УДК 004.655:004.657

В. І. Гайдаржи, *старший викладач,*
Є. С. Сідоров, *магістрант*
Національний технічний університет України «Київський політехнічний інститут»
вул. Політехнічна, 6, корпус № 5, м. Київ, Україна
sidoroveugene@gmail.com

ІНТЕГРАЦІЯ УНІВЕРСАЛЬНОЇ МОВИ ЗАПИТІВ LINQ SQL З ВНУТРІШНЬОЮ МОВОЮ ПРОГРАМУВАННЯ ПОСТРЕЛЯЦІЙНОЇ БАЗИ ДАНИХ CACHE

Задача підвищення ефективності програмних систем породжує проблему створення інструментів побудови універсальних запитів до даних, що зберігаються у послідовностях елементів різних типів.

У статті розглянуто питання щодо об'єднання об'єктно-орієнтованого підходу до представлення та зберігання даних у постреляційних СКБД разом з використанням інструментів побудови універсальних запитів до цих даних, що значно спростить процес розробки застосунків та дозволить використовувати єдиний синтаксис запитів для роботи з послідовностями об'єктів різних типів.

Ключові слова: *LINQ SQL запит, SQL запит, ООБД Cache, мова Cache Object Script.*

Постановка проблеми. У зв'язку з підвищенням вимог до аналітичної складової в системах управління виробництвом та в системах управління технологічними процесами набуває актуальності питання наявності засобів формування універсальних запитів, що надаються у розпорядження розробників програмного забезпечення.

Поширені інструментальні засоби розробки програмних продуктів, таких як, наприклад, Visual Studio 2010, надають розробнику широке коло типів, структур та контейнерів для зберігання даних у вигляді послідовностей відповідних елементів (рядків таблиць баз даних, елементів списку, послідовності екземплярів класів тощо.)

Серед систем керування базами даних, що використовуються розробниками, останнім часом поширилося використання об'єктно-орієнтованих даних, зокрема постреляційна ООБД CACHE, яка надає додаткові можливості проектування програмного забезпечення.

Задача підвищення ефективності програмних систем породжує проблему створення інструментів побудови універсальних запитів до даних, що зберігаються у послідовностях елементів різних типів. Застосування ООБД CACHE для проектування програмних систем з обробкою складних запитів спирається на використання засобів внутрішньої мови Cache Object Script, яка надає декілька типів послідовностей (таких як списки, таблиці, об'єкти та ін.), збережених у базі даних, до яких потрібно будувати складні запити. Мова пропонує декілька засобів формування запитів до даних, а саме: реляційний, об'єктний та прямий доступи. Ці засоби передбачають використання відповідних системних класів та системних функцій, але не передбачено існування деякої універсальної мови запитів.

Водночас існує відома технологія LINQ, яка є технологією формування універсальних запитів, інтегрованих до мов програмування високого рівня, таких як: C#, VB та ін. Було б корисно доповнити засоби мови Cache Object Script (COS) інструментами, необхідними для побудови запитів, аналогічних за своїми ознаками до запитів з використанням технології LINQ.

Саме цій проблемі присвячена дана стаття.

Аналіз підходів до формування запитів у Cache. Компанія InterSystem, яка є автором постреляційної об'єктно-орієнтованої СКБД Cache, запропонувала розробникам наступні засоби формування запитів до поточного змісту бази даних. В. А. Федоров [1] розглянув концепцію єдиної архітектури Cache, описав три основні доступи до бази даних: прямий, об'єктний та реляційний і дав загальний опис реалізації запитів до даних, відповідно до способів звернення. Колективом авторів МІФІ [2] наведено опис реалізації запитів до даних традиційними засобами внутрішньої мови програмування СКБД Cache Object Script.

У матеріалах А. Речитського [3] розглянуто питання модернізації засобів виконання запитів з метою усунення обмеження на необхідність визначення набору полів, що повертаються запитом у тексті COS програми шляхом створення спеціальних збережених процедур.

