УДК 681.03

ОБРОБКА ДАНИХ У ГЕТЕРОГЕННИХ МЕРЕЖАХ ТИПА GRID

А.Ю. Стеняшин

Інститут програмних систем НАН України,
e-mail: andrey.stenyashin@gmail.com, тел.: (068) 365 6466

Розглядаються підходи до обробки даних, переданих між різнорідними науковими програмами e-sciences в гетерогенної середовищі Grid, а також інтерфейс опису параметрів і їх типів даних, які при передачі від однієї до іншої програми можуть зажадати їх перетворення до інших форматів і типів. Запропоновано апарат генерації фундаментальних і загальних типів даних, описаний в стандарті ISO / IE 11404-2007, представлений набором примітивних функцій перетворення і використовуваний при вирішенні задач обробки даних в системі MySQL Grid

The article studies the data processing methods, transferred between the manifold scientific programs e-sciences in the heterogeneous environment such as Grid, as well as the interface of programs description and their data types, which may require to be converted into the other forms and types when transferred from one program to the other one. Basic and general data types generation mechanism is proposed, described in ISO/ IEC 11404-2007 standard, represented by the primitive set of conversion function and used during the data processing solution tasks.

Вступ

Одним з важливих науково-практичних рішень щодо забезпечення *індустрії обчислень* є поява системи Grid у 2006 р. в межах Європейського проекту. Цю систему розроблено як інфраструктуру для глобальних обчислень суперскладних задач з області e–science.

Термін Grid (сітка, решітка) почав використовуватися в середини 90-х років і був обраний з аналогії з мережами передачі та розподілення електроенергії (Power Grids).

Розвиток і впровадження технології Grid носить стратегічний характер. У перспективі ця технологія дозволить створити принципово новий обчислювальний інструмент для підтримки високих технологій у різних видах людської діяльності. Цей інструмент буде забезпечувати глобальну інтеграцію, об'єднання інформаційних і обчислювальних ресурсів на основі мережних технологій і спеціального ПЗ проміжного рівня між базовим і прикладним ПЗ, а також набору стандартизованих служб для надійного доступу до географічне розподілених ресурсів: окремих комп'ютерів, кластерів, глобальних сховищ інформації, даних і різним середовищам.

Для обчислень складних наукових задач у системі Grid застосовуються розроблені або готові програмні й системні ресурси виду: розподілені процесорні потужності; системи сховищ даних; новітні засоби комунікації; фонди reuses з багатьох доменів; нові технічні устаткування і загальні «тули» (конвертори, генератори, трансформатори, верифікатори тощо); моделі та схеми взаємодії програм у середовищі гетерогенних платформ, географічно розташованих у віддалених адміністративних доменах тощо.

Середовище *Grid* як інфраструктура обчислень різних наукових задач пропонує підсистему ETICS для організації зборки різнорідних і різноплатформенних програм рішення задач за їх описом мовами програмування. Завдання з обчислення задач виконуються із застосуванням різних системних і прикладних сервісів Інтернету і досягнення максимальної продуктивності рішення задач глобального масштабу.

Іншим напрямком практичного обчислення задач глобального типу є *хмарні обчислення,* які підтримуються новими засобами Cloud Computing системами IBM-VSphere та системами Microsoft – WCloud, Azure, Amazon, WApps, SkyDriven.

Вони зорієнтовані на збереження даних, доступ до глобальних сховищ даних on-line, синхронізацію даних великих розмірів і викладання їх на віддалені сервери тощо. Тут головна проблема організації обчислень потребує визначення нових методів, таких як координація, кооперація та взаємодія різних сервісів і інших готових ресурсів через конфігураційний файл для виконання обчислень відповідних задач.

Накопичення готових ресурсів у різних глобальних сховищах для доступу до них різних наукових користувачів потребує розвитку індустріальних методів і моделей з урахуванням особливостей і специфіки наукових задач.

Індустрія ПП базується на готових звичайних компонентах (артефактах, reuses, assets і даних) багаторазового використання, що знаходяться у різних бібліотеках, репозиторіях та нових «хмарних» сховищах Інтернету. Головним методом розроблення наукових продуктів є удосконалений метод конфігурування, зборки наукових різнорідних ресурсів, що належить до напрямків e-science, в структури

ПП, котрі після їхнього виготовлення можуть бути подані на різні глобальні сервери для їх застосування при рішенні відповідних наукових задач.

