Минский инновационный университет
Кафедра информационных технологий
Гедранович О.Б.

Анализ данных

Лекция 1 Технологии анализа данных. Хранилище данных

Системы поддержки принятия решений. Базы данных. Организация хранилища данных. Очистка данных.

Считается, что информация удваивается каждые 2—3 года. Этот потоп, цунами данных приходит из науки, бизнеса, Интернета и других источников. В 1989 году, один мегабайт считался размером большой базы данных.

Из-за огромного количества информации очень малая ее часть будет когда-либо увидена человеческим глазом. Наша единственная надежда понять и найти что-то полезное в этом океане информации — широкое применение методов Data Mining.

Технология Data Mining (также называемая Knowledge Discovery In Data — обнаружение знаний в данных) изучает процесс нахождения новых, действительных и потенциально полезных знаний в базах данных. Data Mining лежит на пересечении нескольких наук, главные из которых — это системы баз данных, статистика и искусственный интеллект.

Data Mining широко используется во многих областях с большим объемом данных: в науке — астрономии, биологии, биоинформатике, медицине, физике и других областях; в бизнесе — торговле, телекоммуникациях, банковском деле, промышленном производстве и т. д. Благодаря Интернет Data Mining используется каждый день тысячи раз в секунду — каждый раз, когда кто-то использует Google или другие поисковые системы (search engines) на просторах Интернета.

Виды информации, с которыми работают исследователи, включают в себя не только цифровые данные, но и все более текст, изображение, видео, звук и т. д. Одна новая и быстро растущая часть Data Mining — это анализ связей между данными (link analysis), который имеет приложения в таких разных областях, как биоинформатика, цифровые библиотеки и защита от терроризма.

Математический и статистический подходы являются основой для Data Mining.

СППР

Вычислительная техника создавалась прежде всего для обработки данных. В настоящее время современные вычислительные системы и компьютерные сети позволяют накапливать большие массивы данных для решения задач обработки и анализа. К сожалению, сама по себе машинная форма представления данных содержит информацию, необходимую человеку, в скрытом виде, и для ее извлечения нужно использовать специальные методы анализа данных.

Большой объем информации, с одной стороны, позволяет получить более точные расчеты и анализ, с другой — превращает поиск решений в сложную задачу.

Неудивительно, что первичный анализ данных был переложен на компьютер. В результате появился целый класс программных систем, призванных облегчить работу людей, выполняющих анализ (аналитиков). Такие системы принято называть системами поддержки принятия решений — СППР (DSS, Decision Support System).

Для выполнения анализа СППР должна накапливать информацию, обладая средствами ее ввода и хранения. Можно выделить три основные задачи, решаемые в СППР:

- ввод данных;
- хранение данных;
- анализ данных.

Таким образом, СППР — это системы, обладающие средствами ввода, хранения и анализа данных, относящихся к определенной предметной области, с целью поиска решений.

Ввод данных в СППР осуществляется либо автоматически от датчиков, характеризующих состояние среды или процесса, либо человеком-оператором.

В первом случае данные накапливаются путем циклического опроса или по сигналу готовности, возникающему при появлении информации. Во втором случае СППР должны предоставлять пользователям удобные средства ввода данных, контролирующие корректность вводимых данных и выполняющие сопутствующие вычисления. Если ввод осуществляется одновременно несколькими операторами, то система должна решать проблемы параллельного доступа и модификации одних и тех же данных.

Постоянное накопление данных приводит к непрерывному росту их объема. В связи с этим на СППР ложится задача обеспечить надежное хранение больших объемов данных. На СППР также могут быть возложены задачи предотвращения несанкционированного доступа, резервного хранения данных, архивирования и т. п.

Основная задача СППР — предоставить аналитикам инструмент для выполнения анализа данных. Необходимо отметить, что для эффективного использования СППР ее пользователь-аналитик должен обладать соответствующей квалификацией. Система не генерирует правильные решения, а только предоставляет аналитику данные в соответствующем виде (отчеты, таблицы, графики и т. п.) для изучения и анализа, именно поэтому такие системы обеспечивают выполнение функции поддержки принятия решений. Очевидно, что, с одной стороны, качество принятых решений зависит от квалификации аналитика. С другой стороны, рост объемов анализируемых данных, высокая скорость обработки и анализа, а также сложность использования машинной формы представления данных стимулируют исследования и разработку интеллектуальных СППР. Для таких СППР характерно наличие функций, реализующих отдельные умственные возможности человека.

По степени "интеллектуальности" обработки данных при анализе выделяют три класса задач анализа:

информационно-поисковый — СППР осуществляет поиск необходимых данных. Характерной чертой такого анализа является выполнение заранее определенных запросов;

оперативно-аналитический — СППР производит группирование и обобщение данных в любом виде, необходимом аналитику. В отличие от информационно-поискового анализа в данном случае невозможно заранее предсказать необходимые аналитику запросы;

интеллектуальный — СППР осуществляет поиск функциональных и логических закономерностей в накопленных данных, построение моделей и правил, которые объясняют найденные закономерности и/или прогнозируют развитие некоторых процессов (с определенной вероятностью).

Таким образом, обобщенная архитектура СППР может быть представлена следующим образом (рис. 1.2).

Рис. 1.2. Обобщенная архитектура системы поддержки принятия решений

Рассмотрим отдельные подсистемы более подробно.

Подсистема ввода данных.

В таких подсистемах, называемых OLTP (On-line transaction processing), выполняется операционная (транзакционная) обработка данных. Для реализации этих подсистем используют обычные системы управления базами данных (СУБД).

Подсистема хранения.

Для реализации данной подсистемы используют современные СУБД и концепцию хранилищ данных.

Подсистема анализа.

Данная подсистема может быть построена на основе:

- подсистемы информационно-поискового анализа на базе реляционных СУБД и статических запросов с использованием языка структурных запросов SQL (Structured Query Language);
- подсистемы оперативного анализа. Для реализации таких подсистем применяется технология оперативной аналитической обработки данных OLAP (On-line analytical processing), использующая концепцию многомерного представления данных;
- подсистемы интеллектуального анализа. Данная подсистема реализует методы и алгоритмы Data Mining ("добыча данных").

Базы данных

Ранее было отмечено, что для решения задач анализа данных и поиска решений необходимо накопление и хранение достаточно больших объемов данных. Этим целям служат базы данных (БД).

База данных является моделью некоторой предметной области, состоящей из связанных между собой данных об объектах, их свойствах и характеристиках.

Средства для работы с БД представляют СУБД. Не решая непосредственно никаких прикладных задач, СУБД является инструментом для разработки прикладных программ, использующих БД.

Чтобы сохранять данные согласно какой-либо модели предметной области, структура БД должна максимально соответствовать этой модели. Первой такой структурой, используемой в СУБД, была **иерархическая структура**, появившаяся в начале 60-х годов прошлого века. Иерархическая структура предполагала хранение данных в виде дерева. Это значительно упрощало создание и поддержку таких БД. Однако невозможность представить многие объекты реального мира в виде иерархии привела к использованию таких БД в сильно специализированных областях.

Попыткой улучшить иерархическую структуру была **сетевая структура** БД, которая предполагает представление данных в виде сети.

Графовая база данных — разновидность баз данных с реализацией сетевой модели в виде графа и его обобщений. Основными элементами модели являются узлы и связи. Графовые базы данных применяются для моделирования социальных графов (социальных сетей), в биоинформатике, а также для семантической паутины.

В графовых СУБД как правило разделяют хранилище (англ. underlying storage) и механизм обработки (англ. processing engine).

Для задач с естественной графовой структурой данных графовые СУБД могут существенно превосходить реляционные по производительности, а также иметь преимущества в наглядности представления и простоте внесения изменений в схему базы данных.

