БАЗЫ ДАННЫХ. ИНТЕЛЛЕКТУАЛЬНАЯ ОБРАБОТКА ИНФОРМАЦИИ

Базы данных. Интеллектуальная обработка информации / В.В. Корнеев, А.Ф. Гареев, С.В. Васютин, В.В. Райх. — Москва: Нолидж, 2003. — 400 с.

Принципы построения систем, ориентированных на анализ данных

- **Современные информационные системы**
- **Концепция хранилища данных**
- **OLAP-технология**
- **Модели хранилища данных**
- **Расширения языка SQL для хранилищ данных**
- **Архитектура хранилищ данных**
- **Интеллектуальный анализ данных (Data Mining)**

1. Современные информационные системы

В области информационных технологий существуют два класса информационными систем (и соответственно, два класса задач):

- OLTP-системы и
- DSS-системы.

OLTP-системы — системы оперативной обработки транзакций. Основная функция подобных систем заключается в одновременном выполнении большого количества коротких транзакций от большого числа пользователей. Сами транзакции выглядят относительно просто, например, "снять сумму денег со счета А, добавить эту сумму на счет В".

Системы OLTP характеризуются:

- поддержкой большого числа пользователей;
- малым временем отклика на запрос;
- относительно короткими запросами;
- участие в запросах небольшого числа таблиц.

Практически все запросы к базе данных в OLTP-системах состоят из команд вставки, обновления, удаления. Запросы на выборку в основном предназначены для предоставления пользователям возможности выбора из различных справочников. Большая часть запросов, таким образом, известна заранее еще на этапе проектирования системы. Таким образом, критическим для OLTP-приложений является скорость и надежность выполнения коротких операций обновления данных.

Исторически такие системы возникли в первую очередь, поскольку реализовывали потребности в учете, скорости обслуживания, сборе данных и пр. Однако вскоре пришло понимание, что сбор данных — не самоцель и накопленные данные могут быть полезны: из данных можно извлечь информацию.

И возникает другой тип систем (и соответственно, класс задач) — **системы поддержки принятия решений DSS (Decision Support System)**, ориентированные анализ данных, на выполнение более сложных запросов, моделирование процессов предметной области, прогнозирование, нахождение зависимостей между данными (например, можно попытаться

определить, как связан объем продаж товаров с характеристиками покупателей), для проведения анализа "что если:". (аналитические системы). *Под DSS понимают человеко-машинный вычислительный комплекс, ориентированный на анализ данных и обеспечивающий получение информации, необходимой для принятия решений в сфере управления.*

- использование больших объемов данных;
- добавление в систему новых данных происходит относительно редко крупными блоками (например, раз в квартал загружаются данные по итогам квартальных продаж из OLTP-приложения);
- данные, добавленные в систему, обычно никогда не удаляются;
- перед загрузкой данные проходят различные процедуры "очистки", связанные с тем, что в одну систему могут поступать данные из многих источников, имеющих различные форматы представления для одних и тех же понятий, данные могут быть некорректны, ошибочны;
- небольшое число пользователей (аналитики);
- очень часто новый запрос формулируется аналитиком для уточнения результата, полученного в результате предыдущего запроса (интерактивность);
- скорость выполнения запросов важна, но не критична.

Перечисленные характеристики требуют особой организации данных, отличных от тех, что используются в OLTP-системах (нормализованные реляционные таблицы).

Аналитические системы, ориентированные на аналитика, можно разделить на

- статические DSS, известные в литературе информационными системами руководителя (Executive Information Systems — EIS) и
- динамические DSS.

EIS-системы содержат в себе предопределенные множества запросов и, будучи достаточными для повседневного обзора, неспособны ответить на все вопросы к имеющимся данным, которые могут возникнуть при принятии решений. Результатом работы такой системы, как правило, являются многостраничные отчеты, которые нельзя "покрутить", "развернуть" или "свернуть", чтобы получить желаемое представление данных и после тщательного изучения которых у аналитика появляется новая серия вопросов. Конечно, можно вызвать программиста (если он захочет придти), и он (если не занят) сделает новый отчет достаточно быстро — скажем, в течение часа (это очень сомнительно — так быстро в жизни не бывает: предположим, часа три). Получается, что аналитик может проверить за день не более двух идей. А ему (если он хороший аналитик) таких идей может приходить в голову по несколько в час.

Вторая группа (динамические DSS), напротив, ориентированы на обработку нерегламентированных (ad hoc) запросов аналитиков к данным. Наиболее глубоко требования к таким системам рассмотрел в 1993 г. E. F. Codd, положившей начало концепции OLAP. В последние годы в этом направлении оформился ряд новых концепций хранения и анализа корпоративных данных:

- концепция хранилища данных (Data Warehouse);
- оперативная аналитическая обработка OLAP (On-Line Analytical Processing);
- интеллектуальный анализ данных — (Data Mining).

Концепция хранилища данных определяет процесс сбора, отсеивания, предварительной обработки и накопления данных с целью

