МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ХАРЧОВИХ ТЕХНОЛОГІЙ

ЗАТВЕРДЖУЮ

в.о.Ректора_____ Іванов С.В.

_____ _____2011р.

О.М. ПУПЕНА

ПРОМИСЛОВІ МЕРЕЖІ ТА ІНТЕГРАЦІЙНІ ТЕХНОЛОГІЇ

КУРС ЛЕКЦІЙ

ЧАСТИНА2. ІНТЕГРАЦІЙНІ ТЕХНОЛОГІЇ.

ля студентів напряму 6050202
«Автоматизація та комп'ютерно-інтегровані технології» денної та заочної форм навчання

Реєстраційний номер
електронних методичних вказівок
у НМУ _____

СХВАЛЕНО
на засіданні кафедри автоматизації
та комп'ютерно-інтегрованих
технологій як методичні вказівки
Протокол №___ від _____2011 р.

КИЇВ НУХТ 2011

О.М. ПУПЕНА. Промислові мережі та інтеграційні технології: курс лекцій для студ. напряму 6050202 «Автоматизація та комп’ютерно-інтегровані технології» денної та заочної форм навчання. Частина 2. Інтеграційні технології. – К.: НУХТ, 2011. – 34 с.

Укладачі: **О.М. Пупена,** канд. техн. наук

.

6. ВІДКРИТІ ТЕХНОЛОГІЇ ПРОГРАМНОЇ ІНТЕГРАЦІЇ В СЕРЕДОВИЩІ WINDOWS.

ПЛАН.

6.1. Проблеми програмної інтеграції в ІАС
6.2. Проблеми доступу до даних іншого Процесу
6.3. Технології DDE та NetDDE
6.4. Технології COM/DCOM

6.1. Проблеми програмної інтеграції в інтегрованих автоматизованих системах

Враховуючи визначені стандарти промислових мереж та незалежність сучасних програмних продуктів від апаратної платформи комп'ютерів, проблема апаратної сумісності між рівнями АСУТП та АСУП як правило не розглядається. Зовсім інша ситуація склалася на ринку програмних засобів. Програмне забезпечення інтегрованих автоматизованих систем управління на рівні АСУП можна умовно віднести до однієї з наступних груп: універсальні та спеціалізовані СУБД; системи рівня MES; системи рівня ERP; офісне програмне забезпечення.

До програмних засобів на рівні АСУТП, що потребують інтеграції з іншими ПЗ можна віднести SCADA/HMI та спеціалізовані СУБД реального часу. Задача програмної інтеграції зводиться до забезпечення взаємозв'язку між програмами для їх узгодженої роботи, що приводить до необхідності створення інформаційного каналу між даними в цих програмах.

На ринку програмних продуктів для автоматизованих систем не має єдиного стандарту для вирішення даної задачі. Великі компанії-розробники спеціалізованого ПЗ для промислової автоматизації впроваджують програмні пакети для різного рівня автоматизації, які порівняно легко інтегруються між собою. Крім високої вартості даних рішень, вони теж не вирішують всі необхідні задачі. Все це приводить до необхідності інтеграції різних програмних продуктів, розробники яких можуть навіть не знати про існування один одного. Таким чином крім проблем інформаційної та функціональної інтеграції виникає проблема програмної сумісності

У даній частині курсу розглядаються сучасні відкриті технології інтеграції між програмними засобами, які забезпечують вирішення задач доступу до даних інших програм.

6.2. Проблеми доступу до даних іншого Процесу.

Кожний із ***Процесів*** (Process) в операційній системі має свій адресний простір, тому передача даних між двома різними Процесами – задача не тривіальна і вирішується декількома шляхами. Спеціалістам з автоматизації більш цікаві стандартні підходи, які не потребують написання програм (принаймні складних), а тільки конфігурування для кожного з процесів. Наприклад, для зв'язку таблиці Excel з потрібними даними в базі даних Access, з

подальшим автоматичним відновленням інформації, можна скористатися ресурсами самого Excel. Для того, щоб гнучко використовувати такі вбудовані в програмні засоби можливості, необхідне розуміння внутрішньої сутності функціонування. У зв'язку з цим коротко розглянемо базові поняття функціонування відкритих технологій міжпрограмної (міжпроцесної) взаємодії, а саме DDE/NetDDE та COM/DCOM.

6.3. Технології DDE та NetDDE

6.3.1. Локальний обмін через DDE

Походження. ***DDE*** (Dynamic Data Exchange – динамічний обмін даними) – технологія, яка базується на зв'язку між прикладними програмами з використанням передачі віконних повідомлень (WM - Windows Message).

Перші прикладні програми, які користувалися цією технологією безпосередньо передавали, отримували і обробляли віконні повідомлення спеціального типу, виділені для DDE. Пізніше Microsoft розробила для цього спеціальний набір функцій бібліотеки динамічного обміну даними (***DDEML*** – DDE Management Library), які підтримували сумісність із старими прикладними програмами.

Взаємодія між процесами. Коли дві прикладні програми обмінюються даними за допомогою DDE, то кажуть, що вони зайняті в ***DDE сеансі зв'язку*** (DDE conversation). Ту програму, яка починає сеанс зв'язку називають ***Клієнт DDE*** (DDE client application), а яка відповідає – ***Сервер DDE*** (DDE server application).

Сеанс зв'язку проходить між двома вікнами, по одному для кожного з програм-учасників. Вікно може бути сховане, наприклад у випадку, коли воно створюється тільки для обміну по DDE. Прикладна програма може приймати участь в декількох сеансах зв'язку у той самий час як в якості Клієнта, так і в якості Сервера. Однак для кожного обміну використовується тільки одне вікно як для Клієнта так і для Сервера (наприклад 10 сеансів - це 10 клієнтських вікон + 10 серверних вікон на дві прикладні програми).

Модель ідентифікації даних. Для ідентифікації модуля даних, до яких доступається Клієнт на Сервері, використовується символьне ім'я, яке складається з трьох частин:

- назва прикладної програми (***Application***);
- назва документу або розділу (***Topic***);
- назва елементу даних (***Item***).

На початку сеансу зв'язку DDE Клієнт і Сервер визначають назву прикладної програми і документа. Наприклад, якщо в якості програми-Сервера виступає Microsoft Excel, то назва прикладної програми буде "Excel". Документ в такому випадку може складатися з назви файлу та листа, а якщо файл Excel вже завантажений, то просто з назви листа, наприклад"Лист2". Слід зауважити, що за допомогою цих двох частин назви встановлюється сеанс зв'язку, тобто визначаються вікна Клієнта та Сервера. Це значить, що протягом цього сеансу, наведені назви змінитися не можуть.

Елемент даних DDE – це інформація, яка пов'язана з документом і приймає участь в обміні. Назва цього елементу теж визначається в залежності від

прикладної програми. Так, наприклад, в Excel елемент даних – це комірка на листі, яка адресується номером рядка та колонки (наприклад "R1C1"). Значення елементу даних може передаватись як від Клієнта до Серверу, так і навпаки. Формат даних може бути будь яким, який підтримується буфером обміну.

Системний розділ (System). Будь яка прикладна програма-Сервер DDE повинна підтримати розділ (Topic) з назвою "System". Цей розділ забезпечує контекстною інформацією, яка може представляти інтерес для програми-Клієнта. У табл. 6.1 перераховані деякі назви елементів даних, які підтримуються даним розділом.

Таблиця6.1

Призначення елементів даних.

Item	*Призначення*
Formats	Список підтримуваних форматів буферу обміну даного DDE-серверу
Help	Коротка допомога по використанню DDE
Status	Інформація про стан прикладної програми
SysItems	Список елементів даних, які підтримуються прикладною програмою для розділу System
Topics	Список розділів (документів), які доступні для даної програми в даний момент часу. Він від часу до часу може змінюватись.

Приклад 6.2. Технології програмної інтеграції. Доступ до комірки Excel з використанням DDE.

Завдання. Вказати адресу даних в комірці Excel для доступу по DDE.

Рішення.Щоб доступитися до комірки даних Microsoft Excel у 2 колонці та у 3 рядку на листі "Лист2" необхідно вказати:

- Application – Excel;
- Topic – Лист2;
- Item – R3C2.

Якщо необхідно вказати ще назву файла, то Topic – [Книга1.XLS]Лист2. Часто формат запису сполучає назву програми та документу:

Excel|Лист2 або *Excel|[Книга1.XLS]Лист2*

Інколи необхідний модуль даних вказується повністю. Наприклад, Excel може бути Клієнтом DDE. Тому якщо на комп'ютері завантажені дві програми Excel, можна обмінюватися даними між їх комірками, вказавши в комірці одної з них:

= Excel|Лист2!'R3C2'

Зверніть увагу, що в Excel деякі назви беруться в апострофи.

6.3.2. Обмін через NetDDE в мережі

Основи функціонування. ***Network DDE*** або просто ***NetDDE*** використовується для обслуговування мережних зв'язків, які потрібні для сеансів зв'язку DDE між прикладними програмами, що виконуються на різних комп'ютерах у мережі.

Для запуску обміну через NetDDE необхідно, щоб в системі комп'ютера був завантажений ***агент NetDDE*** (NETDDE.EXE). Зверніть увагу, що програма-агент не запускається автоматично при активації NetDDE, тому її треба запустити як на комп'ютері Клієнта, так і на комп'ютері Сервера DDE. В операційній системі Windows вона запускається у вигляді двох служб: "Служба сетевого DDE" та "Диспетчер сетевого DDE", тому їх треба активувати та

запустити. Агент при необхідності запускає прикладну програму, на якій розміщені дані. Весь обмін проходить саме через вікно програми-агента, тому він є проміжним Сервером (proxy-server), який замінює сторону дійсного Серверу, перенаправляючи всі повідомлення його вікну.

Виділення загальних ресурсів. ***DDE Shares*** – загальні ресурси DDE, якими можуть користуватися програми-Клієнти DDE на даному комп'ютері. Тобто, якщо програма-Клієнт хоче доступитися до даних програми-Сервера на іншому комп'ютері, необхідно на останньому прописати ресурси, з якими можна зв'язатися та їх загальну (shared) назву, яка завжди закінчується символом "$". На комп'ютері програми-Сервера DDE, в базі даних DSDM**,** ресурси зберігаються у вигляді записів типу "назва загального ресурсу" - "розміщення ресурсу". Таким чином, програма-Клієнт буде проводити сеанс з'єднання через ці ресурси.

Для налаштування загальних ресурсів існує спеціальна програма DDESHARE.EXE (вона запускається з командного вікна):

- добавляється новий ресурс, вказується ім'я з "$" в кінці (наприклад "SHARA$");
- вказується назва програми та документ (розділ) модуля даних DDE;
- настроюються опції: автоматичний пуск програми і ін.

Слід врахувати особливості захисту на рівні операційної системи. Конфігурування загальних ресурсів можна проводити віддалено, вказавши необхідний комп'ютер в конфігураторі.

Типи загальних ресурсів. Є три типи загальних ресурсів: по старому стилю, по новому стилю та статичний тип. Вони відрізняються по правилам ідентифікації ресурсів. Як правило всі прикладні програми використовують статичний тип. Старий стиль призначений для програм, які реалізовують DDE сеанс зв'язку за допомогою віконних повідомлень, а новий стиль – на основі бібліотеки DDEML.

З боку Клієнта, при ідентифікації ресурсу DDE вказується:

- Application – \\ім'я_комп'ютера\NDDE$ (наприклад \\COMP1\NDDE$);
- Topic – ім'я_загального_ресурсу (наприклад SHARA$);
- Item – назва_елементу_даних (наприклад R3C2).

Область використання DDE/NetDDE. DDE та NetDDE використовується не тільки для обміну між комірками Excel. Дана технологія широко використовується у програмному забезпеченні для АСУТП та АСУП. Зокрема більшість SCADA-програм надають доступ до своїх даних через DDE, та навпаки – можуть бути Клієнтами DDE. На сьогоднішній день технологія DDE все більше витісняється новими технологіями, такими як COM/DCOM, .NET та іншими.

Приклад 6.3. Технології програмної інтеграції. Доступ до загального ресурсу через NetDDE.

Завдання. Забезпечити створення загального ресурсу на комп'ютері "COMP1" для доступу до комірки Excel, аналогічно прикладу 6.2.

Рішення. Створюємо на ПК Сервері загальнодоступні ресурси, для чого у DDESHARE.EXE виконаємо таку послідовність:

- добавимо ресурс з іменем "SHARA" (назви писати без лапок);

- назва прикладної програми: статичний зв’язок = Excel, всі інші порожні;
- назва документу: статичний зв’язок = Лист2 або статичний зв’язок = [Книга1.XLS]Лист2, всі інші порожні;
- виставити опцію: з дозволу запускати прикладну програму.

З боку Клієнта, при ідентифікації ресурсу DDE вказується:
- Application – \\COMP1\NDDE$;
- Topic – SHARA$;
- Item – R3C2.

Таким чином, для зв’язку двох комірок Excel на різних комп’ютерах, на першому (Серверний бік) налаштовуємо загальний ресурс, а на другому в комірці-приймачі вказуємо рядок:

='\\COMP1\NDDE$'| SHARA$'*!'R3C2'*

Перед цим слід на обох комп’ютерах запустити необхідні служби (NETDDE.EXE).

6.4. Технології COM/DCOM

6.4.1. Доступ до Процесів через COM.

Загальні поняття. ***COM*** (Component Object Model - компонентна модель об’єктів) – це об’єктно-орієнтована технологія, яка дозволяє одній прикладній програмі користуватися об’єктами іншої прикладної програми або бібліотеки. ***DCOM*** (Distributed Object Model – розподілена COM) додала можливість прозорого використання цих об’єктів на віддалених вузлах. У чистому вигляді технологія COM має практичне використання тільки при програмуванні. Тим не менше розуміння фундаментальних засад дозволяє уникнути проблем при використанні похідних від неї технологій.

Технології на базі COM. Сама по собі модель COM реалізована у вигляді бібліотек, які завантажуються з Процесом. Компонент COM – це програмний код, який знаходиться в бібліотеці DLL (OCX) або у виконавчій програмі (типу EXE).

Спочатку технологія розроблялася для підтримки складних документів, наприклад для можливості редагування таблиці Excel в документі Word. Це привело до появи технології ***OLE*** (***Object Linking and Embedding*** – зв’язування та вміщення об’єктів), перша версія якого базувалася на DDE. Оскільки DDE не забезпечувала необхідну гнучкість, наступна версія вже базувалася на COM.

