Требования

Клиентская часть на react + typescript

Проект должен быть расположен в репозитории (один репозиторий и на клиент и на сервер)

Каждая группа реализовывает один проект в рамках группы(один проект на всех)

Тематика проекта любая

В рамках работы над проектом допустимо разделение команды по ролям

Групповой проект ведется в одном едином репозитории

Допустимые роли на проект:

Бекенд

Фронтенд (обязательно адаптивная вёрстка)

Дизайнер1ч

Аналитик 1ч

Тимлид 1ч отвечает полностью за проект

Тестировщик

Любой проект начинается с концепции (цель и суть проекта, описание основного функционала, список платформ на которых он исполняется, основная фишка приложения, краткое описание лора)

На основании концепции аналитик пишет требования, либо тз

Описывается детально требуемый функционал, так чтобы было понятно и заказчику, и исполнителю

Инструменты и организация работы

Вся разработка и все задачи должны вестись в программе Mirror

Там должно быть:

Название приложения

Состав команды и записанными ролями и с именем учетки в гитхабе

Должна быть прописана ссылка на репозиторий с проектом, ссылка на дизайны в фигме

Доска разделённая на колонки:

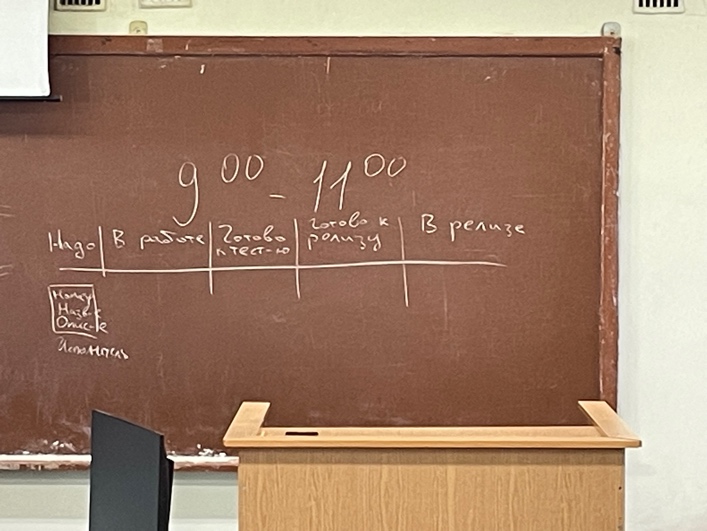
Надо делать

В работе

Можно тестировать

Готово к релизу

В релизе



Алгоритм работы – единственный способ организации работы в команде

Каждая задача выполняется в репозитории в отдельной ветке которая клонироутеся от ветки dev (основное название проекта-номер-название ветки), после того как задача сделана разработчик спуливает все последние изменения из dev и разрешает конфликты которые могут быть, после этого разработчик перетаскивает задачу к готово к тестированию, тестировщик переключается в эту ветку тестирует задачу, находит баги, оформляет баги следующими задачами, когда все баги успешно исправлены тестировщик или разработчик перетаскивает к готово к релизу и оформляет пул реквест на тимлида

В проекте две основные ветки main и dev

В ветке dev в каждый момент времени лежит рабочая версия приложения, в ветку dev пушить может только тимлид (не пушит, а разрешает pull-request)

Работа с репозиторием

Git checkout -b develop создать ветку

git checkout develop переключиться на ветку

git pull origin develop. Спулить изменения с девелоп

git checkout -b BDSM-123-pain создать свою ветку

git add .

git commit -m «комментарий».

Git pull origin develop

Решить конфликты

Git push origin BDSM-123-pain

Аналитик

Концепция требования api описание дизайна и тп должны лежать в этом же репозитории в файлах md

К следующей лекции написать сосавы команд и списком ролей, ссылку на доску и репозиторий, концепция

Поставить программу open server

16.09 Синтаксис PHP

$\_1

$1a

PHP очень требователен к наличию точек с запятой

В PHP нету типа данных undefind, null есть

Конкатенация строк в php осуществляется с помощью инструкции точка.

$a=’Вася’;

$b=’Маша’;;

$a.$b=’ВасяМаша’;

В PHP из коробки в строку мы можем вставить значение переменной.

