ПРИМЕНЕНИЕ ТЕОРИИ ОБЩИХ ТИПОВ ДАННЫХ СТАНДАРТА ISO/IEC 11404 GDT К BIG DATA

Е.М.Лаврищева,
гл.н.с. ИСП РАН, д.ф.-м.н., профессор МФТИ, Москва
Рыжов А.Г. ,
н.с. ИСП РАН, Москва
email: lavrischeva@gmail.com and lavr@ispras.ru, ryzhov@ispras.ru

АННОТАЦИЯ. Рассматриваются вопросы применения теории общих типов данных (GDT) к Большим Данным (Big Data). Подан базовый аппарат представления типов данных (ТД) GDT (стандарта ISO/IEC 11404-2007) и теория генерации нестандартных ТД GDT. Определены операции и набор функций трансформации данных с одной платформы на другую. Предложен подход к анализу неструктурированных данных и генерации с помощью функций, аналогичных функциям библиотеки CTS (Common Type System) VS.Net.

Ключевые слова: общие типы данных; простые, сложные, генерируемые, неструктурные данные; трансформация; генерация; большие данные; библиотеки функций.

APPLICATION OF THE THEORY OF GENERAL DATA TYPES STANDARD ISO/IEC 11404 GDT TO BIG DATA

E.M. Lavrischeva,
main scientist of the Russian Academy of Sciences, D. SC. PhD, Professor MIPT,
A.G. Ryzhov,
B.S. ISP RAS, Moscow,
email: lavryscheva@gmail.com and lavr@ispras.ru, ryzhov@ispras.ru

ABSTRACT.
Discusses the application of fundamental theory (FDT) and the General data types (GDT) to Big Data (Big Data). Filed basic machine representations of data types (TD) FDT and generate complex TD GDT (ISO/IEC 11404-2007) to simple. The following is the description format is different so programs in programming languages (PL). The formal theory of transformation of structural GDT to a simple Data PL. The operations of data exchange and transformation from one platform to another. Developed a set of transform functions standard GDT to simple data that can be generated from different devices and from data stores. The proposed approach to the analysis of unstructured data and generate the individual elements of these structures to more simple data using library functions CTS (Common Type System) VS.Net.

Key words: common data types, simple, complex, generated, non-structural, transformation, generation, big data, library functions.

Вступление

На данное время в компьютерном информационном пространстве накоплено большое количество разнородных программ, которые используются в разных средах для вычисления физических, биологических и других задач. Компоненты программ обмениваются между собой данными, описанными в ЯП. Общие ТД GDT могут поступать как пространственные зрительные образы, отчеты и наборы данных, генерируемых с разных датчиков или специализированной аппаратуры. Такие данных относятся к классу больших данных и при вычислениях требуют нестандартных методов и приемов для их анализа, обработки и организации вычислений [1-3].

Тип данных – это фундаментальное понятие в программировании, которое задает множество значений и операций, применяемых к этим значениям и способам их хранения. Данные, которыми оперируют программы в ЯП реализованы во многих ЯП - Паскаль, Модула-2, Ада, C/C++ и др. Сложные данные в ЯП приводятся к более простым данным с помощью функций библиотеки CTS и вычисляются в современных средах (IBM, VS.Net и др.). Неструктурированные данные поступающие с разных приборов и аппаратуры при съемках недр земли, океана и космоса, образуют Большие Данные огромных размеров. Предлагается подход к применению теории GDT для анализа наборов неструктурированных данных Больших Данных, их представления к виду таблиц описаний данных GDT с семантическими функциями анализа и трансформации каждого отдельного ТД форматам данных платформы выполняемой среды [4-9].

Под *трансформацией* данных будем понимать методы формального отображения типов форматов данных одного компьютера к соответствующему представлению другого компьютера., а также методы установления отличий в представлении типов данных в разных ЯП и методы преобразования данных при замене БД. Данные могут быть представлены в:

– стандартной кодировке (XDR – eXternal Data Representation, CDR – Common Data Representation, NDR – Net Data Representation), требующей их трансформации;

– ЯП и в языке описания интерфейсов RPC (CORBA, DCOM, Google ProtoBuf, Apache Thrift, Apache Avro) и REST (REST API), которые обес-

печивают передачу данных между программами [9-12].

К методам трансформации форматов данных относится кодироване и декодирование данных, линеризация сложных структур для расположения данных в передающей и в принимающей платформе компьютера.

Связь разнородных объектов осуществляется с помощью интерфейсных функций преобразования ТД в системах CORBA, DCOM, Google ProtBuff, ApacheThrift, Apache Avro и др. [9-11].

Проблему взаимосвязи объектов на разных ЯП (Java, C/C++, Smaltalk, Cobol, Ada-95 и др.) обеспечивает брокер ORB OMG с помощью языков IDL и CDR для описания посредников (stub, skeleton) и протокола передачи данных GIOP [14]. В CORBA реализована объектная модель для взаимодействия программ в среде клиент–сервер. В этой модели интерфейсы отделены от реализаций, определяются в терминах языка IDL, независимо от конкретных ЯП и включают в себя полную сигнатуру операций (имя объекта, имя метода, типы передаваемых параметров и тип возвращаемого результата). Спецификация посредника в языке IDL близка средствам описания классов в C++ и отображается в ЯП клиентских и серверных объектов. Формальное преобразование ТД осуществляется с помощью алгебраических систем для каждого типа данных

$T\alpha^t$: $G\alpha^t = <X\alpha^t, \Omega\alpha^t>$,

где t – тип данных (real, integer, boolean, chart, array, set и др.); $X\alpha^t$ – множество значений этих ТД, которые они могут принимать; $\Omega\alpha^t$ – множество операций над этими ТД [11].

1. Типы данных стандарта GDT ISO/IEC 11404

К общим ТД GDT (General Data Types) стандарта ISO / IEC 11404-2007 относятся [1, 4-10]:

- независимые от языка (Independed Language) ТД, которые используются для формального описания концептуальных данных, их элементов и значений;
- полуструктурированные и неструктурированные совокупности данных, в которых ТД являются неизвестной или неопределенной заранее структурой данных;
- расширяемые ТД GDT.