Окрім термінів COM та OLE існують також COM+, OLE Automation та ActiveX. ***COM+*** можна вважати як розширений варіант моделі COM, який вміщує різноманітні служби, які раніше були корисними доповненнями до COM.

OLE Automation – реалізація COM, що вимагає наявність спеціального типу DISP-інтерфейсу. DISP-інтерфейс дає можливість реалізувати механізм пізнього зв’язування з об’єктами COM, що дає можливість використовувати їх в VB, VBA, VB Script, JAVA script та інших системах програмування.

Терміном ***ActiveX*** називають все, що відноситься до OLE Automation, плюс деякі допоміжні можливості, зокрема підтримку сценаріїв та використання ***елементів управління ActiveX***. Останні, до речі, дуже широко використовуються у різноманітних програмах із області промислової автоматизації (SCADA-, MES-системи і т.п.). Хоч наведені терміни широко вживаються, на сьогоднішній день немає чіткого визначення терміну ActiveX, тому часто з цим словом пов’язують все, що стосується OLE Automation та ActiveX.

COM-інтерфейси. Доступ до методів об'єкту COM проводиться через ***COM-інтерфейси***. Під COM-інтерфейсом (надалі інтерфейсом) можна розуміти умовний логічний канал доступу до методів об'єкта. Програма, яка користується цим об'єктом не викликає *його* методи (методи класу), а викликає методи *інтерфейсу*.

Кожний інтерфейс має унікальний 128-бітний ідентифікатор, який зветься ***ідентифікатором інтерфейсу*** (***IID*** - Interface Identifier). Кожний інтерфейс абсолютно унікальний, тобто немає двох інтерфейсів з однаковим ідентифікатором. Якщо два об'єкти підтримують інтерфейси з однаковими ідентифікаторами, вони повинні мати одні і ті самі методи інтерфейсів, які призначені для однієї й тієї ж самої дії. Це значить, що внутрішня реалізація об'єктів може бути різною, а інтерфейси та поведінка - співпадати.

COM-класи. Об'єкти (екземпляри) в COM створюються на основі ***класу об'єкту***. Для людей, знайомими з об'єктно-орієнтованим програмуванням, зрозуміло, що об'єкт – це екземпляр класу. COM об'єкти, які реалізують один і той самий набір COM-інтерфейсів відносяться до одного класу. Кожний клас має унікальний ***ідентифікатор класу*** (***CLSID*** – Class Identifier).

Взаємодія між Процесами через COM. Взаємодія між прикладними Процесами функціонує на базі моделі Клієнт-Сервер. Програми, в яких знаходяться та створюються COM об'єкти називаються ***COM-Серверами*** а програми які користуються цими об'єктами - ***COM-Клієнтами***. В залежності від того де знаходиться програма COM-Сервер по відношенню до COM-Клієнта, виділяють такі типи Серверів:

- ***Внутрішній Сервер*** (***In-Process Server***) – Сервер який знаходиться в середині того самого Процесу, що і Клієнт (наприклад реалізований як бібліотека DLL);
- ***Локальний Сервер*** (***Local Server***) – Сервер який запускається як окремий Процес (EXE модуль);
- ***Віддалений Сервер*** (***Remote Server***) – Сервер, який запускається на віддаленому вузлі в мережі (EXE модуль, або бібліотека DLL в сурогатному Процесі);

Всі класи та інтерфейси реєструються в системному реєстрі Windows. Там крім їх ідентифікатора зберігається додаткова інформація, зокрема про імена файлів, в яких реалізований даний клас. Для доступу до них та їх активації використовується спеціальний процес SCM (диспетчер управління службами).

Контроль доступу до об'єктів COM. Для захисту доступу до об'єктів класів використовується так званий декларативний захист. Механізм активізації і контролю доступу базуються на механізмі ролей. Для кожного зареєстрованого класу записи в реєстрі визначають, які користувачі Windows або групи користувачів мають право створювати екземпляри даного класу, тобто об'єкти. Користувачі визначаються при аутентифікації, тобто при їх перевірці, наприклад вводу паролю при вході в Windows. DCOM підтримує декілька рівнів аутентифікації:

- без аутентифікації;
- при першому з'єднанні з сервером;
- при кожному зверненні;

- всіх пакетів даних;
- пакетів даних з перевіркою їх цілісності;
- пакетів даних з перевіркою їх цілісності і шифрування пакетів даних.

Для настройки захисту у Windows є спеціальна утиліта *Dcomcnfg.exe*, яка запускається з командного рядку.

6.4.2. Використання OLE та ActiveX.

Від користувачів технології COM часто приховані механізми його функціонування. Так, при створенні мнемосхеми на SCADA можна скористатися елементами ActiveX, розмістивши їх на формі (сторінці, мнемосхемі ...) та прив'язавши їх властивості до певної змінної. Звісно цей механізм повинен підтримуватися даною SCADA-програмою, а потрібні елементи ActiveX – присутні на комп'ютері. За допомогою OLE можна помістити на мнемосхему таблицю Excel, документ типу PDF або інші OLE-об'єкти.

Якщо у складі програмного пакета (SCADA, офісна програма, CAD) є вбудована VBA, то програма написана на цій мові може працювати з об'єктами основного програмного пакету теж за допомогою ActiveX. Крім того, межі використання програмного пакету можна значно розширити, за рахунок використання додаткових елементів ActiveX.

Розглядаючи ActiveX треба зазначити, що його поява пов'язана з бурхливим розвитком Інтернет-технологій, де елементи ActiveX можуть використовуватись для оживлення Інтернет-сторінок. Однак в цій технології є ряд недоліків. Так наприклад, заради безпеки вузлів в мережі, брандмауери закривають доступ DCOM пакетам. Це обмежує його використання в мережі Інтернет. Знову ж таки з причин надійності, кращою альтернативою ActiveX елементам в WEB-сторінках можуть бути JAVA-аплети. Адже код, який поставляється в ActiveX навмисно або ненавмисно може пошкодити дані на комп'ютері. Microsoft ввели додатковий механізм захисту під назвою "Цифровий підпис", однак часто навіть "підписані" програми можуть призвести до збоїв в роботі системи. JAVA-аплети ж виконуються на віртуальній машині, тому більш надійні в порівнянні з ActiveX.

<u>Приклад 6.4</u>. Технології програмної інтеграції. Використання ActiveX компонентів.

Завдання. В SCADA Citect забезпечити можливість вибору оператором необхідної дати в зручному графічному інтерфейсі, для подальшого її використання в перегляді історичних трендів. Для цього використати ActiveX елемент "Microsoft Date AND Time Picker Control".

Рішення. Спочатку в Citect Editor визначимо три внутрішні змінні типу INT: SEL_DAY, SEL_MONTH, SEL_YEAR. Ці змінні плануються для використання в мові Cicode для визначення початкової дати перегляду трендів. В Graphics Builder на панелі Tools, вибираємо елемент ActiveX, після чого з'явиться список встановлених на ПК елементів ActiveX. Необхідно вибрати елемент "Microsoft Date AND Time Picker Control", і вибравши вкладку "Appearance", закладку "TagAssociation", у вікні "Propertiies" настроїти наступні властивості:

- властивість Day, вибрати тег SEL_DAY, в полі "Update association on" вибрати "Change" (рис.6.1);
- властивість Month, вибрати тег SEL_MONTH, в полі "Update association on" вибрати "Change";
- властивість Date, вибрати тег SEL_DATE, в полі "Update association on" вибрати "Change";

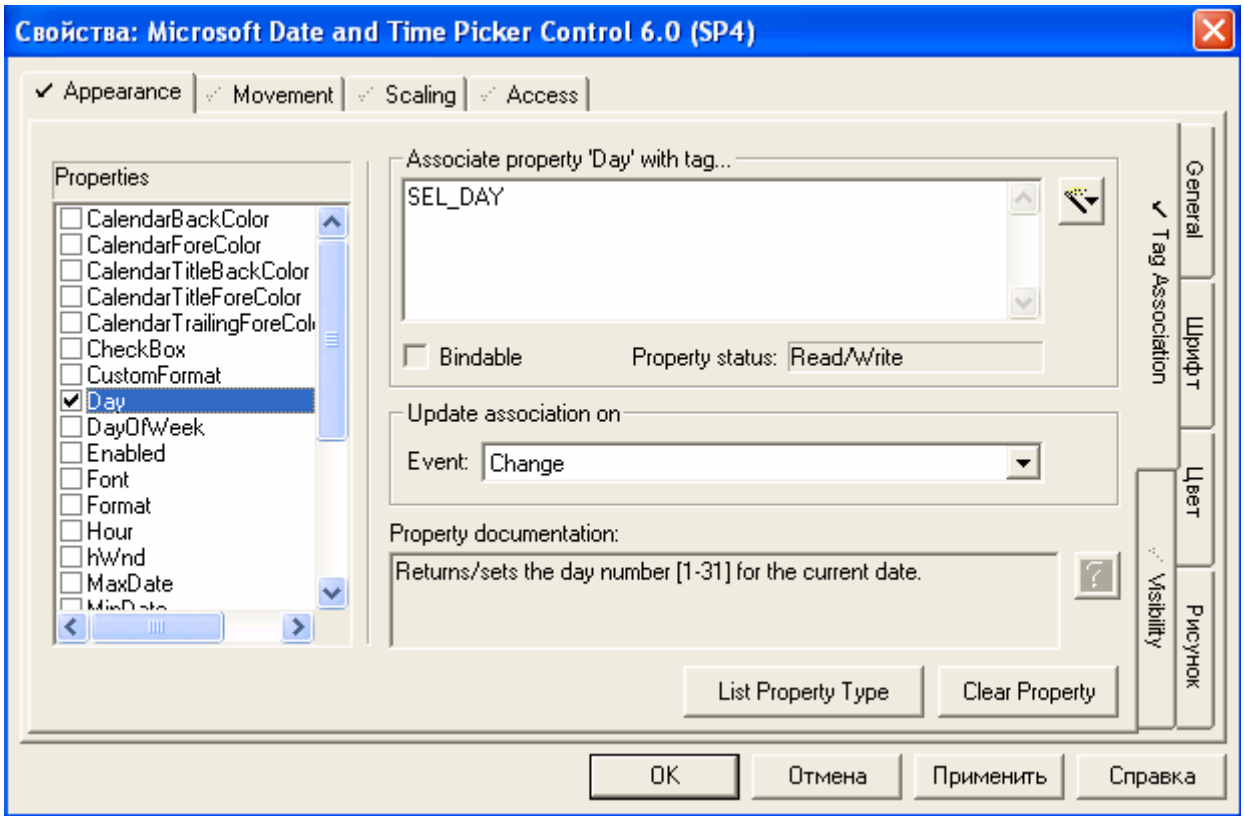

Рис.6.1. Зовнішній вигляд вікна нсатройки елмента ActiveX в SCADA Citect.

Зробимо деякі пояснення. На вкладці "TagAssociation" настроюється зв'язок властивостей ActiveX з значеннями змінних. Запис значень властивостей в змінні проводиться в момент виникнення події, яка задається в полі "Update association on".

Для відображення внутрішніх змінних на мнемосхемі виведемо 3 елементи Number, і в закладці "Display Value" прив'яжемо їх значення до змінних SEL_DAY, SEL_MONTH, SEL_YEAR

Результат роботи RunTime системи з елементом ActiveX "Microsoft Date AND Time Picker Control" показаний на рис 6.2.

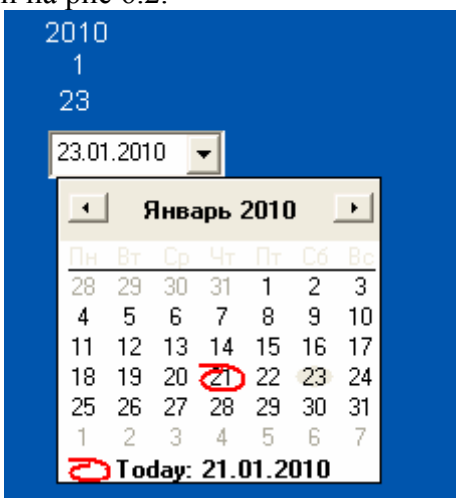

Рис.6.2.Фрагмент мнемосхеми RunTime системи SCADA Citect з демонстрацією роботи ActiveX елемента.

Література: [1], розділ 12.

7. ТЕХНОЛОГІЯ OPC

ПЛАН.

7.1. Передумови виникнення
7.2. Стандарти OPC
7.3. Функціонування OPC з точки зори інтегратора
7.4. OPC модель взаємодії
7.5. Механізми читання та запису даних процесу
7.6. Ідентифікатори ItemID
7.7. Робота OPC-Клієнта з віддаленими OPC Серверами
7.8. Область застосування технології OPC

7.1. Передумови виникнення

Вибір SCADA/HMI по набору підтримуваних комунікацій. Один з критеріїв вибору SCADA програми – перелік підтримуваних комунікацій. Тобто SCADA з одного боку і технічний засіб (надалі контролер) з іншого, повинні підтримувати однаковий протокол (або протоколи) промислової мережі. Найчастіше вибирають ту мережу, яка вже інтегрована в контролер. В цьому випадку при виборі SCADA враховують наявність даних протоколів в переліку комунікаційних драйверів.

Проблеми сумісності SCADA/HMI з контролерами. При інтеграції продуктів одного виробника, наявність в SCADA-програмі драйверів зв'язку з необхідними контролерами є очевидною. Найскладнішим є випадок, коли необхідно інтегрувати засоби від декількох виробників, ряд з яких підтримують закриті протоколи. В цій ситуації дуже важко підібрати таку SCADA-програму, яка б підтримувала всі необхідні протоколи промислових мереж. Розглянемо, які можливі варіанти реалізації подібної системи.

1. Вибір іншої промислової мережі, яка б підтримувалась з боку SCADA та контролеру. Цей варіант не завжди можливо реалізувати.
2. Написання спеціального драйверу, якого не існує в SCADA, для забезпечення зв'язку з контролером. Цей варіант потребує залучення програміста досить високого рівня підготовки, наявності відкритого програмного інтерфейсу з боку SCADA-програми та відкритого протоколу обміну з контролером.
3. Заміна частини одних контролерів іншими, для яких є драйвери зв'язку. Цей варіант потребує значних капітальних затрат і може бути використаний як виключна міра.
4. Використання шлюзів для промислових мереж. Варіант також потребує значних капітальних затрат і не завжди реалізується.

Як бачимо, одні способи сильно обмежують реалізацію системи, а інші потребують значних капітальних затрат. Розглянемо приклад.

