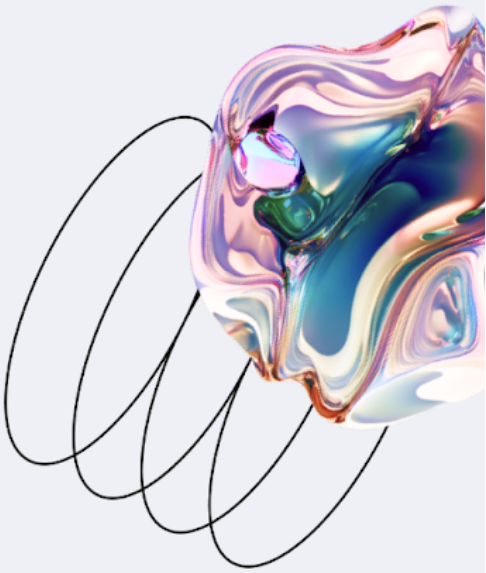
GeekBrains



Модели данных и нормализация таблиц. Схема «звезда»

ETL: автоматизация подготовки данных

Оглавление

Введение

Всем привет! Мы начинаем курс по разработке ETL и автоматизации подготовки данных. На нем научимся извлекать данные, трансформировать их, партицировать, а также строить пайплайны и визуализировать потоки данных.

В курсе будет 8 лекций и 8 практических семинаров. Мы начнем с вводных понятий, узнаем, что такое ETL, из каких частей состоит процесс и какие модели данных существуют. Затем поговорим о нормализации и денормализации таблиц, а также о способах партицирования данных. После этого рассмотрим инструмент для построения пайплайнов и визуализации потоков данных Apache Airflow. А завершим разбором специфики применения ETL в разных областях.

Цель нашего курса — разобраться, как правильно работать с данными и как построить процесс обработки и трансформации сырых данных в вид, нужный для конечной бизнес-модели. Давайте приступать!

Термины, используемые в лекции

ETL — общий термин для процессов, которые происходят, когда данные переносят из нескольких систем в одно хранилище. Аббревиатура от Extract, Transform, Load — извлечение, преобразование, загрузка.

Модель данных — это совокупность структур данных и операций их обработки. Основные типы моделей данных: иерархическая, сетевая и реляционная.

Атрибут — свойство некоторой сущности. Часто называется полем таблицы.

Домен атрибута — множество допустимых значений, которые может принимать атрибут.

Кортеж — конечное множество взаимосвязанных допустимых значений атрибутов, которые вместе описывают некоторую сущность (строка таблицы).

Отношение — конечное множество кортежей (таблица).

Схема отношения — конечное множество атрибутов, определяющих некоторую сущность. Иными словами, это структура таблицы, состоящей из конкретного набора полей.

Проекция — отношение, полученное из заданного путем удаления и (или) перестановки некоторых атрибутов.

Функциональная зависимость — связь между атрибутами или множествами атрибутов. Если между атрибутами X и Y есть функциональная зависимость, то кортежи, совпадающие по значению X, будут совпадать и по Y. Например, если значение атрибута «Название компании» — Canonical Ltd, то значением атрибута «Штаб-квартира» в таком кортеже всегда будет Millbank Tower, London, United Kingdom. Обозначение: {X} -> {Y}.

Нормальная форма — требование к структуре таблиц в теории реляционных баз данных. Цель нормализации — исключить избыточные функциональные зависимости и дублирование данных — причину аномалий, возникающих при добавлении, редактировании и удалении кортежей (строк таблицы).

Метод нормальных форм (НФ) состоит в сборе информации об объектах решения задачи в рамках одного отношения и последующей декомпозиции этого отношения на несколько взаимосвязанных отношений на основе процедур нормализации отношений.

Аномалия — ситуация в таблице, которая приводит к противоречию в БД либо существенно усложняет обработку БД. Причина — излишнее дублирование данных из-за функциональных зависимостей от неключевых атрибутов.

Аномалии-модификации проявляются в том, что изменение одних данных может повлечь просмотр всей таблицы и соответствующее изменение некоторых записей таблицы.

Аномалии-удаления — при удалении какого-либо кортежа из таблицы может пропасть информация, которая не связана напрямую с удаленной записью.

Аномалии-добавления возникают, когда информацию в таблицу нельзя поместить, пока она неполная, либо вставка записи требует дополнительного просмотра таблицы.

Основные функции ETL-систем

В современном мире компании собирают огромное количество данных. Все эти данные могут приходить из разных источников: баз данных, CRM-систем, файлов, внешних и внутренних API и так далее. У каждого источника свой способ предоставления доступа к данным: к базам данных надо подключиться напрямую, к файлам надо получить ссылки, к API — токен авторизации. Часто эти способы недоступны для быстрого использования, поэтому сравнивать данные из разных источников сложно. Чтобы с данными было удобно работать, им надо придать единую структуру, убрать лишнее и положить в хранилище данных. Для этого и нужен ETL.

