МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

«СИСТЕМИ З РОЗПОДІЛЕНИМИ БАЗАМИ ДАНИХ

ЛАБОРАТОРНИЙ ПРАКТИКУМ

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського як навчальний посібник для студентів, які навчаються за спеціальністю 122 «Комп'ютерні науки та інформаційні технології», спеціалізацією «Геометричне моделювання в інформаційних системах»

Київ

КПІ ім. Ігоря Сікорського
2018

Системи з розподіленими базами даних Лабораторний практикум : [Електронний ресурс] навч. посіб. для студентів спеціальності 122 «Комп'ютерні науки та інформаційні технології», спеціалізації «Геометричне моделювання в інформаційних системах» / КПІ ім. Ігоря Сікорського ; уклад.: уклад: В.О. Тихоход., В.І. Гайдаржи. – Електронні текстові дані (1 файл: 0,2 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2018. – 36 с.

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського (протокол № від р.)
за поданням Вченої ради факультету (протокол № від р.)

Електронне мережне навчальне видання

СИСТЕМИ З РОЗПОДІЛЕНИМИ БАЗАМИ ДАНИХ

ЛАБОРАТОРНИЙ ПРАКТИКУМ

Укладачі:
Тихоход Володимир Миколайович, канд. техн. наук ,доцент каф. АПЕПС ТЕФ.
Гайдаржи Володимир Іванович, ст. викладач каф. АПЕПС ТЕФ

Відповідальний редактор:
Коваль О.В., канд. техн. наук, в.о. зав. кафедри АПЕПС ТЕФ

Рецензент: Побіровський Юрій Миколайович – к.т.н., доц. каф. ТЕУТ та АЕС ТЕФ

Посібник розроблений на підставі робочої програми кредитного модуля «Системи з розподіленими базами даних - 2» та призначений для якісної організації виконання лабораторних робіт студентами обсягом 27 годин, підвищення розуміння основ систем з розподіленими базами даних .

Призначений для студентів, які навчаються за освітньою програмою підготовки магістрів за спеціальністю 122 «Комп'ютерні науки та інформаційні технології», спеціалізацією «Геометричне моделювання в інформаційних системах».

Спрямований на формування у студентів умінь та навичок систем з розподіленими базами даних. Забезпечує студентів необхідними теоретичними знаннями для опанування відповідної теми лабораторної роботи та виконання завдань, запланованих впродовж семестру.

1. ВСТУП

Дисципліна “СИСТЕМИ З РОЗПОДІЛЕНИМИ БАЗАМИ ДАНИХ” - складова у підготовці спеціалістів напряму "Комп'ютерні науки", тому що розробка прикладного програмного забезпечення для автоматизації адміністративної та виробничої діяльності уявляє собою систему яка забезпечує обробку інформаційної моделі систем управління або виробництва. Інформаційна модель в більшості випадків є базою даних досить складної структури , яка містить опис зміст багатьох інформаційних об'єктів які містяться в різних місцях зберігання, тобто створюють розподілену в просторі базу даних. Така розробка проводиться з залученням систем автоматизованого проектування програмного забезпечення які надають можливість виконувати необхідні операції над розподіленими базами даних яки є важливою частиною програмного забезпечення ПЕОМ. Програми розроблені з використанням спеціальних засобів проектування розподілених систем та баз значно підвищують якість розробки та зменшують час необхідній для неї.

Зазначена дисципліна включена до циклу "Професійно орієнтованих дисциплін за переліком програми". У структурно-логічній схемі навчання зазначена дисципліна розміщена у 10 семестрі , тобто тоді коли студенти прослухали курси “Конструювання програм та мови програмування”, “Структура та організація даних в ПЕОМ”," Організація баз даних та знань" та "САПР програмного забезпечення" , “ Проектування складних інформаційних об’єктів та систем” і тому набули певного досвіду з програмування мовами високого рівня та методами використання САПР.

II. МЕТА І ЗАВДАННЯ ДИСЦИПЛІНИ

Дисципліна “СИСТЕМИ З РОЗПОДІЛЕНИМИ БАЗАМИ ДАНИХ” викладається на першому курсі спеціальності. Викладання кредитного модулю має за мету формування у студентів наступних компетенцій (ОКХ спеціаліста напряму підготовки 6.050101 "Комп'ютерні науки") :

Базові знання в області проектування інформаційних систем в основі функціонування яких лежать розподілені бази даних.

Знання сучасних методів та технологій розробки програмного забезпечення систем з розподіленими базами даних з використанням SQL серверів баз даних засобами середовища візуального проектування та C#.

В результаті вивчення студент повинен бути здатним виконувати наступні типові завдання діяльності: виконувати проектування інформаційних систем з розподіленими базами даних то розроблювати програмне забезпечення підтримки систем з розподіленими базами даних із застосуванням сучасних технологій проектування з використанням SQL серверів баз даних засобами середовища візуального проектування та C#.

Матеріал дисципліни ґрунтується на знаннях отриманих студентами при вивченні основ об'єктно - орієнтованого програмування , а також на вивчення матеріалу курсу “ Організація баз даних та знань” і “ Проектування складних інформаційних об'єктів та систем” Під час викладання дисципліни розглядаються механізми використання можливостей проектування інформаційних систем з використанням засобів серверів баз даних яки підтримують платформу розробки клієнт-серверних систем .

Детально розглядаються питання моделювання та проектування

розподілених інформаційних системи за допомогою графічної уніфікованої мови моделювання UML .

Розглядаються властивості SQL - серверу Microsoft SQL SERVER щодо підтримки роботи з розподіленими базами даних , особливо питання розробки дволанкової та три ланкової архітектури побудови та використання систем з розподіленими базами даних а також відповідні механізми управління транзакціями.

В результаті вивчення дисципліни студент повинен ЗНАТИ:

- теоретичні основи моделювання інформаційних систем , яке працюють з розподіленими базами даних в середовищі WINDOWS - орієнтованих операційних систем
- теоретичні основи можливостей систем автоматизованого проектування програмного забезпечення DELPHI та C# відносно обробки інформації яка зберігається в розподілених базах даних.
- принципи та технологічні процедури звернення до баз даних яки є керованими різноманітними системами керування базами даних як в дволанковій так і в триланкової архітектурі
- технологічні принципи покладені в основу функціонування інформаційних систем на основі використання можливостей які надають технології , ADONET та CORBA для розробки систем з розподіленими базами даних

В результаті вивчення дисципліни студент повинен УМІТИ:

- Будувати функціональні UML-моделі інформаційних систем , які працюють з розподіленими в мережі базами даних використовуючи спеціалізовані інструментальні засоби моделювання.
- Створювати засобами візуальних середовищ програмування яки підтримують мови DELPHI та C# інформаційні системи з обробки інформації яка зберігається в розподілених базах даних.
- Використовувати технологічні процедури звернення до баз даних яки є керованими різноманітними системами керування базами даних як в дволанкової так і в триланкової архітектури
- Застосовувати можливості технологій DATASNAT, ADONET та CORBA для розробки систем з розподіленими базами даних

ІІІ. ПЕРЕЛІК ЛАБОРАТОРНИХ РОБІТ

№ лаб.	Тема лабораторної роботи	Кіл год
1	Побудова структурної схеми розподіленого підприємства	4
2	Побудова функціональної схеми типової та спеціалізованої філій розподіленого підприємства у вигляді UML –модели	4
3	Побудова функціональної схеми центрального офісу розподіленого підприємства у вигляді UML –модели	2
4	Побудова концептуальної схеми розподіленої БД	4
5	Розробка системи віддаленого доступу з застосуванням дволанкової архітектури	4
6	Розробка триланкової системи архітектури COM/DCOM в NET	2
7	Розробка триланкової системи архітектури REMOTING в NET	2

8	Розробка триланкової системи архітектури WCF в NET	3
9	Розробка системи з використанням триланкової архітектури за змішаною технологією CORBA	2
		27

Лабораторна робота N 1

Розробка структурної схеми розподіленого підприємства

Мета : Вивчити процес створення структурної схеми розподіленого підприємства

Побудова UML –моделі розподіленої програмної системи:

Теоретичні відомості

Розробка структурної схеми розподіленого підприємства

Розробка структурної схеми починається з вивчення документації з описом структури розподіленого підприємства

Під час вивчення необхідно:

- відокремити описи типової філії, яка займається безпосередньою виробничою діяльність, яка відбувається однаковим чином у типових філіях, наприклад автомагазини у обласних центрах
- відокремити описи спеціалізованої філії, яка займається безпосередньою виробничою діяльність, яка відбувається в інтересах типових філій але є унікальної в межах підприємства, наприклад загальних склад авто для підприємства.
- відокремити описи центрального офісу підприємств, до функцій якого відносяться діяльність з контролю, аналізу роботи філій та організаційна з забезпечення ефективної роботи підприємства в цілому

В результати вивчення необхідно скласти схему у вигляді таблиці в який наведені назви структурних підрозділів, назви посад(ролей посадових осіб), та для кожного підрозділу наведено перелік основних вхідних та вихідних документів . Під вхідними розуміються документи що формуються та реєструються у підрозділі та вносяться у базу даних . Під вихідними розуміються документи які формуються для підрозділу на підставі внесених в базу даних

Фрагмент схеми для типової філії виглядає наступним чином

Типова філія, магазин			
Відділи	Посади	Вхідні док. БД	Вихідні док. із БД
Адміністрація	- Директор - секретар	- Журнал реєстрації внутрішніх документів - Журнали реєстрації вхідної та вихідної кореспонденції.	Аналітичні параметризовані звіти по - прибуткам - збуту товару - пропозиціям Параметризовані звіти про: - вхідну кореспонденцію - вихідну