Засобом зв'язку різнорідних програм є інтерфейс, який представляє собою набір операцій з параметрами, що забезпечують визначення видів послуг і способів їх передачі від одного програмного продукту до іншого. В ролі інтерфейсу виступали оператори звернення типу CALL до процедур і функцій програми з відповідними формальними параметрами. Програми, процедури і функції записувались в МП, а оператори звернення до них включали в себе імена процедур і функцій що викликаються і список фактичних параметрів, що задають значення формальним параметрам для отримання результатів. Послідовність і число формальних параметрів відповідало фактичним параметрам. Виконання функції в середовищі програми на одній МП не викликало проблем, так як типи даних параметрів збігалися [1-3].

Коли один із елементів – програма, процедура, функції записані на різних МП і ,крім цього, якщо вони розташовані на різних комп'ютерах, то виникають проблеми неоднорідності представлення типів даних параметрів з цих МП, структур пам'яті платформ комп'ютерів, компіляторів і операційних середовищ їхнього виконання.

Інтерфейс грає роль посередника по передачі інформації від однієї програми що викликається, програмі що викликає, що записані на різних МП. У ньому дано опис формальних і фактичних параметрів, виконана перевірка відповідності параметрів (кількості, порядку розташування), а також типів даних параметрів. Якщо типи даних параметрів виявлялись не релевантними (наприклад, передається ціле число, а результат функції з плаваючою точкою або навпаки), то в посереднику необхідно виконати пряме і зворотне перетворення типів даних з урахуванням структури пам'яті їх комп'ютерів.

У проміжному середовищі розглянутих систем реалізується два способи зв'язування об'єктів : на рівні МП через інтерфейси прикладного програмування і компіляторів IDL, які генерують клієнтські і серверні stub. Інтерфейси визначаються в IDL або APL, а динамічний виклик здійснює ORB для об'єкта - клієнта і об'єкта-сервера. Ці інтерфейси мають окрему реалізацію і доступні із програм, що записані на різних МП. Компілятори IDL, як частина проміжного шару, самі реалізують зв'язування з відповідною МП за допомогою механізму посилань.

Інтерфейс в IDL або в API забезпечує взаємодію компонентів на основі опису їх формальних і фактичних параметрів і порядку завдання операцій передачі параметрів для отримання результатів. Цей опис є нічим іншим як специфікація інтерфейсного посередника двох різномовних об'єктів, які взаємодіють на різних процесах сервера чи клієнта.

У функції інтересного посередника клієнта входять:

- підготовка зовнішніх параметрів клієнта для звернення до сервісу сервера;
- відправка параметрів компонента сервера, його виконання для отримання результату або видача відомостей про помилки.

Загальні функції інтересного посередника сервера полягають у наступному:

- отримання повідомлень від клієнта, запуск віддаленої процедури, обчислення результату і підготовка його для передачі клієнту;
- повернення результату клієнту в параметрах повідомлення, знищення віддаленої процедури й інші.

1. Про перетворення параметрів в операторах викликів

Програми, що розміщені на різних типах комп'ютерів, передають один одному дані через протоколи, формати яких зводяться до формату даних приймаючої серверної платформи (маршалінг даних) з урахуванням порядку і стратегії форматування, прийнятої на цій платформі. Демаршалінг даних – це зворотне перетворення даних (отриманого результату) до вигляду передаючої клієнтської програми. Якщо серед параметрів виклику, що передаються можуть містити не релевантні типи або структури даних, які не відповідають параметрам викликаного об'єкта, то виконується їх перетворення [4].

Стандарт ISO/IEC 11404-2007 регламентує опис загальних типів даних, які можуть використовуватися будь-якими сучасними МП.

До засобів перетворення даних і форматів також належать:

- стандарти кодування даних (XDR-eXternal Data Representation, CDR-Common Representation Data [4], NDR-Net Data Representation) і методи їх перетворення;
- МП і механізми звертання компонентів один до одного;
- мови описання інтерфейсів RPC, IDL, RMI і т.п.

На кожній платформі комп'ютера використовується згода про кодування символів (наприклад, ASCII), про формати цілих чисел і чисел з плаваючою точкою (наприклад, IIEEE,VAX й ін.). Для представлення цілих, як правило ,використовується додатковий код, а для типів float і double – стандарт ANSI /IEEE й ін.

