожньою і не містить компонентів. Вона може бути наповнена компонентами вручну. Zen конструює сторінку для відображення, додаючи компоненти індивідуально один за іншим, згідно з їх описом у класі сторінки. Процес формування сторінки прихований від користувача, він бачить тільки результат.

Для того, щоб відкрити сторінку Zen, її спочатку її треба скомпілювати, після чого за допомогою меню у Caché Studio **View -> Web Page** відкрити вказану сторінку у браузері.

У класі Zen-застосунку, до якого відноситься Zen-сторінка є параметр `HOMEPAGE`, значенням якого можна встановити назву класу відповідної сторінки. Наприклад:

```
Parameter HOMEPAGE = "ZEN.HomePage.cls";
```

При цьому, скомпілювавши клас Zen-застосунку та скориставшись пунктом меню **View -> Web Page** можна відкрити вказану у якості домашньої до застосунку Zen-сторінку.

У якості прикладу для цієї лабораторної роботи показаний приклад створення Zen-сторінки, яка виконує наступні дії:

1. Виводить таблицю з даними про кафедри університету (об'єкти класу `University.Department`). Остання колонка таблиці є посиланням, яке буде розглянуто далі.

Кафедри

	#	Код кафедри	Назва кафедри	Аудиторія	
	1	5	Інтелектуальних інформаційних систем	2-401	вивести викладачів
	2	17	Інформаційних технологій і програмних систем	2-305	вивести викладачів
	3	3	Прикладної та вищої математики	2-404	вивести викладачів

Додати нову кафедру

Рис. 10.6. Таблиця з даними про кафедри університету

2. При виборі певного рядка в таблиці автоматично відкривається форма редагування, у яку заносяться поточні дані вибраного об'єкта. Ця форма місить кнопки «Save» та «Delete», які призначені для збереження даних та видалення відповідного об'єкта. Крім того, сторінка містить посилання «Додати нову кафедру», натиснення на яке призведе до того, що буде відкрита та сама форма редагування, але з порожніми полями для введення.

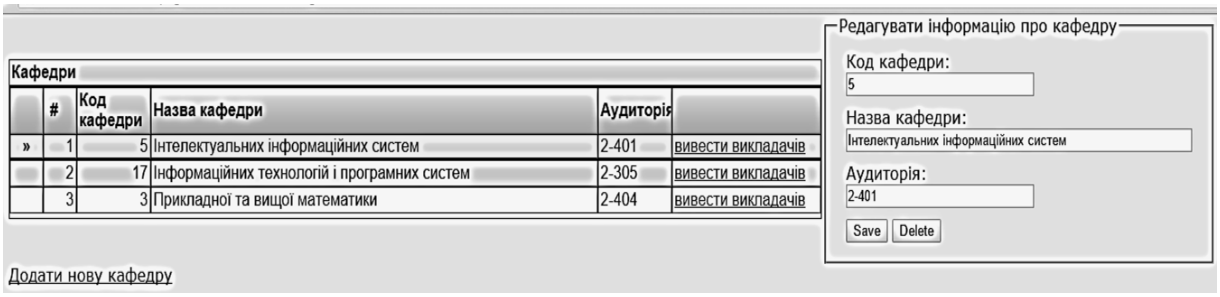


Рис. 10.7. Форма редагування

3. При натисненні на посилання «вивести викладачів» для певної кафедри на сторінку буде виведена інша таблиця, що містить дані про викладачів, які працюють на відповідній кафедрі (клас `University.Teacher`). Нагадаємо, що класи кафедри та викладача поєднані між собою зв'язком «один до багатьох».

