# **1 Билет**

### **1. Тестовые вопросы:**

1. Метод познания, который предполагает разделение целого на отдельные составные части и изучение каждой из этих частей, называется:  
    **г) анализ**
2. Краткосрочному периоду соответствует следующая формула издержек:  
    **а) TC = 180 + 6Q - 25Q²**  
    Пояснение: в краткосрочном периоде постоянные издержки (180) не равны нулю, а функция включает переменные с Q.
3. Разница между конечной стоимостью товара и стоимостью материалов, использованных для его производства — это:  
    **в) добавленная стоимость**
4. Если по мере роста дохода возрастает и величина налоговой ставки, то такая ставка считается:  
    **а) прогрессивной**
5. К источникам экстенсивного пути развития экономической системы относится:  
    **в) увеличение объема факторов производства**

### **2. Сопровождение реляционных баз данных. Технические приемы улучшения работы баз данных**

Сопровождение реляционных баз данных (РБД) — это процесс поддержания базы в рабочем состоянии: чтобы она не тормозила, не "падала" и всегда хранила только нужные и актуальные данные. Это важно, потому что в большинстве компаний все данные (о клиентах, товарах, заказах и т. д.) находятся в базах.

Вот основные действия, которые помогают улучшить работу базы:

1. **Резервное копирование (бэкапы)** — регулярное создание копий базы данных. Если что-то сломается — можно будет быстро всё восстановить.
2. **Следим за производительностью** — анализируем, какие запросы к базе самые медленные, и почему. Иногда программа "запрашивает" данные неэффективно — и из-за этого все тормозит.
3. **Использование индексов** — специальные структуры, которые ускоряют поиск данных. Без индексов база будет медленно обрабатывать даже простые запросы.
4. **Оптимизация запросов** — иногда SQL-запросы пишутся "криво", и это сильно замедляет работу. Поэтому их нужно упрощать и улучшать.
5. **Очистка старых данных** — удаление устаревших или неактуальных записей тоже помогает базе работать быстрее.
6. **Контроль за целостностью данных** — чтобы не было противоречий, дубликатов или логических ошибок.

Примеры инструментов: SQL Profiler, pgAdmin, MySQL Workbench. Они помогают находить "узкие места" и улучшать базу.

Если сопровождать базу грамотно, она будет работать быстро, не "падать", а данные в ней будут всегда в порядке.

### **3. Методы и средства проектирования информационных систем с использованием CASE-инструментов**

Проектирование информационных систем (ИС) — это этап, на котором мы "придумываем", как будет работать будущая программа или система: из чего она состоит, как обрабатывает данные, где хранит информацию и как выглядит интерфейс.

Чтобы этот процесс был не на бумажке и не "в голове", используют **CASE-инструменты** — это специальные программы, которые помогают рисовать схемы, описывать функции системы и даже создавать заготовки кода.

**Примеры таких инструментов:**

* **Enterprise Architect** — позволяет рисовать схемы, делать модели базы данных и описывать, как работает система.
* **Visual Paradigm**, **Erwin**, **Rational Rose** — похожие инструменты для разных задач.
* В простых случаях можно использовать **Draw.io** или **Lucidchart**, если просто нужно нарисовать схему.

**Какие методы проектирования бывают:**

1. **Объектно-ориентированный подход** — мы разбиваем систему на объекты (например, "Пользователь", "Принтер", "Отчет") и описываем, что каждый из них делает.
2. **Функциональный подход (структурный)** — описываем, какие функции выполняет система, и какие данные при этом обрабатываются.
3. **Моделирование бизнес-процессов (BPMN, IDEF0)** — помогает показать, как система вписывается в работу компании в целом.

CASE-инструменты помогают наглядно представить структуру и логику работы будущей информационной системы. Это упрощает командную работу и снижает количество ошибок при разработке.

# **2 Билет**

### **1. Тестовые вопросы:**

1. Позитивная экономика занимается:  
    **б) выявлением содержательной стороны экономических процессов и явлений**  
    Пояснение: позитивная экономика описывает, "что есть", а не "что должно быть".
2. Если валовой доход достигает максимума, то предельный доход при этом:  
    **а) равен нулю**  
    Когда валовой доход больше не растет — это значит, что прирост (предельный доход) уже ноль.
3. Потенциальный объем реального ВВП может возрастать при:  
    **в) внедрении в производство новой, более эффективной технологии**
4. Кривая Лаффера характеризует:  
    **г) связь между размером налоговой ставки и объемом налоговых поступлений в государственный бюджет**
5. Автором теории сравнительных преимуществ в международной торговле является:  
    **б) Д. Риккардо**

### **2. Языки баз данных**

Язык баз данных — это специальный набор команд и правил, которые позволяют нам "общаться" с базой данных: добавлять, изменять, удалять и извлекать нужную информацию.

**Основные виды языков в работе с базами данных:**

1. **DDL (Data Definition Language)** — язык определения данных:  
   С его помощью создаются таблицы, индексы, ограничения.  
   Примеры команд: CREATE, ALTER, DROP.
2. **DML (Data Manipulation Language)** — язык манипулирования данными:  
   Используется для добавления, удаления и изменения информации.  
   Команды: INSERT, UPDATE, DELETE, SELECT.
3. **DCL (Data Control Language)** — язык управления доступом:  
   Позволяет задавать права и ограничения для пользователей.  
   Команды: GRANT, REVOKE.
4. **TCL (Transaction Control Language)** — язык управления транзакциями:  
   Обеспечивает целостность данных при нескольких связанных изменениях.  
   Примеры: BEGIN, COMMIT, ROLLBACK.

**Примеры СУБД, использующих такие языки:**

* MySQL
* PostgreSQL
* Oracle
* MS SQL Server

**Основной язык, на котором пишут запросы — это SQL** (Structured Query Language). Он универсален и используется практически во всех современных СУБД.

Таким образом, знание языков баз данных позволяет управлять хранилищем данных, обеспечивать удобный и безопасный доступ к информации.

### **3. Основные принципы построения информационных систем**

Информационная система (ИС) — это не просто программа, а целая структура, которая помогает собирать, хранить, обрабатывать и передавать информацию. Принципы построения ИС — это правила, которых придерживаются, чтобы система была понятной, удобной и эффективно работала.

Вот ключевые принципы:

1. **Принцип системности**  
   Система — это не просто набор функций, а взаимосвязанная структура. Например, если пользователь вносит данные, они должны попасть в нужную таблицу, отобразиться в отчетах и быть доступными другим модулям.
2. **Принцип модульности**  
   Систему можно разделить на части (модули), каждый из которых выполняет свою задачу: учет клиентов, работа с заказами, генерация отчетов и т. д. Это облегчает разработку и поддержку.
3. **Принцип автоматизации**  
   Система должна выполнять рутинные операции без участия человека: подсчет, проверку, формирование отчетов и т. д.
4. **Принцип надежности и безопасности**  
   Все данные должны быть защищены от потерь, взлома и ошибок. Система должна иметь резервные копии, контроль доступа и проверку корректности данных.
5. **Принцип масштабируемости**  
   Система должна легко расширяться: добавлять новые модули, пользователей, функции — без полной переделки.
6. **Принцип простоты интерфейса**  
   Пользователь должен легко понимать, как работать с системой, без долгого обучения.

Пример: ИС для учета в школе должна позволять учителю быстро вносить оценки, администратору — формировать отчеты, а ученику — смотреть свой дневник. Всё это работает с одними данными, но через разные модули — вот пример хорошего применения этих принципов.

# **3 Билет**

### **1. Тестовые вопросы:**

1. К микроэкономическому утверждению **НЕ** относится:  
    **б) с 2000 г. в России наблюдается экономический рост**  
    Это утверждение относится к макроэкономике, так как описывает общую динамику в стране.
2. Закон убывающей предельной производительности состоит в том, что:  
    **в) убывает предельный продукт переменного ресурса при постоянных затратах прочих факторов**  
    То есть: если увеличивать один ресурс (например, рабочую силу), а остальные оставить как есть — отдача от каждого нового работника будет уменьшаться.
3. ВВП и ВНД отличаются на величину:  
    **г) сальдо факторных доходов**  
    Это разница между доходами от собственности и труда, полученными из-за рубежа и выплаченными за рубеж.
4. Фискальная политика может быть:  
    **г) все ответы верны**  
    Она может быть стимулирующей (повышает расходы), сдерживающей (снижает расходы), а также автоматической (встроенные стабилизаторы, как прогрессивные налоги).
5. Торговый баланс:  
    **а) является частью платежного баланса**  
    Он показывает соотношение экспорта и импорта товаров и входит в более общий документ — платежный баланс.

### **2. Назначение и стадии нормализации реляционных баз данных**

**Нормализация** — это процесс, при котором таблицы в базе данных упрощаются и структурируются так, чтобы:

* не было лишнего дублирования данных;
* все данные хранились логично и удобно;
* база легко обновлялась без ошибок.

**Зачем это нужно:**  
Если таблицы построены "криво", то:

* одни и те же данные могут повторяться (например, имя клиента в 100 строках);
* при удалении одной записи можно потерять кучу связанных данных;
* сложно делать запросы и анализ.

**Стадии нормализации — это уровни (формы), по которым выстраиваются таблицы:**

1. **Первая нормальная форма (1НФ):**  
    - В каждой ячейке должно быть только одно значение.  
    - Пример ошибки: в ячейке "Телефон" записаны два номера через запятую.  
    - Нужно разбить такие записи на отдельные строки.
2. **Вторая нормальная форма (2НФ):**  
    - Каждое неключевое поле должно зависеть от всего ключа, а не от части.  
    - Пример: если ключ состоит из двух полей, но "Адрес" зависит только от одного — это нарушает 2НФ.
3. **Третья нормальная форма (3НФ):**  
    - Поля не должны зависеть друг от друга, только от ключа.  
    - Пример: если у нас есть поле "Город", и рядом "Регион", и "Регион" можно узнать из "Города" — это избыточность.

Нормализация делает таблицы "чистыми", но иногда чересчур — и тогда, наоборот, **используют денормализацию**, чтобы ускорить работу (например, в отчетах).

### **3. Кооперация между параллельными задачами. Проблемы взаимного исключения и синхронизации задач**

Когда в одной системе работает **несколько процессов одновременно** (например, на сервере или в многозадачной программе), они должны как-то **взаимодействовать и не мешать друг другу**. Это и называется **кооперацией между задачами**.

