

БАЗЫ ДАННЫХ



Литература

- Codd E.F. Normalized Data Base Structure: A Brief Tutorial //Proc. of 1971 ACM SIGFIDET Workshop on Data Description, Access and Control.- N.-Y.: ACM. - 1971. - P.1-17.
- Groh M. Access 2010 Bible. Canada:Wiley Publishing, Inc., 2010 – 1395 p.
- Дейт К. SQL и реляционная теория. Как грамотно писать код на SQL. М.: Символ-Плюс, 2010 – 480 с.
- Кляйн К., Кляйн Д., Хант Б. SQL. Справочник, 3-е издание. М.: Символ-Плюс, 2009 – 656 с.
- Коннолли Т., Бегг К., Страчан А. Базы данных. Проектирование, реализация и сопровождение. Теория и практика. М.: Вильямс, 2001 – 1120 с.
- Кренке Д. Теория и практика построения баз данных. СПб.: Питер, 2005 – 864

Определения понятия «базы данных»

Определения из международных стандартов:

- **ГОСТ Р ИСО МЭК ТО 10032-2007: Эталонная модель управления данными (идентичен ISO/IEC TR 10032:2003 Information technology — Reference model of data management): База данных** — совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных.
- **ISO/IEC 2382-1:1993. Information technology — Vocabulary — Part 1: Fundamental terms: База данных** — совокупность данных, организованных в соответствии с концептуальной структурой, описывающей характеристики этих данных и взаимоотношения между ними, причём такое собрание данных, которое поддерживает одну или более областей применения.

Определения из других авторитетных источников:

- **База данных** — организованная в соответствии с определёнными правилами и поддерживаемая в памяти компьютера совокупность данных, характеризующая актуальное состояние некоторой предметной области и используемая для удовлетворения информационных потребностей пользователей.
- **База данных** — совместно используемый набор логически связанных данных (и описание этих данных), предназначенный для удовлетворения информационных потребностей организации.

Элементы теории множеств (определения базовых понятий)

- **Множество** - это неопределяемое понятие, представляющее некоторую совокупность данных. Элементы множества можно отличать друг от друга, а также определять, принадлежит ли данный элемент данному множеству.
- Над множествами можно выполнять операции **объединения, пересечения, разности и дополнения**.
- Новые множества можно строить в том числе при помощи понятия **декартового произведения**. Декартово произведение нескольких множеств - это множество **кортежей**, построенный из элементов этих множеств.
- **Отношение** - это подмножество декартового произведения множеств. Отношения состоят из *однотипных* кортежей. Каждое отношение имеет **предикат отношения** и каждый n -местный предикат задает n -арное отношение.

Отношение является математическим аналогом понятия «**таблица**».

Понятие системы управления базой данных

СУБД (Система Управления Базой данных) - это программное обеспечение, имеющее средства обработки на языке БД и управляющее доступом к БД. Оно эффективно и компактно хранит данные, предоставляет пользователям возможность извлечения и модификации этих данных.

Основная задача СУБД — дать пользователю базы данных возможность работать с ней, *не вникая во все подробности работы на уровне аппаратного обеспечения.*



Функции СУБД:

- Хранение данных;
- Модификация данных;
- Управление данными в различных видах памяти;
- Журналирование изменений;
- Резервное копирование;
- Восстановление данных;
- Обеспечение защиты.

Основные компоненты СУБД:

- Ядро СУБД – отвечает за управление данными в памяти;
- Процессор языка базы данных – обеспечивает выполнение и оптимизацию запросов для работы с данными;
- Подсистема поддержки времени исполнителя – интерпретирует программы работы с данными, создающие пользовательский интерфейс СУБД;
- Утилиты (сервисные программы) – предоставляют дополнительные возможности работы и обслуживания информационной системы.

Категории пользователей СУБД

- **Конечные пользователи** – категория пользователей, в интересах которых создается БД;
- **Администраторы БД** - группа пользователей, которая отвечает за ее оптимальную организацию с точки зрения одновременной работы множества конечных пользователей, а также отвечает за корректность работы СУБД в многопользовательском режиме;
- **Разработчики приложений** – категория пользователей, которая функционирует во время проектирования, создания и реорганизации БД. К этой же группе можно отнести **администраторов приложений**. Всего в составе группы должны быть:
 - ✓ *системные аналитики;*
 - ✓ *проектировщики структур данных и внешнего информационного обеспечения;*
 - ✓ *проектировщики технологических процессов обработки данных;*
 - ✓ *системные и прикладные программисты;*
 - ✓ *операторы и специалисты по техническому обслуживанию.*

История развития СУБД:

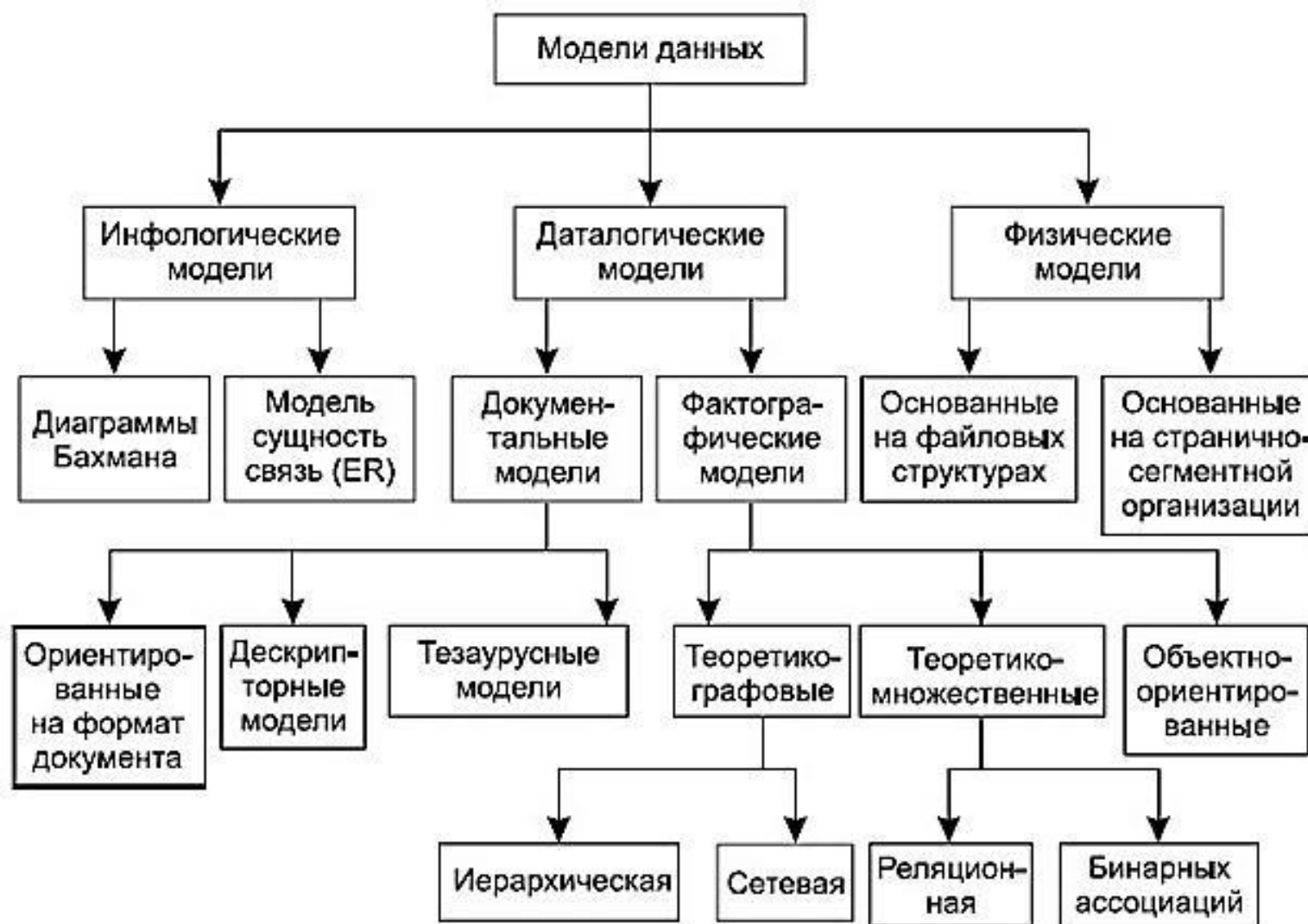
- 1960-е гг. – первый стандарт в области БД ("Подход CODASYL" (CODASYL Approach)), иерархическое построение, отсутствие автоматизированного поиска данных.
- 1970-е гг. – реляционный подход Эдгара Кодда. Работу «*A Relational Model of Data for Large Shared Data Banks*» считается первой работой по реляционной модели данных.
- 2000 – е гг. – постреляционные модели данных.