Приклад 7.1. OPC. Вибір SCADA програми.

Завдання. Система управління технологічним процесом включає три підсистеми, в кожній із яких використовуються різні за типом і виробником контролеру: TSX Premium (Schneider Electric), S7-300 (Siemens), Ломіконт Л-110 (Чебоксари) (рис.7.1.). Необхідно вибрати SCADA-програму серед трьох варіантів (Citect 7, Zenon 6, Trace Mode 6), яка б забезпечувала зв'язок з усіма підсистемами, враховуючи використання вбудованих комунікацій в ПЛК, та без придбання додаткових конвертуючих програмних засобів.

Рішення. При виборі SCADA-програми необхідно перевірити перелік комунікаційних драйверів зв'язку для підтримки вказаних контролерів. Необхідні три типи драйверів: UNI-Telway (для TSX Premium), S7-MPI (для S7-300) та драйвер Сеть Ломиконт (для Л-110).

Протоколи для підключення засобів до SCADA-програми реалізовані у вигляді драйверів зв'язку. Зрозуміло, що інтерфейс драйверів з боку SCADA для кожної реалізації різний, тому драйвери для однієї SCADA-програми не підійдуть для іншої. Тому необхідно вибрати ту, яка має всі три драйвери. Citect 7 та Zenon 6 підтримують всі комунікації, крім Ломіконту. TM6 – всі комунікації, однак S7-MPI потребує наявність драйверів SIMATIC NET і має певні обмеження у використанні. Таким чином жодна з перерахованих SCADA-програм не задовольняє вказаним в умові задачі критеріям вибору. Необхідне використання додаткових програм-конверторів, розглянути можливість вибору іншої SCADA, мережі або написання драйверу зв'язку.

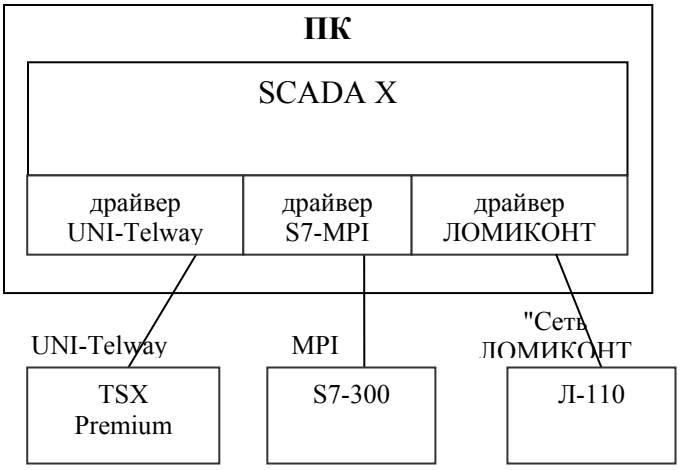

Рис.7.1. Підключення декількох контролерів до SCADA-програми

Наведений вище приклад – це один із багатьох, який обумовлює вирішення проблеми вибору SCADA. Для конкурентноздатності найбільш відомі компанії, які займаються розробкою програмного забезпечення верхнього рівня для систем автоматизованого управління, забезпечили їх великою кількістю драйверів зв'язку для пристроїв найбільш відомих брендів. Однак поставлена задача не завжди може бути вирішена шляхом вибору певної SCADA. Адже пристрій управління може бути розроблений невеликою компанією, яка не є всесвітньо відомим брендом і для якої не розроблявся відповідний драйвер.

OPC – як універсальний драйвер зв'язку. Для подолання цієї проблеми, групою великих компаній було вирішено створити стандартний інтерфейс доступу до даних "драйвера" зі сторони програмного забезпечення верхнього рівня. Таким чином, будь який драйвер зі стандартним інтерфейсом може бути використаний будь-якою SCADA-програмою, яка цей інтерфейс підтримує. Технологія отримала назву OPC.

7.2. Стандарти OPC

Походження. Перша версія стандарту OPC (OPC DA 1.0) розроблена групою компаній, які в 1996 році організували некомерційну організацію ***OPC Foundation*** (www.opcfoundation.org), що займається розвитком та просуненням даної технології на ринку.

Доступні специфікації. На сьогоднішній день стандарт існує в наборі специфікацій, основні з яких:

1) ***OPC DA*** (Data Access) – специфікація доступу до даних реального часу;
2) ***OPC AE*** (Alarms & Events) – для реалізації задач попереджувально-аварійних сигналізацій;

3) ***OPC HDA*** (Historical Data Access) – для реалізації задач ведення архіву та доступу до архівних даних;
4) ***OPC DX*** (Data eXchange) – для безпосереднього обміну між OPC-серверами;
5) ***OPC XML*** – для обміну даними через інтермережі за допомогою структур XML на базі WEB-сервісів та SOAP;
6) ***OPC Batch*** – для реалізації управління рецептурними задачами.
7) ***OPC UA (United Architecture)*** – самий новий платформо-незалежний стандарт, який об'єднує функції всіх наведених вище специфікацій, але функціонує не на базі COM а WEB-сервісах .

Серед наведених вище стандартів найбільшу популярність на сьогоднішній день має OPC DA 2.0, всі інші зустрічаються набагато рідше. Стандарт OPC UA є найновішим стандартом, однак поки що не знайшов широкого вжитку. Планується, що OPC UA в найближчому майбутньому витіснить OPC DA 2.0 та всі супутні йому стандарти..

Визначення OPC. Першопочатково технологія ***OPC*** розшифровувалась як *OLE for Process Control* і являлась промисловим стандартом взаємодії між програмними засобами в області промислової автоматизації, який базується на об'єктній моделі COM/DCOM (OLE). Однак при появі нових специфікацій, зокрема XML та UA, які не базуються на COM, слово "OLE" в абревіатурі перестало відповідати дійсному функціональному змісту технології. На сьогоднішній день немає офіційної розшифровки терміну OPC. Тому будемо користуватися таким поняттям OPC - це відкрита технологія зв'язку (open connectivity) в області промислової автоматизації та управління виробництвом.

Загальна модель. В загальному випадку, технологія OPC забезпечує одній програмі (OPC-Клієнту) доступ до даних процесу іншої програми (OPC-Серверу) через стандартний набір інтерфейсів. Розглянемо набір інтерфейсів, які базуються на COM-технології, через призму їх використання в системах АСУТП (рис.13.2). Це інтерфейси, які описуються специфікаціями OPC DA, OPC A&E та OPC HDA.

OPC DA (Data Access). COM-інтерфейси OPCDA стандартизують доступ OPCDA-Клієнту до даних процесу OPCDA-Серверу. В свою чергу програма OPC-Сервер, як правило здійснює обмін даними з контролерами або розподіленою периферією через специфічний, відмінний від OPC, інтерфейс. В цьому випадку, OPCDA-Сервер надає доступ OPCDA-Клієнту до даних процесу пристроїв, з якими він обмінюється тому служить в якості програми-шлюза.

Всі перераховані в задачі 13.1 SCADA-програми можуть бути OPCDA-Клієнтами, оскільки мають в наявності відповідний драйвер. Тому, придбавши OPCDA-Сервер для Ломіконт, можна зв'язати Zenon та Citect з Л-110, а придбавши OPCDA-Сервер з підтримкою S7-MPI, можна повноцінно поєднати Trace Mode з S7-300. Таким чином в даному випадку OPCDA-Сервер можна назвати універсальним (з боку SCADA) драйвером зв'язку. Слід зазначити, що ряд SCADA-програм повністю базуються на OPC (Genesis, Master SCADA).

OPC AE (Alarms & Events). OPCAE-Клієнт використовує OPCAE-Сервер для контролю за процесом, тобто за виникненням певних подій. Ці події налаштовуються в межах Серверу. OPCAE-Клієнт з'єднується з OPCAE-

Сервером і підписується під отримання повідомлень про виникнення цих подій. При підписці, OPCAE-Клієнт вказує додаткові критерії фільтрації повідомлень. Специфікацією підтримується можливість квітування повідомлень OPCAE-Клієнтом.

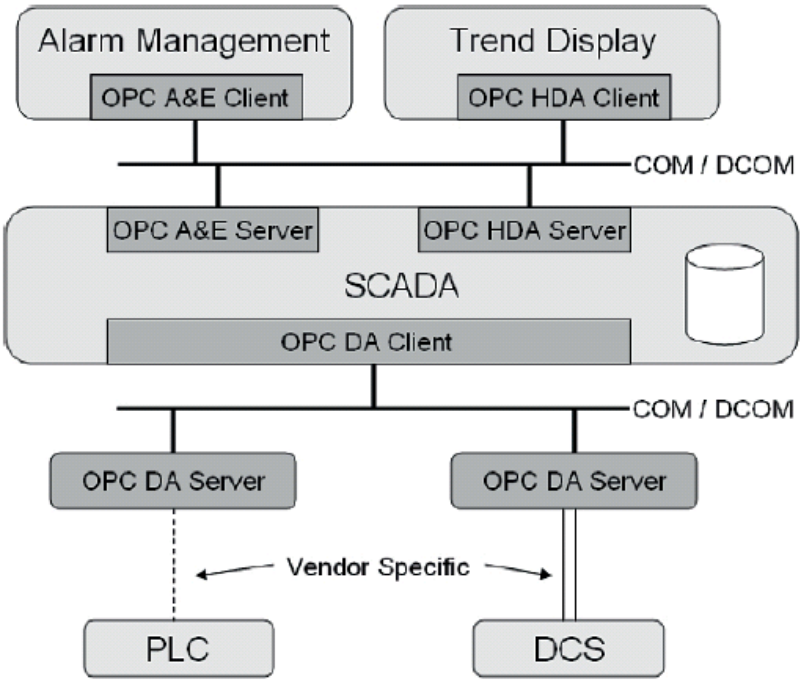

Рис.13.2. Типове використання технології OPC

OPC HDA (Historical Data Access). OPCHDA-Сервер надає доступ OPCHDA-Клієнту для отримання збережених даних. Даний сервер підтримує два COM-Об'єкти: OPCHDAServer – який надає доступ до архівних даних, та OPCHDABrowser, який надає доступ до переліку змінних, які зберігаються в архіві.

Читання архівних даних проводитись з використанням 3-х різних механізмів. Перший механізм передбачає зчитування архівних даних в певному часовому діапазоні для однієї або декількох змінних, які визначені іменами. Кількість зчитаних значень обмежується Клієнтом. Другий механізм передбачає зчитування архівних даних по часу їх відновлення (TimeStamp). Третій механізм дозволяє отримувати статистичну інформацію по збереженим даним. У OPC HDA інтерфейсі також передбачена можливість вставки, заміни або знищення архівних даних. У наступних підрозділах буде розглядатися тільки OPC DA технологія.

7.3. Функціонування OPC з точки зори інтегратора

Найбільш часто OPC-технологія використовується в якості універсального інтерфейсу до драйверів контролерів та периферійних пристроїв. Тобто разом з контролером може поставлятися спеціальна програма - OPC-Сервер, який надає доступ до змінних цього типу контролеру. Тобто OPC-Сервер з одного боку має драйвери для зв'язку з контролерами по конкретним протоколам промислових мереж, а з іншого - надає універсальний OPC-інтерфейс для зв'язку з сервером SCADA-програми. В такій системі SCADA буде OPC-Клієнтом.

На рис.7.3 показана спрощена схема функціонування роботи OPC-технології в контексті описаної системи. База даних реального часу SCADA-програми (з умовною назвою "SamplSCADA"), збирає дані з чотирьох джерел: ПЛК1, ПЛК2, ПЛК3 та ПЛК4. Для перших двох контролерів для збору даних використовуються драйвери зв'язку для цих ПЛК, вірніше для протоколів промислових мереж, по яким вони з'єднуються. Дані зчитуються (або записуються) з ПЛК в БДРЧ. Зв'язок з ПЛК3 та ПЛК4 виконується через OPC-сервери з умовними назвами відповідно "Sampl.OPC" та "Exmpl.OPC" з використанням драйвера OPC-Клієнт. Тобто OPC-Сервери через вбудовані драйвери зчитують дані з ПЛК та зберігають їх в своїй базі даних реального часу. SCADA-програма в свою чергу зчитує дані з OPC-серверів. Запис даних відбувається аналогічно.

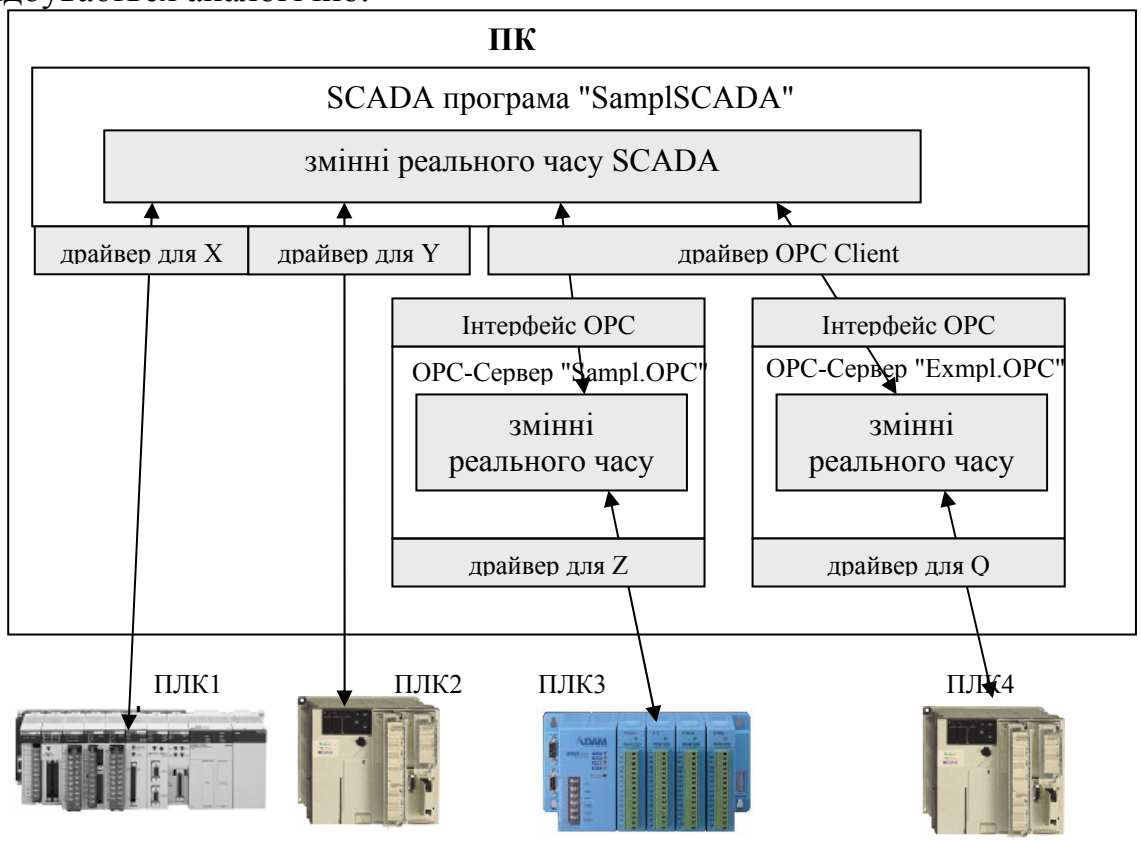

Рис.7.3. Функціонування OPC з точки зору інтегратора

Для реалізації такого зв'язку користувач повинен:

1. Налаштувати OPC-Сервер за допомогою спеціалізованої програми-конфігуратора, що поставляється разом з ним: створити всі необхідні змінні сервера, тобто дати їм ім'я (ItemID) та вказати джерела даних в ПЛК, на які вони посилаються.
2. В SCADA-програмі вказати:
   - назву OPC-Сервера, з яким необхідно зв'язатися (ProgID). У нашому прикладі це будуть два сервери "Sampl.OPC" та "Exmpl.OPC". Інколи SCADA надає можливість вибору ProgID зі списку зареєстрованих OPC-Серверів.
   - для вибраної змінної в якості джерела даних вказати ім'я на OPC-Сервері, тобто ItemID, що був створений на 1-му кроці. Як правило

ItemID вибирається зі списку, який надає Browser на стороні OPC-Клієнта.