			кореспонденцію -
Відділ замовлень-закупівлі	-Зав відділом менеджер по продажам	- Замовлення на товар - Акт повернення товару - Прихідна накладна	Параметризовані звіти по: - закупівлю товару; - зіпсування товару ; наявність товару
Відділ продаж	- Зав.відділом - Продавець - Касир - р -	- Видаткова накладна - Чек - Рахунок фактура - Заявка на склад - Пропозиції щодо продажу	Параметризовані звіти про - продаж товару - наявність товару
Склад	- Завідувач складом - комірник	- накладна про отримання товару - накладна на видачу товару	Параметризовані звіти про - надходження товару - видачу товару - пошкоджений товар - заявки
Господарське обслуговування	- Завгосп	Заявки на інвентар	Параметризовані звіти про - заявки; - наявний інвентар
Бухгалтерія	- Головний бухгалтер - Бухгалтер	- Табель - Вхідні рахунки - Платіжні доручення - Податкові накладні - Касові ордери	Параметризовані звіт про - податки - розрахунки - стан фінансів - заробітну плати
Відділ кадрів	- менеджер по роботі з персоналом - начальник відділу кадрів	- Накази про рух контингенту	Параметризовані звіти про рух контингенту
ІТ-відділ	- начальник - системний адміністратор - програміст	- Заявки на обслуговування - Довідка про виконання	Параметризовані звіт про виконання заявок

Контрольні запитання:

Наведіть перелік складових розподіленого підприємства

Наведіть перелік та склад основних підрозділів типової філії

Наведіть перелік та склад основних підрозділів спеціалізованої філії

Наведіть перелік та склад основних підрозділів центрального офісу розподіленого підприємства

Завдання

1. Проведіть аналіз структури типової філії підприємства та побудуйте структурну схему типової філії підприємства
2. Проведіть аналіз структури спеціалізованої філії підприємства та побудуйте структурну схему спеціалізованої філії підприємства
3. Проведіть аналіз структури центрального офісу підприємства та побудуйте структурну схему центрального офісу

Лабораторна робота N2

Побудова функціональної схеми типової та спеціалізованої філій розподіленого підприємства у вигляді UML –моделі

Вивчити процес створення діаграми прецедентів та взаємодій за нотацією мови UML

Теоретичні відомості

Етапи створення моделі інформаційної системи. Архітектура інформаційної системи описується за допомогою 5 видів або представлень системи кожний з котрих є однією з можливих проекцій організації та структури системи і відповідає окремому аспекту її функціонування. Наведемо ці представлення:

Побудова моделі проводиться наступними кроками:

Необхідно визначити призначення системи. Призначення визначається однією фразою яка стисло визначає основну мету створення системи.

Визначити основні функції системи які вона повинна виконати для досягнення основної мети.

Визначити яке саме представлення системи необхідно представити в моделі Вирішити які діаграми необхідно створити та яки елементи на них відобразити.

Створити визначені діаграми.

Відповідно до отриманої моделі системи побудувати концептуальну схему бази даних

Приступити до реалізації системи.

Діаграми прецедентів. На діаграмі відображаються сукупність прецедентів (варіантів використання) , акторів (користувачів функцій системи) та відношення між ними.

Під час формування необхідно дотримуватися наступних правил:

На діаграмах відображаються тільки тих актори, які представляють майбутніх користувачів системи

Формування визначення прецеденту на останньому рівні декомпозиції має відповідати двом вимогам:

з одного боку користувач системи має чітко розуміти що саме надає йому даний прецедент(уявляти собі відповідний інтерфейс користувача)

з іншого розробник має розуміти, які дані є необхідними для реалізації прецеденту та уявляти собі алгоритм їх обробки

Фрагмент діаграми прецедентів акторів типової філії наведено на рисунку

ТИПОВА ФІЛІЯ

АДМІНІСТРАЦІЯ

ВІДДІЛ КАДРІВ

Контрольні запитання

Наведіть перелік основних діаграм для опису проекту програмної системи

Наведіть склад діаграм прецедентів та взаємодій

Наведіть характеристики елементів діаграми прецедентів

Наведіть характеристики елементів діаграми взаємодій

Завдання

1. Проведіть функціональну декомпозицію типової філії підприємства
2. Побудуйте діаграму прецедентів акторів типової філії підприємства
3. Проведіть функціональну декомпозицію спеціалізованої філії підприємства
4. Побудуйте діаграму прецедентів акторів спеціалізованої філії підприємства

Лабораторна робота N 3

Тема роботи: Побудова функціональної схеми центрального офісу розподіленого підприємства у вигляді UML – моделі.

Мета роботи : Вивчити процес створення діаграми прецедентів та взаємодій за нотацією мови UML для центрального офісу

Теоретичні відомості

Етапи створення моделі інформаційної системи. Побудова моделі проводиться наступними кроками:

Необхідно визначити призначення центрального офісу розподіленого підприємства системи. Призначення визначається однією фразою яка стисло

визначає основну мету створення системи.

Визначити основні функції центрального офісу розподіленого підприємства системи які вона повинна виконати для досягнення основної мети.

Особливу увагу приділити наступним функціям центрального офісу:

контролю за роботою філії

Порівняльному аналізу показників філій роботи

Аналізу роботи підприємства і цілому

Планування роботи філій

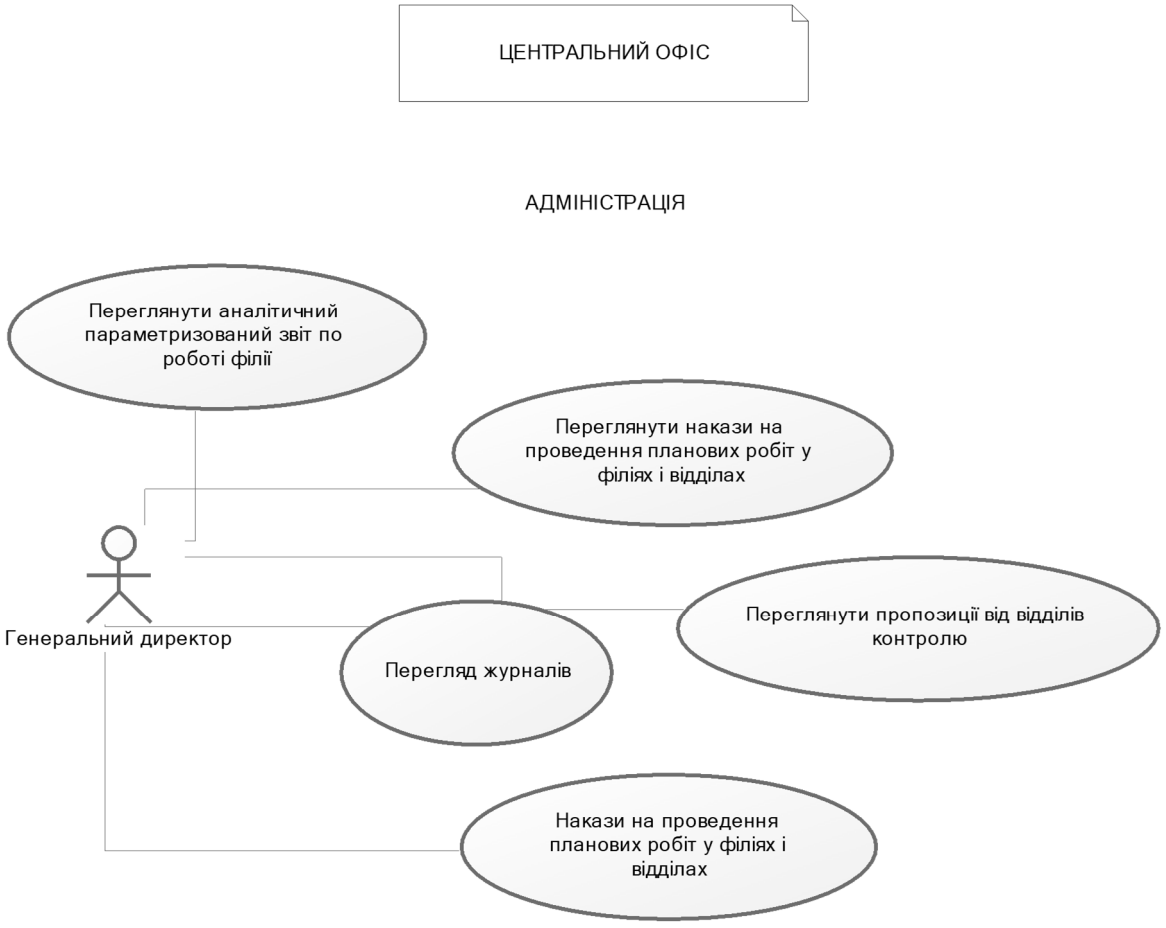
Прогнозування результатів роботи філій та підприємства в цілому

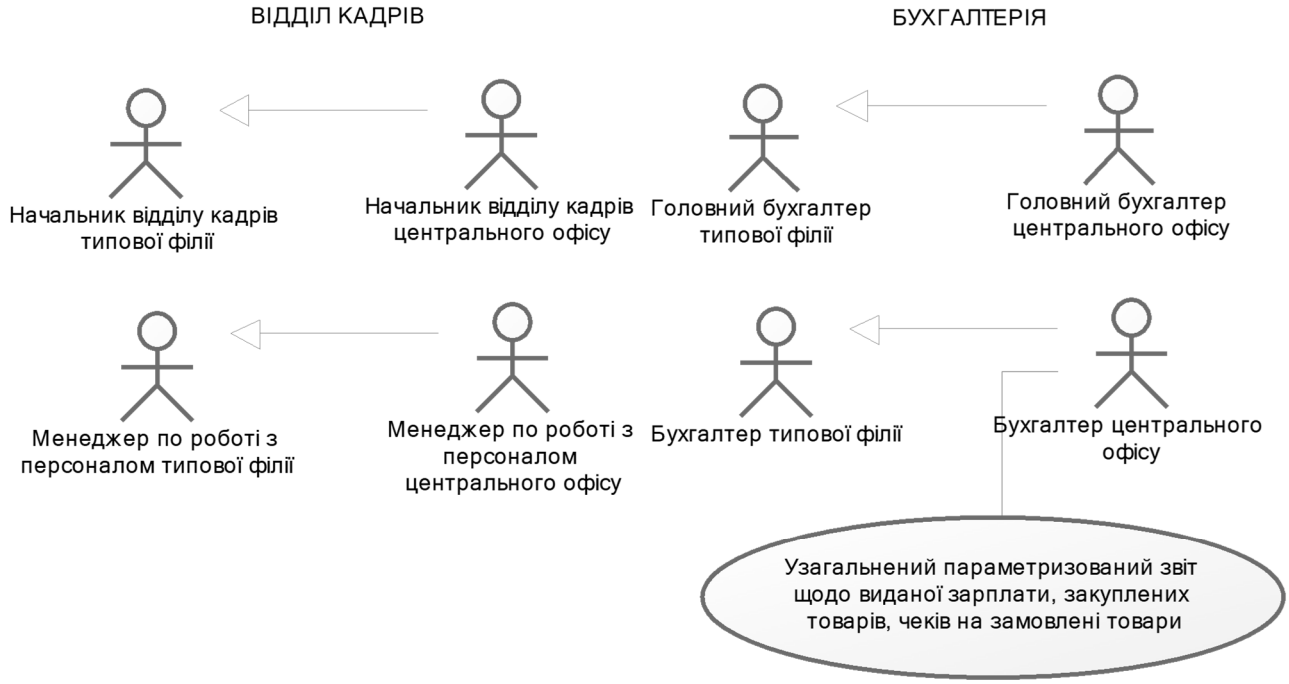
Особливо важливо відокремити функцій з прийняття рішень по основним напрямам діяльності філій та підприємства в цілому

Створити визначені діаграми.