Наиболее распространены в настоящее время **реляционные** БД. Термин "реляционный" произошел от латинского слова "relatio" — отношение. Такая структура хранения данных построена на взаимоотношении составляющих ее частей. Реляционный подход стал широко известен благодаря работам английского математика Эдгара Кодда (Edgar Codd).

В 1985 году в двух статьях в журнале Computer World **Э. Кодд сформулировал правила**, которым должны соответствовать реляционные базы данных. Всего правил было двенадцать. Несколько позже Кодд сформулировал 13-е правило и присвоил ему номер 0. Начнем рассмотрение правил с этого последнего или нулевого.

Правило 0: Основное правило (Foundation Rule):

> Система, которая позиционируется как реляционная система управления базами данных, должна быть способна управлять базами данных, используя исключительно свои реляционные возможности.

Правило 1: Информационное правило (The Information Rule):

> Вся информация в реляционной базе данных на логическом уровне должна быть явно представлена единственным способом: значениями в таблицах. Отсюда кстати вытекает и условие отсутствия порядка строк и столбцов в таблицах.

Правило 2: Гарантированный доступ к данным (Guaranteed Access Rule):

> В реляционной базе данных каждое отдельное (атомарное) значение данных должно быть логически доступно с помощью комбинации имени таблицы, значения первичного ключа и имени столбца. Это требование предполагает:
>
> - Уникальность имени таблицы в базе данных.
> - Уникальность имени столбца в таблице.
> - Уникальность первичного ключа, по крайней мере, в пределах одной таблицы.
>
> В реальных СУБД могут присутствовать дополнительные параметры доступа, например имя пользователя, владельца данной базы. Кроме этого, поскольку система может работать с несколькими базами данных, в качестве еще одного параметра может быть имя базы данных. Наконец, если данные находятся под управлением разных СУБД (распределенная система) дополнительной координатой данных будет (адрес) СУБД.

Правило 3: Систематическая поддержка отсутствующих значений (Systematic Treatment of Null Values):

> Неизвестные, или отсутствующие значения NULL, отличные от любого известного значения, должны поддерживаться для всех типов данных при выполнении любых операций. Например, для числовых данных неизвестные значения не должны рассматриваться как нули, а для символьных данных — как пустые строки.

Правило 4: Доступ к словарю данных в терминах реляционной модели (Active On-Line Catalog Based on the Relational Model):

Словарь данных должен сохраняться в форме реляционных таблиц, и СУБД должна поддерживать доступ к нему при помощи стандартных языковых средств, тех же самых, которые используются для работы с реляционными таблицами, содержащими пользовательские данные. БД хранит два вида таблиц: пользовательские и системные. В пользовательских таблицах хранятся данные, введенные пользователем. В системных таблицах хранятся метаданные: описание таблиц (название, типы и размеры колонок), индексы, хранимые процедуры и др. Системные таблицы тоже доступны, т. е. пользователь может получить информацию о метаданных БД.

Правило 5: Полнота подмножества языка (Comprehensive Data Sublanguage Rule):

Система управления реляционными базами данных должна поддерживать хотя бы один реляционный язык, который

(а) имеет линейный синтаксис,

(б) может использоваться как интерактивно, так и в прикладных программах,

(в) поддерживает операции определения данных, определения представлений, манипулирования данными (интерактивные и программные), ограничители целостности, управления доступом и операции управления транзакциями (begin, commit, rollback). Сейчас таким инструментом стал язык SQL.

Правило 6: Возможность изменения представлений (View Updating Rule):

Каждое представление должно поддерживать все операции манипулирования данными, которые поддерживают реляционные таблицы: операции выборки, вставки, изменения и удаления данных. Представления являются динамическим объединением нескольких таблиц. Изменения данных в представлении должны автоматически переноситься на исходные таблицы (за исключением нередактируемых полей в представлении, например вычисляемых полей).

Правило 7: Наличие высокоуровневых операций управления данными (High-Level Insert, Update, and Delete):

Операции вставки, изменения и удаления данных должны поддерживаться не только по отношению к одной строке реляционной таблицы, но и по отношению к любому множеству строк.

Правило 8: Физическая независимость данных (Physical Data Independence):

Приложения не должны зависеть от используемых способов хранения данных на носителях, от аппаратного обеспечения компьютеров, на которых находится реляционная база данных.

Правило 9: Логическая независимость данных (Logical Data Independence):

Прикладное программное обеспечение не должно зависеть от изменений, вносимых в структуру базы данных, которые теоретически не должны

изменять хранящиеся в базе данные. Приложения, оперирующие с данными реляционных БД, не должны зависеть от организации связей между таблицами (логической организации). При изменении связей между таблицами не должны меняться ни сами таблицы, ни запросы к ним.

Правило 10: Независимость контроля целостности (Integrity Independence):

Под целостностью данных в общем случае понимается готовность БД к работе. Различают следующие типы целостности:

- физическая целостность — сохранность информации на носителях и корректность форматов хранения данных;
- логическая целостность — непротиворечивость и актуальность данных, хранящихся в БД.

Правило 11: Независимость от расположения (Distribution Independence):

База данных может быть распределённой, может находиться на нескольких компьютерах, и это не должно оказывать влияния на приложения. Перенос базы данных на другой компьютер не должен оказывать влияния на приложения.

Правило 12: Согласование языковых уровней (The Nonsubversion Rule):

Если используется низкоуровневый язык доступа к данным, он не должен игнорировать правила безопасности и правила целостности, которые поддерживаются языком более высокого уровня.

На практике в дополнение к перечисленным правилам существует также требование минимизации объемов памяти, занимаемых БД. Это достигается проектированием такой структуры БД, при которой дублирование (избыточность) информации было бы минимальным. Для выполнения этого требования была разработана теория нормализации. Она предполагает несколько уровней **нормализации** БД, каждый из которых базируется на предыдущем. Каждому уровню нормализации соответствует определенная нормальная форма (НФ). В зависимости от условий, которым удовлетворяет БД, говорят, что она имеет соответствующую нормальную форму. Например:

БД имеет 1-ю НФ, если каждое значение, хранящееся в ней, неразделимо на более примитивные (неразложимость значений);

БД имеет 2-ю НФ, если она имеет 1-ю НФ, и при этом каждое значение целиком и полностью зависит от ключа (функционально независимые значения);

БД имеет 3-ю НФ, если она имеет 2-ю НФ, и при этом ни одно из значений не предоставляет никаких сведений о другом значении (взаимно независимые значения) и т. д.

Реляционная модель, однако, имеет существенный недостаток – не каждый тип информации можно представить в табличной форме, например изображение, музыку и др.

OLTP-системы

В соответствии с правилами Кодда СУБД должна обеспечивать выполнение операций над БД, предоставляя при этом возможность одновременной работы нескольким пользователям (с нескольких компьютеров) и гарантируя целостность данных. Для выполнения этих правил в СУБД используется механизм управления транзакциями.

Транзакция — это последовательность операций над БД, рассматриваемых СУБД как единое целое. Транзакция переводит БД из одного целостного состояния в другое.

Как правило, транзакцию составляют операции, манипулирующие с данными, принадлежащими разным таблицам и логически связанными друг с другом. Если при выполнении транзакции будут выполнены операции, модифицирующие только часть данных, а остальные данные не будут изменены, то будет нарушена целостность. Следовательно, либо все операции, включенные в транзакцию, должны быть выполненными, либо не выполнена ни одна из них. Процесс отмены выполнения транзакции называется откатом транзакции (ROLLBACK). Сохранение изменений, производимых в результате выполнения операций транзакции, называется фиксацией транзакции (COMMIT).

Свойство транзакции переводить БД из одного целостного состояния в другое позволяет использовать понятие транзакции как единицу активности пользователя. В случае одновременного обращения пользователей к БД транзакции, инициируемые разными пользователями, выполняются не параллельно (что невозможно для одной БД), а в соответствии с некоторым планом ставятся в очередь и выполняются последовательно. Таким образом, для пользователя, по инициативе которого образована транзакция, присутствие транзакций других пользователей будет незаметно, если не считать некоторого замедления работы по сравнению с однопользовательским режимом.