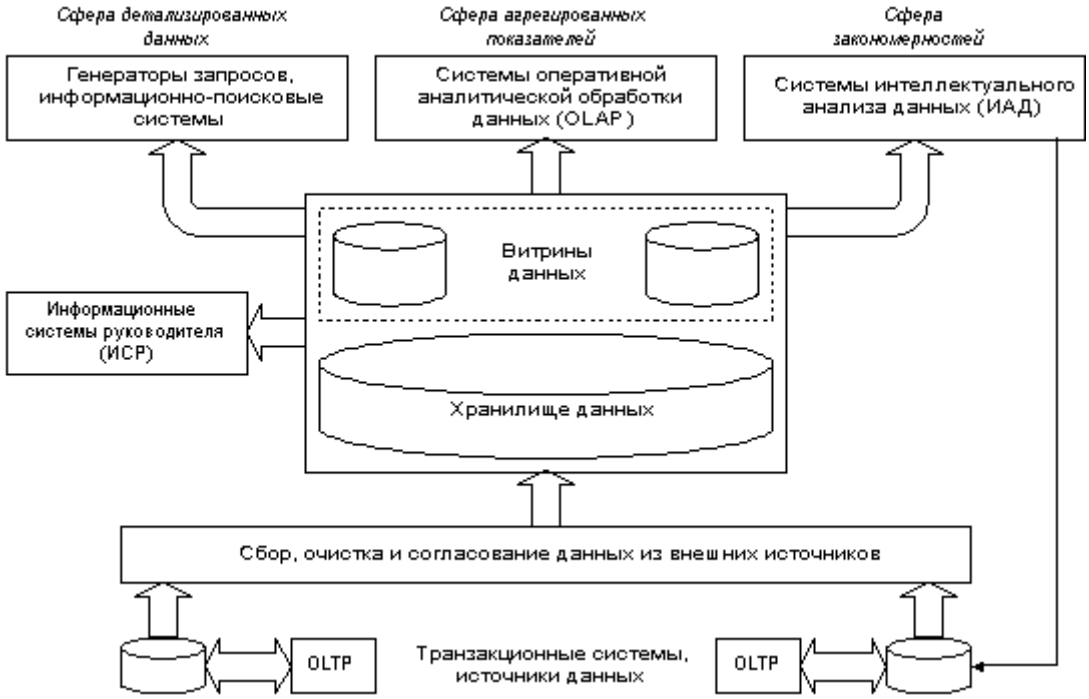
- долговременного хранения данных (1);
- предоставления результирующей информации пользователям в удобной форме для статистического анализа и создания аналитических отчетов (2).

Концепция OLAP — концепция комплексного многомерного анализа данных, накопленных в хранилище. Теоретически средства OLAP можно применять и непосредственно к оперативным данным или их точным копиям (чтобы не мешать оперативным пользователям). Но в этом случае мы рискуем наступить на свои грабли, поскольку беремся анализировать оперативные данные, которые напрямую для анализа непригодны.

Замечание: термин OLAP очень популярен в настоящее время и OLAP-системой зачастую называют любую DSS-систему, основанную на концепции хранилищ данных и обеспечивающих малое время выполнение (On-Line) аналитических запросов, не зависимо от того, используется ли многомерный анализ данных. Что не совсем верно.

Концепция интеллектуального анализа данных определяет задачи поиска функциональных и логических закономерностей в накопленной информации, построение моделей и правил, которые объясняют найденные аномалии и/или прогнозируют развитие некоторых процессов.

Примерная структура информационно-аналитической системы, построенной на основе хранилища данных, показана ниже. В конкретных реализациях отдельные компоненты этой схемы часто отсутствуют.



2. Концепция хранилища данных

Какова побудительная причина появление концепции хранилищ данных?

Казалось бы, зачем строить хранилища данных — ведь они содержат заведомо избыточную информацию, которая и так имеется в базах или файлах оперативных систем? Ответить можно кратко: анализировать данные оперативных систем напрямую невозможно или очень затруднительно. Это объясняется рядом причинами, в том числе

- разрозненностью данных (OLTP-системы, текстовые отчеты, xls-файлы);
- хранением их в форматах различных СУБД и в разных узлах корпоративной сети.

Но даже если на предприятии все данные хранятся на центральном сервере БД (что бывает крайне редко), аналитик почти наверняка не разберется в их сложных, подчас запутанных структурах.

Есть и еще одна причина, оправдывающая появление отдельного хранилища — *сложные аналитические запросы к оперативной информации тормозят текущую работу компании, надолго блокируя таблицы и захватывая ресурсы сервера.*

Можно констатировать, что практически в любой организации сложилась парадоксальная ситуация: — информация вроде бы, где-то и есть, её даже слишком много, но она неструктурированна, несогласованна, разрознена, не всегда достоверна, её практически невозможно найти и получить. В результате можно говорить об *отсутствие информации при наличии и даже избытке.*

Для того, чтобы извлекать полезную информацию из данных, они должны быть организованы способом, отличным от принятого в OLTP-системах Почему?

1. В OLTP-системах используются нормализованные таблицы базы данных. Нормализация эффективна, если отношения часто перестраиваются (вставка,. . .), но дает отрицательный эффект в случае операции выборки (особенно в случае сложных запросов). А в DSS-системах только операции выборки, и данные редко меняются, поэтому данные целесообразно хранить в виде слабо нормализованных отношений, содержащих заранее вычисленные основные итоговые данные. Большая избыточность и связанные с ней проблемы тут не страшны, т.к. обновление происходит только в момент загрузки новой порции данных. При этом происходит как добавление новых данных, так и пересчет итогов.
2. Выполнение некоторых аналитических запросов требует хронологической упорядоченности данных. Реляционная модель не предполагает существования порядка записей в таблицах.
3. В случае аналитических запросов чаще используются не детальные, а обобщенные (агрегированные данные).

В результате данные, применяемые для анализа, стали выделять в отдельные специальные базы данных, впоследствии получивших название хранилищ данных (Data Warehouse).

Хранилище данных (определение Билла Инмона(Bill Inmon)) — предметно-ориентированный, интегрированный, привязанный ко времени и неизменяемый набор данных, предназначенный для поддержки принятия решений. Базовые требования к хранилищу данных:

- *Ориентация на предметную область.* Хранилище должно разрабатываться с учетом специфики предметной области (клиенты, товары, продажи), а не прикладных областей деятельности (выписка счетов, контроль запасов, продажа товаров).
- *Интегрированность и внутренняя непротиворечивость.* Поскольку данные в хранилище поступают из разных источников (OLTP-системы, архивы и пр.),

необходимо привести их к единому формату (дата: 5 января, 5.01,:). В процессе загрузки хранилища должна быть обеспечена, очистка и согласованность данных.
- *Привязка ко времени.* Учет хронологии достигается введением атрибутов "Дата" и "Время". Упорядочение по этим атрибутам позволяет сократить время выполнения аналитических запросов.
- *Неизменяемость.* Данные не обновляются в оперативном режиме, а лишь регулярно пополняются из систем оперативной обработки по заданной дисциплине.
- *Поддержка высокой скорости получения данных из хранилища.*
- *Возможность получения и сравнения так называемых срезов данных (slice and dice);*
- *Полнота и достоверность хранимых данных;*
- *Поддержка качественного процесса пополнения данных.*