Фрагмент функціональної схеми центрального офісу наведено на рисунку

Головний офіс





Контрольні запитання

Наведіть особливості функцій центрального офісу розподіленого підприємства

Наведіть склад діаграм прецедентів

Наведіть характеристики елементів діаграми прецедентів

Завдання

1. Проведіть функціональну декомпозицію центрального офісу підприємства

2. Побудуйте діаграму прецедентів акторів центрального офісу підприємства

Лабораторна робота N 4

Мета роботи: Вивчити процес побудови концептуальної схеми розподіленої бази даних

Теоретичні відомості

1. Основні визначення

Розподілена База Даних (РБД) чи Distributed DataBase (DDB), це сукупність розподілених даних представлених у рамках деякої реляційної моделі, яка відображує дані і зв'язки між ними.

Реляційна модель даних забезпечує однорідність представлення даних і зв'язків між ними у вигляді концептуальної схеми бази даних.

Реалізація реляційної моделі приводить до створення розподіленої бази даних.

РБД, яка створюється заново, є однорідною. Разом з цим, нерідко вона створюється як сукупність групи локальних баз даних, що уже функціонують у ряді систем. У цьому випадку виникає неоднорідна РБД. Обидва типи баз поміщаються в Систему Керування Розподіленою Базою Даних (СКРБД).

У загальному випадку локальні бази даних, що складають розподілену базу даних, не обов'язково повинні бути однорідними (тобто вестися однією СКБД) чи оброблятися в середовищі однієї і тієї ж операційної системи і на комп'ютерах того самого типу.

Наприклад, одна база даних може бути базою Oracle на комп'ютері SUN з операційною системою SUN OS(UNIX), друга база даних може вестися СКБД DB2 на мейнфреймі IBM 3090 з операційною системою MVS, а для ведення третьої бази може використовуватися СКБД SQL/DS також на мейнфреймі IBM, але з операційною системою VM. Обов'язкова тільки одна умова - ЛБД логічно зв'язані між собою, але фізично розміщені на декількох машинах (вузлах), що входять в одну комп'ютерну мережу.

При цьому мережна підтримка РБД характеризується наступними особливостями:

- кожен вузол має власні бази даних;
- вузли працюють погоджено, тобто користувач може одержати доступ до даних на будь-якому вузлі мережі так, ніби - то вони знаходяться на його власному вузлі;
- функціонування РБД відбувається незалежно від типів використовуваних у системах пристроїв;
- забезпечується просторова прозорість, що дає можливість користувачу не знати, де розташовані компоненти бази;
- підтримка повної функціональності, тобто можливість виконання всіх операцій, що можливі в базі даних, яка знаходиться в одній системі;
- гарантія цілісності даних, яку забезпечують функції спостереження за даними, виправлення помилок.

Найважливіші функціональні характеристики розподіленої бази даних такі:

- автономність вузлів розподіленої бази даних яка означає, що ведення кожної бази даних може відбуватися незалежно від інших;

- обробка розподілених запитів - таких запитів (SQL-команда), у ході виконання яких відбувається доступ до об'єктів (таблиць чи представлень) різних баз даних;
- виконання розподілених транзакцій. При виконанні розподілених транзакцій здійснюється погоджене керування усіма залученими базами даних. При цьому часто використовується технологія двофазної передачі інформації для виконання розподілених транзакцій.

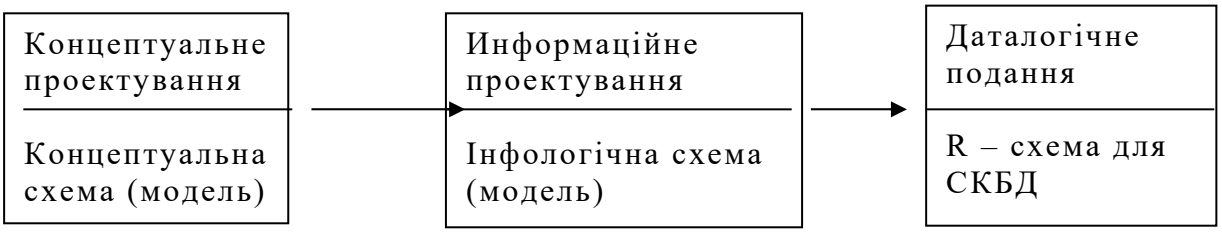
Суть РБД полягає в організації доступу користувачів до великих об’ємів інформації. Це дозволяє розмістити дані так, що останні, з одного боку, знаходяться в пунктах їх найбільшого опиту, а з іншого боку, за допомогою транзакцій забезпечується доступ до будь-яких даних, не залежно від того, де вони знаходяться.

Сучасні комп'ютерні технології при проектуванні розподілених баз даних розглядаються в різних варіантах архітектури "клієнт-сервер" (дворівнева, трирівнева), що орієнтується на об’єктно-орієнтований підхід. У цій архітектурі будь-який об'єкт, що використовує ресурси іншого об'єкта, визначається як клієнт, а об'єкт, що поставляє ресурси називається сервером відповідно до встановленого протоколу обміну.

2. Моделювання розподілених СКБД

Існують різні моделі для побудови схем розподіленої бази даних. Завдяки схемі користувач розглядає РБД як єдину БД.

Різні автори пропонують різні підходи до побудови набору схем, які розроблюються при проектуванні баз даних. Однією з найбільш розповсюджених схем є схема Арсеньєва-Яковлєва (Інтеграція розподілених баз даних, "ЛАНЬ", 2001 рік)



2.1. Концептуальне моделювання

Складається з наступних операцій:

- визначення джерел знань про предмет (закон евристики, існуючі рішення, документи);
- вибір системи класифікації об'єктів;
- виділення абстракції об'єктів і процесів, тобто виділення класів (типізація, абстрагування);
- побудова структури базових класів;
- побудова схеми взаємодії.

Мета концептуальної побудови - побудова фундаментальної семантики в представленій області.

Нотації опису різні, наприклад у мові UML концептуальна схема може бути представлена у вигляді діаграм класів без баз деталізації по атрибутах і методах.

2.2. Інфологічне моделювання

Виконується на основі концептуальної моделі як її уточнення і деталізація.

Основні операції:

- формування базових атрибутів класів, які виходять з типізації;
- формування додаткових атрибутів;
- визначення сукупного спільного використання атрибутів, тобто породження прототипів таблиць;
- визначення зв'язків логічної цілісності і зв'язків навігації;
- побудова інформаційно - логічної моделі.

Нотації різні.

У мові UML - повна діаграма статичних класів.

2.3. Даталогічне моделювання

Даталогічний аналіз і проектування це остаточне проектування реляційної схеми даних з врахуванням середовища обраної СКБД.

Проектування виконується на основі інфологічної схеми і полягає у виконанні таких пунктів:

- виділення таблиць (R-відношень);
- визначення фізичних форматів атрибутів на основі типів СКБД;
- визначення складу індексованих полів;
- визначення зв'язків навігації і логічної цілісності;
- установка бізнес-правил;
- проектування представлень;
- проведення нормалізації реляційних схем (усунення надлишковості, багатозначності).

Результат даталогічного проектування - реляційна схема бази даних.

Технологія побудови розподіленої бази даних заснована на наступних методологічних ознаках:

1. Виділення технологічних даних яки збираються та обробляються на локальних вузлах розподіленої системи (у філіях великої установи) та **формування локальних баз даних** (для типової та спеціалізованих філій) .

2. Виділення та формування інтегрованих даних системи. Принцип припускає створення **"головної бази даних"** (БД центрального офісу великої установи), що також є частиною розподіленої бази даних і містить інтегровані дані. Наприклад, головна БД містить усереднені показники забруднення

3. Виділення сукупності технологічних розподілених даних. Технологічні дані не несуть основної інформації, вони описують структуру основної інформації (склад і розподіл по мережі). Технологічні дані, будучи частиною сукупності розподілених даних, утворюють **базу метаданих системи.**

Контрольні запитання

Наведіть визначення розподіленої бази даних

Наведіть призначення головної БД

Наведіть призначення локальної БД філії

Наведіть призначення бази метаданих

Завдання

1. Проведіть аналіз структурного складу обраної для автоматизації установи.
2. Проаналізуйте функції співробітників центрального офісу, типових та спеціалізованих філій.
3. Проведіть аналіз необхідного змісту бази метаданих
4. Розробить концептуальні схеми складових розподіленої бази даних системи

Лабораторна робота N5

Розробка системи з використанням дволанкової архітектури системи клієнт - сервер

Мета роботи: Вивчити процес створення дволанкової архітектури системи

Контрольні запитання

Відмінні прикмети дволанкової системи

Які є варіанти реалізації дволанкових систем

Наведіть зміст системних файлів для реалізації дволанкової системи на базі сервер MS SQL SERVER

Теоретичні відомості

Для реалізації дволанкової системи клієнт – сервер з використання файл серверу достатньо під час створення аліасо вказати мережеву адресу віддаленої бази даних. Мережева адреса має наступний формат

Мережева адреса віддаленого комп'ютера \шлях до бази даних

Мережевою адресою віддаленого комп'ютера є або його доменне імя , або імя компютера в лакольної мережі , його TCP – адреса

