УДК 681.03

А.Ю. Стеняшин

ПРО ФОРМАЛЬНИЙ ОПИС ТИПІВ І СТРУКТУР ДАНИХ РІЗНОРІДНИХ ПРОГРАМ

Розглядаються підходи щодо формального подання типів і структур даних мов програмування, предметно-орієнтованих мов типу DSL, що використовуються при опису різнорідних програм прикладних доменів. Дається аналіз специфікації готових програм за фундаментальними, загальними та специфічними типами даних при специфікації понять і завдань деяких предметних областей.

Вступ

На даний час в комп'ютерному інформаційному просторі накопичено велику кількість різнорідних програм, які використовуються багатьма постійно або частково у різних середовищах для обчислення деяких завдань. При появі нових комп'ютерів, мейнфреймів, кластерів, систем розподілених обчислювань (типу Grid) проблема опису і накопичення якісних готових програм продовжується. При обчисленні програм виникає проблема взаємозв'язків різних програм між собою, які використовують схожі типи даних (ТД) або несумісні в межах МП чи мов опису специфіки доменів з урахуванням особливостей сучасних архітектур комп'ютерів [1, 2].

Це породжує необхідність проведення поглибленого аналізу фундаментальних ТД у МП, загальних ТД (стандарт ISO/IEC 11404 – General Data Types) та даних із простору доменів, що подаються мовою DSL (Domain Specific Languages).

Дана стаття і присвячена дослідженню особливостей подання ТД в різних програмах [3].

Про особливості теорії подання типів даних в МП

Існує теорія подання фундаментальних ТД (простих, структурних і складних). Її аксіоматику розроблено відомими спеціалістами – Н. Агафонов, З. Дейкстра, Н. Вирт, В. Турський, П. Наур, та інші [4–7]. В ній структурна організація складних ТД (масивів, файлів, таблиць тощо) розглядається як агрегатна структура, що збудована з більш простих типів даних.

У теорії ТД тип розглядається як математичне поняття, яке позначає множину значень елементів. Базовий тип у цій теорії – це є елементарний тип змінної, що описується у деякій програмі. Тип привласнюється змінний, визначальний клас значень, кожне з яких належить одному й тільки одному типу. Операції над значеннями типу задаються аксіомами, а операції над ними – функціями, що слугує встановленню взаємно однозначної відповідності переданих ТД і перетворення значень одного з представлення у значення іншого типу у випадку їхній розбіжності [8].

Тобто ТД використовуються при опису функцій і програм в МП, які реалізуються системами програмування на різних платформах комп'ютерів у проміжний або у вихідний код. Кожна реалізована програма відображає відповідний ТД використаної МП, значення якого передається іншій програмі засобами викликів (звернення) до іншої програми.

Всі ТД МП діляться на *прості, структурні та складні*.

Прості ТД – це перелічені, а саме "цілий – integer (i) ", "дійсний – real (r) ", "булевий – boolean (b)" і "символьний – character (c)". Вони мають вигляд

type $T = X(x_1, x_2, ... x_n)$,

де T – ім'я типу, $(x_1, x_2, ... x_n)$ – імена значень з множини значень X типу T.

Операції над переліченими типами включають бінарні операції зіставлення та унарні операції pred і succ, які визначають відповідно попередній і подальший елементи в множині X. Операції зіставлення ($<$, $\le$,$>$, $\ge$, $=$, $\ne$), які надалі зазначаються $\le$ і визначають лінійний порядок елементів множини X.

Булевий тип визначений на множині значень X^b та операцій Ω^b:

$X^b = \{\text{false, true}\}$,

$\Omega^b = \{\&, \text{V}, \neg, \text{pred, succ}, \le\}$.

Символьний тип визначається на множині значень X^c і множині операцій Ω^c:

$X^c = \{... , 'A' ... , 'X' \ \dots \ , '0', '1', \ldots , '9'\}$,

$\Omega^c = \{\text{pred, succ, ord, char, } \le\}$.

Операція ord надає кожному символу його порядковий номер, а char для цього номера значення.

Аксіоми наведених типів даних мають такий вигляд:

$X.\min \in X, \quad X.\max \in X,$

$(\forall x \in X) \ \& \ (x \ne X.\max) \Rightarrow \text{succ}(x) \in X.$

$(\forall x \in X) \& (x \ne X.\max) \Rightarrow \text{succ}(x) \ne X.\min,$

де X.min і X.max – мінімальний і максимальний елементи множини X.

Числові типи (integer та real) мають обмеження, пов'язані з архітектурою або з явним описом параметрів компонентів. Їм відповідають базові типи, що визначаються як відрізки вигляду:

type $T = (X.\min, \ \dots, X.\max)$,

де X.min і X.max мінімальний і максимальний елементи цього відрізка.

Для будь-якого $x \in X$ виконується умова x.min < x < x.max. Значення x залежить від реалізації цього типу конкретним транслятором з урахуванням архітектури комп'ютера.