#### Что нужно учитывать:

1. **Взаимное исключение (mutual exclusion):**  
    Это правило нужно, чтобы два процесса **не обращались одновременно к одному и тому же ресурсу** (например, к файлу или записи в базе).  
    Если они оба попробуют изменить один файл — может случиться ошибка или потеря данных.

**Как решается:**  
 - Использование **блокировок (lock)** — например, один процесс "блокирует" файл, пока работает с ним.  
 - Протоколы, вроде **mutex** или **семафоров** — это специальные механизмы, которые следят за тем, чтобы доступ был только у одного процесса за раз.

1. **Синхронизация:**  
    Иногда одна задача должна **дождаться**, пока другая закончит.  
    Например, задача 2 может стартовать **только после**, как задача 1 завершила загрузку данных.

**Решения:**  
 - Использование **событий (events)** и **сигналов**.  
 - Условные переменные (wait / notify).

1. **Проблемы, которые могут возникать:**

* **Голодание (starvation):** когда один процесс всё время не может получить доступ.
* **Взаимная блокировка (deadlock):** когда два процесса ждут друг друга — и оба "зависают".

#### Пример из жизни:

Представим, что два пользователя редактируют один документ в Google Docs. Если бы не было системы синхронизации, то их изменения постоянно бы затирали друг друга. Вот почему нужны механизмы кооперации.

# **4 Билет**

### **1. Тестовые вопросы:**

1. К основным чертам традиционной экономики можно отнести:  
    **г) многоукладность экономики, основанной на традициях**  
    Традиционная экономика — это экономика, основанная на обычаях, религии, передаче навыков по наследству.
2. Производственная функция показывает:  
    **в) максимальное количество продукта, которое можно получить, используя различные сочетания ресурсов**  
    Она отражает, как меняется выпуск продукции в зависимости от количества и сочетания используемых ресурсов.
3. Личный доход — это:  
    **б) доход, полученный домохозяйствами в течение данного года**  
    Включает заработную плату, проценты, дивиденды и другие доходы.
4. Государственный долг — это:  
    **г) задолженность правительства перед всеми хозяйствующими субъектами как данной страны, так и зарубежных стран**  
    Включает внутренний и внешний долг — и перед гражданами, и перед другими странами.
5. Благо, доступное одному, а пользоваться им могут одновременно многие — это:  
    **б) общественное благо**  
    Примеры: уличное освещение, общественная безопасность, чистый воздух.

### **2. Методы и средства физического проектирования реляционных баз данных**

**Физическое проектирование базы данных** — это этап, на котором определяют, **как именно данные будут храниться на диске**, чтобы система работала быстро и надежно. Если логическое проектирование отвечает на вопрос "что хранить", то физическое — "как это хранить".

#### Основные методы и действия:

1. **Выбор СУБД и хранилища:**  
   Например, PostgreSQL, MySQL, Oracle и др. — в зависимости от задач и бюджета. Также учитывается тип хранилища: SSD, HDD или облачное.
2. **Настройка индексов:**  
   Индексы — это как оглавление в книге. Они позволяют быстро найти нужную строку в таблице. Проектирование включает выбор, какие поля индексировать и какой тип индекса использовать (B-tree, hash и др.).
3. **Разделение данных (partitioning):**  
   Большие таблицы делят на части — по дате, по категориям и т. д. Это ускоряет запросы и облегчает обслуживание.
4. **Кластеризация данных:**  
   Данные физически хранятся в нужном порядке (например, по дате) — чтобы быстрее выполнять типовые запросы.
5. **Настройка буферов и кэшей:**  
   СУБД выделяет часть оперативной памяти под временное хранение данных — это ускоряет работу. Важно правильно настроить эти параметры под нагрузку.
6. **Оптимизация запросов:**  
   Подбираются структуры таблиц, чтобы минимизировать количество операций чтения/записи. Например, можно использовать материализованные представления или денормализацию в особых случаях.

#### Средства (инструменты):

* **SQL Profiler**, **pgAdmin**, **MySQL Workbench**, **Oracle SQL Developer**
* Мониторы нагрузки (Nagios, Zabbix)
* Планировщики задач (cron, job-сервисы в СУБД)

Физическое проектирование — это фундамент быстрой и устойчивой базы. Даже идеально построенная логическая модель не даст нужного результата без грамотной физической реализации.

### **3. Методология SADT**

**SADT (Structured Analysis and Design Technique)** — это методика структурного анализа и проектирования систем. Её основное предназначение — **визуально и логически описывать, как работает система**, из чего она состоит и как взаимодействуют её элементы.

#### Зачем нужна SADT:

* Чтобы "разложить" сложную систему на простые блоки.
* Чтобы все участники проекта (аналитики, разработчики, заказчики) **одинаково понимали** устройство системы.

#### Как это работает:

SADT использует **графическую нотацию IDEF0** — это схема из **функциональных блоков**, каждый из которых описывает, **что делает система или её часть**.

У каждого блока есть 4 стрелки:

1. **Вход (Input)** — что поступает в блок для обработки.
2. **Управление (Control)** — правила, ограничения, условия (например, закон, инструкция).
3. **Механизм (Mechanism)** — кто или что выполняет функцию (сотрудник, программа).
4. **Выход (Output)** — результат работы блока.

#### Пример:

Если мы моделируем систему продаж:

* Вход: заказ клиента
* Управление: политика компании
* Механизм: менеджер и CRM
* Выход: оформленный заказ

#### Преимущества:

* Простота понимания.
* Хорошо подходит для описания **бизнес-процессов**.
* Можно применять как для IT-проектов, так и для анализа работы организаций в целом.

**Вывод:**  
Методология SADT помогает системно и наглядно описывать сложные процессы. Это делает её популярной среди системных аналитиков, архитекторов и бизнес-инженеров.

# **5 Билет**

### **1. Тестовые вопросы:**

1. Укажите форму собственности для ситуации: государство является владельцем, а коллектив временно использует здание и оборудование за определённую плату:  
    **а) государственная**  
    Собственником остаётся государство, а коллектив — арендатор.
2. Неявные затраты — это:  
    **б) стоимость затраченных ресурсов, являющихся собственностью фирмы**  
    Это, например, когда владелец использует своё помещение под бизнес и не платит за аренду — но потенциальный доход от аренды учитывается как неявная затрата.
3. Установите, по какой формуле может быть рассчитан номинальный ВВП:  
    **в) ВВП ном = ВВП реальный \* Дефлятор**
4. Какая из перечисленных статей не относится к доходам государственного бюджета:  
    **в) трансфертные платежи населению**  
    Это наоборот — расходы бюджета, например, пенсии или пособия.
5. Кривая Лоренца за три года сдвинулась дальше от биссектрисы. Это означает:  
    **г) усиление дифференциации доходов населения**  
    Чем дальше кривая от линии равенства, тем больше неравенство.

### **2. Концепция ER-модели. Структурные ограничения, проблемы**

**ER-модель (Entity-Relationship)** — это способ описания структуры данных, который используется при проектировании баз данных. Она позволяет понять, **какие объекты (сущности)** есть в системе, **какими свойствами** они обладают и **как они связаны** между собой.

#### Основные понятия:

1. **Сущности (Entity)** — это "вещи", которые мы хотим хранить в базе.  
    Например: "Сотрудник", "Заказ", "Принтер".
2. **Атрибуты** — это характеристики сущности.  
    У "Сотрудника" это может быть ФИО, должность, ID и т.д.
3. **Связи (Relationships)** — определяют, как сущности связаны.  
    Пример: "Сотрудник оформляет Заказ" — связь "оформляет".

#### Структурные ограничения:

1. **Кратность (мощность связи):**  
    Определяет, сколько экземпляров одной сущности может быть связано с другой.  
    Например: "Один заказ — от одного клиента" (1:1) или "Один клиент — много заказов" (1:N).
2. **Обязательность участия (обязательные/необязательные связи):**  
    Например, у заказа должен быть клиент (обязательно), но не у каждого клиента должен быть заказ (необязательно).

#### Проблемы ER-моделей:

* **Сложность при большой модели:** если система большая, ER-диаграмма может стать перегруженной.
* **Не все ограничения можно выразить:** например, сложно напрямую показать бизнес-правила типа "в месяц можно максимум 3 заказа".
* **Нужна дальнейшая нормализация:** ER-модель не всегда отражает структуру в виде таблиц — её нужно дополнительно обрабатывать.

**Итог:**  
ER-модель помогает разработчикам и аналитикам визуализировать, какие данные будут в системе и как они связаны. Это шаг перед тем, как создавать таблицы в базе.

### **3. Технологии проектирования информационных систем**

Проектирование информационных систем (ИС) — это процесс, при котором продумывается **архитектура будущей системы**, включая её функции, данные и способы взаимодействия с пользователями.

#### Основные технологии проектирования:

1. **Моделирование процессов и данных:**
   * **UML-диаграммы** (например, use-case, диаграммы классов, активности) — описывают структуру и поведение системы.
   * **IDEF0**, **BPMN** — показывают бизнес-процессы и взаимосвязи между функциями.
   * **ER-модели** — для проектирования базы данных.
2. **CASE-инструменты (Computer-Aided Software Engineering):**  
   Специальные программы, которые помогают создавать схемы, описания и даже автоматически генерировать код.
   * Примеры: **Enterprise Architect**, **Visual Paradigm**, **Erwin**.
3. **Архитектурные подходы:**
   * **Модульная архитектура** — система разбивается на независимые блоки.
   * **Клиент-серверная модель** — разделение на серверную часть (где хранятся данные) и клиентскую (интерфейс пользователя).
   * **Микросервисная архитектура** — каждая функция работает как отдельное приложение.
4. **Проектирование интерфейсов (UI/UX):**
   * Проработка макетов экранов, логики действий пользователя, визуальной структуры.
   * Инструменты: Figma, Adobe XD, Axure.
5. **Документирование:**  
   Создание технической документации: ТЗ, спецификаций, инструкций для пользователей и разработчиков.

#### Зачем это всё нужно:

* Упрощает командную работу (все понимают структуру системы).
* Снижает риски ошибок и недопонимания.
* Помогает внедрять и сопровождать ИС быстрее и качественнее.