Для забезпечення доступу до віддаленої бази даних яка підтримується серверам баз даних ms sql server необхідно виконати наступні дії:

1. Встановити на комп'ютері повнофункціональний екземляр сервера СКДБ MSSQLSERVER
2. Налаштувати встановлений сервер для використання протоколу TCP/IP для забезпечення підключення клієнтів по локальній мережі до сервера
3. Активізувати всі служби сервера

В програми користувача розробленої мовою C# можна використовувати три основних типи з'єднань з базою даних:

- ODBC- за допомогою класів ODBCConnection, ODBCCommand, OdbcParameter та ODBCTableEAdapter
- OLEDB - за допомогою класів OLEDBConnection, OLEDBCommand, OLEDBParameter та OLEDBTableEAdapter
- NATIV - за допомогою класів SqlConnection, sqlCommand, sqlParameter та sqlTableEAdapter

Контрольні запитання

Відмінні прикмети дволанкової системи

Які є варіанти реалізації дволанкових систем

Наведіть зміст системних файлів для реалізації дволанкової системи на базі сервер MS SQL SERVER

Завдання

1. Реалізуйте дволанкову систему з використанням сервера баз даних MS SQL SERVER. Для цього розробить два додаткі один з яких працює з локальною базою даних , а інший працює одночасно з тією самою базою як з віддаленою

Лабораторна робота N 6

Розробка триланкової системи архітектури COM/DCOM в NET

Мета роботи: Вивчити процес створення системи клієнт – сервер архітектури COM/DCOM в NET мовою C#

Теоретичні відомості

1. Особливості технології COM в. NET

Для забезпечення сумісності з програмами клієнт - сервер розробленими в середовищах програмування не. NET (наприклад на мовах DELPHI або C + + Builder) . NET середовища програмування підтримують C # дозволяють вирішувати наступні завдання

- Використовувати в C # - додатках COM-об'єкти створені в попередніх мовах, шляхом їх перетворення в збірки. NET
- Створювати в C # - додатках COM-об'єкти призначені для використання в попередніх мовами
- Використовувати в C # - додатках COM-об'єкти створені в C # - додатках але при цьому вони використовуються як звичайні збірки. NET

2. Створення COM сервера

Створення COM сервера в C # зводиться до створення спеціалізованої бібліотеки класів (модуля *. dll) в який включається клас забезпечений атрибутом [ClassInterface (ClassInterfaceType.AutoDual)] до складу якого включаються інтерфейси містять методи призначені для виклику користувачами. До класі можуть бути присутніми також власні методи (не включені до складу окремих інтерфейсів. Для підтримки технології до складу збірки слід включить бібліотеку using System.Runtime.InteropServices;

У розглянутому прикладі визначено два інтерфейсу:

- Інтерфейс IAdvancedMath у складі якого визначені методи int Multiply (int x, int y) і int Divide (int x, int y);
- Інтерфейс IAdvancedMath2 у складі якого визначено метод int Mul10 (int x)

У прикладі створюється com - сервер MyChrpServerLib COM - містить COM-об'єкт CCharpCalc в який включені ці інтерфейси і реалізовані їхні методи

Крім того в складі ласа визначені два власних методу public int Add (int x, int y) і public int Substruct (int x, int y)

Код збірки виглядає наступним чином

```
using System;
using System.Collections.Generic;
using System.Runtime.InteropServices;
using System.Linq;
using System.Text;

namespace MyChrpServerLib
```

```
{
   public interface IAdvancedMath
   {
      int Multiply(int x, int y);
      int Divide(int x, int y);
   }

   public interface IAdvancedMath2
   {
      int Mul10(int x);
   }

    [ClassInterface(ClassInterfaceType.AutoDual)]

   public class CCharpCalc : IAdvancedMath, IAdvancedMath2
   {
      public CCharpCalc() { }
      public int Add(int x, int y) { return x + y; }
      public int Substruct(int x, int y) { return x - y; }
      public int Multiply(int x, int y) { return x * y; }
      public int Divide(int x, int y)
      {
         if (y == 0) throw new DivideByZeroException();

         return x / y;
      }
      public int Mul10(int x)
      {
         return x * 10;
      }
   }
}
```

Для правильного формування COM - сервера і його автоматичної реєстрації необхідно встановити два керуючих прапора для проекту (Ланцюжок: Project -> імя_проектаProperties:

- на вкладці Application -> кнопка Assembly Information,у вікні Assembly Information включити прапорець Make Assembly COM-Visible
- на вкладці Build включити прапорець Register for COM interop

3. Ручна реєстрація сервера та створення бібліотеки типів

Раннє зв'язування клієнта на основі моделі компонентних об'єктів COM з компонентами. NET

Клієнти, створені на основі на основі моделі компонентних об'єктів Microsoft (COM) зв’язуються з сервером шляхом попереднього (раннього) зв’язування зазвичай використовують інформацію бібліотеки типів для доступу до компонентів на основі моделі компонентних об'єктів Microsoft (COM). Така інформація являє собою зручний спосіб створення примірників класів на основі моделі компонентних об'єктів Microsoft (COM), визначених в цих компонентах, створених на основі моделі компонентних об'єктів Microsoft (COM). Інформація а бібліотеках може зберігатися у файлах TLB, динамічно підключаються бібліотеках (DLL), спеціалізованих керуючих елементах OLE

(OCX) і виконуваних файлах, однак тільки файли TLB спеціально призначені саме для цієї мети.

Пакувальник Callable COM Wrapper, CCW) на основі моделі компонентних об'єктів COM формує з некерованого коду COM керованим кодами. NET

Бібліотека типів може бути згенеровано за допомогою утиліти Tlbexp.exe (Assembly to Type Library Converter - Конвертер збірки в бібліотеку типів) з метаданих в збірці. NET. Бібліотека типів дозволяє клієнтам на основі моделі компонентних об'єктів Microsoft (COM) переглядати компоненти. NET так, як ніби вони є звичайними компонентами, побудованими на основі моделі компонентних об'єктів Microsoft (COM). Традиційний клієнт на основі моделі компонентних об'єктів COM може використовувати інформацію в отриманій бібліотеці типів для доступу до компонентів. NET, застосовуючи раннє зв'язування.

Нижче наведено синтаксис для виклику утиліти командного рядка Tlbexp. exe

Tlbexp AssemblyName [параметри]

де AssemblyName - ім'я DLL (*. dll) бібліотеки містить COM - об'єкт

Параметри утиліти:

- / Out: FileName Файл вихідний бібліотеки типів (*. TLB

/ Nologo Пригнічує висновок протоколу

- / Silent (тихий) Пригнічує відображення повідомлень

- / Verbose (докладна) Додаткова інформація

- /? або / help (допомога) Вивести повідомлення - довідку з використання

Утиліта Tlbexp відкриває для моделі компонентних об'єктів COM тільки керовані загальнодоступні (public) типи. Клієнт на основі моделі компонентних об'єктів COM ніколи безпосередньо не посилається на COM-клас, а замість цього має справу тільки з інтерфейсом класу. Значення AutoDual (Автодуальний), яке задано в атрибуті Classlnterface (ClassInterfaceType:: AutoDual), автоматично генерує дуальний (двоїстий) інтерфейс для доступу до COM класу. Ранньо-зв'язані клієнти на основі моделі компонентних об'єктів Microsoft (COM) можуть використовувати даний файл *. tlb під час компіляції

Вміст файлу бібліотеки типів можна переглянути, використовуючи сервісну програму для перегляду об'єктів OLE / COM - полноекранную утиліту Oleview.exe, розташовану в папці утиліт SDK середовища NET

Для того, щоб середовище моделі компонентних об'єктів COM могла знайти потрібну фабрику класів, шлях до сервера, і т.п. COM-сервер повинен бути зареєстрований тобто повинна відбутися запис інформації про COM - сервер до реєстру.

Утиліта реєстрації збірки Regasm. exe (Assembly Registration Utility) прочитує метадані в збірці і додає необхідні записи до реєстру, що дозволяє клієнтам на основі моделі компонентних об'єктів COM використовувати компоненти збірки. NET так, як ніби вони є звичайними зареєстрованими компонентами, побудованими на основі моделі компонентних об'єктів Microsoft (COM). Звичайно, клієнти при цьому використовують заступник - викликається пакувальник на основі моделі компонентних об'єктів COM (Callable COM Wrapper, CCW).

Синтаксис виклику утиліти реєстрації збирання:.

Regasm AssemblyPath [параметри]

Де AssemblyPath - шлях до реєстрованої збірці містить COM-об'єкт (*. dll)

Параметри утиліти:

- / Unregister Скасувати реєстрацію типів
- / Tlb [: FileName] Вказаний файл бібліотеки типів
- / Regfile [: FileName] Зазначене ім'я вихідного файлу
- / Codebase (кодова сторінка) Встановлює кодову сторінку
- В системному реєстрі
- / Registered (зареєстрований) Звертатися тільки до попередньо
- Зареєстрованим бібліотекам типів
- / Nologo Запобігає висновок протоколу
- / Silent (тихий) Запобігає відображення повідомлень
- / Verbose (докладно) Виводить додаткову інформацію
- /? або / help (допомога) Вивести повідомлення - довідку
- По використанню

Реєстрація дозволяє будь-якому клієнту на основі моделі компонентних об'єктів COM отримувати доступ до компонентів. NET так, як ніби це звичайні компоненти, побудовані на основі моделі компонентних об'єктів Microsoft (COM).

Потім можна використовувати утиліту Regedt32. exe для перевірки того, що інформація була коректно занесена до реєстру

3. Створення клієнта

Як тільки закінчена реєстрація збірки *. dll як компонент на основі моделі компонентних об'єктів Microsoft (COM), можна починати розробку клієнтського додатку.

Для початку слід інсталювати збірку в клієнтський проект. Можна, звичайно, все виконати, просто скопіювавши збірку *. dll в папку клієнта Зробивши це, можна, нарешті, виконати клієнт, побудований на основі моделі компонентних об'єктів COM.