Над змінними цілого типу виконуються аналогічні операції, до них додаються операції цілочисельної арифметики: унарний мінус, +, –,×, div і mod.

Цілий тип даних на заданому відрізку визначається наступними аксіомами:

$X^i = \{X^i.\min, X^i.\max + 1, \dots X^i.\max\}$,

$\Omega^i = \{+, \ \times, \text{div}, -, \le\}$,

type $T^i = (X^i.\min, \dots, X^i.\max)$.

Дійсний тип визначається за допомогою операцій зіставлення та звичайних арифметичних операцій (унарний мінус, +, –,, /). Аксіоми дійсного типу на відрізку такі: $X^r = \{x \mid X^r.\min \le x \le X^r.\max\}$,

$\Omega^r = \{+, \ \times, /, -, \le\}$,

type $T^r = (X^r.\min, \dots, X^r.\max)$.

Будь-які ТД зводяться до базового типу і перетворюються до необхідного вигляду, заданому в програмі. Після отримання результату базовий тип приводиться до вихідного типу за допомогою наступних аксіом загального типу:

$(\forall x \in X) \Rightarrow T(T^0(x)) = x,$

$(\forall x_1 \in X) \& (\forall x_2 \in X) \Rightarrow (x_1 \le x_2) \equiv$

$\equiv (T^0(x_1) \le T^0(x_2))$.

У них T^0 позначає базовий тип для типу T. Операції перетворення значення типу $T^0(x)$ до $T(x)$ визначають відповідний базовий тип при виконанні арифметичних операцій вигляду $\oplus$:

$(\forall x_1 \in X) \& (\forall x_2 \in X) \Rightarrow (x_1 \oplus x_2) \equiv$

$\equiv T(T^0(x_1) \oplus T^0(x_2))$.

Структурні ТД – це масив, запис та таблиця, що будуються з базових і не структурних типів даних і таких, що містять набір впорядкованих елементів, кожен з яких обробляється як самостійний тип.

Масив конструюється з базових перелічених типів і представляється механізмом відображення множини індексів I масиву на множині значень Y елементів масиву:

$$M: I \rightarrow Y.$$

Над даним типом – масив можуть виконуватися наступні операції:

– упорядкування елементів масивів;

– додавання і віднімання елементів однотипних масивів,

– множення двовимірних масивів за правилами множення матриць та ін.

Операція множення накладає обмеження на область значень індексів масивів, що визначаються математичними правилами множення матриць. Операції додавання і віднімання виконуються для числових масивів, входять до складу множини спільних операцій над масивами і мають вигляд:

$$X^{\alpha} = \{x \mid (\forall x_1 \in X^{\alpha}) \,\&\, (\forall x_2 \in X^{\alpha}) \Rightarrow I(x_1) = I(x_2)) \,\&\, (Y(x_1) \cup Y(x_2) \subset \overline{Y}(X^{\alpha}))\}, \Omega^{\alpha} = \{\leq\},$$

type T^a = array $T(I)$ of $T(\overline{Y})$,

де $I(x)$ – множина індексів x для масиву, $Y(x)$ – множина значень елементів масиву, $\overline{Y}(X^{\alpha})$ – множина значень елементів для будь-яких типів масиву, T^a – ТД *масив*; $T(I)$ – тип даних індексів масиву; $T(\overline{Y})$ – тип даних для множини значень елементів масивів типу T^a.

Виходячи з другого виразу наведеного визначення до даного типу належать лише ті масиви, у яких множини індексів збігаються, а множини значень їх елементів належать одній і тій же числовій множині, що характеризує даний тип. Операції виконуються над масивами як над єдиним значенням, відповідним типу даних масиву. Дані типу $T(Y)$ в описі масиву T^a у свою чергу можуть визначатися рекурсивно, тобто через масив за наступною схемою:

type T^a = array $T(I^1)$ of $T(Y^1)$,

type $T(Y^1)$ = array $T(I^2)$ of $T(Y^2)$, що еквівалентно наступному запису:

type T^a = array $T(I^1 \times I^2)$ *of* $T(Y^2)$.

Запис, як і масив, представляє собою конкатенацію окремих компонентів, які можуть мати різні типи. Множина значень типу *запис* – прямий добуток множини значень її компонентів. До множини операцій, що виконуються над записами, належать тільки операції зіставлення, які використовуються для порівняння однотипних структур (типи компонентів порівнюваних записів та їх порядок слідування однакові):

$$X^{z} = \{x \mid (x=x^{v1} \times \ldots \times x^{vn}) \,\&\, (x^{v1} \in \in X^{v1}) \,\&\, \ldots \,\&\, (x^{vn} \in X^{vn})\}, \; \Omega^{z} = \{\leq\}.$$

Загальна форма представлення типу для запису має вигляд:

type $T^z = (S_{v1}:T^{v1}; \ldots S_{vn} : T^{vn})$,

де $S_{v1}, \ldots, S_{vn}$ – селектори, а $T^{v_1}, \ldots, T^{v_n}$ – типи даних компонентів запису.