Є. Каратаєв [4] детально розглянув фізичну структуру збереження даних у глобалах Cache і питання індексації глобалів та організації конкурентного доступу з обробкою помилок без опису ефективних засобів звернення до глобалів.

Розглянемо основні засоби виконання запитів до бази даних мовою Cache Object Script.

Запити шляхом використання вбудованого SQL. В постреляційній базі даних Cache історично першою реалізована технологія Cache SQL – технологія реляційного доступу, що забезпечує максимальну продуктивність реляційних додатків з використанням вбудованого SQL. Cache SQL відповідає стандарту SQL 92. Крім цього, розробник може використовувати різні типи тригерів і збережених процедур.

Запити шляхом використання SQLStatement. Для усунення обмежень та незручностей вбудованого SQL в Cache додатково реалізована технологія формування запитів за рахунок використання класу %SQLStatement.

В усіх наведених вище випадках об'єкт підтримує стандартний інтерфейс. Кожний об'єкт містить властивості %SQLCODE, %ROWCOUNT, %MessageProperties та методи (наприклад, %Display()) для подальшої обробки отриманих результатів.

Запити в складі класу cache (SQL-запити та COS-запити). У СКБД Cache користувач має можливість формувати запити як складові класу, або безпосередньо використовуючи мову Cache ObjectScript, або мову SQL. Такі запити можуть бути представлені у вигляді збережених процедур або ж уявлень (View), які можуть бути оброблені засобами мови SQL.

Запити шляхом використання технології Cache Direct Access. Cache Direct Access містить набор функцій прямого доступу до структур зберігання (глобалей), який забезпечує максимальну продуктивність, але водночас потребує певного контролю логічної цілісності даних з боку програміста. Розробники додатків отримують можливість працювати безпосередньо із структурами зберігання, що дозволяє оптимізувати зберігання даних програми і створювати швидкі алгоритми обробки даних. Прямий доступ забезпечує максимальну продуктивність і може бути використаний для реалізації таких операцій, в яких застосування звичайних збережених процедур, що ґрунтуються на SQL, не може забезпечити необхідну продуктивність.

Запити шляхом використання технології Cache Objects. Cache Objects – набір системних класів, які забезпечують об'єктний доступ, з максимальною продуктивністю розробки програм користувача з застосуванням Java, Visual C++, VB й інших ActiveX-сумісних засобів розробки, таких як PowerBuilder і Delphi. У Cache повністю підтримуються спадкування (в тому числі й множинне), інкапсуляція і поліморфізм. Cache, підтримуючи об'єктну модель даних, дозволяє природним чином використовувати об'єктно-орієнтований підхід як при проектуванні предметної області, так і при реалізації програм в об’єктно-орієнтованих засобах розробки (Java, C++, Delphi, VB) [5].

Кожен із розглянутих засобів має свої особливості, які необхідно враховувати розробникам. Для підвищення зручності обробки запитів виникає потреба у застосуванні деякого універсального підходу до формування запитів, притаманного вищезазначеній мові запитів LINQ.

Метою статті є опис реалізації технології впровадження до мови Cache Object Script (COS) інструменту побудови універсальних запитів LINQ для доступу до послідовностей елементів, що створюються у COS.

Основна частина. Мова LINQ SQL. Використання технології LINQ надає можливість формувати запити мовою LINQ SQL – багатофункціональною мовою запитів до послідовностей різної природи з подальшою обробкою обраних елементів послідовності.

Для виконання операції запиту потрібно: визначити джерело даних (послідовність елементів списку, рядків таблиці, набору об’єктів тощо), сформувати вираз запиту, створити змінну запиту, яка буде містити вираз запиту. Для виконання запиту формується цикл foreach, в тілі якого користувач здійснює доступ до результату запиту. Результатом запиту є послідовність елементів, отримана із послідовності вхідних елементів, відповідно до умов виразу запиту.

Вираз запиту має починатися пропозицією from і закінчуватися пропозицією select або group. Між першою пропозицією from і останньою пропозицією select або group може міститися одна або кілька необов'язкових пропозицій where, order by, join, let або додаткових пропозицій from. Ключове слово into забезпечує збереження отриманих даних з використанням додаткових пропозицій (join або group) у тексті запиту.