$arr=[ ]; - массив

$arr[]=123; - добавить элемент в массив

Ассоциативные массивы != объектам

$arr=array(

‘name’=>9,

‘age’=>18);

$arr=[‘name’]

Объекты в PHP

$obj = new stdClass();

$obj->key1=’Вася’;

Классы

class Point{…}

Наследование с помощью команды extends

Class Name {

Public

Protected

Private}

Модификатор доступа может быть задан для любого Элемента. По умолчанию используется значение public.

Public – позволяет обратиться к методу или свойству класса в экземпляре класса (снаружи) и в классе потомке к методу или свойству родителя.

Protected – запрещает вызов метода в экземпляре класса,

Private – protected, только дополнительно запрещены вызовы методов классов родителей, в классе потомка.

У классов  PHP есть метод конструктора и деструктора.

Function\_\_construct(){

}

Function decstruct(){

}

Parent::\_\_construct();

В PHP можно управлять вызовом конструктора родителя, потому что вызов конструктора родителя является необязательным.

Конструктор родителя можно явно вызвать в конструкторе потомка, в случае если его не вызвали, он не будет вызван. Если в родителе конструктор есть, а в потомке конструктора нет, то в потомке конструктор родителя будет вызван автоматически.

Статичный класс – класс методы которого могут быть вызваны без создания экземпляра класса.

$this->

Супер глобальная переменна - $\_GET

$\_POST параметры постового запроса

У гетового запроса отсутствует тело запроса, и его параметры передаются в адресной строке. У постового запроса есть body, и его параметры мы можем передать не только в адресной строке, но и в body.

Для того чтобы написать бекенд мне не нужен клиент.

В запросах могут быть заголовки, к запросу могут быть прикреплены куки файлы.

Основной способ аунтефикации пользователя, это его определение по специализированному ключу – токену, который имеет ограниченный срок жизни и выдается клиенту на 1 сессию.

Для снижения нагрузки на бекенд применяется механизм кэширования, суть которого заключается в следующем: каждый новый запрос в бекенд, обрабатывается, на него формируется ответ и этот ответ складывается в так называемый кэш и в случае прихода в бекенд аналогичного запроса бекнд отвечает на него ответом из кэша.

Принцип работы бэкенда на проекте:

Любой запрос на бэкенд должен быть обработан единообразным способом, и на него должен быть дан однообразный ответ.

Result:’ok’|’error’,

Data?:any

Error?:{

Code:number,

Text:string

}

В проекте должен быть api, где описаны все запросы, которые может обработать бэкенд (форматы ответов и возможные ошибки)

Порядок работы:

Если необходимо написать новый метод на бэкенде, сначала он описывается в файлике api.md, описывается адрес метода, его параметры, ответы, возможные ошибки.

ДЗ: на основании проекта необходимо разработать документацию по методам необходимым в рамках проекта.

На клиенте необходимо разработать структуру страниц клиента и расписать базовый функционал.

***30.09 Разбор NoPaneNoGame***

Репозиторий состоит из двух папок: client и server.

В файле readme написана информация о развертывании проекта и его запуску.

Файл API – документ, описывающий методы бэкенда.

При разработке нового сервисного метода он сначала описывается в файле api, и только после этого пишется код. Ответственность за ведение документа api лежит на бэкенд разработчике. Тимлид имеет право не принимать pull-request на новый метод, при отсутствии его описания в документе.

Бэкенд.

Для запуска бэкенда необходимо всё содержимое папки server скопировать в соответствующую папку в openServer. Либо можно настроить openserver, чтобы он целился в папку на проекте. Бэкенд всегда возвращает ответы в виде json файлов. Бэкенд спроектирован в парадигме MVC (Model – бизнес логика приложения, View – представление, Controller – связующее звено). Модели не знают о представлениях, а представления не знают о модели. Представления должны быть тупыми, не содержать никакой бизнес-логики. Модели не знают, как они представляются и работают только с данными.

Структура бекенда.

Весь бэкенд

Api/

Index.php

Application/

Модули

Answer.php

Application.php

Файл index.php является единственной точкой входа в php, в нём мы определяем формат возвращаемого ответа и список поддерживаемых методов, также там разрешены кросс-доменные запросы.

В index.php есть два header один определяет формат ответа второй разрешает кросс-доменные запросы

Подключение файлов в php осуществляется множеством способов, в частности будем использовать метод requireOnce.