У МП запис $(S_{v1}, \ldots, S_{vn})$ описується наступним чином:

type T^z = record

$S_{v1}: T^{v_1}$, : S_{v1}: T^{v_n} end.

Операції виконуються над записом як над єдиним структурним значенням. Обробка окремих компонентів запису виконується за допомогою операцій селектора, аналогічно відповідній операції для обробки масиву. Проведений аналіз стосується фіксованих записів. Для варіантних записів, послідовність компонентів яких визначається спеціальною ознакою, можна діяти в такий спосіб. Ознака є змінною перерахованого типу з кінцевою множиною значень. Кожному конкретному значенню відповідає визначе-ний вид запису. Тому замість одного варіантного запису розглядається сімейство фіксованих записів для кожного значення ознаки. Сімейство кінцеве, тому що кінцеве число елементів перерахованого типу.

Аналіз варіантного запису можливо звести до перебору отриманої родини даних й обробки конкретного фіксованого запису, тобто з розглядом тільки фіксованих записів.

Таблиця (table) належить до складних структур даних, складається з кінцевого числа рядків і стовпців. Вони

однозначно визначаються іменами, а їхнє положення в таблиці задається вказівкою номера (або ім'я) рядка й стовпця. Імена стовпців називають атрибутами (або ознаками, реквізитами), рядка – записами або кортежами, а їхні елементи – компонентами або полями.

Таблиця завдає ТД, значення якого становлять колекції значень з простору одного або декількох типів даних, що визначається полем, так що кожне значення задає асоціації між значеннями його полів. ТД поле можуть бути невизначено, тоді довільне значення типу даних таблиця містить задане число асоціацій.

Загальна форма представлення типу для таблиці має вигляд:

type $T^t = (S_{t1}:T^{t1}; \ldots S_{tn} : T^{t1})$,

де S_{t1} ,..., S_{tn} – селектори, а T^{t1},..., T^{t1} – типи даних компонентів таблиці.

У МП таблиця (S_{t1} ,..., S_{tn}) описується наступним чином:

type T^t = table

$S_{t1}:T^{t1}$; ,..., $S_{tn} : T^{t1}$

end.

Операції над таблицею виконуються як над єдиним структурним значенням. Обробка окремих компонентів таблиці виконується за допомогою операцій селектора.

Складні ТД. До них належать: множини, об'єднання, списки, послідовності, дерева та ін. Деякі з цих ТД – стандартні в конкретних МП, інші реалізуються шляхом моделювання через відповідні структури та операції над ними. При цьому подання даних операцій у МП менш формалізоване.

Множини. Загальна форма представлення цього типу даних має вигляд

type T = powerset T^0,

де T – тип множини; T^0 – базовий тип для його елементів.

Тип T включає операції над множинами як над математичними типами – об'єднання, перетин, різниця, включення, тотожність та ін. За допомогою операцій селектора проводиться вибір типу T^0 з об'єкта типу T, а за рахунок операції конструювання – формування з одного або з декількох елементів T^0 об'єкта типу T.

Об'єднання. Загальна форма для типу – об'єднання має вигляд:

type T = union (T^{v_1} ,..., T^{v_n}),

де Т – тип об'єднання, T^{v_1} ,..., T^{v_n} – базові типи.

Будь-який об'єкт типу T має значення і ознаку, за яким визначається один з типів, T^{v_1} ,..., T^{v_n} для даного значення. Механізм реалізації об'єднання подібний механізму реалізації варіантних записів. Відмінність полягає у тому, що сама ознака прихована на відміну від ознаки варіантної записи, в яку вона входить окремо.

Списки. Це конструктивний елемент МП ЛІСП. Елемент списку описується як запис з однією або декількома компонентами посилального типу, над якими виконуються операції, аналогічні операціям над фіксованими записами. Існують також операції, що застосовуються до цілого списку: вибір початкового елемента, отримання залишку списку, підключення списків, їх порівняння, інвертування, пошук елементів у списку та ін. Ці операції належать до засобів мови ЛІСП.

Послідовність. Загальна форма має вигляд: type T = sequence T^0, де T – тип послідовності, а T^0 – базовий тип. Послідовність – один з варіантів типу списку, в якого кожен елемент містить тільки одне посилання для забезпечення однобічного зв'язку. Операції над послідовностями аналогічні операціям над списками. Один з різновидів послідовності – рядок. Для рядка кожний елемент, крім посилальної змінної, містить елемент символьного типу.

Дерева. Це спискові структури, які використовуються для представлення графів або інших аналогічних об'єктів. Множина операцій над деревами аналогіч-

на множині операцій над списками. Реалізація цих операцій залежить від конкретних програм.

Крім розглянутих типів, у програмуванні використовуються різноманітні комбінації вищеперерахованих ТД, а також нові типи даних, що використовуються на практиці.