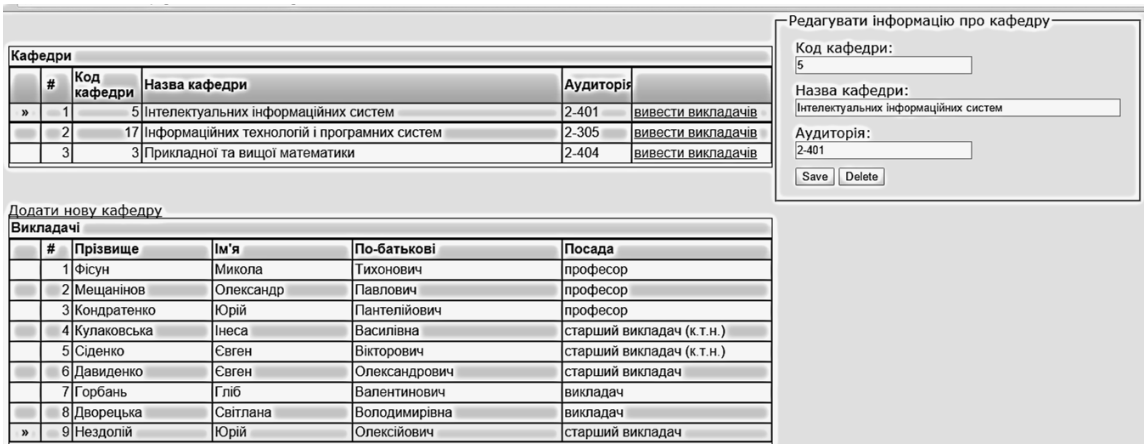


Рис. 10.8. Клас `University.Teacher`

Наведені вище скріншоти представляють собою зовнішній вигляд Zen-сторінки після додавання до неї відповідних компонентів та написання відповідних методів реалізації, що будуть розглянуті далі.

Наповнення сторінки компонентами

Усі компоненти Zen наслідуються від класу `%ZEN.Component.component`. Проте, коли Zen-компонент розміщується на сторінці, програмування ведеться, насамперед, на XML, а не на ObjectScript.

При підготовці класу Zen для програми, компоненти розміщуються на сторінці, використовуючи блок `XData Contents`, в якому вони описуються на мові XML. Кожен елемент XML відповідає класу компонента з тим же ім'ям.

Коли компілюється клас, що містить блок `XData Contents`, Zen генерує код для відображення сторінки в браузері. Під час виконання цей код представляє зазначені класи компонентів як нащадки об'єкта сторінки. Zen керує цією ієрархією автоматично.

Програміст не працює з класами компонентів безпосередньо, він працює з XML-проекціями цих класів. Для кожного компонента існує проекція, яка складається з:

- XML-елемента з тією ж назвою, як і у класу (`<page>`, `<html>`, `<hgroup>`, і т. ін.);
- XML-атрибутів, відповідних властивостей класу (ширина, висота, і т. ін.).

Групи компонентів

Компонент групи наслідується класу від `%ZEN.Component.group`. Головним призначенням компонента групи є виконання функції контейнера для інших компонентів. Прикладами груп є компоненти `page`, `pane`, `menu`, `form` і `composite`, усі вони наслідуються від класу `%ZEN.Component.group`.

Кожна група призначена для розміщення компонентів, які вона містить на сторінці. Група може мати горизонтальне або вертикальне розташування. Кожна група за замовчуванням має вертикальне розташування за винятком `<hmenu>` і `<hgroup>`.

Потім додамо до сторінки елемент `<hgroup>` сторінки і три елементи `<vgroup>` елементів у `<hgroup>`. Компонент hgroup використовує 100 % від доступної ширини сторінки. Компоненти vgroup використовувати відповідно 5 %, 90 %, і 5 % ширини `hgroup`.

```
<page xmlns="http://www.intersystems.com/zen" title="DepartmentPage">
<hgroup width="100%">
   <vgroup width="5%"></vgroup>
   <vgroup width="90%"></vgroup>
   <vgroup width="5%"></vgroup>
</hgroup>
</page>
```

Компонент hgroup розміщує свої дочірні компоненти горизонтально на сторінці, тоді як компонент vgroup розміщує свої дочірні компоненти вертикально. При додаванні 3 елементів `vgroup` до `hgroup` вони будуть розташовані по горизонталі.

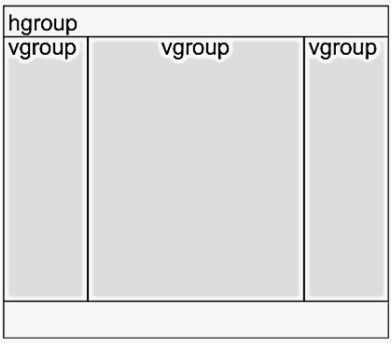


Рис. 10.9. Компонент hgroup

Ми будемо додавати компоненти на сторінці до центрального великого елементу vgroup. Ліва та права вертикальні групи будуть грати роль границь.

Додавання таблиці до сторінки

Zen-таблиці дозволяють переглядати дані в зручному для користувача вигляді. Таблиця бере результат повернень SQL-запитом і відображує його у вигляді таблиці HTML. Існує кілька можливостей генерації результуючої множини для Zen-таблиці. Так, можна визначити SQL-скрипт в самій таблиці, вказати посилання на зумовлений запит або визначити необхідні вхідні параметри для генерації SQL-скрипта «на льоту». Таблиці у Zen відповідає клас компонента `tablePane`.

