Интернет-журнал «Отходы и ресурсы» https://resources.today
Russian Journal of Resources, Conservation and Recycling

2024, Том 11, № 4 / 2024, Vol. 11, Iss. 4 https://resources.today/issue-4-2024.html

URL статьи: https://resources.today/PDF/09INOR424.pdf

DOI: 10.15862/09INOR424 (https://doi.org/10.15862/09INOR424)

2.3.1. Системный анализ, управление и обработка информации, статистика (технические науки)

Ссылка для цитирования этой статьи:
Диомидов, И. А. Модель интегрированной системы обработки информации о лесах и лесных ресурсах / И. А. Диомидов // Отходы и ресурсы. — 2024. — Т. 11. — № 4. — URL: https://resources.today/PDF/09INOR424.pdf DOI: 10.15862/09INOR424

For citation:
Diomidov I.A. Integrated forest and forest resources information processing system model. *Russian Journal of Resources, Conservation and Recycling*. 2024;11(4): 09INOR424. Available at: https://resources.today/PDF/09INOR424.pdf. (In Russ., abstract in Eng.) DOI: 10.15862/09INOR424

УДК 004.051

Диомидов Иван Андреевич
ФГБОУ ВО «Иркутский государственный университет путей сообщения», Иркутск, Россия
Аспирант
ФГБУ «Рослесинфорг», Иркутск, Россия
Прибайкаллеспроект
Ведущий-инженер программист
Аспирант
E-mail: pilad0hwtts@yandex.ru
РИНЦ: https://elibrary.ru/author_profile.asp?id=1220889

Модель интегрированной системы обработки информации о лесах и лесных ресурсах

Аннотация. Проблема интеграции актуальна для лесного хозяйства, поскольку качество данных напрямую влияет на эффективность процессов управления. Целью данного исследования является построение формальной модели системы интеграции данных. Для достижения этой цели был проведен анализ публикаций по процессам и подходам к интегрированной обработке и управлению данными. Задача интеграции поставлена как консолидация данных. При построении модели выполнено ее декомпозиция на задачу установления зависимостей между множествами типов данных, проектируемых при создании конкретных систем обработки. Построена модель запросов над данными. Модель запросов разделена на логическую (перечень реализованных логических групп операций — функциональных или интеграционных) и физическую (последовательность преобразований). На основе модели запросов была предложена модель системы интеграции, где каждый ее элемент — модуль — состоит из функциональных и логических компонентов для определенных типов данных. Каждый модуль данной системы разрабатывается для решения двух задач: реализации конкретных прикладных требований и разработки конкретных логических групп операций. Достоинством данной модели является ее структурная близость к программным системам обработки информации. В работе приведен пример применения данной модели — проектирование набора нескольких модулей с учетом их возможной интеграции. Исследование вносит вклад в развитие теоретических подходов к решению задачи интеграции данных и может быть полезным для специалистов и исследователей в области системного анализа и лесного хозяйства.

Ключевые слова: системный анализ; базы данных; лесное хозяйство; интеграция; лесоустройство; запросы

Введение

Инвентаризация лесов является процессом, на котором основывается лесное хозяйство. Результат инвентаризации — актуальная и достоверная информация о местоположении, количественных и качественных характеристик лесов и лесных ресурсов. Она представлена в виде государственного лесного реестра — основа всех процессов управления в отрасли [1; 2]. Для нашей страны позитивным фактором являются значительные объемы этих ресурсов: Российская Федерация занимает первое место по площади покрытых лесом территорий. В то же время, это создает вызов: эффективное управление лесными ресурсами невозможно без надлежащего информационного обеспечения. Совершенствуются методы сбора данных, меняются регламенты проведения лесоустройства и связанных с ним работ. Основные направления по развитию отрасли — повышение точности методов сбора информации и увеличение частоты проведения работ по инвентаризации. Существуют также аналитические системы, выполняющие анализ лесной растительности с научной точки зрения, в том числе с использованием дистанционных методов. Несмотря на наличие множества инструментов, как систем общего пользования, так и специализированных, разнородность решаемых задач, обширность территорий и динамичность данных приводят к тому, что инструменты функционируют изолированно. Интеграция же данных без изменения подходов к их обработке окажется достаточно неэффективной, поскольку информация в этой сфере быстро устаревает в силу естественных причин. Кроме этого, продолжают развиваться новые методы анализа и обработки информации о лесе, создаются новые технологии. Поэтому интеграцию данных и методов обработки необходимо рассматривать как единый процесс. Решение данной проблемы может быть найдено с помощью программных систем интеграции. Однако конкретные методические подходы к построению таких систем, особенно для лесного хозяйства, изучены недостаточно. Данная работа посвящена формализации решения сложившейся проблемы. Для этого выполняется анализ существующих работ по совершенствованию и унификации методов сбора, обработки и интеграции данных и предлагается методика создания системы интеграции, с учетом специфики данной предметной области. Объектом исследования являются системы управления информацией и их взаимодействие, предметом — процессы и подходы, направленные на решение проблемы интеграции данных и методов.

Связанные работы

Проектирование систем для лесного хозяйства и лесных экосистем рассматривается в работах М.Р. Вагизова [3; 4]. В них представлен методологический подход, архитектура и содержание моделирования лесных экосистем. Автором подчеркивается многоуровневость информации о лесах: от лесного массива до одиночного дерева, ярусов насаждений и напочвенного покрова. Также отмечается разнообразие используемых методов — полевые измерения, дешифровочные и нейросетевые методы. Такое многообразие методов обуславливает широкий спектр решаемых задач: создание прогностических моделей, оценка качества данных. Важным требованием к разрабатываемой системе является ее модифицируемость и расширяемость, возможность работы с гетерогенными данными и учет специфики работ в различных условиях, например, в полевых.