КЛАССИФИКАЦИИ СУБД

- По моделям данных;
- По степени распределенности;
- По типу взаимодействия с обрабатывающей программой.



Классификация по моделям данных



Классификация по моделям данных

При разработке БД чаще всего упоминаются и используются следующие модели:

- Иерархические;
- Сетевые;
- Реляционные;
- Постреляционные (объектно-ориентированные на базе реляционных).

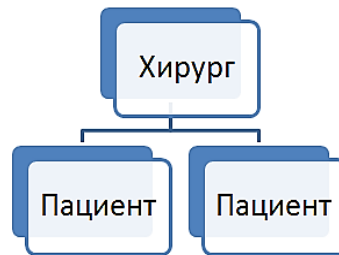
Иерархическая модель

Иерархическая модель базы данных состоит из объектов с указателями от родительских объектов к потомкам, соединяя вместе связанную информацию.

Модель 1



Модель 2



Проблема модели 1: если врач работает более одного пациента, то возникнет *избыточность*. Кроме того, возникает *аномалия включения* – в БД нельзя включить врача без пациентов.

Проблема модели 2: при увольнении хирурга исчезнет информация обо всех его пациентах (*аномалия удаления*).

Пример:

Если иерархическая база данных содержала информацию о книгах и их авторах, то будет существовать объект «автор» (родитель) и объект «книга» (дочерний). Объект «автор» будет иметь указатели от фамилии каждого автора к физическому расположению наименования книг в объект «КНИГИ».

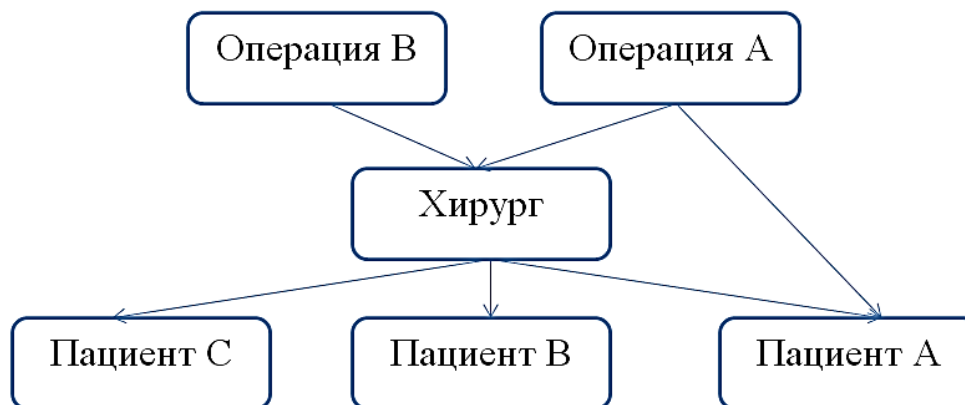
Наиболее распространенная иерархическая СУБД: Information Management System (IMS), Time-Shared Date Management System, InterSystems Caché, Google App Engine Datastore API.

Такие файловые системы как FAT, NTFS и некоторые другие основаны на принципах иерархической базы данных.

Сетевая модель

Сетевая модель базы данных является расширением иерархической и состоит из набора экземпляров определенного типа записи и набора экземпляров определенного типа связей между этими записями.

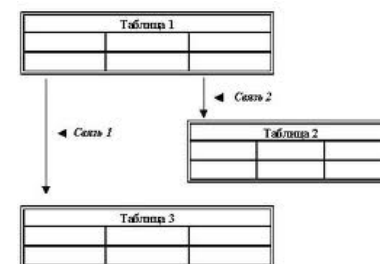
Примеры: Integrated Database Management System (IDMS), СООБЗ Cerebrum, ИСУБД CronosPRO, dbVista, Caché, GT.M



Основная проблема модели: сложность реорганизации базы данных.

Реляционная модель

- Реляционная модель базы данных является самой распространенной и использует двумерные массивы для хранения данных.

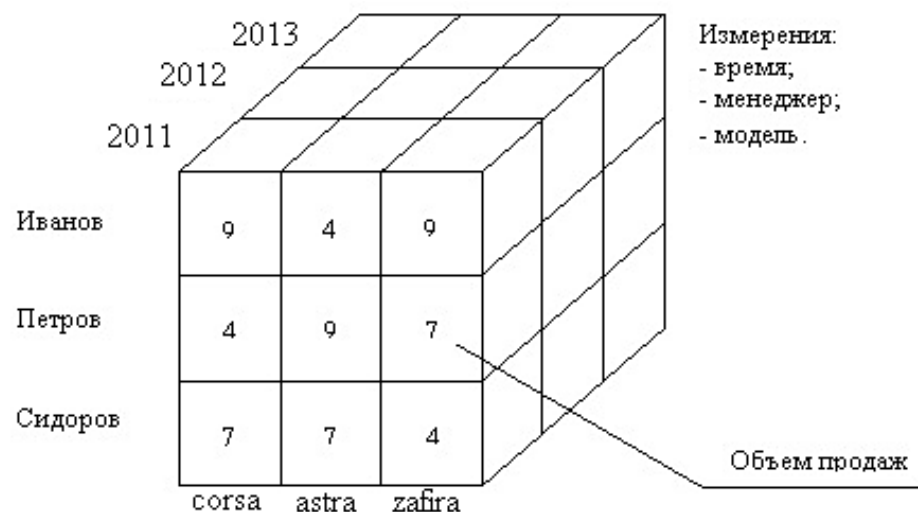


Постреляционная модель

Постреляционная модель базы данных (объектно-ориентированная) – многомерная структура на основе реляционной модели, представляет данные в виде многомерных таблиц.

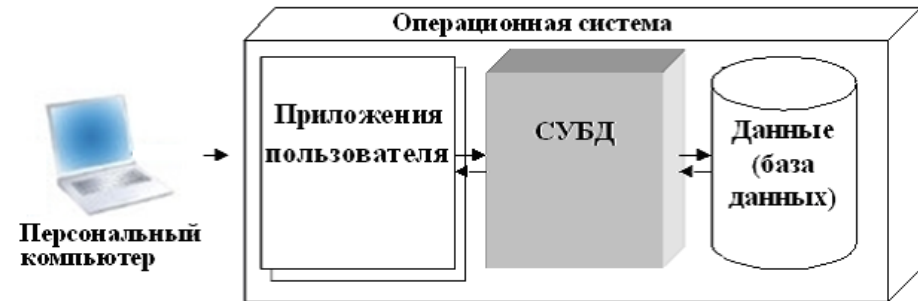
Активно используется для обеспечения аналитической обработки данных в режиме реального времени (OLAP - Online Analytical Processing).

Используется в структуре CRM, ERP, АБС – систем.



Классификация по степени распределенности:

- Локальные СУБД (все части локальной СУБД размещаются на одном компьютере);
- Распределённые СУБД (части СУБД могут размещаться на двух и более компьютерах).



Классификация по типу взаимодействия с обрабатывающей программой

СУБД могут быть:

- Автономные;
- Включающие язык программирования.

На взаимодействие с СУБД существенное влияние оказывают языковые факторы как самой СУБД, так и операционной системы, в среде которой работает СУБД.

Как правило, все СУБД поддерживают структурированный язык запросов SQL (Structured Query Language).

Помимо SQL СУБД может располагать дополнительными программно-инструментальными средствами программного интерфейса (VBA для MS Access).

Выбор СУБД зависит от:

1. Скорости работы (эффективности);
2. Количества пользователей;
3. Объема данных;
4. Имеющихся аппаратных и программных ресурсов;
5. Размещения данных в web.

Жизненный цикл программного продукта (ЖЦПП)

Это период времени, начинающийся с момента принятия решения о необходимости создания ПП и заканчивающийся в момент его полного изъятия из эксплуатации.

Этапы ЖЦПП

Этап ЖЦПП – это часть процесса разработки ПП, ограниченная по времени и заканчивающаяся определенным результатом. Этапы ЖЦПП:

- Постановка задачи (управление требованиями заказчика);
- Планирование работ по созданию ПП;
- Проектирование ПП;
- Разработка (программирование);
- Тестирование;
- Сопровождение и эксплуатация.

Жизненный цикл БД:

1. Исследование и анализ проблемы, для решения которой создаётся база данных.
2. Построение инфологической и даталогической модели.
3. Нормализация полученных инфологических и даталогических моделей. По окончании этого этапа, как правило получают заготовки таблицы БД и набор связей между ними (первичные и вторичные ключи).
4. Проверка целостности БД.
5. Выбор физического способа хранения и эксплуатации (аппаратная/программная платформы) базы данных.
6. Проектирование входных и выходных форм и отчетов.
7. Разработка интерфейса приложения.
8. Тестирование и отладка приложения.
9. Эксплуатация и сопровождение.
10. Вывод из эксплуатации: перенос данных в новую СУБД.

