УДК 378.147

Побудова системи об'єктно-реляційної проекції

Микола Бондаренко,
кандидат технічних наук, професор,
Українська інженерно-педагогічна академія, Харків,

Антон Макаренко,
Українська інженерно-педагогічна академія, Артемівськ

Ключовою вимогою до мови програмування для побудови системи об'єктно-реляційної проекції є можливість рефлексивного програмування.

Велике значення для інформаційних технологій має розробка об'єктно-реляційної проекції (ОРП), яка пов'язує реляційні бази даних з концепціями об'єктно-орієнтованого програмування (ООП) шляхом створення «віртуальних об'єктних баз даних». Тому кваліфікаційні вимоги до спеціалістів з ООП передбачають оволодіння навичками підготовки та реалізації технологій ОРП, яка є типовою для мов програмування високого рівня [1]. На основі аналізу отриманих результатів розроблено рекомендації узагальнення теоретичної інформації в галузі проектування систем об'єктно-реляційної проекції, визначено ключові технічні вимоги для проектування ОРП-систем, проаналізовано механізми проекції та інтерфейси існуючих ОРП-систем [2]. Ключовою вимогою до мови програмування для побудови системи об'єктно-реляційної проекції є рефлексивне програмування [3]. За допомогою рефлексії можливо: змінювати ім'я класів, функцій та членів класів під час виконання; розбирати та виконувати програмний код, що надходить із зовнішніх джерел; знаходити та модифікувати конструкції коду програми як об'єкти першого класу; отримувати значення членів класу та викликати методи класу з їх іменами. На даний час існує декілька відомих реалізацій механізму рефлексії для мови програмування C++:

— Qt Framework;
— реалізація C++/CLI від корпорації Microsoft;
— бібліотека Mirror, яку засновано на останньому стандарті мови програмування C++ [4].

Результати аналізу підходів до організації рефлексії для мови програмування C++ подано нижче (табл. 1).

Мета-компіляція — це прийом програмування, який дозволяє оптимізувати програмний код алгоритму [5]. Вона дозволяє мінімізувати використання засобів статичного поліморфізму. Мета-дані використовуються не тільки як основна складова для опису класу, але й для опису реляційних таблиць та їх складових.

Розроблений механізм мета-компіляції схематично наведено на рис. 1.

Етап мета-компіляції складається з синтаксичного, лексичного аналізу та генерації мета-об'єктного коду. У результаті мета-компіляції мають бути сформовані:

Таблиця 1

Аналіз підходів до реалізації механізму рефлексії мови програмування С++

Підхід	Переваги	Недоліки
Аналіз інформації для відладки.	Використання стандартного методу опису класу. Можливість отримати повну інформацію про типи даних.	Необхідність побудови програми в режимі відладки. Залежить від компілятора.
Препроцесор аналізу та опису класу.	Використання стандартного методу опису класу.	Специфіка роботи компіляторів з членами класу та таблицями віртуальних функцій. Розробка аналізатора коду С++.
Модифікований компілятор з підтримкою рефлексії.	Відсутність додаткових етапів побудови програми. Генерація ефективного коду для засобів рефлексії.	Відтворення та адаптація програмної інфраструктури.
Ручне формування мета-опису класу засобами мови програмування.	Незалежність від компілятора. Проста реалізація.	Написання додаткового програмного коду, для опису метаінформації. Людський фактор.
Формування мета-опису класу в конфігурації зовнішніх класів.	Використання стандартного методу опису класу. Можливість використання стандартних засобів опису структур даних (XML, JSON).	Додаткові часові та ресурсні видатки на отримання мета-інформації.
Формування мета-опису класу засобами скриптової мови програмування.	Використання вбудованих можливостей скриптових мов програмування.	Часові та ресурсні витрати на запуск та роботу з інтерпретатором скриптової мови програмування.

— опис класу об’єкта віртуальної об’єктно-орієнтованої бази даних;
— допоміжний мета-об’єкт класу С++;
— вхідні дані для компіляції;
— опис реляційної таблиці.