3. OLAP-технология

Термин OLAP был предложен в 1993 г. Эдвардом Коддом (E. Codd — автор реляционной модели данных) По Коду *OLAP-технология — это технология комплексного динамического синтеза, анализа и консолидации больших объемов многомерных данных.* Он же сформулировал 12 принципов OLAP, которые позже были переработано в так называемый тест **FASMI:**

- *Fast (быстрый)* — предоставление пользователю результатов анализа за приемлемое время (обычно не более 5 с), пусть даже ценой менее детального анализа;
- *Analysis (анализ)* — возможность осуществления любого логического и статистического анализа, характерного для данного приложения, и его сохранения в доступном для конечного пользователя виде;
- *Shared (разделяемой)* — многопользовательский доступ к данным с поддержкой соответствующих механизмов блокировок и средств авторизованного доступа;
- *Multidimensional (многомерной)* — многомерное концептуальное представление данных, включая полную поддержку для иерархий и множественных иерархий (ключевое требование OLAP);
- *Information (информации)* — возможность обращаться к любой нужной информации независимо от ее объема и места хранения.

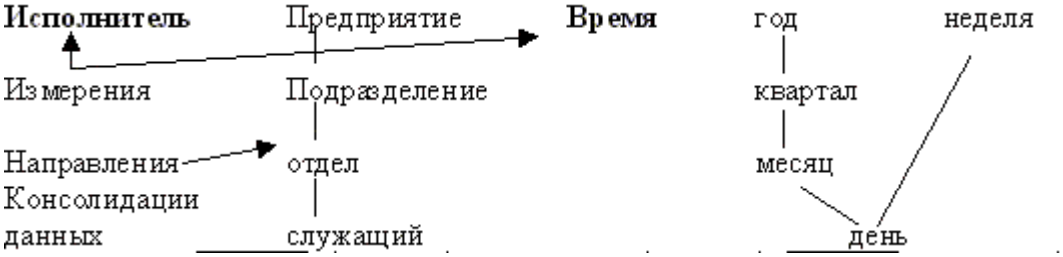
В самом общем виде OLAP=многомерное представление=куб

OLAP-технология представляет для анализа данные в виде многомерных (и, следовательно, нереляционных) наборов данных, называемых многомерными кубами (гиперкуб, метакуб, кубом фактов), оси которого содержат параметры, а ячейки — зависящие от них агрегатные данные

При том *гиперкуб является концептуальной логической моделью организации данных, а не физической реализацией их хранения, поскольку храниться такие данные могут и в реляционных таблицах* ("реляционные БД были, есть и будут наиболее подходящей технологией для хранения корпорационных данных" — E. Codd).

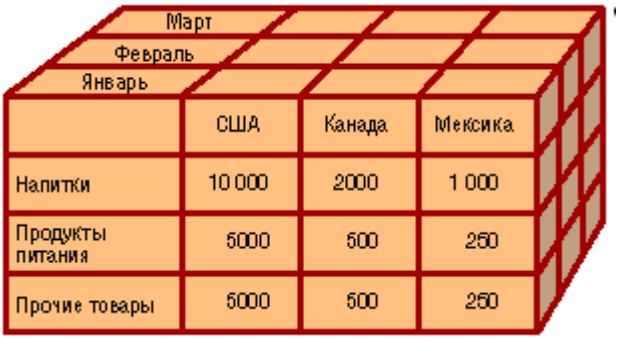
По Кодду, многомерное концептуальное представление (multi-dimensional conceptual view) представляет собой множественную перспективу, состоящую из нескольких независимых измерений, вдоль которых могут быть проанализированы определенные совокупности данных. Одновременный анализ по нескольким измерениям определяется как многомерный анализ. Осями многомерной системы координат служат основные атрибуты анализируемого бизнес-процесса (то, по чему ведется анализ). Например, для продаж это могут быть тип товара, регион, тип покупателя. В качестве одного из измерений используется время. На

пересечениях осей - измерений (dimensions) — находятся данные, количественно характеризующие процесс — меры (measures): суммы и иные агрегатные функции (min, max, avg, дисперсия, ср. отклонение и пр.). Каждое измерение включает направления консолидации данных, состоящие из серии последовательных уровней обобщения (уровней иерархии), где каждый вышестоящий уровень соответствует большей степени агрегации данных по соответствующему измерению (различные уровни их детализации). В этом случае становится возможным произвольный выбор желаемого уровня детализации информации по каждому из измерений.



Благодаря такой модели данных пользователи могут формулировать сложные запросы, генерировать отчеты, получать подмножества данных.

Пример. Трехмерный куб, где в качестве фактов использованы суммы продаж, а в качестве измерений — время, товар и магазин, определенных на разных уровнях группировки: товары группируются по категориям, магазины — по странам, а данные о времени совершения операций — по месяцам.



Значения, "откладываемые" вдоль измерений, называются членами или метками (members). Метки используются в операциях манипулирования измерениями.

Метки могут объединяться в иерархии, состоящие из одного или нескольких уровней детализации (levels). Например, метки измерения "Магазин" (Store) естественно объединяются в иерархию с уровнями:

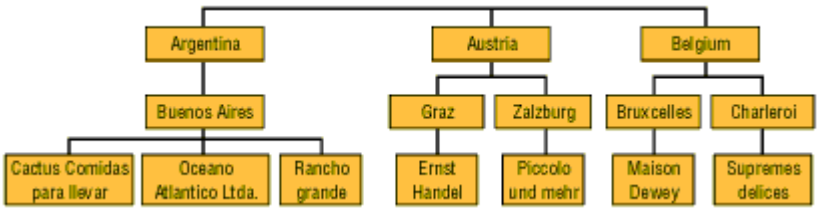
All(мир)

 Country(страна)
State(штат)
City(город)
Store(магазин)

В соответствии с уровнями иерархии вычисляются агрегатные значения, например объем продаж для USA (уровень "Country") или для штата California (уровень "State"). В одном