Постановка бизнес-задачи

1. Почему необходимо создание системы?
2. В чем Вы видите назначение системы?
3. Какие бизнес-возможности система должна реализовать?
4. Какие проблемы система должна решить?

Предметная область

Предметная область – это сфера человеческой деятельности, выделенная и описанная согласно установленным критериям. В описываемое понятие должны входить сведения о:

- Элементах;
- Явлениях;
- Отношениях;
- Процессах;

отражающих различные аспекты этой деятельности.

В описании предметной области должны присутствовать характеристики возможных **воздействий окружающей среды** на элементы и явления предметной области, а также **обратные воздействия** этих элементов и явлений **на среду**.

Специфика предметной области может оказывать существенное влияние на характер функционирования проектируемой интеллектуальной системы, выбор метода представления знаний, способов рассуждения о знаниях, и т. д.

Анализ предметной области

Шаги анализа предметной области:

- Выделение всех сущностей;
- Определение первоначальных требований к функциональности;
- Определение границ проекта;
- Документирование модели предметной области.

Исследование предметной области

Предметную область можно определить как объект или производственную систему со всем комплексом понятий и знаний о ее функционировании.

При исследовании проблемной области необходимы следующие знания:

1. О задачах, решаемых в производственной системе,
2. О стоящих перед системой целях.

Определяются также возможные стратегии управления и знания, используемые в процессе эксплуатации производственной системы.

Основные этапы проектирования базы данных

- Создание инфологической (концептуальной) модели;
- Создание даталогической (логической) модели;
- Физическое проектирование.

Инфологическое проектирование

Инфологическая модель создаётся без ориентации на какую-либо конкретную СУБД и модель данных. Конкретный вид и содержание инфологической модели базы данных определяется выбранным для этого формальным аппаратом.

Обычно инфологическая модель базы данных включает в себя:

- описание информационных объектов, или понятий предметной области и связей между ними.
- описание ограничений целостности, т.е. требований к допустимым значениям данных и к связям между ними.

Даталогическое проектирование

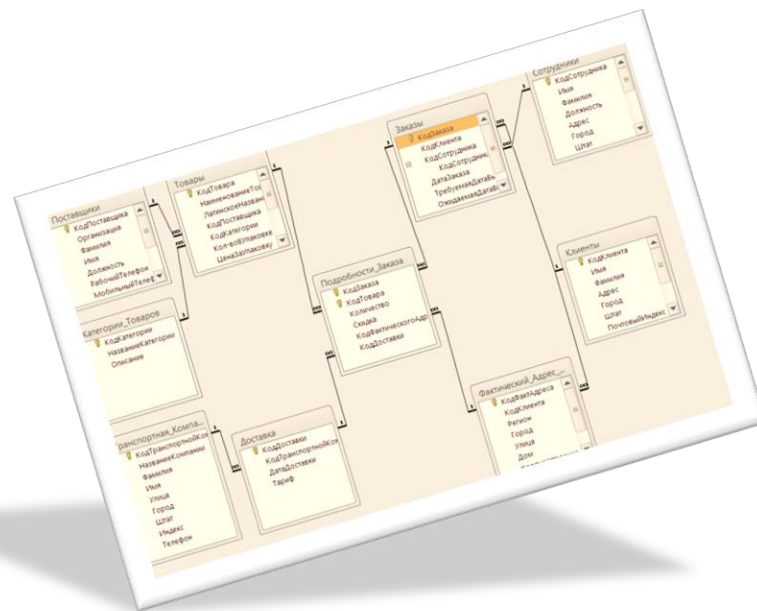
Даталогическое проектирование — создание схемы базы данных на основе конкретной модели данных, например, реляционной модели данных. Для реляционной модели данных даталогическая модель — набор схем отношений, обычно с указанием первичных ключей, а также «связей» между отношениями, представляющих собой внешние ключи.

На этапе логического проектирования учитывается специфика конкретной модели данных, но может не учитываться специфика конкретной СУБД.

Физическое проектирование

Физическое проектирование — создание схемы базы данных для конкретной СУБД. Специфика конкретной СУБД может включать в себя ограничения на именование объектов базы данных, ограничения на поддерживаемые типы данных и т.п. Кроме того, специфика конкретной СУБД при физическом проектировании включает выбор решений, связанных с физической средой хранения данных (выбор методов управления дисковой памятью, разделение БД по файлам и устройствам, методов доступа к данным), создание индексов и т.д.

РЕЛЯЦИОННАЯ МОДЕЛЬ ДАННЫХ



Целью разработки любой базы данных является хранение и использование информации о какой-либо предметной области. Для реализации этой цели имеются следующие инструменты:

- Реляционная модель данных - удобный способ представления данных предметной области;
- Язык SQL (Structured Query Language)- универсальный способ манипулирования такими данными.

Реляционная модель

- Каждый объект представлен в виде двумерной таблицы;
- Таблица – это двумерный массив, содержащий информацию об объекте. БД может состоять из одной или нескольких таблиц.

Таблица реляционной базы данных

Таблица состоит из следующих элементов:

- **Поле (столбец)** – это наименьшая единица хранимых данных, содержит значение одного из признаков, характеризующих объекты БД. Число полей в таблице соответствует числу признаков, характеризующих объекты БД.
- **Запись (строка)** – набор взаимосвязанных хранимых полей.
- **Ячейка** – содержит конкретное значение соответствующего поля (признака данного объекта).

Артикул	Наименование	Цена	Количество
1	Чашка "Золотая осень"	245,00р.	400
2	Блюдце "Золотая осень"	134,00р.	400
3	Набор "Керамика"	768,00р.	50

→ запись

↓ поле

ячейка

Создание таблицы








1. Создать новый объект типа таблица.
2. Задать:
 - Имена полей таблицы;
 - Типы данных;
 - Свойства полей;
 - Описание полей;
3. Задать первичный ключ;
4. Определить индексы;
5. Сохранить структуру таблицы.

Создание нового объекта типа «таблица»

Некоторые СУБД позволяют выбрать тип таблицы:

- MyISAM;
- InnoDB;
- MERGE;
- MEMORY (HEAP);
- ARCHIVE;
- CSV;
- И некоторые другие.

Имя таблицы: Добавить поле(я)

Имя	Тип 	Длина/значения ¹	По умолчанию ²
<input type="text"/>	INT 	<input type="text"/>	Нет 
<small>Много данных ENUM или SET? Открыть расширенный редактор</small>			
<input type="text"/>	INT 	<input type="text"/>	Нет 
<small>Много данных ENUM или SET? Открыть расширенный редактор</small>			
<input type="text"/>	INT 	<input type="text"/>	Нет 
<small>Много данных Открыть р ред</small>			

Комментарий к таблице:

MRG_MYISAM
MyISAM
BLACKHOLE
CSV
MEMORY
ARCHIVE
InnoDB

Пример: типы таблиц MySQL 5.5

В версии MySQL 5.5 поддерживается 9 различных типов таблиц:

- **InnoDB** – тип с поддержкой транзакций, откатов и защитой от потери данных. В данном типе таблиц используются блокировки на уровне записи и не блокирующее чтение, что позволило улучшить производительность при многопользовательском режиме работы. InnoDB сохраняет пользовательские данные в кластерных индексах, что позволяет компенсировать в/в для простых запросов основанных на первичных ключах.
- **MyISAM** – тип таблиц MySQL используемый в основном в Web-приложениях, хранилищах данных и других программных средах. Данный тип таблиц поддерживается всеми инсталляциями MySQL.
- **Memory** (HEAP) - хранит данные в оперативной памяти для очень быстрого доступа.
- **Merge** - используется для логического объединения одинаковых MyISAM таблиц и обращение к ним, как к единому объекту. Хорошо подойдет для очень больших хранилищ данных.
- **Archive** - решение для хранения больших объёмов информации, к которой не требуется частый доступ.
- **Federated** - предоставляет возможность объединять различные MySQL сервера для создания одной логической базы данных из нескольких физических машин. Подойдет для архитектур, которые поддерживают распределенное хранение данных.
- **CSV** - хранит пользовательские данные в текстовых файлах разделяя значения запятыми. Используется если необходим простой обмен с приложениями, которые умеют экспортировать/импортировать данные из CSV формата.
- **Blackhole** - принимает, но не возвращает никаких данных. Результатами любых запросов из таких хранилищ будут пустые выборки.
- **Example** - тестовый движок, не выполняет никаких функций, будет полезен только разработчикам, которые собираются писать свой движок, в качестве примера.