Клієнти не. Net технологій, а традиційних COM програм слід створити COM об'єкт традиційним способом, наприклад за допомогою імпорту бібліотеки типів для забезпечення раннього зв'язування, а потім викликом CreateOleObject

Клієнти. Net технологій використовують сервер як звичайну dll бібліотеку підключається до CC допомогою оператора using, в нашому випадку MyChrpServerLib;

Долинний робота може йти двома шляхами:

З явним використанням наявності інтерфейсів

У цьому випадку активізація сервера виробляється з використанням змінної типу object і активізацією інтерфейсів

```
 public  object myComObjOB;
   IAdvancedMath i1;
   IAdvancedMath2 i2;

// Create OLE As Object and activate interfaces
   myComObjOB = Activator.CreateInstance(typeof(CCharpCalc));
```

```
i1 = (IAdvancedMath) myComObjOB;
i2 = (IAdvancedMath2)myComObjOB;
this.label4.Text = "Creating OBJECT is success";
```

Після цього можна викликати методи відповідного інтерфейсу, наприклад:

```
/ / Call metods in firs interface IAdvancedMath i1;
r = i1.Multiply (x, y);
this.label1.Text = this.label1.Text + "=" + r.ToString ();
```

З використанням усіх методів без явного поділу по інтерфейсах
У цьому випадку активізація сервера виробляється з використанням змінної типу класа та активізацією інтерфейсів

```
public CCharpCalc myComObjCalc;
// Create OLE As  class CCharpCalc
myComObjCalc = new CCharpCalc();
this.label5.Text = "Creating CLASS is success";
```

После этого можно вызывать любые методы класса, например:

```
// Call all methods in class
//   Method in interface IAdvancedMath
r = myComObjCalc.Multiply(x, y);
this.CalcMulty.Text = this.CalcMulty.Text +  "="+ r.ToString();
//   Method in interface IAdvancedMath2
r = myComObjCalc.Mul10(2);
this.CalcMul10.Text = this.CalcMul10.Text + "="+ r.ToString();
//  Direct Method in class myComObjCalc
r = myComObjCalc.Add(x,y);
this.CalcAd.Text = this.CalcAd.Text + "="+ r.ToString();
```

Контрольні запитання

1. Наведіть особливості технології COM в . NET
2. Наведіть технологію створення сервера
3. Опишіть реєстрацію сервера та створення бібліотеки типів
4. Наведіть технологію створення клієнта

Завдання

1. Визначить бажану функціональність сервера додатків
2. Реалізуйте систему клієнт – сервер з використанням визначеного сервера додатків в технології .NET мовою C#

Лабораторна робота N 7

Розробка триланкової системи архітектури REMOTING в NET

Мета роботи: Вивчити процес створення системи клієнт сервер з використанням технології архітектури REMOTING в NET мовою C#/

Теоретичні відомості

В основі використання технології Remoting лежить клас RemotingConfiguration. який надає різні статичні методи для конфігурації

інфраструктури віддаленого взаємодії Простір імен: System.Runtime.RemotingЗбірка: mscorlib (в mscorlib.dll)Тип RemotingConfiguration надає наступні члени.

Методи

Ім’я	Опис
Configure	Перевантажений.
CustomErrorsEnabled	Вказує, чи повертають канали сервера в цьому домені додатку фільтровані або повні відомості про винятки в локальні чи віддалені викликають оператори.
GetRegisteredActivatedClientTypes	Витягує масив типів об'єктів, зареєстрованих з боку клієнта як віддалено активованих типів.
GetRegisteredActivatedServiceTypes	Витягує масив типу об'єктів, зареєстрованих з боку служби, який може бути активований на запит клієнта
GetRegisteredWellKnownClientTypes	Витягує масив типів об'єктів, зареєстрованих з боку клієнта як добре відомих типів.
GetRegisteredWellKnownServiceTypes	Витягує масив типів об'єктів, зареєстрованих з боку служби як добре відомих типів.
IsActivationAllowed	Повертає логічне значення, що показує, чи дозволено клієнтська активація зазначеного типу
IsRemotelyActivatedClientType	Перевантажений. Перевіряє, чи зареєстрований тип зазначеного об'єкта як віддалено активованого типу клієнта..
IsWellKnownClientType	Перевантажений. Перевіряє, чи зареєстрований тип зазначеного об'єкта як добре відомого типу клієнта.
RegisterWellKnownServiceType	Перевантажений. Реєструє об'єкт Type з боку сервера в якості добре відомого типу (єдиного виклику або одноелементні множини
RegisterWellKnownClientType	Перевантажений. Реєструє об'єкт Type з боку клієнта як добре відомого типу (єдиного виклику або одноелементні множини).
RegisterActivatedServiceType	Перевантажений. Реєструє об'єкт Type з боку служби як об'єкт, який може активуватися за запитом клієнта.

RegisterActivatedClientType	Перевантажений. Реєструє об'єкт Type з боку клієнта як типу, який може активуватися сервером.

Властивості класу

Ім’я	Опис
ApplicationId	Повертає ідентифікатор встановлений у справжній момент програми.
ApplicationName	Повертає або встановлює ім'я програми віддаленого доступу..
CustomErrorsMode	Повертає або задає значення, що визначає, як виконується обробка користувальницьких помилок.
ProcessId	Отримує ідентифікатор встановлений у справжній момент процесу.

Наведемо опис основних методів технології Remoting

Метод RemotingConfiguration.RegisterWellKnownServiceType (WellKnownServiceTypeEntry)

Реєструє об'єкт Type, записаний в наданому WellKnownServiceTypeEntry з боку служби як добре відомого типу.

Список перевантажень

Имя	Опис
RegisterWellKnownServiceType (WellKnownServiceTypeEntry)	Реєструє об'єкт Type як об’єкт добре відомого типу (єдиного виклику або одноелементні множини, активізується сервером за зверненням клієнта).
RegisterWellKnownServiceType (Type, String, WellKnownObjectMode)	Реєструє об'єкт Type з боку клієнта як добре відомого типу, використовуючи параметри, задані для ініціалізації нового екземпляра класу WellKnownServiceTypeEntry

Будь-який клієнт, що знає URI зареєстрованого добре відомого об'єкта, може отримати проксі для об'єкта, реєструючи кращий канал ChannelServices і активуючи об'єкт, викликавши метод new або Activator.GetObject. Клієнтська програма має дві можливості в для того щоб активувати добре відомий об'єкт

- Застосування методу new. Для цього спочатку необхідно зареєструвати тип добре відомого об'єкта, використовуючи метод RegisterWellKnownClientType. Виклик методу RegisterWellKnownClientType дає інфраструктурі віддаленого взаємодії місцезнаходження віддаленого об'єкта, який дозволяє зарезервоване слово new створити його.
- Застосування метод Activator.GetObject. Ветом випадку необхідно задати як аргумент методу наступні параметри: URL об'єкта, попередня реєстрація на стороні клієнта не потрібно.

Коли сервер отримує виклик,. NET Framework витягує URI з повідомлення, досліджує таблиці віддаленого доступу, щоб знайти посилання на об'єкт із заданим URI і, при необхідності, створює екземпляр цього об'єкта, передаючи виклик методу об'єкту. Якщо об'єкт зареєстрований як SingleCall, то він знищується після його завершення методу. Новий екземпляр об'єкту створюється для кожного викликаного методу. Єдина різниця між Activator.GetObject і new у тому, що перший з них дозволяє вказати URL-адресу як параметр, а останній отримує URL-адрес з конфігурації.

Метод RemotingConfiguration.RegisterActivatedServiceType (Type)
Реєструє вказаний тип об'єктів, зареєстрованих з боку служби, в якості типу, який може бути активований за запитом клієнта.
Список перевантаження

Имя	Опис
RegisterActivatedServiceType (ActivatedServiceTypeEntry)	Реєструє тип об'єкта у наданому з боку служби ActivatedServiceTypeEntry в якості типу, який може активуватися за запитом клієнта
RegisterActivatedServiceType (Type)	Реєструє вказаний тип об'єктів, зареєстрованих з боку служби, в якості типу, який може бути активований за запитом клієнта.Параметри System.Type тип об'єкта для реєстрації.

Щоб створити на сервері екземпляр об'єкта, активоване клієнтом, необхідно знати його тип. Цей об'єкт також має бути зареєстрований на сервері з використанням методу RegisterActivatedServiceType. Щоб отримати проксі для екземпляра об'єкта, активованого клієнтом, клієнту необхідно спочатку зареєструвати канал службою ChannelServices, а потім активувати об'єкт, викликавши нову або Activator.CreateInstance.
Щоб активувати тип активируемого клієнтом об'єкта ключовим словом новий, необхідно спочатку зареєструвати тип об'єкта з боку клієнта за допомогою методу RegisterActivatedClientType. Виклик методу RegisterActivatedClientType дає інфраструктурі віддаленого взаємодії місцезнаходження віддаленого додатки, де нові намагається його створити. Якщо для створення нового екземпляра об'єкта, активованого клієнтом, використовується метод CreateInstance, то необхідно задати як параметр URL-адресу віддаленого додатка. Попередня реєстрація на стороні клієнта не потрібно. Щоб надати методом CreateInstance URL-адресу сервера, на якому потрібно створити об'єкт, необхідно включити URL-адресу в екземпляр класу UrlAttribute.
У наведеному нижче прикладі коду показана реєстрація типу об'єкта з боку сервера в якості типу, який може активуватися клієнтом. Для коду клієнта, який відповідає представленому коду сервера, см. приклад для методу
RegisterActivatedClientType.

```
using System;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Tcp;
public class ServerClass {
   public static void Main()  {
      ChannelServices.RegisterChannel(new TcpChannel(8082));
```

```
RemotingConfiguration.RegisterActivatedServiceType(typeof(HelloServiceClass));
        Console.WriteLine("Press enter to stop this process.");
        Console.ReadLine();
    }
}
```

У наведеному нижче прикладі коду показаний об'єкта служби, зареєстрований наведеному вище прикладі коду.

```
using System;
public class HelloServiceClass : MarshalByRefObject {
   static int n_instance;
   public HelloServiceClass() {
      n_instance++;
      Console.WriteLine(this.GetType().Name + " has been created.  Instance # =
{0}", n_instance);
   }
   ~HelloServiceClass() {
      Console.WriteLine("Destroyed instance {0} of HelloServiceClass.",
n_instance);
      n_instance --;
   }
   public String HelloMethod(String name) {
      // Reports that the method was called.
      Console.WriteLine();
      Console.WriteLine("Called HelloMethod on instance {0} with the '{1}'
parameter.",
                        n_instance, name);
      // Calculates and returns the result to the client.
      return "Hi there " + name + ".";
   }
}
```

Метод RemotingConfiguration.RegisterWellKnownClientType

Реєструє об'єкт Type з боку клієнта як добре відомого типу (єдиного виклику або одноелементні множини).