История развития СУБД тесно связана с совершенствованием подходов к решению задач хранения данных и управления транзакциями. Развитый механизм управления транзакциями в современных СУБД сделал их основным средством построения OLTP-систем, основной задачей которых является обеспечение выполнения операций с БД.

OLTP (англ. *Online Transaction Processing*), **транзакционная система** — обработка транзакций в реальном времени. Способ организации БД, при котором система работает с небольшими по размерам транзакциями, но идущими большим потоком, и при этом клиенту требуется от системы минимальное время отклика.

OLTP-системы предназначены для ввода, структурированного хранения и обработки информации (операций, документов) в режиме реального времени.

OLTP-системы оперативной обработки транзакций характеризуются большим количеством изменений, одновременным обращением множества пользователей к одним и тем же данным для выполнения разнообразных операций — чтения,

записи, удаления или модификации данных. Для нормальной работы множества пользователей применяются блокировки и транзакции.

Эффективная обработка транзакций и поддержка блокировок входят в число важнейших требований к системам оперативной обработки транзакций.

К этому классу систем относятся, кстати, и первые СППР — информационные системы руководства (ИСР, Executive Information Systems). Такие системы, как правило, строятся на основе реляционных СУБД, включают в себя подсистемы сбора, хранения и информационно-поискового анализа информации, а также содержат предопределенное множество запросов для повседневной работы. Каждый новый запрос, непредусмотренный при проектировании такой системы, должен быть сначала формально описан, закодирован программистом и только затем выполнен. Время ожидания в этом случае может составлять часы и дни, что неприемлемо для оперативного принятия решений.

Практика использования OLTP-систем показала неэффективность их применения для полноценного анализа информации. Такие системы достаточно успешно решают задачи сбора, хранения и поиска информации, но они не удовлетворяют требованиям, предъявляемым к современным СППР. Подходы, связанные с наращиванием функциональности OLTP-систем, не дали удовлетворительных результатов. Основной причиной неудачи является противоречивость требований, предъявляемых к системам OLTP и СППР.

Хранилище данных

Противоречивость требований к OLTP-системам и системам, ориентированным на глубокий анализ информации, усложняет задачу их интеграции как подсистем единой СППР. В настоящее время наиболее популярным решением этой проблемы является подход, ориентированный на использование концепции хранилищ данных.

Общая идея хранилищ данных заключается в разделении БД для OLTP-систем и БД для выполнения анализа и последующем их проектировании с учетом соответствующих требований.

Стремление объединить в одной архитектуре СППР возможности OLTP-систем и систем анализа, требования к которым во многом противоречивы, привело к появлению концепции хранилищ данных (ХД).

Концепция ХД так или иначе обсуждалась специалистами в области информационных систем достаточно давно. Первые статьи, посвященные именно ХД, появились в 1988 г., их авторами были Б. Девлин и П. Мэрфи. В 1992 г. У. Инмон подробно описал данную концепцию в своей монографии "Построение хранилищ данных" ("Building the Data Warehouse", second edition — QED Publishing Group, 1996).

В основе концепции ХД лежит идея разделения данных, используемых для оперативной обработки и для решения задач анализа. Это позволяет применять структуры данных, которые удовлетворяют требованиям их хранения с учетом использования в OLTP-системах и системах анализа. Такое разделение позволяет

оптимизировать как структуры данных оперативного хранения (оперативные БД, файлы, электронные таблицы и т. п.) для выполнения операций ввода, модификации, удаления и поиска, так и структуры данных, используемые для анализа (для выполнения аналитических запросов). В СППР эти два типа данных называются соответственно **оперативными источниками данных (ОИД)** и хранилищем данных.

В своей работе Инмон дал следующее определение ХД.

Хранилище данных — предметно-ориентированный, интегрированный, неизменчивый, поддерживающий хронологию набор данных, организованный для целей поддержки принятия решений.

Рассмотрим свойства ХД более подробно.

Предметная ориентация. Это фундаментальное отличие ХД от ОИД. Разные ОИД могут содержать данные, описывающие одну и ту же предметную область с разных точек зрения (например, с точки зрения бухгалтерского учета, складского учета, планового отдела и т. п.). Решение, принятое на основе только одной точки зрения, может быть неэффективным или даже неверным. ХД позволяют интегрировать информацию, отражающую разные точки зрения на одну предметную область. Предметная ориентация позволяет также хранить в ХД только те данные, которые нужны для их анализа (например, для анализа нет смысла хранить информацию о номерах документов купли-продажи, в то время как их содержимое — количество, цена проданного товара — необходимо). Это существенно сокращает затраты на носители информации и повышает безопасность доступа к данным.

Интеграция. ОИД, как правило, разрабатываются в разное время несколькими коллективами с собственным инструментарием. Это приводит к тому, что данные, отражающие один и тот же объект реального мира в разных системах, описывают его по-разному. Обязательная интеграция данных в ХД позволяет решить эту проблему, приведя данные к единому формату.

Поддержка хронологии. Данные в ОИД необходимы для выполнения над ними операций в текущий момент времени. Поэтому они могут не иметь привязки ко времени. Для анализа данных часто бывает важно иметь возможность отслеживать хронологию изменений показателей предметной области. Поэтому все данные, хранящиеся в ХД, должны соответствовать последовательным интервалам времени.

Неизменяемость. Требования к ОИД накладывают ограничения на время хранения в них данных. Те данные, которые не нужны для оперативной обработки, как правило, удаляются из ОИД для уменьшения занимаемых ресурсов. Для анализа, наоборот, требуются данные за максимально большой период времени. Поэтому, в отличие от ОИД, данные в ХД после загрузки только читаются. Это позволяет существенно повысить скорость доступа к данным, как за счет возможной избыточности хранящейся информации, так и за счет исключения операций модификации.

При реализации в СППР концепции ХД данные из разных ОИД копируются в единое хранилище. Собранные данные приводятся к единому формату, согласовываются и обобщаются. Аналитические запросы адресуются к ХД (рис. 1). Такая модель неизбежно приводит к дублированию информации в ОИД и в ХД. Однако Инмон в своей работе утверждает, что избыточность данных, хранящихся в СППР, не превышает 1%! Это можно объяснить следующими причинами.

При загрузке информации из ОИД в ХД данные фильтруются. Многие из них не попадают в ХД, поскольку лишены смысла с точки зрения использования в процедурах анализа. Информация в ОИД носит, как правило, оперативный характер, и данные, потеряв актуальность, удаляются. В ХД, напротив, хранится историческая информация. С этой точки зрения дублирование содержимого ХД данными ОИД оказывается весьма незначительным. В ХД хранится обобщенная информация, которая в ОИД отсутствует.

Во время загрузки в ХД данные очищаются (удаляется ненужная информация), и после такой обработки они занимают гораздо меньший объем.