У тексті запиту є можливість використовувати допоміжні функції з обробки елементів послідовності такі як: *об'єднання* (Aggregate, Average, Count, Min, Max, Sum), *конкатенація* (Concat), *перетворення* (Cast, OfType, ToArray, ToDictionary, ToList, ToLookup, ToSequence), *визначення елементу* (DefaultIfEmpty, ElementAt, ElementAtOrDefault, First, FirstOrDefault, Last, LastOrDefault, Single, SingleOrDefault), *визначення еквівалентності* (SequenceEqual), *створення елементів* (Empty, Range, Repeat).

Мова запитів дозволяє виконувати такі операції: *групування* (GroupBy), *приєднання* (GroupJoin, Join), *впорядкування* (OrderBy, OrderByDescending, ThenBy, ThenByDescending, Reverse), *розподілу на частини* (Skip, SkipWhile Take, TakeWhile), *проекцій* (Select SelectMany) та *обмеження* (All, Any, Contains, Where, Distinct, Except, Intersect, Union).

Опис технології Linq. LINQ (Language-Integrated Query) – це назва набору технологій, що ґрунтуються на інтеграції можливостей запиту безпосередньо в мову C# (а також у Visual Basic і, можливо, в будь-які інші мови NET). Завдяки LINQ запит є одним із основних структурних елементів мови, подібно класам, методам, подіям і т.д.

Технологія LINQ дозволяє використовувати SQL-подібний синтаксис безпосередньо в коді програми, написаними мовами програмування високого рівня, наприклад, мовою C#.

При створенні LINQ запиту користувач має можливість використовувати анонімні типи, вирази лямбда-обчислень, побудову вкладених запитів та визначення алгоритмів перетворення даних у процесі формування результатів запиту.

Підтримуючи механізм запитів для колекцій об'єктів у пам'яті, реляційних баз даних і даних у форматі XML, LINQ володіє розширеною архітектурою, яка дозволяє стороннім розробникам реалізувати доступ до їх сховищ даних через механізм LINQ.

В поширених мовах програмування використовуються наступні типи LINQ запитів [6], які представлені нижче (рис. 1):

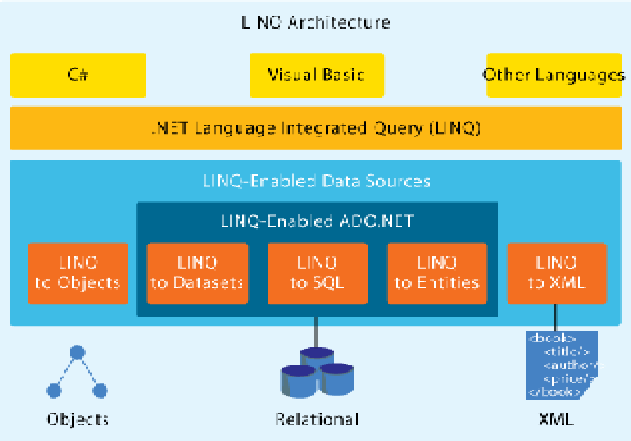

Рис. 1. Архітектура LINQ та доступні джерела даних

1. LINQ to Objects – дозволяє виконувати запити до масивів і колекцій даних, які знаходяться в пам'яті.

2. LINQ to XML – інтерфейс LINQ API, який потрібен для роботи з XML документами.

3. LINQ to DataSet – інтерфейс, який дає змогу використовувати технологію запитів LINQ до об'єктів, що базуються на використанні DataSet.

4. LINQ to SQL – програмний інтерфейс IQueryable<T>, який надає можливість використовувати LINQ запити для роботи з MS SQL Server.

5. LINQ to Entities – альтернативний інтерфейс, який використовується для звернен-

ня до баз даних. Він відокремлює сутнісну об'єктну модель від фізичної бази даних.