Список перевантаження

Имя	Опис
RegisterWellKnownClientType (WellKnownClientTypeEntry)	Реєструє об'єкт Type, записаний в наданий WellKnownClientTypeEntry з боку клієнта як добре відомого типу, який може бути активований сервером.
RegisterWellKnownClientType (Type, String)	Реєструє об'єкт Type з боку клієнта як добре відомого об'єкта, який може бути активований на сервері, використовуючи дані параметри для ініціалізації нового екземпляра класу WellKnownClientTypeEntry.

Будь-який клієнт, що знає URI зареєстрованого добре відомого об'єкта, може отримати проксі для об'єкта, реєструючи переважний канал ChannelServices і активуючи об'єкт, викликавши нову або Activator.GetObject.
Щоб активувати добре відомий об'єкт новий, спочатку необхідно зареєструвати тип добре відомого об'єкта, використовуючи метод RegisterWellKnownClientType. Виклик методу RegisterWellKnownClientType дає інфраструктурі віддаленого взаємодії місцезнаходження віддаленого об'єкту, який дозволяє зарезервованих слів нову створити його. Якщо для активації добре відомого об'єкта використовується метод Activator.GetObject, то необхідно задати як аргумент URL об'єкта, попередня реєстрація на стороні клієнта не потрібно.

Метод RemotingConfiguration.RegisterActivatedClientType (ActivatedClientTypeEntry) Реєструє об'єкт Type з боку клієнта в якості типу, який може активуватися сервером. Список перевантаження

Имя	Опис
RegisterActivatedClientType (ActivatedClientTypeEntry)	Реєструє об'єкт Type, записаний в наданий ActivatedClientTypeEntry на стороні клієнта в якості типу, який може бути активований сервером.
RegisterActivatedClientType (Type, String)	Реєструє об'єкт Type з боку клієнта в якості типу, який може активуватися сервером, використовуючи дані параметри для ініціалізації нового екземпляра класу ActivatedClientTypeEntry.

Метод реєструє об'єкт Type, записаний в наданий ActivatedClientTypeEntry на стороні клієнта в якості типу, який може бути активований сервером. Параметри Тип: System.Runtime.Remoting.ActivatedClientTypeEntry Параметри конфігурації для активованого клієнтом типу.
Щоб створити на сервері екземпляр об'єкта, активоване клієнтом, необхідно знати його тип. Цей об'єкт також має бути зареєстрований на сервері з використанням методу RegisterActivatedServiceType. Щоб отримати проксі для екземпляра об'єкта, активованого клієнтом, клієнту необхідно спочатку зареєструвати канал службою ChannelServices, а потім активувати об'єкт, викликавши новий.
Щоб активувати тип активируемого клієнтом об'єкта ключовим словом новий, необхідно спочатку зареєструвати тип об'єкта з боку клієнта за допомогою методу RegisterActivatedClientType. Виклик методу RegisterActivatedClientType дає інфраструктурі віддаленого взаємодії місцезнаходження віддаленого додатки, де нові намагається його створити. Якщо для створення нового екземпляра об'єкта, активованого клієнтом, використовується метод Activator.CreateInstance, то необхідно задати як параметр URL-адресу віддаленого додатка. Попередня реєстрація на стороні клієнта не потрібно. Щоб надати методом Activator.CreateInstance URL-адресу сервера, на якому потрібно створити об'єкт, необхідно включити URL-адресу в екземпляр класу UrlAttribute. Детальний опис об'єктів, активованих клієнтом, см. в розділі Client Activation. Дозволи: для конфігурації інфраструктури віддаленого взаємодії. Значення за вимогою: SecurityAction.Demand. Значення за дозволом: SecurityPermissionFlag.RemotingConfiguration.

Контрольні запитання

1. Особливості технології REMOTING
2. Компоненти технології REMOTING

Завдання

1. Визначить бажану функціональність клієнтського додатку
2. Реалізуйте систему клієнт – сервер з використанням архітектури REMOTING мовою C#

Лабораторна робота N 8

Розробка системи з використанням технології WSF

Мета роботи: Вивчити процес створення систем клієнт – сервер технології WSF

Теоретичні відомості

Технологія WCF є засобом для того, щоб корінним чином замінити техніку реалізації розподіленого програмування для розробників, що використовують рішення Microsoft® .NET Framework NET. WCF об'єднує той, що існує засоби розподіленого програмування в .NET у єдину модель програмування, яка надає нові рівні функціональності і функціональної сумісності, і виправдовує всі надії користувачів в цій області . Ця стаття представляє WCF, програмування, і показує, з чого слід почати.

Як випливає з її назви, технологія WCF забезпечує використання .NET у якості підставою для розробки програмного коду, який дозволяє встановлювати зв'язки між компонентами, застосуваннями, і системами. Технологія WCF проектувалася згідно з принципами функціонування служби сервісів в .NET. Сервіс в .NET - частину коду використовуючи яку організовується обмін повідомленнями. Сервіси пасивні. Вони чекають вхідних повідомлень перед виконанням деякої дії . Клієнти - ініціатори. Клієнти відправляють сервісам повідомлення щоб, щоб запитати деяку дію.

Сервіси надають можливість відправлення повідомлення одній або декільком кінцевим точкам. Кожна крайова точка (точки комунікації) складається (визначається) з адреси, з'єднання, і контракту (див. Рис. 1). Позначення кінцевої точки:

– АПК (адреса, прив'язка (з'єднання), контракт)

- ABC (address, binding, contract);

Адреса конкретизує, куди послати треба повідомлення щоб його отримала певна крайова точка. З'єднання описує спосіб передачі (канал), використовуваний для пересилки повідомлення. Контракт описує набір склад повідомлення та функцій, які надаються сервісом кінцевої точки. Клієнтам потрібно дізнатися цю інформацію перед тим звернутися до сервісу. Описані в контракті операції відображаються на методи класу, що реалізовує оконечную точку, і включають зокрема типи параметрів, що передаються кожному методу й отримуваних від нього.

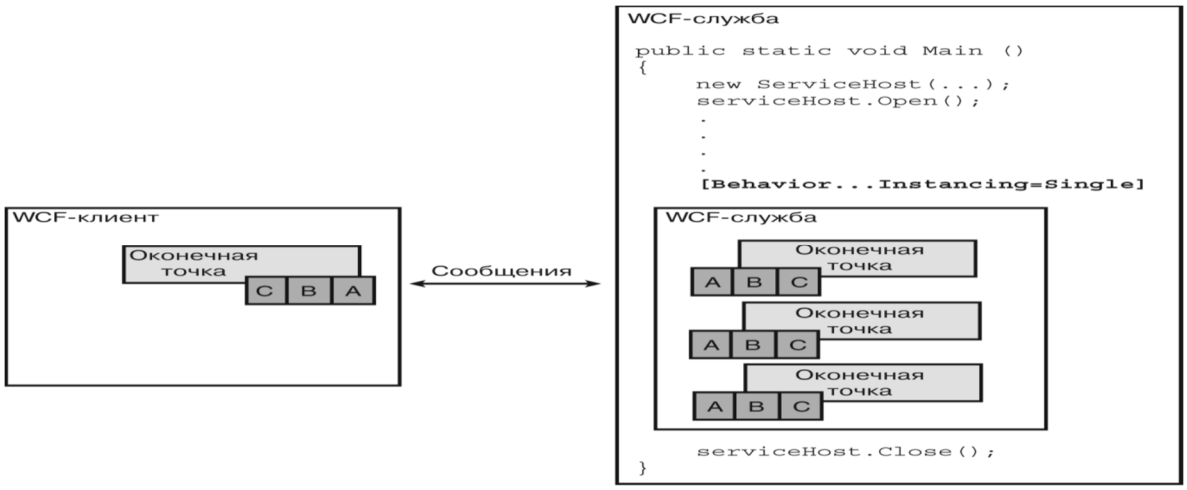


Рис 1 Сервіси і кінцеві точки

Сервіси можуть упакувати опису кінцевих точок, для надання їх розподіленим клієнтам, використовуючи (як правило) мова опису Веб-сервісів (Services Description Language, WSDL). Клієнти можуть використовувати створене опис, для того щоб генерувати код в межах свого середовища, здатний до розсилки і отримання повідомлень

Сервіси можуть упакувати описи кінцевих точок, для надання їх розподіленим клієнтам, використовуючи (як правило) мова опису Веб-сервісів (Services Description Language, WSDL). Клієнти можуть використовувати створений опис, для того щоб генерувати код в межах свого середовища, здатний до розсилки і отримання повідомлень

Технологія WCF надає нову бібліотеку класів, що знаходяться в просторі імен System.ServiceModel яка реалізує сервісно - орієнтовану модель. До цього простору звертаються всі WCF - орієнтовані програми.

У WCF програміст є або розробником сервісу надає клієнтам кінцеві або розробником клієнтів, які взаємодію з кінцевими точками. Таким чином, кінцеві точки є центральною ланкою моделі програмування WCF і її інфраструктури. Зв'язок кінцевих точок моделі WCF з використовуваними класами та інтерфейсами. NET наведена в таблиці.

Модель програмування WCF

ЕЛЕМЕНТ	Використовуваний клас або інтерфейс. NET
Кінцева точка Endpoint	Endpoint System.ServiceModel.ServiceEndpoint
Адреса Address	System.Uri
З'єднання Binding	System.ServiceModel.Binding
Контракт Contract	Interfaces annotated with System.ServiceModel attributes

Формуючи сервіс WCF, ви зазвичай починаєте з визначення. NET інтерфейсу, щоб сформувати контракт (набір методів) сервісу. Потім ви здійснюєте імплементацію контракту сервісу в. NET класі, відомому, як тип севіса, і формуєте його поведінку. Далі, ви визначаєте кінцеві точки, що надаються сервісом, конкретизуєте адресу, з'єднання, і контракт для кожного сервісу. Нарешті, ви розміщуєте тип служби в додатку, що використовує інфраструктури для хостингу (розміщення в робочому стані) WCF. Як тільки тип служби розміщений, клієнти можуть відшукати опису його оконечной точки і почати взаємодію з ним.