Работа с полями таблицы

- Имена полей (зависят от синтаксиса конкретной СУБД);
- Типы данных полей;
- Атрибуты полей;
- Описание полей (комментарии).

Понятие типа данных

Тип данных — фундаментальное понятие теории программирования. Тип данных определяет множество значений, набор операций, которые можно применять к таким значениям, и, возможно, способ реализации хранения значений и выполнения операций. Любые данные, которыми оперируют программы, относятся к определённым типам.

Общее представление о типах данных в СУБД

- **Числовые данные** — это все целые числа (без дробной части) и вещественные числа (с дробной частью).
- **Строковые данные** — последовательность символов, заключенная в кавычки.
- **Календарные данные** — тип для обозначение даты и времени.
- **NULL** — неопределенный тип данных.

Полный перечень поддерживаемых типов данных зависит от конкретной СУБД.

Основные атрибуты полей

- Размер поля;
- Поле счетчика (автоинкремента);
- Неотрицательность числового значения;
- Значение поля по умолчанию;
- Обязательное для заполнения (not null);
- Уникальное значение поля;
- Индексированное поле;
- Ключевое поле.

Ключевые поля таблиц

Ключ – это поле, которое однозначно определяет значения всех остальных полей в таблице. Может быть простым и составным. Каждое поле, не входящее в состав ключа называется не ключевым.

Ключ всегда уникален, т.е. в любой момент времени таблица БД не может содержать никакие две различные записи, имеющие одинаковые значения ключевых полей.

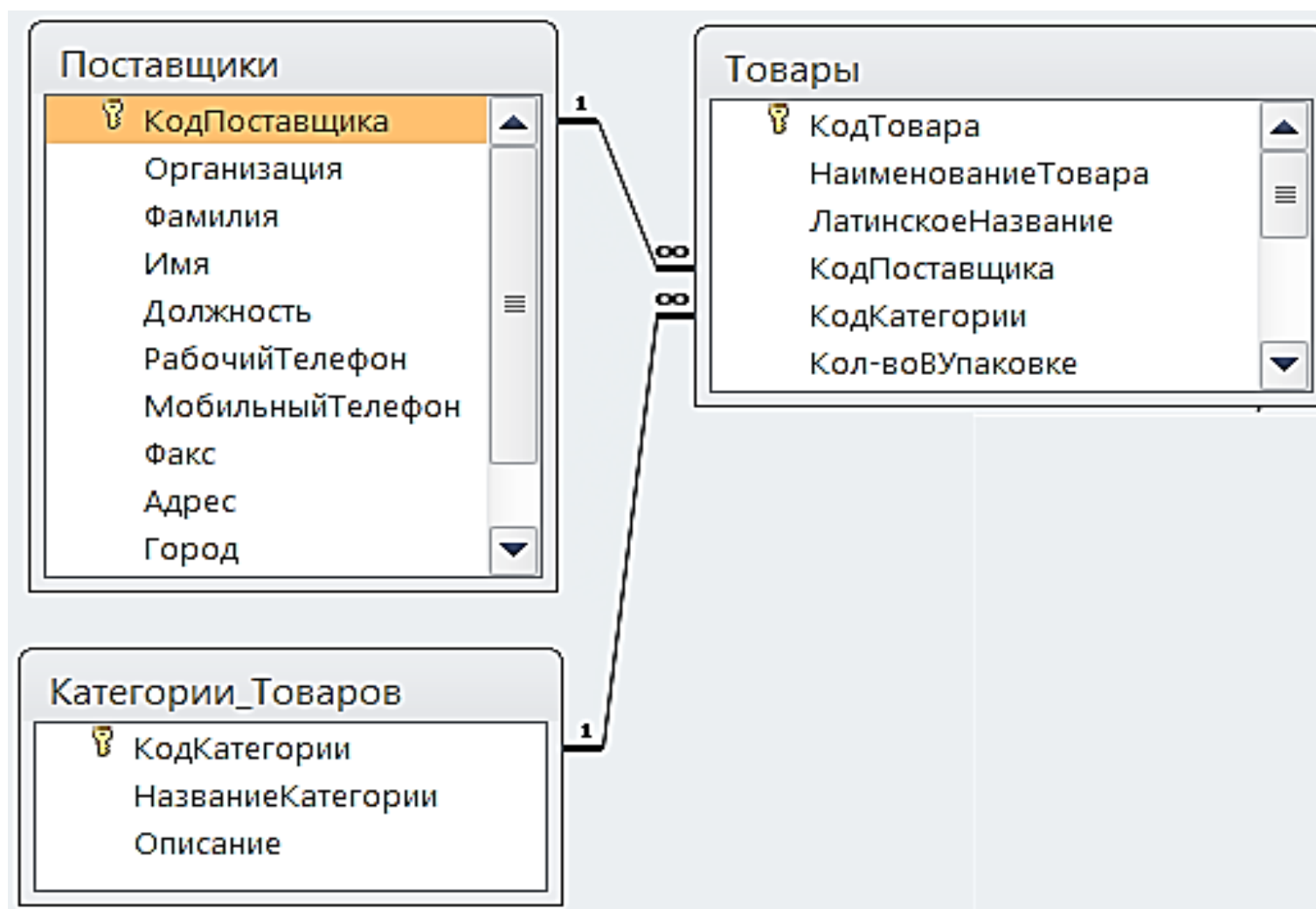
Первичный ключ

- Уникальный идентификатор записи в таблице;
- Увеличивает скорость поиска данных;
- Может быть **простым** (состоящим из одного поля) или **составным** (состоящим из нескольких полей таблицы);
- Может быть **естественным** (состоящим из информационных полей таблицы, однозначно определяющих каждую запись) или **искусственным** (представляющим собой дополнительное служебное поле).

Внешний ключ

представляет собой одно или несколько полей (столбцов), содержащих ссылку на поле или поля первичного ключа в другой таблице. Внешний ключ определяет способ связи таблиц.

Ключи



Повышение эффективности БД

Для повышения эффективности работы с данными средствами СУБД используются как минимум:

- нормализация баз данных;
- индексация данных.

Кроме того, существует множество способов повышения эффективности работы с данными средствами СУБД, которые зависят от программно-аппаратной реализации базы данных.

Нормализация баз данных

Нормализация – это процесс приведения отношений базы данных к виду, отвечающему нормальным формам (NF).

Как правило, обязательным и, зачастую, достаточным является приведение к третьей нормальной форме (3NF).

Основные цели нормализации:

- Исключение избыточности данных;
- Упрощение структуры БД;
- Устранение аномалий обновления.

Аномалии обновления в БД

Если в таблице данные хранятся с очевидной избыточностью (как правило - данные о нескольких объектах в одной таблице), то при обращении к этим данным с целью их изменения возникает множество проблем. Исторически эти проблемы получили название **аномалии обновления**.

Аномалии обновления могут трактоваться как неадекватности модели данных предметной области, (что говорит о том, что логическая модель данных неверна) или как необходимости дополнительных усилий для реализации всех ограничений, определенных в предметной области.

Виды аномалий

- Аномалия вставки (INSERT);
- Аномалия обновления (UPDATE);
- Аномалия удаления (DELETE).

Сотрудники						
КодСотрудника	Фамилия	НомерОтдела	Телефон	НомерПроекта	НазваниеПроекта	НомерЗдания
1	Иванов	1	123-45-67	1	Заря	1
2	Иванов	1	123-45-67	2	СтройТех	1
3	Озеров	1	123-45-68	1	Заря	2
4	Меньшова	2	123-44-45	1	Заря	3
5	Меньшова	2	123-44-45	2	СтройТех	2

Аномалия вставки

Невозможно вставить данные о проекте, в котором пока не задействован ни один сотрудник.

Причина аномалии: хранение в одной таблице данных более, чем об одном объекте.

Аномалия обновления

Фамилии сотрудников, наименования проектов, номера телефонов повторяются во многих записях (кортежах отношения). Если сотрудник меняет фамилию / проект меняет наименование / меняется номер телефона, то такие изменения необходимо *одновременно* выполнить во всех местах, где эти данные встречаются, иначе отношение станет некорректным (например, один и тот же проект в разных кортежах будет называться по-разному). Таким образом, обновление базы данных одним действием реализовать невозможно.

Причина аномалии: избыточность данных из-за хранения в одном отношении (таблице) разнородной информации.

Для поддержания отношения в целостном состоянии можно написать ***триггер***, который при обновлении одной записи корректно исправлял бы данные и в других местах.

Хранимые процедуры и триггеры

Большинство СУБД содержит некоторое количество программного кода в виде триггеров и хранимых процедур.