А.З. Швиденко подчеркивает необходимость создания адаптивных систем управления лесным хозяйством в связи с глобальными изменениями климата [5]. Изменение климатических условий требует пересмотра подходов к управлению лесами, включая используемые данные и инструменты обработки.

В работе [6] показано применение CASE-технологий для моделирования процессов лесоустройства. В ней продемонстрирована глубокая взаимосвязь информации на разных этапах лесоустройства. В состав системы входят подготовленные снимки, данные о договорах, протоколы совещаний, материалы предыдущего лесоустройства и планы хозяйственной деятельности. В исследовании описываются основные потоки данных и процессы лесоустройства. Авторы отметили необходимость дальнейшего дробления имеющихся этапов на процессы и процедуры.

Коллектив ученых (Грабарник П.Я., Чертов О.Г., Чумаченко С.И. и др.) выполнил разработку и интеграцию нескольких специализированных имитационных моделей в единую модель для комплексной оценки лесных экосистемных функций [7]. Для интеграции каждой модели был добавлен программный модуль, обеспечивающий чтение и запись обменных файлов, а также преобразование данных между моделями. Кроме того, в работе были решены проблемы, связанные с синхронизацией временных и пространственных масштабов. Авторы подробно рассмотрели нюансы интеграции моделей, касающиеся специфики лесного хозяйства. Результатом работы стала конкретная система оценки экосистемных функций лесов. Ключевой проблемой при интеграции моделей является сопоставление типов данных.

В работе [8] была выполнена интеграция систем ГИС с моделями ForestGALES и Ecological Site Classification. Авторы провели детальный анализ применения ГИС для пространственного анализа рисков и лесопользовательских решений, а также операционного применения и визуализации данных.

Предлагались концепции создания распределённой базы данных для возобновляемых природных ресурсов [9]. Основная цель таких систем — обеспечить непрерывную актуализацию данных, что позволяет эффективно управлять лесными ресурсами. Работа затрагивает проблемы хранения, интеграции и синхронизации картографической информации. Также рассматривается разработка системы репликации для обеспечения одинакового состояния центральной и узловых баз данных. Отмечается предпочтение СУБД PostgreSQL в качестве инструментального средства, так как она легко интегрируется в ГИС.

Более подробный анализ применяемых в отрасли систем обработки информации, а также вопросы интеграции и их связь с системой инвентаризации рассмотрены настоящей статьи автором в следующей работе [10].

Исследования показывают, что задача интеграции данных в лесном хозяйстве уже рассматривалась ранее. Общий вывод: система интеграции должна учитывать различные аспекты информации о лесах и быть адаптивной к непрерывному обновлению как самих данных, так и методов их обработки. Однако, важность рассмотрения ее как проблемы сопоставления типов данных недостаточно подробно раскрывается в доступных работах. Она рассматривается вторично, по отношению к целям предметной области.

Метод исследования

Моделирование системы интеграции выполняется в несколько этапов: постановка задачи интеграции и ее декомпозиция. Прим этом проводится адаптация нескольких подходов к предметной области. В данной работе большой акцент будет сделан на формализации методов и подходов применяющихся при создании программных систем. Так исходная задача интеграции поставлена на базе подхода ETL (Extract Transform Load) [11; 12]. Основным способом организации запросов рассматриваются интегрированные языковые запросы LINQ (Language Integrated Query) [13–17]. На их основе выполняется декомпозиция задачи и предлагается методический подход по организации компонентов системы. Сама система интеграция в этом случае является синтезом созданных в рамках каждого компонента

преобразований в единую цепочку обработки. Каждый компонент реализует отдельный этап процесса.

Задача интеграции

Процесс консолидации данных может быть записан следующим образом [11]:

$$\left\langle \{\langle D_i, \Sigma_i \rangle | i = 1, \dots, N\}, ETL(\langle D_i, \Sigma_i \rangle), \langle D_I, \Sigma_I \rangle \right\rangle, \quad (1)$$

где:

$i = 1, \dots, N$ — индекс исходных наборов данных, где N — их количество;

$\langle D_i, \Sigma_i \rangle$ — множество входных наборов данных, где Σ_i — схема данных для каждого набора D_i;

$\langle D_I, \Sigma_I \rangle$ — выходной набор данных, где Σ_I — схема данных для набора D_I;

$ETL(\langle D_i, \Sigma_i \rangle, \langle D_I, \Sigma_I \rangle)$ представляет применение процедур извлечения ($Extract$), трансформации ($Transform$) и загрузки ($Load$) для преобразования множеств входных наборов данных в выходной набор данных.

Для иллюстрации многоэтапности процесса интеграции, обозначим множество входных наборов как $\langle D_{\text{source}}, \Sigma_{\text{source}} \rangle$, а выходной набор — как $\langle D_{\text{target}}, \Sigma_{\text{target}} \rangle$. Тогда ETL может быть представлен как последовательность выполнения функций $Extract$, $Transform$ и $Load$:

$$\langle D_{\text{target}}, \Sigma_{\text{target}} \rangle = \text{Load}\left(\text{Transform}\left(\text{Extract}(\langle D_{\text{source}}, \Sigma_{\text{source}} \rangle)\right)\right), \quad (2)$$

где:

$\langle D_{\text{intermediate}}, \Sigma_{\text{intermediate}} \rangle = \text{Extract}(\langle D_{\text{source}}, \Sigma_{\text{source}} \rangle)$;

$\langle D_{\text{transformed}}, \Sigma_{\text{transformed}} \rangle = \text{Transform}(\langle D_{\text{intermediate}}, \Sigma_{\text{intermediate}} \rangle)$;

$\langle D_{\text{target}}, \Sigma_{\text{target}} \rangle = \text{Load}(\langle D_{\text{transformed}}, \Sigma_{\text{transformed}} \rangle)$.