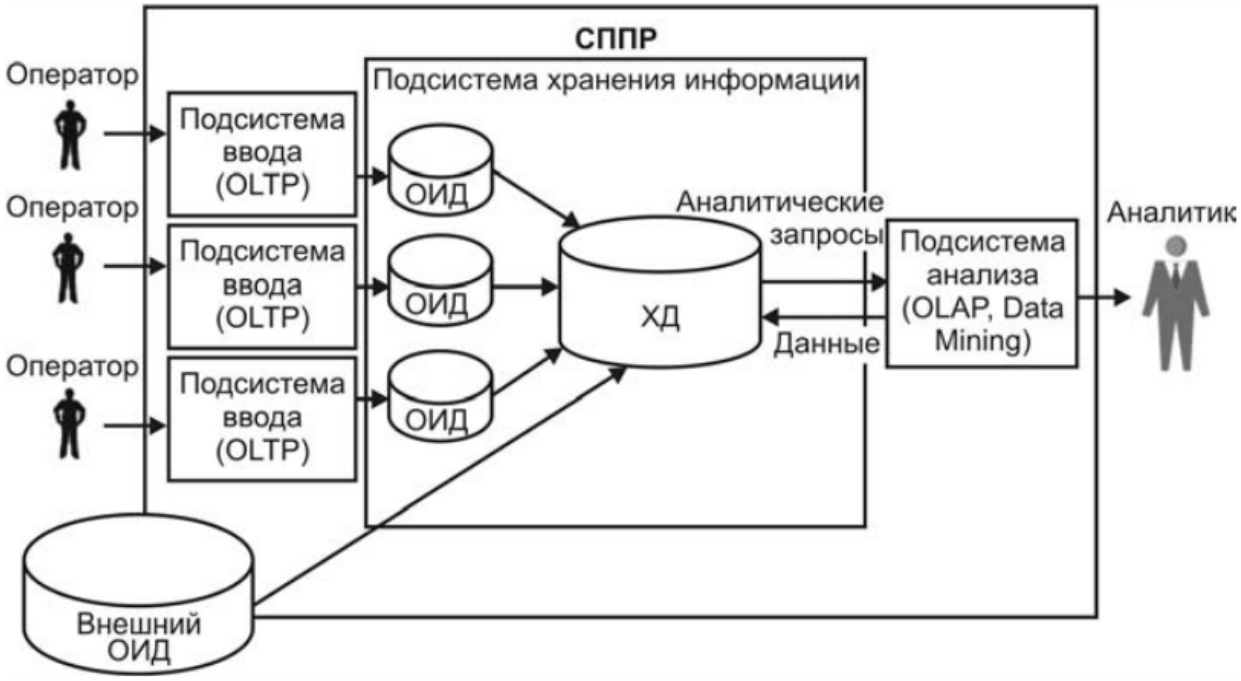

Рисунок 1. Структура СППР с физическим ХД

Избыточность информации можно свести к нулю, используя **виртуальное** ХД.

В данном случае в отличие от классического (физического) ХД данные из ОИД не копируются в единое хранилище. Они извлекаются, преобразуются и интегрируются непосредственно при выполнении аналитических запросов в оперативной памяти компьютера. Фактически такие запросы напрямую адресуются к ОИД (рис. 2). Основными достоинствами виртуального ХД являются:

- минимизация объема памяти, занимаемой на носителе информацией;
- работа с текущими, детализированными данными.

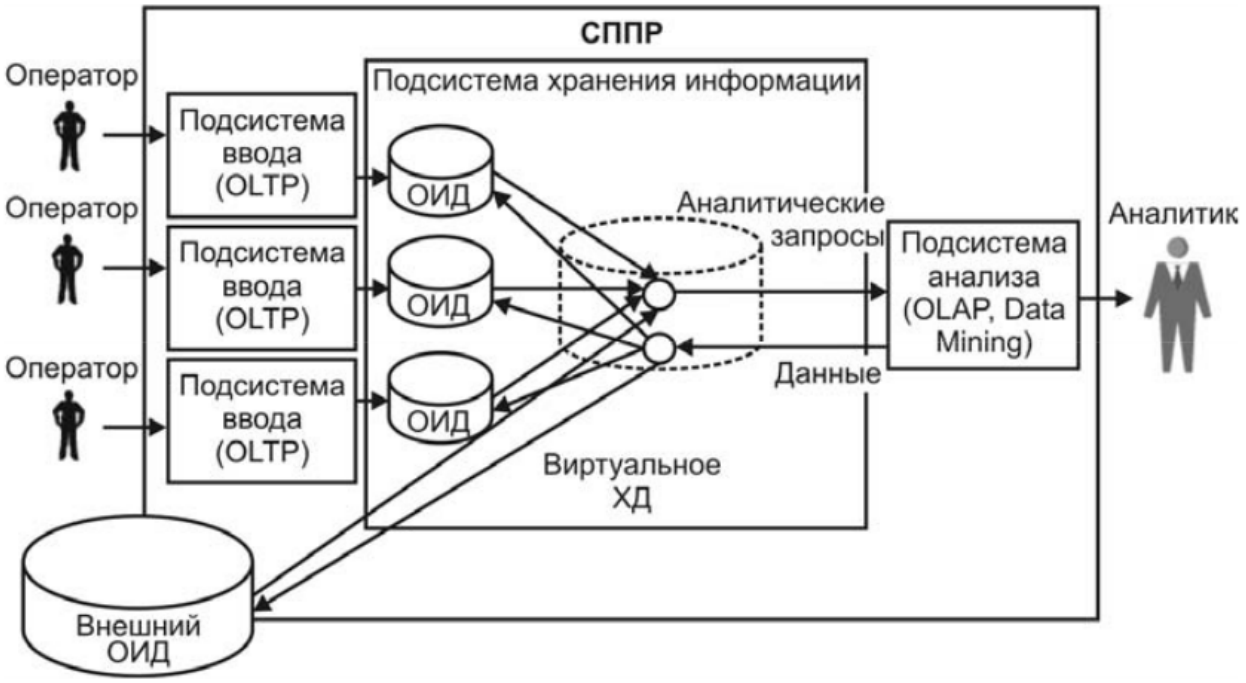

Рисунок 2. Структура СППР с виртуальным ХД

Однако такой подход обладает многими недостатками. Время обработки запросов к виртуальному ХД значительно превышает соответствующие показатели для физического хранилища. Кроме того, структуры оперативных БД, рассчитанные на интенсивное обновление одиночных записей, в высокой степени нормализованы. Для выполнения же аналитического запроса требуется объединение большого числа таблиц, что также приводит к снижению быстродействия.

Интегрированный взгляд на виртуальное хранилище возможен только при выполнении условия постоянной доступности всех ОИД. Таким образом, временная недоступность хотя бы одного из источников может привести либо к невыполнению аналитического запроса, либо к неверным результатам.

Выполнение сложных аналитических запросов над ОИД требует значительных ресурсов компьютеров. Это приводит к снижению быстродействия OLTP-систем, что недопустимо, т. к. время выполнения операций в таких системах часто весьма критично.

Различные ОИД могут поддерживать разные форматы и кодировки данных.

Часто на один и тот же вопрос может быть получено несколько вариантов ответа. Это может быть связано с несинхронностью моментов обновления данных в разных ОИД, отличиями в описании одинаковых объектов и событий предметной области, ошибками при вводе, утерей фрагментов архивов и т. д. В таком случае цель — формирование единого непротиворечивого взгляда на объект управления — может быть не достигнута.

Главным же недостатком виртуального хранилища следует признать практическую невозможность получения данных за долгий период времени. При отсутствии физического хранилища доступны только те данные, которые на

момент запроса есть в ОИД. Основное назначение OLTP-систем — оперативная обработка текущих данных, поэтому они не ориентированы на хранение данных за длительный период времени. По мере устаревания данные выгружаются в архив и удаляются из оперативной БД.

Несмотря на преимущества физического ХД перед виртуальным, необходимо признать, что его реализация представляет собой достаточно трудоемкий процесс. Остановимся на основных **проблемах создания ХД**:

- необходимость интеграции данных из неоднородных источников в распределенной среде;
- потребность в эффективном хранении и обработке очень больших объемов информации;
- необходимость наличия многоуровневых справочников метаданных;
- повышенные требования к безопасности данных.

Рассмотрим эти проблемы более подробно.