Особливості типів даних стандарту з GDT

Стандарт ISO/IEC 11404 визначає такі нові ТД, які відсутні у фундаментальних ТД.

Раціональний (rational) – це математичний тип даних, що відповідає раціональним (дійсним) числам.

Масштабований (scaled) – це родина типів даних, простором значень якого є підмножина простору раціональних чисел і кожний окремий тип даних має фіксований знаменник; цей тип даних передбачає апроксимацію значення.

Дійсний (real) – це родина типів даних, які є обчислювальними апроксимаціями щодо відношення до математичного ТД, що відповідає дійсним числам.

Комплексний (complex) – це родина типів даних, кожний з яких задає числову апроксимацію математичного типу даних, що подає комплексні числа. Кожний такий ТД становить колекцію математичних комплексних величин, які відомі у застосуваннях з деякою кінцевою точністю і мають розрізнятися принаймні з цією точністю у цих застосуваннях.

Пустий (void) – це тип даних, що подає об'єкт з необхідними синтаксичними і семантичними вимогами, але не несе жодної інформації у цьому конкретному випадку.

Крім того він включає нові складні ТД, а саме.

Набір (set) завдає ТД, простір значень якого становить набір всіх піднаборів простору значень типу даних елемент з операціями, властивими математичній множині *set.*

Портфель (bag) завдає ТД, значення якого становлять колекції зразків значень типу даних елемент. Численні зразки того ж значення можуть подаватися у цій колекції; а порядок, у якому вони присутні в колекції, несуттєвий.

У стандарті завдається простір значень ТД, як сукупність (колекції) значень типу даних, яка визначається одним з наступних способів:

– переліченням;

– аксіоматичним визначенням згідно основних положень;

– підмножиною вже визначеного простору значень, яка має той же набір властивостей;

– комбінацією будь-яких значень деякого, вже визначеного простору значень за допомогою специфікованої процедури конструювання нових значень.

Кожне окреме значення належить тільки одному типу даних, хоча воно може належати і декільком підтипам цього типу даних.

У кожному просторі значень існує поняття *рівності* (equality) за такими правилами:

– для будь-яких двох значень (*a, b*) з простору значень виконується умова значення *a* **дорівнює** *b* ($a = b$), або *a* **не дорівнює** *b* ($a \neq b$);

– не існує пари таких значень (*a, b*) з простору значень, для яких одночасно виконуються умови $a = b$ і $a \neq b$;

– для кожного значення *a* з простору значень виконується умова $a = a$;

– для будь-яких двох елементів значень (*a, b*) з простору значень $a = b$ тоді і тільки тоді, коли $b = a$;

– якщо для довільних трьох елементів значень (*a, b, c*) з простору значень виконуються умови $a = b$ і $b = c$, то виконується умова $a = c$.

Для кожного типу даних операція *рівності* Equal визначається як властивість рівності простору значень. Для будь-яких значень *a* і *b* з простору значень Equal (*a*, *b*) **є** *true* (істина)*,* якщо $a = b$, і *false* у протилежному випадку.

Простір значень *впорядкований*, якщо для нього встановлено відношення **порядку** (order), яке позначається як знак

≤ і задовольняє наступним умовам:

– для кожної пари значень (*a, b*) з простору значень виконується умова $a \le b$ або $b \le a$, чи обидві ці умови;

– для будь-яких двох значень (*a, b*), якщо $a \le b$ і $b \le a$, то $a = b$;

– для будь-яких трьох значень (*a, b, c*), якщо $a \le b$ і $b \le c$, то $a \le c$.

Запис $a < b$ визначає нотації відношень: $b \le a$ і $a \ne b$.

Даний ТД визначається на його просторі значень. Операція InOrder визначається так. Для довільних двох значень *a* і *b* з простору значень InOrder (*a, b*) є true, якщо $b \le a$, і false у протилежному випадку.

Простір значень ґрунтується на математичній концепції численності або кардинальності (cardinality), тобто він може бути скінченним, зчисленним нескінченним (зліченим) або незчисленним нескінченним. Тип даних повинен мати кардинальність (потужність) свого простору значень за категоріями ТД, простір значень яких такий:

– скінченний;

– точний (exact) і зчисленний нескінченний;

– наближений має скінченну або зчисленну нескінченну модель, хоча концептуальний простір значень може бути незчисленно нескінченним.

Кожний концептуальний тип даних обов'язково точний. Необчислюваний ТД є незчисленним нескінченним.

Якщо кожне значення у просторі значень концептуального типу даних можна відрізнити від іншого значення у просторі цієї моделі, то тип даних вважається точним (exact)**.**

Математичні типи даних, що мають значення, які не мають певного подання, звуться наближеними (approximate) та формуються наступним чином. Нехай *M* – математичний тип даних, а *C* – відповідний обчислюваний ТД, *P* – перетворення простору значень *M* на простір значень *C*. Тоді для кожного значення *v'* з *C* існує відповідне значення ТД *v* з *M* і таке дійсне значення *h*, що $P(x) = v'$ для всіх *x* з *M* та $|v - x| < h$.