# **6 Билет**

### **1. Тестовые вопросы:**

1. Экономическая система, в которой управление и координация экономической деятельности осуществляется посредством централизованного планирования:  
    **в) административная экономика**  
    В такой системе государство определяет, что, как и для кого производить.
2. Кривая спроса для фирмы в условиях совершенной конкуренции совпадает с:  
    **а) кривой предельной и средней выручки**  
    Фирма не влияет на цену, поэтому выручка за каждую единицу товара одинакова.
3. Какие из перечисленных агрегатных величин **НЕ** используются при определении национального дохода (НД):  
    **б) государственные трансфертные платежи**  
    Это перераспределение доходов, а не результат производства.
4. К прямым налогам относится:  
    **а) налог на прибыль**  
    Прямые налоги взимаются с доходов или имущества конкретного лица.
5. К «провалам» рынка **НЕ** относят:  
    **в) производство частных благ**  
    Частные блага легко делимы и поддаются рыночному распределению.

### **2. Этапы проектирования баз данных, основанных на различных моделях данных**

Проектирование базы данных — это создание структуры, в которой будет удобно и безопасно хранить нужную информацию. В зависимости от модели (реляционной, иерархической, объектной и др.) этапы могут немного различаться, но общая логика остаётся одинаковой.

#### Основные этапы:

1. **Сбор и анализ требований:**
   * Определяют, какие данные нужно хранить, кто будет ими пользоваться и какие операции выполнять.
   * Пример: в базе данных для школы нужны данные об учениках, классах, оценках.
2. **Концептуальное проектирование (модель "что есть"):**
   * Создаётся **ER-модель** — диаграмма, где отображаются сущности (например, "Студент", "Курс") и связи между ними.
   * Модель не зависит от типа СУБД (системы управления базами данных).
3. **Логическое проектирование:**
   * Модель переводится в конкретную структуру таблиц.
   * Определяются ключи, типы данных, связи (например, один ко многим).
4. **Физическое проектирование:**
   * Настраиваются индексы, типы хранения, способ размещения данных на диске.
   * Важно для ускорения работы базы.
5. **Реализация в СУБД:**
   * Создаются таблицы, ограничения, триггеры, хранимые процедуры.
   * Пример: в PostgreSQL или MySQL прописываются команды CREATE TABLE.
6. **Тестирование и оптимизация:**
   * Проверка, корректно ли работает база.
   * Устраняются ошибки и "узкие места".

#### В зависимости от модели:

* В **реляционной** модели акцент на таблицы, связи, ключи.
* В **иерархической** — на структуру "родитель-дочерний" (как дерево).
* В **объектной** — на объединение данных и поведения (как в программировании).

**Итог:**  
Проектирование БД — это путь от идеи до работающей системы хранения информации. Правильная структура — основа для быстрой и безопасной работы.

### **3. Функционально-ориентированное проектирование ЭИС**

Функционально-ориентированное проектирование ЭИС (экономических информационных систем) — это подход, при котором система создаётся **вокруг задач и функций**, которые она должна выполнять. То есть сначала думают не о том, какие экраны и кнопки будут, а **какие именно действия должна совершать система**.

#### Основные принципы подхода:

1. **Выделение функций:**
   * Определяются основные действия, которые должна выполнять система. Например: "рассчитать зарплату", "выдать отчёт", "записать клиента".
2. **Построение DFD (Data Flow Diagrams):**
   * Рисуются диаграммы потоков данных: какие данные куда поступают, как обрабатываются, и куда отправляются дальше.
   * Это помогает понять, как вся система "перетекает" — от ввода до результата.
3. **Разделение системы на модули:**
   * Каждая функция становится отдельным модулем. Пример: модуль учёта товаров, модуль заказов, модуль отчётности.
   * Это упрощает реализацию и поддержку.
4. **Учет пользователей:**
   * Определяется, кто и как будет использовать каждую функцию: бухгалтер, менеджер, администратор и т. д.
5. **Описание входов/выходов:**
   * Для каждой функции описываются входные данные, что с ними делать, и какой результат должен получиться.

#### Преимущества подхода:

* Удобно для автоматизации конкретных процессов.
* Можно чётко связать бизнес-задачи и компоненты системы.
* Хорошо подходит для ИС, где много повторяющихся рутинных операций.

#### Пример:

В ЭИС для торговли одна из функций — "оформление заказа". В функциональном проектировании сначала расписываются шаги: ввод данных → проверка наличия → резервирование → печать документа. Только потом проектируются кнопки, формы и база под это.

# **7 Билет**

### **1. Тестовые вопросы:**

1. Сдвиг кривой производственных возможностей вправо означает:  
    **а) увеличение ресурсов**  
    Это показывает, что страна может производить больше благодаря новым технологиям или росту трудовых/природных ресурсов.
2. На рынке совершенной конкуренции отдельный продавец:  
    **а) не влияет на цену**  
    Он принимает рыночную цену как заданную, потому что конкуренция очень высокая.
3. Трансфертные платежи — это:  
    **а) выплаты домашним хозяйствам, не обусловленные предоставлением с их стороны товаров и услуг**  
    Примеры: пенсии, пособия — люди получают деньги, не производя товаров или услуг.
4. Отрицательный наклон кривой совокупного спроса (AD) связывают с:  
    **в) эффектом богатства**  
    При снижении цен люди чувствуют себя «богаче», потому что их сбережения покупают больше.
5. Предложение земли как фактора производства является:  
    **а) абсолютно неэластичным**  
    Площадь земли не изменяется от цены — её количество фиксировано.

### **2. Назначение и стадии нормализации реляционных баз данных**

**Нормализация** — это процесс упорядочивания структуры таблиц в реляционной базе данных, чтобы **избежать избыточности и ошибок**, а также сделать систему хранения данных более понятной и эффективной.

#### Зачем нужна нормализация:

* Убирает дублирование данных.
* Упрощает обновление информации.
* Обеспечивает логичность структуры базы.

#### Основные стадии нормализации:

1. **Первая нормальная форма (1НФ):**
   * Каждая ячейка таблицы содержит **только одно значение**.
   * Пример: нельзя хранить "Москва, Санкт-Петербург" в одной ячейке — надо разделить.
2. **Вторая нормальная форма (2НФ):**
   * Устранение частичной зависимости от составного ключа.
   * Каждое неключевое поле зависит от **всего** ключа, а не от части.
3. **Третья нормальная форма (3НФ):**
   * Нет транзитивной зависимости: поля не зависят друг от друга, только от ключа.
   * Пример: если есть "Город" и "Регион", и регион зависит от города — нужно вынести в отдельную таблицу.

#### Дополнительно:

* **Нормализация помогает сделать базу гибкой и удобной в обслуживании.**
* Иногда, наоборот, применяют **денормализацию** — чтобы ускорить выборку в больших системах отчётности.

### **3. Методология RAD (Rapid Application Development)**

**RAD (быстрая разработка приложений)** — это методика, при которой **информационные системы разрабатываются быстро, с активным участием заказчика и короткими итерациями**. Цель — получить рабочий продукт в кратчайшие сроки, с возможностью его доработки в процессе.

#### Ключевые особенности RAD:

1. **Быстрая разработка и запуск:**
   * Вместо длительного планирования — разработка MVP (минимально жизнеспособной версии) и её тестирование.
2. **Итеративный подход:**
   * Проект делится на короткие циклы. После каждой итерации клиент оценивает результат и вносит корректировки.
3. **Модульность:**
   * Приложение создается как набор отдельных компонентов, которые можно менять и улучшать независимо.
4. **Активное участие пользователя:**
   * Клиент тесно работает с командой и быстро предоставляет обратную связь.
5. **Прототипирование:**
   * Делается не абстрактная схема, а реальный прототип — пусть и не финальный, но работающий.

#### Преимущества RAD:

* Быстрый результат.
* Минимизация недопонимания между заказчиком и разработчиком.
* Гибкость — можно менять требования "на ходу".

#### Недостатки:

* Не всегда подходит для крупных систем.
* Требует высокой вовлеченности клиента.
* Не рекомендуется, если проект должен быть строго задокументирован.

**Итог:**  
RAD — отличный выбор, когда важна **скорость**, а требования могут **меняться в процессе**. Особенно часто применяется при разработке интерфейсов, веб-приложений и CRM-систем.

# **8 Билет**

### **1. Тестовые вопросы:**

1. Кривая производственных возможностей **НЕ** иллюстрирует:  
    **в) преимущество производства одного блага вместо другого**  
    Кривая показывает выбор, эффективность, редкость ресурсов, но не «преимущество».
2. Рынок совершенной конкуренции более эффективен, потому что на нём:  
    **г) больший объем производства и ниже цена товара**  
    Из-за высокой конкуренции и отсутствия влияния на цену.
3. Какое из перечисленных явлений **НЕ** соответствует периоду экономического спада:  
    **г) уменьшение объема пособий по безработице**  
    Наоборот, при спаде пособия чаще увеличиваются из-за роста безработицы.
4. Какие факторы сдвигают кривую совокупного спроса (AD) в сторону увеличения:  
    **в) увеличение чистого экспорта**
5. Спрос на рынке труда зависит от:  
    **б) спроса на готовую продукцию, для производства которой он используется**  
    Если продукция нужна, нужно больше работников — спрос на труд растёт.

### **2. Функции и компоненты СУБД (Систем управления базами данных)**

**СУБД (система управления базами данных)** — это программа, которая позволяет создавать, управлять и использовать базы данных. Она обеспечивает взаимодействие между пользователем, приложением и самой базой.

#### Основные **функции СУБД**:

1. **Хранение и организация данных**  
    — данные структурируются в таблицы, связи и индексы.
2. **Манипулирование данными**  
    — возможность добавлять, удалять, изменять и извлекать информацию (через SQL).
3. **Обеспечение целостности данных**  
    — правила, которые не позволяют вносить противоречивую или некорректную информацию.
4. **Контроль доступа и безопасности**  
    — кто и что может делать с данными (например, только админ может удалять записи).
5. **Управление транзакциями**  
    — обеспечение корректной работы нескольких операций (например, если что-то пошло не так — можно отменить).
6. **Резервное копирование и восстановление**  
    — сохранение данных на случай сбоев или потерь.

#### Основные **компоненты СУБД**:

1. **Ядро СУБД** — основа, которая обрабатывает запросы и управляет данными.
2. **Язык запросов (обычно SQL)** — позволяет пользователю взаимодействовать с базой.
3. **Хранилище данных** — физическое место, где лежат таблицы.
4. **Механизмы безопасности** — пароли, роли, разрешения.
5. **Журнал транзакций** — запись всех операций для отката или восстановления.

**Примеры СУБД:**  
MySQL, PostgreSQL, Oracle, MS SQL Server, SQLite.

### **3. Средства и методологии проектирования информационных систем**

Проектирование ИС — это этап, на котором продумывается, **как будет устроена и как будет работать будущая информационная система**. Это важно, чтобы система выполняла нужные функции, была удобной и понятной.