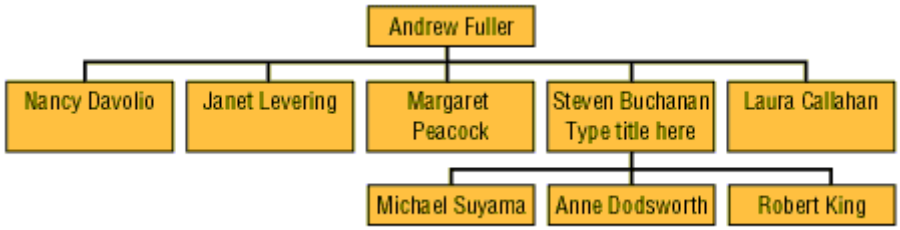
измерении можно реализовать более одной иерархии — скажем, для времени: {Год, Квартал, Месяц, День} и {Год, Неделя, День}.

Поскольку в рассмотренном примере в общем случае в каждой стране может быть несколько городов, а в городе — несколько клиентов, можно говорить об иерархиях значений в измерении — регион. В этом случае на первом уровне иерархии располагаются страны, на втором — города, а на третьем — клиенты.

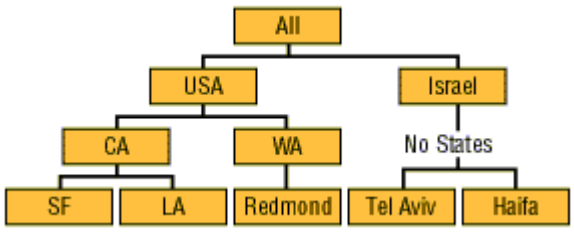


Иерархии могут быть сбалансированными (balanced), как, например, иерархия, представленная выше (такова же иерархии, основанные на данных типа "дата-время"), и несбалансированными (unbalanced). Типичный пример несбалансированной иерархии — иерархия типа "начальник-подчиненный".

Иногда для таких иерархий используется термин Parent-child hierarchy.



Существуют также иерархии, занимающие промежуточное положение между сбалансированными и несбалансированными (они обозначаются термином ragged — "неровный"). Обычно они содержат такие члены, логические "родители" которых находятся не на непосредственно вышестоящем уровне (например, в географической иерархии есть уровни Country, City и State, но при этом в наборе данных имеются страны, не имеющие штатов или регионов между уровнями Country и City).



Аналитические OLAP-операции:

- *Сечение.* При выполнении операции сечения формируется подмножество гиперкуба, в котором значение одного или более измерений фиксировано (значение параметров для фиксированного, например, месяца).
- *Вращение (rolling).* Операция вращения изменяет порядок представления измерений, обеспечивая представление метакуба в более удобной для восприятия форме.

- *Консолидация (rolling up).* Включает такие обобщающие операции, как простое суммирование значений (свертка) или расчет с использованием сложных вычислений, включающих другие связанные данные. Например, показателю для отдельных компаний могут быть просто просуммированы с целью получения показателей для каждого города, а показатели для городов могут быть "свернуты" до показателей по отдельным странам.
- *Операция спуска (drill doun).* Операция, обратная консолидации, которая включает отображение подробных сведений для рассматриваемых консолидированных данных.
- *Разбиение с поворотом (slicing and dicing).* Позволяет получить представление данных с разных точек зрения. Например, один срез данных о доходах может содержать все сведения о доходах от продаж товаров указанного типа по каждому городу. Другой срез может представлять данные о доходах отдельной компании в каждом из городов.

Поддержка многомерной модели данных и выполнение многомерного анализа данных осуществляются отдельным приложением или процессом, называемым ***OLAP-сервером.*** Клиентские приложения могут запрашивать требуемое многомерное представление и в ответ получать те или иные данные. При этом OLAP-серверы могут хранить многомерные данные разными способами

4. Модели хранилища данных

Обеспечивая *многомерное концептуальное представление со стороны пользовательского интерфейса к исходной базе данных*, все продукты OLAP делятся на несколько классов по типу исходной БД. Многомерный гиперкуб, используемый в OLAP-технологии, может быть реализован в рамках реляционной модели или существовать как отдельная база данных специальной многомерной структуры. В зависимости от этого принято различать многомерный (MOLAP) и реляционный (ROLAP) подходы к построению хранилища данных.

4.1. MOLAP (Multidimensional OLAP)

В MOLAP-модели многомерное представление данных реализуется физически. В специализированных СУБД, основанных на многомерном представлении данных, данные организованы не в форме реляционных таблиц, а в виде упорядоченных многомерных массивов:

1. гиперкубов (все хранимые в базе данных ячейки должны иметь одинаковую размерность, то есть находиться в максимально полном базисе измерений) и
2. поликубов (каждая переменная хранится с собственным набором измерений, и все связанные с этим сложности обработки перекладываются на внутренние механизмы системы).

Использование многомерных баз данных в системах оперативной аналитической обработки имеет следующие достоинства:

- Высокая производительность. В случае использования многомерных СУБД поиск и выборка данных осуществляется значительно быстрее, чем при многомерном концептуальном взгляде на реляционную базу данных, так как многомерная база данных денормализована, содержит заранее агрегированные показатели и обеспечивает оптимизированный доступ к запрашиваемым ячейкам.
- Многомерные СУБД легко справляются с задачами включения в информационную модель разнообразных встроенных функций, тогда как объективно существующие

ограничения языка SQL делают выполнение этих задач на основе реляционных СУБД достаточно сложным, а иногда и невозможным.

Недостатки MOLAP-модели:

- Многомерные СУБД не позволяют работать с большими базами данных.
- Многомерные СУБД по сравнению с реляционными очень неэффективно используют внешнюю память. В подавляющем большинстве случаев информационный гиперкуб является сильно разреженным, а поскольку данные хранятся в упорядоченном виде, неопределенные значения удаётся удалить только за счет выбора оптимального порядка сортировки, позволяющего организовать данные в максимально большие непрерывные группы. Но даже в этом случае проблема решается только частично. Кроме того, оптимальный с точки зрения хранения разреженных данных порядок сортировки скорее всего не будет совпадать с порядком, который чаще всего используется в запросах.