Нижче наведено приклад використання LINQ в програмі, написаною мовою програмування C#, який використовує як вхідну послідовність масив даних, обирає з неї елементи за вказаною умовою, перетворює їх та формує вихідну послідовність.

```
class LINQQueryExpressions
{
static void Main ()
 {
/ / визначення джерела даних.
int [] scores = new in t [] {97, 92, 81, 60};
  / / визначення змінної запиту з текстом запиту.
IEnumerable<int> scoreQuery = from score in scores
where score> 80
                    select score + 10;
  / / Виконання та обрбобка результату
foreach (int i in scoreQuery)
{ Console.Write (i + ""); }
  }
 }
```

Результатом запиту буде послідовність 107, 102, 91.

Основні принципи, які покладено в основу реалізації технології LINQ в межах мови Cache Object Script. Реалізація технології LINQ виконується в такій послідовності:

- Підготовка виразу-запиту.
- Трансляція виразу запиту в конструкції мови Cache Object Script, які передбачають формування динамічного SQL з використанням системного класу %SQLStatement, що відповідає LINQ запиту.
- Виконання запиту, яке полягає в створенні об'єкта класу %SQLResultSet, що містить результат виконання запиту.
- Звернення до результатів запитів звичайним для Cache Object Script чином, використовуючи методи класу %SQLResultSet.

Трансляція LINQ SQL запиту до SQL запиту здійснена з використанням описаних Раттцом ідей трансляції LINQ запитів [7].

Компілятор, під час обробки запиту, знаходить в запиті шаблони коду (code patterns) і перетворює їх у стандартну доступну для мови програмування нотацію (C#, наприклад). Компілятор виконує декілька кроків трансляції у встановленому порядку, для того щоб обробити весь текст запиту. Кожен з цих кроків шукає один або декілька взаємопов'язаних шаблонів коду. Компілятор повинен багаторазово транслювати усі входження шаблонів коду для даного кроку трансляції, перед тим як перейти до наступного кроку. На кожному кроці він виходить з припущення, що усі кроки для попередніх шаблонів вже перетворено.

Реалізація алгоритму трансляцій мови LINQ для СКБД Cache. В основі процесу обробки LINQ запиту покладено послідовні процеси перевірки запиту на наявність помилок, формування запиту мовою Cache Object Scrip та його виконання. Декомпозиція процесу обробки (рис. 2) складається з таких етапів:

1. Формування запиту – перетворення LINQ запиту до запиту мовою Cache Object Script, враховуючи тип елементів послідовності, до якої йде звернення у тілі запиту. Цей етап полягає у пошуку в тексті запиту основних шаблонів (термів) та відокремленні службових частин запиту (назв таблиць, полів, списків тощо).

2. Пошук логічних операторів – на сьогодні реалізована можливість побудови запиту з зазначенням умови пошуку необхідних елементів з вхідної послідовності (Equals, NotEqual). Основною задачею цього етапу є визначення умов, за якими буде відбуватися відбір елементів з вхідної послідовності, тобто визначення параметра пошуку, логічної операції та значення параметра, з яким буде відбуватися порівняння елементів вхідної послідовності.

3. Пошук математичних операторів – на сьогодні реалізована можливість побудови запиту з використанням математичних операцій (+, -, *, /) для додаткового перетворення результатів. Операція перетворення елементів результуючої послідовності визначається на етапі формування тексту запиту.

4. Пошук операторів конкатенації – на сьогодні реалізована можливість використання операцій конкатенації строкових змінних (_) для додаткового перетворення результатів. Операція конкатенації елементів результуючої послідовності визначається на етапі формування тексту запиту.

5. Підготовка запиту до виконання запиту. Оскільки сформований запит є об'єктом класу %SQLStatement шляхом виклику методу Prepare він готується до виконання. В ході підготовки відбувається його перевірка сис-

темними лексичним та синтаксичним аналізаторами.

6. Аналіз результатів перевірки. В разі відсутності помилок запит передається на виконання, інакше формується відповідне повідомлення.

7. Виконання запиту – перевірений запит відправляється на виконання за допомогою виклику відповідного методу системного класу %SQLStatement.

8. Відображення результатів – відображення результатів виконання запиту за допомогою виклику методу системного класу %SQLStatement.