На порядок перебудови типів даних впливає розташування залежно від структури платформи комп'ютерів (Big Endian і Little Endian) і від старшого до молодшого байта і від молодшого до старшого. Процесори UltraSPARC і Power PC підтримують обидві можливості. При передачі даних з однієї платформи на іншу враховується можливість не співпадання порядку байтів даних.

Технологія Grid орієнтована на інтенсивну обробку даних з забезпеченням максимальної швидкості обчислень за рахунок глобального розподілення обчислення між комп'ютерами (наприклад, проект DEISA (www.desia.org), як об'єднання супер комп'ютерних великих обсягів даних за допомогою простих програм

типа «одна задача – один процесор». При цьому в доставки даних для обробки та передачі результатів під егідою Європейського Союзу (www.eu-egee.org) представниками будуть більш 90 наукових і навчених організацій світу.

Побудова проекту EGEE орієнтовано на застосування його для обробки даних в фізики високих енергій, що проводяться на базі Європейського центру ядерних досліджень (CERN, www.cern.ch) прискорителя LHC. Проект EGEE тісно пов'язаний з проектом LCG (LHC Computing Grid), який є технологічною базою (www.egee-rdig.ru).

2. Аналіз обробки даних в системі Grid

З 2007 р. почав діяти Європейський проект Grid – мережна інфраструктура для організації розподілених обчислень у задачах різних наукових напрямках (фізика, математика, медицина, біологія й ін.). До його складу входить GCube - операційне середовище, ETICS - збіркове середовище тощо [5, 6].

Зокрема, система ETICS призначена для автоматизованої побудови, конфігурування, інтеграції і тестування різного роду програм і систем для вирішення науково-дослідних задач. Її архітектура базується на каркасі FP6 ЕС, націленому на створення розподілених систем шляхом інтеграції існуючих процедур, інструментальних засобів і ресурсів даної інфраструктури з застосуванням Веб-порталу для керування мультиплатформенними ресурсами середовища Grid.

Головним поняття в системі ETICS – **проект** (рис. 1) і модулі даних, що завдають конкретні типи даних простої і складної структури. Програмні об'єкти в Etics розробляються відповідно стандарту CIM (Common Information Model), який призначений для опису програм і систем об'єктно –орієнтованими засобами, а саме, класи, об'єкти, методи і властивості. Цей стандарт встановлюю єдині вимоги з подання об'єктів і даних і дозволяє керувати різними об'єктами, наведеними з використанням уніфікованого набору визначень, команд і сценаріїв роботи **(use case).**

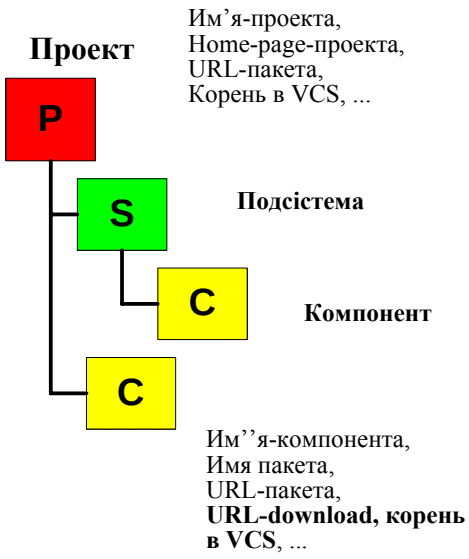

Рис. 1. Структура проекту в Etics

Проект (Project). Контейнер компонентів ПС, що визначають властивості і політики (правила), застосовувані за замовчуванням для всіх компонентів (наслідувані).

.**Підсистема (Subsystem)**. Логічний фрагмент архітектури ПС, що забезпечує підмножина функціональності проекту. Контейнер для компонентів зі специфічними властивостями і політиками, наслідуваними від підсистеми.

Компонент (Component). Мінімальна одиниця функціональності в архітектурі проекту. Належить підсистемі або безпосередньо проектові.

Модуль загальна назва для проекту, підсистеми і компонент.

Модулі в ETICS повинні мати однозначну відповідність з модулями **VCS**, використовуваної для проекту. Для модулів типу «проект» і «підсистема» у VCS може не бути установлені правила відображення.