Таким чином, *v'* – це наближення у *C* для всіх значень з *M*, які знаходяться в *h* – околі значення v''. Крім того, принаймні для одного значення *v'* з *C* існує більш ніж одне таке значення *y* з *M*, що $P(y) = v'$. Таким чином, *C* – це неточна модель *M*.

Особливості мови DSL

Мова DSL спрямована на відображення специфіки предметної області (ПрО), а МП загального призначення (Java, C++ й ін.) – на будь-якої тип програм. Ця мова застосовується для специфікації особливостей ПрО, вона близька до мов типу HTML, XML і має:

– абстракції, які забезпечують визначення концепцій і абстрактних понять із ПрО;

– синтаксис, призначений для природного представлення понять домену і запобігання синтаксичній неузгодженості понять ПрО;

– опис в DSL реалізується задопомогою спеціалізованих статичних аналізаторів для перевірки синтаксису, щоб знайти помилки в описі специфікації моделі або в опис деякої МП;

– предметні абстракції у МП базуються на використанні бібліотек програм і КПВ, що містять функції, класи і структури даних;

– інструменти підтримки DSL вимагають оточення, наприклад, середовища, редактора, засобу контролю версій і т.п.

Специфіка ПрО є загальною моделлю генерації домену GDM, що відбиває:

– поняття, характеристики ПрО і членів родини в просторі проблем;

– набір членів родини і їхніх специфікацій у мовах типу DSL, RSL, CSP і ін.;

– задачі ПрО в просторі задач для їхньої реалізації компонентами і з наступним збиранням їх у конфігурацію визначених членів родини;

– знання про конфігурацію мови (Configuration knowledge) відбивають характеристики членів родини і їхнє об'єднання в конфігурації;

– модель характеристики (feature models) відображає загальні, змінювані й незмінні характеристики членів родини, а також правила конструювання систем ро-

дини з урахуванням і їхньої взаємозалежності та залежності від КПВ;

– архітектуру (каркас) родини систем;

– реалізацію компонентів архітектури у мовах програмування.

При проектуванні каркаса або моделі ПрО у вигляді моделі GDM можуть задаватися механізми змінювання, синхронізації, безпеки з використанням аспектного програмування.

Головною частиною моделі ПрО є моделі характеристик, що призначені для подання вимог до сімейств ПС, різних характеристик систем, що входять до складу ПрО та застосування КПВ. Ці моделі поділені на дві категорії [3]:

– моделі характеристик FODA (Feature-oriented Development Architecture), FORM (Feature-oriented Reuse Method), RSEB (Reuse-Driven Software Engineering Business), FeatureRSEB (з інтеграцією у RSEB – діаграм характеристик з FODA-com) тощо [9];

– моделі варіантів сценаріїв використання (Use Case Variants) [10].

У першу категорію потрапляють традиційні методології моделювання, використовувані при побудові ПС, у другу – методології, засновані на аналізі сценаріїв діяльності потенційних користувачів ПС у ПрО, що будуються за допомогою UML (Sequence diagrams та Activity diagrams).

Про трансформацію опису моделей ПрО в DSL. Як сказано раніше, модель предметної області $M_{про}$ є загальною моделлю GDM домену, який складається з декількох моделей окремих прикладних систем $M_{про1}$, $M_{про2}$, ..., $M_{проN}$. Кожна з цих моделей відображає поняття і специфіку відповідної системи із родини систем домену. На їхньому перетині існують загальні поняття, характеристики й обмеження, які відображаються у моделі характеристик ПрО. Опис кожної моделі $M_{проі}$ виконується відповідною проблемно-орієнтованою DSL_i – мовою. Цей опис трансформується безпосередньо у відповідну $МП_i$, тобто мову реалізації, або в іншу DSL_j – мову, яка потім трансформується у МП. Програма у МП транслюється до вихідного коду стандартними компіляторами.

Аналіз опису різнорідних програм

Аналіз опису програм з використанням різних ТД показує, що головною проблемою взаємодії програм є інтерфейс, в якому дається набір параметрів з описом їх типів даних. Для забезпечення інформаційного зв'язування пари різномовних модулів склалися три класи операцій:

1) P – операції перетворення ТД модулів, що зв'язуються між собою;

2) S – операції селектора компонентів складного ТД;

3) C – операції конструювання структурних типів.

Операції класу 1 дозволяють здійснювати безпосереднє перетворення типу даних T_a^t у T_β^t без додаткових операцій зміни рівня структурування [2, 8]. Операцію перетворення ТД запишемо у вигляді: $P^{tq}{}_{a\beta} = (T_a{}^t\ ,\ T^q{}_\beta)$.