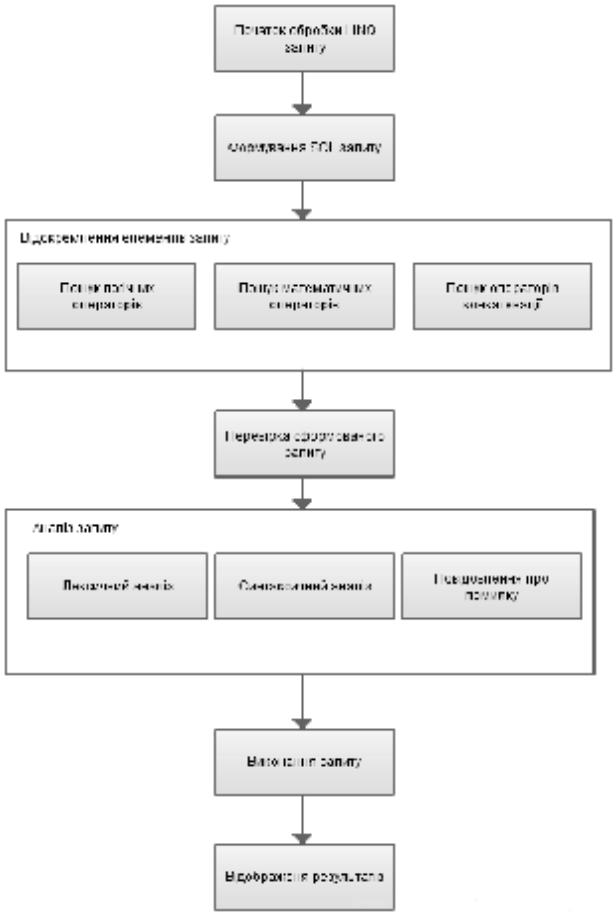

Рис. 2. Функціональна декомпозиція процесу трансляції запиту

На першому етапі інтеграції мови LINQ до внутрішньої мови програмування ООБД Cache, а саме – Cache Object Script, реалізовано систему, яка обробляє LINQ запити до таких послідовностей, як LB-списки та таблиці ООБД Cache.

З цією метою створено систему класів, методи яких дозволяють перетворити запити, написані мовою LINQ, до запитів, які реалізуються мовою Cache Object Script .

Запити LINQ в обох випадках перетворюються до COS запитів, які обробляються за допомогою технології Cache Object Script, що ґрунтується на використанні методів класу %SQLStatement, залежно від типу об’єкта, до якого будується запит.

Діаграма створених класів (рис. 3) включає такі класи:

1. Translator.LINQToList є нащадком системного класу (%SQLResultSet). В цьому класі виконано перевизначення методу Prepare(), в тілі якого відбувається виклик функцій трансляції. Перевизначення методу Prepare() здійснює попередню обробку LINQ запиту до LB списку з метою перетворення його до SQL запиту та виклик стандартного методу Prepare() для вирішення задач лексичного та синтаксичного аналізаторів перетвореного запиту.

2. Translator.LINQToTables є нащадком системного класу (%SQLResultSet). В цьому класі виконано перевизначення методу Prepare(), в тілі якого відбувається виклик функцій трансляції. Аналогічно перевизначений метод Prepare() здійснює попередню обробку LINQ запиту до таблиць з метою перетворення його до SQL запиту та виклик стандартного методу Prepare() для вирішення задач лексичного та синтаксичного аналізаторів перетвореного запиту.

3. Клас Translator.Parser містить допоміжні функції, що призначені для виділення окремих функціональних та логічних частин з виразу запиту, для подальшого використання під час формування виразу SQL Statement.