Функция result определяет какие методы поддерживает бэкенд и каким методом из application этот входящий запрос будет обработан.

Каждый новый метод сначала объявляется в index.php. Index.php формирует ответ с помощью команды echo, функции json\_encode и метода response класса answer.

Answer.php

В бэкенде все реализовано на клссах, кроме answer.php.

В answer.php объявлен класс, отвечающий за формирование ответа, также в нём расписаны все возможные коды ошибок.

Коды ошибок задаются в виде ассоциативного массива, где ключ — это код ошибки (число), а значение это текст ошибки.

Статический метод response в answer проверяет следующее:

1. Если переданные в него данные не false и это данные, которые содержат ассоциативный массив с ключом error, в противном случае возвращается успешный ответ
2. Если данные никакие не переданы, то ошибка 9000

Статические методы не имеют доступа к контексту класса.

В php вызов cстатических методов или обращение к статическому свойству осуществляется через ‘::’

По концепции MVC класс answer является представлением, и ещё частично файл index.php.

В index.php в функцию result передаётся аргументом суперглобальная переменная get.

Application.php

Класс, выполняющий функцию контроллера. В конструкторе этого класса объявляются экземпляры всех моделей: user и chat.

Также в конструкторе класса создается экземпляр для работы с базой данных и этот экземпляр передаётся в каждую модель. Передавать модели друг в друга запрещено!

В Application для каждого запроса создаётся свой метод. В application каждый метод как обработчик запроса проверяет и валидирует входящие параметры запроса, и в случае, если, они хорошие вызывает соответствующий метод соответствующей модели. Если модели для работы необходимы данные для работы, то эти данные получаем в application.

Метод sendMessage.

Этот метод отвечает за отправку сообщения в чат. Сначала проверяется наличие параметров запроса – параметров token и message. В случае если их нету, возвращается ошибка. Потом из модуля user мы получаем пользователя по токену.

В случае если такого пользователя нет, возвращается другая ошибка. Если пользователь есть, то вызывается метод чата sendMessage. Который передаёт идентификатор пользователя и его сообщение. В модели chat, в методе sendMessage сначала добавляется в базу данных новое сообщение, вторым методом обновляется hash сообщений.

Авторизация осуществляется с помощью метода login. В этом метод мы передаем Login, hash – может быть использована любая hash-функция, в частности md5(md5(логин+пароль)+rnd), rnd – случайное целое число.

В базе данных в поле пароль хранится хэш от логина + пароля. Пароль в базе данных в явном виде хранить – катастрофический моветон.

Процесс авторизации.

По логину из базы данных мы получаем пароль добавляя к нему параметр rnd мы берем от него hash и получаем контрольную сумму. Которая сравнивается с переданным хэшем с клиента. Если они равны то создаётся токен для этого пользователя, он записывается в базу данных и вместе с информацией по пользователю возвращается клиенту.

Logout

По токену берем пользователя из базы данных, если он нашелся – токен очищаем.

Сохранённые данные между запросами не сохраняются. Единственный способ сохранить данные между запросами – записать их в бд.

ДЗ: на бэкенде сделать новую модель ‘PainOfDomashka’ туда перенести методы derivative, касательная к производной,

Клиент должен поддерживать логин логаут и построений графиков клиент на React.

***14.10.24 Взаимодействие с бекендом из клиента.***

Клиент общается с бэкендом посредством аяксовых запросов. Аяксовые запросы посылаются с помощью стандартного JavaScript метода fetch. Все запросы собраны в сервисе Server. Рядом с сервером лежит файл Types. В нём описаны все типы всех ответов, которые могут прийти с бэкенда. В типах описан общий формат ответа вместе с дженериком, описаны все форматы ошибок.

Класс Server представляет из себя список методов каждый из которых отправляет запрос в бэкенд. При отправке запроса используется приватный метод request.

В противном случае он с помощью Callback возвращает ошибку.

Если это необходимо, полученные с бэкенда данные записываются в хранилище. Экземпляр хранилища передаётся в контекст сервера. В общем случае все запросы с клиента подписываются токеном, который клиент получает в случае успешной авторизации. В случае авторизации сервер выписывает клиенту токен, каждый раз новый.

Класс Store.

Класс, отвечающий за хранение всей информации в клиенте. В обязательном порядке для каждых новых данных должны быть созданы методы set и get. Set – устанавливает метод, get – позволяет данные получить.