Деякий модуль має метаопис в ETICS, структура якого визначається видом модуля.

Зв'язки між цими об'єктами задаються **інтерфейсом** в мовах IDL або API.

ETICS містить типовий набір характеристик і процедур для побудови і тестування нових пакетів і послуг. Цей набір буде розширюватися шляхом додавання нових плагінів зі спеціалізованими послугами для конкретної області знань. Кожен плагін включає публічний інтерфейс з описом послуг для споживачів або постачальників. Спочатку цей набір охоплює засоби керування робочих місць завданнями, зв'язаними з ОС, архітектурою CPU і компіляторами в МП. Крім того, у нього входить механізм специфікації залежностей між різними пакетами і їх тестами, що автоматично керують побудовою, тестуванням, компілюванням або розгортанням. Велика кількість функціональних плагінів призначена для перевірки договорів, тестів виконання різних елементів систем, генерації документації і ведення готових об'єктів програм в оперативному або постійному репозиторіях ETICS.

Технологія створення великих наборів пакетів з вихідних або комбінацій перекомпільованих двійкових елементів підтримується процесом доступу до репозиторіїв для формування розподіленої версії і її репродукції. Скомпільована структура автоматично знаходить і завантажує з репозиторію підходящі двійкові елементи. При цьому головною проблемою тут, як і в інших розглянутих середовищах, є проблема відображення елементів системи на альтернативну платформу. Вона вирішується шляхом побудови перехресних посилань (cross platform) до необхідної платформи і генерації модулів, що розставляють необхідні

прапорці в скомпільований код (наприклад, при виконанні готових 32-розрядних двійкових елементів на 64-розрядних платформах) мережного середовища Grid.

Наступна важлива проблема – це стандартизація опису типів даних для споруджуваних у системі ETICS трьох головних об'єктів: *Проект, Підсистема і Компонент. Проект* може складатися з Підсистем або Компонентів. Підсистема може містити тільки Компоненти. У системі запропонована модель даних з типовим форматом CIM для взаємин між різними об'єктами. Сутність її полягає в наступному.

Модель даних, як і модель CIM, дозволяє уводити формальні сутності в структурі проектів ПО, описувати об'єкти і взаємини між ними, а також надавати результати виконання ПО. Внутрішнє збереження даних ґрунтується на моделі даних реляційного типу, реалізованої засобами MySQL. Вона підтримує різні моделі розгортання (веб-ервіс, веб-астосування і БД), що знаходяться на одному вузлі, а використовуваний сервіс, база даних – на іншому вузлі.

Опис моделі даних засновано на наступних базових положеннях:

– кожен компонент містить опис властивостей (ім'я, ліцензія, URL репозиторія і т. д.), глобального унікального ідентифікатора – ID (GUID):

– об'єкт конфігурації містить інформацію про версії, зв'язку з репозиторієм, GUID, вид платформи і зв'язку цього ідентифікатора з глобальним у конфігурації;

– об'єкт містить команди перевірки (checkout) скомпільованого елемента, тестових команд і GUIDs, а також зв'язки з кожною конфігурацією;

– при визначенні конфігурації і платформи в кожному об'єкті з'являється GUID, його властивості, середовище виконання і залежності, що можуть бути оголошені статично або динамічно. Статична залежність – це взаємовідношення між двома конфігураціями, динамічна залежність – взаємовідношення між конфігурацією і модулем.

ETICS за функціями найближчий до сучасної фабрики програм, базується на наборах характеристик, послуг і процедур виготовлення пакетів. Вони можуть об’єднуватись плагінами (plugins) з описом послуг для споживачів або постачальників, засобами керування завданнями з робочих місць, а також доступу до ОС, архітектури CPU, компіляторів з МП і засобами специфікації залежностей між різними пакетами та їхніми тестами при збірці й розгортанні. Множина функціональних плагінів забезпечує перевірку контрактів, тестів виконання різних елементів систем, генерацію документації, ведення готових програм в оперативному або постійному репозиторії ETICS.

Технологію створення великих наборів пакетів із вихідних або сукупностей перекомпільованих програм забезпечено процесами доступу до репозиторіїв, формування розподілених версій та їх репродукцій. Головним гальмом є перебудова деяких компонентів систем для альтернативної платформи гетерогенного середовища комп’ютерів шляхом посилань з 16-, 32-розрядної платформи на 64-розрядну платформу середовища Grid.