Выделим две основные ситуации, когда нам нужны ETL-процессы.

1. **Данных становится слишком много.** Например, торговая компания хранит свои данные в базе. БД организована так, что каждая транзакция создает новую строку для быстрой записи или доступа к транзакции. Нам нужно провести анализ транзакций за определенное время и чем больше данных накапливается в БД, тем дольше будут выполняться запросы. Здесь нам нужен ETL для перехода к аналитическому виду данных — он поможет ускорить процесс получения ценной бизнес-информации из сырых данных транзакций.
2. **Данные приходят из различных источников**, и это затрудняет аналитику. Например, банк выдает кредиты. В одной системе лежит информация о денежных переводах, а в другой — статус заявки на кредит. ETL связывает данные о платежах по кредитам с данными анкеты. Эту информацию используют аналитики, чтобы понимать, какие факторы в анкете влияют на исполнение обязательств по кредиту.

Что же такое ETL? Это один из центральных процессов в системах хранения данных.

Этапы ETL:

- **Extract** — извлечение данных из разных источников.
- **Transform** — трансформация и очистка данных, приведение к единообразию или в соответствии с бизнес-задачами.
- **Load** — загрузка в хранилище данных.

По мере роста популярности баз данных в 1970-х годах ETL был представлен как процесс интеграции и загрузки данных для вычислений и анализа, который в итоге стал основным методом обработки данных для проектов хранилища данных.

ETL обеспечивает основу для анализа данных и рабочих потоков машинного обучения. С помощью ряда бизнес-правил ETL очищает и организует данные таким образом, чтобы удовлетворить потребности бизнес-аналитики (например, для ежемесячной отчетности). Но также подходит для более сложной аналитики, которая может улучшить внутренние процессы или взаимодействие с конечными пользователями.

Компании часто используют ETL, чтобы:

- извлечь данных из устаревших систем;
- очистить данные, чтобы улучшить их качество и обеспечить согласованность;
- загрузить данные в целевую БД.

Как работает ETL

Самый простой способ разобраться в том, как работает ETL, — понять, что происходит на каждом этапе процесса.

Извлечение

Необработанные данные копируются или экспортируются из исходных местоположений в промежуточную область. Группы управления данными могут извлекать данные из различных источников — структурированных или неструктурированных.

Примеры источников:

- SQL- или NoSQL-серверы,
- CRM- и ERP-системы,
- плоские файлы,
- электронный адрес,
- интернет-страницы.

Трансформация

В промежуточной области необработанные данные обрабатываются: преобразуются и консолидируются.

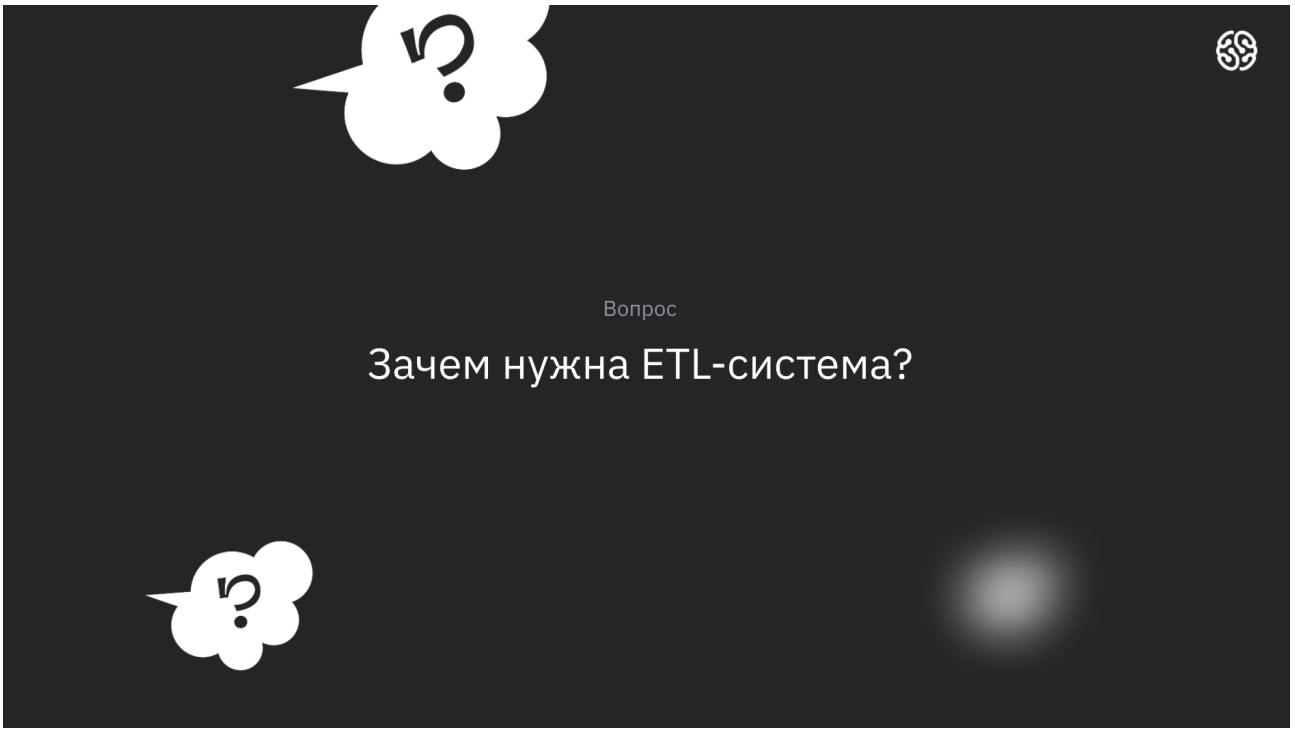
Примеры задач на этапе:

- Фильтрация, очистка, устранение дубликатов, проверка и аутентификация данных.
- Выполнение расчетов, переводов или суммирования на основе необработанных данных. Может включать изменение заголовков строк и столбцов для согласованности, конвертацию валют или других единиц измерения, редактирование текстовых строк и многое другое.
- Проведение аудитов для обеспечения качества данных и соответствия.
- Удаление, шифрование или защита данных регулируется отраслевыми или государственными регулирующими органами.
- Форматирование данных в таблицы или объединенные таблицы в соответствии со схемой целевого хранилища данных.

Загрузка

Это последний шаг — преобразованные данные перемещаются из промежуточной области в целевое хранилище. Как правило, этап включает первоначальную загрузку всех данных с последующей периодической загрузкой изменений. Реже — с полным обновлением: удалением и заменой данных в хранилище.

В большинстве организаций, использующих ETL, этот процесс автоматизирован, четко определен, непрерывен и управляется пакетами. Обычно ETL выполняется в нерабочее время, когда трафик в исходных системах и хранилище данных минимален.



Вопрос: зачем же нужна ETL-система?

Ответ: у бизнеса есть потребность — получить достоверную отчетность из того бардака, который творится в данных любой ERP-системы. Для этого и нужен ETL.

Этот бардак есть всегда, он бывает двух видов:

1. Случайные ошибки, возникшие на уровне ввода, переноса данных, или из-за багов.
2. Различия в справочниках и детализации данных между смежными ИТ-системами.

При этом, если первый вид бардака побороть можно, то второй по большей части не является ошибкой: контролируемые различия в структуре данных — это нормальная оптимизация под цели конкретной системы.

Из-за этой особенности ETL-системы должны в идеале решать не одну, а две задачи:

1. Привести все данные к единой системе значений и детализации, попутно обеспечив их качество и надежность.
2. Обеспечить аудиторский след при преобразовании (transform) данных, чтобы после преобразования можно было понять, из каких именно исходных данных и сумм собралась каждая строчка преобразованных данных.

Помнить об этих двух задачах полезно, особенно если вы пишете ETL-процесс вручную, или делаете его с использованием фреймворков низкой готовности, в которых не задана готовая структура промежуточных таблиц.

Легко упустить вторую задачу и получить много проблем с поиском причин ошибок в трансформированных данных.

Как работает ETL-система

Основные функции ETL-системы умещаются в процесс:

В разрезе потока данных это несколько систем-источников (обычно OLTP) и система-приемник (обычно OLAP), а также пять стадий преобразования между ними:

1. **Загрузка.** Задача этапа — затянуть в ETL данные произвольного качества для дальнейшей обработки. Здесь важно сверить суммы пришедших строк: если в исходной системе больше строк, чем в RawData (сырых данных), загрузка прошла с ошибкой.
2. **Валидация данных.** Данные последовательно проверяются на корректность и полноту, составляется отчет об ошибках для исправления.

3. **Мэппинг данных с целевой моделью.** То есть сопоставление объектов одной модели данных с другой. Яркий пример — контекстная реклама в браузере. Google собирает информацию о ваших интересах и делает ее мэппинг с доступной для показа рекламой. В итоге вы видите рекламу, которая соответствует вашим интересам.

   На этом этапе к валидированной таблице пристраивается еще n столбцов по количеству справочников целевой модели данных, а потом по таблицам мэппингов в каждой пристроенной ячейке, в каждой строке проставляются значения целевых справочников. Значения могут проставляться как 1:1, так и *:1, 1:* и *:*. Для настройки последних двух вариантов используют формулы и скрипты мэппинга, реализованные в ETL-инструменте.