Стандарт GDT устанавливает номенклатуру и семантику наборов ТД, которые используются в ЯП и в интерфейсах программных систем (ПС). В этом стандарте специфицированы базовые ТД и сложные, которые полностью или частично определяются с помощью простых ТД.

Термин «независимый от языка» ТД означает, что специфицированные ТД образуют классы, представители которых в ЯП соответствуют общей концепции ТД стандарта GDT ISO/IEC 11404 и частично совпадают с фундаментальными ТД ЯП:

- примитивные ТД (real, integer, char, boolean ...),
- сложные ТД (массив, запись, последовательность, портфель, ...),
- сгенерированные ТД с помощью генератора стандарта,
- генератор новых ТД.

Примитивные ТД GDT

Рациональный (rational) – математический ТД, который соответствует действительным числам.

Масштабированный (scaled) – это семейство ТД, пространством значений которого является подмножество рациональных чисел, а каждый отдельный ТД имеет фиксированный знаменатель и предполагает аппроксимацию его значений.

Комплексный (complex) – это семейство ТД, каждый из которых задает числовой математический тип данных для комплексных чисел.

Пустой (void) – это тип данных, который задает объект с необходимыми синтаксическими и семантическими описаниями и не несет никакой информации.

1.1. Основные положения GDT

Пространство значений – это совокупность (коллекция) значений типа данных, которая определяется одним из следующих способов:

1) перечисление;

2) аксиоматическое определение;

3) подмножество пространства значений с набором свойств;

4) комбинация любых значений для уже определенного пространства значений с помощью процедуры конструирования новых значений.

Каждое отдельное значение принадлежит только одному ТД, хотя оно может принадлежать и нескольким подтипам этого ТД.

Равенство. Для каждого пространства значений существует понятие равенства (equality), задаваемого следующими аксиомами.

Аксиома 1. Для любых двух значений (a, b) из пространства значений выполняется условие равенства b, специфицированное как $a=b$, или не равняется b, специфицированное как $a \neq b$;

Аксиома 2. Не существует пары таких значений (a, b) из пространства значений, для которых одновременно выполняются условия $a=b$ и $a \neq b$;

Аксиома 3. Для каждого значения a из пространства значений выполняется условие $a=a$;

Аксиома 4. Для любых двух элементов значений (a, b) из пространства значений $a=b$, тогда и только тогда, когда $b=a$;

Аксиома 5. Если для произвольных трех элементов значений (a, b, c) из пространства значений выполняются условия $a=b$ и $b=c$, то тогда $a=c$.

Для каждого ТД операция равенства *Equal* определяется как свойство равенства пространства значений. Для любых значений a и b из пространства значений Equal (a, b) есть *true*, если $a=b$ и *false* в противном случае.

Порядок. Пространство значений упорядочено, если для него установлено отношение порядка

(order), которое задается знаком меньше или равно (≤) и удовлетворяет правилам:

1) для каждой пары значений (*a, b*) из пространства значений выполняется условие $a \leq b$ или $b \leq a$ или оба эти условия;

2) для любых двух значений (*a, b*), если $a \leq b$ и $b \leq a$ то $a=b$;

3) для любых трех значений (*a, b, c*), если $a \leq b$ и $b \leq c$, то $a \leq c$.

Запись $a < b$ используется для нотации: $a \leq b$.

Тип данных упорядочен, если отношение порядка определено на пространстве значений. Тогда операция InOrder определяется для произвольных двух значений *a* и *b* из пространства значений InOrder(*a, b*) как true, если $b \leq a$, и false в противном случае.

Ограниченность. ТД ограничен сверху, если он упорядочен и существует такое значение U из его пространства значений, при котором для всех значений *s* этого пространства выполняется условие $s \leq U$. Значение *U* образует верхнюю границу пространства значений. Аналогично, ТД ограничен снизу, если он упорядоченный и существует такое значение *L* из его пространства значений, что для всех *s* этого пространства выполняется условие $L \leq s$. Значение *L* образует нижнюю границу пространства значений. ТД называется ограниченным, если его пространство значений имеет верхнюю и нижнюю границу.

Кардинальность. Пространство значений основывается на математической концепции кардинальности (cardinality): конечное или бесконечное. ТД должен иметь кардинальность (мощность) своего пространства значений. Предусмотрены три категории ТД, пространство значений которых:

1) конечное;

2) точное (exact) и бесконечное;

3) приближенное и имеет конечную или бесконечную модель, концептуальное пространство значений которой может быть бесконечным.

Точный и приближенный. Если каждое значение в пространстве значений концептуального типа данных можно отличить от другого значения в пространстве этой модели, то ТД считается точным (exact).

Математические ТД, которые имеют значения и не имеют определенного представления, называются **приближенными** (approximate). Пусть *M* – математический ТД, а C – соответствующий вычисляемый ТД, *P* – преобразователь пространства значений *M* в *C*. Тогда для каждого значения *v′* с *C* существует соответствующее значение типа данных v с *M* и такое действительное значение *h*, что $P(x)=v'$ для всех x с *M* и $|v-x|<h$. Таким образом, *v′* – это приближение в *C* для всех значений *M* и находится в *h*-области значение *v″*. Кроме того, по крайней мере, для одного значения *v′* в *C* существует более чем одно такое значение в *M* такое, что $P(y) = v'$. Вывод, *C* не является точной моделью *M*.

Числовой. ТД называется числовым (numeric), если его концептуальное значение определяется количественно (в системе нумерации). ТД, значение которого не имеет этого свойства, называется нечисловым (non numeric).