Система сервісу в Grid. Веб-служба OGSA (Open Grid Services Architecture) забезпечує опис і виявлення необхідних сервісів при розробленні і виконанні ПС; автоматичну генерацію програмних частин на сторонах клієнта або сервера на основі описів сервісів; зв'язування описів сервісів з ітероперабельними мережними протоколами; зв'язок з високорівневими відкритими стандартами, службами й інструментами. Термін *архітектура* вказує на певний набір основних інтерфейсів, за допомогою яких можна конструювати ПС із заданими властивостями, а термін *відкрита* – розширюваність і нейтральність щодо виробників. OGSA полегшує відображення або адаптацію високорівневих функцій Grid-служб до апаратно-програмних засобів локальних платформ, а також забезпечує можливості багаторівневої віртуалізації ресурсів як усередині розподілених Grid-конфігурацій, так і усередині виконавчого апаратно-програмного середовища [8].

Основу служби складає *Мова опису Web-сервісу (Web Services Description Language – **WSDL),*** яка розширена конструкціями для підтримки множини інтерфейсів і керування змінами. WSDL дозволяє описувати угоди WSDL- інтерфейсів для Grid-служб, інформацію про поточний стан екземпляра сервісу, керування часом життя сервісу, сертифікатами і віртуалізацією.

Інтегратори IT-систем об’єднують розподілені комп'ютерні ресурси і керують робочим потоком розподіленого комп’ютинга, а також відстежують доступність і продуктивність координаційних робот, пов’язаних з внесенням коректив інформації на різних платформах системи. Провайдери послуг (service providers – SPs) надають такі послуги як Веб-хостинг, розподіл інформаційного наповнення, доступ до застосувань і систем зберігання даних шляхом використання стандартних процесів електронного бізнесу (наприклад, створення Web-порталів для численних користувачів).

Ці служби будуть додаватися новими сервісами, орієнтованими на забезпечення загальних функцій перетворення типів даних , що передаються між програмами, розташованими на різних платформах системи Grid**.**

Система збірки й обчислення в Grid. Середовище Grid (Collection Instuments and Tools) включає: платформи Intell, Sun, IBM, Apple, MS.Net, Кластери Мови 4-5 GPL C, C++, Visual C++, Visual Basic, Smalltalk,Java, LabView, Perl, Java, Pyton; зв’язки RPC, Active, RMI, CRL, SiDL, API, інтерфейсний, конфігураційний файли, dll-data. Засоби OGSA, SDK, Protocol NFS, User Domain and Application, HTTP, Model Interconnection, MS Foundation, репозиторії ресурсів [1, 6].

GDT ISSN 0372-6436 включає їхній опис зі зверненнями до різних компонентам за таких умов: - кожен компонент має властивості (ім’я, ліцензію, URL, репозиторії), глобальний унікальний ідентифікатор - GUID,

команди перевірки скомпільованого компонента, тестові команди, зв'язки з конфігураційним файлом версії системи; – конфігураційний файл окремого компонента чи системи містить інформацію про версію, зв'язки з репозиторієм, GUID, про види платформ і зв'язків із глобальними компонентами і пакетами для обчислень; – між двома різними конфігураціями встановлено статичну й динамічну залежність даних і компонентів.

У середовищі Grid на першому плані ресурси в широкому розумінні, а саме: динамічне керування застосуваннями й сервісами в кожний момент часу при глобальному обчислюванні наукових завдань. Ресурси – це різні програмні артефакти, засоби комунікації, інформаційні системи, СПС, системи збереження даних, СКБД, програмні фонди із різних доменів, технічні, комп'ютерні, людські ресурси тощо. Ресурс подає логічний файл або фізичний об'єкт (кластер, комп'ютер, суперкомп'ютер тощо). Їхні протоколи (NFS) завантажуються системою Grid і забезпечують моніторинг, керування, доступ до локальних ресурсів фабрики програм з метою отримання інформації про безпеку, захист, взаємозв'язки тощо.