Необходимость интеграции данных из неоднородных источников в распределенной среде. ХД создаются для интегрирования данных, которые могут поступать из разнородных ОИД, физически размещающихся на разных компьютерах: БД, электронных архивов, публичных и коммерческих электронных каталогов, справочников, статистических сборников. При создании ХД приходится решать задачу построения системы, согласованно функционирующей с неоднородными программными средствами и решениями. При выборе средств реализации ХД приходится учитывать множество факторов, включающих уровень совместимости различных программных компонентов, легкость их освоения и использования, эффективность функционирования и т. д.

Потребность в эффективном хранении и обработке очень больших объемов информации. Свойство неизменности ХД предполагает накопление в нем информации за долгий период времени, что должно поддерживаться постоянным ростом объемов дисковой памяти. Ориентация на выполнение аналитических запросов и связанная с этим денормализация данных приводят к нелинейному росту объемов памяти, занимаемой ХД при возрастании объема данных. Исследования, проведенные на основе тестового набора TPC-D, показали, что для баз данных объемом в 100 Гбайт требуется память, в 4,87 раза бо́льшая объемом, чем нужно для хранения полезных данных.

Необходимость многоуровневых справочников метаданных. Для систем анализа наличие развитых метаданных (данных о данных) и средств их предоставления конечным пользователям является одним из основных условий успешной реализации ХД. Метаданные необходимы пользователям СППР для понимания структуры информации, на основании которой принимается решение. Например, прежде чем менеджер корпорации задаст системе свой вопрос, он должен понять, какая информация имеется, насколько она актуальна, можно ли ей доверять, сколько времени может занять формирование ответа и т. д. При

создании ХД необходимо решать задачи хранения и удобного представления метаданных пользователям.

Повышение требований к безопасности данных. Собранная вместе и согласованная информация об истории развития корпорации, ее успехах и неудачах, о взаимоотношениях с поставщиками и заказчиками, об истории и состоянии рынка дает возможность анализа прошлой и текущей деятельности корпорации и построения прогнозов для будущего. Очевидно, что подобная информация является конфиденциальной и доступ к ней ограничен в пределах самой компании, не говоря уже о других компаниях. Для обеспечения безопасности данных приходится решать вопросы аутентификации пользователей, защиты данных при их перемещении в хранилище данных из оперативных баз данных и внешних источников, защиты данных при их передаче по сети и т. п.

Снижения затрат на создание ХД можно добиться, создавая его упрощенный вариант — витрину данных (Data Mart).

Витрина данных (англ. Data Mart; другие варианты перевода: хранилище данных специализированное, киоск данных, рынок данных) — срез хранилища данных, представляющий собой массив тематической, узконаправленной информации, ориентированный, например, на пользователей одной рабочей группы или департамента.

ВД максимально приближена к конечному пользователю и содержит данные, тематически ориентированные на него (например, ВД для работников отдела маркетинга может содержать данные, необходимые для маркетингового анализа). ВД существенно меньше по объему, чем ХД, и для ее реализации не требуется больших затрат. Они могут быть реализованы как самостоятельно, так и вместе с ХД.

Самостоятельные ВД (рис. 3) часто появляются в организации исторически и встречаются в крупных организациях с большим количеством независимых подразделений, решающих собственные аналитические задачи. Достоинствами такого подхода являются:

- проектирование ВД для ответов на определенный круг вопросов;
- быстрое внедрение автономных ВД и получение отдачи;
- упрощение процедур заполнения ВД и повышение их производительности за счет учета потребностей определенного круга пользователей.

Недостатками автономных ВД являются:

- многократное хранение данных в разных ВД, что приводит к увеличению расходов на их хранение и потенциальным проблемам, связанным с необходимостью поддержания непротиворечивости данных;
- отсутствие консолидированности данных на уровне предметной области, а следовательно — отсутствие единой картины.

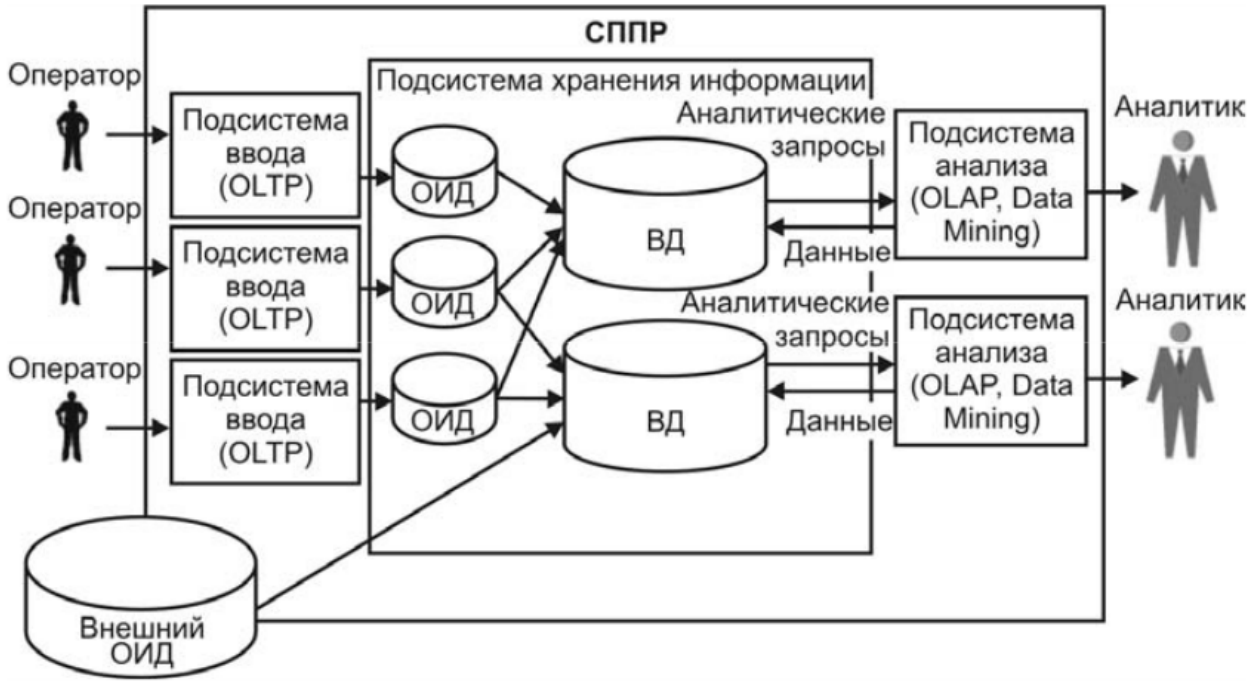

Рисунок 3. Структура СППР с самостоятельными ВД

В последнее время все более популярной становится идея совместить ХД и ВД в одной системе. В этом случае ХД используется в качестве единственного источника интегрированных данных для всех ВД (рис. 4).

ХД представляет собой единый централизованный источник информации для всей предметной области, а ВД являются подмножествами данных из хранилища, организованными для представления информации по тематическим разделам данной области. Конечные пользователи имеют возможность доступа к детальным данным хранилища, если данных в витрине недостаточно, а также для получения более полной информационной картины.

Достоинствами такого подхода являются:

- простота создания и наполнения ВД, поскольку наполнение происходит из единого стандартизованного надежного источника очищенных данных — из ХД;
- простота расширения СППР за счет добавления новых ВД;
- снижение нагрузки на основное ХД.

К недостаткам относятся:

- избыточность (данные хранятся как в ХД, так и в ВД);
- дополнительные затраты на разработку СППР с ХД и ВД.

В случае архитектур с физическим ХД и/или ВД необходимо уделить внимание вопросам организации (архитектуры) ХД и переносу данных из ОИД в ХД.