Пространство значений базируется на математической концепции или свойстве кардинальности (cardinality), то есть оно может быть конечным или бесконечным. Тип данных должен иметь кардинальность (мощность) своего пространства значений по категориям ТД, пространство значений которых может быть: конечным, точным (exact), бесконечным и приближенным. Каждый концептуальный ТД являются точным. Невычислимый ТД является бесконечным. Если каждое значение концептуального ТД отличается от другого значения этой модели, то тип данных считается точным (exact)**.**

1.2. Сгенерированные типы данных GDT

Сгенерированные ТД (generated datatypes) – это ТД, полученные в результате генерации типов данных. ТД, с которым работает генератор, называется **параметрическим** или **компонентным.** Сгенерированный ТД семантически зависит от параметрических ТД, но имеет собственные характеристические операции. Важной характеристикой всех генераторов ТД является то, что генератор может применяться к разным параметрическим ТД. Генераторы указателя и процедуры дают ТД, значения которых атомарные, тогда как генератор Выбора и агрегатных типов данных выдает ТД, значения которых позволяют производить их декомпозицию.

Генератор ТД (datatype generator) – это концептуальная операция над одним или несколькими ТД, которая создает новый ТД. Генератор ТД оперирует с типами данных, а не с его значениями и представляет собой:

1) набор критериев для характеристик ТД, над которыми будут выполнены операции;

2) процедуры конструирования, которые допускают набор ТД с данным критерием для создания нового пространства значений из пространств значений этих ТД;

3) набор характеристических операций, которые применяются в конечном пространстве значений для завершения определения нового ТД.

Агрегатный ТД (aggregate datatype) – это сгенерированный ТД, каждое значение которого получено из значений параметрических ТД. Параметрические ТД агрегатного ТД или его генератор включают в себя имена компонентов ТД. Генератор агрегатного ТД выдает ТД с помощью алгоритмической процедуры в пространстве его значений.

В отличие от других сгенерированных ТД агрегатный ТД обеспечивает доступ к компонентам значений через характеристические операции. Агрегатные значения разных типов различаются между собой свойствами, которые задают отношение между компонентами ТД и между каждым компонентом и агрегатным значением.

Сложные типы данных GDT и генераторы ТД

Множество (set) задает ТД, пространство

значений которого составляет набор всех поднаборов пространства значений. Операции соответствуют математическому множеству *set.*

Стандарт включает генераторы ТД сложных типов данных: выбор (choice), указатель (pointer), процедура (procedure), запись (record), набор (set), портфель (bag), последовательность (sequence), массив (array), таблица (table) и т.п.

Выбор (choice) генерирует ТД. Каждое значение образуется из любого набора альтернативных ТД. Этот ТД учитывает их соответствие значению другого типа данных с признаком (tag).

Указатель (pointer) генерирует ТД, каждое значение которого устанавливает средства ссылки на значение другого типа данных, специфицированного типом данных *element-type*. Эти значения типа данных указателя - атомарные.

Процедура (procedure) генерирует ТД, значение которого является значением других ТД, называемых **параметр**. Такой ТД включает в себя набор всех операций над значениями конкретной коллекции ТД, концептуально атомарных.

Запись (record) генерирует ТД, значение которого составляет совокупность значений компонентов ТД и каждая совокупность имеет значение для каждого компонента типа данных, специфицированного фиксированным идентификатором поля field-identifier.

Набор (set) генерирует ТД из пространства значений поднаборов пространства значений элемент с операциями, свойственными математическому множеству *set.*

Портфель (bag) генерирует ТД, значения которого составляют коллекции образцов значений типа данных элемент. Многочисленные образцы того же значения могут подаваться в этой коллекции, а порядок их в коллекции - несущественный.

Последовательность (sequence) генерирует ТД, значениями которого являются упорядоченные последовательности значений типов данных из значений, несвойственных этому типу данных; одно и то же значение может встречаться многократно в этой последовательности.

Массив (array) генерирует ТД, значения которого ассоциируются с произведением пространств одного или нескольких конечных ТД, которые называются **индексными ТД**. Пространство значений этого ТД такое, что каждому значению из пространства индексного типа данных соответствует только одно значение элемента.

Таблица (table) генерирует ТД, значение которого составляют коллекции значений из пространства одного или нескольких типов данных как поле, такое что каждое значение задает ассоциации между значениями его полей.

Объявленный ТД (defined) – это ТД, определенный посредством объявления типа type-declaration, который задает идентификатор некоторого объявленного типа и ссылается на ТД или генератор ТД, определенным как Actual-type-parameters, если он соответствует номеру и типу объявления type-declaration. Type-declaration идентифицирует в type-reference как один ТД, семейство типов данных или генератор типов данных.

Характеристические операции создают значение любого типа с помощью генератора ТД в пространстве значений параметрических ТД. Такие операции необходимы для выделения ТД по их названиям и генерации агрегатных ТД как композиции следующих операций:

1) с нулевой арностью генерируемых значений этого ТД;

2) с унарной операцией (арности 1), которая превращает значение этого ТД в новое значение этого ТД или в значение boolean;

3) с арностью 2, которые преобразуют пары значений этого ТД в значение этого ТД или в значение boolean;

4) с n-арностью, преобразующей упорядоченные n-элементные группы значений, каждая из которых относится к определенному ТД и может быть параметрическим типом или агрегатным.

Практически не существует уникальной коллекции характеристических операций для заданных ТД. Одна коллекция операций ТД (или генератора типов), достаточна для выделения этого ТД среди других из пространства значений той же мощности.

Таким образом, существует посимвольная замена, которая преобразует все пространство значений одного ТД (domain) в подмножество значений пространства другого ТД (диапазон, range**)** так, чтобы значение отношений и характеристических операций сохранялись бы в соответствующих значениях отношений и характеристических операций диапазона ТД.

1.3. Преобразование ТД ISO/IEC 11404-96

Стандарт определяет LI–язык, который преобразует ТД независимо от ЯП, и включает следующие виды преобразований:

– внешнее преобразование внутренних ТД ЯП в LI–типы данных;

– внутреннее преобразование LI–типа данных в ТД ЯП;

– обратное внутреннее преобразование к внешнему.

Суть *внешнего преобразования* ТД генераторов ТД состоит в следующем:

а) для каждого примитивного типа для сгенерированного внешнего типа данных преобразование устанавливается связь с одним LI–типом данных;

в) для каждого внутреннего типа данных преобразование задает связь между допустимым его значением и эквивалентным значением соответствующего LI–типа данных;

с) для каждого значения LI–типа данных определяется значение любого внутреннего типа данных, преобразуемого в LI–тип данных его значения.