Следовательно, использование многомерных СУБД оправдано только при следующих условиях:

1. Объем исходных данных для анализа не слишком велик (не более нескольких гигабайт), то есть уровень агрегации данных достаточно высок.
2. Набор информационных измерений стабилен (поскольку любое изменение в их структуре почти всегда требует полной перестройки гиперкуба).
3. Время ответа системы на нерегламентированные запросы является наиболее критичным параметром.

ПримерыOLAP-серверов, использующих MOLAP-архитектуру: Oracle Express Server фирмы Oracle, IBM Informix MetaCube, IBM DB2 OLAP, Arbor Essbase.

4.2. ROLAP (Relational OLAP)

Системы оперативной аналитической обработки реляционных данных (ROLAP) позволяют представлять данные, хранимые в реляционной базе, в многомерной форме, обеспечивая преобразование информации в многомерную модель через промежуточный слой метаданных. В этом случае гиперкуб эмулируется СУБД на логическом уровне.

Для большинства хранилищ данных наиболее эффективным способом моделирования N-мерного куба фактов является **схема "звезда" (star schema).**

Основными составляющими структуры хранилищ данных являются таблица фактов (fact table) и таблицы измерений (dimension tables).

Таблица фактов является основной таблицей хранилища данных. Как правило, она содержит сведения об объектах или событиях, совокупность которых будет в дальнейшем анализироваться. Если проводить аналогию с многомерной моделью, то строка таблицы фактов соответствует ячейке гиперкуба. Обычно говорят о четырех наиболее часто встречающихся типах фактов. К ним относятся:

- факты, связанные с транзакциями (Transaction facts). Они основаны на отдельных событиях (типичными примерами которых являются телефонный звонок или снятие денег со счета с помощью банкомата);

- факты, связанные с "моментальными снимками" (Snapshot facts). Основаны на состоянии объекта (например, банковского счета) в определенные моменты времени, например на конец дня или месяца. Типичными примерами таких фактов являются объем продаж за день или дневная выручка;
- факты, связанные с элементами документа (Line-item facts). Основаны на том или ином документе (например, счете за товар или услуги) и содержат подробную информацию об элементах этого документа (например, количестве, цене, проценте скидки);
- факты, связанные с событиями или состоянием объекта (Event or state facts). Представляют возникновение события без подробностей о нем (например, просто факт продажи или факт отсутствия таковой без иных подробностей).

Таблица фактов индексируется по сложному ключу, составленному из ключей отдельных изменений. При этом как ключевые, так и некоторые неключевые поля таблицы фактов должны соответствовать будущим измерениям OLAP-куба. Помимо этого таблица фактов содержит одно или несколько числовых полей, на основании которых в дальнейшем будут получены агрегатные данные.

Замечания.

1. Для многомерного анализа пригодны таблицы фактов, содержащие как можно более подробные данные, то есть соответствующие членам нижних уровней иерархии соответствующих измерений.
2. В таблице фактов отсутствуют какие-либо сведения о том, как группировать записи при вычислении агрегатных данных.

Таблица измерений содержит неизменяемые или редко изменяемые данные. В каждой таблице измерений перечислены возможные значения одного из измерений гиперкуба. В подавляющем большинстве случаев эти данные представляют собой по одной записи для каждого члена нижнего уровня иерархии в измерении. Таблицы измерений также содержат как минимум одно описательное поле (обычно с именем члена измерения) и, как правило, целочисленное ключевое поле (обычно это суррогатный ключ) для однозначной идентификации члена измерения. Каждая таблица измерений должна находиться в отношении "один ко многим" с таблицей фактов.

Причина, по которой данная схема названа "звездой" достаточно очевидна. Концы звезды образуются таблицами измерений, а их с таблицей фактов, расположенной в центре, образуют лучи. В терминологии Кодда, каждый луч схемы звезды задает направление консолидации данных по соответствующему измерению.

В схеме "звезда" каждое измерение куба содержится в одной таблице, в том числе и при наличии нескольких уровней иерархии (государство — регион — нас.пункт в таблице "Покупатель", год — месяц — день в таблице "Период").

В сложных задачах с многоуровневыми измерениями используются различные расширения схемы "звезда" — **схема "снежинка" (snowflake schema).** Это расширение может проявляться в двух разновидностях.

1. В случае большого числа сложных атрибутов в таблице измерений, некоторые атрибуты могут быть детализированы в отдельных таблицах измерений. Иными словами отдельные измерения содержатся не в одной, а в нескольких связанных между собой таблицах. Дополнительные таблицы измерений в такой схеме, обычно соответствующие верхним уровням иерархии измерения и находящиеся в соотношении "один ко многим" в главной таблице измерений, соответствующей нижнему уровню иерархии, иногда называют консольными таблицами (outrigger table).

Например, из таблицы "Покупатель" можно изъять описания региона, населенного пункта (оставив лишь их ключи) и хранить их в отдельных дополнительных таблицах. Это уменьшит степень дублирования информации, но снижает скорость выполнения запросов, поскольку увеличивает степень нормализации. Поэтому даже при наличии иерархических измерений с целью повышения скорости выполнения запросов к хранилищу данных нередко предпочтение отдается схеме "звезда".

2. Другое расширение связано с созданием отдельных таблиц фактов для всех возможных сочетаний уровней обобщения различных измерений.

Увеличение числа таблиц фактов в базе данных может проистекать не только из множественности уровней различных измерений, но и из того обстоятельства, что в общем случае факты имеют разные множества измерений. При абстрагировании от отдельных измерений пользователь должен получать проекцию максимально полного гиперкуба, причем далеко не всегда значения показателей в ней должны являться результатом элементарного суммирования. Таким образом, при большом числе независимых измерений необходимо поддерживать множество таблиц фактов, соответствующих каждому возможному сочетанию выбранных в запросе измерений.