#### Методологии (подходы к проектированию):

1. **Классический водопадный подход (Waterfall):**
   * Проект делится на чёткие этапы: анализ → проектирование → разработка → тестирование.
   * Каждый этап выполняется строго по порядку.
2. **Итерационный и гибкий подход (Agile, Scrum):**
   * Разработка идёт по коротким циклам (итерациям).
   * Возможны изменения и улучшения в процессе.
   * Пользователь активно участвует в разработке.
3. **RAD (Rapid Application Development):**
   * Быстрое создание рабочих прототипов.
   * Пригоден для проектов, где сроки важнее полного описания.
4. **SADT и IDEF0 (структурное проектирование):**
   * Используются для описания функций и потоков данных в системе.
5. **Объектно-ориентированное проектирование (UML):**
   * Система описывается в виде объектов и их взаимодействий (классы, диаграммы).

#### Средства проектирования:

* **CASE-средства** (Enterprise Architect, Visual Paradigm, Erwin)  
   — помогают рисовать схемы, модели, генерировать код, писать документацию.
* **Инструменты моделирования интерфейсов** (Figma, Axure)  
   — создают макеты и прототипы интерфейсов.
* **Среды разработки (IDE)** (например, Visual Studio, IntelliJ, PyCharm)  
   — для создания и тестирования приложений.

#### Зачем это всё нужно:

* Упрощает взаимодействие в команде.
* Позволяет заранее увидеть слабые места системы.
* Ускоряет внедрение и снижает риск ошибок.

**Итог:**  
Правильное проектирование — залог надёжной, удобной и масштабируемой информационной системы.

# **9 Билет**

### **1. Тестовые вопросы:**

1. Если цена билета — 200 руб, матч длится 3 часа, а зарплата — 700 руб/час, то альтернативные издержки:  
    **г) 900**  
    Альтернативные издержки — это потерянный заработок за 3 часа: 700 × 3 = 2100 руб. Но так как билет стоит 200, а не только время, правильный ответ — ***2100***. Поэтому:  
    **Правильный ответ: а) 2100**
2. В условиях совершенной конкуренции в долгосрочном периоде соблюдается равенство:  
    **б) MR = MC = AC = P**  
    В долгосрочном равновесии фирма не получает сверхприбыли, все выручки и издержки уравновешены.
3. Для депрессии **НЕ** характерны:  
    **б) затоваривание и падение цен**  
    Это характерно скорее для ***кризиса***, а депрессия — это уже затяжная фаза, где уровень цен может стабилизироваться или даже начать расти.
4. Классический вариант кривой AS предполагает, что в долгосрочном периоде изменение совокупного спроса:  
    **б) окажет воздействие на уровень цен, но не на объем выпуска**  
    В классической модели ВВП определяется факторами предложения, а спрос влияет только на цены.
5. На рынке совершенной конкуренции отдельный продавец:  
    **а) не влияет на цену**  
    Он "принимает" рыночную цену как данность.

### **2. Назначение реляционной алгебры и реляционного исчисления. Начальная реляционная алгебра Э. Кодда**

Реляционная алгебра и реляционное исчисление — это **формальные языки работы с базами данных**, разработанные Эдгаром Коддом, чтобы строго описывать **запросы к реляционным базам данных**.

#### Зачем нужны:

* Обеспечивают **теоретическую основу** для работы SQL и других языков запросов.
* Позволяют **строго и формально** описывать выборку, фильтрацию и объединение данных.
* Используются при оптимизации запросов и построении компиляторов СУБД.

### **Реляционная алгебра**

Это **набор операций над таблицами** (отношениями), результатом которых тоже является таблица.

#### Основные операции:

1. **Выборка (σ)** — фильтрация строк по условию.  
    Пример: выбрать всех сотрудников с зарплатой > 50000.
2. **Проекция (π)** — выбор определённых столбцов.  
    Например: только имена и должности из таблицы сотрудников.
3. **Объединение (∪)** — объединяет строки из двух таблиц с одинаковой структурой.
4. **Пересечение (∩)** — только те строки, которые есть в обеих таблицах.
5. **Разность (-)** — строки, которые есть в одной таблице, но нет в другой.
6. **Декартово произведение (×)** — соединяет все строки двух таблиц попарно.
7. **Сочетание (Join)** — объединяет таблицы по какому-либо условию.

### **Реляционное исчисление**

В отличие от алгебры, здесь **описывается не как получить результат, а каким он должен быть**. Это декларативный подход.

Есть два вида:

* **Кортежное исчисление** — работает с кортежами (строками).
* **Доменное исчисление** — работает с отдельными значениями (ячейками).

Пример:  
«Найти такие X, где X — имя студента, и студент учится на 4 курсе».  
(Это как логическое высказывание, а не SQL-запрос.)

### **Зачем это важно:**

* Эти формализмы легли в основу **SQL**.
* Помогают понять, как работают СУБД "внутри".
* Используются в теоретических дисциплинах, оптимизации запросов, построении планов выполнения.

### **3. CASE-метод Баркера**

**CASE-метод Баркера** — это метод структурного анализа информационных систем, разработанный **Ричардом Баркером**, и применяемый в рамках CASE-средств (Computer-Aided Software Engineering).

Он входит в семейство методологий **SSADM** (Structured Systems Analysis and Design Method) — стандарт Великобритании для проектирования ИС.

### Основные идеи метода:

1. **Разделение проектирования на три модели:**

* **Модель данных** — показывает, какие данные хранит система (через ER-диаграммы).
* **Функциональная модель** — показывает, что делает система (через DFD — диаграммы потоков данных).
* **Модель поведения (событийная модель)** — как система реагирует на внешние события.

1. **Использование ER-диаграмм:**
   * Поддержка нормализации, ключей, атрибутов, связей.
   * Сильная проработка понятий **обязательности**, **мощности связей** и **правил целостности**.
2. **Формализация требований:**
   * Чёткая документация требований заказчика, пошаговая декомпозиция функций.
   * Акцент на **гибкость** и **повторное использование** компонентов.
3. **Интеграция с CASE-инструментами:**
   * Метод Баркера реализован в инструментах вроде Oracle Designer, Visible Analyst и др.
   * Позволяет автоматизировать часть проектных работ и даже генерировать SQL-код.

### Пример:

При проектировании системы учета студентов можно с помощью метода Баркера:

* Построить ER-диаграмму с сущностями "Студент", "Курс", "Успеваемость".
* Нарисовать диаграмму потоков: "Добавить оценку", "Сформировать отчёт".
* Определить события: "Начало семестра", "Закрытие сессии".

**Итог:**  
CASE-метод Баркера — это мощный и формальный подход к проектированию ИС, который помогает грамотно описывать структуру данных и бизнес-логику системы ещё до её реализации.

# **10 Билет**

### **1. Тестовые вопросы:**

1. Из кривой производственных возможностей можно сделать вывод о существовании экономического закона:  
    **г) возрастания альтернативных затрат**  
    Чтобы произвести больше одного блага, приходится жертвовать всё большим количеством другого.
2. В условиях монополии:  
    **б) предельная выручка меньше цены**  
    Чтобы продать больше, монополист должен снизить цену, из-за чего MR < P.
3. В классическом представлении промышленный экономический цикл состоит из:  
    **б) четырех**  
    Фазы: подъем, пик, спад (кризис), депрессия.
4. К основным положениям классической модели макроэкономического равновесия **НЕ** относят:  
    **б) сбережения определяются уровнем располагаемого дохода**  
    Это характерно для кейнсианской модели, а классическая — за приоритет процентной ставки.
5. Неявные затраты — это:  
    **б) стоимость затраченных ресурсов, являющихся собственностью фирмы**  
    То, за что не платят явно, но что имеет экономическую стоимость (например, собственное здание фирмы).

### **2. Анализ архитектур многозвенных приложений для работы с базами данных**

**Многозвенные приложения (multi-tier architecture)** — это архитектура программ, в которой функциональные компоненты разделены по "слоям" (звеньям). Особенно часто используется в информационных системах с базами данных.

### Основные архитектуры:

#### 1. **Двухзвенная (2-tier) архитектура:**

* Пример: клиент → СУБД.
* Приложение (например, на ПК) напрямую обращается к базе данных.
* **Плюсы:** простота, легко реализовать.
* **Минусы:** плохая масштабируемость, сложный контроль доступа, нагрузка на БД.

#### 2. **Трёхзвенная (3-tier) архитектура:**

* **Клиент (интерфейс)** – отображает данные.
* **Сервер приложений (логика)** – обрабатывает запросы.
* **Сервер базы данных** – хранит информацию.

**Плюсы:**

* Повышенная безопасность: клиент не имеет прямого доступа к БД.
* Легко обновлять логику без переустановки у пользователей.
* Масштабируемость.

**Минусы:**

* Усложнение инфраструктуры.
* Требует отдельного сервера приложений.

#### 3. **Многозвенная (n-tier) архитектура:**

* Больше трёх слоёв, добавляются слои API, кэширования, балансировки и т. д.
* Используется в крупных корпоративных системах.

### Применение в ИС:

* Трёхзвенная модель часто используется в **веб-приложениях**: фронтенд (браузер), бэкенд (сервер логики) и БД (PostgreSQL, Oracle и т.д.).
* Например: сайт интернет-магазина → обращение к серверу → обработка корзины → работа с базой.

**Вывод:**  
Многозвенные архитектуры делают систему **безопасной, гибкой и удобной для масштабирования**, особенно в случае большого числа пользователей.

### **3. Автоматизированные средства проектирования ИС**

Автоматизированные средства проектирования ИС — это **программы и платформы**, которые помогают аналитикам и разработчикам **проектировать, моделировать и документировать** будущую систему без необходимости вручную рисовать схемы и писать всё "с нуля".

### Основные возможности таких средств:

1. **Моделирование данных и процессов:**
   * Построение **ER-диаграмм**, **DFD**, **UML-диаграмм**.
   * Например: определить, какие таблицы нужны для системы учета заказов, как они связаны.
2. **Генерация документации и кода:**
   * Автоматическое создание ТЗ, SQL-скриптов, описания структуры системы.
3. **Анализ связей и целостности:**
   * Помогает избежать ошибок на этапе проектирования.
4. **Совместная работа над проектом:**
   * Команда может работать над схемой в одном проекте, отслеживать изменения.