Внутреннее преобразование задает связь примитивного ТД или сгенерированного в LI–тип данных с внутренним ТД ЯП. Данное преобразование обладает следующими свойствами:

а) для каждого LI–типа данных (примитивного или сгенерированного) преобразование определяет наличие этого типа данных в ЯП;

в) для каждого LI–типа данных преобразование определяет отношение между этими типами и эквивалентным значением соответствующего внутреннего типа ЯП;

с) для каждого значения внутреннего типа данных преобразование определяет является ли это значение образом какого–то значения LI–типа данных.

Обратное внутреннее преобразование LI–типа данных состоит в преобразовании значений внутреннего ТД в соответствующее значение LI–типа при наличии соответствия и отсутствия двусмысленности. Это преобразование для ЯП является коллекцией обратных внутренних преобразований LI–типа данных.

1.4. Генерация ТД стандарта GDT

Разработана схема генерации ТД GDT в структуры фундаментальных ТД ЯП. Согласно (рис.1) предлагается разработать библиотеку функций генерации ТД GDT, элементы которой для всех новых ТД этого стандарта выполняют следующие виды операций:

- преобразование ТД, содержащихся в ЯП ($ЯП_1$, …, $ЯП_n$) и которые входят в состав фундаментальных ТД ЯП [1, 4];

- функции трансформации сложных ТД GDT, включенных в стандартную библиотеку CTS VS.Net и используются трансляторами с ЯП этой системы для преобразования сложных ТД к более простым;

- операции взаимодействия компонентов повторного использования, записанных в разных ЯП, интерфейс которых задается в языке IDL.

Все ТД GDT представлены в виде следующих классов алгебраических систем [11]:

$\Sigma_1 = \{G_\alpha^b, G_\alpha^c, G_\alpha^u, G_\alpha^r\}$,

$\Sigma_2 = \{G_\alpha^a, G_\alpha^z, G_\alpha^u, G_\alpha^e\}$,

$\Sigma_3 = \{G^s, G^{ns}\}$, (1)

где $G_\alpha^t = <X_\alpha^t, \Omega_\alpha^t>$, t – ТД языков L, X_α^t – множество значений ТД, Ω_α^t – множество операций над ТД; Σ_1 - алгебраическая система простых ТД, Σ_2 - алгебраическая система сложных ТД, Σ_3 – алгебраическая система неструктурированных данных.

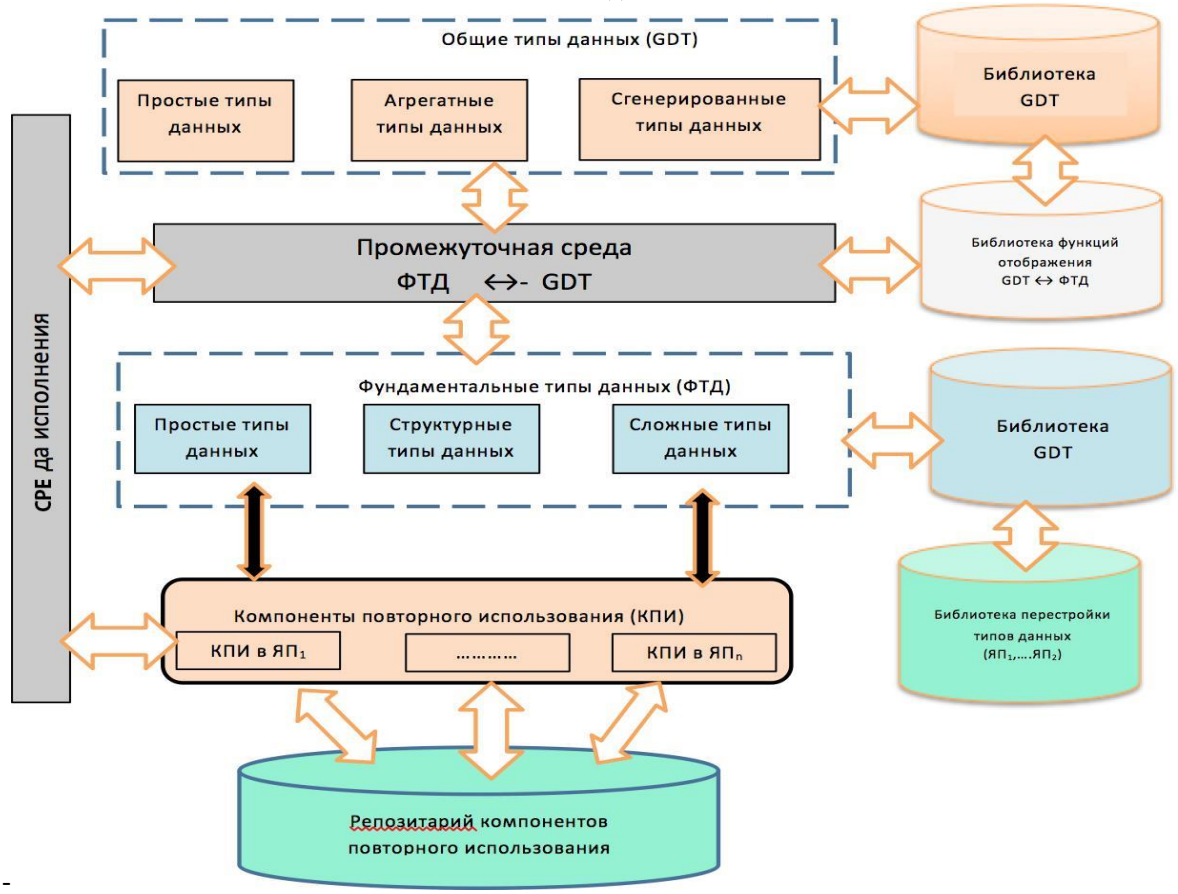

Рис.1. Схема трансформации GDT

В каждом классе этих систем преобразование $t \to q$ для пары языков l_t и l_q основано на таких свойствах отображений:

1) системы G_α^t и G_β^q – изоморфны, если их q, t определены на одном и том же множестве ТД;

2) между значениями X_α^t и X_β^q типов данных t, q существует изоморфизм, если множество операций Ω_α^t и Ω_β^q разные.