Экземпляры классов Store и сервер прописываются в так называемый контекст. Контекст - специальная функция в React, которая позволяет обернуть какую-либо компоненту в какой-либо контекст. Все дочерние компоненты обернутой компоненты будут иметь доступы до этого контекста.

Плохая практика в контексте:

1. Менять контекст в процессе исполнения приложения.
2. Для различных компонент приложения использовать разные контексты.

Хорошая практика – использовать общий единый контекст, который оборачивает приложение.

В случае изменения состояния контекст перезаписывается. Переключение страниц в приложении осуществляется в компоненте pageManager.

Компонента Login.

Реализация чата.

На бэкенде хранятся сообщения чата. При добавлении каждого нового сообщения отдельно сохраняется новый хэш этих сообщений. Клиент постоянно посылает запросы в бэкенд на получение сообщений из чата. В каждом запросе прописывается текущий хэш чата, который знает клиент. Бэкенд сравнивает между собой хэши чатов, которые прислал клиент и хэши чатов, которые сохранены на сервере. Если они разные, то бэкенд отдает клиенту сообщение и сохраненный хэш.

Этот хэш клиент у себя перезаписывает и использует для дальнейших запросов. Если хэши одинаковые, бэкенд не отвечает ошибкой, но отвечает пустым ответом. Когда пользователь пишет сообщение в чат, посылается запрос, это сообщение записывается список сообщений и на бэкенде формируется новый хэш сообщений.

Можно ещё сильнее снизить нагрузку на бэкенд.

На бэкенде можно сделать кэш – сохранённые на бэкенде ответы, для которых выставляется определённый срок жизни. Уникальной подписью ответа является параметр запроса. Правильно настроенный кэш снижает нагрузку на бэкенд на 90-95%.

ДЗ: поменять верстку и сделать прототип.

28.10 Базы данных

Файловый способ хранения информации имеет два главных недостатка:

1. Неизвестная структура файлов
2. В каждый момент времени с файлом может взаимодействовать только один процесс

Для хранения информации были разработаны так называемые базы данных. Основная классификация: реляционные и не реляционные базы данных. Мы будем работать с реляционными базами данных. Большинство информации на планете хранится реляционных базах данных. В основе реляционных баз данных лежит математика множеств.

База данных – набор информации, которая хранится упорядоченно в электронном виде. В реляционных базах данных ***ВСЯ*** информация хранится таблицах. Таблица представляет собой набор записей, каждая из которых должна обладать уникальным ключом, для того чтобы можно было однозначно взять какую-нибудь запись из таблицы. Запись в таблице – это строка. Уникальный ключ может быть составным (состоять из нескольких столбцов (очень плохая практика)). Ключ, определяющий запись в таблице, называется первичным ключом. Хорошей практикой является отделение ключей от хранимой информации. Информация хранится в таблице в одних столбцах, а ключи хранятся в других, между собой не связанных. В первичный ключ писать число уникальное от каждой записи. Отношение — это зависимость записи/записей в одной таблице, от записи/записей в другой таблице. Отношение может быть создано в рамках одной таблицы. Отношения необходимы чтобы разрешить множественность хранимых данных. Отношениях бывают трех типов – 1/1, 1/многим, много/многим (плохой, необходимо разрешать (преобразовывать к предыдущим двум)). Отношение много/многим разрешается с помощью добавления новой таблицы к которой две исходные таблицы имеют отношение много/многим. Ссылка на первичный ключ в другой таблице называется внешним ключом. С помощью внешних ключей настраиваются отношения между таблицами. Во внешних ключах лежат значения первичных ключей из других таблиц.

Правила работы с базой данных:

1. Все названия в базе данных (полей и таблиц) должны быть написаны в латинской раскладке без использования специальных символов и только в нижнем регистре. Составные названия пишутся в снек-кейсе.
2. Кодировка базы данных – только UTF-8
3. Все таблицы в базе данных должны содержать первичный ключ – id
4. Любые внешние ключи должны называться следующим образом – таблица\_id
5. Первичные и внешние ключи – это числа
6. В таблице допустимы только связи 1/1 или 1/многим
7. Между таблицами может не быть связей
8. У каждого поля таблицы есть свой тип данных, который там лежит и его необходимо соблюдать
9. Для взаимодействия с базами данных существуют так называемые СУБД (системы управления) – процесс, который запускается на компьютере и обеспечивает взаимодействие с базами данных, в общем случае количество бд не ограничено.
10. В субд поддерживаются различные пользователи6 группы пользователей и их полномочия
11. У любой субд есть суперпользователь который может всё.
12. В субд на каждую базу данных создаётся свой отдельный пользователь который имеет право работать с любой бд.

