***Лекція №2***

**Тема:** «Об’єктна модель в СКБД Caché. Властивості класів у СКБД Caché та їх види»

**Об’єктна модель в СКБД Caché**

Об'єктна модель Caché розроблена у відповідності зі стандартом ODMG. **Клас** - це якийсь шаблон, відповідно до якого будуються конкретні екземпляри класу. **Об'єкт** у Caché має певний тип, тобто є екземпляром якогось класу. Стан об'єкту задають значення його властивостей, поведінку об'єкту задають його методи (операції). Інтерфейс об'єкта може мати одну або декілька реалізацій. Об'єктна модель Caché представлена ​​на рис. 1.

У Caché реалізовано два типи класів:

• Класи типів даних (літерали).

• Класи об'єктів (об'єкти).

Класи типів даних задають допустимі значення констант (літералів) і дозволяють їх контролювати. Кожен об'єкт має унікальний ідентифікатор, в той час як літерал не має ідентифікації, і від нього не можуть утворюватися екземпляри. Класи типів даних мають визначений набір методів перевірки та перетворення значень атрибутів. Ці класи не можуть містити властивостей.

Класи типів даних підрозділяються на два підкласи типів:

* Атомарні.
* Структуровані.

Атомарними літеральними типами в Caché є традиційні скалярні типи даних (% String,% Integer,% Float,% Date та ін.) У Caché реалізовані дві структури класів типів даних - список і масив. Кожен літерал унікально ідентифікується індексом в масиві або порядковим номером у списку.

Класи

Класи типів даних

Класи об’єктів

Незареєстровані класи

Зареєстровані класи

Класи, що зберігаються

Вбудовані класи

Рис. 1 – Види класів в СКБД Caché

Класи об'єктів підрозділяються на незареєстровані і зареєстровані класи.

Об'єкт може існувати в двох формах: у пам'яті процесу і у вигляді версії об'єкту, що зберігається у базі даних. У відповідності до цього розрізняють два типи посилань на об'єкт: для ідентифікації об'єкту в оперативній пам'яті використовується посилання **OREF** (object reference), якщо ж об'єкт зберігається в базі даних, йому призначається довготривалий об'єктний ідентифікатор - **OID** (object ID). Об'єкт отримує OID в момент першого записі в базу даних. OID не змінюється, поки існує об'єкт. OREF - призначається об'єкту, коли він потрапляє в оперативну пам'ять. При кожному новому завантаженні в оперативну пам’ять об'єкт може отримати нове об'єктне посилання OREF.

**Незареєстровані класи** не несуть в собі зумовленої поведінки, всі їх методи розробник робить сам. При цьому він сам відповідає за призначення та підтримку унікальних ідентифікаторів об'єктів (OID) і об'єктних посилань (OREF). Через те, що ідентифікація не реалізована на рівні Caché, незареєстровані класи виявляють ряд обмежень:

- Система не виділяє пам'ять для значень властивостей об'єктів;

- Відсутня автоматична підкачка об'єктів, на які робляться посилання;

- Поліморфізм не підтримується;

- Змінні, що посилаються на незареєстровані об'єкти, повинні явно декларуватись із зазначенням відповідного класу.

**Зареєстровані класи**, навпаки, забезпечені великим набором вбудованих методів, наприклад, таких як %New() для створення екземпляра об'єкта або %Close() - для видалення його з пам'яті. Ця зумовлена поведінка наслідується від системного класу %RegisteredObject. Екземпляри зареєстрованих класів існують лише тимчасово в пам'яті процесу, тому їх називають тимчасовими об'єктами. Протилежність ним складають вбудовані і класи, що зберігаються, які можуть тривалий час зберігатися в базі даних. Створення нових зареєстрованих об'єктів класів і управління ними в оперативній пам'яті виконує Caché. При цьому програмісту надається об'єктне посилання на екземпляр об'єкта, яке дозволяє посилатися на об'єкти оперативній пам'яті. Зареєстровані класи підтримують поліморфізм.