3) если множество $\Omega = \Omega_\alpha^t \cap \Omega_\beta^q$ не пустое, то имеет место изоморфизм двух систем $G_\alpha^{t'} = < X_\alpha^t, \Omega>$ и $G_\beta^{q'} = < X_\beta^q, \Omega>$.

4) Если типы данных отличаются, например, t - строка, а тип q – вещественное, то между множествами X_α^t и X_β^q не существует изоморфного со-

ответствия.

Отображения сохраняют линейный порядок элементов к виду линейной упорядоченности элементов алгебраических систем из этих классов.

2. Неструктурированные данные больших данных

В результате проведенных нами исследований неструктурированных данных из класса больших данных (Big Data), рассмотрена применимость к ним стандартных ТД ЯП и GDT. Отличительной особенностью больших данных является то, что наборы данных поступают с разного рода датчиков исследования недр земли и воздушного пространства, с записями многочисленных страховых компаний, задаваемых разными изображениями, фотографиями, документами, а также сервисов и услуг Интернета и др. Большие данные (Big data) образуют: масштабируемые хранилища "ключ-значение" (Berkeley DB, Amazon DynamoDB); хранилище семейств колонок (разреженные матрицы с согласованием); документно-ориентированные СУБД с иерархическими структурами данных; базы данных на основе графов и с большим количеством связей (социальные сети) (Neo4j, Walmart, OrientDB и др.) [17].

2.1. Неструктурированные данные GDT – это совокупность данных, которые:

- структурированы в виде компонентных ТД или метода доступа;
- наполовину структурированные данные, которые имеют один ТД или метод доступа к значению;
- неструктурированные данные, которые включают набор данных неодинаковой природы.

Такие данные представляются в разных форматах: XML в пространственных зрительных образах и отчетах. Эти данные накапливаются как большие объемы неструктурированных данных. Они интегрируются с существующими приложениями, соответствуют нормативным требованиям и расширяют возможности для пользователей при работе с информационными системами. Для неструктурированных ТД еще не разработан стандарт формального описания больших данных, как это сделано для GDT. А термин «большие данные» относится к наборам данных, размер которых превосходит возможности типичных баз данных (БД) в плане хранения, управления и анализа информации. Он включает анализ больших объемов данных с целью структуризации информации и управления ею.

Согласно отчету McKinsey Institute большие данные представляют собой новый рубеж для инноваций, конкуренции и производительности. Мировые репозитории (библиотеки) данных постоянно растут. В отчете аналитической компании IDC (Digital Universe Study) «Исследование цифровой вселенной, 2011» и компании EMC, предсказывалось, что общий мировой объем созданных и реплицированных данных составляет около 1,8 зеттабайта (1,8 трлн. гигабайт) — примерно в 9 раз больше того, что было создано ранее. Такие данные предполагают больше, чем просто анализ огромных объемов информации, поскольку они в основном представлены в форматах, плохо соответствующих традиционным форматам БД. К ним относятся веб-журналы, видеозаписи, текстовые документы, машинный код или, например, геопространственные данные и др. Большие данные хранятся в разнообразных хранилищах. По прогнозам, количество данных на планете будет удваиваться каждые два года вплоть до 2020 года. Традиционные методы анализа информации не могут удовлетворять проблемам увеличения объемов данных, их накопления и обновления. Поэтому требуются формальные методы их представления в хранилищах для обработки.

2.2. Хранение неструктурированных данных

Приложения, основанные на работе с реляционными и нереляционными данными, в основном используют одну из трех архитектур:

- реляционные данные находятся в БД, а большие нереляционные данные двоичных объектов (BLOB), которые находятся в файловых системах или на файловых серверах;
- нереляционные данные из хранилищ, предназначенных для BLOB-данных;
- реляционные и нереляционные данные, которые находятся в БД.

Эти данные используются разными приложениями путем:

- создания, загрузки, обновления и удаления неструктурированных данных и использования транзакционной согласованности между источниками неструктурированных данных;
- индексирования неструктурированных данных и их поиска;
- извлечения метаданных в явной форме (например, из полей форм или из атрибутов файлов) и предоставление их пользователям;
- анализа и преобразования содержимого документов к форматам для выполнения поиска и составления запросов (например, преобразование звуковых файлов в текстовые и выполнение поиска по запросу БД).

Хранение неструктурированных данных в хранилищах проводится с помощью BLOB-данных. При хранении BLOB-данных в БД централизованного хранилища снижаются затраты и быстродействие.

Пример неструктурированных данных с датчика космических исследований (2012) дан в [13]:

{58} n_vers:byte; {номер версии программы записи информации}

{59-62} n_krit:longint; {номер критерия экспресс обработки}

{63-64} n_bad:integer; {кол-во кадров отбракованных экспресс обработкой}

{65-76} fam_fiz:array[1..12]of char; {фамилия дежурного физика}

{77-116} iskr_k:array[1..40] of byte; {информация с камеры}

{117-118} n_vved, {кол-во введенных кадров с разными номерами}

{119-120} n_good:integer; {кол-во записанных в первичный банк кадров} и др.

В журнале «Открытые системы» №3, 2016 рассмотрен ряд технических подходов, которые связаны с платформами Интернет, процессорами HPC, машинным обучением, с «умным» хранением Больших данных и др. С учетом проведенных экспериментов Больших данных в данной работе поставлена цель – провести анализ отдельных наборов Больших данных, определить составные элементы, сравнить их со стандартными структурами GDT и для некоторых их них предложить механизмы трансформации элементов данных с помощью имеющихся библиотечных функций CTS, CLR MS.Net или разработать новые функции согласно теории GDT.