4. **Агрегация данных.** Этот процесс нужен из-за разности детализации данных в OLTP (системе транзакций) и OLAP (аналитической системе).

5. **Выгрузка в целевую систему.** Это технический процесс использования коннектора и передачи данных в целевую систему.

Лучшие практики ETL-процесса

Не пытайтесь очистить все данные. Каждая компания хотела бы, чтобы все данные были чистыми, но большинство не готовы платить за ожидание или не готовы ждать. Очистка данных может занимать много времени, поэтому лучше не пытаться очистить их все.

При этом никогда не пропускайте этап очистки данных. Главная причина создания хранилища данных — предлагать более чистые и надежные данные. Перед очисткой важно определить стоимость очистки для каждого «грязного» элемента данных.

Обработка данных состоит из нескольких этапов. На этих этапах мы выполняем трансформации и группировки, а также можем строить индексы по дате. Лучше хранить эти данные отдельно, вне самого хранилища, чтобы использовать в каждой последующей итерации препроцессинга для его ускорения.

Модели данных

Представим, что нам нужно построить структуру базы данных или хранилища данных. С чего начать? Нужно представить все объекты предметной области и продумать взаимосвязь между ними. Когда мы построим архитектуру системы, это и будет простейшей моделью данных.

Итак, давайте обсудим, что же такое модель данных.

Модель данных — это совокупность структур данных и операций их обработки. Рассмотрим три основных типа моделей данных: иерархическую, сетевую и реляционную.

Иерархическая модель — это совокупность элементов, расположенных в порядке подчинения от общего к частному. Образует перевернутое по структуре дерево (граф).

Основные понятия иерархической структуры: уровень, узел и связь.

Узел — это совокупность атрибутов данных, описывающих некоторый объект. На схеме иерархического дерева узлы представляются вершинами графа. Каждый узел на более низком уровне связан только с одним узлом, находящимся на более высоком уровне.

У иерархического дерева только одна вершина, не подчиненная никакой другой. Она находится на самом верхнем, первом уровне, а зависимые (подчиненные) узлы — на втором, третьем и так далее. Количество деревьев в базе данных определяется числом корневых записей. К каждой записи базы данных существует только один иерархический путь от корневой записи.

В сетевой структуре при тех же основных понятиях (уровень, узел, связь) каждый элемент может быть связан с любым другим.

В реляционной модели данных объекты и связи между ними представлены в виде таблиц, при этом связи тоже рассматриваются как объекты. У всех строк, составляющих таблицу в реляционной базе данных, должен быть первичный ключ. Все современные средства СУБД поддерживают реляционную модель данных.

Характеристики реляционной модели — простая структура данных, удобное для пользователя табличное представление, возможность использовать формальный аппарат алгебры отношений и реляционного исчисления для обработки данных.

Каждая реляционная таблица представляет собой двумерный массив и обладает следующими свойствами:

1. Каждый элемент таблицы соответствует одному элементу данных.
2. Все столбцы в таблице однородные, то есть у всех элементов в столбце одинаковый тип и длина.

3. У каждого столбца уникальное имя.
4. Одинаковых строк в таблице нет.
5. Порядок следования строк и столбцов может быть произвольным.

Далее мы будем рассматривать реляционную модель данных и нормализацию данных для этой модели.

Нормализация таблиц

Нормализация — это процесс реорганизации данных через ликвидацию их избыточности и других противоречий. Цель нормализации — привести таблицы к виду, который позволяет выполнять непротиворечивое и корректное редактирование данных.

Как правило, нормализация применяется при восходящем подходе проектирования базы данных, когда все атрибуты, которые надо сохранить в БД, мы группируем по сущностям, для которых затем создаются таблицы. Однако при нисходящем подходе, когда вначале выявляются сущности, а затем их атрибуты и связи между ними, нормализация также применима. Например, для проверки корректности спроектированных таблиц.

- **Ненормализованная таблица** может хранить информацию о двух и более сущностях. В ней могут быть повторяющиеся столбцы, а в столбцах — повторяющиеся значения.
- **Нормализованная таблица** хранит информацию только об одной сущности.

Нормализация предполагает применение нормальных форм к структуре данных. Существуют 7 нормальных форм. Каждая (за исключением первой) подразумевает, что к данным уже была применена предыдущая нормальная форма. Например, прежде чем применить третью нормальную форму, к данным должна быть применена вторая. База данных считается нормализованной, если к ней применяется третья нормальная форма и выше.

- **Первая нормальная форма** (1NF, 1НФ) — сохраняемые данные на пересечении строк и столбцов должны представлять скалярное значение, а в таблицах не должны быть повторяющихся строк.
- **Вторая нормальная форма** (2NF, 2НФ) — каждый столбец, не являющийся ключом, должен зависеть от первичного ключа.