Хранимые процедуры - это процедуры и функции, хранящиеся непосредственно в базе данных в откомпилированном виде и которые могут запускаться пользователями или приложениями, работающими с базой данных. Хранимые процедуры обычно пишутся либо на специальном процедурном расширении языка SQL (например, PL/SQL для ORACLE или Transact-SQL для MS SQL Server), или на некотором универсальном языке программирования с включением в код операторов SQL в соответствии со специальными правилами такого включения. Основное назначение хранимых процедур - реализация бизнес-процессов предметной области.

Триггеры - это хранимые процедуры, связанные с некоторыми событиями, происходящими во время работы базы данных. В качестве таких событий выступают операции вставки, обновления и удаления строк таблиц. Если в базе данных определен некоторый триггер, то он запускается *автоматически* всегда при возникновении события, с которым этот триггер связан. Важным является то, что пользователь не может обойти триггер. Триггер срабатывает независимо от того, кто из пользователей и каким способом инициировал событие, вызвавшее запуск триггера. Таким образом, основное назначение триггеров - автоматическая поддержка целостности базы данных.

Аномалии удаления

При удалении некоторых данных может произойти потеря другой информации. Например, если закрыть проект «Заря» и удалить все строки, в которых он встречается, то будут потеряны все данные об участвующих в нем сотрудниках. Если удалить сотрудника Иванова, то будет потеряна информация о номере телефона, и т.п.

Причина аномалии - хранение в одном отношении разнородной информации (и о сотрудниках, и о проектах, и о работах по проекту).

Нормальные формы

Процесс проектирования БД представляет собой процесс нормализации схем отношений, причем каждая следующая нормальная форма обладает свойствами лучшими, чем предыдущая. Каждой нормальной форме соответствует некоторый определенный набор ограничений, и отношение находится в некоторой нормальной форме, если удовлетворяет свойственному ей набору ограничений.

Основные свойства нормальных форм:

- каждая следующая нормальная форма в некотором смысле лучше предыдущей;
- при переходе к следующей нормальной форме свойства предыдущих нормальных свойств сохраняются.

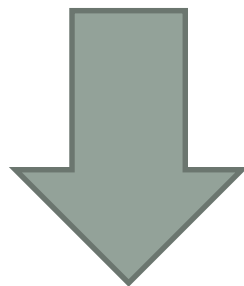
Первая нормальная форма (1NF)

Таблица находится в первой нормальной форме, если каждый её атрибут атомарен и любое ключевое поле не пусто.

Атрибут **атомарен**, если его значение теряет смысл при *любом* разбиении на части или переупорядочивании. И наоборот, если какой-либо способ разбиения на части не лишает атрибут смысла, то атрибут **неатомарен**.

Первая нормальная форма (1NF)

BookOrder				
Код	ДатаЗаказа	Покупатель	ИнформацияОЗаказе	
1	03.03.2014	Иванов Иван	Петров В.В. Занимательная математика. М.:2009, 450 с., 1 шт.	
2	05.03.2014	Петров В.С.	Шекспир У. Сонеты. М.: изд. Рипол Классик, 2010, 352 с., 2 шт.	
3	14.03.2014	Алешина В.Р.	Гумилев Л. Открвтие Хазарии. М., изд. "Акт", 2010, 336 с., 1 шт.	
4	25.03.2014	И. Краснова	Балашов М.Я. Вязание крючком. 2 шт.	



BookOrder_NF1												
Код	ДатаЗаказа	ПокупательФамил	ПокупательИмя	ПокупательОтч	АвторФамили	АвторИмя	НазваниеКни	ГородИзд	Издательст	ГодИзд	КоличествоСтр	КоличествоЭкз
1	03.03.2014	Иванов	Иван		Петров	Виктор	Занимательная	СПб	Питер	2009	450	1
2	05.03.2014	Петров	Василий	Сергеевич	Шекспир	Уильям	Сонеты	М	Рипол Классик	2010	352	2
3	14.03.2014	Алешина	Вероника	Романовна	Гумилев	Лев	Открытие Хазар	М	Акт	2010	336	1
4	25.03.2014	Краснова	Валентина		Балашов	Михаил	Вязание крбчко	СПб	Питер	2013	178	1

Вторая нормальная форма (2NF)

Соблюдена первая нормальная форма и все поля таблицы, не входящие в первичный ключ, зависят от первичного ключа

BookOrder_NF1												
Код	ДатаЗаказа	ПокупательФамил	ПокупательИм	ПокупательОт	АвторФамили	АвторИмя	НазваниеКни	ГородИзд	Издательств	ГодИзд	КоличествоСтр	КоличествоЭкз
1	03.03.2014	Иванов	Иван		Петров	Виктор	Занимательная	СПб	Питер	2009	450	
2	05.03.2014	Петров	Василий	Сергеевич	Шекспир	Уильям	Сонеты	М	Рипол Классик	2010	352	
3	14.03.2014	Алешина	Вероника	Романовна	Гумилев	Лев	Открытие Хазар	М	Акт	2010	336	
4	25.03.2014	Краснова	Валентина		Балашов	Михаил	Вязание крбчко	СПб	Питер	2013	178	



Третья нормальная форма (3NF)

Соблюдена вторая нормальная форма и ни одно из не ключевых полей таблицы не зависит от любого другого не ключевого поля.

Копия BookOrder_NF3					
Код	ДатаЗаказа	КодПокупателя	КодАвтора	КодКниги	КоличествоЭкз
1	03.03.2014	1	1	1	1
2	05.03.2014	2	2	2	2
3	14.03.2014	3	3	3	1
4	25.03.2014	4	4	4	1

Clients				
КодПокупат	Фамилия	Имя	Отчество	Информация
1	Иванов	Иван		
2	Петров	Василий	Сергеевич	
3	Алешина	Вероника	Романовна	
4	Краснова	Валентина		

Books						
КодКниги	КодАвтора	Название	Издательство	ГородИзд	ГодИзд	КоличествоСтр
1	1	1 Занимательная мат	Питер	СПб	2009	450
2	2	2 Сонеты	Рипол Классик	М	2010	352
3	3	3 Открытие Хазарии	Акт	М	2010	336
4	4	4 Вязание крючком	Питер	СПб	2013	178

Authors			
КодАвтора	Фамилия	Имя	Информация
1	Петров	Виктор	
2	Шекспир	Уильям	
3	Гумилев	Лев	
4	Балашов	Михаил	

Нормальные формы более высоких порядков

В большинстве случаев достаточно привести базу данных к 3NF. Но существуют более высокие формы нормализации:

- нормальная форма Бойса-Кодда (BCNF);
- четвертая нормальная форма (4NF);
- пятая нормальная форма (5NF);
- доменно-ключевая нормальная форма (DKNF);
- шестая нормальная форма (6NF).

Алгоритм приведения базы к 3NF:

1NF

- Устраните повторяющиеся группы в отдельных таблицах;
- Создайте отдельную таблицу для каждого набора связанных данных;
- Идентифицируйте каждый набор связанных данных с помощью первичного ключа.

2NF

- Создайте отдельные таблицы для наборов значений, относящихся к нескольким записям;
- Свяжите эти таблицы с помощью внешнего ключа.

3NF

- Устраните поля, не зависящие от ключа.

Денормализация

намеренное приведение структуры базы данных из 3NF в состояние, не соответствующее критериям нормализации, обычно проводимое с целью ускорения операций чтения из базы за счет добавления избыточных данных.

Основная цель денормализации — увеличение производительности за счет увеличения избыточности, возможных коллизий, а также некоторым неудобством составления сложных выборок.

Основные виды денормализации:

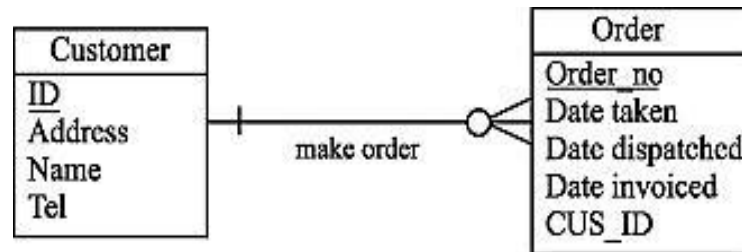
- Нисходящая;
- Восходящая;
- Внутритабличная;
- Методом «Разделяй и властвуй»;
- Методом слияния таблиц.