3. Подход к обработке неструктурированных данных

3.1. Теоретический подход

При обмене данными проводится трансформация передаваемых данных и в случае несовпадения типов данных с форматами данных платформ вычислительной среды, проверяется количество передаваемых параметров и др. [11]. Задача взаимодействия пары разнородных программ состоит в установлении взаимно однозначного соответствия между множествами фактических параметров $V = \{v_1, v_2, ..., v_к\}$ и формальных параметров $F = \{f_1, f_2, ..., f_к\}$ программ (k и k1 могут отличаться). Для отображения множеств V и F требуется провести разбиение множеств таким образом, чтобы каждому подмножеству из V соответствовало только одно подмножество из F. Для каждого $f^i \in F$ рассматривается полный прообраз $V^i \in V$. Различные прообразы V^i и V^j могут иметь или не иметь одинаковые элементы.

Если $V^i \cap V^j \neq \Theta$, то объединяется V^1 и V^j в одно подмножество. Соответственно будет проведено объединение в одно подмножество элементов F^i и F^j. Данная процедура применяется до тех пор, пока не будет исчерпано множество V. В результате получается два семейства подмножеств.

$\Pi = \{V^1, V^2 \ldots, V^m\}$ и $\Phi = \{F^1, F^2, \ldots, F^m\}$ таких, что

$$\bigcup_{t=1}^{m} V^t = V,\ V^t \bigcap V^{t1} = \Theta \text{ при } t \neq t_1$$

$$\bigcup_{t=1}^{m} F^t = F,\ F^t \bigcap F^{t1} = \Theta, \text{ при } t \neq t \qquad (2)$$

для которых существует однозначное отображение, записываемое в виде

$$A: \Pi \rightarrow \Phi. \quad (3)$$

В зависимости от количества элементов во множествах V^t и F^t имеют место следующие случаи:

1) $|V^t|=|F^t|=1$. Отображение А для данных подмножеств включает операции преобразования типов данных.

2) $|F^t|>1$ и $|V^t|=1$. Это означает, что одному фактическому параметру структурного типа данных соответствует несколько формальных параметров скалярных типов или структурных с меньшим уровнем структурирования. Отображение А включает операции селектора отдельных компонентов и преобразования типов данных.

3) $|V^t|>1$ и $|F^t|=1$. Это означает соответствие нескольких фактических параметров одному формальному. Отображение А содержит операции преобразования типов и конструирование структурного типа с более высоким уровнем структурирования, чем у передаваемых параметров.

4) $|V^t|>1$ и $|F^t|>1$. Это свидетельствует о существовании глубокой связи вызывающего и вызываемого модулей, которая зависит от внутренней логики функционирования модулей. Такие связи противоречат свойствам модулей и не поддается формальному анализу при сборке модулей. На основе проведенного анализа свойств отображения А выделяются операции для информационного сопряжения модулей.

Операции трансформации ТД T_a^t в T_β^t имеют вид:

$$P^{tq}{}_{a\beta} = (T_a{}^t,\ T^q{}_\beta), \qquad (4)$$

где данные типа T_a^t преобразуются в $T^q{}_\beta$, a и β соответствуют языкам l_a и l_β. Предполагается, что множество типов данных каждого ЯП упорядочено и индексы t и q определяют конкретные элементы этого множества. Для ЯП, имеющих средства конструирования новых типов, t и q будут функциями от других индексов и упорядоченность типов может определяться тем, что новый тип t будет конструироваться из типов, для которых индексы не больше t. Каждый ЯП имеет определенное множество предопределенных ТД и базовых операций конструирования, что определяет основу всего множества типов. Новый тип будет иметь индекс, функционально зависящий от индексов предопределенных типов и конкретных операций конструирования.

2.2. Системные средства обработки сложных данных

В системе Microsoft.NET для решения проблем обработки данных в ЯП реализована система общих типов CTS (Common Type System, рис 2.) [14].

CTS – это стандартная система типов, которая включает описание всех ТД, и выполнения программ со строками, целочисленными данными и с плавающей точкой. Взаимодействие программ друг с другом осуществляет с помощью CLR (Common Language Runtime) MS.NET. Система осуществляет выявление и загрузку ТД, а также управление безопасностью и интеграцией разных программ в любых ЯП. CLS (Common Language Specification) позволяет специфицировать: классы, структуры, интерфейсы, типы, перечисленння, а также встроенные ТД (табл.1).

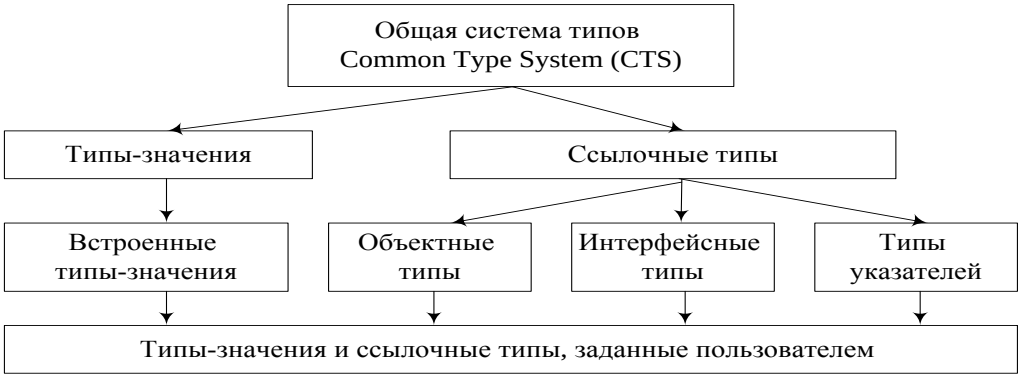

Рис.2. Система типов данных CTS в MS.Net.

В процессе трансляции программ в ЯП создается программа в DLL или EXE (сборщик – IL Assembler) на языке IL. Код IL не зависит от платформы, на которой будет осуществляться его выполнение. Промежуточный язык MSIL задает промежуточный уровень процесса взаимодействия кода на любых языках VS.NET. MSIL конвертируется в код CPU для разных архитектур.