Для того, щоб розмістити компонент Zen на сторінці, також можна скористатись майстром. Для цього слід помістити курсор між відкриваючим та закриваючим тегами `page`, після чого скористатись меню **Tools -> Templates** та обрати **Zen Element Wizard**

Першим кроком роботи майстра є вибір нового компонента зі списку. Вказуємо необхідний компонент `tablePane`.

Рис. 10.10. Zen Element Wizard

На наступному кроці можна побачити стислу інформацію про відповідний компонент, що буде доданий на сторінку, після чого, натиснувши ще раз **Next**, можна задати значення його XML-атрибутам, що відповідають властивостям у відповідному класі.

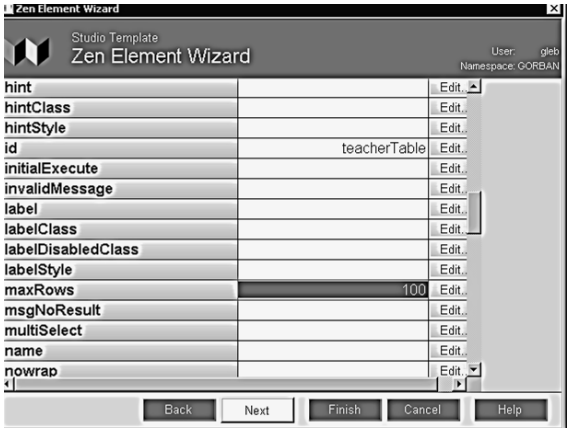


Рис. 10.11. XML-атрибути

Значення атрибутів компонента можна задати і вручну. Наприклад:

```
<tablePane
       width="900px"
       id="departTable"
       tableName="University.Department"
       maxRows="1000"
       pageSize="10"
       showRowNumbers=«true"
       showZebra="true"
       caption="Кафедри"
      valueColumn=«ID">
</tablePane>
```

Таблиця автоматично генерує SQL-запит, який використовується для вилучення даних Caché. Значення атрибута `tableName` (у цьому випадку `University. Department`), визначає речення FROM у запиті.

Зазначимо, що клас, дані якого будуть виведені у таблицю та прив'язані до форми редагування) повинен, крім класу *%Persistent* також наслідувати клас *%ZEN.DataModel. Adaptor*.

```
Class University.Department Extends (%Persistent, %ZEN.DataModel.Adaptor).
```

Тепер додамо до таблиці стовпчики за допомогою компонента `column` (відповідний клас *%Zen.Auxiliary.column*. Відповідні компоненти необхідно описати між відкриваючим та закриваючим тегами таблиці.

```
<column colName="ID" hidden="true"/>
    <column header="Код кафедри" width=«10%" colName="CodeDepart"/>
    <column header="Назва кафедри" width="60%" colName="NameDepart"/>
    <column header="Аудиторія" width="10%" colName="AudNumber"/>
```

Елементи `<column>` через їх атрибути `ColName`, формують речення SELECT у відповідному запиті. Однак існуть і інші способи визначення запиту, який заповнює таблицю. Один із способів буде розглянутий нижче у даній ЛР.

Звернемо увагу на те, що сповпець з назвою ID є прихованим, а також саме цей стовпець вказує значення таблиці шляхом присвоєння відповідного значення "ID" атрибута `valueColumn` у таблиці.

Створення форми на сторінці

Тепер ми реалізуємо форму введення даних, використовуючи Zen Model-View-Controller framework. Архітектура MVC відокремлює даних програми (модель) зі своїх користувальницьких інтерфейсів (вид), вставляючи шар коду (контролера) між ними.

Далі, ми додамо компонент `dataController`. Даний компонент є невидимим, тому його можна описати у будь-якому місці, але в межах відкриваючого та закриваючого тегу `page`.

```
<dataController id="departData"
   modelClass="University.Department"
   modelId=""
/>
```

Атрибут контролера `modelClass` використовується для зв'язування контролера з моделлю. Також ми встановлюємо значення атрибута `modelId` у порожній рядок (`" "`). Наше застосунок динамічно встановить значення цього атрибута, яке буде представляти ID певного об'єкта.