Формуючи клієнта WCF, необхідно: по-перше отримати опис цільової

кінцевої точки, до якої потрібно звернутися. Опис кінцевої точки використовується, для того щоб динамічно створити необхідний типізований проксі. WCF надає утиліту SvcUtil.exe для автоматизації цього процесу. Потім можна розробити код звернення до отриманого проксі, для відсилання відповідних повідомлень цільової оконечной точціКонтракт служби і диспетчеризація поведінки

Для реалізації WCF_сервіса потрібно написати клас на одній з мов. NET, а потім забезпечити його атрибутами з простору імен System.ServiceModel. Це простір імен встановлюється разом з. NET 3.0 і містить велику частину коду WCF.

Приклад повної програмної реалізації служби

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Collections.Generic;
using System.Runtime.Serialization;
using System.ServiceModel;

namespace EssentialWCF
{
   [ServiceContract]
  public interface IStockService
   {
      [OperationContract]
      double GetPrice(string ticket);
   }
   public class  StockService:IStockService
      {
       public double GetPrice(string ticket)
      {
          Console.WriteLine("Запрос"+ ticket);
         if (ticket == "tic1") { return 100.25;}

         if (ticket == "tic2") { return 125.25;}
         return 100;

      }
      }
   public class Service
   {
       public static void Main()
      {
         ServiceHost serviceHost = new ServiceHost(typeof(StockService), new
Uri("http://localhost:8000/EssentialWCF"));
         serviceHost.AddServiceEndpoint(typeof(IStockService),          new
BasicHttpBinding(), "");
         Console.WriteLine("Активизируем сервис");
         serviceHost.Open();
         Console.WriteLine("Для завершения нажмте Ввод");
         Console.ReadLine();
         serviceHost.Close();
      }
```

```
    }
}
```

У даному прикладі виконуються наступні дії:

–Визначення контракту. Пишеться клас, який робить дещо корисне, після чого він забезпечується атрибутами WCF. Атрибут [ServiceContract] позначає клас, як контракт. В термінах мови WSDL [ServiceContract] визначає тип порту PortType. Атрибут [OperationContract] визначає методи класу, які можна викликати через інтерфейс служби. Одночасно він визначає, які повідомлення можна передати цим методам і отримати від них. З точки зору WSDL, цей атрибут відповідає розділам операцій і повідомлень. У лістингу визначено клас StockService, що містить єдиний метод GetPrice.

– Визначається крайова точка. У визначенні оконечной точки адресу, прив'язка і контракт задаються за допомогою методу AddServiceEndpoint класу ServiceHost. Адреса ми залишаємо порожнім, це означає, що адреса оконечной точки такий же, як адресу самої служби. В якості прив'язки вказується BasicHttpBinding, сумісна зі специфікацією WS_I BP 1.1 і забезпечує інтероперабельність з більшістю систем, в яких реалізовані Web_служби на базі XML.

– Розміщується сервіс в процесі EssentialWCF, щоб вона могла прослуховувати вхідні запити. У прикладі служба розміщується в консольному додатку за допомогою класу ServiceHost.

Сервіс очікує надходження запитів на адресу http://localhost:8000/EssentialWCF.

Реалізація клієнта цілком в коді

Якщо крайова точка повинна визначити свої АПК, щоб WCF могла розкрити її можливості при запитах з мережі, то клієнт повинен знати АПК, якщо хоче цими можливостями скористатися. Тому при написанні коду, що звертається до крайовим точкам служби, АПК включаються в клієнтську програму. З адресою кінцевої точки все просто - це мережева адреса, на який відправляються повідомлення. Його формат визначений транспортним протоколом, заданим в прив'язці. Прив'язка оконечной точки точно визначає механізм комунікації, який використовує дана точка. У комплект поставки WCF входить ряд заздалегідь конфігурованих прив'язок, наприклад: NetTcpBinding, wsHttppBinding і basicHttppBinding. Контракт визначає точний формат XML, розпізнаваний службою. Зазвичай він задається за допомогою атрибутів [ServiceContract] і [DataContract] у визначенні класу і / або інтерфейсу, а WCF серіалізуются структуру класу у вигляді XML для передачі по мережі.

У лістингу 1,6 приведений код для виклику операції служби. У нього «зашиті» АПК кінцевої точки, що дозволяють скористатися її можливостями. Перш за все, клієнт визначає інтерфейс, до якого збирається звернутися. Визначення інтерфейсу являє собою точний опис того, як звертатися до служби, і включає ім'я операції та її параметри. Далі клієнт створює екземпляр класу ChannelFactory, передаючи його конструктору АПК кінцевої точки. В даному випадку ми вказуємо адресу сервера IIS, в якому розміщена служба, в якості прив'язки задаємо

BasicHttpBinding, а в якості контракту - інтерфейс IStockService. Нарешті, клієнт отримує від фабрики канал для встановлення зв'язку зі службою і «викликає метод» служби.

Приклад реалізації WCF / клієнта цілком в коді

```
using System; using System.Collections.Generic; using System.Linq;
using System.Text;
```

```
using System.ServiceModel;
namespace WCFClient
{
   [ServiceContract]
   public interface IStockService
   {
      [OperationContract]
       double GetPrice(string ticket);
   }
   class Client
    {
      static void Main()
      {
         ChannelFactory<IStockService> myChannelFactory = new
ChannelFactory<IStockService>
            (new BasicHttpBinding(), new
EndpointAddress("http://localhost:8000/EssentialWCF"));
          IStockService wcfClient = myChannelFactory.CreateChannel();
           string ticket;
           ticket = "1";
           while (ticket != "")
          {
             Console.WriteLine("Введите тип билета");
             ticket = Console.ReadLine();
             double p = wcfClient.GetPrice(ticket);
             Console.WriteLine("Price is {0}", p);
          }
      }
   }
```

Контрольні запитання

1. Особливості технології WSF
2. Компоненти технології WSF
3. Технологія створення системи клієнт – сервер за технологією WSF

Завдання:

1. Спроектуйте функціональність серверу додатків систему клієнт – сервер
2. Реалізуйте систему клієнт – сервер з сервером додатків за технологією WSF

Лабораторна робота N 9

Розробка розподіленої системи з використанням архітектури CORBA

Мета роботи: Вивчити процес створення розподіленої системи архітектури CORBA мовою C#

1. Створення системи клієнт – сервер за технологією CORBA

Теоретичні відомості.

Технологія CORBA

CORBA є промисловим стандартом для реалізації функціональності міжплатформову взаємодії гетероморфний систем. Основні розділи цього стандарту - це стандарт для брокерів об'єктних запитів (CORBA ORB), стандарт мови визначення інтерфейсів OMG IDL, і стандарт переносимого протоколу взаємодії між ORB (IIOP).

Достоїнствами стандарту CORBA є висока переносиміть, гнучкість, ефективність і масштабованість рішень заснованих на ньому, а також наявність великої кількості (часто вільних) реалізацій брокерів об'єктних запитів під різні програмно-апаратні платформи і технології розробки.

Недоліками цього стандарту є його безпосередня складність (і, іноді, неоднозначність), складність навчання розробці на основі цього стандарту, і високий рівень спадкоємності багатьох недоліків (як і достоїнств) технологій C + + і Java.

На сьогоднішній день можна відзначити рівень розвитку стандарту CORBA і його реалізацій, достатній для того, щоб будувати на основі нього модулі взаємодії (компонентів) промислових інформаційних систем.

Таким чином, при наявності відповідних інструментальних засобів, на даний момент технологію CORBA можна досить вигідно використовувати як інтеграційне засіб для наведення програмних мостів між гетероморфний системами. Далі в статті розповідається, як цю технологію можна застосувати розробникам, що використовують (або збираються використовувати) Microsoft. NET для розробки компонентів масштабних інформаційних систем.

IIOPNet

IIOPNet є засобом, що дозволяє програмним компонентам, побудованим на базі платформи Microsoft. NET, використовувати багато можливості стандарту CORBA. IIOPNet розповсюджується за ліцензією LGPL.

Перевагою використання IIOPNet є наявність простих, не відвідних розробника в сторону від ідіом і засобів. NET, способів використання можливостей CORBA при його допомозі.

IIOPNet використовує архітектуру. NET Remoting як базову, що дозволяє використовувати протокол IIOP для маршалинга викликів методів віддалених об'єктів, розміщених не тільки на серверах CORBA, але і на серверах. NET Remoting. Крім того, при роботі з IIOPNet проявляються багато переваги архітектури Remoting в порівнянні з «рідної» середовищем CORBA. Наприклад, при роботі з IIOPNet можна і потрібно використовувати ідіоми Remoting, пов'язані з часом життя серверних об'єктів і збором сміття на сервері (пов'язані з інтерфейсом ILease і т.д.); реалізація ж подібного механізму, наприклад, в C + +, є дуже трудомісткою (особливо в налагодженні) завданням.

У IIOPNet є два основних варіанти використання.

1. Реалізація CORBA-клієнта на базі. NET. За допомогою бібліотеки IIOPChannel, що входить до складу IIOPNet реєструється канал підтримки IIOP; після цього можна використовувати сервер CORBA, що працює на будь-якій платформі також прозоро, як це відбувається при застосуванні Remoting. При цьому необхідно, щоб розробляється посилалося на збірку з проксі-класами, скомпільовану за допомогою IIOPNet з даних IDL-файлів CORBA-сервера.
2. Реалізація CORBA-сервера на базі. NET. У цьому випадку використовується та ж бібліотека IIOPChannel, але подальший код публікує реалізацію серверних інтерфейсів. Після цього сервер може використовуватися з додатків, написаних на будь-яких мовах, підтримуваних виробниками CORBA ORB. Для такого використання

необхідно, щоб малися опису IDL, відповідні публікованим інтерфейсам сервера. Ці описи можуть бути створені вручну (з подальшою трансляцією в. NET проксі-збірку), або згенеровані з публікованих. NET інтерфейсів розроблювального серверного додатка. Для реалізації описаних вище варіантів використання потрібно написати всього декілька рядків коду (як це буде видно в прикладах далі).

Для опису інтерфейсів сервера в рамках технології CORBA використовується мова OMG IDL. Для опису інтерфейсів в додатках на базі. NET використовуються власне інтерфейси. NET, інформація про які доступна у формі метаданих. Хороша сторона IIOPNet полягає в тому, що цей засіб містить інструменти, які дозволяють перетворювати інформацію про інтерфейси з одного виду в інший. Таким чином, для кожного з описаних вище варіантів використання в залежності від характеру первинної інформації про інтерфейси потрібно використовувати один із цих засобів - CLSToIDLGenerator або IDLToCLSCompiler.

Приклади систем, що складаються з різнорідних компонентів

Для забезпечення кроссплатформенного взаємодії різних компонентів необхідно, щоб способи їх взаємодії (набір інтерфейсів) були описані в незалежному від платформи вигляді, і це опис було доступно для будь-якої з взаємодіючих сторін.

У стандарті CORBA для опису інтерфейсів використовується мова OMG IDL. Його синтаксис успадкований від C + +. Ось приклад дуже простого IDL-файла First.idl.