**Вбудовані класи** наслідують свою поведінку від системного класу %SerialObject і можуть тривалий час зберігатися в базі даних. Основною особливістю зберігання вбудованого класу є те, що об'єкти таких класів існують в оперативній пам'яті як незалежні екземпляри, однак можуть бути збережені в базі даних, тільки будучи вбудованими в інший клас. Основною перевагою використання вбудованих класів є мінімум витрат при зміні структури класів.

Вбудовані об'єкти представляються в оперативній пам'яті і базі даних вдосконалення абсолютно по-різному:

* В оперативній пам'яті цей об'єкт нічим не відрізняється від інших об'єктів. На нього вказує об'єктне посилання - OREF.
* У базі даних вбудований об'єкт зберігається як частина об'єкта, в який він вбудований. При цьому у вбудованого об'єкта відсутній OID, і він не може використовуватися іншими об'єктами без об'єкта, у який він міститься.

**Класи, що зберігаються,** наслідують свою поведінку від системного класу %Persistent. %Persistent надає своїм спадкоємцям великий набір функцій, що включає: створення об'єкта, підкачку об'єкта з бази даних в пам'ять, видалення об'єкта і т. п. Кожний екземпляр зберігаємого класу має 2 унікальних ідентифікатора - OID і OREF. OID ідентифікує об'єкт, записаний в базу даних, а OREF ідентифікує об'єкт, який був підкачаний з бази даних і знаходиться в пам'яті.

Об'єктна модель Caché в повному обсязі підтримує всі основні концепції об'єктної технології:

1. **Наслідування.** Об'єктна модель Caché дозволяє наслідувати класи від довільної кількості батьківських класів.
2. **Поліморфізм.** Об'єктна модель Caché дозволяє створювати додатки цілком і повністю незалежними від внутрішньої реалізації методів об'єкта.
3. **Інкапсуляція.** Об'єктна модель Caché забезпечує приховування окремих деталей внутрішнього устрою класів від зовнішніх по відношенню до нього об'єктів чи користувачів. Поділяють інтерфейсну частину класу і конкретну реалізацію. Інтерфейсна частина необхідна для взаємодії з будь-якими іншими об'єктами. Реалізація ж приховує особливості реалізації класу, тобто все, що не відноситься до інтерфейсної частини.

Клас у Caché зберігається в двох формах:

• Описова форма. Підтримується розвинена мова опису класів об'єктів UDL (unified definition language), побудована на базі XML (extensible markup language).

• Об'єктна run-time форма. Використання класу можливе тільки після його компіляції у об’єктний код.

**Елементи класів**

Повний список елементів визначення класу охоплює:

• **Однозначне ім'я класу**;

• **Ключові слова** - кілька ключових слів, які дозволяють модифікувати визначення класу

• **Властивості (стани)** - елементи даних для зберігання в екземплярах класу. Можуть бути константами, вбудованими об'єктами і посиланнями на збережені об'єкти. Класи типів не містять властивостей.

• **Методи** - програмний код, що визначає поведінку об'єкта.

• **Параметри класу** - константи, що здійснюють настройку функціональних можливостей класу під час його компіляції (зазвичай з використанням генераторів методів);

• **Запити** - операції з безліччю екземплярів класу;

• **Індекси** - структури в довготривалій пам'яті, що оптимізують доступ до об'єктів.

Робота з класами виконується в Caché Studio за допомогою майстрів, які дозволяють в режимі діалогу створювати класи, параметри класу, властивості, методи, запити, індекси. Перед використанням клас необхідно відкомпілювати.

Синтаксис визначення класу:

Class <Пакет.Імя класу> Extends <Список суперкласів>[Ключове слово = значення, ...]

{

Код

}

**Приклад 1.**

Class Cinema.TicketConfirm Extends (%CSP.Page, Cinema.Utils)

{

…

}

де Cinema – ім’я пакету, TicketConfirm – клас, %CSP.Page, Cinema.Utils – суперкласи, наслідником яких є клас TicketConfirm.