Приклад 7.2. OPC. Конфігурація OPC-Серверів та OPC-Клієнта (SCADA).

Завдання. Налаштувати OPC-сервери (Schneider-Aut.OFS, VIPA.OPC-Server) та SCADA (Citect) для зчитування наступних змінних (рис.7.4) :

– MW100, що відповідає за температуру з PLC1(VIPA200) по протоколу MPI;
– %MW100, що відповідає за тиск з PLC2 (TSX Micro) по протоколу Modbus RTU;

Для зв'язку з контролерами використовуються OPC-сервери:

- OFS-Servrer від Шнейдер Електрик (ProgID=Schneider-Aut.OFS), що підтримує ряд протоколів, зокрема Modbus RTU;
- VIPA OPC-Server від фірми VIPA (ProgID=VIPA.OPCServer), що підтримує протокол MPI.

Рішення..

1. **Конфігурування OFS.** Відповідно до рис.7.4 на даному OPC сервері необхідно створити змінну з назвою "Pressure", джерелом даних для якої буде змінна %MW100 на PLC2. Для конфігурування OFS-сервера використовується утиліта OFS Configuration Tool.

Дані визначаються наступним чином (рис.7.5):

- створюється псевдонім, який буде вказувати на адресу конкретного пристрою, з яким може обмінюватись OPC-сервер. У нашому випадку псевдонім пристрою має ім'я PLC2);
- для створеного псевдоніму вказується драйвер зв'язку, адреса пристрою та додаткові параметри, що уточнюють місцезнаходження його в мережі; в нашому випадку в результаті конфігурування створиться адреса: MODBUS01:1/T;
- для створеного псевдоніму пристрою вказати файл, в якому будуть знаходитись символьні імена та відповідні їм змінні контролера. У нашому випадку вибраний файл PLC2.CSV, в якому сформований запис:

%MW100 PRESSURE

Відповідно до правил іменування змінних в OFS, ідентифікатор потрібної змінної буде формуватися наступним чином:

ItemID = Ім'я_псевдоніму_пристрою!Ім'я_змінної

В нашому випадку ідентифікатор змінної буде −PLC2!PRESSURE.

2. Конфігурування VIPA-OPC. Конфігурування для VIPA OPC-Server виконується з використанням утиліти OPC Editor. Аналогічно OFS, на сервері створюються пристрої, в межах яких визначаються змінні, однак їх порядок і форма дещо відрізняються (рис.7.6).

Спочатку вибирається драйвер мережі (в нашому випадку MPI). В межах мережі створюється PLC (в нашому випадку PLC1), а в межах пристрою вказується назва змінної (Tag) та її адреса в ПЛК (TEMPERATURE - MW100) Відповідно до правил іменування змінних в VIPA OPC ідентифікатор змінної буде формуватися наступним чином:

ItemID = Ім'я_PLC/Ім'я_змінної

В нашому випадку ідентифікатор змінної буде: PLC1/TEMPERATURE

3. Створення проекту для Citect. Всі змінні в Citect (VariableTag) створюються в межах віртуальних пристроїв (IODevices). Тому необхідно створити два IODEvices, які будуть відповідати за OPC Сервери. Для створення IODevices скористаємось Express Communication Wizard (рис.7.7). В обох випадках вибирається драйвер OPC DA Client. В полях Address вказується ім'я OPC-Серверів, тобто їх ProgID.

Далі необхідно створити змінні (рис.7.8) з прив'язкою до створених IODevices, та в полі адреси вказати відповідні ItemID.

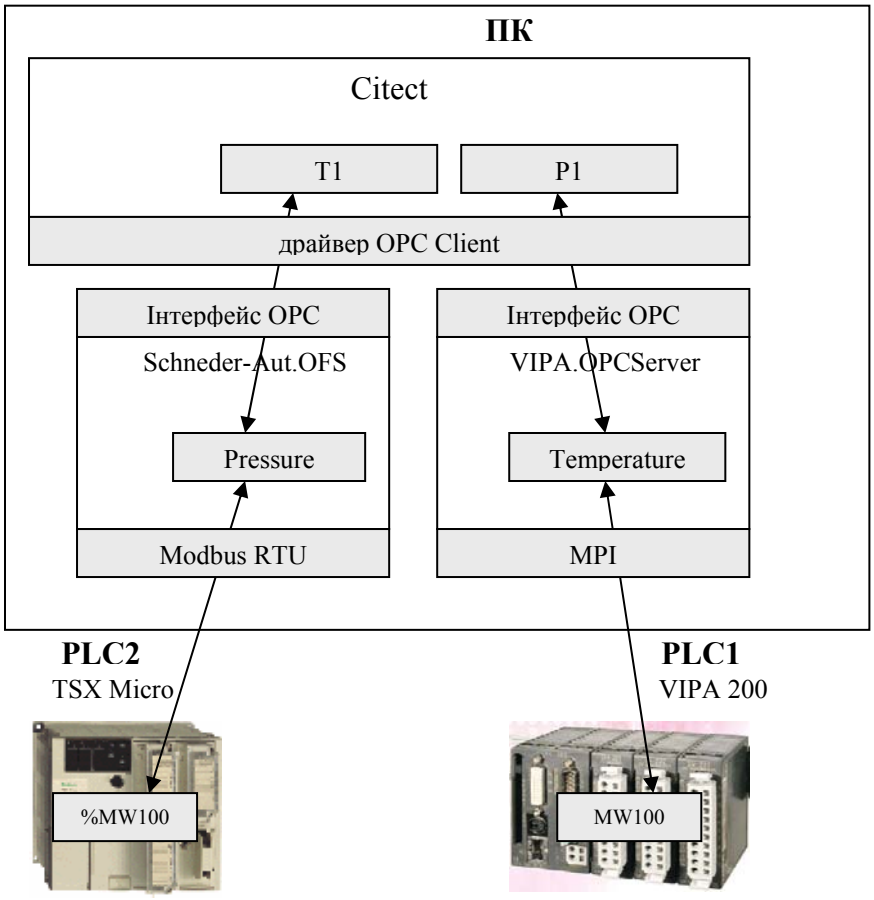

Рис.7.4. Схема обміну даним SCADA Citect з використанням OPC.

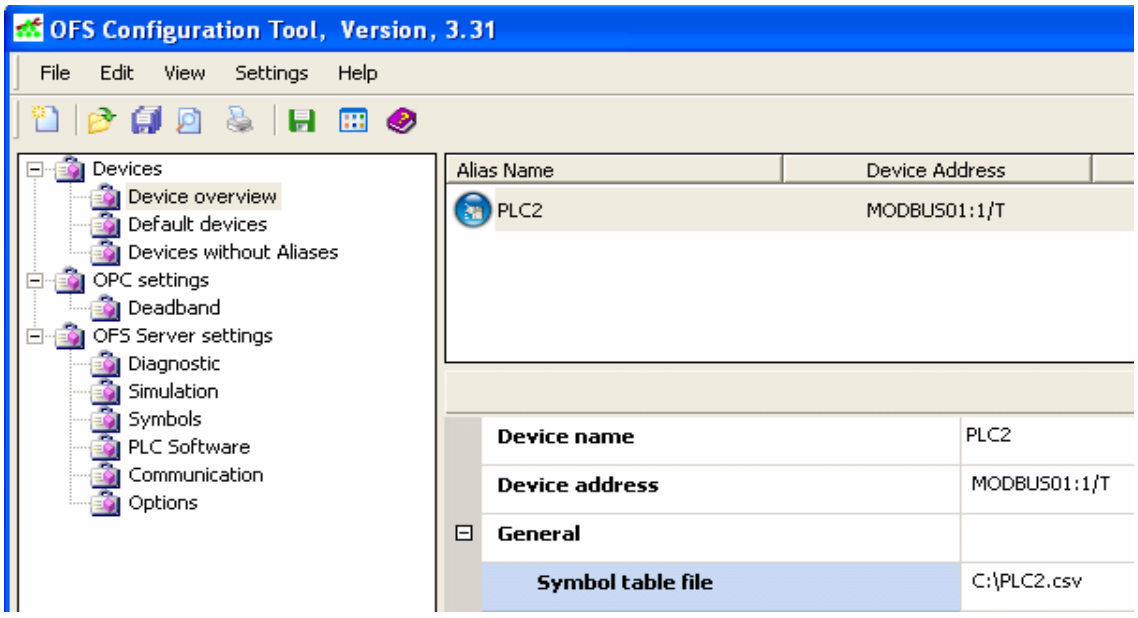

Рис.7.5. Конфігурування OFS

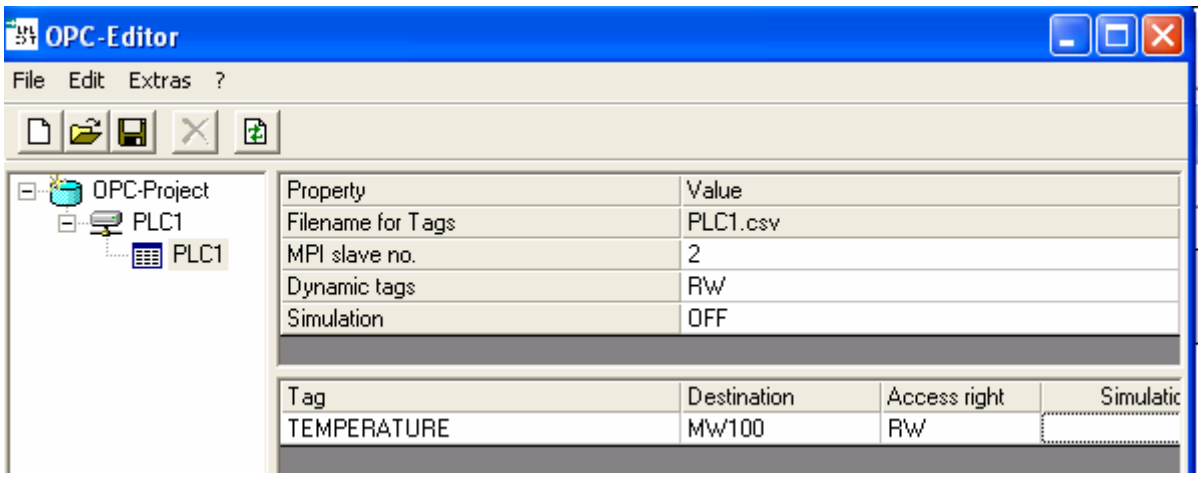

Рис.7.6. Конфігурування VIPA-OPC.

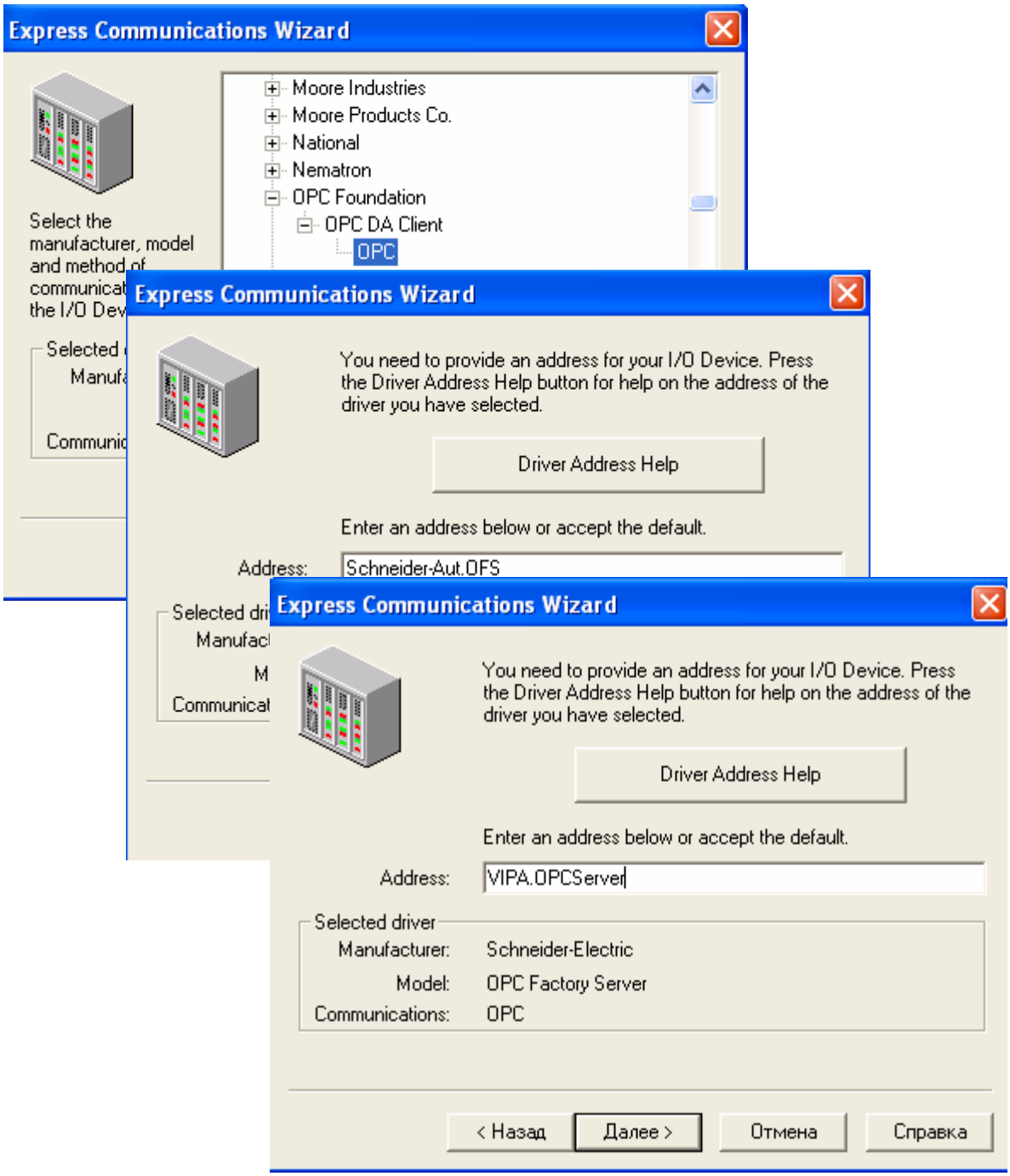

Рис.7.7. Створення в Citect IODevices, прив'язаних до OPC-серверів .