- **Третья нормальная форма** (3NF, 3НФ) — каждый столбец, не являющийся ключом, должен зависеть только от первичного ключа.
- **Нормальная форма Бойса-Кодда** (BCNF, НФБК) — немного более строгая версия третьей нормальной формы.
- **Четвертая нормальная форма (**4NF, 4НФ) — устранение многозначных зависимостей (multivalued dependencies), то есть зависимостей, где столбец с первичным ключом имеет связь один-ко-многим со столбцом, который не является ключом. Эта нормальная форма устраняет некорректные отношения многие-ко-многим.
- **Пятая нормальная форма** (5NF, 5НФ) — разделение таблицы на малые таблицы для устранения избыточности данных. Разбиение идет до тех пор, пока нельзя будет воссоздать оригинальную таблицу путем объединения малых таблиц.
- **Шестая нормальная форма** (domain key normal form, 6NF, 6НФ) — каждое ограничение в связях между таблицами должно зависеть только от ограничений ключа и ограничений домена, где домен представляет набор допустимых значений для столбца. Эта форма предотвращает добавление недопустимых данных, устанавливая ограничения на уровне отношений между таблицами, но не на уровне таблиц или столбцов. Как правило, не применима на уровне СУБД, в том числе и в SQL Server.

Теперь подробнее рассмотрим, как выглядят нормальные формы.

Первая нормальная форма

Отношение находится в 1НФ, если все его атрибуты простые. Все используемые домены должны содержать только скалярные значения. Не должно быть повторений строк в таблице.

Например, есть таблица «Автомобили»:

Марка	**Модель**
Nissan	GT-R
Porsche	911, Cayenne, Taycan

Нарушение нормализации 1НФ есть в моделях Porsche, так как в одной ячейке список из трех элементов: 911, Cayenne, Taycan. То есть он не атомарный.

Преобразуем таблицу к 1НФ:

Марка	**Модель**
Nissan	GT-R
Porsche	911
Porsche	Cayenne
Porsche	Taycan

Вторая нормальная форма

Отношение находится в 2НФ, если оно уже в 1НФ и каждый неключевой атрибут неприводимо зависит от первичного ключа (ПК).

Неприводимость означает, что в составе потенциального ключа отсутствует меньшее подмножество атрибутов, от которого можно вывести данную функциональную зависимость.

Например, дана таблица:

Модель	**Марка**	**Цена**	**Скидка**
GT-R	Nissan	7500000	10%
911	Porsche	15000000	5%
Cayenne	Porsche	12000000	5%
Taycan	Porsche	9000000	5%

Она в первой нормальной форме, но не во второй. Цена машины зависит от модели и марки. Скидки зависят от марки, то есть зависимость от первичного ключа неполная. Исправить это можно путем декомпозиции на два отношения, в которых неключевые атрибуты зависят от ПК.

Модель	Марка	Цена
GT-R	Nissan	7500000
911	Porsche	15000000
Cayenne	Porsche	12000000
Taycan	Porsche	9000000

Марка	Скидка
Nissan	10%
Porsche	5%

Третья нормальная форма

Отношение находится в 3НФ, когда оно уже в 2НФ и каждый неключевой атрибут нетранзитивно зависит от первичного ключа.

Проще говоря, второе правило требует выносить все неключевые поля, содержимое которых может относиться к нескольким записям таблицы, в отдельные таблицы.

Рассмотрим таблицу:

Марка	Автосалон	Телефон
Nissan	Люкс-авто	745-32-05
Porsche	Люкс-авто	745-32-05
Lada	Жигуль-топ	733-11-14

Она в 2НФ, но не в 3НФ. В отношении атрибут «Марка» является первичным ключом. Личных телефонов у автомобилей нет, телефон зависит исключительно от автосалона.

Таким образом, в отношении есть следующие функциональные зависимости: Марка → Салон, Салон → Телефон, Марка → Телефон.

Зависимость Марка → Телефон — транзитивная, следовательно, отношение не находится в 3НФ.

В результате разделения исходного отношения получаются два отношения, находящиеся в 3НФ:

Автосалон	**Телефон**
Люкс-авто	745-32-05
Жигуль-топ	733-11-14

Марка	**Автосалон**
Nissan	Люкс-авто
Porsche	Люкс-авто
Lada	Жигуль-топ

Нормальная форма Бойса-Кодда

Нормальная форма Бойса-Кодда (НФБК) — это частная форма 3НФ.

Определение 3НФ не совсем подходит для следующих отношений:

1. у отношения два или более потенциальных ключа;
2. два и более потенциальных ключа — составные;
3. они пересекаются, то есть у них есть хотя бы один общий атрибут.