Цепочка обработки

Для представления данных используются различные модели: объектные, онтологические, документные, реляционные, графовые, иерархические и сетевые. Каждый тип модели поддерживает свой набор запросов: OQL (Object Query Language), ORM (Object-Relational Mapping), DSL (Domain-Specific Language), SPARQL (SPARQL Protocol and RDF Query Language), SQL (Structured Query Language), XQuery, DML (Data Manipulation Language) [18].

Исторически сложилось так, что объектная модель получила широкое распространение в программных системах. Безусловно, большим фактором для ее развития стала парадигма объектно-ориентированного программирования. Для работы с данными в этой модели могут применяться операторы: объявление переменных, присвоения значений и так далее. С целью обобщение данных операций над перечисляемыми типами данных (массивы, списки), данные группы операторов были обобщены и представлены в виде набора операций — интегрированных языковых запросов (в дальнейшем LINQ) [13]. С их помощью можно решать типовые задачи над наборами данных: проецирование, фильтрацию, группировку, агрегацию, сортировку и так далее. За счет того, что они выполняются в рамках конкретной программной системы, имеется возможность вызывать системные операторы, влиять на сам процесс вычисления запросов, отправлять запросы к другим информационными системам и т. д. Таким образом, обработка конкретных наборов данных может выполняться параллельно с выполнение системных процессов.

Все операторы в LINQ основывается на концепции абстрактных типов данных и парадигме метапрограммирования. Поддерживаемые операции могут выполняться над типами данных (примитивные, структурные, перечисления). Множество $Type$ типов определим как:

$$Type = \{Type_1, Type_2, \dots, Type_{\text{nt}}\}, \quad (3)$$

где:

$Type_{\text{it}} \in Type$ — тип данных, $it \leq nt$;

nt — количество типов данных, $nt > 0$.

В рамках ETL-подхода операция преобразования обозначается как $Transform$. Она в данной работе рассматривается как полиморфная функция, что означает возможность ее изучения на разных уровнях абстракции: уровне типов, логическом и физическом уровне. Примем соглашение, что с помощью типа данных возможно полно отобразить любую схему данных (в рамках ETL задачи), т. е.:

$$\Sigma \mapsto Type, \quad (4)$$

где:

Σ — схема данных;

$Type$ — соответствующий тип данных, отображающий схему.

Тогда в первом приближении, задачу функции $Transform$ можно выразить через отображение:

$$\text{Transform: } Type_{\text{intermediate}} \rightarrow Type_{\text{transformed}}, \quad (5)$$

где:

$Type_{\text{intermediate}} = \Sigma_{\text{intermediate}} \mapsto Type_{\text{intermediate}}$ — тип промежуточных данных (схема данных полученных после извлечения);

$Type_{\text{transformed}} = \Sigma_{\text{transformed}} \mapsto Type_{\text{transformed}}$ — тип преобразованных данных (целевая схема данных).

Для определения данной связи в рамках подхода LINQ применяются операции. Введем множество поддерживаемых операций O:

$$O = \{o_1, o_2, \dots, o_{\text{no}}\}, \quad (6)$$

где:

$o_{\text{io}} \in O$ — операция LINQ, $io \leq no$;

no — количество поддерживаемых операций.

В данной работе операция $o_{io} \in O$ определяется как атомарная единица обработки данных. Каждая такая операция принимает один терм в качестве входных данных и возвращает один терм в качестве результата. Это можно представить как метод, у которого параметры и возвращаемый тип выделены в отдельные классы объектов (структуры).

Так же для наглядности введем следующие описательные индексы для ряда операторов, так $o_1 = o_{\text{project}}$, $o_2 = o_{\text{filter}}$, $o_3 = o_{\text{group}}$, $o_4 = o_{\text{aggregate}}$. Полно все операции представлены в работе [13].

Операции в LINQ могут выполняться над объектами, поддерживающими перечисляемые типы, за счет механизмов ковариантности и контравариантности. Тогда $Transform$ может быть определена в минимальном выражении как отображение между типами входного и выходного терма:

$$\begin{gathered}\text{Transform} = \langle o_{\text{project}} \rangle \\ o_{\text{project}}: Type_{\text{intermediate}} \to Type_{\text{transformed}}, \\ \text{Transform}: Type_{\text{in}}(o_{\text{project}}) \to Type_{\text{out}}(o_{\text{project}}) \equiv Type_{\text{intermediate}} \overset{o_{\text{project}}}{\to} Type_{\text{transformed}}\end{gathered} \quad (7)$$

где:

$Type_{\text{in}}(o_{\text{io}})$ — функция, возвращающая тип входного терма операции $o_{io} \in O$;

$Type_{\text{out}}(o_{\text{io}})$ — функция, возвращающая тип выходного терма операции $o_{io} \in O$;

o_{project} — операция проекции с входным типом $Type_{\text{intermediate}}$ и с выходным типом $Type_{\text{transformed}}$.

На практике, преобразование не ограничивается одной операцией. Введем понятие последовательности операций в форме кортежа. Если S — последовательность операций, то:

$$\begin{gathered}\text{S} = \langle o_{1,io_1}, o_{2,io_2}, \dots, o_{ns,io_{ns}} \rangle \\ \text{S}: Type_{\text{in}}(S) \to Type_{\text{out}}(S)\end{gathered}, \quad (8)$$

где:

$o_{is,io_{is}} \in O$, для $is < ns$, ns — количество операций $ns \geq 1$, $Type_{\text{in}}(S) = Type_{\text{in}}(o_{1,io_1})$;

$Type_{\text{out}}(S) = Type_{\text{out}}(o_{ns,io_{ns}})$, $\forall 1 \leq i < ns: Type_{\text{out}}(o_{is,io_i}) = Type_{\text{in}}(o_{is+1,io_{i+1}})$.