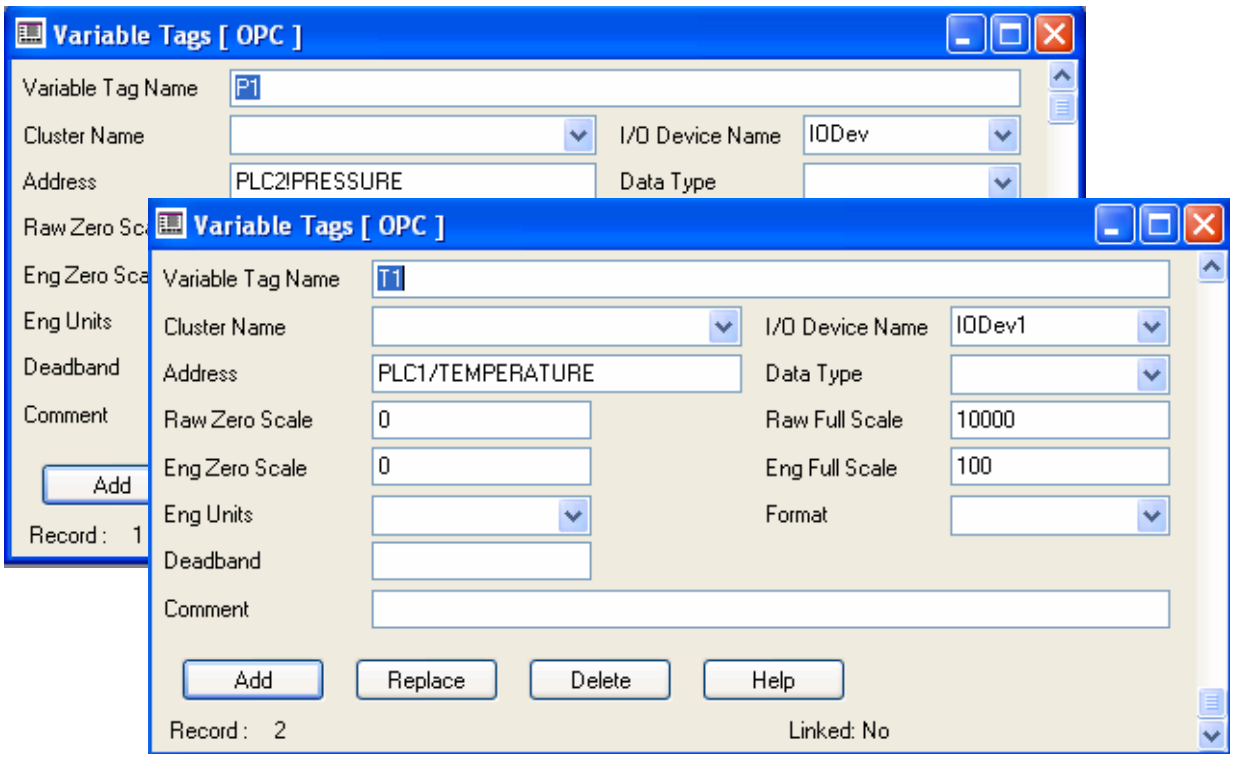

Рис.7.8. Створення в Citect Variable Tags .

7.4. OPC модель взаємодії.

Клієнт-Серверна модель. OPC DA технологія базується на Клієнт-Серверній архітектурі. OPC-Клієнт користується послугами OPC-Сервера, використовуючи COM-інтерфейси його об'єктів. У наведеному на рис.7.9 прикладі, *OPC-Клієнтом* є SCADA-програма, задачею якої є відображення чотирьох змінних (%MW100-%MW103) які знаходяться на ПЛК. OPC-Сервер отримує необхідні дані через драйвери зв'язку і зберігає їх у своїй базі даних реального часу. Для того щоб доступитися до даних OPC-Сервера, OPC-Клієнт створює для себе OPC-Group (Group1, Group2), в яких створює OPC Item (Item1, Item2), що посилаються на ці дані.

***OPC-Клієнт* (*OPC Client*)** – прикладна програма, яка вміє користуватися об'єктами OPC-Сервера за допомогою OPC-інтерфейсів (підмножина COM-інтерфейсів).

***OPC-Сервер* (*OPC Server*)** – прикладна програма, яка надає доступ до визначених в специфікації OPC COM-об'єктів за допомогою OPC-інтерфейсів.

З одним OPC-Сервером можуть з'єднатися декілька OPC Клієнтів. З іншого боку, одна і та сама програма OPC- Клієнт, може одночасно користуватися послугами декількох OPC-Серверів. Тобто OPC технологія є мультиклієнтною і мультисерверною.

Ідентифікація OPC-Серверу. Так як OPC-Сервер – це COM-Сервер, він реєструється на комп'ютері унікальним числовим ідентифікатором (GUID) та має унікальний строковий програмний ідентифікатор (***ProgID***). Тобто, для того щоб для OPC-Клієнта визначити з яким OPC-Сервером на тому самому ПК йому необхідно з'єднатися, достатньо вказати його ProgID.

Об'єкти OPC-Item та ідентифікація даних. Об'єкт ***OPC-Item*** надає доступ до джерела даних (надалі ***тег***) в межах OPC-Сервера, яке ідентифікується

унікальним в межах сервера ідентифікатором ***ItemID***. Тому при створенні OPC-Item'а, вказується ItemID необхідного тега. Правила ідентифікації даних залежать від реалізації OPC-Сервера, а механізм визначення їх джерел (наприклад адреса пристрою та змінної в ПЛК) як правило реалізується в конфігураторі цього сервера. У прикладі 7.2 ми вже розглянули два варіанта формування ItemID, нижче більш детально розгялнуті способи ідентифікації тегів. Тут тільки зазначимо, що весь список ItemID може мати деревовидну ієрархічну структуру, що дозволяє зручніше використовувати цей механізм в проектах з великою кількістю даних. Для навігації по списку/дереву ідентифікаторів OPC-Сервер, як правило, має об'єкт ***OPC Browser***.

OPC-Item належить Клієнту, який його створив і тому його не можуть використовувати декілька Клієнтів. Тим не менше є можливість посилатися на одні і ті ж дані. На рис. 7.9 два Клієнта одночасно використовують дані з %MW100 та %MW102, однак створюють для цього різні OPC-Item. Слід відмітити, що джерелом даних не обов'язково є змінна на зовнішньому пристрої, це можуть бути внутрішні дані самого Серверу.

З кожним OPC-Item'ом асоціюється плинне значення (***Value***), відмітка часу (***Time Stamp***) та якість (***Quality***).

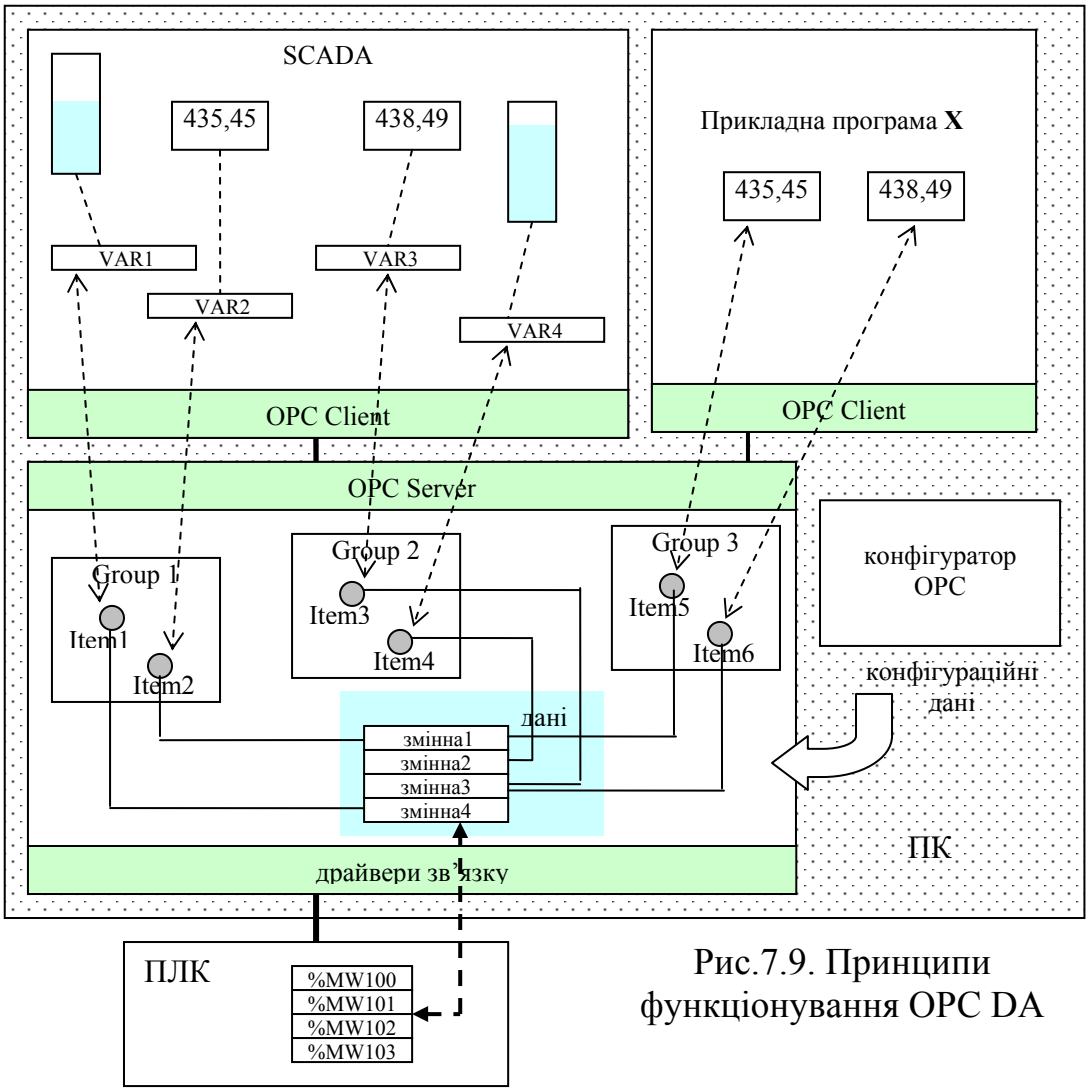

Рис.7.9. Принципи функціонування OPC DA

Групування OPC-Item в OPC-Group. ***OPC-Group*** – об'єкт OPC-Сервера, який призначений для виконання групових операцій над OPC-Item'ами. Так як OPC-Item не може існувати без цього об'єкту, спочатку OPC-Клієнт створює OPC-Group, а потім в його межах створює OPC-Item'и. В інтерфейсі OPC DA 2.0 кожний OPC-Group, як і все його наповнення, належить окремому Клієнту. Механізм групування дозволяє розділяти дані за принципом читання/запису, періодичністю операцій та активувати/деактивувати відновлення змінних.

7.5. Механізми читання та запису даних процесу

Загальні підходи. Технологія OPC надає двохсторонній доступ до даних, тобто як для читання так і для запису. Механізми реалізації цих сервісів практично однакові за принципом, однак мають свої особливості у різних версіях специфікації OPC DA. Ми розглянемо їх в контексті 2-ї версії цієї специфікації, оскільки на сьогодні вона є найбільш популярною.

Читання зводиться до вирішення наступних питань:

- коли на OPC-Сервері повинні відновлюватися дані з пристроїв для кожного з OPC-Item'ів;
- яким чином про відновлення даних дізнається OPC-Клієнт і як він їх отримає.

Операції читання та запису проводиться одночасно для всіх Item'ів в межах OPC-Group.

Синхронне читання (Sync Read). Ініціація процесу відновлення змінних на OPC-Сервері може проводитись самим OPC-Клієнтом. Тобто при необхідності OPC-Клієнт робить запит на відновлення певної OPC-Group. В такому випадку Клієнт може заморозити виконання своєї програми (потоку), поки не дочекається результату читання від OPC-Сервера. Такий спосіб називається ***Синхронним Читанням*** (***Sync Read***). На рис.7.10 графічно зображений процес обміну між OPC-Клієнтом та OPC-Сервером. При необхідності Клієнт робить запит за допомогою виклику метода SyncRead для OPC-Group "myGroup" та чекає поки той не поверне відповідь.

Асинхронне читання (Async Read). **М**еханізм синхронного читання гальмує роботу програми (потоку) Клієнта, тому доречний для читання невеликих об'ємів даних. Альтернативою йому може бути використання ***Асинхронного Читання*** (***Async Read***), при якому OPC-Клієнт теж ініціює обмін, однак не чекає результату обробки. Замість цього, при закінченні процесу читання OPC-Сервер викликає функцію зворотного виклику OPC-Клієнта (обробник події ***AsyncReadComplete***), в яку передає результат читання. Для реалізації цього механізму необхідно, щоб в об'єкті OPC-Group був активований механізм ***Підписки*** (***Subscript***).