### Популярные средства:

* **Enterprise Architect** – мощная система с поддержкой UML, BPMN, баз данных.
* **Visual Paradigm** – удобен для командной работы, генерация кода.
* **Erwin Data Modeler** – специализируется на моделировании баз данных.
* **Oracle Designer**, **IBM Rational Rose** – классические CASE-средства.
* **Draw.io, Lucidchart** – онлайн-редакторы для простых схем.

### Преимущества:

* Ускорение проектирования.
* Повышение точности и качества систем.
* Упрощение сопровождения и масштабирования.

**Вывод:**  
Использование автоматизированных средств позволяет **повысить эффективность разработки ИС**, минимизировать ошибки и стандартизировать проектную документацию.

# **11 Билет**

### **1. Тестовые вопросы:**

1. Излишек потребителя – это:  
    **б) разница между максимальной ценой, которую потребитель готов заплатить за единицу товара, и его рыночной ценой**  
    Это выгода, которую получает покупатель, когда он платит меньше, чем готов был.
2. Фирма действует на рынке \_\_\_\_\_\_\_\_\_\_\_\_, если производство в отрасли распределено между несколькими фирмами, продукт однородный и/или дифференцированный:  
    **г) олигополии**  
    Олигополия — это несколько крупных игроков, например, рынок авиаперевозок.
3. Необходимость обновления основных капитальных благ, требующих длительное время и огромных затрат — причина:  
    **в) длинных волн конъюнктуры**  
    Это 40–60-летние циклы (волны Кондратьева), связанные с технологическими изменениями.
4. Если располагаемый доход вырос с 600 тыс. до 650 тыс., а потребление увеличилось на 40 тыс., то предельная склонность к сбережению:  
    **в) 0,2**  
    ΔСбережения = 50 тыс. – 40 тыс. = 10 тыс.; ΔДохода = 50 тыс. → 10 / 50 = 0,2
5. Закон убывающей предельной производительности состоит в том, что:  
    **в) убывает предельный продукт переменного ресурса при постоянных затратах прочих факторов**  
    То есть: если добавлять работников, а оборудование оставить тем же — отдача от каждого нового снижается.

### **2. Концепция ER-модели. Структурные ограничения, проблемы**

**ER-модель (Entity-Relationship)** — это способ представить, **какие данные будут храниться в базе**, и **как они связаны между собой**. Она активно применяется на стадии концептуального проектирования баз данных.

### Основные компоненты ER-модели:

1. **Сущности (Entity)** — объекты, о которых мы храним данные.  
    Например: «Студент», «Курс», «Преподаватель».
2. **Атрибуты** — характеристики сущностей.  
    Пример: у «Студента» — имя, номер зачётки, дата рождения.
3. **Связи (Relationships)** — отображают отношения между сущностями.  
    Например: «Студент записан на Курс».

### Структурные ограничения:

1. **Мощность связи (кардинальность):**  
    - Один к одному (1:1)  
    - Один ко многим (1:N)  
    - Многие ко многим (M:N)
2. **Обязательность участия:**  
    Показывает, обязательна ли связь для сущности.  
    Пример: у студента **обязательно** должен быть хотя бы один курс.
3. **Идентифицирующие связи:**  
    Когда связь влияет на ключ сущности (например, связь "зависит от").

### Проблемы и ограничения ER-моделей:

* **Сложность на больших проектах** — диаграммы перегружаются.
* **Не всегда очевидно, как реализовать в таблицах** — требует дополнительного преобразования в реляционную модель.
* **Ограниченная выразительность** — сложно описывать бизнес-правила (например, "максимум 5 студентов на курсе").

**Итог:**  
ER-модель — важный инструмент при проектировании БД, позволяющий **понятно и визуально представить структуру данных**, но требует грамотного применения и перехода к физической реализации.

### **3. Моделирование данных**

**Моделирование данных** — это процесс, в котором мы **структурируем информацию**, чтобы удобно и эффективно хранить её в базе данных. Цель — понять, **какие данные**, **в каком виде** и **с какими связями** должны быть в системе.

### Основные этапы:

1. **Концептуальное моделирование (что хранить):**  
    - Используется **ER-модель** — описываются сущности, атрибуты и связи.  
    - Независимо от конкретной СУБД.
2. **Логическое моделирование (как представить в БД):**  
    - Преобразование в таблицы, определение ключей, типов данных, связей.  
    - Пример: сущность "Клиент" → таблица с полями: ID, Имя, Телефон.
3. **Физическое моделирование (как реализовать на диске):**  
    - Определяются индексы, типы хранения, способы разбиения таблиц (partitioning).

### Виды моделей:

* **Иерархическая** — древовидная структура (редко используется).
* **Сетевая** — сложные связи, применяется в специализированных системах.
* **Реляционная** — основана на таблицах, самая популярная модель.
* **Объектно-ориентированная** — данные хранятся в виде объектов (подходит для программирования).

### Почему моделирование важно:

* Уменьшает ошибки при проектировании.
* Делает структуру данных понятной для всех участников проекта.
* Упрощает последующую разработку и поддержку.

**Итог:**  
Моделирование данных — фундамент баз данных и информационных систем. Это процесс, который определяет **структуру**, **взаимосвязь** и **логическую целостность** всей информации в системе.

# **12 Билет**

### **1. Тестовые вопросы:**

1. Товар считается нормальным, если:  
    **а) потребитель с увеличением дохода увеличивает объем его потребления**  
    Нормальные товары — это такие, которые покупают больше при росте дохода (например, одежда, техника).
2. Конкуренция, основанная на сервисе и дополнительных услугах, называется:  
    **г) неценовой**  
    Пример: бесплатная доставка, бонусы, гарантия — всё, что не связано напрямую с ценой.
3. Открытая инфляция характеризуется:  
    **а) постоянным повышением уровня цен**  
    Цены растут и это явно видно на рынке — в отличие от скрытой инфляции, когда дефицит и цены официально стабильны.
4. Средняя склонность к потреблению — это:  
    **а) размеры потребления к величине дохода**  
    Показывает, какую часть дохода люди в среднем тратят.
5. Краткосрочному периоду соответствует формула издержек:  
    **а) TC = 180 + 6Q - 25Q²**  
    Присутствует постоянный элемент (180) — это фиксированные издержки, характерные для краткосрочного периода.

### **2. Предпосылки появления баз данных. История развития технологий хранения и обработки информации**

Базы данных появились как ответ на **необходимость упорядоченного хранения и быстрого доступа к большим объёмам информации**.

#### До появления БД:

* Данные хранились в бумажных архивах, журналах, карточках — это было неудобно, медленно и ненадёжно.
* Затем появились **файловые системы**: информация записывалась в текстовые или бинарные файлы на компьютерах.

### Основные проблемы файлового подхода:

* Дублирование данных.
* Трудности в обновлении (при изменении одной записи нужно менять все дубликаты).
* Нет стандартного языка запросов.
* Сложность в обеспечении целостности.

### История развития баз данных:

1. **1960-е — Иерархические и сетевые БД:**
   * Примеры: **IMS (IBM)**.
   * Данные связаны жёстко: как дерево (иерархия) или сеть.
   * Быстрые, но негибкие.
2. **1970-е — Реляционная модель Эдгара Кодда (IBM):**
   * Прорыв: данные представлены в виде таблиц.
   * Отношения между таблицами, строгая структура, SQL.
   * Пример: Oracle, PostgreSQL, MS SQL Server.
3. **1980–90-е — Объектно-ориентированные и клиент-серверные СУБД:**
   * Возможность хранить более сложные данные (фото, документы).
   * Появление первых **графических интерфейсов**.
4. **2000-е — Веб-приложения, масштабирование:**
   * Появились **NoSQL**-базы (MongoDB, Cassandra) для хранения неструктурированных данных.
5. **Современный этап:**
   * Облачные решения, автоматическое масштабирование, гибридные СУБД.
   * Примеры: Firebase, Amazon RDS, Google BigQuery.

**Итог:**  
Появление баз данных — это эволюционный шаг от неудобных бумажных или файловых архивов к гибким, надёжным и мощным системам, которые лежат в основе большинства современных приложений и сервисов.

### **3. Методология IDEF1X**

**IDEF1X** (Integration Definition for Information Modeling) — это **методология моделирования данных**, разработанная для формального и стандартизированного описания **структуры информационной системы**.

Она особенно полезна при проектировании **реляционных баз данных**.

### Основные особенности IDEF1X:

1. **Ориентирована на реализацию:**
   * Модели, построенные в IDEF1X, могут быть **непосредственно преобразованы в схемы таблиц** БД.
2. **Поддержка ключей и ссылочной целостности:**
   * Чётко указывается, какие поля являются первичными ключами, какие — внешними.
   * Пример: в таблице «Заказы» внешний ключ ссылается на таблицу «Клиенты».
3. **Разделение сущностей на:**
   * **Идентифицируемые** — те, у которых есть собственный первичный ключ.
   * **Зависимые (внедрённые)** — не могут существовать без родительской сущности.
4. **Обозначение связей с точной кратностью и обязательностью:**
   * Например: «Один заказ может содержать много товаров» (1:N).

### Сравнение с ER-моделью:

* ER-диаграммы более **обобщённые и визуальные**, ориентированы на концептуальное проектирование.
* IDEF1X — **более формальный и строгий подход**, заточенный под **реализацию и работу с БД**.

### Пример использования:

При проектировании информационной системы для банка можно с помощью IDEF1X построить модель:

* Сущности: Клиент, Счёт, Транзакция.
* Связи: Один клиент может иметь много счетов; каждый счёт связан с транзакциями.

**Итог:**  
Методология IDEF1X — это инструмент точного, чёткого моделирования структур данных. Она широко применяется в инженерных и корпоративных проектах, где важна безошибочная структура базы и её автоматическая генерация.

# **13 Билет**

### **1. Тестовые вопросы:**

1. Реализацию интересов рыночных субъектов и соединение интересов продавца и покупателя обеспечивает:  
    **б) посредническая функция рынка**  
    Рынок выступает как посредник между продавцом и покупателем, обеспечивая их взаимодействие.
2. К общим чертам рынков монополистической и совершенной конкуренции относятся:  
    **в) множество продавцов и покупателей, оперирующих на рынке**  
    В обоих типах рынков есть большое количество участников, но различие — в однородности или разнообразии товара.
3. Если в январе цена на хлеб выросла на 3,5%, а на молоко на 4,1%, то это:  
    **г) несбалансированная инфляция**  
    Цены растут неравномерно по товарам — это и есть признак несбалансированной инфляции.
4. Парадокс бережливости означает, что увеличение сбережений приводит к:  
    **в) уменьшению дохода**  
    Если все начинают больше сберегать, меньше тратят — падает спрос, производство и доход.
5. От потребительских предпочтений зависит:  
    **б) положение и наклон кривых безразличия**  
    Кривые безразличия показывают, какие наборы товаров одинаково желанны для потребителя.