Это позволяет добиться лучшей производительности, но часто приводит к избыточности данных и к значительным усложнениям в структуре базы данных, в которой оказывается огромное количество таблиц фактов

Ниже приведен фрагмент для одного измерения в схеме "снежинка".

При такой структуре базы данных большинство запросов из области делового анализа объединяют центральную таблицу фактов с одной или несколькими таблицами измерений.

Пример: получить средние объемы продаж товаров каждого поставщика с разбивкой по покупателям и по месяцам.

```
Select avg(количество), поставщик, покупатель, месяц
      from таблица фактов, таблица поставщиков, таблица покупателей
            where таблица фактов.код_поставщика =
                  таблица поставщиков. код_поставщика
             and таблица фактов. код_покупателя =
                  таблица покупателей. код_покупателя
group by поставщик, покупатель, месяц
order by поставщик, покупатель, месяц
```

В любом случае, если многомерная модель реализуется в виде реляционной базы данных, следует создавать длинные и "узкие" таблицы фактов и сравнительно небольшие и "широкие" таблицы измерений. Таблицы фактов содержат численные значения ячеек гиперкуба, а остальные таблицы определяют содержащий их многомерный базис измерений. Часть информации можно получать с помощью динамической агрегации данных, распределенных по незвездообразным нормализованным структурам, хотя при этом следует помнить, что включающие агрегацию запросы при высоконормализованной структуре базы данных могут выполняться довольно медленно.

Достоинства использования реляционных баз данных в системах аналитической оперативной обработки:

1. При использовании ROLAP размер хранилища не является таким критичным параметром, как в случае MOLAP.
2. Внесение изменений в структуру измерений не требует физической реорганизации базы данных, как в случае MOLAP.
3. Реляционные СУБД обеспечивают значительно более высокий уровень защиты данных и хорошие возможности разграничения прав доступа.

Главный недостаток ROLAP по сравнению с многомерными СУБД — меньшая производительность.

Примеры OLAP-серверов, использующих ROLAP-архитектуру: IBM Informix Red Brick, HighGate Project фирмы Sybase, Microsoft SQL Server 2000 Analysis Services фирмы Microsoft.

4.3. Другие модели построения хранилищ данных

Гибридные системы (Hybrid OLAP, HOLAP) разработаны с целью совмещения достоинств и минимизации недостатков, присущих предыдущим классам. К этому классу относится Media/MR компании Speedware. По утверждению разработчиков, он объединяет аналитическую гибкость и скорость ответа MOLAP с постоянным доступом к реальным данным, свойственным ROLAP.

Примеры OLAP-серверов, использующих HOLAP-архитектуру: Microsoft SQL Server 2000 Analysis Services фирмы Microsoft, SAS Institute.

Помимо перечисленных средств существует еще один класс — инструменты управляемой среды запросов (MQE), дополненные функциями OLAP или интегрированные с внешними средствами, выполняющими такие функции. Эти хорошо развитые системы осуществляют выборку данных из исходных источников (реляционные базы данных, электронные таблицы), преобразуют их и помещают в динамическую многомерную базу данных, функционирующую на клиентской станции конечного пользователя. Построенный куб данных анализируется средствами многомерного OLAP, сохраняется и сопровождается локально.

Достоинства:

- относительная простота инсталляции, администрирования и сопровождения;
- способность каждого пользователя создавать свои собственные кубы данных.

Основными представителями этого класса являются BusinessObjects одноименной компании, PowerPlay компании Cognos.

5. Расширения языка SQL для хранилищ данных

Теоретически реляционная база данных звездообразной структуры обеспечивает хороший фундамент для выполнения произвольных, нерегламентированных запросов из области делового анализа. Однако между типичными аналитическими запросами и возможностями базового языка SQL имеются серьезные несоответствия:

- *Сортировка данных.* Многие аналитические запросы явно или неявно требуют предварительной сортировки данных (первые 10%, самые низкоприбыльные и т.п.). Однако SQL оперирует с несортированными строками. Единственным средством сортировки является фраза Order by в операторе Select, причем сортировка выполняется в самом конце процесса, когда данные уже отобраны.
- *Хронологические последовательности.* Многие запросы к хранилищам данных предназначены для анализа изменения некоторых показателей во времени (сравнение результатов одного года с другим, одного месяца некоторого года с результатами некоторого месяца другого года). Средствами стандартного SQL это зачастую сделать весьма трудно.
- *Сравнение с итоговыми данными.* Многое аналитические запросы сравнивают значения отдельных элементов (например, объемы продаж отдельных офисов) с итоговыми данными (например, объемами продаж по регионам). Такой SQL-запрос трудно выражаем средствами SQL. А сводный отчет классического формата — с

детальными данными, промежуточными и сводными итогами — единым SQL-запросом вообще не выразить, поскольку структура всех строк запроса должна быть одинаковой.

Решая эти проблемы, производители СУБД предлагают расширения SQL для аналитической обработки. Например, СУБД Red Brick одноименной фирмы, которая впоследствии бала куплена компанией Informix, а та соответственно IBM в язык RISQL (Red Brick Intelligent SQL) включены следующие расширения:

- *диапазоны* — позволяет сформировать запросы типа "отобрать первые 10 записей" (функция Rank . . .when);
- *перемещение итогов и средних* — используется для хронологического анализа, требующего предварительной обработки данных;
- *расчет текущих итогов и средних* — позволяет получать данные по отдельным месяцам плюс годовой итог на текущую дату и выполнять подобные запросы (функция **Cume** для вычисления текущего и общего итога для некоторого столбца, функции **Movingavg(n)** и **Movingsum(n)** для вычисления скользящего среднего и скользящей суммы по столбце на основе текущей и (n-1) предыдущих строк);
- *сравнительные коэффициенты* — позволяет создавать запросы, выражающие отношения отдельных значений к общим и промежуточным итогам без использования сложных вложенных запросов;
- *промежуточные итоги* — позволяет получать результаты запросов, в которых объединена детализированная и итоговая информация, причем с несколькими уровнями итогов (функция **Tertile**);
- *декодирование* — производит замену кодов из таблиц измерений понятными именами (функция **Decode**).

Примеры запросов на языке RISQL.