Таблица 1.

Встроенные типы данных CTS в C# и C++/CLI.

CTS ТД	VS.Net	C#	C++/CLI
System.Byte	Byte	Byte	unsigned char
System.Sbyte	Sbyte	Sbyte	signed char
System.Int16	Short	Short	Short
System.Int32	Integer	Int	int or long
System.Int64	Long	Long	int64
System.UInt16	Ushort	Ushort	unsigned short
System.UInt32	UInteger	Uint	unsigned int or long
System.UInt64	Ulong	Ulong	unsigned _int64
System.Single	Single	Float	Float
System.Double	Double	Double	Double
System.Object	Object	Object	Object^
System.Char	Char	Char	wchar_t
System.String	String	String	String^
System.Decimal	Decimal	Decimal	Decimal
System.Boolean	Boolean	Bool	Bool

Компонентная модель MS.Net реализует проектирование приложений методом сборки объектов на основе интерфейсов (или фрагментов программ), представляющих собой независимые компоненты. Программы создаются как инсталляционные комплекты в форме *сборок*. Каждый тип сборки имеет уникальный идентификатор – номер версии сборки, как самодостаточный компонент для развертывания, тиражирования и повторного использования. Сборка может включать несколько пространств имен, занимать несколько сборок, которые объединяются в манифест сборки. Манифест содержит метаданные о компонентах сборки, идентификатор автора и версии, сведения о типах и зависимостях, а также режим и политику сборки. Ссылочные типы включают в себя типы: объектные (object type); интерфейсные (interface type); указатели (pointer type).

CTS включает в себя другие библиотеки: *CLR* (Common Language Runtime), *CLS* (Common Language Specification) и *CIIL*. Сервисы в CLR предоставлены библиотекой классов (более 1000) и моделью ASP.NET.

Средства сборки компонентов в JAVA. Основные типы компонентов в языке JAVA – это проекты, формы (AWT-компоненты), beans компоненты, COBRA компоненты, RMI-компоненты, стандартные классы-оболочки, JSP компоненты, сервлеты, XML-документы, DTD-документы и файлы разных типов и др. Интерфейс является частью спецификации названных компонентов и способствует проведению интеграции компонентов в среде системы JAVA [15]. Создание нового проекта состоит в обеспечении взаимодействия компонентов и использования шаблонов повторного использования: BlankAntProject,и SampleAntProject и CastomTask. К основному классу относится Class, Main, Empty (пустой класс), как шаблон типа: exception, persistence Capable и interface. Для построения классов с помощью шаблонов используются классы-оболочки (Boolean, Character, BigInteger, BigDecimal, Class) и AWT библиотека классов,, которая содержатся формы контейнеров для графических элементов и интерфейсов пользователя, а также системы классов Abstract Window Toolkit для построения абстрактного окна. Для обеспечения взаимодействия используется метод RMI, который дополняет язык JAVA стандартной моделью EJB (Enterprise JAVA Beans) компании SUN. Механизм развертывания JAVA–компонентов

типа beans на сервере базируется на программах в исходном языке, а сервер создает для них среду для сборки и взаимодействия разных сред JAVA и MS.Net.

Комплекс ИТК [16] реализует подход к сборке MS.Net и принцип взаимодействия компонентов, в языке VS.Net ↔ Eclipse, созданных в среде JAVA и MS.Net с помощью промежуточного модуля и плагина Eclipse.

3.3. Средства поддержки приложений, работающих с Большими Данными

При разработке таких приложений используются операции анализа и описания данных [17]:

- A/B Testing, Crowdsourcing Data Fusion;
- Integration Genetic Algorithms Machine Learning;
- Natural Language Processing;
- Signal Processing Simulation and Visualization;
- Massively Parallel Processing;
- Search-Based Applications, Data Mining и др.

Большие данные могут также быть представлены как tensors, которые управляют вычислением, как например, полилинейное обучение подпространств (multilinear subspace learning). Технологии обращения к большим данным, включают массив параллельно-обрабатывающей (MPP) базы данных и извлечение данных на основе приложения, распределенные файловые системы, распределенные базы данных и инфраструктуры (приложения, хранение и вычисляющие ресурсы) Интернет. Формальные механизмы работы с неструктурированными данными еще полностью не сформированы. Предстоит разработать набор функций для обработкиэлементов неструктурированных данных.

Инструментальные средства для разработки приложений с Big Data:

- Oracle Designer и Oracle Developer, который состоит из Oracle Forms Oracle Discoverer и Oracle Reports;
- Oracle JDeveloper;
- NetBeans;
- Oracle Application Express;
- Oracle SQL Developer;
- OEPE, Пакет Pach for Eclipce.

Исходя из анализа современных подходов к обработке структурных и небструктуриророванных ТД, сделан вывод о том, что формальные стандартные механизмы работы с такими данными еще не сформировались. Их требуется разрабатывать.

3.4. Проведение анализа нестандартных данных из наборов Больших данных

К методам анализа данных относятся статистические методы (дескриптивный анализ, корреляционный и регрессионный анализ, компонентный анализ и др.), а также методы синтаксического анализа раздела описания сложных данных GDT. Регрессионный и компонентный метод математически ориентированы на оценку необходимой величины экспертом и сравнения ее с другими величинами. Описание ТД представляется в виде таблицы СМ метода [7] терминальных символов и семантических программ их реализации. Каждому представлению терминальных символов соответствует операционный знак его обработки (+, - , / и др.). Функции их реализации могут повторяться и для других ТД. Для проведения анализа ТД предлагается создать таблицу ТД и набор функций их реализации в языке XML. Другая таблица включает набор неструктурированных ТД, которым прикреплены функции трансформации таких ТД к имеющимся в первой таблице. Результатом обработки этой таблицы является разложение неструктурных данных в виду простых данных в том порядке, в котором они заданы в исходной таблице.