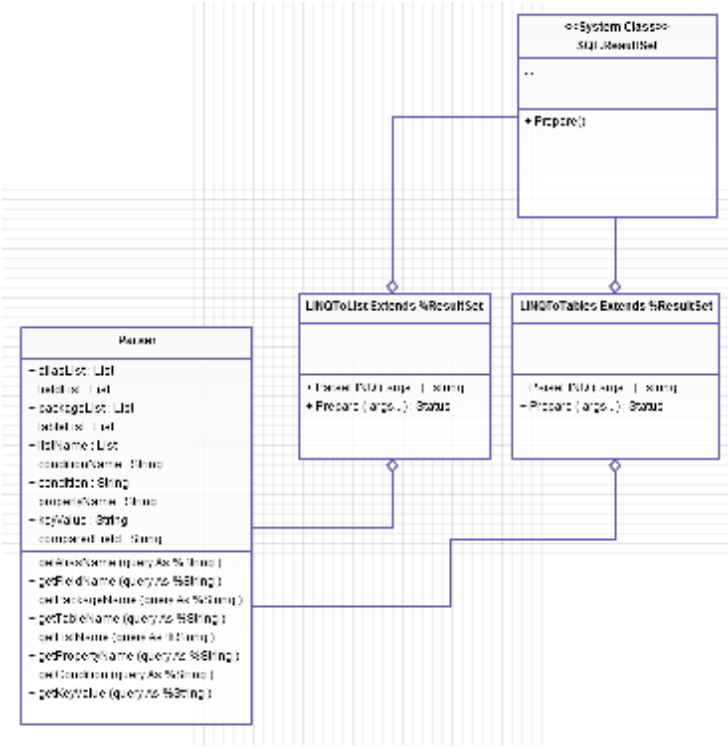

Рис. 3. Діаграма класів системи

Технологія використання мови LINQ в Cache Object Script.

Розглянемо наступний фрагмент коду, який реалізує LINQ запит в Cache Object Script.

```
set listOfInteger = $LISTBUILD(10, 15, 10, 30)
set lStatement2 = ##class(Translator.LINQToList).%New() )
set lQuery2 = "from elemenet in listOfDifferenTypes where element.NotEqual('10') select element + 10 ;"
do lStatement2.Prepare(.lQuery2, listOfInteger)
do lStatement2.%Execute()
w " List Second :", !
do lStatement2.%Display()
```

В цьому фрагменті виконуються такі дії:

- Створення деякої послідовності у вигляді списку (set listOfInteger = $LISTBUILD(10, 15, 10, 30))
- Створення об’єкта класу Translator.LINQToList з метою отримання доступу до методів виконання LINQ запитів (set lStatement2 = ##class(Translator.LINQToList).%New())
- Формування LINQ виразу у вигляді строкової змінної (set lQuery2 = "from elemenet in listOfDifferenTypes where element.NotEqual('10') select element + 10 ;")
- Підготовка запиту до виконання завдяки методу Prepare(), в який передається текст LINQ запиту та при необхідності (у випадку використання LINQ до списків) список як другий параметр

- Виконання запиту, використовуючи метод %Execute() (наприклад, do lStatement2.%Execute())
- Відображення результатів виконання запиту, завдяки використанню методу %Display (наприклад do lStatement2. %Display())

Висновки. Запропонований підхід реалізації LINQ запитів в Cache Object Script дозволить використовувати єдиний синтаксис запитів до будь яких послідовностей, що зберігаються в ООБД Cache.

Враховуючи той факт, що результат LINQ запиту, запропонованого в підході, є об‘єктом системного класу %SQL.ResultSet, можна застосовувати до нього всі методи цього класу для збереження та подальшого використання отриманих результатів.

Запропонований підхід реалізує побудову запитів LINQ до елементів таблиць та списків. В подальшому передбачається поширення цієї технології на інші елементи Cache Object Script (наприклад, на послідовності об’єктів).