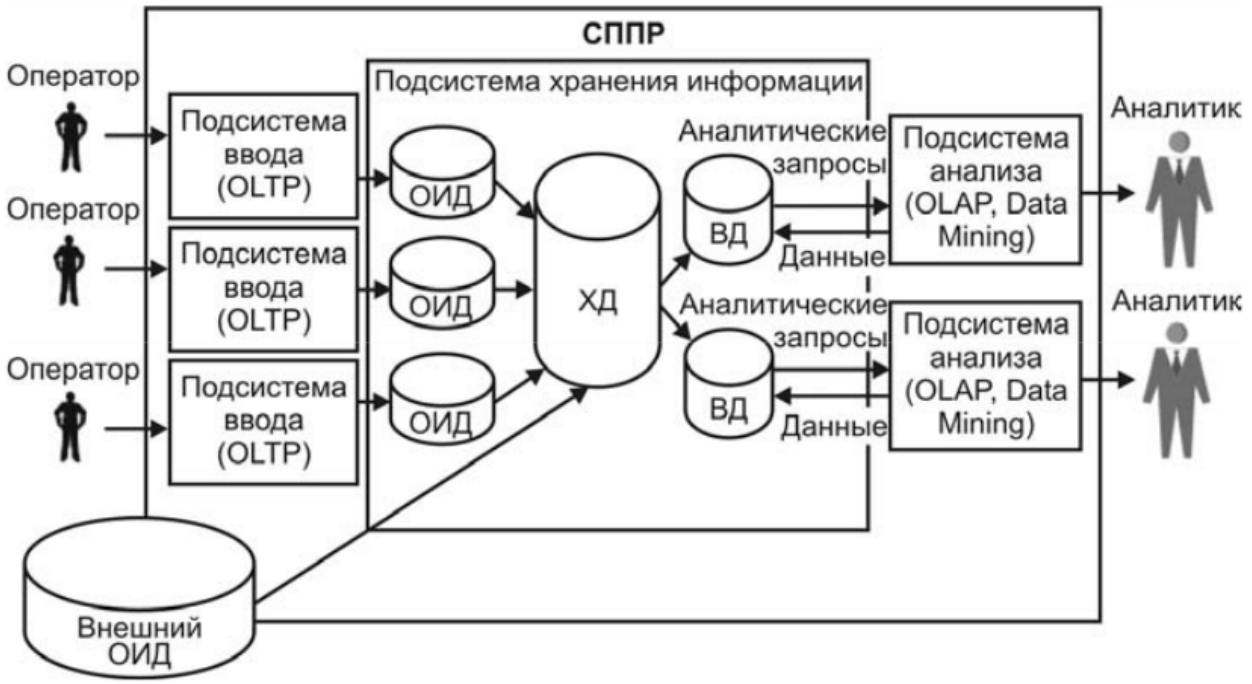

Рисунок 4. Структура СППР с ХД и ВД

Организация ХД

Все данные в ХД делятся на три основные категории (рис. 5):

- детальные данные;
- агрегированные данные;
- метаданные.

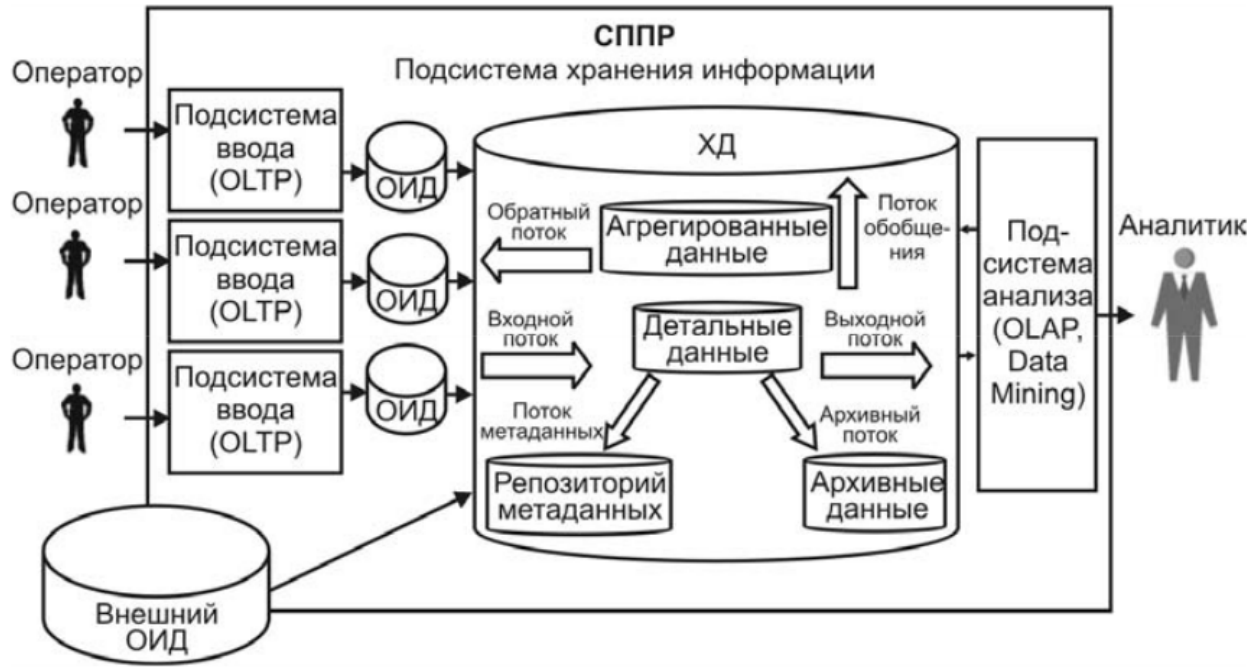

Рисунок 5. Архитектура ХД

Детальными являются данные, переносимые непосредственно из ОИД. Они соответствуют элементарным событиям, фиксируемым OLTP-системами (например, продажи, эксперименты и др.). Принято разделять все данные на измерения и факты. **Измерениями** называются наборы данных, необходимые для описания событий (например, города, товары, люди и т. п.). **Фактами** называются данные, отражающие сущность события (например, количество проданного товара, результаты экспериментов и т. п.). Фактические данные могут быть представлены в виде числовых или категориальных значений.

В процессе эксплуатации ХД необходимость в ряде детальных данных может снизиться. Ненужные детальные данные могут храниться в архивах в сжатом виде на более емких накопителях с более медленным доступом (например, на магнитных лентах). Данные в архиве остаются доступными для обработки и анализа. Регулярно используемые для анализа данные должны храниться на накопителях с быстрым доступом (например, на жестких дисках).

На основании детальных данных могут быть получены **агрегированные** (обобщенные) данные. Агрегирование происходит путем суммирования числовых фактических данных по определенным измерениям. В зависимости от возможности агрегировать данные они подразделяются на следующие типы:

- **аддитивные** — числовые фактические данные, которые могут быть просуммированы по всем измерениям;
- **полуаддитивные** — числовые фактические данные, которые могут быть просуммированы только по определенным измерениям;
- **неаддитивные** — фактические данные, которые не могут быть просуммированы ни по одному измерению.

Большинство пользователей СППР работают не с детальными, а с агрегированными данными. Архитектура ХД должна предоставлять быстрый и удобный способ получать интересующую пользователя информацию. Для этого необходимо часть агрегированных данных хранить в ХД, а не вычислять их при выполнении аналитических запросов. Очевидно, что это ведет к избыточности информации и увеличению размеров ХД. Поэтому при проектировании таких систем важно добиться оптимального соотношения между вычисляемыми и хранящимися агрегированными данными. Те данные, к которым редко обращаются пользователи, могут вычисляться в процессе выполнения аналитических запросов. Данные, которые требуются более часто, должны храниться в ХД.