Рассмотрим пример. Следующая последовательность операций S включает в себя операции фильтрации, проецирования, группировки и агрегирования:

$$\begin{gathered}S = \langle o_{\text{filter}}, o_{\text{project}}, o_{\text{group}}, o_{\text{aggregate}} \rangle \\ o_{\text{filter}}: Type_{\text{intermediate}} \to Type_{\text{intermediate}} \\ o_{\text{project}}: Type_{\text{intermediate}} \to Type_{\text{projected}} \\ o_{\text{group}}: Type_{\text{projected}} \to Type_{\text{grouped}} \\ o_{\text{aggregate}}: Type_{\text{grouped}} \to Type_{\text{transformed}}\end{gathered}, \quad (9)$$

Тогда функция $Transform$ может быть определена через данную последовательность, т. е.:

$$\begin{gathered}\text{Transform} = \langle S \rangle \\ Type_{\text{in}}(S) = Type_{\text{in}}(\text{o}_{\text{filter}}) = Type_{\text{intermediate}} \\ Type_{\text{out}}(S) = Type_{\text{out}}(\text{o}_{\text{aggregate}}) = Type_{\text{transformed}} \\ \text{Transform}: Type_{\text{intermediate}} \overset{S}{\to} Type_{\text{transformed}}. \\ \equiv Type_{\text{intermediate}} \overset{o_{\text{filter}}}{\to} Type_{\text{intermediate}} \overset{o_{\text{project}}}{\to} Type_{\text{projected}} \\ \overset{o_{\text{group}}}{\to} Type_{\text{grouped}} \overset{o_{\text{aggregate}}}{\to} Type_{\text{transformed}}\end{gathered} \quad (10)$$

Данную запись можно рассматривать как физическую модель функции $Transform$. На этом уровне процесс преобразования представляет как последовательная цепочка обработки (англ. pipeline) типов данных, где результат выполнения одной операции передается в другую. Здесь иллюстрируется сущность подхода, основанная на использовании LINQ: часть операций (project, group и aggregate) возвращают *производный тип данных*. Это является ключевым отличием интегрированных языковых запросов от, например, SQL. Разработчик системы не только решает исходную задачу интеграции, но и вынужден декомпозировать ее, как минимум за счет производных (промежуточных) наборов данных.

Из-за этого, целесообразно рассматривать общую последовательность как набор несколько независимых. Это позволит группировать операций по смыслу решаемой задачи. Для обеспечения процесса интеграции ETL, конкретизируем часть последовательностей. Очевидно, часть из них должна выполнять непосредственную обработку схем и наборов данных.

Интеграционные последовательности — последовательности, непосредственно реализующие различные аспекты задачи интеграции. Они выполняют преобразования над данными, включая их очистку, сопоставление сущностей, агрегация и обогащение данных. Эти компоненты изменяют структуру или содержание данных в соответствии с задачами ETL.

Примеры таких последовательностей: выполняющие очистку данных, идентификацию сущностей, группировку.

Остальные последовательности, которые не относятся напрямую к задаче ETL (преобразования схем и наборов), обозначим как функциональные последовательности.

Функциональные последовательности — процессы, обеспечивающих вспомогательные функции в процессе обработки данных, но не связанных напрямую с преобразованием самих данных. Они не изменяют исходную структуру и содержание данных, но обеспечивают функциональность процесса.

Примеры таких последовательностей: инициализация подключений, настройка параметров систем, чтение конфигураций, логирование и другие операции, обеспечивающие работоспособность самого процесса.

С формальной точки зрения, появление новых типов данных не имеет большого значения для решения задач ETL. Однако, как уже было показано ранее, они могут возникать естественным образом в ходе выполнения любых преобразований над данными.

Интеграционные последовательности будем обозначать как I, функциональные как T. Общее множество последовательностей будет являться результатом объединения:

$$S = I \cup T. \tag{11}$$

Рассмотрим пример:

$$\begin{aligned} I &= \{I_{\text{cleaning}}, I_{\text{entity\_identification}}, I_{\text{aggregation}}\} \\ I_{\text{cleaning}}&: Type_{\text{intermediate}} \rightarrow Type_{\text{intermediate}} \\ I_{\text{entity\_identification}}&: Type_{\text{intermediate}} \rightarrow Type_{\text{intermediate}}, \\ I_{\text{aggregation}}&: Type_{\text{intermediate}} \rightarrow Type_{\text{transformed}} \end{aligned} \tag{12}$$

где:

I_{cleaning} — очистка данных (удаление дубликатов, удаление выделов с ошибочной информацией);

$I_{\text{entity\_identification}}$ — идентификация и сопоставление сущностей (сопоставление лесотаксационных выделов, лесных участков);

$I_{\text{aggregation}}$ — агрегация и обогащение данных и преобразование (расчет общих показателей площадей, запасов).

$$\begin{aligned} T &= \{T_{\text{connect}}, T_{\text{to}_{\text{segment}}}, T_{\text{calculate}}\} \\ T_{\text{connect}}&: Type_{\text{connection\_opt}} \rightarrow Type_{\text{intermediate}} \\ T_{\text{to\_segment}}&: Type_{\text{intermediate}} \rightarrow Type_{\text{intermediate}}, \\ T_{\text{calculate}}&: Type_{\text{transformed}} \rightarrow Type_{\text{statistics}} \end{aligned} \tag{13}$$

где:

T_{connect} — инициализация подключения к лесной базе данных (ретроспективная база данных, СУБД, Excel файлы);

$T_{to_{segment}}$ — метод сегментации базы для параллельной обработки (лесные базы данных как правило достигают больших объемов, при этом обработку можно выполнять параллельно);

$T_{\text{calculate}}$ — расчёт статистики (количество успешно обработанных выделов, количество ошибок и их содержание) и передача в документные базы данных, пользовательский интерфейс.