1. Получить значения ежеквартальных объемов продаж для отделения компании, указав общий итог с данного года и до текущего месяца.

Используется таблица **Branch_sales (отделение_компании, квартал, объем_продаж_за_квартал)**

```
Select квартал, объем_продажс_за_квартал,
                Cume(объем_продажс_за_квартал) as Year-To-Date
from Branch_sales where отделение_компании = “B3”;
```

Результат:

Квартал	Ежеквартальный объем продаж	Суммарный объем продаж с начала года
1	960000	960000
2	1290000	2250000
3	2000000	4250000
4	1500000	5750000

2. Получить значения ежемесячных объемов продаж для отделения компании для первых шести месяцев без учета сезонной составляющей.

Используется таблица **Branch_sales(отделение_компании, месяц, объем_продаж_за_месяц)**

Решение задачи достигается использованием функции Movingavg, вычисляющей трехмесячное скользящее среднее значение и функции Movingsum, вычисляющее трехмесячное скользящее суммарное значение, что позволит исключить сезонную составляющую.

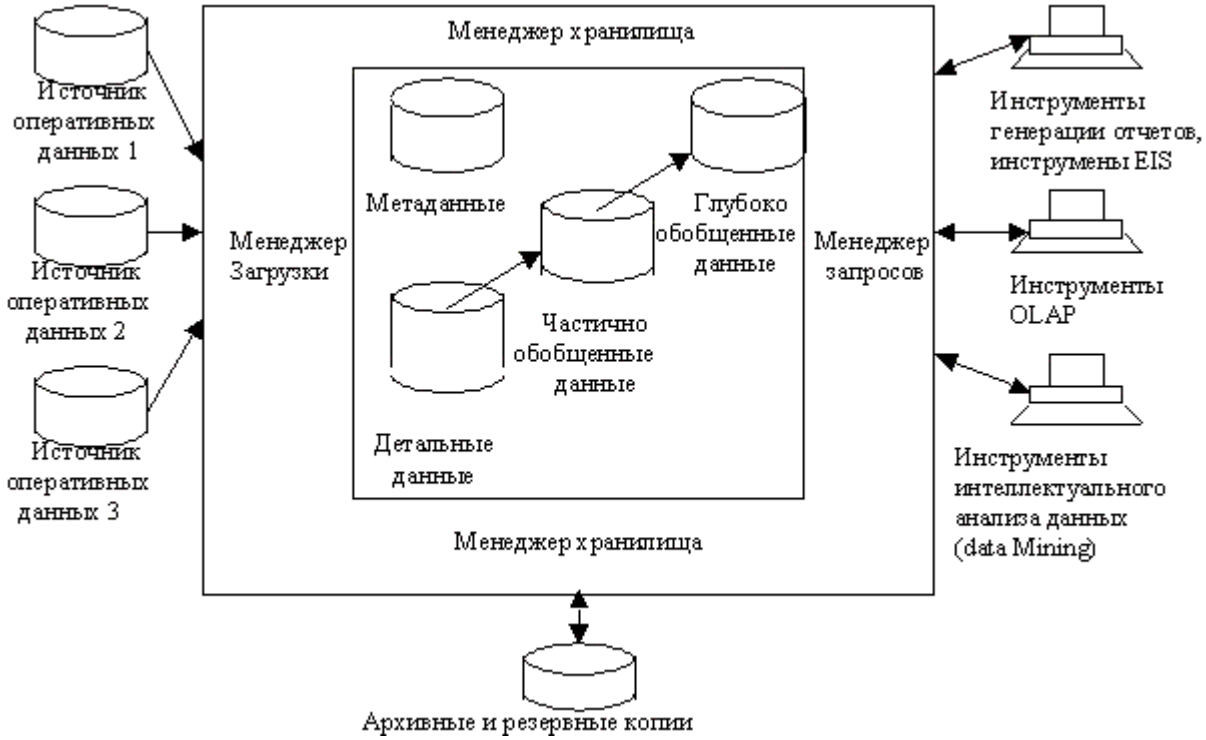
```
Select месяц, объем_продаж_за_месяц,
        Movingavg (объем_продаж_за_месяц) as 3-Month-Moving-Avg,
        Movingsum (объем_продаж_за_месяц) as 3-Month-Moving-Sum
from Branch_sales where отделение_компании = "B3";
```

Результат:

Месяц	Ежемесячный объем продаж	3-месячное скользящее среднее значение	3-месячное скользящее суммарное значение
1	210000	-	-
2	350000	-	-
3	400000	320000	960000
4	420000	390000	1170000
5	440000	420000	1260000
6	430000	430000	1290000

6. Архитектура хранилищ данных

Типичная архитектура хранилища данных приведена ниже:



Менеджер загрузки (load manager) выполняет операции, связанные с извлечением и загрузкой данных в хранилище.

Менеджер хранилища (warehouse manager) выполняет операции, связанные с управлением информацией, помещенной в хранилище данных:

- анализ непротиворечивости данных;
- создание индексов и представлений для базовых таблиц;
- денормализация данных (при необходимости);
- обобщение данных (при необходимости);
- резервное хранение и архивирование.

Менеджер запросов (query mamager) выполняет операции, связанные с управлением пользовательскими запросами.