Наприкінці ми додаємо представлення, у нашому випадку – компонент форми. Така форма дозволяє користувачу редагувати дані вибраного об'єкту класу Department. Крім того, ми розмістимо форму всередині компонента `fieldSet`, так що ми можемо створити границю і легенду, а також приховувати і показувати форму. На початку форма є прихованою, тому атрибут `hidden` елементу `fieldSet` має значення `true`.

```
<fieldSet id="departFormGroup" hidden="true" legend="Редагувати інформацію
про кафедру">
<form id="departForm"
     controllerId="departData"
     layout="vertical"
     cellStyle="padding: 2px; padding-left: 5px; padding-right: 5px; ">

</form>
</fieldSet>
```

Значення атрибута `controllerId` пов'язує форму з контролером. Тепер модель, представлення та контролер пов'язані між собою.

Далі ми додамо до форми візуальні елементи.

```
<text label=«Код кафедри: " required="true"
dataBinding="CodeDepart" id="CodeDepart" name="CodeDepart" size="25"/>
<spacer height=".5"/>
<text label="Назва кафедри:
" dataBinding="NameDepart" id="NameDepart" name="NameDepart"
          size="50" />
<spacer height=".5"/>
<text label="Аудиторія:
" dataBinding="AudNumber" id="AudNumber" name="AudNumber" size="25"/>
```

Тут потрібно звернути увагу на атрибут dataBinding для кожного з полів введення. Значення цього атрибута прив'язує візуальний елемент до певної властивості класу моделі.

Розміщення таблиці та форми на сторінці

Тепер розмістимо форму на сторінці. Припустимо, ми хочемо, щоб форма з'явилась праворуч від таблиці. Добитися цього можна шляхом розміщення компонентів форми і таблиці всередині компонента `hgroup` так, що макет буде виглядати таким чином:

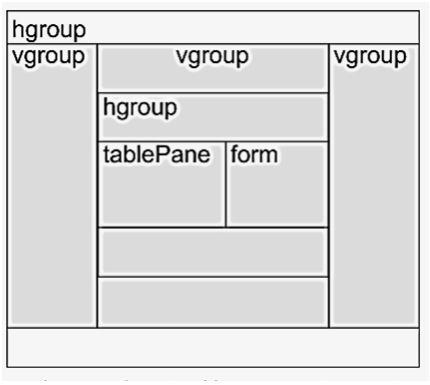


Рис. 10.12. Компонент `hgroup`

По-перше, помістимо елементи `<tablePane>` і `<form>` між тегами `<hgroup>` і `</hgroup>`.

По-друге, створимо деякий простір між таблицею і формою, додавши компонент `<spacer>` між ними з шириною 0.5.

```
<spacer width=".5"/>
```

Написання клієнтських методів Zen-сторінки

Тепер зробимо нашу сторінку більш інтерактивною. Зокрема визначимо для сторінки наступну поведінку: за замовчуванням форма редагування на сторінці є прихованою та стає видимою, коли користувач обере певний рядок у таблиці. При цьому до форми будуть передані значення властивостей обраного об'єкта.

Для реалізації описаної поведінки для нашої Zen-сторінки напишемо клієнтські методи мовою JavaScript.

Спочатку створимо метод `rowSelected`, який буде викликатись кожен раз при виборі рядка таблиці. Для цього для нашого компонента `tablePane` додамо атрибут `onselectrow`, в якості значення якого вкажемо виклик цього методу.

```
onselectrow="zenPage.rowSelected(zenThis);"
```

У цьому випадку змінна zenPage вказує на поточну сторінку, тобто нашу сторінку HomePage. У свою чергу змінна zenThis у Zen є аналогом посилання на поточний об'єкт this. У даному випадку дана змінна буде вказувати на нашу таблицю.

```
ClientMethod rowSelected(table) [ Language = javascript ]
{
    var id = table.getProperty('value');
    if (id) zenPage.showDepartForm(id);
}
```

Метод `rowSelected` має один аргумент `table`, через який знаходиться id обраного об'єкта, після викликається інший клієнтський метод сторінки `showDepartForm`, який пов'язує контролер з обраним об'єктом та робить форму видимою.

```
ClientMethod showDepartForm(id) [ Language = javascript ]
{
    var controller = zenPage.getComponentById('departData');
    controller.setProperty('modelId',id);
    var contactFormGroup=zenPage.getComponentById("departFormGroup");
    contactFormGroup.setProperty('hidden',false);
}
```

Даний метод виконує такі дії:

– встановлює значення атрибута `modelId` компонента `<dataController>` у значення id обраного об'єкта, що був переданий у метод в якості аргумента. Це у свою викликає контролер для завантаження даних для зазначеного об'єкта у форму;

– встановлює значення атрибута `hidden` компонента `<fieldSet>` у «false». Це робить форму редагування видимою на сторінці.