Операции здесь опущены, так как это уже является деталями реализации конкретных последовательностей. Но логически $Transform$ может быть определена, как последовательность вызовов следующих интегрированных и функциональных последовательностей:

$$\text{Transform} = \left\langle \text{T}_{\text{connect}}, \text{T}_{\text{to}_{\text{segment}}}, \text{I}_{\text{cleaning}}, \text{I}_{\text{entity\_identification}}, \text{I}_{\text{aggregation}}\, \text{T}_{\text{calculate}} \right\rangle$$
$$\text{Transform: Type}_{\text{intermediate}} \rightarrow \text{Type}_{\text{transformed}}. \quad (14)$$

Таким образом, на уровне типов функция $Transform$ обеспечивает отображение между двумя типами данных $Type_{intermediate}$ и $Type_{transformed}$, физически представляет цепочку обработки, а логически декомпозируется на два класса последовательностей: функциональные и интеграционные. Таким образом удается достичь двойственности процессов интеграции (данных и методов). Интеграционные преобразования обеспечивают изменения в конкретных наборах данных, а функциональные позволяют достичь интегрируемости системы с внешними:

- функция $Transform$ выполняет преобразования над конкретными наборами данных с целью обеспечения интеграции данных *ETL*;
- функция $Transform$ является синтезируемой и адаптивной за счет внедрения интеграционных последовательностей I, что делает ее адаптивной;
- функция $Transform$ является интегрируемой, за счет функциональных последовательностей T.

Модель системы интеграции.

Цель системы интеграции — реализовать достаточное количество интеграционных последовательностей I для процесса ETL. Для этого предлагается выполнять создание данных последовательностей в рамках небольших задач или аспектов системы. Тогда каждый элемент системы может рассматриваться как независимая единица, решающая определенную задачу — модуль. Введем понятие модуля M, который реализует набор обработчиков и поддерживает набор типов:

$$M^* = \{M_1, M_2, \dots, M_{nm}\}$$
$$M_{\text{im}} = (Type^*_{\text{im}}, I^*_{\text{im}}, T^*_{\text{im}})' \quad (15)$$

где:

$M_{\text{im}} \in M^*$, $im \leq nm$, nm — число модулей, $nm > 0$;

$Type^*_{\text{im}}$ множество поддерживаемых типов в модуле M_{im};

I^*_{im} множество интеграционных последовательностей в модуле M_{im};

T^*_{im} множество функциональных последовательностей в модуле M_{im}.

Издательство «Мир науки» \ Publishing company «World of science» http://izd-mn.com

Модулем — может быть существующая программная система, создаваемая программная система. Ключевым при рассмотрении модуля является те типы данных, что в нем поддерживаются, а также те функциональные и интеграционные компоненты, что в нем реализованы. Для того чтобы рассматривать модули как целостную систему, единственным требованием является общая система типов.

При наличии общих типов данных, существует возможность использовать результат выполнения одной последовательности для передачи в следующем. На уровне модуля, это позволяет создавать новые модули на основе уже существующих. Обозначим множество зависимостей между модулями как D:

$$D \subseteq M \times M, \tag{16}$$

где:

$(M_{im}, M_{dm}) \in D$ означает, что модуль M_{im} зависит от модуля M_{dm}, $dm \leq nm$.

Эксперты отмечают, что основным источником неопределенности в системе являются требования [19]. В данной работе автор принимает такое соглашение, что на этапе проектирования требования являются частично формализованными. Причины следующие:

- функциональные компоненты нацелены на обеспечения самого процесса интеграции (преобразования типов), и их сущность лишь косвенно связана с исходной задачей преобразования схем;
- каждая из операций приводит к созданию производного типа данных, который ранее не был определен;
- заранее неизвестно как и какие аспекты исходных схем данных будут задействованы.

Вопрос выбора способ моделирования требований является неопределенным к ним все еще существует множество различных подходов. Однако, специфика программной системы заключается в том, что любое требование в конце концов, может быть сведено к определенной зависимости между типами:

$$\begin{gathered} R = \{r_1, r_2, \dots, r_n\} \\ r_i : Type_{r_i in} \rightarrow Type_{r_i out}, \end{gathered} \tag{17}$$

где:

r_i — требование из множества R;

n — количество определенных требований;

$Type_{r_i in}$ — начальный тип для требования i;

$Type_{r_i out}$ — конечный тип для требования i.

В случае если требование не может быть сведено к зависимости между типами, это означает что либо требование необходимо конкретизировать, либо требование невозможно реализовать в рамках программной системы.

Таким образом, проблема построения требований к системе двойственна:

1. Необходимо четко и ясно определить начальный и выходной тип для конкретизации поставленной задачи.
2. Используемые типы должны быть совместимы с уже существующими для обеспечения синтезируемости создаваемых последовательностей;

Совместимость используемых типов является одновременно основным критерием интегрируемости системы, вместе с этим это конечный результат моделирования требований. Основной источник неопределенности системы заключается в том, что начальные и входные типы в рамках конкретной последовательности часто могут быть определены часто могут быть определены только в последний момент. Это вызвано тем, что, в частности, выходной тип определяется именно операцией. Иными словами, требования неопределённы и динамичны, а значит в рамках системы интеграции при создании новых модулей будут расширяться. Обозначим множество требований на версии ver как R^{ver}:

$$R^{ver} = R_0 \cup \Delta R_1 \cup \Delta R_2 \cup \ldots \cup \Delta R_{ver}, \quad (18)$$

где:

ΔR_i — новые подмножества требований, добавленные на версии i.