Для отношений, имеющих один потенциальный ключ (первичный), НФБК является 3НФ.

Отношение находится в НФБК, когда каждая нетривиальная и неприводимая слева функциональная зависимость обладает потенциальным ключом в качестве детерминанта.

Предположим, рассматривается отношение, представляющее данные об аренде автомобиля по часам (каршеринг):

Номер автомобиля	Время начала	Время окончания	Тариф
1	09:30	10:30	Эконом
1	11:00	12:00	Эконом
1	14:00	15:30	Стандарт
2	10:00	12:00	Премиум-В
2	12:00	14:00	Премиум-В
2	15:00	18:00	Премиум-А

У каждого тарифа уникальное название, он зависит от выбранного автомобиля и наличия льгот:

- « Эконом» — автомобиль 1 для льготников,
- «Стандарт» — автомобиль 1 для нельготников,
- «Премиум-А» — автомобиль 2 для льготников,
- «Премиум-В» — автомобиль 2 для нельготников.

Таким образом, возможны следующие составные первичные ключи:

- {Номер автомобиля, Время начала},
- {Номер автомобиля, Время окончания},
- {Тариф, Время начала},
- {Тариф, Время окончания}.

Отношение находится в 3НФ. Требования 2НФ выполняются, так как все атрибуты входят в какой-то из потенциальных ключей, а неключевых атрибутов в отношении нет. Также нет и транзитивных зависимостей, что соответствует требованиям третьей нормальной формы.

Тем не менее существует функциональная зависимость Тариф → Номер автомобиля, в которой левая часть (детерминант) не является потенциальным ключом отношения, то есть отношение не находится в нормальной форме Бойса-Кодда.

Недостаток этой структуры в том, что, например, по ошибке можно приписать тариф «Эконом» к бронированию второй стоянки, хотя он может относиться только к первой.

Можно улучшить структуру с помощью декомпозиции отношения на два и добавления атрибута «**Имеет льготы**», получив отношения, удовлетворяющие НФБК (подчеркнуты атрибуты, входящие в первичный ключ).

Тарифы:

Тариф	**Номер автомобиля**	**Имеет льготы**
Эконом	1	Да
Стандарт	1	Нет
Премиум-А	2	Да
Премиум-В	2	Нет

Бронирования:

Тариф	**Время начала**	**Время окончания**
Эконом	09:30	10:30
Эконом	11:00	12:00
Стандарт	14:00	15:30
Премиум-В	10:00	12:00
Премиум-В	12:00	14:00
Премиум-А	15:00	18:00

Четвертая нормальная форма

Отношение находится в 4НФ, если оно уже в НФБК и все нетривиальные многозначные зависимости фактически являются функциональными зависимостями от ее потенциальных ключей.

В отношении R (A, B, C) существует **многозначная зависимость** R.A -> -> R.B в том и только в том случае, если множество значений B, соответствующее паре значений A и C, зависит только от A и не зависит от C.

Предположим, что рестораны готовят разные виды пиццы, а их службы доставки работают только в определенных районах города. Составной первичный ключ соответствующей переменной отношения включает три атрибута: {Ресторан, Вид пиццы, Район доставки}.

Такая переменная отношения не соответствует 4НФ, так как есть многозначная зависимость:

{Ресторан} → {Вид пиццы}
{Ресторан} → {Район доставки}

То есть, например, при добавлении нового вида пиццы придется внести по одному новому кортежу для каждого района доставки. Возможна логическая аномалия, при которой определенному виду пиццы будут соответствовать лишь некоторые районы доставки из обслуживаемых рестораном районов.

Чтобы предотвратить аномалии, нужно декомпозировать отношение, разместив независимые факты в разных отношениях. В примере следует выполнить декомпозицию на {Ресторан, Вид пиццы} и {Ресторан, Район доставки}.

Однако, если к исходной переменной отношения добавить атрибут, функционально зависящий от потенциального ключа, например, цену с учетом стоимости доставки ({Ресторан, Вид пиццы, Район доставки} → Цена), то полученное отношение будет находиться в 4НФ и его уже нельзя подвергнуть декомпозиции без потерь.

Пятая нормальная форма

Отношения находятся в 5НФ, если оно уже в 4НФ и отсутствуют сложные зависимые соединения между атрибутами.

Если «Атрибут_1» зависит от «Атрибута_2», «Атрибут_2» — от «Атрибута_3», а «Атрибут_3» — от «Атрибута_1», то все три атрибута обязательно входят в один кортеж.

Это очень жесткое требование, которое можно выполнить лишь при дополнительных условиях. На практике трудно найти пример реализации этого требования в чистом виде.