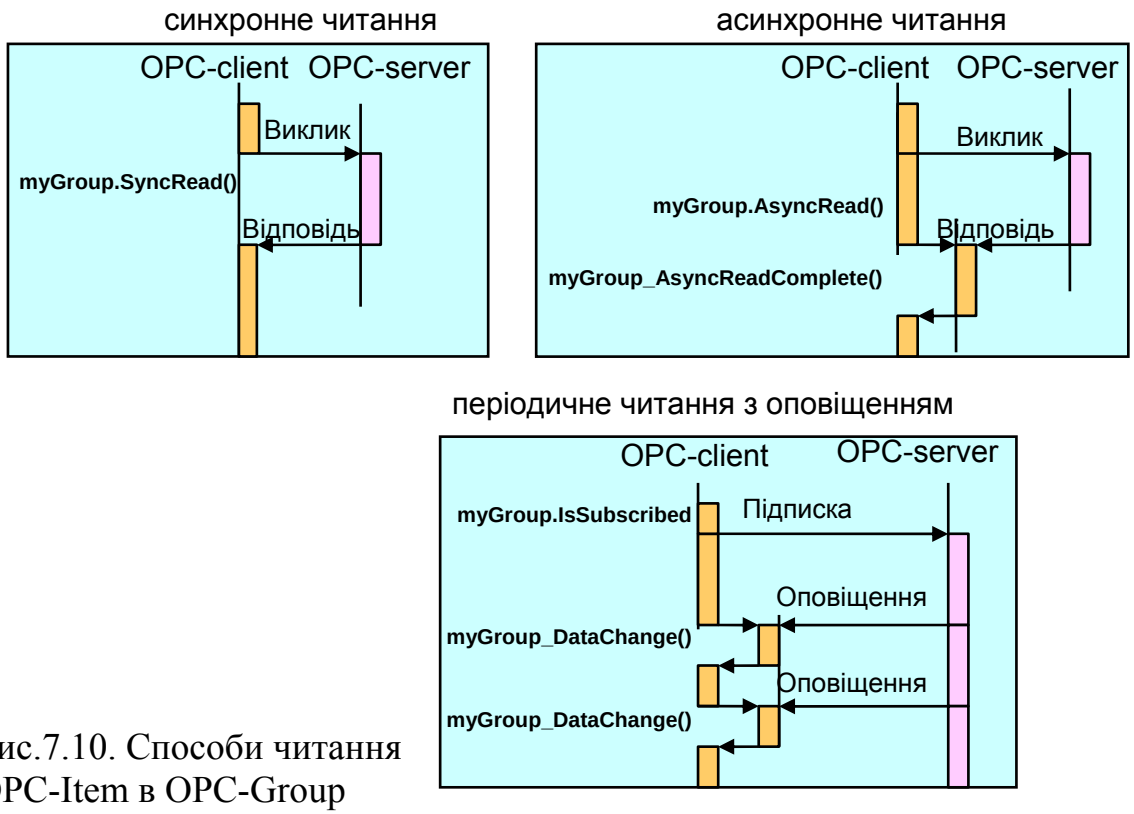

Рис.7.10. Способи читання OPC-Item в OPC-Group

Періодичне Читання з Оповіщенням (Periodical Read with Notify). При необхідності відновлення даних, обидва наведених вище способи потребують від OPC-Клієнта кожний раз проводити запит до OPC-Сервера. Однак як правило дані необхідно читати періодично через певні інтервали часу. Для цього в специфікаціях OPC DA є механізм ***Періодичного Читання з Оповіщенням*** (***Periodical Read with Notify***). При створенні OPC-Group, Клієнт замовляє частоту відновлення Item'ів в межах цієї групи. Через вказані проміжки часу OPC-Сервер буде відновлювати ці дані, а результат буде зберігати в ***Кеші*** (***Cache***). Якщо дані (Value або Quality) хоча б для одного OPC-Item'а в OPC-Group змінилися, буде викликана зворотна функція ***Оповіщення*** (***Notify***), тобто обробник події ***DataChange*****,** в параметрах виклику якого будуть передані нові значення. Для ефективного використання цього механізму можна скористатися зоною нечутливості (***Deadband***). Необхідно зазначити, що в об'єкті OPC-Group повинен бути активований механізм Підписки та прапорець ***Активності*** (***ACTIVE FLAG***). Крім того, періодично відновлюватись будуть тільки Активні OPC-Item.

Синхронний та асинхронний запис. Операції запису можуть проводитись двома способами: ***Синхронний Запис*** (***Sync Write***) та ***Асинхронний Запис*** (***Async Write***). Функціонування повністю аналогічне як і в операціях читання (рис.7.11).

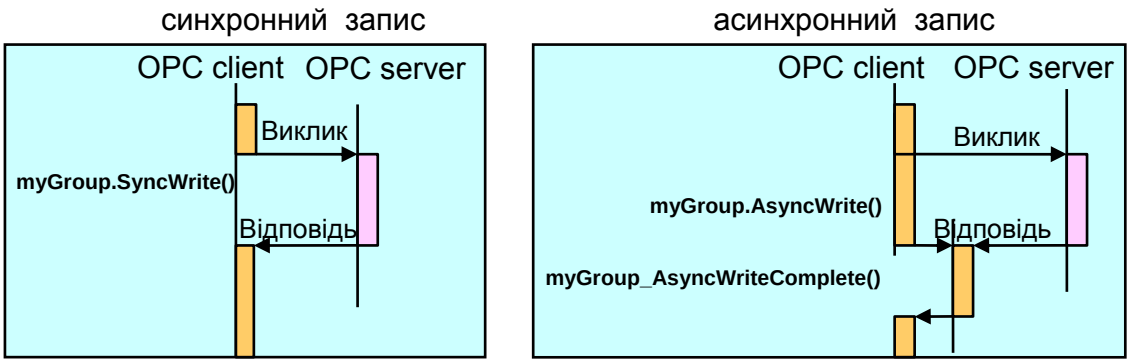

Рис.7.11. Способи запису OPC Group

7.6. Ідентифікатори ItemID.

Способи ідентифікації даних. ItemID – це унікальний, в межах OPC-Сервера, символьний ідентифікатор, який однозначно ідентифікує дані (теги) на цьому Сервері. Тобто він не повинен вказувати, з якого пристрою беруться дані, а тільки де вони розміщуються на Сервері.

Частіше всього ItemID створюються за допомогою конфігуратора OPC Сервера. Саме там і конфігурується розміщення джерела даних. У прикладі 7.2. таким способом ідентифікуються теги на VIPA.OPCServer - "Temperature".

В деяких реалізаціях ItemID може створюватись автоматично. В цьому випадку розміщення джерела даних, зона нечутливості, мінімум та максимум і інше вказується в самому символьному рядку ідентифікатора. Наприклад, символьний рядок ItemID "MODBUS01:5!%MW100" в OFS Server (OPC Server від Schneider Electric) означає, що джерело даних розміщується на шині Modbus у Веденого з адресою 5, в змінній %MW100.

Доступ до списку ItemID (Об'єкт OPCBrowser). Для зручності ідентифікації джерела даних, OPC-Сервер опціонально може підтримувати об'єкт-навігатор OPCBrowser. В версіях OPC DA 1.0/2.0 та OPC DA 3.0 реалізація механізмів навігації відрізняється, однак і в першому і в другому випадку весь перелік ItemID може формувати плаский список (***flat***) або ієрархічне дерево (***hierarchical***). Ієрархічна структура формується у вигляді дерева, приклад якого наведений на рис.7.12.

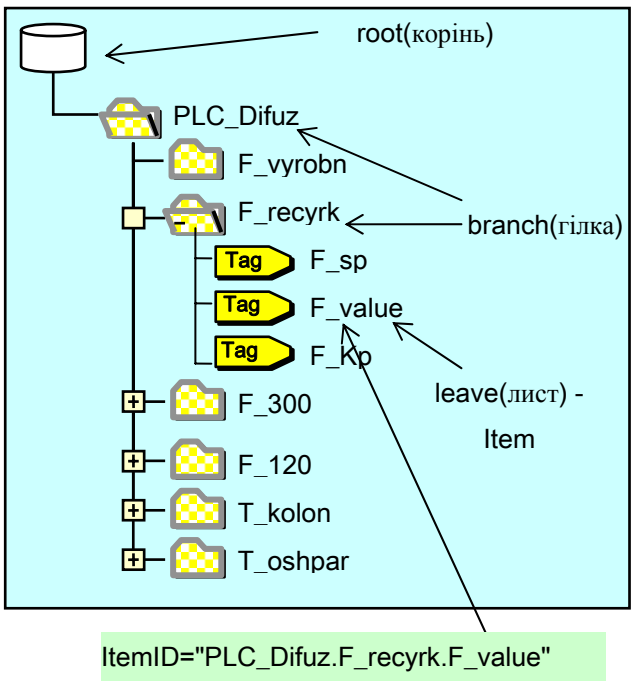

Рис.7.12. OPC Browser.

OPCBrowser, як правило, потрібен тільки для Клієнтів, які необхідно конфігурувати, наприклад SCADA. У випадку його відсутності, користувачу треба добре знати

правило формування символьного рядку ItemID для конкретного OPC-Сервера, адже це не обумовлено в стандарті. Так наприклад, в рядку ItemID з рис.7.12 замість відокремлюючих крапок можуть використовуватися інші символи, наприклад "!".

7.7. Робота OPC-Клієнта з віддаленими OPC Серверами

OPC-Клієнт та OPC-Сервер на одному і тому самому ПК запускаються як окремі Процеси. Обмін даними між цими Процесами відбувається по правилам COM-технології. Інколи виникає необхідність у з'єднані OPC-Клієнта з віддаленим OPC-Сервером, який знаходиться у мережі на іншому ПК. Для такого з'єднання використовуються сервіси DCOM.

На рис.7.13 наведений приклад, у якому OPC-Клієнт (SCADA) на ПК1 з'єднується з локальним OPC-Сервером (OPCServer1) та двома віддаленими (OPCServer2 на ПК2 та OPCServer3 на ПК3). Для реалізації такого з'єднання для OPC-Клієнта окрім ProgID необхідно вказати розміщення ПК з OPC-Сервером, а також вірно налаштувати DCOM-конфігуратор. Таким чином необхідно виконати наступну послідовність:

1. Налаштувати DCOM Конфігуратор на вузлі Серверу та Клієнта (див. розділ 12).
2. Вказати ***Server Node*** (Ім'я вузла OPC Серверу) або його IP.
3. Вказати ProgID Сервера.

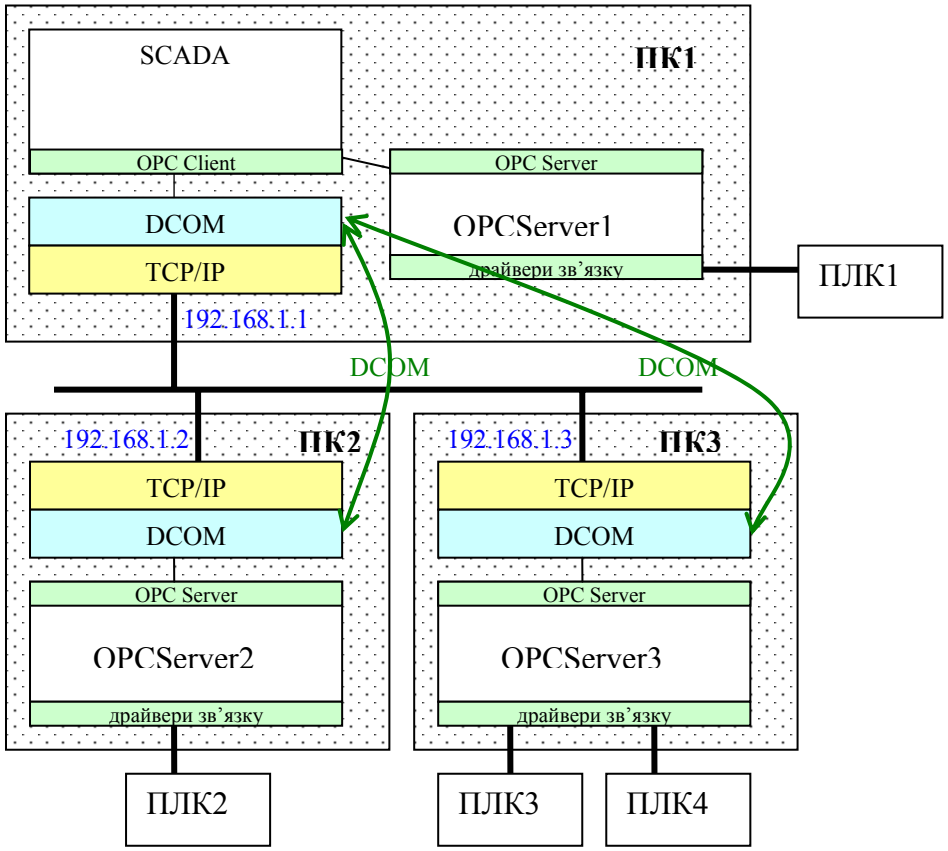

Рис.7.13. Приклад з'єднання трьох пристроїв через DCOM

Однак зв'язатися з віддаленим OPC за допомогою OPC DA можливо тільки у випадку коли вузли знаходяться в межах одного домену або робочої групи

Windows та не розмежовуються брандмауерами. Останні можуть не пропустити пакети COM (порти RPC як правило закриті для доступу), тому для з'єднання через Інтернет необхідно вдаватися до неабияких хитрощів. Щоб вирішити цю проблему OPC Foundation пропонує технології OPC XML та OPC UA.

7.8. Область застосування технології OPC

У прикладах, наведених вище, розглянута технологія OPC-DA в контексті вирішення проблеми доступу до даних ПЛК зі SCADA. Тобто OPC-Сервер розглядався у якості стандартного драйвера зв'язку. Однак область застосування OPC цим не обмежується.