Додавання, редагування та видалення об'єктів на Zen-сторінці

Тепер реалізуємо операції додавання, редагування та видалення об'єктів. Для цього спочатку додамо до форми кнопки Save та Delete та задамо для них відповідні обробники подій. Для цього додамо наступний код до нижньої частини форми, безпосередньо перед закриваючим тегом `</form>`.

```
<hgroup>
  <button caption="Save" onclick="zenPage.saveDepart();" />
  <spacer width="5"/>
  <button caption="Delete" onclick="zenPage.deleteDepart();" />
</hgroup>
```

Після чого створимо для Zen-сторінки відповідні клієнтські методи.

```
ClientMethod saveDepart() [ Language = javascript ]
{
  var form = zenPage.getComponentById('departForm');
  form.save();
  var table=zenPage.getComponentById('departTable');
  table.executeQuery();
}
```

Указаний метод виконує 2 завдання:

– викликає метод `save()` компонента `form`, який у свою чергу викликає невидимо для нас метод `save()` компонента `dataController`;

– після оновлення даних у БД викликає метод `executeQuery()` компонента `tablePane`, щоб оновити дані у таблиці.

```
ClientMethod deleteDepart() [ Language = javascript ]
{
 var controller = zenPage.getComponentById('departData');
 controller.deleteId(controller.getModelId());
 var table = zenPage.getComponentById('departTable');
 table.executeQuery(true);
 controller.update();
}
```

Для того, щоб створити можливість додавання нового об'єкта до БД, додамо до нашої Zen-сторінки посилання, при натисненні на яке на сторінці стане видимою порожня форма, за допомогою якої користувач зможе створити та зберегти новий об'єкт.

```
<link align="left" caption="Додати нову
кафедру" href="javascript:zenPage.newDepart();"/>
```

А тепер реалізуємо метод створення нової кафедри.

```
ClientMethod newDepart() [ Language = javascript ]
{
  var controller = zenPage.getComponentById('departData');
  var contactFormGroup=zenPage.getComponentById("departFormGroup");
  contactFormGroup.setProperty('hidden',false);
  controller.createNewObject();
}
```

Виведення на сторінку об'єктів інших класів, пов'язаних з обраним об'єктом

Наприкінці виконаємо останнє завдання: виведемо на сторінку викладачів, які працюють на кафедрі, яка була вибрана. Для цього в таблиці, що містить об'єкти класу кафедри додамо нову колонку.

```
<column header="" width=«20%" linkCaption="вивести викладачів"
        link="javascript:zenPage.displayTeachers('#(%query.ID)#'); "/>
```

Ця колонка не буде містити певну властивість класу `University.Department`. Замість цього вона буде містити посилання, при натисненні на яке буде викликаний наш

майбутній метод, що виведе на сторінку дані про викладачів. Цей метод буде мати аргумент, що відповідає id кафедри, яка була вибрана користувачем. Змінна `%query` вказує на об'єкт класу `ResultSet`, що прив'язаний до таблиці кафедр. При створенні компонента `tablePane` та наповненні його колонками такий запит автоматично створюється «за лаштунками».

Після цього створимо нову таблицю, яка буде виводити дані про викладачів, розмістимо її на сторінці та вкажемо, що вона спочатку є прихованою. Вона стане видимою при натисненні на посилання «вивести викладачів» у таблиці з інформацією про кафедри.

```
<hgroup>
<tablePane
   width="900px"
   id="teacherTable"
   tableName="University.Teacher"
   maxRows="1000"
   pageSize="10"
   showRowNumbers="true"
   showZebra="true"
   caption="Викладачі"
   valueColumn="ID"
   hidden="true"
   sql="SELECT Surname, Name, Midname, Stat-
>NameStat From University.Teacher Where Depart=?">
   <column header="Прізвище" width="20%" colName="Surname"/>
   <column header="Ім'я" width="20%" colName="Name"/>
   <column header="По-батькові" width="30%" colName="Midname"/>
   <column header="Посада" width="30%" colName="NameStat"/>
   <parameter id="param"/>
</tablePane>
</hgroup>
```

Звернемо увагу на атрибут `sql`, значенням якого ми вказуємо текст запиту. Потім його поля будуть відображені у колонках таблиці. При цьому запит, за яким формуються дані у компоненті `tablePane` має умову: id об'єкта кафедри повинен збігається з деяким параметром, який описаний у таблиці за допомогою компонента `<parameter>`. Саме з цим параметром і пов'язаний знак питання в тексті запиту.