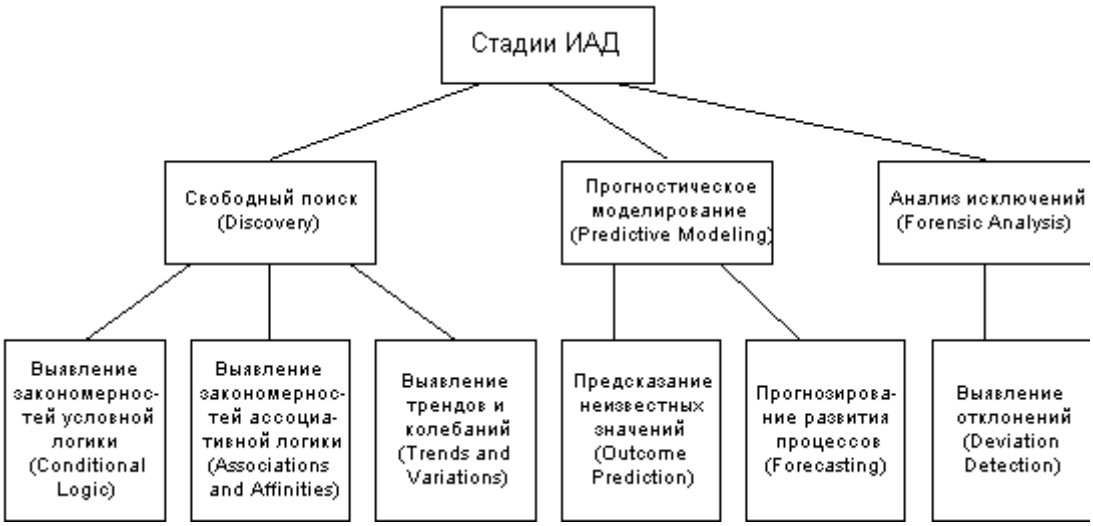
7. Интеллектуальный анализ данных (Data Mining)

Интеллектуальный анализ данных (ИАД — Data Mining) — это процесс поддержки принятия решений, основанный на поиске в данных скрытых закономерностей. При этом накопленные сведения автоматически обобщаются до информации, которая может быть охарактеризована как знания.

В общем случае процесс ИАД состоит из трёх стадий:

- выявление закономерностей (свободный поиск);
- использование выявленных закономерностей для предсказания неизвестных значений (прогностическое моделирование);
- анализ исключений, предназначенный для выявления и толкования аномалий в найденных закономерностях.

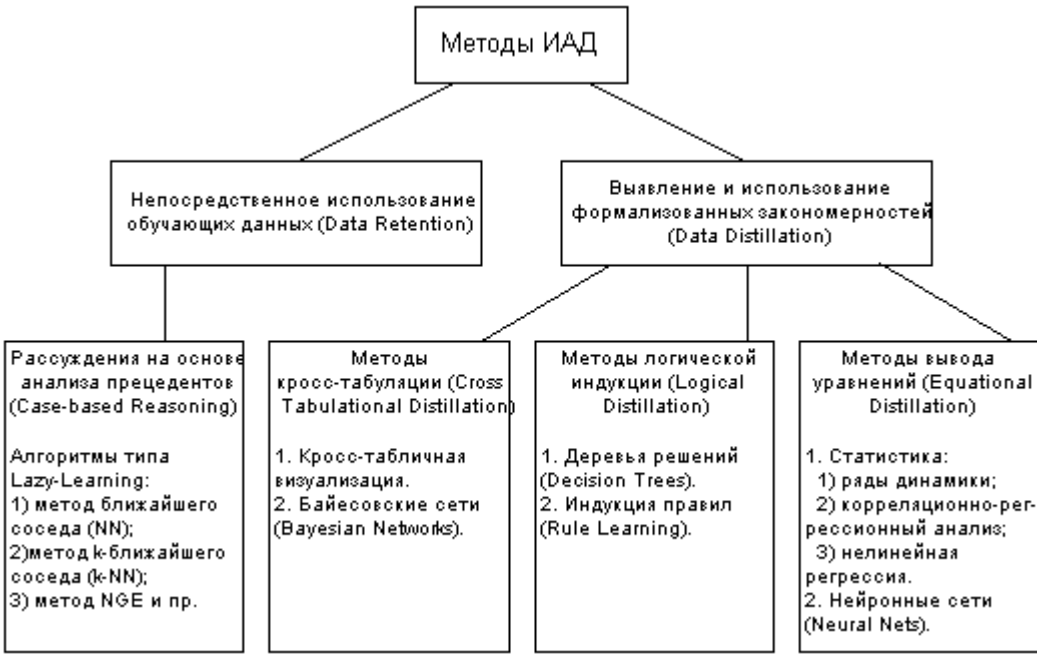
Иногда в явном виде выделяют промежуточную стадию проверки достоверности найденных закономерностей между их нахождением и использованием (стадия валидации).



Все методы интеллектуального анализа данных подразделяются на две большие группы по принципу работы с исходными обучающими данными.

1. В первом случае исходные данные могут храниться в явном детализированном виде и непосредственно использоваться для прогностического моделирования и/или анализа исключений; это так называемые методы рассуждений на основе анализа прецедентов. Главной проблемой этой группы методов является затрудненность их использования на больших объемах данных, хотя именно при анализе больших хранилищ данных методы интеллектуального анализа данных приносят наибольшую пользу.
2. Во втором случае информация вначале извлекается из первичных данных и преобразуется в некоторые формальные конструкции (их вид зависит от конкретного метода). Согласно предыдущей классификации, этот этап выполняется на стадии свободного поиска, которая у методов первой группы в принципе отсутствует. Таким образом, для прогностического моделирования и анализа исключений используются результаты этой стадии, которые гораздо более компактны, чем сами массивы исходных данных. При этом полученные конструкции могут быть либо "прозрачными" (интерпретируемыми), либо "черными ящиками" (нетрактуемыми).

Две эти группы и примеры входящих в них методов представлены ниже.


Методы ИАД
Непосредственное использование обучающих данных (Data Retention)
Выявление и использование формализованных закономерностей (Data Distillation)
Рассуждения на основе анализа прецедентов (Case-based Reasoning)
Алгоритмы типа Lazy-Learning:
1) метод ближайшего соседа (NN);
2)метод k-ближайшего соседа (k-NN);
3) метод NGE и пр.
Методы кросс-табуляции (Cross Tabulational Distillation)
1. Кросс-табличная визуализация.
2. Байесовские сети (Bayesian Networks).
Методы логической индукции (Logical Distillation)
1. Деревья решений (Decision Trees).
2. Индукция правил (Rule Learning).
Методы вывода уравнений (Equational Distillation)
1. Статистика:
1) ряды динамики;
2) корреляционно-регрессионный анализ;
3) нелинейная регрессия.
2. Нейронные сети (Neural Nets).