Например, таблица содержит три атрибута: «Поставщик», «Товар» и «Покупатель». Покупатель_1 приобретает несколько Товаров у Поставщика_1. Покупатель_1 приобрел новый Товар у Поставщика_2. Тогда в силу изложенного выше требования Поставщик_1 обязан поставлять Покупателю_1 тот же самый новый Товар, а Поставщик_2 должен поставлять Покупателю_1, кроме нового Товара, всю номенклатуру Товаров Поставщика_1.

На практике этого не бывает. Покупатель свободен в своем выборе товаров. Чтобы убрать это затруднение, все три атрибута разносят по разным отношениям (таблицам). После выделения трех новых отношений (Поставщик, Товар и Покупатель) нужно помнить, что при извлечении информации (например, о покупателях и товарах) в запросе нужно соединить все три отношения. Любая комбинация соединения двух отношений из трех приведет к извлечению некорректной информации.

У некоторых СУБД есть механизмы, устраняющие извлечение недостоверной информации. Тем не менее следует придерживаться общей рекомендации: структуру базы данных строить таким образом, чтобы избежать 4НФ и 5НФ.

Пятая нормальная форма ориентирована на работу с зависимыми соединениями. Указанные зависимые соединения между тремя атрибутами встречаются очень редко. Зависимые соединения между четырьмя, пятью и более атрибутами указать практически невозможно.

Доменно-ключевая нормальная форма

Переменная отношения находится в ДКНФ тогда и только тогда, когда каждое наложенное на нее ограничение является логическим следствием ограничений доменов и ограничений ключей, наложенных на данную переменную отношения.

Ограничение домена — ограничение, предписывающее использовать для определенного атрибута значения только из некоторого заданного домена. Ограничение — это, по сути, задание перечня (или логического эквивалента перечня) допустимых значений типа и объявление о том, что указанный атрибут имеет данный тип.

Ограничение ключа — ограничение, утверждающее, что некоторый атрибут или комбинация атрибутов является потенциальным ключом.

Любая переменная отношения, находящаяся в ДКНФ, обязательно находится в 5НФ. Однако не любую переменную отношения можно привести к ДКНФ.

Шестая нормальная форма

Переменная отношения находится в 6НФ тогда и только тогда, когда она удовлетворяет всем нетривиальным зависимостям соединения. Из определения следует, что переменная находится в 6НФ, когда она неприводима, то есть не может быть подвергнута дальнейшей декомпозиции без потерь. Каждая переменная отношения, которая находится в 6НФ, также находится и в 5НФ.

Идея «декомпозиции до конца» выдвигалась до начала исследований в области хронологических данных, но не нашла поддержки. Однако для хронологических баз данных максимально возможная декомпозиция позволяет бороться с избыточностью и упрощает поддержание целостности базы данных.

Для хронологических баз данных определены U_операторы, которые распаковывают отношения по указанным атрибутам, выполняют соответствующую операцию и упаковывают полученный результат. В примере соединение проекций отношения должно производиться при помощи оператора U_JOIN.

Сотрудники:

№	Время	Должность	Телефон
3322	01-03-2020:11-04-2022	аналитик	+79374422755
3124	31-07-2019:14-03-2022	программист	+79374443111
3322	12-04-2022:21-09-2022	программист	+79374422755

Переменная отношения «Сотрудники» не находится в 6НФ и может быть подвергнута декомпозиции на переменные отношения «Должность» и «Телефон».

Должность сотрудников:

№	Время	Должность
3322	01-03-2020:11-04-2022	аналитик
3124	31-07-2019:14-03-2022	программист

Телефоны сотрудников:

№	Время	Телефон
3322	01-03-2020:11-04-2022	+79374422755
3124	31-07-2019:14-03-2022	+79374443111

Мы рассмотрели нормальные формы таблиц и способы приведения ненормализованных таблиц к одной из нормальных форм. На практике проектирования схем баз данных достижение третьей нормальной формы (3НФ) — достаточное условие для большинства случаев.

Что такое хранилище данных?

Хранилища данных — это система сбора и управления данными из различных источников. Ядро системы BI для анализа данных и отчетности.

Хранилище данных — это смесь технологий и компонентов, которая помогает стратегическому использованию данных. Это электронное хранилище большого объема информации, предназначенное для бизнеса, для обработки запросов и анализа вместо обработки транзакций. Это процесс преобразования данных в информацию и своевременного предоставления их пользователям, чтобы изменить ситуацию.

Концепция хранилища данных

Основная концепция хранилища данных состоит в том, чтобы упростить для компании единую версию правды для принятия решений и прогнозирования.

Хранилище данных – это информационная система, которая содержит исторические и коммутативные данные из одного или нескольких источников. Концепция хранилища данных упрощает процесс отчетности и анализа организации.

Что такое многомерная схема?