Система интеграции, в общем виде, есть результат синтеза всех модулей M^* и их зависимостей D. Обозначим итоговую систему интеграции как IS:

$$IS^{ver} = (M^*, D, R^{ver}), \quad (19)$$

где:

M^* — определенные в рамках конкретной задачи модули;

D — множество зависимостей между модулями;

R^{ver} — множество требований к системе на версии ver.

Дискуссия и практический аспект

Применимость данной модели для лесного хозяйства обусловлена обширностью территорий, наличием множества организаций, ретроспективных баз данных и новых методов обработки. Разработка единого алгоритма по обработке и сведению всех этих данных, с учетом уровня начальной неопределенности в базах данных и изменчивости в отрасли является трудоемкой задачей.

Однако, предложенный подход на основе декомпозиции системы интеграции на отдельные модули является достаточно применимым. Каждый модуль может отражать, например, обработку определенного аспекта данных, или определенного специфичного набора данных.

Рассмотрим следующий практический пример. Несколько лесоустроительных баз данных хранятся в различных организациях. Данные в этих базах могут быть как перекрестными (за один временной период на одну и ту же территорию), так и дополняющими (на территории других лесных участков). Также доступно хранилище спутниковых изображений, используемых в работах, а также публично размещенные данные космического мониторинга, которые могут охватывать обследуемые участки. Подобные случаи подробно рассмотрены в работах [9; 20; 21]. Очевидно, что даже без учета конкретных потребностей организации, эти данные могли бы быть рассмотрены в общем ключе при наличии соответствующих технических средств и потребности.

Для этого необходимо разработать модули, которые с одной стороны, были бы эффективны в разработке и внедрении, а с другой — могли бы использоваться для решения задачи интегрированной обработки. Результаты моделирования представлены в таблице (табл. 1).

Таблица 1

Содержание и структура модулей

M	Type	I	T	D	R
Обработчик НСИ (M_1)	Справочники, Таблица преобразований для справочника	Сопоставление сущностей справочников	Обработка справочных таблиц, Анализ метаданных, Систем кодирования	Отчетный модуль (Драйверы форматов)	Возможность загружать ретроспективные данные со старыми справочниками
Отчетный модуль (M_2)	Выделы, Промежуточные расчетные модели, Таблицы	Драйверы для различных форматов файлов, Возможность пакетной обработки нескольких баз	Алгоритмы создания файлов или таблиц отчетов и ведомостей, Обработка специфичных форматов отчетов	—	Возможность строить отчетные документы в соответствии (или адаптированные старые, или новые)
Модуль актуализации (M_3)	Выделы, Модели актуализации	Алгоритм анализа характеристик выделов за разные промежутки времени	Построение отчетов о проведенной актуализации, базы данных	Отчетный модуль (Драйверы форматов, Построение отчетов и ведомостей)	Возможность выполнять анализ и новой и ретроспективной базой данных
Загрузчик космоснимков (M_4)	Метаданные, Растровые изображения	Сбор метаданных из хранилищ по зоне интереса (покрытие, дата),	Механизмы загрузки больших файлов	—	Возможность выгрузки на необходимые участки космоснимков
Аналитический модуль ДЗ (M_5)	Растровые изображения, Индексы	—	Реализация алгоритмов анализа ДЗ для решения задач классификации, сегментации для определенного типа спутников	—	Возможность выполнять анализ космического изображения с применением методов интеллектуального анализа
Система хранения космоснимков (M_6)	Метаданные, Растровые изображения	Система анализа метаданных и файлами космоснимков	—	Загрузчик космоснимков (Метаданные)	Возможность выполнять анализ сохранённых снимков внутри хранилища организации
Модуль внесения изменений (M_7)	Выделы, Запрос в терминах предметной области	—	Заполнение характеристик выделов по заданному запросу	Отчетный модуль (Драйверы форматов)	Возможность пакетно вносить изменения в выделы по запросу

Разработано автором

Некоторые модули, например, отчетный (M_2) не имеют прямых связей с другими. Хотя его основная функция — построение отчетов, его можно использовать для обеспечения согласованности между различными драйверами из-за большого количества файлов, которые через него будут проходить. Разработка такого пакета модулей позволит решить интеграционную задачу на введенном ранее наборе данных и построить общую цельную

модель, за счет реализованных запросов. В этом ключевую роль сыграют интеграционные компоненты. Их комбинация обеспечит решение задачи с помощью уже разработанных модулей.

Каждый модуль реализует свой небольшой аспект интеграции, что упрощает внесение изменений. Для извлечения можно использовать интеграционные компоненты из модулей M_2 и M_5. Это позволит выгрузить информацию из других баз данных и преобразовать ее в конкретные объектные модели. Модули M_1, M_3 и M_4 выполняют очистку и проверку данных на валидность путем совместного анализа. Результат этого этапа может быть использован для заполнения исходных хранилищ. Модуль M_7 используется для обновления исходных хранилищ. Использование предложенной схемы существенно упрощает задачу интегрированной обработки данных. Интеграционные компоненты уже отлажены и распределены по готовым модулям. Это делает схему интегрированного хранилища более прозрачной. При возникновении ошибок или определении новых требований, модули могут быть дополнены и расширены.

С практической точки зрения, наибольший эффект от интеграции возможен в случае, если разработанные модули не только объединяется в конкретную интегрированную систему, но и сами исходные коды и иная документация становится публичной. Таким образом, достичь единства типов становится возможным за счет использования единой кодовой базы.