Тут дані ТД T_a^t перетворяться в $T^q{}_\beta$, a і β відповідають мовам l_a і l_β. Множина ТД кожної МП впорядкована й індекси t і q визначають конкретні елементи цієї множини. Для деяких МП t і q будуть функціями від інших індексів і упорядкованість їх типів може визначатися конструюванням нового типу t з типів у яких індекси не більше t. Новий тип буде мати індекс, що функціонально залежить від індексів визначених типів і конкретних операцій конструювання.

Операції класу 2 слугують вибору зі структурного типу його окремих компонентів з меншим рівнем структурування. Механізми реалізації цих операцій відмінні від аналогічних, наявних у МП, тому що вони не повинні змінювати структури даних, тих, що безпосередньо обробляються в модулях.

Операції класу 3 є зворотними щодо операцій селектора. Їхні механізми конс-

трування відмінні від аналогічних операцій, що є в МП.

Множина розглянутих операцій класу 1, 2, 3 охоплює перетворення ТД для МП, функції конструювання структурних типів і вибору окремих компонентів. Детальний розгляд показує, що для даної множини операцій повнота відсутня. Причини цього розглядаються далі.

Виходячи з наведених визначень постановка задачі перетворення ТД за заданим набором операцій класів 1–3 має такий вигляд.

Нехай дано клас $L = \{l_1, l_2, \ldots, l_n\}$ і для кожної з окремої мови відомі множини ТД і операції конструювання нових типів.

Необхідно: побудувати для структурних типів операції перетворення ТД $P = \{ P^{tq}_{\alpha\beta} \}$, операції селектора S і конструювання C.

Для рішення поставленої задачі проводимо розбивку множин фактичних і формальних параметрів і будування відображень ТД за допомогою операцій P, S і C. Якщо відображення одного типу до релевантного типу іншого не вдається, то це означає, зв'язування пари модулів не можливо або воно може бути реалізоване з порушенням деяких властивостей, що розглядалися.

Визначимо апарат опису ТД МП. Кожен ТД характеризується множиною значень, що можуть приймати змінні цього типу, і множини операцій над цими змінними.

Тому найбільш підходящим методом є опис ТД як алгебраїчних систем.

Операції перетворення типів $P^{tq}_{\alpha\beta}$ мають забезпечувати не тільки однозначну відповідність множини значень перетворюваних типів даних у модулях, які викликають, та тих модулях що викликаються, але й однакову інтерпретацію операцій над даними в цих модулях. При цьому має здійснюватися як пряме перетворення даних викликаючого до викликуваного модуля, так і зворотне. При такому підході операції перетворення $P^{tq}_{\alpha\beta}$ відповідають ізоморфним відображенням однієї алгебраїчної системи в іншу. Тобто ми маємо діло з ізоморфним відображенням множин фактичних і формальних параметрів.

Підходи до опису даних БД

Проблема перетворення даних має місце при використанні різних система керування базами даних (СКБД), оскільки дані мають різні способи їх подання і зберігання. Серед даних можуть опинитися несумісні ТД або доступ до них здійснюється різними мовами маніпулювання і засобами.

Перетворення форматів даних. Програми, що розташовані на різних типах комп'ютерів, передають один одному дані через протоколи, повідомлення, їхні формати перетворюються до формату даних приймаючої серверної платформи (так званий маршалінг даних) з урахуванням порядку і стратегії вирівнювання, прийнятої на цій платформі. Демаршалінг даних – це зворотне перетворення даних (тобто отриманого результату) до виду клієнтської передавальної програми. Якщо серед переданих параметрів оператора виклику містяться нерелевантні типи або структури даних, які не відповідають параметрам викликаного об'єкта, то проводиться пряме і зворотне їхнє перетворення засобами стандарту або МП [11]. Засобами перетворення даних за їхніми форматами є:

– стандарти XDR – eXternal Data Representation, CDR – Common Representation Data, NDR – Net Data Representation) і методи їхнього перетворення;

– МП і мови опису інтерфейсів (RPC, IDL і RMI) для передачі даних між різними компонентами.

XDR – стандарт містить у собі засоби опису структур даних довільної складності і засоби перетворення даних, що передаються на платформи (Sun, VAX, IBM і ін.). Програми в МП можуть використовувати дані в XDR – форматі, не зважаючи на те, що компілятори вирівнюють їх в пам'яті машини по – різному. Перетворення даних – це кодування (code) або декодування (decode) XDR – процедурами форматування простих і складних типів даних. Вирівнювання даних – це розміщення значень базових типів з адреси, кратної дійсному розміру в байтах (2, 4, 8, 16). Межі даних вирівнюються за найбільшою довжиною (наприклад, 16). Системні процедури оптимізують розташування полів пам'яті під складні структури даних і перетворюють їх до формату приймальної платформи. Оброблені дані декодуються назад до виду формату передавальної платформи.