**Приклад 2.**

Class Im.Class1 Extends %Persistent [classType = persistent, procedureBlock]

{

…

}

де Im – ім’я пакету, Class1 – ім’я класу, %Persistent – суперклас, classType і procedureBlock – ключові слова.

Тепер поговоримо більш детально про кожний елемент класу.

1. **Ім’я класу**

Це унікальний ідентифікатор класу, для полегшення сприйняття допускається використання обох регістрів букв. Усередині Caché імена перетворюються в верхній регістр, що слід враховувати при призначенні імені. Імена, що починаються зі знаку %(відсоток) - зарезервовані для елементів системних класів. Обмежень на довжину імені немає, але тільки перші 25 символів використовуються для ідентифікації класу.

1. **Ключові слова**

Визначення класу може бути модифіковане допомогою декількох ключових слів. Усі ключові слова необов'язкові і мають стандартне значення на той випадок, якщо вони не задані. Ключові слова необхідні, перш за все, при розробці визначень класів. Приклади ключових слів:

* **Abstract** - означає, що не можна створити екземпляр даного класу;
* **ClassType** - визначає поведінку класу. Допустимі значення: datatype, persistent,і serial. наприклад:

1. ClassType = datatype - означає, що це клас типу даних.

2. ClassType = persistent – клас, що зберігається.

3. ClassType = serial - вбудований клас.

* **Final** - означає, що це фінальний клас, тобто від нього неможливе створення підкласів.
* **Super** - задає один або кілька суперкласів для даного класу. За замовчуванням клас не має суперкласу.

1. **Властивості**

Властивості представляють стан об'єктів. Детально вони будуть розглянуті нижче.

1. **Методи**

Це операції, асоційовані з об'єктом. Метод виконується всередині Caché процесу. Кожен метод може мати ім'я, список формальних параметрів, значення, що повертається, і програмний код. Ім'я методу повинне бути унікально всередині свого класу. Як правило, методи реалізовані у вигляді функцій або процедур на мові Caché Object Script (COS). Параметри методу можуть передаватися як за посиланням, так і за значенням. Тип значення, що повертається може бути яким завгодно, але, як правило, це тип % Library.Status, який повертає інформацію про статус завершення методу.

Є методи класів і методи екземплярів. Для завдання методу класу використовується ключове слово **ClassMethod**, для завдання методу екземпляра використовується ключове слово **Method**.

Наприклад, наступний синтаксис використовується для завдання методу класу:

ClassMethod <ім'я методу> (список параметрів) [ключові слова]

{ Код }

Такий синтаксис використовується для завдання методу екземпляру:

Method <ім'я методу> (список параметрів) [ключові слова]

{ Код }

1. **Параметри класу**

Це константи, тобто значення, що встановлюються під час визначення класу або в будь який інший час перед компіляцією класу. Параметри класу встановлюються для всіх об'єктів цього класу і можуть використовуватися в методах класу.

Під час виконання значення параметрів класу змінам не підлягають. Значення параметрів класу наслідуються і можуть перекриватися в похідних класах, або поповнюватися подальшими параметрами класу.

Наприклад, клас MyApp.A має параметр ABC із значенням 100

Class MyApp.A

{

Parameter ABC = 100;

}

Підклас MyApp.B може перекрити значення цього параметра.

Class MyApp.B Extends MyApp.A

{

Parameter ABC = 200;

}

Параметри класу зазвичай використовуються при компіляції з участю генераторів методів. Генератор методу використовує параметр класу з метою управління ходом генерації, наприклад для конкретизації поведінки типів даних.

Параметр класу має спеціальне поведінку, коли використовується з класами типів даних. З класом типів даних, параметр класу використовується для забезпечення способу зміни поведінки типу даних.

Наприклад, клас типу даних% Integer має параметр класу MAXVAL, який визначає максимальне значення для властивості типу% Integer. Визначимо клас з властивістю NumKids наступним чином:

Property NumKids As% Integer (MAXVAL = 10);

Це означає, що параметр MAXVAL для класу %Integer повинен бути встановлений рівним 10 для властивості NumKids.

1. **Запити**

Надають в розпорядження розробника операції з множинами екземплярів класів. Можна вважати, що запити утворюють для об'єктів класу якийсь фільтр.