Пример: нисходящая денормализация

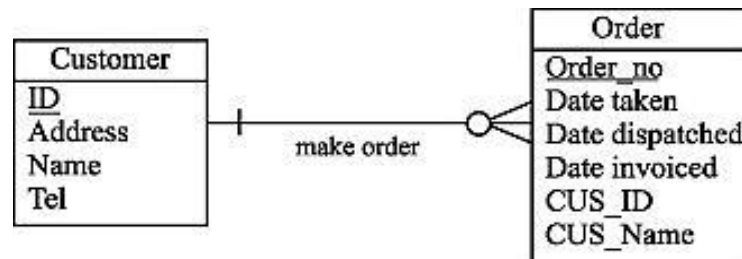
предлагает перенос атрибута из одной (родительской) сущности в подчиненную (дочернюю) сущность.

Представляет собой процесс введения избыточных колонок в подчиненных таблицах с целью устранения операций соединения.

- До денормализации



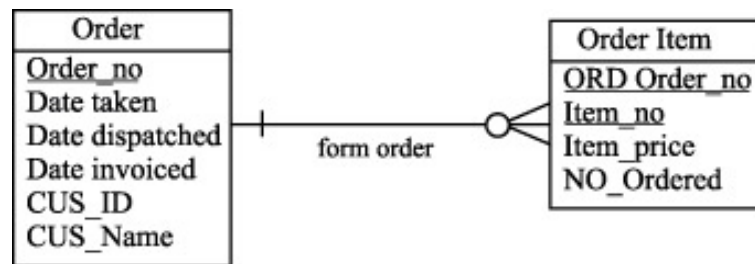
- После денормализации



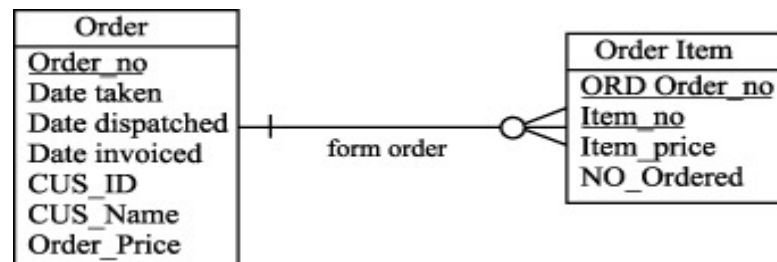
Пример: восходящая денормализация

предлагает перенос атрибута из подчиненной (дочерней) сущности в родительскую сущность, обычно в форме итоговых данных.

- До денормализации



- После денормализации



Индексы баз данных

- Внутренний способ организации данных в таблице для их оптимальной выборки.
- Внутренняя таблица, состоящая из наборов уникальных для данной таблицы значений и соответствующий им список указателей на страницы данных, где значения находятся физически.

Назначение индексов - ускорить выборку записанной в базе данных информации.

Индексы

Индекс аналогичен по своей структуре и назначению предметному указателю, приведенному в конце книги. Как и предметный указатель книги, индекс базы данных упорядочен, и каждый элемент индекса содержит признак искомого объекта (поле индексирования), а также один или несколько указателей на место его расположения.

Файл, содержащий логические записи, называется **основным файлом данных**, а файл, содержащий индексные записи, — **индексным файлом**.



Индексы

- ***Уникальные индексы***

- не допускается повторяющихся значений в индексе

- ***Неуникальные индексы***

- допускаются повторяющиеся значения в индексе

- ***Составные индексы***

- состоят из нескольких полей, могут быть уникальными и не уникальными

Индексы

- ***Кластеризованные индексы***

- Физически отсортированная таблица.
- Записи физически отсортированы и в процессе работы поддерживают заданный порядок сортировки.
- Физический порядок записей соответствует порядку индексирования.
- Таблица может содержать только один кластеризованный индекс.

- ***Некластеризованные индексы***

- Набор с номерами страниц, где находятся указатели на нужную страницу.
- Физический порядок отличается от индексного.
- Можно создать более одного индекса для таблицы.

Многоуровневые индексы

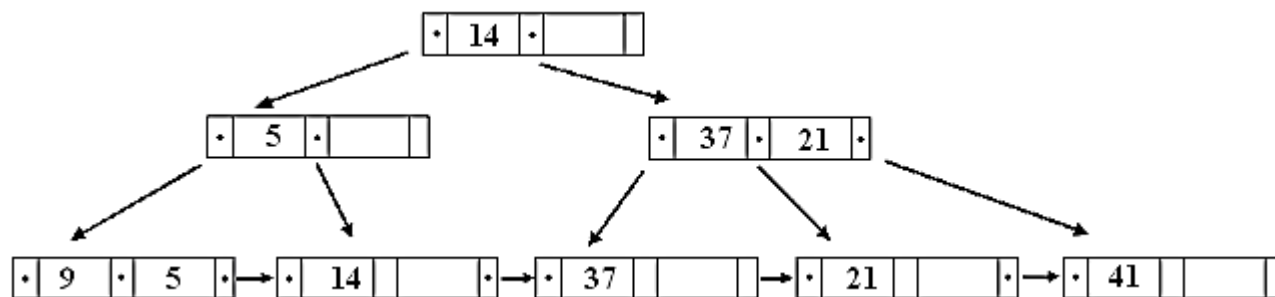
При возрастании размера индексного файла и расширении его содержимого, время поиска нужного индекса, а также время его обновления также значительно возрастает. Данная проблема решается, в частности, за счет применения многоуровневого индекса.

Идея многоуровневого индекса которого заключается в дроблении одного общего индекса на иерархию нескольких индексов меньшего объема. Таким образом получается иерархия индексов, каждый элемент которой способен разместиться в память. Начальную загрузку для этого метода делают из сортированного файла.

Многоуровневые индексы

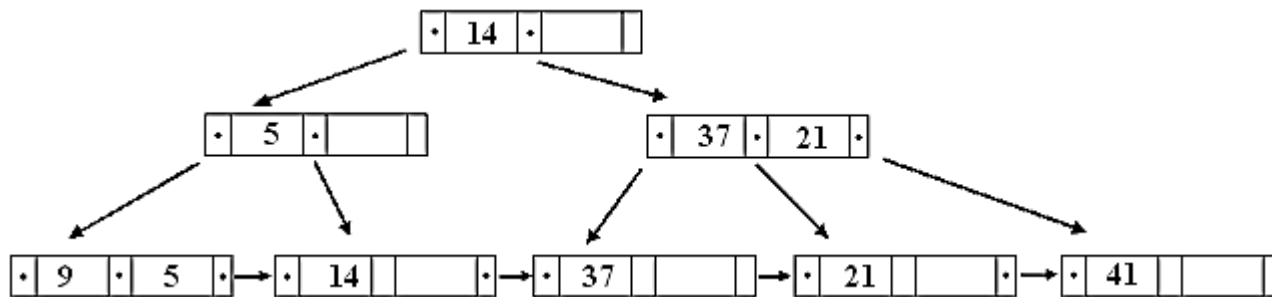
Многоуровневые индексы, как правило, строятся с использованием деревьев:

Таб_Номер	Фамилия	Должность	Оклад	Отдел
5	Мамцов	Менеджер	115,00р.	3
9	Серов	Менеджер	115,00р.	3
14	Сидоров	Менеджер	115,00р.	3
21	Иванов	Инженер	110,00р.	5
37	Петров	Инженер	140,00р.	5
41	Муромцев	Механик	97,00р.	5



Поиск в таком индексе осуществляется следующим образом: если поисковое значение меньше или равно значению хранимому в узле, то для перехода к следующему узлу используется указатель, расположенный в данном узле последним.

Пример (алгоритм) поиска с помощью многоуровневого индекса:



- Для обнаружения записи 21 (или любой другой) следует начать поиск с корневого узла.
- Поскольку значение 21 больше 14, необходимо перейти по указателю, расположенному справа, который приведет нас к узлу второго уровня с ключевыми значениями 37 и 21.
- Далее нужно перейти по указателю, расположенному слева от значения 21, который и приведет нас к искомой записи 21.