Для удобства работы с ХД необходима информация о содержащихся в нем данных. Такая информация называется **метаданными** (данные о данных). Согласно концепции Дж. Захмана, метаданные должны отвечать на следующие вопросы — что, кто, где, как, когда и почему:

- *что* (описание объектов) — метаданные описывают объекты предметной области, информация о которых хранится в ХД. Такое описание включает: атрибуты объектов, их возможные значения, соответствующие поля в информационных структурах ХД, источники информации об объектах и т.п.;

- *кто* (описание пользователей) — метаданные описывают категории пользователей, использующих данные. Они описывают права доступа к данным, а также включают в себя сведения о пользователях, выполнявших над данными различные операции (ввод, редактирование, загрузку, извлечение и т. п.);
- *где* (описание места хранения) — метаданные описывают местоположение серверов, рабочих станций, ОИД, размещенные на них программные средства и распределение между ними данных;
- *как* (описание действий) — метаданные описывают действия, выполняемые над данными. Описываемые действия могли выполняться как в процессе переноса из ОИД (например, исправление ошибок, расщепление полей и т. п.), так и в процессе их эксплуатации в ХД;
- *когда* (описание времени) — метаданные описывают время выполнения разных операций над данными (например, загрузка, агрегирование, архивирование, извлечение и т. п.);
- *почему* (описание причин) — метаданные описывают причины, повлекшие выполнение над данными тех или иных операций. Такими причинами могут быть требования пользователей, статистика обращений к данным и т. п.

Так как метаданные играют важную роль в процессе работы с ХД, то к ним должен быть обеспечен удобный доступ. Для этого они сохраняются в репозитории метаданных с удобным для пользователя интерфейсом.

Данные, поступающие из ОИД в ХД, перемещаемые внутри ХД и поступающие из ХД к аналитикам, образуют следующие информационные потоки (см. рис. 5):

- входной поток (Inflow) — образуется данными, копируемыми из ОИД в ХД;
- поток обобщения (Upflow) — образуется агрегированием детальных данных и их сохранением в ХД;
- архивный поток (Downflow) — образуется перемещением детальных данных, количество обращений к которым снизилось;
- поток метаданных (MetaFlow) — образуется переносом информации о данных в репозиторий данных;
- выходной поток (Outflow) — образуется данными, извлекаемыми пользователями;
- обратный поток (Feedback Flow) — образуется очищенными данными, записываемыми обратно в ОИД.

Самый мощный из информационных потоков — входной — связан с переносом данных из ОИД. Обычно информация не просто копируется в ХД, а подвергается обработке: данные очищаются и обогащаются за счет добавления новых атрибутов. Исходные данные из ОИД объединяются с информацией из внешних источников — текстовых файлов, сообщений электронной почты, электронных

таблиц и др. При разработке ХД не менее 60% всех затрат связано с переносом данных.

Процесс переноса, включающий в себя этапы извлечения, преобразования и загрузки, называют **ETL-процессом** (E — extraction, T — transformation, L — loading: извлечение, преобразование и загрузка соответственно). Программные средства, обеспечивающие его выполнение, называются ETL-системами.

Традиционно ETL-системы использовались для переноса информации из устаревших версий информационных систем в новые. В настоящее время ETL-процесс находит все большее применение для переноса данных из ОИД в ХД и ВД.

Рассмотрим более подробно этапы ETL-процесса.

Извлечение данных. Чтобы начать ETL-процесс, необходимо извлечь данные из одного или нескольких источников и подготовить их к этапу преобразования. Можно выделить два способа извлечения данных:

- Извлечение данных вспомогательными программными средствами непосредственно из структур хранения информации (файлов, электронных таблиц, БД и т. п.). Достоинствами такого способа извлечения данных являются:
    - отсутствие необходимости расширять OLTP-систему (это особенно важно, если ее структура закрыта);
    - данные могут извлекаться с учетом потребностей процесса переноса.
- Выгрузка данных средствами OLTP-систем в промежуточные структуры. Достоинствами такого подхода являются:
    - возможность использовать средства OLTP-систем, адаптированные к структурам данных;
    - средства выгрузки изменяются вместе с изменениями OLTP-систем и ОИД;
    - возможность выполнения первого шага преобразования данных за счет определенного формата промежуточной структуры хранения данных.

Преобразование данных. После того как сбор данных завершен, необходимо преобразовать их для размещения на новом месте. На этом этапе выполняются следующие процедуры:

- **обобщение данных (aggregation)** — перед загрузкой данные обобщаются. Процедура обобщения заменяет многочисленные детальные данные относительно небольшим числом агрегированных данных. Например, предположим, что данные о продажах за год занимают в нормализованной базе данных несколько тысяч записей. После обобщения данные преобразуются в меньшее число кратких записей, которые будут перенесены в ХД;

- **перевод значений (value translation)** — в ОИД данные часто хранятся в закодированном виде для того, чтобы сократить избыточность данных и память для их хранения. Например, названия товаров, городов, специальностей и т. п. могут храниться в сокращенном виде. Поскольку ХД содержат обобщенную информацию и рассчитаны на простое использование, закодированные данные обычно заменяют на более понятные описания;
- **создание полей (field derivation)** — при создании полей для конечных пользователей создается и новая информация. Например, ОИД содержит одно поле для указания количества проданных товаров, а второе — для указания цены одного экземпляра. Для исключения операции вычисления стоимости всех товаров можно создать специальное поле для ее хранения во время преобразования данных;
- **очистка данных (cleaning)** — направлена на выявление и удаление ошибок и несоответствий в данных с целью улучшения их качества. Проблемы с качеством встречаются в отдельных ОИД, например, в файлах и БД могут быть ошибки при вводе, отдельная информация может быть утрачена, могут присутствовать "загрязнения" данных и др. Очистка также применяется для согласования атрибутов полей таким образом, чтобы они соответствовали атрибутам базы данных назначения.

Загрузка данных. После того как данные преобразованы для размещения в ХД, осуществляется этап их загрузки. При загрузке выполняется запись преобразованных детальных и агрегированных данных. Кроме того, при записи новых детальных данных часть старых данных может переноситься в архив.

Очистка данных

Одной из важных задач, решаемых при переносе данных в ХД, является их очистка. С одной стороны, данные загружаются постоянно из различных источников, поэтому вероятность попадания "грязных данных" весьма высока. С другой стороны, ХД используются для принятия решений, и "грязные данные" могут стать причиной принятия неверных решений. Таким образом, процедура очистки является обязательной при переносе данных из ОИД в ХД. Ввиду большого спектра возможных несоответствий в данных их очистка считается одной из самых крупных проблем в технологии ХД. Основные проблемы очистки данных можно классифицировать по следующим уровням:

- уровень ячейки таблицы;
- уровень записи;
- уровень таблицы БД;
- уровень одиночной БД;
- уровень множества БД.

Рассмотрим перечисленные уровни и соответствующие им проблемы более подробно.

Уровень ячейки таблицы. На данном уровне задача очистки заключается в анализе и исправлении ошибок в данных, хранящихся в ячейках таблиц БД. К таким ошибкам можно отнести следующие.

- **Орфографические ошибки (опечатки)** — возникают при вводе информации. Они могут привести к неправильному пониманию, а также к искажению реальных данных. Например, при продаже товара вместо количества 1 000 было введено 10 000 или вместо названия товара "Водка" было введено название "Вода".
- **Отсутствие данных** — такие ошибки происходят из-за отсутствия у оператора соответствующих данных при вводе информации. Главной задачей OLTP-систем является обеспечение ежедневных операций с данными, поэтому оператор может пропустить ввод неизвестных ему данных, а не тратить время на их выяснение. Как следствие, в БД могут оставаться незаполненные ячейки (содержащие значение NULL).
- **Фиктивные значения** — это значения, введенные оператором, но не имеющие смысла. Наиболее часто такая проблема встречается в полях, обязательных для заполнения, но при отсутствии у оператора реальных данных он вынужден вводить бессмысленные данные, например: номер социального страхования 999-99-9999, или возраст клиента 999, или почтовый индекс 99999. Проблема усугубляется, если существует вероятность появления реальных данных, которые могут быть приняты за фиктивные, например, номер социального страхования 888-88-8888 для указания на статус клиента-иностранца "нерезидент" или месячный доход в размере $99 999,99 для указания на то, что клиент имеет работу.
- **Логически неверные значения** — значения, не соответствующие логическому смыслу, вкладываемому в данное поле таблицы. Например, в поле "Город" находится значение "Россия", или в поле "температура больного" значение 10.
- **Закодированные значения** — сокращенная запись или кодировка реальных данных, используемая для уменьшения занимаемого места.
- **Составные значения** — значения, содержащие несколько логических данных в одной ячейке таблицы. Такая ситуация возможна в полях произвольного формата (например, строковых или текстовых). Проблема усугубляется, если отсутствует строгий формат записи информации в такие поля.