Запити можна формулювати або на мові Caché Object Script, або на SQL. Результат запиту стає доступний через ResultSet - спеціальний інтерфейс для оброблення результатів запиту в прикладних програмах, написаних на мові Caché ObjectScript, або на будь-якій іншій з використанням ActiveX, або на мові Java. Крім того,запити можуть бути представлені у вигляді збережених процедур SQL або ж представлень (View), а значить, далі оброблятися засобами SQL.

1. **Індекси**

Це не що інше, як шлях доступу до екземплярів класу. Індекси використовуються для оптимізації швидкості виконання запитів. Зазвичай індекс будується для всіх екземплярів класу, включаючи всі його підкласи. Наприклад, індекс класу Person з підкласом Student включить в себе не тільки всіх «просто» осіб, але також всіх студентів. Індекс підкласу студент Student, навпаки, буде містити тільки студентів.

1. **Пакети**

Пакет це простий спосіб групування взаємопов'язаних класів під загальним ім’ям. Наприклад, програмний додаток може мати систему рахунків: "Accounting" і систему вкладень: "Inventory". Класи, реалізовані в цьому додатку, можуть бути організовані в пакети "Accounting" і "Inventory".

Inventory

Accounting

Warehouse

Invoice

Item

Item

На кожен з цих класів можна посилатися, використовуючи повне його повне ім'я, яке складається з імені пакету і імені класу:

Do ##class(Accounting.Invoice).Method()

Do ##class(Inventory.Item).Method()

Якщо ім'я пакету може бути визначене з контексту, то ім'я пакета може бути опущено:

Do ##class(Invoice).Method()

Пакет це просто угода про імена: він не забезпечує ніякої іншої функціональності, крім іменування класу. Так само як і клас, визначення пакету існує всередині робочої області Caché. Робоча область Caché - це логічне представлення даних, як правило, однієї бази даних.

Пакети маються на увазі при іменуванні класів. Найпростіший спосіб створення пакета - задати його ім'я при створенні нового класу в Caché Studio з використанням майстра, в якому також можна переглянути список всіх пакетів. При видаленні останнього класу пакета, сам пакет автоматично видаляється. Пакет можна вказати при програмному створенні класу. наприклад:

Class Accounting.Invoice{}

Клас Invoice всередині пакету "Accounting".

Існує два шляхи використання імені класу:

• Використання повного імені класу разом з ім'ям пакета. Наприклад: пакет.класс

• Використання короткого імені класу, дозволивши компілятору класів самому вирішувати якого пакету буде належати клас.

Для того, щоб компілятор сам міг вирішувати до якого імені належатиме клас, необхідна директива # IMPORT в коді .MAC або всередині класу. Якщо такої директиви немає, то ім'я класу буде асоційоване з пакетом "User" або "%Library".

Директива #IMPORT дозволяє задавати пакет, де слід шукати клас. наприклад:

# import Lab

/ / Клас "Patient" всередині пакету Lab

Set patient = ##class(Patient).%New()

Можна використовувати декілька директив # IMPORT всередині програми MAC. Порядок проходження директив # IMPORT не має значення і може призвести до помилки в разі двозначного використання. Наприклад, якщо є одне і те ж ім'я класу в двох різних пакетах, заданих директивою #IMPORT. Щоб уникнути цього використовуйте повне ім'я класу.

Директива IMPORT визначає, який пакет використовується для дозволу посилання всередині класу. Якщо директива не розміщена, то передбачається виконання наступного коду: # import User. Якщо одна директива вже визначена, то директива з User автоматично не підставляється, потрібно написати:

# import MyPackage

# import User

Кожен пакет відповідає SQL схемі. Наприклад, якщо клас називається Team.Player (клас Player в пакеті "Team"), відповідна таблиця називається"Team.Player" (таблиця Player в схемі "Team").

Пакет за замовчуванням "User" відповідає схемі "SQLUser". Отже, клас з ім'ям User.Person відповідає таблиці з ім'ям SQLUser.Person.