### **2. Сравнительная характеристика поколений СУБД**

СУБД (системы управления базами данных) развивались в несколько **поколений**, каждое из которых отражает изменение подходов к хранению, обработке и доступу к данным.

### **1-е поколение: Файловые системы (1950–1960 гг.)**

* Данные хранились в виде простых файлов (текстовых или бинарных).
* **Нет структуры**, нет связи между данными.
* **Минусы**: дублирование информации, трудности с поиском и обновлением.

### **2-е поколение: Иерархические и сетевые СУБД (1960–1970 гг.)**

* Примеры: **IMS (IBM)**, **CODASYL**.
* Иерархическая модель: данные организованы как дерево.
* Сетевая модель: данные связаны множественными связями.
* **Плюсы**: быстрее файлов, связь между записями.
* **Минусы**: сложность в сопровождении, жёсткая структура.

### **3-е поколение: Реляционные СУБД (1970–настоящее время)**

* Основаны на **табличной структуре** (Эдгар Кодд, 1970 г.).
* Примеры: **MySQL, PostgreSQL, Oracle, MS SQL Server**.
* Используется **SQL** — универсальный язык запросов.
* **Плюсы**: гибкость, масштабируемость, безопасность.
* **Минусы**: не всегда удобно хранить сложные данные (изображения, JSON и т.д.)

### **4-е поколение: Объектно-ориентированные СУБД**

* Интеграция с объектно-ориентированным программированием.
* Примеры: **db4o**, **ObjectDB**.
* **Плюсы**: хорошо подходят для сложных объектов.
* **Минусы**: ниже производительность, меньшая поддержка.

### **5-е поколение: NoSQL и NewSQL (2000-е — сегодня)**

* Предназначены для **больших объемов неструктурированных данных**.
* Примеры: **MongoDB, Cassandra, Redis, Neo4j**.
* Поддерживают масштабирование, гибкость в структуре.
* **NewSQL** — новые решения, сочетающие SQL + масштабируемость.

**Вывод:**  
Каждое поколение СУБД решает задачи своего времени: от простого хранения — к гибкому управлению сложной информацией. Современные системы сочетают элементы нескольких поколений.

### **3. Классификация методов проектирования информационных систем (ИС)**

Проектирование ИС — это процесс создания модели будущей системы, включая её функции, структуру данных, интерфейсы и поведение.

Существует несколько **классификаций методов проектирования**, в зависимости от подхода и уровня детализации.

### **1. По уровню абстракции:**

* **Концептуальное проектирование** — общее представление о системе, её функциях и целях (без деталей реализации).
* **Логическое проектирование** — структура данных, логика процессов.
* **Физическое проектирование** — конкретная реализация (таблицы, индексы, интерфейсы, код).

### **2. По методу описания:**

* **Функционально-ориентированное проектирование**  
   Основа — функции системы и потоки данных.  
   Инструмент — **DFD-диаграммы**.  
   Пример: методология SADT.
* **Объектно-ориентированное проектирование**  
   Фокус — объекты, классы, методы.  
   Инструмент — **UML-диаграммы**.  
   Применяется при разработке ПО.
* **Информационно-ориентированное проектирование**  
   Основной объект — данные.  
   Используются **ER-диаграммы**, **IDEF1X**, **реляционная модель**.

### **3. По способу организации:**

* **Каскадный (Waterfall)** — этапы идут строго друг за другом (анализ → проектирование → реализация).
* **Итерационный (Agile, Scrum)** — постоянные циклы доработки, гибкое внесение изменений.
* **Прототипный (RAD)** — быстрая реализация прототипов и их тестирование.

**Вывод:**  
Выбор метода проектирования зависит от типа проекта, требований, команды и желаемой гибкости. Часто применяются **гибридные подходы**, сочетающие сильные стороны нескольких методов.

# **14 Билет**

### **1. Тестовые вопросы:**

1. Степень воздействия потоварного налога на объем продаж зависит от:  
    **а) наклонов линий спроса и предложения**  
    Чем более эластичен спрос или предложение, тем сильнее меняется объем продаж при введении налога.
2. Какие последствия вызывает проводимая монополистом ценовая дискриминация:  
    **в) растет прибыль и увеличивается объем продаж монополиста**  
    Монополист может продавать одному покупателю дороже, другому дешевле, что увеличивает общую прибыль и охват рынка.
3. Если номинальная заработная плата увеличилась на 10%, а инфляция — на 13%, то реальная зарплата:  
    **г) сократилась на 3%**  
    Реальная зарплата = номинальная – инфляция = 10% – 13% = –3%.
4. В кейнсианской теории спрос на деньги определяется мотивом:  
    **г) трансакционным, спекулятивным и предосторожности**  
    Люди хранят деньги для текущих расходов, страховки на случай непредвиденного и возможных выгодных вложений.
5. Предельная полезность – это:  
    **б) полезность, которую потребитель получает от потребления дополнительной единицы блага**  
    Например, удовольствие от одной лишней чашки кофе.

### **2. Сравнение архитектур реляционных и объектно-ориентированных СУБД**

**Реляционные СУБД (Relational DBMS)** и **Объектно-ориентированные СУБД (OODBMS)** — это два разных подхода к хранению и организации данных. Оба типа имеют свои сильные и слабые стороны в зависимости от задач.

### **Реляционные СУБД:**

**Основа:**

* Данные хранятся в **таблицах** (строки и столбцы).
* Используется **SQL** — стандартный язык запросов.

**Преимущества:**

* Надёжность, проверенная временем.
* Стандартизация (SQL).
* Отлично подходит для транзакционной обработки (например, банки, ERP).

**Недостатки:**

* Сложности при работе с **сложными структурами** данных (например, графы, вложенные объекты).
* "Импеданс несоответствия" — разрыв между объектами в коде и таблицами в БД.

### **Объектно-ориентированные СУБД:**

**Основа:**

* Данные хранятся в виде **объектов**, как в ООП: с полями и методами.
* Объекты могут содержать другие объекты (вложенность).

**Преимущества:**

* Прямое отображение кода программы на базу данных.
* Удобно хранить сложные структуры (например, конфигурации, документы).

**Недостатки:**

* Нет стандартного языка, как SQL.
* Меньше зрелых решений, ниже производительность при больших объемах данных.
* Сложнее масштабировать.

### **Сравнительная таблица:**

| **Критерий** | **Реляционные СУБД** | **Объектно-ориентированные СУБД** |
| --- | --- | --- |
| Хранение данных | Таблицы | Объекты |
| Язык запросов | SQL | Языки запросов OQL, API |
| Структура данных | Простая (табличная) | Сложная (вложенные объекты) |
| Гибкость | Меньше | Больше |
| Скорость при больших объёмах | Высокая | Ниже (обычно) |

**Вывод:**  
Реляционные СУБД лучше для классических бизнес-задач, где важна стабильность и скорость. Объектно-ориентированные — удобны при тесной интеграции с программным кодом и работе с "нестандартными" данными.

### **3. Масштабирование приложений. Миграция файл-серверных приложений в клиент-серверную среду**

#### **Масштабирование** — это процесс увеличения возможностей системы справляться с ростом нагрузки (больше пользователей, данных, операций).

### **Типы масштабирования:**

1. **Вертикальное (scale-up):**
   * Увеличение мощности одного сервера (CPU, RAM).
   * Простой, но дорогой путь. Есть пределы.
2. **Горизонтальное (scale-out):**
   * Добавление новых серверов.
   * Более гибко и эффективно, особенно для веб-приложений и микросервисов.

### **Файл-серверная архитектура:**

* Все данные и логика — в одном файле или папке, часто на сетевом диске.
* Пользователи напрямую открывают и работают с файлами.
* **Недостатки:**
  + Низкая безопасность.
  + Высокий риск повреждения данных.
  + Плохая производительность при одновременной работе.

### **Клиент-серверная архитектура:**

* **Сервер** — хранит данные и выполняет основную логику.
* **Клиент** — интерфейс пользователя, делает запросы к серверу.
* Пример: пользователь в приложении отправляет запрос → сервер обрабатывает и возвращает результат.

### **Миграция: как это происходит**

1. **Анализ текущей системы:**
   * Что используется, какие данные, кто работает.
2. **Разделение логики и интерфейса:**
   * Отделяется бизнес-логика от пользовательского интерфейса.
3. **Переход на СУБД:**
   * Вместо файлов данные перемещаются в базу данных (например, PostgreSQL).
4. **Создание API или серверного приложения:**
   * Сервер обрабатывает запросы, управляет логикой.
5. **Обновление клиента:**
   * Новый клиент (программа, веб-приложение) работает с сервером по сети.

**Преимущества клиент-серверной модели:**

* Безопасность.
* Быстродействие.
* Централизованное управление.
* Легче масштабировать.

**Вывод:**  
Переход с файл-серверной модели на клиент-серверную — это важный шаг в развитии любой системы, который обеспечивает **устойчивую, масштабируемую и безопасную архитектуру**.

# **15 Билет**

### **1. Тестовые вопросы:**

1. Если переменным фактором спроса является только цена, то изменяется:  
    **б) только величина спроса**  
    При изменении цены происходит движение вдоль кривой спроса, а не её сдвиг.
2. Спрос на факторы производства отличает от спроса на продукт его:  
    **г) производный характер**  
    Потребность в ресурсах зависит от спроса на конечный продукт.
3. Согласно закону Оукена, превышение безработицы на 2% означает, что ВВП отстаёт на:  
    **в) 4%**  
    Закон Оукена: 1% ↑ безработицы ≈ 2% ↓ ВВП по сравнению с потенциальным.
4. Встроенные стабилизаторы остаются безразличными к изменениям:  
    **а) платежного баланса**  
    Они автоматически реагируют на внутренние экономические изменения (доход, занятость), но не на внешнеторговые показатели.
5. Если спрос на товар эластичный, то снижение цены приведёт к:  
    **а) повышению дохода от продажи данного товара**  
    Так как количество проданного товара возрастёт сильнее, чем упала цена.

### **2. Сравнительная характеристика структур хранения данных**

**Структуры хранения данных** — это способы организации информации в базах данных и информационных системах. От выбора структуры зависит скорость работы, удобство обработки и масштабируемость.