Заключение

В данной работе была предложена модель системы интеграции данных. Она рассмотрена в нескольких приближениях: как процесс интеграции и как результат взаимодействия нескольких модулей. На актуальном для лесного хозяйства примере было выполнено проектирование системы, состоящей из небольших модулей, после разработки имеются компоненты, облегчающие переход к системе интеграции данных. Это достигается за счет того, что каждый из модулей решает определенный аспект интеграции. Предложенная формальная модель делает процесс интеграции более прозрачным, за счет постановки задачи интеграции как обеспечение согласованности и определения зависимостей между типами данных, где в качестве языка этих зависимостей предлагается интегрированных языковых запросов. Данные запросы позволяют гибко описывать связи между типами. Также, конкретные виды зависимостей между типами могут меняться со временем из-за требований к системе, для чего и выполняется декомпозиция задачи на отдельные модули, организующиеся прежде всего в рамках небольшого набора требований. Данная модель системы интеграции отвечает требованиям. Дальнейшими направлениями в данной работе могли быть поиск моделей требований к системе, анализ типов связей между типами и модулями, более подробное рассмотрение конкретных интегрированных языковых запросов, рассмотрение конкретных типов данных.

ЛИТЕРАТУРА

1. Сериков, М.Т. Роль лесоустройства и проблемы в обеспечении экологичного освоения лесных рекреационных ресурсов / М.Т. Сериков // Лесотехнический журнал. — 2013. — № 2(10). — С. 75–83. — EDN RMXRKP — URL: https://www.elibrary.ru/item.asp?edn=rmxrkp (дата обращения: 13.12.2024).

2. M. Yu. Vasenev, “Decision Support Systems Utilization in Forestry: Environmental Aspect”, jour, vol. 20, no. 1, pp. 5–17, May 2022, doi: 10.25205/1818-7900-2022-20-1-5-17 — URL: https://www.researchgate.net/publication/360680890_Decision_Support_Systems_Utilization_in_Forestry_Environmental_Aspect?__cf_chl_tk=4Tnzx8smuPJbg7xOSvOkOvLbkTGn3uDBggwB1WBc1DE-1734664761-1.0.1.1-vLLnqysksy4UTbgP_q9n02vM7d9E7nQgAvMiIJVxKIw (дата обращения: 13.12.2024).

3. M. Vagizov, E. Istomin, O. Kolbina, N. Yagotinceva, A. Morshchihina, and K. Konzhgoladze, “Development of a smart geoinformation system module for forest taxing data processing”, Geoinformatika, no. 3, pp. 4–13, Oct. 2021, doi: 10.47148/1609-364X-2021-3-4-13 — URL: https://www.researchgate.net/publication/355083841_Development_of_a_smart_geoinformation_system_module_for_forest_taxing_data_processing (дата обращения: 13.12.2024).

4. Вагизов, М.Р. Разработка технологии геоинформационного моделирования лесных экосистем (часть 1) / М.Р. Вагизов // Геоинформатика. — 2021. — № 4. — С. 43–49. — DOI 10.47148/1609-364X-2021-4-43-49. — EDN HNTFNW — URL: https://www.elibrary.ru/item.asp?edn=hntfnw&ysclid=m4w6neqkgt259712865 (дата обращения: 13.12.2024).

5. Швиденко, А.З. Глобальные изменения и российская лесная таксация / А.З. Швиденко // Лесная таксация и лесоустройство. — 2012. — № 1(47). — С. 52–76. — EDN RMXIAT. — URL: https://www.elibrary.ru/item.asp?id=20787678&ysclid=m4w6num8dv19182680 (дата обращения: 13.12.2024).

6. Функциональное моделирование лесного хозяйства / А.Т. Гурьев, Л.В. Абрамова, С.В. Торхов, Д.В. Трубин // Известия высших учебных заведений. Лесной журнал. — 2004. — № 1. — С. 135–137. — EDN ICGRIX — URL: https://www.elibrary.ru/item.asp?id=9608214 (дата обращения: 13.12.2024).

7. Интеграция имитационных моделей для комплексной оценки экосистемных услуг лесов: методические подходы / П.Я. Грабарник, О.Г. Чертов, С.И. Чумаченко [и др.] // Математическая биология и биоинформатика. — 2019. — Т. 14, № 2. — С. 488–499. — DOI 10.17537/2019.14.488. — EDN NMSPOC — URL: https://www.elibrary.ru/item.asp?id=42429013&ysclid=m4w6qrl0t4345692584 (дата обращения: 13.12.2024).

8. S. Pizzirani and S. Bathgate, “Integration of forestry decision support systems in gis”, J. Env. Assmt. Pol. Mgmt., vol. 14, no. 2, p. 1250014, Jun. 2012, doi: 10.1142/S1464333212500147. — URL: https://www.researchgate.net/publication/254458482_Integration_of_forestry_decision_support_systems_in_GIS (дата обращения: 13.12.2024).

9. Гурьев А.Т. Концепция создания распределённой базы данных возобновляемых природных ресурсов // ГИАБ. 2010. № 12. URL: https://cyberleninka.ru/article/n/kontseptsiya-sozdaniya-raspredelyonnoy-bazy-dannyh-vozobnovlyaemyh-prirodnyh-resursov (дата обращения: 13.12.2024).

10. Диомидов, И.А. Развитие систем обработки информации о лесах и лесных ресурсах / И.А. Диомидов // Отходы и ресурсы. — 2024. — Т. 11. — № 2. — DOI: 10.15862/02INOR224 — URL: https://resources.today/PDF/02INOR224.pdf (дата обращения: 13.12.2024).