Глобальні задачі в Grid взаємодіють між обчислювальними вузлами мережі через протоколи (Resource, Connectivity) при виконанні пакетів завдань. На основі стандартних протоколів будують сервіси, інтерфейси в API, засоби розроблення систем SDK (Software Development Kits). Зборку програм на канальному рівні з різних фабрик, а також виконання компонентів, підсистем і систем наукового призначення в різних МП реалізують глобальні протоколи (Global Protocols). Тобто ця мережа зорієнтована на майбутні обчислення з побудованих різними науковцями ПП. Крім розглянутих видів фабрик є й інші: інтегроване середовище Eclipse; система Oberon, яка з 90-х рр. виконує зборку різнорідних програм і підключила на цей час нові МП для опису й виготовлення ПП на комерційній основі; а також поповнення цього середовища новою мовою наукового інтерфейсу SIDL (Scientifically IDL) для забезпечення взаємодії наукових програм у мовах C, C++, Pyton, Java через прошарок типу glue для платформ Linux, AIX, Solaris, IBM з орієнтацією названої мови на майбутнє використання в інфраструктурі проекту Grid.

Усі дані в системі Grid накопичуються і підтримуються системою MySQL, засоби роботи з якими розглядаються далі.

3. Підходи до реалізації типів даних

У роботі [5] запропоновано нове фундаментальне рішення завдань передачі даних по мережі, суть його полягає у розробленні оригінальної системи генерації GDT загальних типів даних із FDT сучасних МП, на яких описують програми для фізичних, біологічних та інших наукових завдань і експериментів. Базовою підтримкою цього рішення є стандарт ISO/IEC 11404–2007 (General Data Types), який орієнтований саме на генерацію загальних типів даних до фундаментальних.

3.1. Загальна характеристика типів даних FDT і GDT

Основні положення системи генерації GDT<=>FDT викладені у стандарті ISO/IEC 11404–2007 GDT [6]. Розглянемо їх.

Фундаментальні типи даних – FDT визначають типи, операції над їх значеннями і форми подання їх на комп'ютері. Тип – це математичне поняття, що позначає множину значень елементів. Базовий тип є елементарним типом (ціле, дійсне та ін.), значення якого визначається апаратурою, компіляторами програм з МП та ін. Тип присвоюється змінній у програмі з МП, що визначає клас значень, кожне із яких належить одному і тільки одному типу. Операції над значеннями типу – це аксіоми, що перетворюють відображення значень одного типу в значення іншого типу. ФДТ включають прості, структурні та складні дані, множину операцій і значень типів даних, їх властивості та зв'язки з іншими типами даних. Прості типи – це перераховані та числові, структурні – це масиви, записи тощо; складні – це множини, об'єднання, динамічні об'єкти даних, списки, послідовності, стеки та ін.

Типи даних призначені для опису функцій та програм в МП. Вони реалізуються системами програмування на різних платформах комп'ютерів у вихідний код, який є джерелом не лише для виконання програми на цій МП, але й для забезпечення взаємодії у різних сучасних середовищах, коли вони відрізняються між собою. Кожна реалізована програма відображає тип даних конкретної МП, значення якого передається іншій програмі шляхом виклику (звернення) для обчислення.

Типи даних загального призначення стандарту GDT (ISO/IEC 11404–2007) за номенклатурою та семантикою наборів типів даних, що найчастіше використовуються в МП та інтерфейсах ПС мають такий вигляд.

LI-типи даних (*LI* – Language Independed), які незалежні від МП, вони специфіковані, як примітивні (базові, незалежні від інших) типи даних і не примітивні, що повністю чи частково визначені через інші типи даних і становлять класи типів даних, реальні представники яких використані у МП, що ґрунтуються на концепції FDT. Механізмом звернення є виклик процедур із завданням інтерфейсу до кожного стандартного сервісного засобу або деякої системи.

Типи даних стандарту створюють простір значень – сукупність (колекцію) значень деякого типу, що визначається одним із наступних способів: перелічення; аксіоматичне згідно основних положень; підмножиною вже визначеного простору значень; комбінацією будь-яких значень вже визначеного простору

значень шляхом процедур конструювання нових значень. Кожне окреме значення належить тільки одному типу даних, хоча воно може належати і декільком підтипам цього типу даних.

Модель типів даних є обчислювальною абстрактною моделлю. Вона має справу із властивостями одиниць інформації для визначення характерних операцій щодо типів даних і їх генераторів, типи даних яких можуть бути поданні у комп'ютерні системи.

Типи даних можуть належать одному сімейству, якщо існує заміна, яка перетворює весь простір значень одного типу даних (domain) у підмножину (діапазон) простору значень іншого типу даних таким чином, щоб значення відношень і характеристичних операцій з домену і діапазону зберігалися.