Тепер напишемо відповідний метод, який виводить дані про викладачів. Цей метод програмним шляхом задає необхідне значення параметру таблиці, яким є id вибраної користувачем кафедри.

```
ClientMethod displayTeachers(id) [ Language = javascript ]
{
  var table = zenPage.getComponentById('teacherTable');
  table.setProperty('hidden',false);
  var param = zenPage.getComponentById("param");
  param.value=id;
  table.executeQuery();
}
```

Тепер ми описали всі необхідні дії для того, щоб наша Zen-сторінка мала такий вигляд, який був указаний на початку лабораторної роботи.

Завдання

Створити застосунок ZEN, що містить ZEN-сторінку, яка виводить інформацію про дані певного класу власної бази даних, а також дозволяє створювати, редагувати та видаляти об'єкти цього класу. Відповідна сторінка має можливість перегляду пов'язаних з виділеним об'єктом об'єктів інших класів. Класи вибрати на власний розсуд.

Контрольні питання

1. З чого складається архітектура технології Intersystems Zen?
2. З чого складається застосунок Zen?
3. Якою є структура Zen-сторінки?
4. Як сформувати таблицю Zen?
5. Які типи методів реалізовані у класі Zen-сторінки?
6. Для чого призначений компонент dataController?
7. Як прив'язати дані до форми у Zen?

ВАРІАНТИ ЗАВДАНЬ ДЛЯ ЛАБОРАТОРНИХ РОБІТ

Для вирішення поставленої задачі необхідно скористатися однією з діаграм UML – діаграмою класів. Для цього необхідно визначити основні об'єкти предметної області, їх властивості, а також відношення між обраними об'єктами.

Предметна область повинна містити як мінімум один суперклас і як мінімум один клас, що містить колекцію. Варіанти завдань наведено в таблиці.

Студент може запропонувати свою предметну область, що *суттєво* відрізняється від наведених нижче. Можливість вибору своєї предметної області *узгоджується* з викладачем. При виборі своєї предметної області варіанти завдань для наступних лабораторних робіт, зокрема види запитів, форм і т. ін., також буде необхідно узгоджувати з викладачем.

№	Завдання
1	**Предметна область.** У базі даних інтернет-магазину зберігається інформація про клієнтів, товари, історію замовлень кожного клієнта, категорії товарів. **Завдання SQL.** Для обраного клієнта визначити найбільш популярну категорію й вивести список з товарів цієї категорії, які клієнт ще не замовляв.
2	**Предметна область.** У базі даних бібліотеки зберігається інформація про читачів, книги, журнали, відділи бібліотеки, історії видачі книг (журналів). **Завдання SQL.** Необхідно вивести список з 5 найбільш часто видаваних книг (журналів) за заданим відділом.
3	**Предметна область.** У базі даних зберігається інформаціях про домашню колекцію книг і дисків. Крім інформації про книги й диски, зберігається інформація про історію обміну дисками й книгами з іншими людьми. **Завдання SQL.** Необхідно вивести інформацію про всі віддані книги й диски, що перебувають зараз на руках.
4	**Предметна область.** У базі даних окружної виборчої комісії зберігається інформація про людей, райони, виборчі ділянки. **Завдання SQL.** Необхідно вивести список усіх виборців, що мають право голосу. Необхідно вивести кількість людей, що мають право голосу на введений із клавіатури момент часу, по кожному району й виборчій дільниці.
5	**Предметна область.** У базі даних ресторану зберігається інформація про персонал, страви, замовлення, клієнтів, столики. **Завдання SQL.** Необхідно вивести інформацію про 3 найбільш популярні страви за введений місяць.
6	**Предметна область.** У готельній базі даних зберігається інформація про постояльців, номери, персонал, поселення постояльців у номери. **Завдання SQL.** Необхідно вивести список усіх постійних клієнтів: клієнтів, що займали готельні номери більш 3 разів.
7	**Предметна область.** У базі даних книгарні зберігається інформація про книги, авторів, співробітників магазину й продажі книг. **Завдання SQL.** Вивести список з 10 найменш продаваних книг за введений період.
8	**Предметна область.** У базі даних кадрового агентства зберігається інформація про фактичних робітників, роботодавців, вакансії, резюме. **Завдання SQL.** Вивести список із трьох найбільш відповідних одна одній пар «вакансія/резюме».
9	**Предметна область.** У банківській базі даних зберігається інформація про клієнтів (фізичні і юридичні особи), кредити, графік погашення кредитів і про платежі погашення кредитів. **Завдання SQL.** Вивести список усіх клієнтів, що мають заборгованість з погашення кредиту.
10	**Предметна область.** У базі даних поштового агентства зберігається інформація про жителів міста, райони та вулиці, поштарів та про кореспонденцію, яка надходить. Кожний поштар закріплений за певним набором вулиць. **Завдання SQL.** Необхідно для введеного поштаря визначити список усієї кореспонденції, яку необхідно доставити в поточний день.
11	**Предметна область.** У базі даних страхового агентства зберігається інформація про клієнтів, види страхування, співробітників і укладені страхові договори. **Завдання SQL.** Визначити відсоток договорів, за якими проводилися страхові виплати за введений період.