На рис.7.14 показаний приклад використання інтерфейсів OPC в якості „мосту" між двома прикладними програмами на різних ПК. При горизонтальній інтеграції може знадобитися об'єднання в єдиний інформаційний простір

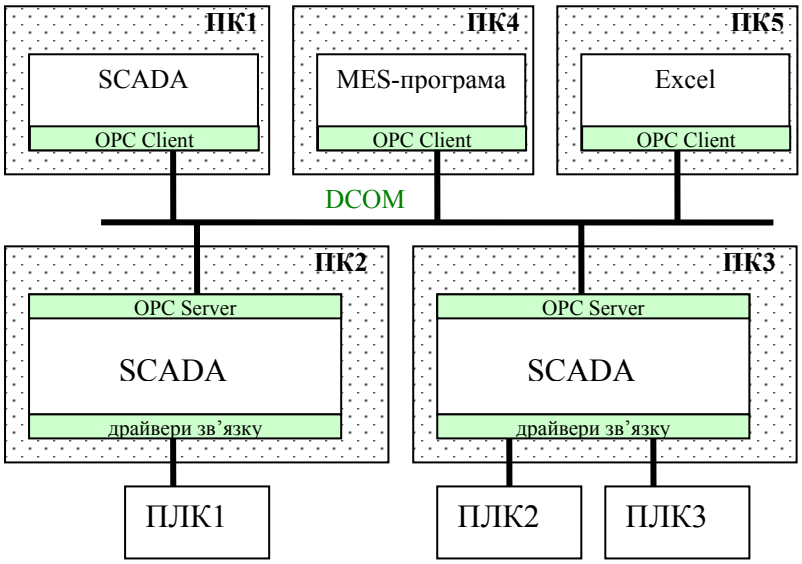

Рис.7.14. Приклад використання OPC в якості „мосту".

SCADA програм. Популярність OPC-технології призвела до появи в останніх не тільки клієнтської сторони OPC, а і серверної. Тобто SCADA може виступати як OPC-Клієнтом так і OPC-Сервером.

Існування в SCADA Серверного інтерфейсу дає можливість доступитись до її даних зі сторони прикладних програм рівня MES чи ERP. Офісні програми завдяки наявності VBA та ActiveX теж надають таку можливість.

Література: [1], розділ 13.

8. СТАНДАРТНІ ТЕХНОЛОГІЇ ДОСТУПУ ДО БАЗ ДАНИХ

ПЛАН.

8.1. Проблеми доступу до баз даних
8.2. Мова SQL
8.3. ODBC та DAO
8.4. OLE DB, ADO та ADO.NET

8.1. Проблеми доступу до баз даних

У інтегрованих автоматизованих системах управління бази даних використовуються, як правило, для ведення історії подій та збереження значень даних для трендів. У цьому контексті можна виділити два способи доступу до даних: для запису та для читання. В області ІАСУ як правило доступаються до архівних баз даних такі прикладні програми:

- SCADA-програми для запису плинних даних та читання історичних;
- програми MES-систем для читання історичних даних рівня АСУТП та запису агрегованих показників;
- програми ERP-систем для читання та запису організаційно-економічних даних;
- службові програми та СУБД.

Дані в архів зберігаються, як правило, в тому форматі, який пропонує прикладна програма (наприклад SCADA). Тобто тип архіву та формат записуваних даних диктується вимогами цього програмного засобу, або вибирається з ряду стандартних. Рідше є можливість в корегуванні розміщення, типу та формату бази даних. Тому можна умовно пропустити, що при записі даних, проблеми з сумісністю не виникають.

Інша ситуація виникає при доступу до даних для читання. Якщо програмні засоби які пишуть дані і які їх читають підтримують різні формати, то виникає проблема пов’язана з сумісністю форматів, а отже і з доступом до даних для читання. Очевидно, що одне з рішень, яке може бути запропоновано – це вибір програмних засобів, які підтримують єдиний формат доступу баз даних. Однак як правило вибір програмних засобів часто виступає в якості обмеження при побудові систем, особливо коли останні впроваджується поетапно. Крім того такий підхід не дає гнучкості при реалізації системи, адже обмеження диктуються вибором засобів одного виробника. Інше рішення – це використання додаткового спеціалізованого програмного забезпечення, яке перетворює дані з одного формату в інший, або написання додаткових бібліотек. Такий підхід дорогий в рішенні і потребує тривалої апробації.

Найбільш простим рішенням яке, як правило, доступне в сучасних програмних засобах ІАСУ, є використання стандартних технологій доступу до баз даних. Серед них можна виділити використання СУБД з підтримкою стандартної мови запитів SQL, а також стандартних інтерфейсів доступу до баз даних ODBC та OLEDB. Крім наведених технологій є й інші відомі і можливо більш прогресивні рішення, однак на сьогоднішній день в системах ІАСУ вони поки що не знайшли великої популярності.

8.2. Мова SQL

SQL (Structured Query Language – мова структурованих запитів) – це універсальна мова для створення, модифікації і управління даними в реляційних базах даних. Дана мова описана в стандарті ANSI 1992 року як ***SQL2*** **(*SQL-92*).** На сьогоднішній день практично всі відомі СУБД підтримують даний стандарт, з деяким розширеннями до нього для адаптації під свої формати даних та функціональні можливості серверів. Такі мови називають діалектами SQL. Так, наприклад діалект СУБД MS SQL Server має назву Transact-SQL.

SQL використовується для створення структури бази даних, маніпуляції з даними, тобто їх вибірки і модифікації, та для їх адміністрування. Будь яка операція по вибірці, модифікації, визначенню або адмініструванню виконується за допомогою ***оператору*** (***statement***) або ***команди*** (***command***) SQL.

Є дві можливості операцій по маніпуляції з даними – ***вибірка даних*** (data retrieval) і ***модифікація даних*** (data modification). Вибірка – це пошук необхідних даних, а модифікація означає добавлення, знищення або заміна даних. Операції по вибірці називають ***SQL запитами*** (***SQL queries***). Вони проводять пошук в базі даних, найбільш ефективно вибирають необхідну інформацію і відображають її. У всіх запитах SQL використовується ключове слово SELECT. Операції по модифікації виконуються відповідно з використанням ключових слів INSERT, DELETE та UPDATE.

На практиці частіше всього приходиться робити вибірку даних, тому коротко розглянемо оператор SELECT. Спрощена його конструкція має вигляд:

SELECT список_стовбчиків
FROM таблиця[-ці]
[WHERE умови],

де слова в квадратних дужках [] – не обов’язкові.

В списку стовпчиків (полів записів) вказуються ті поля таблиці, які повертаються після обробки запиту. Список таблиць визначає з яких таблиць необхідно проводити вибірку, а в умовах вказують умови для вибірки рядків.

Приклад 8.1. Бази даних. Формування SQL-запитів 1.

Завдання. Записати запит для вибірки даних з поля (колонки) Value таблиці Group_1_1 бази даних Difuzija_1, які були записані після 19 вересня 2006 року. Дата запису знаходиться в полі Group_1_1.TDate.

Рішення.

```
SELECT Value
FROM  Difuzija_1.Group_1_1
WHERE Group_1_1.TDate>'2006-09-19 00:00:00'
```

Приклад 8.2. Бази даних. Формування SQL-запитів 2.

Завдання. Записати запит для вибірки даних з усіх полів таблиці Group_1_1 бази даних Difuzija_1, які були записані після 19 вересня 2006 року. Дата запису знаходиться в полі Group_1_1.TDate.

```
SELECT *
FROM  Difuzija_1.Group_1_1
WHERE Group_1_1.TDate>'2006-09-19 00:00:00'
```

У списку SELECT вказуються ті поля (стовпчики), які необхідно повернути запитом. Є можливість зробити деякі операції над полями перед відображенням: +,-,*, /.

Умови в WHERE задаються операторами порівняння (=,<,>, >=, <=, != або <>), логічними (AND, OR, NOT), визначення діапазону (BETWEEN і NOT BETWEEN) та ін.

8.3. ODBC та DAO

ODBC (Open Database Connectivity) – один із стандартних інтерфейсів доступу до реляційних баз даних, які засновані на мові SQL. Архітектура ODBC показана на рис.8.1.

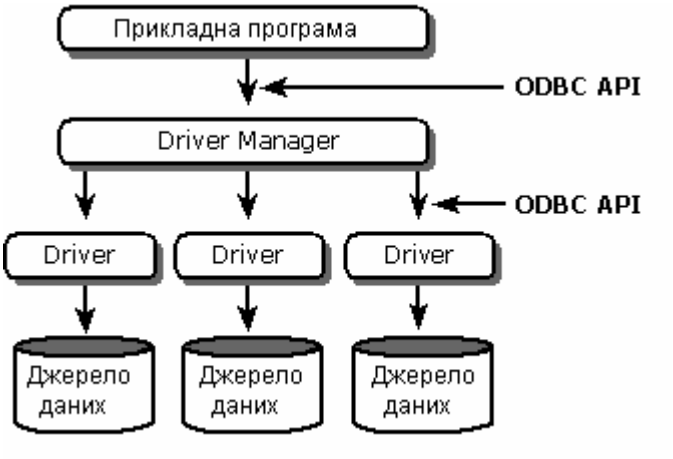

Рис.8.1. Архітектура ODBC

Зв'язок з різними джерелами даних проводиться за допомогою Драйверів ODBC, які оформлені у вигляді динамічних бібліотек (DLL) і підтримують єдиний інтерфейс ODBC API, що базується на виконанні SQL-запитів. Тобто, незалежно від типу джерела даних, прикладна програма генерує SQL-запити, які відправляються потрібному драйверу, що працює з даним джерелом. Драйвер забезпечує потрібне перетворення цих запитів в мову джерела, або сам виконує необхідні операції, що вказані в запиті.

Для користувача даної технології, необхідно зробити наступну послідовність операцій. В ***Адміністраторі ODBC*****,** який знаходиться в панелі управління адміністратору Windows, створюється ***DSN*** (Data Source Name) для необхідного джерела даних:

- вибирається тип DSN: System DSN (доступний всім користувачам), User DSN (доступний плинному користувачу), File DSN (настройки зберігаються в окремому файлі);
- вказується ім'я DSN;
- вибирається драйвер ODBC;
- налаштовується драйвер для конкретного джерела даних.

В клієнтській прикладній програмі в якості джерела даних вказується ім'я DSN. З'єднання прикладної програми з необхідним джерелом даних забезпечиться Диспетчером Драйверів (Driver Manager), який реалізований у вигляді бібліотеки ODBC.DLL (рис.8.1).

При написанні прикладних програм з використання ODBC більш зручний доступ до ODBC через RDO та DAO (рис.8.2). ***RDO*** (Remote Data Object) – являється об'єктним інтерфейсом доступу до ODBC джерел даних.

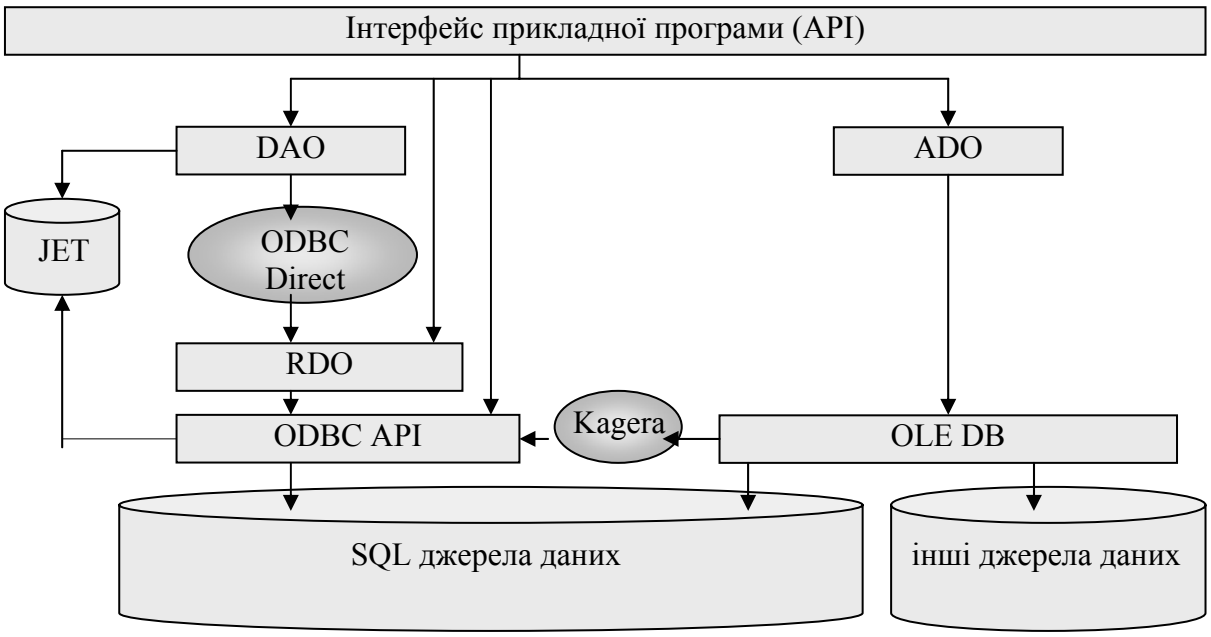

Рис.8.2. Принцип доступу до джерел даних через стандартні інтерфейси

DAO (Data Access Object) - являється об'єктним COM-інтерфейсом до процесора баз даних ***Jet*** (Joint Engine Technology database engine), а також до надбудовою над RDO-інтерфейсом.

Режим ODBCDirect перетворює всі об'єкти та методи DAO в еквівалент RDO.

Приклад 8.3. Бази даних. Доступ до змінних Citect через ODBC інтерфейс.

Завдання. Використовуючи технологію ODBC забезпечити періодичне відновлення даних в комірках листу Microsoft Excel, що посилаються на змінні з прикладу 7.2.

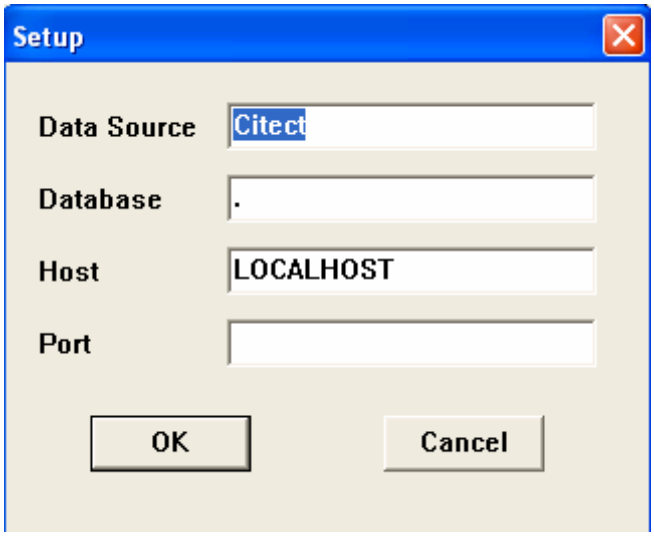

Рис.8.3. Зовнішній вигляд вікна настройки DSN для джерела даних CitectDriver

Рішення. SCADA Citect надає можливість доступу до бази даних реального часу через інтерфейс ODBC. При інсталяції SCADA на комп'ютер, в списку драйверів ODBC з'явиться драйвер CitectDriver.