11. Vysotska, V., Berko, A., Chyrun, L., Chyrun, S., Havrylyshyn, O., Smirnova, O., Sokulska, N., Sokhatska, O., & Shakleina, I. Formal Data Integration Models Development for Intelligent Electronic Commerce Systems. // CEUR Workshop Proceedings, — 2024 — Vol. 3688 — URL: https://ceur-ws.org/Vol-3688/paper21.pdf (дата обращения: 13.12.2024).
12. Simitsis, Alkis & Vassiliadis, Panos. A Methodology for the Conceptual Modeling of ETL Processes / The 15th Conference on Advanced Information Systems Engineering (CAiSE '03), Klagenfurt/Velden, Austria, 16–20 June, 2003, Workshops Proceedings, Information Systems for a Connected Society — 2003 — С. 305–316 — URL https://www.researchgate.net/publication/220920725_A_Methodology_for_the_Conceptual_Modeling_of_ETL_Processes (дата обращения: 13.12.2024).
13. Cheney, J., Lindley, S., & Wadler, P. The essence of language-integrated query. // Computer Science — 2013 — URL: https://www.semanticscholar.org/paper/The-essence-of-language-integrated-query-Cheney-Lindley/0cb242d33771eef4528d3454183fa2745deee12e (дата обращения: 13.12.2024).
14. B. Frąckowiak and R. Dąbrowski Using LINQ as a universal tool for defining architectural assertions // Position Papers of the Federated Conference on Computer Science and Information Systems, — 2016 — С. 275–282. doi: 10.15439/2016F587 — URL: https://www.annals-csis.org/proceedings/2016/pliks/587.pdf (дата обращения: 13.12.2024).
15. A. Kalnins, E. Kalnina, A. Sostaks, E. Celms, and I. Tabernakulovs, LINQ as Model Tansformation Language for MDD // Baltic Journal of Modern Computing, — 2016 — vol. 4, no. 4, doi: 10.22364/bjmc.2016.4.4.21 — URL: https://www.researchgate.net/publication/311781673_LINQ_as_Model_Tansformation_Language_for_MDD (дата обращения: 13.12.2024).
16. K. Wang and Y. Zheng, Using LINQ as an instructional bridge between object-oriented and database programming, // 2009 4th International Conference on Computer Science & Education, Nanning, China: IEEE, Jul. 2009, pp. 1464–1468. doi: 10.1109/ICCSE.2009.5228565 — URL: https://www.researchgate.net/publication/224585912_Using_LINQ_as_an_instructional_bridge_between_object-oriented_and_database_programming (дата обращения: 13.12.2024).
17. Garcia, M., & Prithiviraj, R. Rethinking the Architecture of O/R Mapping for EMF in terms of LINQ. / Presented at the Eclipse Modeling Symposium at Eclipse Summit // Hamburg University of Technology (TUHH), Hamburg, Germany — 2008 — URL: https://www.odbms.org/wp-content/uploads/2013/11/047.01-Garcia-Rethinking-the-Architecture-of-OR-Mapping-for-EMF-in-terms-of-LINQ-October-2008.pdf (дата обращения: 13.12.2024).
18. M. Fotache and D. Cogean, NoSQL and SQL Databases for Mobile Applications. Case Study: MongoDB versus PostgreSQL // Informatica Economica, — 2013 — vol. 17, no. 2, pp. 41–58, doi: 10.12948/issn14531305/17.2.2013.04 — URL: https://www.researchgate.net/publication/274454559_NoSQL_and_SQL_Databases_for_Mobile_Applications_Case_Study_MongoDB_versus_PostgreSQL (дата обращения: 13.12.2024).
19. Ebert, C., & De Man, J. Requirements uncertainty: influencing factors and concrete improvements. / Software Engineering, 2005. ICSE 2005. Proceedings. 27th International Conference on — 2005 — С. 553–560 — URL: https://www.researchgate.net/publication/4200449_Requirements_Uncertainty_Influencing_Factors_and_Concrete_Improvements (дата обращения: 13.12.2024).

20. Сидорова В.С. Меры достоверности неконтролируемой классификации данных дистанционного зондирования // Интерэкспо Гео-Сибирь. 2008. № 2. URL: https://cyberleninka.ru/article/n/mery-dostovernosti-nekontroliruemoy-klassifikatsii-dannyh-distantsionnogo-zondirovaniya (дата обращения: 13.12.2024).

21. A. Kuusk and M. Lang, Integration of statistical forest reflectance model and Sentinel-2 MSI images into a continuous forest inventory system, // BALTIC FORESTRY, — 2020 — vol. 26, no. 2, doi: 10.46490/BF467 — URL: https://www.researchgate.net/publication/343563600_Integration_of_statistical_forest_reflectance_model_and_Sentinel-2_MSI_images_into_a_continuous_forest_inventory_system (дата обращения: 13.12.2024).

Diomidov Ivan Andreevich
Irkutsk State Transport University, Irkutsk, Russia
Roslesinforg, Irkutsk, Russia
Pribaikallesproekt
E-mail: pilad0hwtts@yandex.ru
RSCI: https://elibrary.ru/author_profile.asp?id=1220889

Integrated forest and forest resources information processing system model

Abstract. The problem of integration is relevant for forestry, as data quality directly affects the efficiency of management processes. The purpose of this study is to develop a formal model for a data integration system. To achieve this goal, an analysis of publications on processes and approaches to integrated data processing and management was conducted. The integration task is formulated as data consolidation. During the model development, it was decomposed into the task of establishing dependencies between sets of data types designed for specific processing systems. A query model over the data was constructed. The query model is divided into logical (a list of implemented logical groups of operations, whether functional or integrative) and physical (a sequence of transformations). Based on the query model, a data integration system model was proposed, where each element — a module — consists of functional and logical components for specific data types. Each module of this system is designed to address two objectives: fulfilling specific application requirements and implementing specific logical groups of operations. The advantage of this model lies in its structural similarity to software information processing systems. The study includes an example application of this model — designing a set of several modules with consideration for their potential integration. This research contributes to the development of theoretical approaches to solving the data integration problem and may be useful for specialists and researchers in the fields of systems analysis and forestry.

Keywords: systems analysis; databases; forestry; integration; forest inventory; queries