Якщо ім'я пакету містить точку, то вона замінюється подчерком. Наприклад: клас MyTest.Test.MyClass (клас MyClass, пакет "MyTest.Test") стає таблицею MyTest\_Test.MyClass (MyClass - таблиця, "MyTest\_Test" - схема).

Для сумісності з ранніми версіями використовуються «вбудовані» пакети:

• "% Library" – будь-який %class без імені пакета це просто частина пакета "%Library"

• "User" - будь-який не-% class без імені пакета належить пакету "User".

**Властивості класів у СКБД Caché та їх види**

Властивості являють собою стан об'єктів. Існує два типи властивостей:

• Властивості, що містять значення;

• Властивості-зв'язки, що задають зв'язки між об'єктами.

Властивість має однозначне ім'я, тип, необов'язковий список ключових слів, необов'язковий список параметрів, визначених для відповідного типу даних.

**Синтаксис:**

Property <Ім'я властивості> as <тип> (параметри) [ключові слова]

Можна змінювати визначення властивості, використовуючи одне або декілька ключових слів. Усі ключові слова є необов'язковими і мають значення за замовчуванням, якщо ключове слово явно не задано. Використовуються такі ключові слова:

**Calculated** - задає обчислювану властивість, яка не зберігається. Підкласи наслідують це ключове властивість і не можуть його перекривати;

**Description** - опис властивості, яку Caché використовує для побудови документації класу. Підкласами не наслідується;

**Final** - властивість, яка не може перекриватися в підкласах. За замовчуванням, властивості не є фінальними. Наслідується підкласами;

**InitialExpression** - задає початкове значення для властивості. За замовчуванням властивість не має початкового значення. Наслідується підкласами і може перекриватися;

**Private** - задає властивість типу private. За замовчуванням, властивості не є private. Наслідується підкласами і не може перекриватися;

**Required** - властивість повинна отримати значення перед збереженням на диску. За замовчуванням властивості не є required. Наслідується підкласами і може перекриватися;

**Transient** - визначає тимчасову властивість, яка не зберігається в базі даних. За замовчуванням властивості не є transient. Успадковується підкласами і не може перекриватись;

**Type** - задає ім'я класу, асоційованого з властивістю, це може бути клас типу даних, що зберігається або вбудований клас. За умовчанням це рядок (% String). Наслідується підкласами.

Існує кілька видів властивостей:

• Властивості типів даних;

• Властивості посилання на об'єкти;

• Вбудовані об'єкти;

• Властивості колекції;

• Потоки даних;

• Багатовимірні властивості.

1. **Властивості типів даних;**

Найпростіший тип властивостей - це властивості типів даних. Це літеральні значення, поведінка яких задається класом типу даних, асоційованим з властивістю. Наприклад:

Property Count As% Integer;

де Count - ім'я властивості, %Integer - тип даних, асоційований з властивістю.

При завданні властивості можна використовувати параметр, наприклад:

Property Count As% Integer (MAXVAL = 100);

Враховуючи приналежність до особливої форми класів, типи даних мають принципові відмінності від класів об'єктів:

• від класів типів даних неможливе утворення екземплярів;

• класи типів даних не можуть містити властивостей;

• методи класів типів даних надаються програмісту через інтерфейс типів даних.

Класи типів даних володіють наступними функціональними можливостями:

• Відповідають за перевірку значень, яка в подальшому може конкретизуватись за допомогою параметрів класів типів даних;

• Вони визначають перетворення між форматом зберігання (у базі даних), логічним (в пам'яті) і форматом відображення значень;

• Забезпечують взаємодію з SQL, ODBC, ActiveX і Java, надаючи вїх розпорядження необхідні в кожному випадку операції та методи перетворення даних.

Класи типів даних, що підтримуються Caché наведено в табл. 1.