### **1. Табличная (реляционная) структура:**

* **Форма:** строки и столбцы (таблицы).
* **Пример:** MySQL, PostgreSQL.
* **Плюсы:** удобна для стандартных бизнес-данных (клиенты, заказы).
* **Минусы:** плохо справляется с вложенными или гибкими структурами.

### **2. Иерархическая структура:**

* **Форма:** дерево, один "родитель" — много "детей".
* **Пример:** XML-документы, старые СУБД (IMS).
* **Плюсы:** понятна визуально, быстра при иерархическом доступе.
* **Минусы:** негибкая, тяжело модифицировать.

### **3. Сетевая структура:**

* **Форма:** записи связаны множественными путями (граф).
* **Плюсы:** позволяет задавать сложные связи.
* **Минусы:** сложно управлять, неудобна для типовых задач.

### **4. Объектная структура:**

* **Форма:** объекты, как в ООП, с методами и полями.
* **Пример:** ObjectDB, db4o.
* **Плюсы:** удобно для приложений, где логика работы тесно связана с данными.
* **Минусы:** ниже производительность при большом объеме данных.

### **5. Документно-ориентированная (NoSQL):**

* **Форма:** JSON, BSON, XML — документы.
* **Пример:** MongoDB, CouchDB.
* **Плюсы:** гибкость, легко хранить неструктурированные данные.
* **Минусы:** нет строгих связей, ограниченная поддержка транзакций.

### **6. Ключ-значение (Key-Value):**

* **Форма:** пара "ключ — значение".
* **Пример:** Redis, DynamoDB.
* **Плюсы:** очень быстро, просто.
* **Минусы:** мало структурности, плохо подходит для сложных запросов.

**Вывод:**  
Выбор структуры зависит от задач.

* Реляционная — для чётких, структурированных данных.
* NoSQL — для гибкости и масштабируемости.
* Объектная — для тесной интеграции с программным кодом.

### **3. Особенности построения информационных систем, основанных на модели COMET**

**COMET (Collaborative Object Modeling and Architectural Design Method)** — это методология проектирования информационных систем на основе **объектно-ориентированного моделирования**, разработанная для создания **надежных, гибких и масштабируемых приложений**, особенно в **распределённых системах**.

### **Ключевые особенности модели COMET:**

1. **Основывается на UML и ООП:**
   * Использует классы, объекты, связи, интерфейсы.
   * Широко применяет **диаграммы компонентов, последовательностей, классов.**
2. **Разделение логики по архитектурным слоям:**
   * **Представление (UI)** — интерфейс пользователя.
   * **Бизнес-логика (доменный слой)** — правила и процессы.
   * **Управление взаимодействием (control)** — связь между слоями.
   * **Доступ к данным (persistence)** — работа с базой данных.
3. **Поддержка паттернов проектирования:**
   * COMET активно использует **шаблоны проектирования** (MVC, фабрики, наблюдатель и др.).
   * Это повышает переиспользуемость кода и надёжность.
4. **Масштабируемость и модульность:**
   * COMET позволяет легко разрабатывать **распределённые приложения** (например, клиент-сервер, микросервисы).
   * Чёткое разграничение ответственности между компонентами.

### **Преимущества COMET:**

* Ясная архитектура и логика.
* Удобно для командной разработки.
* Подходит для сложных, распределённых и долго живущих систем.
* Легко внедрять изменения.

### **Пример применения:**

При создании корпоративной CRM-системы по COMET:

* Выделяют бизнес-объекты: Клиент, Сделка, Контакт.
* Определяют логику обработки заказов.
* Создают интерфейс (веб или десктоп).
* Изолируют слой данных (связь с базой).
* Используют шаблоны, чтобы обеспечить стабильность и поддержку.

**Вывод:**  
COMET — мощный объектно-ориентированный подход, который помогает строить **структурированные, масштабируемые и гибкие ИС**, особенно если они сложны или рассчитаны на долгосрочное развитие.

# **16 Билет**

### **1. Тестовые вопросы:**

1. **Товар Гиффена** — это товар, объем спроса на который:  
    **а) растет при росте его цены**  
    Это редкое исключение из закона спроса, характерное для жизненно важных, но "низших" товаров (например, хлеб для очень бедных слоёв населения).
2. Спрос на рынке труда зависит от:  
    **б) спроса на готовую продукцию, для производства которой он используется**  
    Если растёт спрос на продукцию — нужны рабочие, значит, растёт спрос на труд.
3. Среди утверждений определите **неверное**:  
    **г) в условиях полной занятости уровень фрикционной безработицы должен быть нулевым**  
    Фрикционная безработица существует ***всегда***, даже при полной занятости — это временные поиски новой работы.
4. Главным показателем экономического роста является:  
    **в) увеличение объема реального ВВП на душу населения**  
    Этот показатель учитывает инфляцию и рост численности населения, отражает реальное улучшение благосостояния.
5. Спрос на товар будет неэластичным, если:  
    **г) коэффициент ценовой эластичности меньше единицы**  
    Это значит, что изменение цены слабо влияет на объём спроса (например, лекарства).

### **2. Сравнение функций централизованных и распределённых СУБД**

**СУБД (системы управления базами данных)** могут быть централизованными и распределёнными в зависимости от того, **где физически хранятся данные** и **как к ним осуществляется доступ**.

### **Централизованные СУБД:**

* **Данные хранятся в одном месте** (на одном сервере).
* Пользователи подключаются к **одному источнику** данных.

#### Преимущества:

* Простота настройки и администрирования.
* Единая точка контроля и резервного копирования.
* Надёжные транзакции.

#### Недостатки:

* Ограниченная масштабируемость.
* Высокая нагрузка на один сервер.
* Уязвимость: сбой = остановка всей системы.

### **Распределённые СУБД:**

* Данные хранятся **на нескольких серверах**, возможно — в разных географических точках.
* Система выглядит как единое целое, но "под капотом" данные распределены.

#### Преимущества:

* Масштабируемость: можно добавлять узлы по мере роста.
* Устойчивость: сбой одного узла не останавливает всю систему.
* Быстрый доступ к данным ближе к пользователю (геораспределённость).

#### Недостатки:

* Сложность синхронизации данных.
* Требует продвинутых механизмов согласованности и безопасности.
* Труднее администрировать.

### **Сравнительная таблица:**

|  |  |  |
| --- | --- | --- |
| **Критерий** | **Централизованная СУБД** | **Распределённая СУБД** |
| Хранение данных | Один сервер | Несколько серверов |
| Надежность | Ниже (одна точка отказа) | Выше (резервирование узлов) |
| Масштабируемость | Ограничена | Высокая |
| Администрирование | Проще | Сложнее |
| Пример использования | Локальные ИС предприятий | Глобальные приложения, SaaS |

**Вывод:**  
Централизованные СУБД — хороший выбор для небольших, простых систем. Распределённые — для крупных, отказоустойчивых систем с высокой нагрузкой.

### **3. Структурный подход к проектированию информационных систем**

**Структурный подход** — это классическая методология анализа и проектирования ИС, которая основана на **разделении системы на функции и потоки данных**. Цель — детально описать, что делает система, какие данные обрабатывает и как части системы взаимодействуют.

### **Ключевые принципы:**

1. **Функциональное моделирование:**
   * Система описывается как **набор функций** (процессов), каждый из которых принимает входные данные и выдает результат.
   * Пример: "Принять заказ", "Рассчитать зарплату".
2. **DFD (Data Flow Diagram):**
   * Основной инструмент. Показывает:
     + Источники/получатели информации (внешние сущности),
     + Обработку данных (процессы),
     + Хранилища данных,
     + Потоки данных между компонентами.
3. **Декомпозиция функций:**
   * Сложные функции разбиваются на простые — от общего к частному.
   * Это делает систему понятной и управляемой.
4. **Модель данных (ER-диаграммы):**
   * Используются для описания структуры данных: какие сущности есть, как они связаны.

### **Преимущества:**

* Хорошо подходит для бизнес-приложений с чёткими алгоритмами.
* Легко читается и документируется.
* Удобен для взаимодействия с заказчиком на ранних этапах.

### **Недостатки:**

* Мало внимания интерфейсам и пользовательскому опыту.
* Сложно адаптироваться к изменениям требований (жёсткая структура).
* Плохо применим для интерактивных и гибких систем (например, веб-приложений с частыми изменениями).

**Пример:**  
Система управления заказами:

* Вход: заявка клиента.
* Процессы: "Проверка клиента", "Формирование накладной", "Выпуск счёта".
* Выход: документ заказа.
* Всё это — звенья в цепочке DFD.

**Вывод:**  
Структурный подход — это **логичная, иерархическая методика**, которая позволяет грамотно спланировать ИС с акцентом на **функции и обработку данных**. Хорошо подходит для автоматизации учёта и документооборота.

# **17 Билет**

### **1. Тестовые вопросы:**

1. Если коэффициент перекрёстной эластичности спроса по цене больше нуля, то товары являются:  
    **а) взаимозаменяемыми**  
    Если цена одного товара растёт, спрос на другой увеличивается (пример: масло и маргарин).
2. \_\_\_ является доходом на капитал:  
    **а) процент**  
    Капитал (в виде вложений, вкладов) приносит доход в форме процента.
3. Естественная норма безработицы:  
    **б) является нормой, при которой существует баланс между количеством вакансий и работников, ищущих работу**  
    Это уровень безработицы, при котором сохраняется фрикционная и структурная безработица, но нет циклической.
4. К источникам экстенсивного пути развития экономической системы относится:  
    **в) увеличение объема факторов производства**  
    Например, привлечение дополнительной рабочей силы или расширение площадей.
5. Товар Гиффена – это товар, объем спроса на который:  
    **а) растет при росте его цены**  
    Редкое явление для «низших» товаров, когда эффект дохода превышает эффект замещения.

### **2. Анализ архитектур многозвенных приложений для работы с базами данных**

**Многозвенная (многоуровневая) архитектура** — это способ организации программных приложений, где разные части системы отвечают за свои конкретные задачи. Особенно важно это при работе с базами данных, так как она обеспечивает **масштабируемость, безопасность и надёжность**.

### **Основные архитектуры:**

#### **1. Двухзвенная архитектура (2-tier):**

* **Клиент** напрямую обращается к **базе данных**.
* Пример: приложение Access или Excel с прямым подключением к БД.

**Плюсы:**

* Простая реализация.
* Подходит для малых офисных приложений.

**Минусы:**

* Низкая безопасность.
* Плохая масштабируемость.
* Высокая нагрузка на БД.