ОБЕСПЕЧЕНИЕ ЦЕЛОСТНОСТИ ДАННЫХ

Межтабличные связи

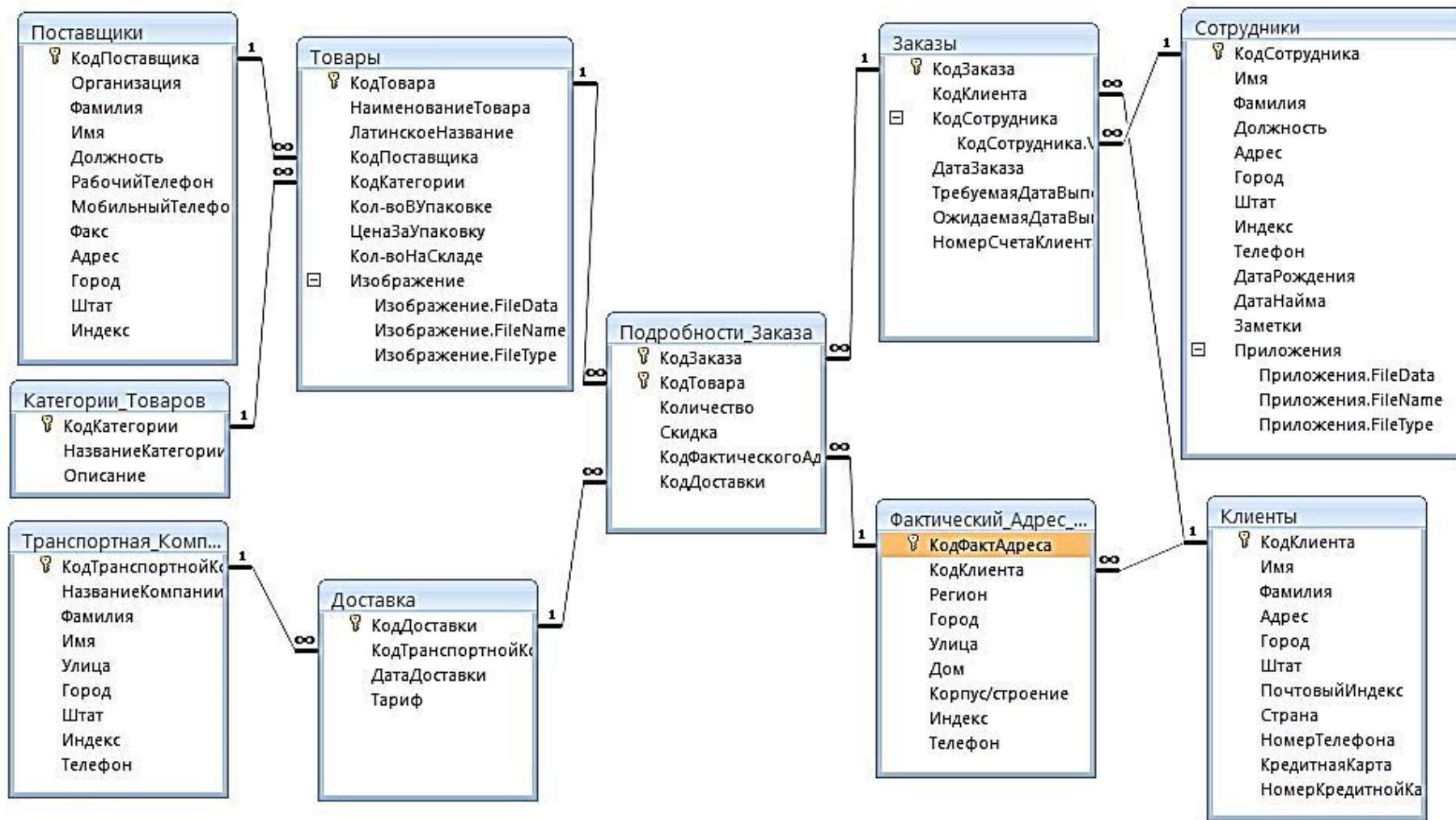
База данных, состоящая из нескольких таблиц

Реляционная база данных на физическом уровне состоит из таблиц, между которыми могут существовать связи по ключевым полям.

Для ее построения необходимо:

- Привести данные к 3 NF;
- Связать таблицы по ключевым полям;
- Определить типы связей между таблицами;
- Проверить целостность данных;
- Определить индексы.

Пример схемы базы данных



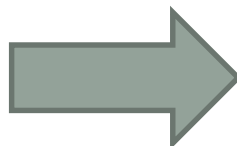
Функции межтабличных связей:

- Межтабличные связи предоставляют сведения для структуры запросов;
- Межтабличные связи предоставляют сведения для структуры форм и отчетов;
- Межтабличные связи являются основой, с помощью которой можно обеспечить целостность данных.

Связи между таблицами

Один к одному

Businesses	
КодКлиента ▾	НаименованиеКлиента
1	Suncoast Cinema
2	Fisher Trailers
3	Quick Auto Detail
4	Keller And Company
5	Cooler Air Conditioning
6	Western Air USA
7	Car Care
8	Rising Waters
9	Genie, Inc.
10	Tranck Retailer
11	Express Cleaners
12	Hytter Services



SecurityIDs	
КодКлиента ▾	СпецКодКлиента ▾
1	34545
2	45645
3	45645
4	67899
5	25765
6	68980
7	23546
8	96467
9	38674
10	24565
11	57979
12	26452

Связи между таблицами

Один ко многим

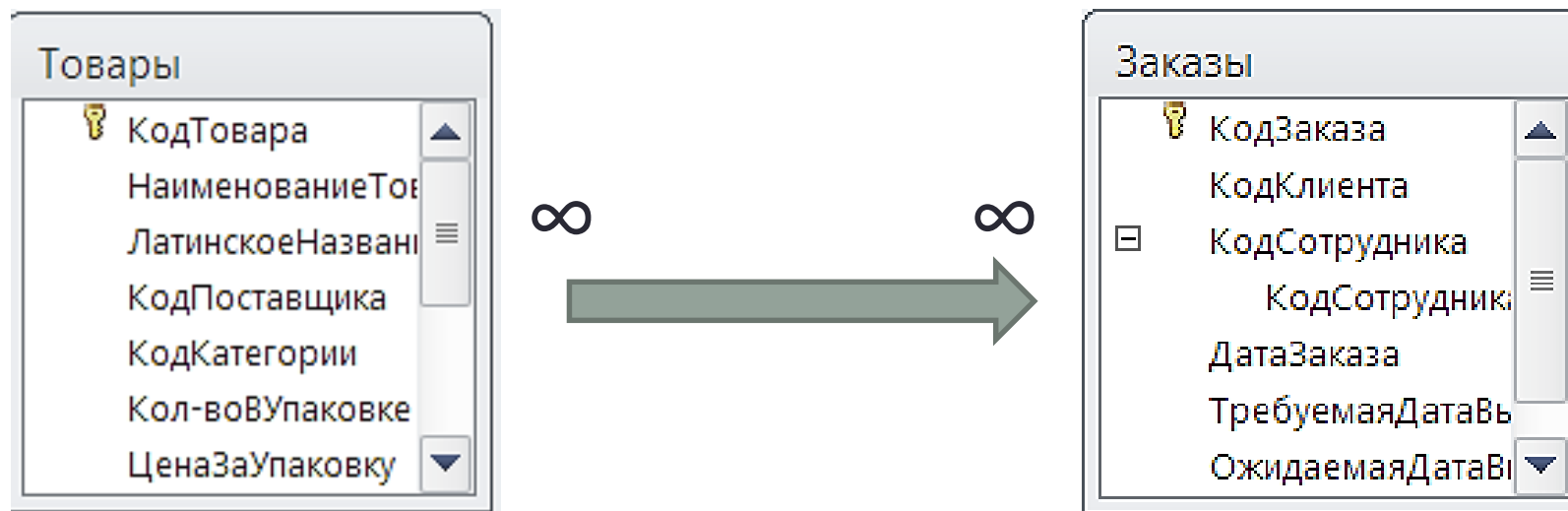
Категории_Товаров		
Код типа	НазваниеКатегории	
1	Луковые	Весна, лето и осень, сло
2	Кактусы	Комнатные кактусы
3	Почвопокровные	Многолетняя трава, веч
4	Газонная трава	Газонная трава для хол
5	Цветы	Широкий выбор цветов
6	Болотные растения	Растения для водных са
7	Грунт/Песок	Грунт для пересадки, то
8	Удобрение	Разнообразные удобрен
13	Деревья	Вечнозеленые и листвен
14	Душистые травы	Для придание вкуса и а
15	Бонсай Оборудование	Оборудование для бонс
16	Розы	Много сортов роз
17	Рододендрон	Морозостойкое растени
18	Борьба с вредителями	Нетоксичные средства
19	Плотоядные	Плотоядные растения
20	Инструменты	Садовый инвентарь
21	Ягодные кустарники	Небольшие плодовые к



Товары		
Код товара	НаименованиеТовара	КодКатегории
2	Осеннецветущий крокус	1
69	Анемон (ветреница корончат	1
71	Штернбергия желтая	1
160	Сибирский Ирис	1
161	Желтый нарцисс	1
162	Пионы	1
163	Лилия	1
164	Бегония	1
190	Тюльпаны	1
123	Опунция	2
9	Вязель пестрый	3
10	Плющ обыкновенный	3
22	Копытень европейский	3
74	Зверобой чашечковидный	3
125	Папоротник	3
24	Пырей	4
179	полевица тонкая	4
180	Полевица болотная	4
181	Красная овсяница	4
182	Многолетний плевел	4
183	Полевица белая	4
80	Ландыш майский	5
81	Гортензия метельчатая	5