Табл. 1

|  |  |  |
| --- | --- | --- |
| **Тип даних** | **Призначення** | **Аналогічний SQL-тип** |
| %Binary | Двійкове значення | BINARY, BINARY VARYING, RAW, VBINARY |
| %Boolean | Логічне значення (1 – істина, 0 – хибність) | Не визначений |
| %Currency | Валюта | MONEY, SMALLMONEY |
| %Date | Дата (внутрішній формат) | DATE |
| %Float | Число з плаваючою точкою | DOUBLE, DOUBLE PRECISION, FLOAT, REAL |
| %Integer | Ціле число | BIT, INT, INTEGER, SMALLINT, TINYINT |
| %List | Дані у форматі $ List,специфічному дляCaché | Не визначений |
| %Name | Ім'я в форматі "Прізвище.Імя " | Не визначений |
| %Numeric | Число з фіксованою точкою | DEC, DECIMAL, NUMBER, NUMERIC |
| %Status | Код помилки | Не визначений |
| %String | Рядок символів | CHAR, CHAR VARYING, CHARACTER,  CHARACTER VARYING,  NATIONAL CHAR, NATIONAL CHAR  VARYING, NATIONAL CHARACTER,  NATIONAL CHARACTER VARYING,  NATIONAL VARCHAR, NCHAR,  NVARCHAR, VARCHAR, VARCHAR2 |
| %Time | Час у внутрішньому представленні | TIME |
| %TimeStamp | Відмітка часу, що складається з дати і часу | TIMESTAMP |

Класи типів даних підтримують різні параметри, які змінюються відтипу до типу і виконують різні дії. Приклади параметрів:

MAXLEN - задає максимальне число символів, яке може містити рядок; MAXVAL - задає максимальне значення;

MINLEN - задає мінімальне число символів, яке може містити рядок; MINVAL - задає мінімальне значення;

FORMAT - задає формат відображення. Значення параметра відповідає опції форматування функції $ FNUMBER, яка виконує форматування;

SCALE - визначає число цифр після десяткової крапки;

PATTERN - задає шаблон, якому повинен відповідати рядок. Значення параметра PATTERN повинно бути правильним виразом шаблону Caché. Маска - це рядок, який задається спеціальним чином: числа визначають кількість обов'язкових повторів символу, зазначеного за числом. Тобто запис виду 3"a" означає обов'язкове повторення трьох символів "a", 14"1" - підряд чотирнадцять одиниць. Крім самих символів можна вказати спеціальні коди: **c** - означає будь керуючий символ (ASCII коди з 31 по 127); **n** - означає будь-яку цифру; **a** - означає будь-який символ з латинського алфавіту; **p** - будь-який символ пунктуації. Існують і деякі інші спеціальні символи. Наприклад, 3a1u означає - три поспіль літери, а потім обов'язково одна буква верхнього регістру. Отже, рядок "lovE" буде прийнята, як задовольняє масці, а от рядок "1миР" буде відкинута, оскільки містить 1, а це не буква алфавіту.;

TRUNCATE - якщо значення дорівнює 1, то рядок символів повинна обрізатися додовжини, зазначеної в параметрі MAXLEN (умовчання), якщо дорівнює 0, то немає;

VALUELIST - задає список значень для перерахованих властивостей;

DISPLAYLIST - визначає додатковий список значень перерахованих властивостей, задає формат відображення для перерахованої властивості, використовується разом з па-параметром VALUELIST. У табл. 2 наведені параметри, що підтримуються певними класами типів даних.

Табл. 2

|  |  |
| --- | --- |
| **Тип даних** | **Підтримувані параметри** |
| %Binary | MAXLEN, MINLEN |
| %Boolean |  |
| %Currency | DISPLAYLIST, FORMAT, MAXVAL, MINVAL, VALUELIST |
| %Date | DISPLAYLIST, FORMAT, MAXVAL, MINVAL, VALUELIST |
| %Float | DISPLAYLIST, FORMAT, MAXVAL, MINVAL, SCALE,VALUELIST, XSDTYPE |
| %Integer | DISPLAYLIST, FORMAT, MAXVAL, MINVAL, VALUELIST,  XSDTYPE |
| %List | ODBCDELIMITER |
| %Name | COLLATION, INDEXSUBSCRIPTS, MAXLEN, XSDTYPE |
| %Numeric | DISPLAYLIST, FORMAT, MAXVAL, MINVAL, SCALE,  VALUELIST |
| %Status |  |
| %String | COLLATION, DISPLAYLIST, MAXLEN, MINLEN, PATTERN,  TRUNCATE, VALUELIST, XSDTYPE |
| %Time | DISPLAYLIST, FORMAT, MAXVAL, MINVAL, VALUELIST |
| %TimeStamp | DISPLAYLIST, MAXVAL, MINVAL, VALUELIST |