Уровень записи. На данном уровне возникает проблема противоречивости значений в разных полях записи, описывающей один и тот же объект предметной области, например, когда возраст человека не соответствует его году рождения:

```
age=22, bdate=12.02.50.
```

Уровень таблицы БД. На данном уровне возникают проблемы, связанные с несоответствием информации, хранящейся в таблице и относящейся к разным объектам. На этом уровне наиболее часто встречаются следующие проблемы.

- Нарушение уникальности. Значения, соответствующие уникальным атрибутам разных объектов предметной области, являются одинаковыми.
- Отсутствие стандартов. Из-за отсутствия стандартов на формат записи значений могут возникать проблемы, связанные с дублированием данных или с их противоречивостью:
    - дублирующиеся записи (один и тот же человек записан в таблицу два раза, хотя значения полей уникальны):

```
emp1=(name="John Smith", ...);
emp2=(name="J.Smith", ...);
```

    - противоречивые записи (об одном человеке в разных случаях введена разная информация о дате рождения, хотя значения полей уникальны):

```
emp1=(name="John Smith", bdate=12.02.70);
emp2=(name="J.Smith", bdate=12.12.70);
```

Уровень одиночной БД. На данном уровне, как правило, возникают проблемы, связанные с нарушением целостности данных.

Уровень множества БД. На этом уровне возникают проблемы, связанные с неоднородностью как структур БД, так и хранящейся в них информации. Можно выделить следующие основные проблемы этого уровня:

- различие структур БД (различие наименований полей, типов, размеров и др.);
- в разных БД существуют одинаковые наименования разных атрибутов;
- в разных БД одинаковые данные представлены по-разному;
- в разных БД классификация элементов разная;
- в разных БД временная градация разная;
- в разных БД ключевые значения, идентифицирующие один и тот же объект предметной области, разные и т. п.

При решении задачи очистки данных, прежде всего, необходимо отдавать себе отчет в том, что не все проблемы могут быть устранены. Возможны ситуации, когда данные не существуют и не могут быть восстановлены, вне зависимости от количества приложенных усилий. Встречаются ситуации, когда значения настолько запутаны или найдены в стольких несопоставимых местах с такими на вид различными и противоположными значениями одного и того же факта, что любая попытка расшифровать эти данные может породить еще более неверные результаты, и, возможно, лучшим решением будет отказаться от их обработки. На самом деле не все данные нужно очищать. Как уже отмечалось, процесс очистки требует больших затрат, поэтому те данные, достоверность которых не влияет на процесс принятия решений, могут оставаться неочищенными.

В целом, очистка данных включает в себя несколько этапов:

- выявление проблем в данных;
- определение правил очистки данных;
- тестирование правил очистки данных;
- непосредственная очистка данных.

Выявление проблем в данных. Для выявления подлежащих удалению видов ошибок и несоответствий необходим подробный анализ данных. Наряду с ручной проверкой следует использовать аналитические программы. Существует два взаимосвязанных метода анализа: профайлинг данных и Data Mining.

- Профайлинг данных ориентирован на грубый анализ отдельных атрибутов данных. При этом происходит получение, например, такой информации, как тип, длина, спектр значений, дискретные значения данных и их частота, изменение, уникальность, наличие неопределенных значений, типичных строковых моделей (например, для номеров телефонов) и др., что позволяет обеспечить точное представление различных аспектов качества атрибута.
- Data Mining помогает найти специфические модели в больших наборах данных, например отношения между несколькими атрибутами. Именно на это направлены так называемые описательные модели Data Mining, включая группировку, обобщение, поиск ассоциаций и последовательностей. При этом могут быть получены ограничения целостности в атрибутах, например, функциональные зависимости или характерные для конкретных приложений бизнес-правила, которые можно использовать для восполнения утраченных и исправления недопустимых значений, а также для выявления дубликатов записей в источниках данных. Например, правило объединения с высокой вероятностью может предсказать проблемы с качеством данных в элементах данных, нарушающих это правило. Таким образом, 99-процентная вероятность правила "итого = количество × единицу" демонстрирует несоответствие и потребность в более детальном исследовании для оставшегося 1 процента записей.

Определение правил очистки данных. В зависимости от числа источников данных, степени их неоднородности и загрязненности, они могут требовать достаточно обширного преобразования и очистки. Первые шаги по очистке данных могут скорректировать проблемы отдельных источников данных и подготовить данные для интеграции. Дальнейшие шаги должны быть направлены на интеграцию данных и устранение проблем множественных источников. На этом этапе необходимо выработать общие правила преобразования, часть из которых должна быть представлена в виде программных средств очистки.

Тестирование правил очистки данных. Корректность и эффективность правил очистки данных должны тестироваться и оцениваться, например, на копиях данных источника. Это необходимо для выяснения целесообразности корректировки правил с целью их улучшения или исправления ошибок.

Этапы определения правил и их тестирование могут выполняться итерационно несколько раз, например, из-за того, что некоторые ошибки становятся заметны только после определенных преобразований.

Непосредственная очистка данных. На этом этапе выполняются преобразования в соответствии с определенными ранее правилами. Очистка выполняется в два приема. Сначала устраняются проблемы, связанные с отдельными источниками данных, а затем — проблемы множества БД.

Очищенные данные сохраняются в ХД и могут использоваться для анализа и принятия на их основе решений. За формирование аналитических запросов к данным и представление результатов их выполнения в СППР отвечают подсистемы анализа. От вида анализа также зависит и непосредственная реализация структур хранения данных в ХД.

Концепция ХД не является законченным архитектурным решением СППР и тем более не является готовым программным продуктом. Цель концепции ХД — определить требования к данным, помещаемым в ХД, общие принципы и этапы построения ХД, основные источники данных, дать рекомендации по решению потенциальных проблем, возникающих при выгрузке, очистке, согласовании, транспортировке и загрузке данных.

Необходимо понимать, что концепция ХД:

- это не концепция анализа данных, скорее, это концепция подготовки данных для анализа;
- не предопределяет архитектуру целевой аналитической системы. Концепция ХД указывает на то, какие процессы должны выполняться в системе, но не где конкретно и как они будут выполняться.

Таким образом, концепция ХД определяет лишь самые общие принципы построения аналитической системы и в первую очередь сконцентрирована на свойствах и требованиях к данным, но не на способах организации и представления данных в целевой БД и режимах их использования. Концепция ХД описывает построение аналитической системы, но не определяет характер ее использования. Она не решает ни одну из следующих проблем:

- выбор наиболее эффективного для анализа способа организации данных;
- организация доступа к данным;
- использование технологии анализа.

Проблемы использования собранных данных решают подсистемы анализа.

Литература:

Анализ данных и процессов : учеб. пособие / А. А. Барсегян, М. С. Куприянов, И. И. Холод, М. Д. Тесс, С. И. Елизаров. — 3-е изд., перераб. и доп. — СПб : БХВ-Петербург, 2009. — 512 с.

Пирогов, В.Ю. Информационные системы и базы данных: организация и проектирование : учеб. пособие / В.Ю. Пирогов. — СПб : БХВ-Петербург, 2009. — 528 с.