Продовження таблиці

12	**Предметна область.** У базі даних шахового клубу зберігається інформація про шахістів, змагання (матчі й турніри) і зіграні партії. **Завдання SQL.** Для заданого гравця вивести список із 5 найбільш успішних для нього змагань (з найбільшим відсотком набраних очок).
13	**Предметна область.** У базі даних футбольної федерації зберігається інформація про футбольні команди, змагання, гравців і матчі. **Завдання SQL.** Вивести список п’яти найбільш результативних (тих, що забили найбільше голів) гравців у заданому змаганні.
14	**Предметна область.** У базі даних обліку сімейного бюджету зберігається інформація про доходи/витрати, статті витрат (постійні і випадкові) і джерела доходів (постійні і випадкові), а також про членів родини. **Завдання SQL.** Визначити найбільш й найменш видаткову статтю постійних витрат сімейного бюджету.
15	**Предметна область.** У базі даних підприємства зберігається інформація про замовників, постачальників, види продукції й поставки/відвантаження продукції. **Завдання SQL.** Необхідно вивести список усіх поставок/відвантажень продукції за введений період за заданим клієнтом.
16	**Предметна область.** У базі даних туристичного агентства зберігається інформація про заздалегідь певні туристичні маршрути (закордонні, місцеві і змішані), клієнтів, туристичні групи (що складаються із клієнтів) і поїздки груп. **Завдання SQL.** Вивести список найбільш популярних турів окремо за кожною категорією.
17	**Предметна область.** У базі даних залізничної станції зберігається інформація про пункти призначення поїздів, поїзди, час прибуття й від’їзду поїздів (планований й фактичний), платформи і шляхи. **Завдання SQL.** Вивести інформацію про всі спізнілі поїзди на задане число.
18	**Предметна область.** У складській базі даних зберігається інформація про поставки і відвантаження продукції, партнерів, інші відділи підприємства. **Завдання SQL.** Необхідно вивести список усіх поставок/відвантажень продукції за певний період.
19	**Предметна область.** У базі даних зберігається інформація про програмістів, команди, лідерів команди і проекти. **Завдання SQL.** Необхідно вивести команду програмістів та прізвище її лідера, яка виконала найбільшу кількість проектів за останній рік.
20	**Предметна область.** У базі даних хімічної лабораторії зберігається інформація про запаси реактивів (простих і складних речовини), співробітників лабораторії, відділи лабораторії, використання реактивів співробітниками. **Завдання SQL.** Необхідно для введеного відділу визначити список з 5 найбільш часто використовуваних реактивів.
21	**Предметна область.** У базі даних курсів іноземних мов зберігається інформація про мови, що вивчаються, студентів, групи, викладачів і проміжні іспити, які студенти складають щомісяця для контролю успішності. **Завдання SQL.** Визначити групу з найвищою середньою успішністю й вивести список її студентів.
22	**Предметна область.** У базі даних супермаркету зберігається інформація про товари (продовольчі і не продовольчі), торговельні залах супермаркету, продажах товарів і про касирів, що відпускають товар (касири, що закріплені за торговельними залами). **Завдання SQL.** Необхідно визначити торговельний зал, що продав товарів (обох типів) на найбільшу суму за певну дату.
23	**Предметна область.** У базі даних публікацій викладачів зберігається інформація про авторів (викладачів), видання, публікацій; наукометричні бази, до яких входять публікації. **Завдання SQL.** Необхідно визначити обсяг публікацій певного викладача в певній наукометричній базі.
24	**Предметна область.** У базі даних комп’ютерного обладнання університету зберігається інформація про пристрої, їх інвентарні номери, адміністраторів, відділи, приміщення, типи приміщення (комп’ютерний клас, лабораторія, кафедра і т. ін.). **Завдання SQL.** Необхідно визначити сумарний об’єм пам’яті жорстких дисків комп’ютерів, які закріплені за певним адміністратором.
25	**Предметна область.** У базі даних ЖЕКу зберігається інформація про будинки, типи будинків, квартири, мешканців, рахунки за оплату комунальних послуг. **Завдання SQL.** Необхідно визначити кількість трикімнатних квартир, у яких проживає більш ніж 3 особи і де є заборгованість за останній місяць.