Язык взаимодействия с бд – язык SQL. Язык SQL позволяет совершить все действия с бд :

1. Создание бд
2. Создание пользователя
3. Создание таблицы
4. Получение информации из таблиц
5. Добавление новой записи в таблицу
6. Удаление записи из таблицы
7. Изменение записи в таблице

Получение:

SELECT id, login, password FROM users; - вернет все записи из таблицы

SELECT \* - возьмутся все поля из таблицы

Результатом выполнения запроса SELECT всегда является таблица, даже если она не содержит ни одной записи. SELECT не меняет записи из исходной таблицы.

SELECT id, login, password FROM users WHERE login=’vasya’ (AND, OR, NOT) LIMIT=10 OFFSET=80 ORDER BY ‘name’(лимит по отданным записям и смещение на 80)

В таблице порядок записей не важен.

Изменение записей в таьлице:

UPDATE users SET token=’123’ WHERE id=’17’

Добавление записей:

INSERT users (login, password, name) VALUES (‘petya’, ’111’, ’Петя’);

DELETE FROM users WHERE id=’13’;

Все эти запросы могут быть выполнены относительно одной или нескольких таблиц.

Для работы с базой данных из языка программирования существуют драйвера которые осуществляют коннект в базу данных.

***25.11 Базы данных Лекция 2***

Оптимизация скорости работы с базами данных

При выборе данных из какой-либо не было таблицы субд перебирает все записи этой таблицы, сравнивая их с условием выбора. Либо происходит проверка между всеми парами записей в случае выбора из 2х таблиц.

Способы оптимизации работы с бд:

1. Индексация – одно или несколько полей в таблице помечаются как индексные и данные в этой таблице пересортировываются по заданному индексу. В случае выбора из таблиц у которой в условии присутствует индексное поле выбор осуществляется только по записям, имеющим этот индекс. Лучше всего чтобы индекс был численный. Индексные поля в таблице должны быть существенные. Хорошо сделанная индексация увеличивает скорость выборки из таблицы на 90%. Минусы:
   1. Индексация занимает место
   2. Может существенно увеличиться время записи таблиц
2. Горизонтальное дробление – неиспользуемые или редко используемые данные из таблицы переносятся в другую таблицу – хранилище. Грамотное разбиение увеличивает скорость работы на 90%. Минусы:
   1. Иногда существенным образом усложняется селект
   2. Необходимо настроить регулярный, но не постоянный перенос данных из одной таблицы в другую
3. Вертикальное разбиение – подразумевает перенос данных из таблицы зачастую с группировкой в другую таблицу. В случае вертикального разбиения зачастую нарушается целостность данных – появляется их дублирование, либо избыточность. Применять в последнюю очередь и в самом крайнем случае.

Не имеет смысла заранее оптимизировать таблицы.

Необходимо оптимизировать только те таблицы, которые быстро растут.

Имеет смысл заранее вводить индексацию только для особо быстро растущих таблиц.

Дополнительный функционал бд:

1. Транзакция – 1 или несколько запросов объединённые в транзакцию которые выполняются последовательно и для успешного выполнения транзакции обязательно выполнения всех запросов. В случае возникновения ошибки производится откат всех запросов. Используются в основном в финансовой сфере.
2. Хранимые процедуры – язык SQL является языком программирования и поддерживает все основные конструкции другого языка, в частности операции условного выбора циклы и процедуры. Хранимая процедура — это процедура, которую мы можем объявить в базе данных и использовать её в запросах. Хранимая процедура позволяет выполнить какую-то логику при исполнении запроса в скрытом от пользователя режиме. Категорически не рекомендуется:
   1. В хранимых процедурах реализовывать бизнес логику приложения

Хранимые процедуры можно использовать для логирования либо для какого-нибудь сложного выбора. Триггер – это обработанное кастомное событие, которое может быть вызвана по тому или иному запросу. Тригер может вызывать выполнение любых запросов или исполнение процедур. Триггерами надо пользоваться очень аккуратно, потому что в общем случае они прячут логику внутрь базы данных.