```
#ifndef __org_uneta_iiopnet_examples_first__
#define __org_uneta_iiopnet_examples_first__
#pragma prefix "Org.Uneta.Iiopnet.Examples"

module First
{
    interface IHello
    {
        wstring SayHello(in wstring name);
    };
};

#endif
```

Тут визначаються модуль First з префіксом Org.Uneta.Iiopnet.Examples, всередині якого знаходиться інтерфейс IHello, що містить єдиний метод SayHello, що приймає строковою параметр і повертає рядок.

Такі IDL-файли можуть бути написані вручну (вивчення IDL не представляє великої праці для розробників, знайомих з синтаксисом C + + або C #), або згенеровані з описів NET -. Інтерфейсів.

Основне застосування IDL-описів на стороні NET-додатків в разі їх ручного написання -. Це їх компіляція в NET-збірки і подальше використання цих зборок для прозорої асоціації з інтерфейсами CORBA .. У разі IDL-файлів, генерованих з. NET-збірок, компілювати їх, природно, не доводиться.

За допомогою такої команди можна скомпілювати наведений вище IDL-файл в готову. NET збірку з допоміжними класами. IDLToCLSCompiler.exe First First.idl

При цьому на виході ми отримаємо. NET-збірку First.dll. При її дизассемблирования можна побачити наступне.

Результат дизассемблированияОтже, в даній збірці міститься один простір імен (префікс плюс ім'я модуля з IDL-файла), в якому міститься інтерфейс IHello, успадкованих від Ch.Elca.Iiop.Idl.IIdlEntity і позначений спеціальним атрибутом.

Створення простого CORBA-клієнта на C #Розглянемо створення дуже простого консольного застосування на C #, яке, використовуючи збірку First.dll, отриману методами, описаними вище, бібліотеку IIOPChannel, і відомості про локації CORBA-сервера, який реалізує наявні інтерфейси (IHello), з'єднується з CORBA-сервером, запитує з консолі ім'я користувача, отримує посилання на клієнтський проксі CORBA-об'єкта, і викликає у нього викликає метод SayHello інтерфейсу IHello, передаючи в нього як параметр отриману від користувача рядок. Повернута методом рядок виводиться на консоль і є CORBA-вітанням користувачеві.Нижче приведений вихідний код такого додатка.using System;

```
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting;
using Ch.Elca.Iiop;

namespace Org.Uneta.Iiopnet.Examples.First
{
public class FirstClient
{
    [STAThread]
    public static void Main(string[] args)
    {
            try
            {
                    // Адрес CORBA-сервера.
                    const string serverHost = "localhost";
                    const int serverPort = 1234;

                    // Запрашиваем имя пользователя.
                    Console.WriteLine("Введите ваше имя: ");
                    string userName = Console.ReadLine();

                    // Регистрируем канал IIOP.
                    IiopClientChannel channel = new IiopClientChannel();
                    ChannelServices.RegisterChannel(channel);
```

```
                    // Адрес CORBA-сервера.
                    string addressString = "iiop://localhost:1234/hello";
                    // Получаем ссылку на клиентский прокси.
            IHello hello = (IHello)RemotingServices.Connect(typeof(IHello),
addressString);


            // Вызываем CORBA-метод.
                    string serverResponse = hello.SayHello(userName);

                    // Выводим ответ.
                    Console.WriteLine("Ответ сервера: " + serverResponse);
                    }
                    catch (Exception e)
                    {
                            Console.WriteLine("Возникло исключение: " + e);
                    }
            }
    }
}
```

З даного прикладу можна зрозуміти, що IIOPNet активно використовує засоби Remoting для реалізації своєї функціональності. Для того, щоб використовувати протокол IIOP в роботі нам потрібно було всього лише зареєструвати канал за допомогою класів ChannelServices і IiopClientChannel. Далі ми користуємося звичайним викликом методу Connect () класу RemotingServices для отримання посилання на клієнтський проксі CORBA-об'єкта. Адреса для отримання посилання формується з імені хоста, номера порту, і «імені об'єкта» CORBA ("привіт")Поглянувши на код такого додатка, можна здивуватися елегантності способу роботи з CORBA з. NET. Порівнявши цей код з аналогічним кодом, скажімо, на C + +, можна сильно вразив простоті.

Результат роботи цього додатка показаний нижче.

```
Введіть ваше ім'я
Женя
Відповідь сервера:
Привет от CORBA, Женя
Prees any key to continue_
```

Створення простого CORBA-сервера на C #

У даному прикладі ми створимо CORBA-сервер для клієнта, наведеного вище.

Серверний додаток повинно визначити клас-реалізацію для інтерфейсу IHello, зареєструвати серверний канал IIOP, створити екземпляр такої реалізації інтерфейсу, опублікувати його за допомогою зареєстрованого каналу, і чекати клієнтських запитів. Вихідний код такого додатка показаний нижче.

```
using System;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using System.Threading;
using Ch.Elca.Iiop;

namespace Org.Uneta.Iiopnet.Examples.First
```

```
{
    public class HelloImplementation : MarshalByRefObject, IHello
    {
                public override object InitializeLifetimeService()
                {
                        // Жизнь не кончается.
                        return null;
                }

                public string SayHello(string name)
                {
                        return "Привет от CORBA, " + name + ".";
                }

    }

    public class FirstServer
    {
            [STAThread]
            public static void Main(string[] args)
            {
                    // Регистрируем серверный канал IIOP.
                    int serverPort = 1234;
                    IiopChannel channel = new IiopChannel(serverPort);
                    ChannelServices.RegisterChannel(channel);

                    // Создаем реализацию интерфейса IHello и публикуем
её.
    HelloImplementation helloImplementation = new HelloImplementation();
                            string objectURI = "hello";
            RemotingServices.Marshal(helloImplementation, objectURI);

                    Console.WriteLine("Сервер готов.");
                    Thread.Sleep(Timeout.Infinite);
                    }
    }
}
```

Як і попередній, даний приклад дає уявлення про простоту використання IIOPChannel для роботи з CORBA з. NET.

Контрольні запитання

1. Наведіть склад сервера технології CORBA
2. Наведіть Склад клієнта технології CORBA

Завдання:

1. Спроектуйте функціональність серверу додатків систему клієнт – сервер
2. Реалізуйте систему клієнт – сервер сервером додатків за технологією CORBA мовою C#

Рекомендована література

Базова

1. Конспект лекцій
2. Дейт К. Дж. Введение в системы баз данных.: Пер. с англ. - 6-е изд. - К.:Диалектика, 1998. - 784 с. : ил.
3. Арсеньев Б.П. , Яковлев С.А. Интеграция распределенных баз данных . СПб.:Издательство "ЛАНЬ", 2001г.464с
4. Грейди БУЧ, Джеймс Рамбо, Айвар Джекобсон. Язык UML . Руководство пользователя . ДМК – М.:,2000.-432 с.
5. Елманова Н., Трепалин С., Тенцер А технология COM . СПб.: Питер, 2002.-640с – (Серия "Мастер- класс")
6. C# 4.0 и платформа .NET 4 для профессионалов: Диалектика, Вильямс 2011 стр 1440
7. Пол Нильсен SQL SERER 2005 Библия Пользователя. М ООО « И..Д. Вильямс», 20008, 1232стр
8. Петкович Д.MS SQL SERVER 2012/. BHV-СП6, 2013 стр 816
9. В.І. Гайдаржи, О.Є. Круш. Компонентне програмування-К.:Знання Украіни, 2010, стр 247
10. Вячеслав Понамарев. Программирование на C++/C# в Visual Studio .NET 2003.: БХВ-Петербург Год издания: 2004
11. Кристиан Нейгел, Билл Ивьен, Джей Глинн, Карли Уотсон, Морган Скиннер . C# 4.0 и платформа .NET 4 для профессионалов. Диалектика. Год издания: 2011.- 1440 стр., с ил.;,

Додаткова література

1. Стивен Бобровски. Oracle7 и вычисления клиент/сервер: -М.: ЛОРИ. - 652 с.
2. Кен Хендерсон . DELPHI 3 и системы КЛИЕНТ/СЕРВЕР. Руководство разработчика. Киев “Диалектика” 1997.-735с
3. Мэтью Мак-Дональд, Адам Фримен, Марио Шпунта. Microsoft ASP.NET 4.0 с примерами на C# 2010 для профессионалов. Из–во Вільямс. Год издания: 2011.- .1424 стр., с ил.;

Інформаційні ресурси

1. http://otimtp.nltu.edu.ua/index.php/using-joomla/extensions/components/content-component/article-categories/89-dystsypliny/dystsypliny-mahistra/216-proektuvannia-rozpodilenykh-baz-danykh-ta-ekspertnykh-system
2. http://studbase.com/books/1/44/ РОЗПОДІЛЕНІ БАЗИ ДАНИХ
3. http://www.rusnauka.com/7_NND_2009/Informatica/42665.doc.htm К.т.н., ст.преподаватель Исмаилов Х.Б., Алимбекова А.Т. Южно-Казахстанский государственный университет им. М. Ауезова, Казахстан Проектирование распределенных баз данных информационных систем