Список літератури

1. Федоров В. А. Постреляційна СКБД Cache' [Електронний ресурс] / В. А. Федоров. – 2001. – Режим доступу : URL:http://citforum.ru/database/articles/cache.shtml
2. Постреляционная технология Caché для реализации объектных приложений / [Кречетов Н. Е., Петухов Е. А., Скворцов В. И. и др.]. – М. : МИФИ, 2001.
3. Речитський А. SQL-доступ до NoSQL-даних: реалізація SQL-процедур у Caché з динамічним визначенням метаданих, що повертаються [Електронний ресурс] / А. Речитський. – Б.м., 2012. – Режим доступу : URL:http://habrahabr.ru/company/intersystems/blog/160147/
4. Каратаев Е. MUMPS СУБД / Е. Каратаев. – М. : СОЛОН ПРЕСС, 2013. – 550 с.
5. СКБД Cache: объектно-ориентированная разработка приложений : учебный курс / [Кирстен В., Иринге М., Рёнинг Б., Шульте П.]. – СПб : Питер, 2001 – 384 с.
6. Барт Де Смет. IQueryable таблиці – LINQ до LDAP [Електронний ресурс] / Барт Де Смет. – Частина 0: Вступ – B#.NET. – Режим доступу : URL:http://community.bartdesmet.net/blogs/bart/archive/2007/04/05/the-iqueryable-tales-linq-to-ldap-part-0.aspx
7. Раттц Д. LINQ язык интегрированных запросов в C# 2008 для профессионалов / Д. Раттц ; пер з англ.. – М. : Вильямс, 2008. – 560 с.

References

1. Fedorov, V. A. (2001) Postrelational DBMS Cache'. [Internet]. Available from: <http://citforum.ru/database/articles/cache.shtml> [in Ukrainian].
2. Krechetov, N. E., Petukhov, E. A., Skvortsov, V. I. et al. (2001) Postrelational technology Caché to implement object applications. MEPI [in Russian].
3. Rechytskyy, A. (2012) SQL-access to NoSQL-database: SQL-procedures implementation in Caché with dynamic determination of returned metadata. [Internet]. Available from: <http://habrahabr.ru/company/intersystems/blog/160147/> [in Ukrainian].
4. Karataev, E. (2013) MUMPS database. Moscow: SOLON Press, 550 p. [in Russian].
5. Kirsten, V., Irinher, M., Roninh, B. and Schulte, P. (2001) DBMS Cache: object-oriented application development. St. Petersburg: Peter, 384 p. [in Russian].
6. Bart De Smet. IQueryable table - LINQ to LDAP [Internet]. Part 0: Introduction - B #.NET Available from: <http://community.bartdesmet.net/blogs/bart/archive/2007/04/05/the-iqueryable-tales-linq-to-ldap-part-0.aspx> [in Ukrainian].
7. Rattts, D. (2008) LINQ language of integrated queries in C # 2008 for professionals. Moscow: Williams, 560 p. [in Russian].

Стаття надійшла до редакції 22.04.2014.

V. I. Gaydarzhi, *senior lecturer*,
E. S. Sidorov, *undergraduate*
National Technical University of Ukraine "Kyiv Polytechnic Institute"
Polytechnic str., 6, building No 5, Kyiv, Ukraine
sidoroveugene@gmail.com

INTEGRATION OF UNIVERSAL LINQ SQL QUERY LANGUAGE WITH INTERNAL PROGRAMMING LANGUAGE OF POST-RELATIONAL CACHE DATABASE

The problem of increasing of software systems efficiency creates the problem of increasing of the efficiency of processing of huge amounts of data which are stored in sequences of elements of different types.

Object-oriented approach in object-oriented database management system provides developers with the benefits of using the basic paradigms of object-oriented technologies in data processing, presenting the information, stored in the database, in the form of objects (entities).

In its turn, LINQ, which is the technology of construction of universal queries, integrates the known technology into higher-level programming languages, such as C #, VB, etc.

As a consequence, a problem arises about the integration of the technology in the construction of universal post-relational query of object-oriented database management Cache system.

In this article the problem of combining object-oriented approach to data storage in a post-relational database management system and the use of Cache technology for constructing universal LINQ queries to different types of data sequences with internal programming Cache Object Script language (lists, tables, globalization, objects) is discussed.

The offered approach realizes the construction of LINQ queries to elements of tables and lists. In prospect, the further development of the system, providing the construction of queries to more elements of Cache Object Script is offered.

Keywords: *LINQ SQL query, SQL query, OODB Cache, Cache Object Script language.*