Для створення DSN викликаємо Адміністратор джерел даних: "Пуск"-> "Настройка"->"Панель управления"-> "Администрирование"->"Источники данных ODBC". На вкладці "Системные DSN " створюємо DSN типу CitectDriver. Настроюємо DSN як на рис.8.3. Якщо необхідно з'єднатися з Citect на іншому ПК, замість LOCALHOST вказується ім'я даного ПК.

В Microsoft Excel створити зв'язок з зовнішніми даними, використовуючи майстра по створенню SQL запитів MicrosoftQuery:

1. "Данные"->"Импорт внешних данных"->"Создать запрос";
2. Вибрати джерело даних з іменем Citcet;
3. Вибрати стовбці NAME,VALUE;

4. На відповідній вкладці вибрати "Просмотр или изменение в Microsoft Query", а потім нажати кнопку SQL, для перегляду створеного майстром запиту (рис.8.4).
5. Після закриття MicrosoftQuery, вказати ячейки, де будуть розмішуватися данні.

Для автоматичного періодичного відновлення даних, в контекстному меню комірок з імпортованими даними вибираємо "Свойства диапазона данных"->"Обновлять каждые"->"1 мин".

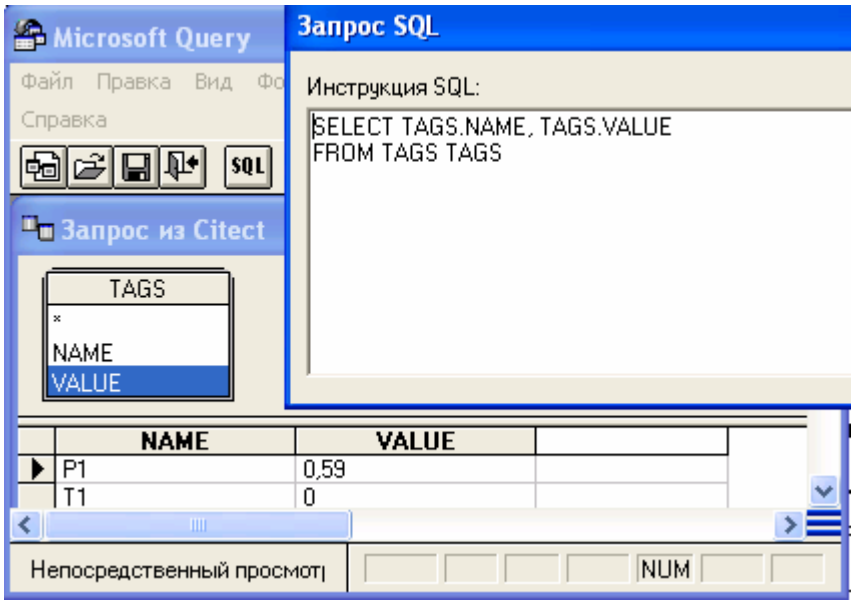

Рис.8.4. Зовнішній вигляд вікна настройки запитів в Microsoft Query.

8.4. OLE DB, ADO та ADO.NET

Альтернативою ODBC є ***OLE DB*** – це об'єктно-орієнтована технологія доступу до даних, яка базується на COM. Спеціалістами Microsoft запропонована стратегія універсального доступу до будь яких джерел даних ***UDA*** (Universal Data Access), де драйвери OLE DB займають нижній щабель ієрархії (рис.8.5). В цій архітектурі виділяються ***OLE DB Consumers*** (***OLE DB Споживачі***) – це будь-яка частина прикладної програми, яка *користується* OLE DB-інтерфейсами та ***OLE DB Providers*** (***OLE DB Провайдери***) – це частина прикладної програми, яка *надає* свої послуги через OLE DB-інтерфейси.

Розрізняють два види OLE DB Провайдерів: ***OLE DB Провайдер Даних*** (***OLE DB Data Provider***), ***OLE DB Провайдер Сервісів*** (***OLE DB Service Provider***). OLE DB Провайдер даних надає послуги доступу до власних баз даних у вигляді таблиць. Провайдер Сервісів не має власних даних, однак як Провайдер – він надає набір послуг (сервісів). Споживачі користуються цими сервісами, частина з яких використовує Провайдерів Даних для доступу до даних. Тобто Провайдер Сервісів по відношенню до Провайдерів Даних виступає як Споживач.

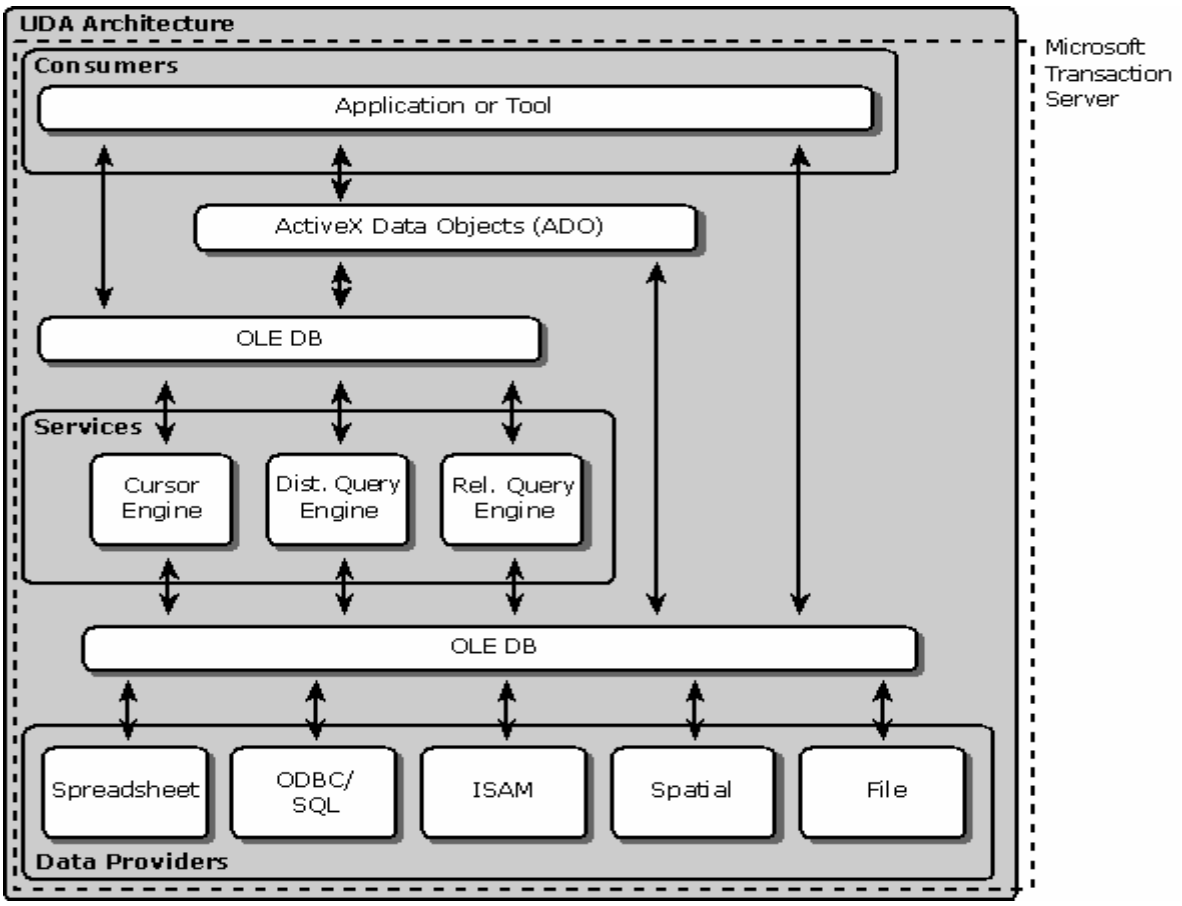

Рис.8.5. UDA Архітектура

На відміну від ODBC, який надає доступ тільки до реляційних даних, OLE DB може надати доступ до будь якого джерела даних, оскільки базується на об'єктній моделі, яка спроектована для представлення даних довільного формату. Для сумісності з розробленими ODBC драйверами, OLE DB має інтерфейс доступу до них через Провайдера ODBC. Інтерфейс OLE DB – складний в реалізації і призначений для розробки драйверів для нового типа джерела даних. Для кінцевого користувача необхідно мати зручний і легко зрозумілий прикладний інтерфейс (API), яким служить технологія ADO.

ADO (ActiveX Data Object) визначає модель програмування – послідовність дій, які необхідні для отримання доступу та модифікації джерела даних. Це об'єктна модель, тобто представляє собою набір об'єктів зі своїми методами, властивостями та подіями.

Модель базується на наступних ключових поняттях.

1. ***Connection*** (***Підключення***) - це джерело даних, з яким необхідно з'єднатися, може бути представлений у вигляді символьного ***Рядка Підключення (Connection String)*** або ***Uniform Resource Locator*** (***URL***). В рядку вказується вся послідовність (транзакція) підключення, частини (кроки) якої розділені спеціальними роздільниками. В Рядку Підключення спочатку вказується Провайдер OLE DB, а потім всі інші параметри, які відрізняються в залежності від Провайдера. Слід зазначити, що транзакція виконується повністю, або не виконується взагалі, тобто виконання підключення до частини вказаної в Рядку Підключення неможливе.

2. ***Command*** (***Команда***) – вказує операцію, яку необхідно зробити з джерелом даних визначених Підключенням (добавити, знищити, модифікувати, знайти дані по заданим параметрам). Розрізняють декілька типів команд:
   - по замовченню: тип команди визначається самим Провайдером;
   - текст (SQL) : команда записується у вигляді SQL-виразу;
   - таблиця: команда являє собою назву таблиці, всі колонки якої повертаються з запитом;
   - збережена процедура (stored procedure) – назва процедури, яка повинна бути викликана;
   - файл: вказується ім'я файлу, з яким необхідно з'єднатися.
1. ***Parameters*** (***Параметри***) – це параметри команди, які можуть змінюватися. Команда може виступати як функція, тобто частина команди незмінна, а інша частина виступає в якості параметрів (параметричні запити).
2. ***Recordset*** – це об'єкт, за допомогою якого можна доступитись до рядків таблиці, які повертає Команда-запит.
3. ***Field*** (***Поле***) – об'єкт, який надає доступ до полів об'єкту Recordset.
4. ***Error*** (***Помилка***) – об'єкт, який містить в собі інформацію про помилку. Ці об'єкти асоціюються з кожним Підключенням.
5. ***Property*** (***Властивість***) – кожний ADO-об'єкт має набір динамічних та статичних властивостей.
6. ***Record*** (***Запис***) – це об'єкт, який надає доступ до даних, які представляються у вигляді ***контейнеру (container***) та ***місткості (content***). Для прикладу в файловій системі каталоги – це контейнери, які можуть вміщувати інші каталоги (контейнери) або файли (місткість). Цей об'єкт дає можливість працювати з нереляційними даними.
7. ***Stream(Потік***) – це об'єкт, за допомогою якого можна працювати з потоками байтів файлів чи буферів пам'яті (дані content).
8. ***Collection (Колекція***) – це об'єкт, який вміщує декілька об'єктів одного типу, до яких можна доступитися по імені, чи індексу.
9. ***Event (Подія***).

Таким чином для доступу до даних за допомогою ADO, та їх модифікації необхідно провести таку послідовність:

1. Вказати джерело даних для Підключення, тобто його розміщення в Рядку Підключення або в URL. Виконати з'єднання з вказаним джерелом.
2. Визначити Команду для доступу до джерела даних, її тип та при необхідності параметри. Виконати команду.
3. Для реляційних баз даних результати виконання команди у вигляді таблиці записуються в кеш. При необхідності їх можна модифікувати за допомогою об'єкту Recordset.

Технологія ADO, в якості API інтерфейсу OLE DB, надає стандартні сервіси та методи роботи з даними незалежно від Провайдеру даних. Однак, в зв'язку зі специфікою останніх, можливі деякі особливості при їх використанні. Відмінності, як правило, стосуються Рядку Підключення (Connection String), використання Команди та об'єкту Recordset. Зупинимося на основних правилах формування Рядку Підключення та Команди.

Рядок Підключення (Connection String) формується з послідовності виразів типу *аргумент=значення* розділених крапкою з комою. Типові аргументи наведені в табл.8.1

Таблиця 8.1

Типові аргументи ADO

Аргумент	Пояснення
Provider=	Назва Провайдеру даних
File Name=	Назва файлу, де зберігаються наперед визначені настройки Підключення
Remote Provider=	Назва віддаленого Провайдеру (тільки для Remote Data Service)
Remote Server=	Шлях до віддаленого серверу (тільки для Remote Data Service)
URL=	URL-шлях до файла чи папки

Команда (Command) вказується за допомогою текстового рядку. Правила його формування залежать від типу команди. Зазвичай - це команда SQL, діалект якої залежить від Провайдеру даних, однак це може бути інший текст.

На сьогоднішній день найбільш використовувані Провайдери Даних: Microsoft OLE DB Provider for ODBC, OLE DB Provider for Microsoft Jet, Microsoft OLE DB Provider for SQL Server. Для більш детальної інформації по викладеному матеріалу та використання описаних технологій в інструментах програмування можна звернутися до довідкової інформації в MSDN.

Все більшого використання набуває технологія .NET, яка лягла в основу стандартного механізму доступу до даних під назвою ***ADO.NET***. Так само як ADO базується на OLE DB, ADO.NET базується на .NET Framework. Принципи застосування залишилися однакові. На момент написання книги відомі такі провайдери від Microsoft:

- Провайдер даних NET Framework для SQL-сервера;
- Провайдер даних .NET Framework для OLE DB;
- Провайдер даних .NET Framework для ODBC;
- Провайдер даних .NET Framework для Oracle.

Як видно зі списку, технологія ADO.NET підтримує сумісність із OLE DB та ODBC.

Література: [1], розділ 14.

Список літератури.

1. Пупена О.М., Ельперін І.В., Луцька Н.М., Ладанюк А.П. Промислові мережі та інтеграційні технології в автоматизованих системах: Навчальний посібник. – К.: Вид-во "Ліра-К", 2011. – 552 с.
2. https://sites.google.com/site/fieldbusbook/.