1. **Властивості-посилання на об’єкти**

У Caché кожен клас можна розглядати і як тип даних. Якщо при визначенні властивості замість класу типу даних вказати клас, який зберігаємо, - буде створене посилання на зберігаємий об'єкт.

У цьому випадку значеннями властивості будуть посилання на екземпляри відповідного класу. Наприклад, так можна визначити властивість Doc, що є посиланням на зберігаємий об'єкт типу Doctor:

Property Doc As Doctor;

1. **Вбудовані об'єкти**

Вбудовані об'єкти функціонують аналогічно посиланнях на об'єкти. Суттєва відмінність полягає в тому, що в якості типу даних тепер задається не об'єкт, що зберігається, а вбудований клас об'єктів. Отже, значення цієї властивості - не посилання на якийсь самостійний екземпляр об'єкта, а навпаки: сам цей об'єкт, що вбудовується в об'єкт, який на нього посилається. З точки зору синтаксису визначення ідентично посиланню на Зберігаємий клас об'єктів. Наприклад

Property Adr As Address;

Де Adr це ім'я властивості, Address - це вбудований об'єкт.

1. **Відношення**

Відношення являють собою двонаправлені залежності між збереженими об'єктами. Детально відношення будуть розглянуті в лабораторній роботі №4.

1. **Властивості-колекції**

Властивість-колекція складається з набору елементів одного типу. Caché підтримує 2 типи колекцій: списки (List) і масиви (Array).

Колекція-список складається з упорядкування набору елементів (тобто кожному елементу ставиться у відповідність його порядковий номер).

|  |  |
| --- | --- |
| **Позиція** | **Значення елементу** |
| 1 | Book.Student 5 |
| 2 | Book.Student 1 |
| 3 | Book.Student 3 |
| 4 | Book.Student 2 |

Колекція-масив складається з набору пар виду: <ключ> <значення>, упорядкованих за ключем.

|  |  |
| --- | --- |
| **Ключ** | **Значення елементу** |
| Іванов А.А. | 28.03.1950 |
| Іванов А.Б. | 15.09.1967 |
| Іванов А.Г. | 12.03.1938 |
| Іванов А.Д. | 16.02.2000 |

1. **Потоки даних**

Потоком даних називають велику неструктуровану множину даних, яку потрібно зберігати в базі даних. Caché SQL забезпечує зберігання даних типу BLOBs (Binary Large Objects) і CLOBs (Character Large Objects) в базі даних. BLOBs використовуються для зберігання двійкових даних, таких як: малюнки, зображення, в той час як CLOBs використовуються для зберігання символьної інформації. Наприклад, потоки даних можуть застосовуватися для зберігання документів, технічних креслень чи малюнків. Окільки вони можуть бути дуже великі, Caché не оперує з потоками даних як з атомарною інформацією, а надає прикладним програмам методи блочного запису та зчитування даних.

Більш детально про потоки даних можна прочитати у книгах «Кирстен В., Ирингер М., Рёриr 5., Шульте П. СУБД Cache: объектно­ориентированная разработка приложений. Учебный курс.» та «Иванчева Н.А., Иваньчева Т.А. Постреляционная СУБД Caché (методическое пособие)».

1. **Багатовимірні властивості**

Властивість може бути оголошена як багатовимірна. Багатовимірне властивість поводиться як багатовимірна змінна, для якої можуть використовуватися функції роботи з багатовимірними змінними, які будуть розглянуті дещо пізніше.

Наприклад, властивість abc задана як багатовимірна:

Property abc [MultiDimensional];