4. Заключение

Рассмотрены общие и неструктурированные типы данных. Дано формальное описание всех приведенных ТД и подходов к их генерации и отображению сложных типов данных GDT к более простым. Представлены алгебраические системы простых и сложных ТД и операции их преобразования данных для MS.Net, IBM SCA, OMG MDA и др. Рассмотрен набор процедур преобразования неструктурных и неструктурированных ТД GDT к формату платформы и к более простым ТД. Сделан вывод о возможности использования этих подходов к формальному описанию отдельных наборов больших данных.

Литература

[1]. Standard ISO/IEC 11404 General Data Types, 1997.- 147 p.

[2]. Леман Д., Смит М. Типы данных // Данные в языках программирования.– М.: Мир, 1982.– С.

196– 213.

[3]. Вирт Н. Алгоритм + структуры данных= программы: Пер. с англ.–М.: Мир, 1985, 406 с.

[4]. Замулин А.В. Типы данных в языках программирования и базах данных.– М.: Наука, 1987.–

152 с.

[5]. Danahue P. On the semantics of data types // SIAM J. Comput.– 1979.– 8, N 4.– P. 546–560.

[6]. Hoar K. О Структурной организации данных. //Структурное программирование.– М.: Мир,

1975.– с.92 –97.

[7]. Лаврищева Е.М. Грищенко В.Н. Связь разноязыковых модулей в ОС ЕС. 1982, М.: Финансы и

статистика, 136 С.

[8]. Лаврищева Е.М. Грищенко В.Н. Сборочное программирование.– Киев.: Наук. Думка, 1991.–

209 с.

[9]. Эммерих В. Конструирование распределенных объектов. Методы и средства

программирования интероперабельных объектов в архитектурах OMG/CORBA, Microsoft

COM и Java RMI. – М.: Мир, 2002. – 510с.

[10]. Сигел Дж. CORBA 3. - Москва: Малип, - 2002. – 412 с.

[11].Лаврищева Е.М. Software Engineering. Парадигмы, технологии, CASE– средства программирования.–Наук. Думка, 2014, 284 с.

[12]. Библиотека MSDN:http://msdn.microsoft.com/

[13]. Описание Kadr1250–Лунев.

[14]. Эммерих В. Конструирование распределенных объектов. Методы и средства программирования интероперабельных объектов в архитектурах OMG/CORBA, Microsoft COM и Java RMI. – М.: Мир, 2002. – 510с.

[15]. Вебер Д. Технология Java в подлиннике: Пер. С англ. – СПб.: BHV – Санкт-Петербург, 1999. – 1104 с.

[16]. Лаврищева Е.М. Инструментально–технологический комплекс для разработки и обучения приемам производства программных систем.– Киев.–Весник НАН, 2012.–№3.–с.67–80.

[17]. wikipedia.org/wiki/Большие_данные, wikipedia.org/wiki/MapReduce

ОБ ОСОБЫХ КРИВЫХ СПЕЦИАЛЬНОЙ СИСТЕМЫ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ В ЧАСТНЫХ ПРОИЗВОДНЫХ ВТОРОГО ПОРЯДКА, ОПРЕДЕЛЯЕМЫХ ЛИНИЯМИ ВТОРОГО ПОРЯДКА

Тасмамбетов Жаксылык Нурадинович
профессор, доктор физико-математических наук,
Актюбинский государственный университет имени К.Жубанова, г. Актобе
tasmam45@mail.ru

АННОТАЦИЯ

Целью настоящей работы является установление особых кривых рассматриваемой системы дифференциальных уравнений в частных производных второго порядка. Для построения решения системы вблизи особых точек применяется метод Фробениуса-Латышевой. Установлены возможные особенности системы. Подробно изучен случай, когда система имеет особенности, определяемые линиями второго порядка. Проведена классификация регулярных и иррегулярных особенностей. Показаны виды решения вблизи установленных особенностей.

ABSTRACT

The aim of this work is to establish the special curves of system of second order partial differential equations. To construct a system of solution near singular points the Frobenius-Latysheva method is used. The possible features of the system are established. The case when system has the features that are defined by the lines of second order is studied in details. The classification of regular and irregular features was done. The kinds of solutions near the established features are shown.

Ключевые слова: особые точки, особые кривые, регулярные, иррегулярные, построение решения, классификация, система.

Keywords: singular points, singular curves, regular, irregular, construction of solution, classification, system.

1. Предварительные сведения. Изучением особых точек и их классификацией занимались такие известные математики как К.Вейерштрасс, Л.Фукс, Б.Риман, Г.Фробениус, К.Гаусс, П.Пенлеве, Я.Горн, Л.Томе и др. Они занимались построением аналитических решений в окрестности особых точек. Разделение особых точек интегралов дифференциальных уравнений на два класса – неподвижные и подвижные, принадлежит основоположнику аналитической теории дифференциальных уравнений Л.Фуксу [1]. Неподвижными особыми точками обладают линейные обыкновенные дифференциальные уравнения. Поэтому, аналитический характер решений линейных дифференциальных уравнений вполне определяется их поведением в области неподвижных особых точек. Особыми точками таких уравнений могут быть особые точки их коэффициентов, нули, коэффициент при старшей производной, точка на бесконечности. Отсюда возникает необходимость выяснения характера аналитической функции, определяющей аналитическое решение уравнений.

Классификация особых точек однозначных функций комплексного переменного и их названия были предложены Вейерштрассом в 1876 г. [2]. Он подразделял их на несущественно и существенно особые, имея ввиду изолированные особые точки. Дальнейшая классификация особых точек связана с их регулярностью и иррегулярностью. Введение термина «регулярное решение» связано с именем Л.Томе [3]. Линейные дифференциальные уравнения, решения которых имеют все точки регулярными, называются уравнениями класса Фукса. К.Я.Латышева регулярность и иррегулярность особых точек определяет [4] с помощью понятия ранга $p = 1 + k$ (k – подранг), введенного А.Пуанкаре [5] и антиранга $\mu = -1 - \chi$ (χ – антиподранг), введенного Л.Томе.

Обобщение понятия особых точек на функции многих переменных также было дано К. Вейерштрассом в 1880 г. В отличие от случая одного комплексного переменного, аналитическая функция двух и более переменных не может иметь изо-