Опис типів даних стандарту GDT

Стандарт GDT включає такі типи даних.

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 повинні розрізнятися принаймні з цією точністю в 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 (колекцію) 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* ≠ *b*);
- не існує пари таких значень (*a, b*) з простору значень, для яких одночасно виконуються умови *a* = *b* і *a* ≠ *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* ≤ *b* або *b* ≤ *a*, чи обидві ці умови;
- для будь-яких двох значень (a, b), якщо *a* ≤ *b* і *b* ≤ *a*, то *a* = *b*;
- для будь-яких трьох значень (*a, b, c*), якщо *a* ≤ *b* і *b* ≤ *c*, то *a* ≤ *c*.

Запис *a* < *b* визначає нотації відношень: *b* ≤ *a* і *a* ≠ *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* ≤ *a*, і false у протилежному випадку.

Простір значень ґрунтується на математичній концепції з численності або кардинальності (cardinality), тобто він може бути скінченим, зчисленним нескінченним (зліченим) або незчисленним нескінченним. Тип

даних повинен мати кардинальність (потужність) свого простору значень за категоріями ТД, простір значень яких такий:

- скінчений;
- точний (exact) і зчисленний нескінченний;
- наближений має скінчену або зчисленну нескінченну модель, хоча концептуальний простір значень може бути незчисленно нескінченим.

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 М – 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 М і таке дійсне значення *h*, що $P(x) = v'$ для усіх *x* з М та $|v - x| < h$.

Таким чином, *v'* – це наближення у *C* для усіх значень з *M*, які знаходяться у *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*.

Нові механізми генерації типів даних GDT у стандарті

Стандарт дає опис декількох генераторів типів даних GDT, а саме.

Генератор типів даних (datatype generator) – це концептуальна операція над одним або декількома типами даних, яка створює новий тип даних. Він формує генерований чи параметричний тип даних. Генерованим типом даних є агрегатний тип даних, кожне значення якого отримано зі значень параметричних типів даних і мають назву компонента типу даних**.** Їх значення відрізняються між собою властивостями й відношеннями між кожним компонентом і агрегатним значенням.

Генеровані типи даних (generated datatypes) – це типи даних, які отримуються в результаті застосування генератора типів даних, як операції для створення нового типу з одного або декількох типів даних, а не для генерації значень. Генерований тип даних семантично залежить від різних параметричних типів даних і має власні характеристичні операції. Стандарт включає наступні генератори типів даних: вибір (choice), покажчик (pointer), процедура (procedure), запис (record), набір (set), портфель (bag), послідовність (sequence), масив (array), таблиця (table) тощо.

З практичної точки зору типи даних генеруються за допомогою спеціального набору процедур (функцій), які треба розробляти для використання їх у різних комп'ютерних системах. Пропонується нова схема генерації GDT<=>FDT .

Основні функції для практичної генерації типів даних GDT

Відповідно спроектованої нами схеми генерації необхідно розробити набір бібліотек функцій (процедур) у загально прийнятій мові XML для застосування їх при відображенні різних типів даних у програмах на сучасних або майбутніх МП:

функцій перебудови типів даних $МП_1$, …, $МП_{n;}$

функцій подання типів даних ФДТ;

функцій представлення GDT для оброблення з апробованої схеми ФДТ;

функцій відображення GDT<=>FDT.

Теорія подання ФДТ була розроблена з прикладами опису для класу МП 4GL Для реалізації зазначеного набору функцій у науковому проекті необхідно провести:

1) створення бібліотек функцій для перетворення типів даних GDT (примітивних, агрегатних і генерованих) до FDT типів даних (простим, структурним і складним) МП, як необхідних елементів середовища взаємодії різномовних компонентів, підсистем і проектів системи Grid;

2) специфікацію зовнішніх типів даних компонентів, підсистем і систем в МП засобами мови GDT із накопиченням їх в одному з репозиторіїв середовища фабрики;

3) розробка формату нових посередників подібно stub з операціями звертання до відповідних функцій GDT<=>FDT з метою передачі взаємодіючому компонентові й зворотно не релевантних і перебудованих типів даних.