Схема опису реляційної таблиці в СКБД SQLite наведена на рис. 2.

На основі аналізу структурних схем опису реляційних таблиць в СКБД, слід визначити необхідний перелік базових

Рис. 1. Механізм мета-компіляції

характеристик, які потрібні для опису полів мета-об'єктної таблиці:

1) Ім'я поля реляційної таблиці.

2) Тип даних, що зберігаються в полі.

3) Кількість знаків, що зберігаються в полі.

4) Можливість заборони пустих полів.

5) Значення, що присвоюється полю таблиці за замовченням.

6) Задання первинного ключа таблиці.

Загальними параметрами для поля класу і поля реляційної таблиці з встановленого переліку є ім'я та тип даних.

Виходячи з цього, необхідно встановити типи даних, що мають використовуватися для опису полів об'єктної таблиці, та показати їх співвідношення з типами даних C++ (табл. 2).

На основі співвідношення типів даних, структури опису поля таблиці пропонується наступна форма опису поля мета-об'єктної таблиці:

```
ТИП_ДАНИХ[(КІЛЬКІСТЬ_ЗНАКІВ
[,КІЛЬКІСТЬ_ЗНАКІВ])]
ІМ'Я_ПОЛЯ
  [NOT NULL]
  [DEFAULT]
```

Для відокремлення полів мета-об'єктної таблиці необхідно використовувати наступний маркер доступу — property.

```
    CREATE TABLE Person (
          id INT AUTO_INCREMENT
PIMARY KEY,
          name VARCHA(20) NOT NULL,
          surname VARCHAR(30) NOT
NULL
  )
  class Person{
    properties:
          VARCHAR(20) name NOT NUL;
          VARCHAR(30) name NOT NUL;
  };
```

Поле id у даному випадку генерується автоматично.

В роботі отримано такі основні результати.

1. Введено терміни об'єктно-реляційної проекції: проекція, проектувати, атрибут, проекція атрибута, проекція зв'язку.

2. Виконано теоретичний аналіз аспектів проектування архітектури системи ОРП та дизайну інтерфейсу користувача.

3. Показано, що системи ОРП програмного засобу заміщують компоненти логіки доступу до даних.