CDR – стандарт середовища CORBA забезпечує перетворення даних у формати платформи, що їх передає або приймає. Маршалінг даних виконує інтерпретатор TypeCode і брокер ORB. Перетворення складних типів виконуються процедурами encoder ( ) і decoder ( ) інтерпретатора TypeCode, який використовує базові примітиви при вирівнюванні інформації і розміщенні її в буфері. Для складного типу визначається розмір і межі вирівнювання, а також їхнє розміщення в таблиці з урахуванням індексів значень, які ініцилізуються брокером ORB.

XML – стандарт забезпечує усунення неоднорідності у взаємозв'язках компонентів у різних МП за допомогою XML – формату даних, який враховує різні платформ і середовища. XML має різну системну підтримку: браузер Internet Explorer для візуалізації XML – документів, об'єктна модель DOM (Document Object Model) для відображення XML – документів і інтерфейс IDL в системі CORBA. Тобто, XML – мова дозволяє представляти об'єкти для різних моделей на єдиній концептуальній, синтаксичній і семантичній основі. Він не залежить від платформи і середовища взаємодії компонентів прикладного рівня.

Перетворення даних БД. Перетворення даних в БД пов'язане з різницею логічних структур даних (ієрархічні, мережні, реляційні) в різних БД і СКБД, в довідниках, класифікаторах і в системах кодування інформації, а також різних мов для подання інформації тощо.

Перехід до реляційної моделі даних і СКБД ґрунтується на теорії множин, математичній логіці та опису програм в SQL – мові. При переході від одної БД до БД нового типу зіставляються дані старої і нової БД і змінюється довідкова інформація і класифікатори. В нових БД може бути декілька мов, тому для зберігання даних з простим доступом до текстових даних встановлюється відповідність текстових даних, записаних у різних мовах. Наявність явної несумісності типів і структур різних моделей даних забезпечується різними мовами шляхом внесення змін в БД та концептуальну модель даних. Внесені зміни відображаються в довідниках і класифікаторах, що забезпечує перенесення даних із старої БД до нової з урахуванням поточних змін.

Перетворення даних БД може проводитися кілька разів шляхом створення спеціальних скриптів і DBF – файлів з урахуванням раніше введених даних, без їхнього дублювання і коректного зведення несумісних типів даних. Це перетворення з перенесенням даних у різні СКБД розв'язується через спеціальний драйвер або транзитні файли, в які копіюються дані із старої БД для перенесення їх у нову БД.

У випадку коли дані із старої БД переносяться до транзитних файлів, SGL –

скриптів, DBF – файлів з заданими форматами даних, вони пересилаються до нової транзитної БД засобами мережі. Якщо друга СКБД реляційного типу, то дані в транзитних файлах перетворяться до табличного вигляду. При не реляційна СКБД дані приводяться до табличного вигляду і першої нормальної форми. Подальша нормалізація даних і зведення їх до структури нової БД здійснюється в транзитній БД з використанням 3 – або 4 – нормальної форми. Кожна вища форма нормалізації містить у собі як підмножину нижчу форму, наприклад, першу нормальну форму у вигляді скалярних значень.

Іншими словами, відношення знаходяться в першій нормальній формі, якщо вони зберігаються в табличному вигляді (всі чарунки в рядку таблиці розташовані в строго певній послідовності) і кожний елемент таблиці містить у собі тільки атомарні значення (елемент не є множиною).

Відношення знаходиться в третій нормальній формі тоді і тільки тоді, коли кожний кортеж складається із значення первинного ключа, що ідентифікує деяку суть, і набору пустих значень або значень незалежних атрибутів цієї суті. Тобто відношення знаходиться в третій нормальній формі, коли не ключові атрибути – взаємно незалежні, але залежать від первинного ключа.

Два або декілька атрибутів – взаємно незалежні, якщо жодний з них не залежить функціонально від комбінації решти атрибутів. Подібна незалежність припускає, що кожен атрибут можна оновлювати незалежно від інших.

Процес нормалізації відношень дозволяє позбавитися проблем, які можуть виникнути при оновленні, внесенні або видаленні даних, а також при забезпеченні цілісності даних. Структури старих БД не завжди можна привести до третьої нормальної форми, тому потрібно, щоб дані, що знаходяться в транзитних файлах, існували хоч би в першій нормальній формі й належали до реляційної моделі.

Приклади перебудови даних для БД. Як приклад розглянемо підхід до перетворення ТД, а саме строкових, двійкових, символьних щодо сучасної MSDN (Microsoft Developer Network) [12].

Ці дані мають різний опис (char, varchar, nchar, nvarchar, binary, varbin ary тощо) та представлення у БД. Перетворення їх до єдиного виду створює сервер. Коли ТД binary або varbinary подаються нерівної довжини, то SQL Server перетворює ці дані шляхом їх усікання праворуч. При перетворенні цих типів даних у тип binary або varbinary вони доповнюються ліворуч шістнадцятирічними нулями. Перетворення цих ТД будь – якого досить великого значення у двійкове й обернено до первісного значення, виконуються в одній і тій же версії сервера. Залежно від версії SQL Server двійкове подання значень може мінятися.