Таким чином, проблема перебудови типів даних потребує системної реалізації запропонованих бібліотек і програм генерації посередників, які перебудовують формати даних, які створюються на різних архітектурах комп'ютерів.

3.2. Підходи до перетворення даних, які зберігаються в БД

Проблема перетворення даних має місце при використанні різних 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 [9, 10].

Перетворення даних БД. Перетворення даних в БД пов'язане з різницею логічних структур даних (ієрархічні, мережні, реляційні) у 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 у БД та концептуальну модель даних. Внесені зміни відображаються у 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 типа, 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 і належали до реляційної моделі.

Про перебудову даних БД

Як приклад розглянемо підхід до перетворення ТД, а саме строкових, двійкових, символьних тощо стосовно сучасної MSDN (Microsoft Developer Network) [1].

Ці дані мають різний опис (char, varchar, nchar, nvarchar, binary, varbinary 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 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(123456 AS BINARY(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 одержанні нового типу даних. Якщо символьне вираження перетвориться в символьне вираження іншого типу даних або розміру, наприклад з char(5) в varchar(5) або з 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, перетворені 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 в системі SQL Server 2008.

Висновки

Дослідження проведено в межах фундаментальних проектів ІІІ-1-07 ІПС НАНУ з вирішення проблем генерувального програмування [10] та засобів подання і обробки даних у проекті стосовно сучасної системи Etics в межах Європейського проекту Grid, що створений для організації обчислень наукових задач e-sciense. Основу обробки даних створює інтерфейс, який забезпечує зв'язок між різномовними програмами задля його специфікації спеціальною мовою IDL (Interface Definition Language), згідним для всіх видів програм наукового типу в МП. Основним и елементами інтерфейсу є дані простої і складної структури, якими обмінюються між собою програми.

Розглянуто інтерфейс з даними, який необхідний для завдання зв'язків різнорідних компонентів в мові IDL. В МП використовуються типи даних фундаментального і загального типа. Ці типи даних формально подані у стандарті GDT (ISO/IEC 11404–2007), а також метод генерації загальних до простих фундаментальних типів даних. Цей механізм необхідний тоді, коли програми обмінюються різноформатними даними і деякі потребують перетворення один до іншого для спрощення обчислень. Розглянути у даної роботі типи даних і які пропонуються цим стандартом орієнтовані на побудову нових засобів, механізмів і функцій обробки різнорідних даних.

Запропонований підхід до перетворення фундаментальних і загальних типів даних між собою дуже важливий при підготовки програм, систем і пакетів, що будуть обчислюватися в такому розподіленому середовищі як Grid.

Реалізація цього підходу додасть додаткові можливості по обробленню даних не тільки для середовища Grid, що включає до свого складу систему обробки даних MySQL, але і для нового напряму Cloud Computing з новітніми засобами зберігання даних у різноформатних сховищах даних.

1. *Стеняшин А.Ю.* Про формальний опис типів і структур даних в різнорідних программах // Проблеми програмування. – 2011.– №2. - С. 50–61.
2. *Лаврищева Е.М.* Интерфейс в программировании // Проблеми програмування.– 2007. – № 2. – С. 126–139.
3. *Лаврищева Е.М.* Проблема интероперабельности разнородных объектов, компонентов и систем. Подходы к ее решению // Матер. 7 Міжнар. конф. з програмування "УкрПрог-2008". 2011.– №2-3. – С. 28–41.
4. *Стеняшин А.Ю.* Про формальний опис типів і структур даних в різнорідних программах // Проблеми програмування. – 2011.– №2. - С. 50–61.
5. *Таковицкий О.* Технология Grid computing. – 2012. – С. 1–9.
6. *Андон П.І., Лавріщева К.М.* Розвиток фабрик програм в інформаційному світі // Вісник НАН України. – 2010.– № 10.– С. 15–41.
7. *Стеняшин А.Ю.* Особливості мовного підходу до опису моделей доменів // Тези матер. "VIII Всеукр. наук.-метод. конф.".-Київ, 24-25травня 2011р. – С. 101–106.
8. *Лаврщева Е.М., Грищенко В.М.* Сборочное программирование. Основи индустрии программных продуктов. – Второе изд. – Киев.: Академпериодика, 2009. – С. 371.
9. *Лавріщева К.М.* Генерувальне програмування програмних систем і сімейств // Проблеми програмування. – 2009.– № 1. – С. 3–16.