Не реляционные базы данных – базы данных, в которых данные хранятся не в таблицах, а в множествах – объектах. Понятия отношение – связь там не существует. Особенностью не реляционных бд является очень высокая скорость получения данных. Минус – увеличенное время записи.

***9.12 Нормальная форма и нормализация отношений***

Всего выделяется 6 нормальных форм, процесс проектирования любой бд производят от последовательного перехода отношений от одной нормальной формы к другой. Этот процесс итеративный.

Атрибут – это свойство некоторой сущности, ещё его называют полем таблиц.

Домен атрибута – множество допустимых значений, которое может принимать атрибут.

Кортеж – это конечное множество взаимосвязанных допустимых значений атрибута, которые вместе описывают некоторую сущность.

Отношение – это конечное множество кортежей(таблица).

Схема отношения – это конечное множество атрибутов, не определяющих некоторую сущность, структура таблицы, состоящая из конкретного набора полей.

Проекция – это отношение, полученное из заданного путём удаления или перестановки некоторых атрибутов.

Функциональная зависимость между атрибутами (множествами атрибутов x и y) означает что для любого допустимого набора кортежей в данном отношении, если два кортежа совпадают по значению x, то они совпадают по значению y.

Нормальная форма – это требование, предъявляемое к структуре таблиц, в теории реляционных баз данных для устранения из базы, **избыточных функциональных зависимостей между атрибутами.**

Цель нормализации – исключить избыточное дублирование данных, которое является причиной аномалии, возникающей при добавлении, редактировании и удалении кортежей.

Аномалия – это такая ситуация в таблице бд, которая приводит к противоречию в базе данных, либо существенно усложняет обработка базы данных.

Аномалии-модификации – проявляются в том, что изменение одних данных, может повлечь пересмотр всей таблицы с соответствующими изменениями.

Аномалии удаления – при удалении кортежа из отношения может пропасть информация, которая не связана напрямую с удаляемой записью.

Аномалии добавления – возникают, когда информацию невозможно занести в отношение пока он неполный, либо ставка кортежа требует дополнительного пересмотра всего отношения.

Отношение находится в первой нормальной форме если все его атрибуты являются простыми, все используемые домены должны содержать только скалярное значение, не должно быть повторений строк в таблице.

Используемый домены должны содержать только скалярное значение.

Вторая нормальная форма – отношение находится во второй нормальной форме, если оно находится в первой нормальной форме и каждый не ключевой атрибут не приводимо зависит от первичного ключа.

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

3-ая нормальная форма – если находится во второй и каждый неключевой атрибут не транзитивно зависит от первичного ключа. Это правило требует выносить все неключевые поля, содержимое которых может относится к нескольким записям в таблице в отдельной таблице.

Нормальная форма Войса-Кодда – частная форма третьей нормальной формы.

3-ая нормальная форма не подходит для отношений:

1. Если отношение имеет два или более потенциальных ключа.
2. Два или более потенциальных ключа составные.
3. Ключи пересекаются (имеют один или более общих атрибута)
4. Для отношений, имеющих один первичный ключ нормальная форма Войса-Кодда является третьей нормальной формой

Отношение находится в нормальной форма Войса-Кодда, когда каждое нетривиальное и неприводимое слева функциональная зависимость обладает потенциальным ключом в качестве детерминанта.

Отношение находится в 4-й нормальной форме если оно находится в нормальной форме Войса-Кодда и все нетривиальные многозначные зависимости фактически являются функциональными зависимостями от её потенциальных ключей.

Отношение находится в 5-й нормальной форме если оно находится в 5-й нормальной форме и отсутствуют сложные зависимые соединения между атрибутами.

Доменно-ключевая нормальная форма – перемена отношения находится в доменно-ключевой нормальной форме тогда\, когда каждое наложено е на неё ограничений является логическим следствием ограничений доменов и ограничений ключей.

Ограничение домена - ограничение предписывающее использовать для определённого атрибута значения только из некоторого заданного домена.

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

Переменное отношение находится в шестой нормальной форме тогда и только тогда\, когда она удовлетворяет всем нетривиальным зависимостям соединений, когда она не может быть подвергнута дальнейшей декомпозиции без потерь данных.