Перетворення даних цілого типу int, smallint і tinyint до типу даних binary або varbinary приводиться тільки до значення binary і обернено до цілочисельного шляхом усікання вихідного подання.

Наприклад, SELECT CAST(123456 AS BINARY(4)) представляє цілочисельне значення 123456 у вигляді двійкового 0x0001e240. Якщо цільовий тип binary SELECT CAST (123456ASBINARY (2)) розміру 2, то для зберігання цільового значення початкові цифри усікаються й те ж саме число зберігається як 0xe240. Перетворення будь-якого типу даних до типу binary завжди залежить від версії SQL Server.

Перетворення символьних даних БД**.** При перетворенні символьного вираження в символьний тип довгі дані так само усікаються при отриманні нового

типу даних. Якщо символьне вираження перетвориться в символьне вираження іншого типу даних або розміру, наприклад, з char (5) в varchar (5) або char (20) в char (15), то перетвореному значенню привласнюються параметри сортування вхідного значення. Якщо не символьне вираження перетвориться в символьний тип даних, то перетвореному значенню привласнюються параметри сортування, задані за замовчуванням у поточної БД.

Перетворення кодових сторінок виконується для типів даних char і varchar. Символьні вираження, які перетворяться в наближений тип даних numeric, можуть містити необов'язкову експонентну нотацію (символ *E* нижнього регістра або *E* верхнього регістра, за яким ідуть необов'язковий знак плюс (+) або мінус (–) і число). Символьні вираження, пре утворені в точний тип даних numeric, складатися із цифр, десяткового роздільника й необов'язкового знака плюс (+) або мінус (–). Перетворення типу integer у дані типу character, SQL Server вставляє символ з кодом ASCII 42 – зірочку (*). Цілочисельні константи, що перевищують 2 147 483 647, перетворяться в тип даних decimal.

Таким чином, дана стисла характеристика основних принципів перетворення даних у системі SQL Server 2008.

Висновок

У роботі розглянуто загальну аксіоматику ТД, що використовуються у сучасних МП. Результатом дослідження є опис фундаментальних типів даних у МП і їх використання при рішенні задач взаємодії різномовних програм у межах сучасних середовищ. Наведено формальне подання простих, структурних і складних типів даних МП. Обґрунтовано місце цього напряму дослідження при інтеграції різномовних програм і забезпеченні їх взаємодії у сучасних середовищах. Деяка частина роботи займає аналіз сучасних підходів до опису даних БД і методів їх перетворення при поданні або переносі даних з однієї БД в іншу. Результати роботи будуть корисні в інфраструктурах середовищ виконання програм, а також при обчислюванні задач в системах типу Grid.

1. *Лаврищева Е.М.* Проблема интероперабельности разнородных объектов, компонентов и систем. Подходы к ее решению // Матер. 7 Міжнар. конф. з програмування “Укрпрог–2008”. – С. 28–41.
2. *Лаврищева Е.М., Грищенко В.Н.* Сборочное программирование. Основы индустрии программных продуктов.– К.: Наук. думка, 2009. – 371 с.
3. *Jeff Grey et al*. OOPSLA’02 Workshop on Domain Specific Visual Languages, 2002. http://www.cis.uab.edu/info/OOPSLA–DSVL2
4. *Агафонов В.Н.* Типы и абстракция данных в языках программирования // Данные в языках программирования. – М.: Мир, 1982. – С. 267 – 327.
5. *Замулин А.В.* Типы данных в языках программирования и базах данных. – М.: Наука, 1987. – 152 с.
6. *Hoar K.O* структурной организации данных // Структурное программирование. – М.: Мир, 1975.– С. 92 – 197.
7. *Турский В.* Методология программирования: Пер. с англ. – М.: Мир, 1981. – 265 с.
8. *Лаврищева Е.М.* Сборочное программирование. Теория и практика // Кибернетика и системный анализ. – 2009. – № 6. – С. 3 – 12.
9. *Kang K.C., Cohen S.G., Hess J.A., Novak W.E. and Peterson A S.* Feature-oriented domain analysis (FODA) feasibility study. Technical Report CMU/SEI-90–TR–21, Software Engineering Institute, Carnegie Mellon University, 1990.
10. *Consel C*. From a program family to a domain specific language // Symposium on Principles

of Programming Languages, Charleston, SC, USA, ACM Press, 1993. – P. 19–29.
11. *Лаврищева Е.М.* Програмна інженерія.– Підручник. – К.: Академперіодика, 2008. – 317 с.
12. Библиотека MSDN:http://msdn.microsoft.com/ru–ru/library/ms191530.aspx

Отримано 24.02.2011

Про автора:

Стеняшин Адрій Юрійович,
аспірант Інституту програмних систем НАН України.

Місце роботи автора:

ПТЗТ «Скарбниця».
E–mail: andrey.stenyashin@gmail.com