Многомерная схема разработана для моделирования систем хранилищ данных. Схемы предназначены для удовлетворения уникальных потребностей очень больших баз данных, разработанных для аналитических целей.

Ниже приведены 3 основных типа многомерных схем, у каждой из которых свои уникальные преимущества.

- Схема «Звезда»
- Схема «Снежинка»
- Схема «Галактика»

В процессе построения качественной аналитической платформы главная цель дизайнера системы — сделать так, чтобы аналитические запросы было легко писать, а различные статистики считались эффективно. Для этого, в первую очередь, нужно определить модель данных.

Почему нужно заботиться о модели данных?

Сегодня многие столкнулись с проблемой смены BI-платформы, но это не единственная неприятность, которая может случиться, если при работе с данными специалисты не придерживаются стандартов.

Дело в том, что ландшафт может измениться в любое время: например, если вам потребуются дополнительные источники данных, если расчеты станут сложнее, если компанию поглотит другая организация или произойдет слияние двух структур.

Часто, начав с проекта, где отчеты строятся из нескольких таблиц, со временем мы сталкиваемся с экспоненциальным ростом сложности ландшафта данных. Поэтому, если не следить за порядком, даже для одного пользователя все это превращается в зоопарк, а для многопользовательской среды неуправляемый рост наборов данных станет кошмаром.

Что такое схема «Звезда»?

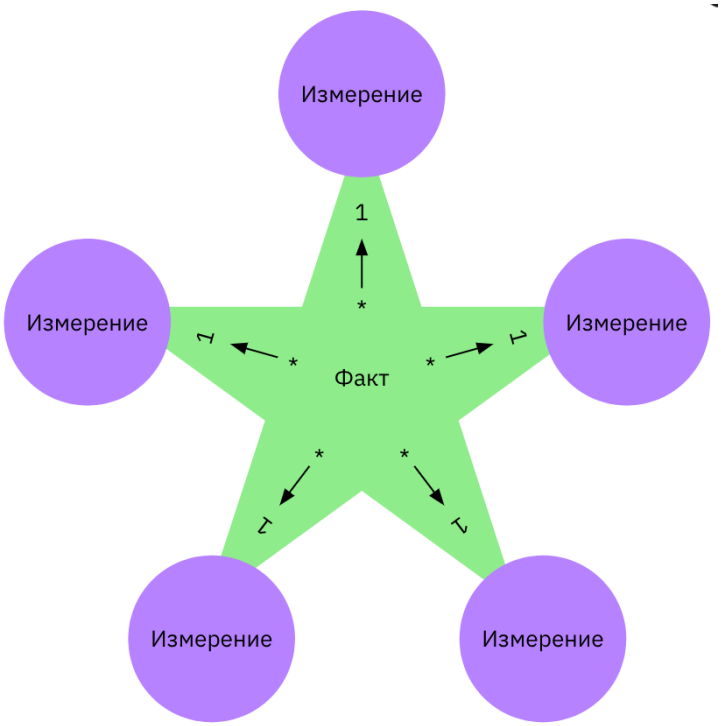
Среди всех моделей данных, которые пытаются найти идеальный баланс между двумя подходами, одна из самых популярных (мы используем ее в Airbnb) — схема «Звезда».

Она основана на построении нормализованных таблиц (таблиц фактов и таблиц измерений, о них мы поговорим на следующем уроке), из которых, в случае чего, могут быть получены денормализованные таблицы. В результате такой дизайн пытается найти баланс между легкостью аналитики и сложностью поддержки ETL.

Характеристики схемы «Звезда»:

- Каждое измерение представлено единственной одномерной таблицей.
- Таблица измерений должна содержать набор атрибутов.
- Таблица измерений присоединяется к таблице фактов с помощью внешнего ключа.
- Таблицы измерений не соединены друг с другом.
- Таблица фактов будет содержать ключ и меру.
- Схема «Звезда» проста для понимания и обеспечивает оптимальное использование диска.
- Таблицы измерений **не нормализованы**.
- Схема широко поддерживается BI-инструментами.

Схема «Звезда» — это самый простой тип схемы хранилища данных. Она названа так, потому что ее структура напоминает звезду:



Заключение

Сегодня мы узнали, что такое ETL, из каких этапов он состоит и в каких случаях нужен компаниям. Поговорили о моделях данных и их видах. Подробно рассмотрели процесс нормализации данных и 7 нормальных форм. А также разобрались, что такое хранилище данных и как устроена многомерная схема «Звезда».

Что можно почитать еще?

1. «Бизнес-процессы. Языки моделирования, методы, инструменты», Франк Шенталер, Готфрид Фоссен, Андреас Обервайс, Томас Карле.
2. «Путь аналитика. Практическое руководство IT-специалиста», Андрей Перерва, Вера Иванова.