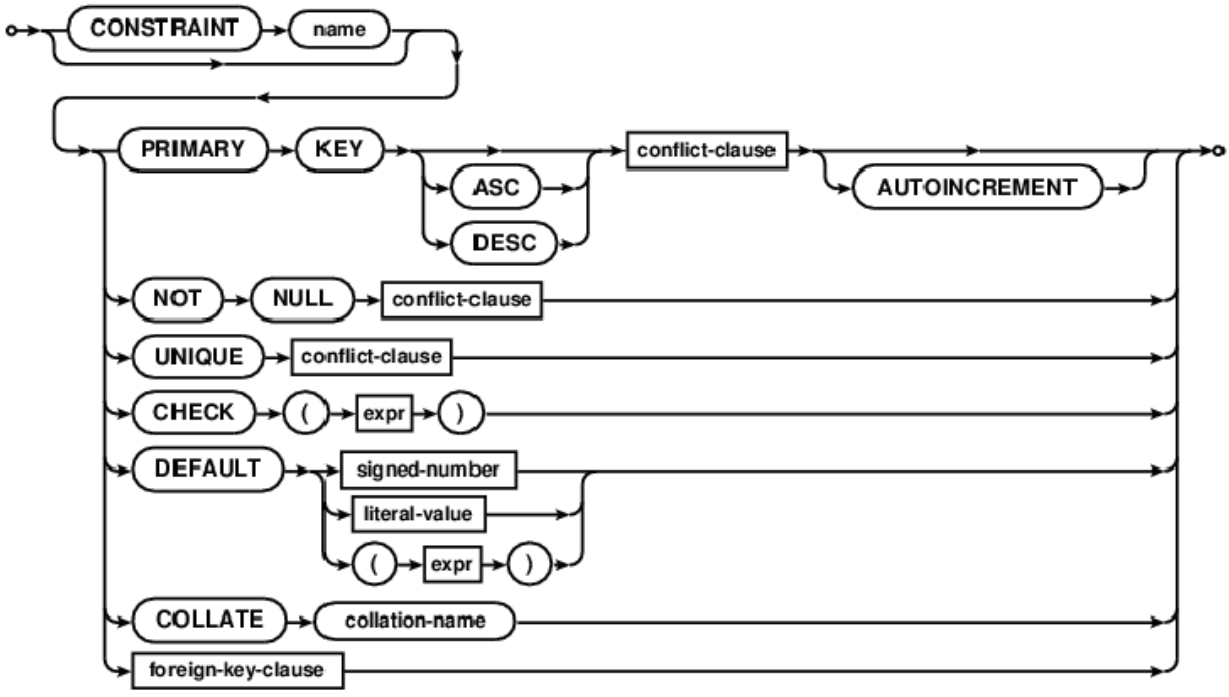

Рис. 2. Схема опису реляційної таблиці в СКБД SQLite

Таблиця 2

Співвідношення типів даних полів реляційних таблиць та типів даних C++

№ п/п	Тип даних в реляційній таблиці	Діапазон/формат значень	Тип даних C++
1	TINYINT	-127...128	char
2	SMALINT	-32768...32767	short
3	INT	-2147683648 ... 2147683648	int
4	BIGINT	263 ... 263-1	long
5	BOOLEAN	0/1 (false/true)	bool
6	FLOAT	-2 147 483 648.0 ... 2 147 483 647.0	float
7	DOUBLE	-9 223 372 036 854 775 808 .0 ... 9 223 372 036 854 775 807.0	double
8	CHAR	-	char[]
9	VARCHAR	-	char*/string
10	TEXT	-	char**/string
11	ENUM	0 ...	enum
12	DATE	от '1000-01-01' до '9999-12-31'	Розроблений конкретний клас
13	TIME	от '-838:59:59' до '838:59:59'	Розроблений конкретний клас
14	DATETIME	от '1000-01-01 00:00:00' до '9999-12-31 23:59:59'	Розроблений конкретний клас
15	TIMESTAMP	от '1970-01-01 00:00:00' до '2037-12-31 23:59:59'	Розроблений конкретний клас
16	YEAR(M)	от 1970 до 2069 для M=2 и от 1901 до 2155 для M=4	Розроблений конкретний клас

4. Механізм рефлексії є ключовою технічною вимогою до мови програмування для організації проекції реляційної таблиці на клас.

5. Стандарт мови C++ частково реалізує механізм інтроспекції типів даних — можливості доступу об'єкта до структури свого класу під час його виконання.

6. Показано, що загальними підходами для реалізації механізму рефлексії для C++ є аналіз інформації для відладки, створення препроцесора, модифікація компілятора, ручне формування метаопису класу. Системи ОРП для C++ відходять від об'єктно-орієнтованої концепції побудови програмних інтерфейсів.

Література

1. *Coleman, G., Verbruggen, R. A* Quality Software Process for Rapid Application Development // Software Quality Control. — 1998. — Vol. 7, № 2. — P. 107–122.

2. *Van Zyl P., Kourie D.G., Boake A.* Comparing the performance of object databases and ORM tools // Proceedings of the 2006 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries. Republic of South Africa: South African Institute for Computer Scientists and Information Technologists, 2006. — P. 1–11.

3. *Van Zyl P. et al.* The influence of optimisations on the performance of an object relational mapping tool // Proceedings of the 2009 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists. New York, NY, USA: ACM, 2009. — P. 150–159.

4. *Ireland C. et al.* A Classification of Object-Relational Impedance Mismatch // First International Conference on Advances in Databases, Knowledge, and Data Applications, 2009. DBKDA '09. 2009. — P. 36–43.

5. *Smith B.C.* Procedural reflection in programming languages: Thesis. Massachusetts Institute of Technology, 1982.

29.12.2014