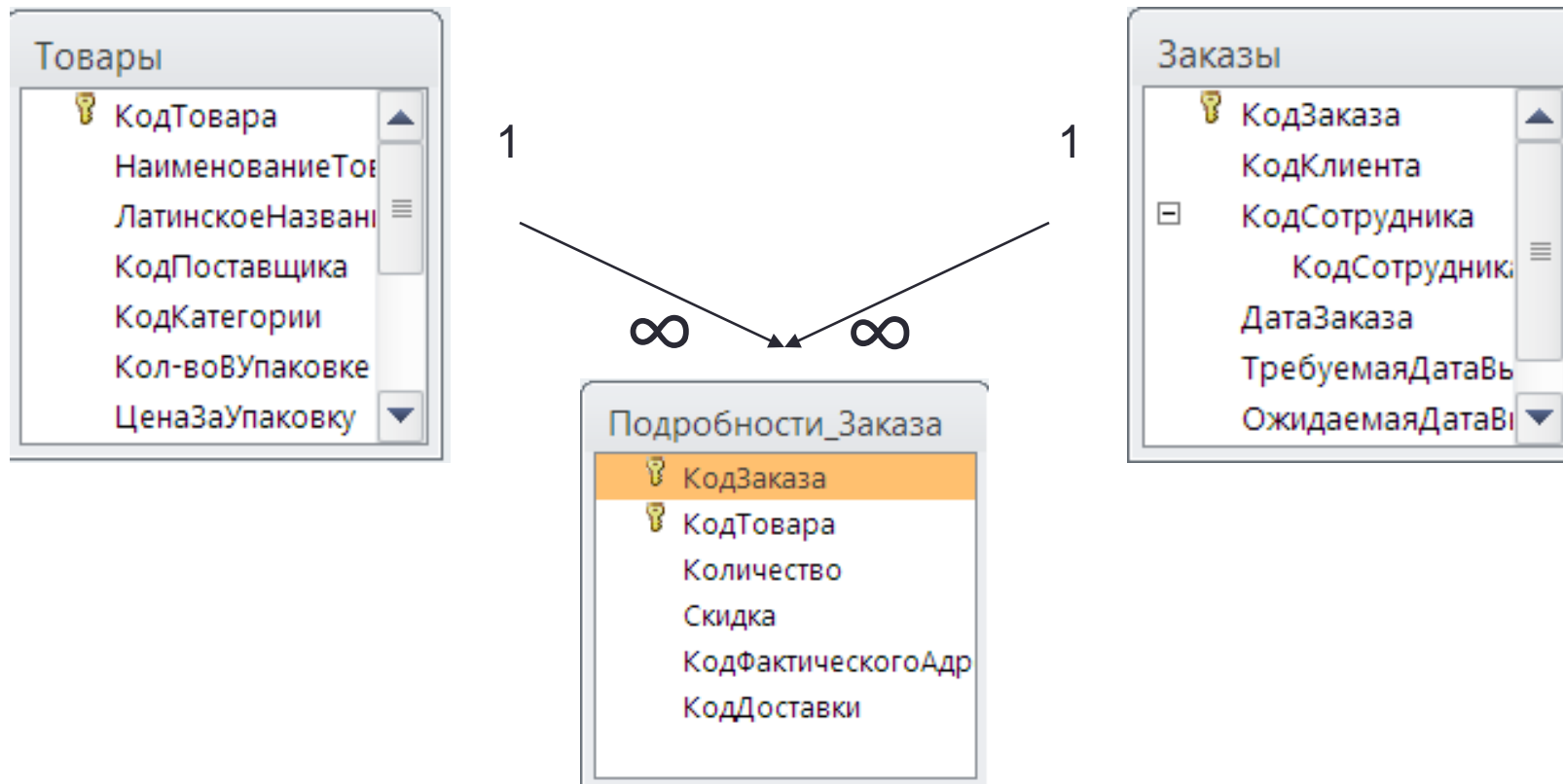
Связи между таблицами

Многие ко многим



Связи между таблицами

Многие ко многим (реализация: один ко многим)



Обеспечение целостности данных

Целостность данных означает систему правил, используемых для поддержания связей между записями в связанных таблицах, а также обеспечивающих защиту от случайного удаления или изменения связанных данных.

Основные правила обеспечения целостности данных

- Невозможно ввести в поле внешнего ключа связанной таблицы значение, не содержащееся в ключевом поле главной таблицы;
- Не допускается удаление записи из главной таблицы, если существуют связанные с ней записи в подчиненной таблице;
- Невозможно изменить значение первичного ключа в главной таблице, если существуют записи, связанные с данной записью.

Нарушение ссылочной целостности

Ссылочная целостность может нарушиться в результате операций вставки (добавления), обновления и удаления записей в таблицах.

В определении ссылочной целостности участвуют две таблицы - родительская и дочерняя, для каждой из них возможны перечисленные выше операции, поэтому существует шесть различных вариантов, которые могут привести либо не привести к нарушению ссылочной целостности.

Варианты нарушения ссылочной целостности для главной (родительской) таблицы:

- **Вставка.** Возникает новое значение первичного ключа. Существование записей в главной таблице на которые нет ссылок из подчиненной таблицы допустимо, операция не нарушает ссылочной целостности.
- **Обновление.** Изменение значения первичного ключа в записи **может привести к нарушению ссылочной целостности.**
- **Удаление.** При удалении записи удаляется значение первичного ключа. Если есть записи в подчиненной таблице, ссылающиеся на ключ удаляемой записи, то значения внешних ключей станут некорректными. **Операция может привести к нарушению ссылочной целостности.**

Варианты нарушения ссылочной целостности для подчиненной (дочерней) таблицы:

- **Вставка.** Нельзя вставить запись в подчиненную таблицу, если для новой записи значение внешнего ключа некорректно. **Операция может привести к нарушению ссылочной целостности.**
- **Обновление.** При обновлении записи в подчиненной таблице можно попытаться некорректно изменить значение внешнего ключа. **Операция может привести к нарушению ссылочной целостности.**
- **Удаление.** При удалении записи в подчиненной таблице ссылочная целостность не нарушается.

Основные стратегии поддержания ссылочной целостности

- **RESTRICT (ОГРАНИЧИТЬ)** - не разрешать выполнение операции, приводящей к нарушению ссылочной целостности. Это самая простая стратегия, требующая только проверки, имеются ли кортежи в дочернем отношении, связанные с некоторым кортежем в родительском отношении.
- **CASCADE (КАСКАДИРОВАТЬ)** - разрешить выполнение требуемой операции, но внести при этом необходимые поправки в других отношениях так, чтобы не допустить нарушения ссылочной целостности и сохранить все имеющиеся связи. Изменение начинается в родительском отношении и каскадно выполняется в дочернем отношении. В реализации этой стратегии имеется одна тонкость, заключающаяся в том, что дочернее отношение само может быть родительским для некоторого третьего отношения. При этом может дополнительно потребоваться выполнение какой-либо стратегии и для этой связи и т.д. Если при этом какая-либо из каскадных операций (любого уровня) не может быть выполнена, то необходимо отказаться от первоначальной операции и вернуть базу данных в исходное состояние. Это самая сложная стратегия, но она хороша тем, что при этом не нарушается связь между кортежами родительского и дочернего отношений.

Эти стратегии являются стандартными и присутствуют во всех СУБД, в которых имеется поддержка ссылочной целостности.

Дополнительные стратегии поддержания ссылочной целостности

- **SET NULL (УСТАНОВИТЬ В NULL)** - разрешить выполнение требуемой операции, но все возникающие некорректные значения внешних ключей изменять на null-значения. Эта стратегия имеет два недостатка. Во-первых, для нее требуется допустить использование null-значений. Во-вторых, записи дочернего отношения теряют связь с записями родительского отношения.
- **SET DEFAULT (УСТАНОВИТЬ ПО УМОЛЧАНИЮ)** - разрешить выполнение требуемой операции, но все возникающие некорректные значения внешних ключей изменять на некоторое значение, принятое по умолчанию. Достоинство этой стратегии по сравнению с предыдущей в том, что она позволяет не пользоваться null-значениями. Недостатки: во-первых, в родительском отношении должен быть потенциальный ключ как значение по умолчанию для внешних ключей; во-вторых, как и в предыдущем случае, кортежи дочернего отношения теряют связь с кортежами родительского отношения. Установить, с каким кортежем родительского отношения были связаны измененные кортежи дочернего отношения, после выполнения операции уже нельзя.
- **IGNORE (ИГНОРИРОВАТЬ)** - выполнять операции, не обращая внимания на нарушения ссылочной целостности.