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Харрингтон Д. Проектирование объектно-ориентированных баз данных / Д. Харрингтон ; пер. с англ. – М. : ДМК Пресс, 2001. – 272 с. : ил.
2. Джордан Д. Обработка объектных баз данных в C++. Программирование по стандарту ODMG / Д. Джордан ; пер. с англ. – М. : Издательский дом «Вильямс», 2001. – 384 с. : ил.
3. Дейт К. Дж. Введение в системы баз данных, 6-е издание / К. Дж. Дейт ; пер. с англ. – К. ; М. ; СПб. : Издательский дом «Вильямс», 2000.
4. Кирстен В. Постреляционная СУБД Cache 5. Объектно-ориентированная разработка приложений / В. Кирстен, М. Ирингер, М. Кюн, Б. Рериг. – 3-е изд., перераб. и дополн. – М. : ООО «Бином-Пресс», 2008. – 416 с. : с ил.
5. Постреляционная технология Cache для реализации объектных приложений / Н. Е. Кречетов, Е. А. Петухова, В. И. Скворцов та ін. – М. : Московский государственный инженерно-физический институт, 2001. – 152 с. : с ил.
6. Иванчева Н. А. Постреляционная СУБД Cache / Н. А. Иванчева, Т. А. Иванчева. – Новосибирск : Новосибирский государственный университет, 2004. – 120 с. : с ил.
7. Пасічник В. В. Організація баз даних та знань : підручник для вузів / В. В. Пасічник, В. В. Резніченко. – К. : Видавнича група «BHV», 2006. – 384 с. : іл.
8. Конноли Т. Базы данных : проектирование, реализация и сопровождение. Теория и практика, 2-е издание / Т. Конноли, К. Бегг, А. Страчан. – М., 2000. – 1112 с.
9. Гарсиа-Молина Г. Системы баз данных. Полный курс / Г. Гарсиа-Молина, Дж. Ульман, Дж. Уидом. – М. : Вильямс, 2003. – 1088 с.
10. Крёнке Д. Теория и практика построения баз данных. – 8-е изд. / Д. Крёнке. – СПб. : Питер, 2003. – 800 с.
11. Рамбо Дж. UML 2.0. Об'єктно-ориентированное моделирование и разработка / Дж. Рамбо, М. Блаха. – 2-е изд. – СПб. : Питер, 2007. – 544 с. : ил.
12. Фаулер М. UML. Основы / М. Фаулер, К. Скотт. – Пер. с англ. – СПб. : Символ-Плюс, 2002. – 192 с. : ил.
13. Якунин Ю. Учебное пособие «Технология ZEN/PROTOTYPE 6 / Ю. Якунин, Р. Квитунов. – Красноярск, 2008. – 148 с.

ДЛЯ НОТАТОК

ДЛЯ НОТАТОК

Навчальне видання

Гліб Валентинович
Горбань

ОБ'ЄКТНІ БАЗИ ДАНИХ

Методичні вказівки
до виконання лабораторних робіт

Випуск 266

Редактор *Я. Котенко.*
Технічний редактор, комп'ютерна верстка *Н. Хасянова.*
Друк, фальцювально-палітурні роботи *С. Волинець.*

Підп. до друку 08.04.2019
Формат 60x84$^{1}/_{16}$. Папір офсет.
Гарнітура «Times New Roman». Друк ризограф.
Ум. друк. арк. 12,56. Обл.-вид. арк. 3,67.
Тираж 5 пр. Зам. № 5705.

Видавець і виготовлювач: ЧНУ ім. Петра Могили.
54003, м. Миколаїв, вул. 68 Десантників, 10.
Тел.: 8 (0512) 50–03–32, 8 (0512) 76–55–81, e-mail: rector@chmnu.edu.ua.
Свідоцтво суб'єкта видавничої справи ДК № 6124 від 05.04.2018.