#### **2. Трехзвенная архитектура (3-tier):**

* **Клиентский уровень (UI)** — интерфейс пользователя.
* **Сервер приложений (логика)** — обработка бизнес-правил.
* **Сервер БД** — хранение и доступ к данным.

**Плюсы:**

* Чёткое разделение обязанностей.
* Повышенная безопасность.
* Легкость масштабирования (можно расширить сервер приложений).

#### **3. Многозвенная архитектура (n-tier):**

* Добавляются слои: кэш, API, балансировка, аналитика и др.
* Используется в крупных корпоративных и веб-приложениях.

**Преимущества:**

* Надёжность и отказоустойчивость.
* Высокая производительность.
* Возможность работы в распределённой среде (кластеризация, микросервисы).

**Вывод:**  
Многозвенная архитектура позволяет строить гибкие и масштабируемые приложения. Для баз данных она особенно важна, так как обеспечивает **быстрый доступ**, **распределённую нагрузку** и **централизованное управление данными**.

### **3. Понятие информационной системы. Задачи, решаемые информационными системами**

**Информационная система (ИС)** — это совокупность программных, аппаратных, организационных и человеческих компонентов, предназначенных для **сбора, хранения, обработки, передачи и представления информации**.

### **Состав ИС:**

1. **Аппаратное обеспечение** — серверы, ПК, сети.
2. **Программное обеспечение** — СУБД, приложения, интерфейсы.
3. **Персонал** — пользователи, администраторы, аналитики.
4. **Методы** — правила, алгоритмы, бизнес-процессы.

### **Задачи, решаемые ИС:**

#### 1. **Сбор данных:**

* Автоматизация ввода данных (сканеры, формы, API).
* Пример: регистрация заказов, анкетирование клиентов.

#### 2. **Хранение информации:**

* Использование баз данных для долгосрочного хранения и обеспечения целостности данных.

#### 3. **Обработка информации:**

* Расчёты, агрегация, фильтрация.
* Пример: расчёт заработной платы, отчётность.

#### 4. **Передача данных:**

* Организация сетевого взаимодействия между элементами системы.
* Пример: обмен между офисами, филиалами.

#### 5. **Представление информации:**

* Интерфейсы, отчёты, графики.
* Пример: дашборды в BI-системах, печатные формы.

### **Типы ИС:**

* **Управленческие (MIS)**
* **Бухгалтерские**
* **ERP-системы**
* **CRM-системы**
* **SCADA (в промышленности)**

**Вывод:**  
Информационные системы — основа цифровой инфраструктуры предприятия. Они позволяют **ускорить бизнес-процессы**, **снизить ошибки** и **повысить эффективность** управления на всех уровнях.

# **18 Билет**

### **1. Тестовые вопросы:**

1. Спрос на товар будет неэластичным, если:  
    **г) коэффициент ценовой эластичности меньше единицы**  
    Это означает, что покупатели слабо реагируют на изменение цены: спрос почти не меняется.
2. Предложение земли как фактора производства является:  
    **а) абсолютно неэластичным**  
    Количество земли фиксировано и не зависит от цены.
3. Уровень безработицы при полной занятости:  
    **б) не учитывает циклическую безработицу**  
    При полной занятости возможна фрикционная и структурная безработица, но ***не*** циклическая.
4. К интенсивным факторам экономического роста можно отнести:  
    **а) внедрение научно-технического прогресса в производство**  
    Интенсивные факторы улучшают ***качество*** ресурсов, а не их количество.
5. Если переменным фактором спроса является только цена, то изменяется:  
    **б) только величина спроса**  
    Это движение вдоль кривой спроса, а не её сдвиг.

### **2. Трехуровневая архитектура ANSI/SPARC**

**ANSI/SPARC** — это эталонная модель архитектуры баз данных, предложенная Американским национальным институтом стандартов. Она состоит из **трех уровней**, каждый из которых играет свою роль в абстракции данных.

### **Уровни архитектуры:**

#### **1. Внутренний уровень (Internal level):**

* Отвечает за **физическое хранение данных**.
* Включает: формат хранения, пути доступа, индексы.
* **Пример:** как СУБД размещает таблицы на диске, как оптимизирует хранение.

#### **2. Концептуальный уровень (Conceptual level):**

* Определяет **структуру всей базы данных** — без привязки к конкретным пользователям.
* Включает: таблицы, поля, связи, ограничения целостности.
* **Пример:** схема базы данных — какие сущности есть и как они связаны.

#### **3. Внешний уровень (External level):**

* Отвечает за **представление данных конкретным пользователям** или приложениям.
* Один пользователь может видеть одну часть данных, другой — другую.
* **Пример:** бухгалтер видит отчёты, а менеджер — список клиентов.

### **Преимущества трехуровневой архитектуры:**

* **Изоляция уровней:** изменение физической структуры не влияет на пользователей.
* **Гибкость:** легко создать несколько представлений для разных ролей.
* **Безопасность:** можно ограничить доступ на уровне представлений.

**Итог:**  
Модель ANSI/SPARC помогает **разделить логику хранения, описания и отображения данных**, упрощает разработку, поддержку и масштабирование БД.

### **3. Специфика построения файл-серверных приложений**

**Файл-серверные приложения** — это программы, в которых **вся база данных размещается на сервере в виде файлов**, а клиентская часть работает с этими файлами **напрямую** по сети.

### **Как это работает:**

* Сервер предоставляет пользователю доступ к файлу (например, базе в формате .mdb).
* Программа на клиентском компьютере открывает файл и работает с ним как с локальным.
* Пример: Microsoft Access, старые версии 1С, FoxPro.

### **Преимущества:**

* Простота реализации.
* Не требует отдельного сервера приложений.
* Подходит для небольших рабочих групп (2–5 пользователей).

### **Недостатки:**

1. **Низкая производительность:**
   * Все данные переносятся по сети к клиенту.
   * При работе нескольких пользователей возникают конфликты и торможения.
2. **Риск повреждения данных:**
   * Если клиент завис или сеть оборвалась, файл может быть повреждён.
3. **Отсутствие централизованной логики:**
   * Вся бизнес-логика находится на клиенте, сложно контролировать корректность операций.
4. **Ограниченная безопасность:**
   * Нет изоляции доступа, каждый пользователь может повредить файл.

### **Когда используется:**

* В небольших организациях без выделенного сервера.
* В учебных целях.
* Там, где важна скорость реализации, а не масштабируемость.

**Вывод:**  
Файл-серверная модель проста и дешева в реализации, но **ограничена в масштабируемости, безопасности и производительности**. Современные системы переходят к клиент-серверной и многозвенной архитектуре.

# **19 Билет**

### **1. Тестовые вопросы:**

1. Если уменьшение цены на 10 % приводит к снижению объема предложения на 16 %, то данное предложение:  
    **б) эластично**  
    Если процент изменения предложения больше, чем изменение цены (в данном случае 16% > 10%), это говорит об ***эластичности*** предложения.
2. Если продавцы компьютерной техники знают гораздо больше о товарах, чем покупатели, то имеет место:  
    **а) асимметричность информации**  
    Одна сторона (продавец) владеет большей информацией, чем другая — это приводит к рыночным сбоям.
3. Уравнение обмена И. Фишера:  
    **а) M \* V = P \* Q**  
    Оно показывает взаимосвязь между количеством денег, скоростью их обращения, ценами и объемом произведённых товаров.
4. Экономический рост может быть проиллюстрирован:  
    **б) сдвигом вправо кривой производственных возможностей**  
    Это означает увеличение производственного потенциала экономики.
5. Реализацию интересов рыночных субъектов и соединение интересов продавца и покупателя обеспечивает:  
    **б) посредническая функция рынка**  
    Рынок как посредник согласует интересы сторон — продавца и покупателя.

### **2. Методы и средства концептуального проектирования реляционных баз данных**

**Концептуальное проектирование** — это первый и наиболее абстрактный этап создания базы данных. Он описывает **что будет храниться**, **какие объекты (сущности) существуют**, и **как они связаны между собой**, без детализации таблиц или SQL-кода.

### **Основная цель:**

Понять и структурировать данные в логической форме, понятной и пользователям, и разработчикам, не привязываясь к конкретной СУБД.

### **Методы:**

1. **ER-моделирование (модель "сущность–связь"):**
   * Описание сущностей, их атрибутов и связей.
   * Пример: "Клиент", "Заказ", "Товар" — сущности; "делает заказ" — связь.
2. **IDEF1X (вариант ER-моделирования):**
   * Более строгий подход: точно задаются первичные и внешние ключи, типы связей.
3. **UML-диаграммы классов (иногда используются):**
   * Особенно в объектно-реляционных и программных системах.

### **Средства:**

* **CASE-средства (Computer-Aided Software Engineering):**
  + **Erwin Data Modeler**
  + **Oracle SQL Developer Data Modeler**
  + **Visual Paradigm**
  + **StarUML**
  + Позволяют рисовать модели, проверять связи и даже генерировать SQL.

### **Преимущества концептуального проектирования:**

* Отражает бизнес-логику системы.
* Независимо от конкретной платформы.
* Позволяет выявить противоречия и пробелы до реализации.

**Итог:**  
Концептуальное проектирование — это **фундамент будущей базы данных**, формализующий структуру информации и её взаимосвязи на высоком уровне.

### **3. Моделирование потоков данных (DFD — Data Flow Diagrams)**

**Моделирование потоков данных** — это один из ключевых методов структурного анализа, который описывает **как информация перемещается** внутри системы, кто её использует и куда она направляется.

### **Что такое DFD (Data Flow Diagram):**

Это диаграмма, на которой отображаются:

* **Процессы** — что делает система (круг или овал).
* **Хранилища данных** — где данные сохраняются (две параллельные линии).
* **Внешние сущности** — пользователи, системы вне ИС (прямоугольники).
* **Потоки данных** — линии со стрелками, показывающие направление движения информации.

### **Пример:**

Для системы интернет-магазина:

* Внешняя сущность: Покупатель.
* Процесс: "Оформить заказ".
* Хранилище: "База данных заказов".
* Поток: "Информация о заказе" от покупателя в систему.

### **Преимущества DFD:**

* Наглядность — легко объяснить заказчику.
* Упрощает анализ процессов.
* Помогает выявить избыточность, дублирование и ошибки в потоках.

### **Правила моделирования:**

* У каждого потока есть **начало и конец**.
* Данные не могут идти напрямую от внешней сущности в хранилище — всегда через процесс.
* Уровни моделей: сначала верхний уровень (контекстная диаграмма), потом — детализация.

**Итог:**  
Моделирование потоков данных — это **инструмент визуального анализа процессов**, помогающий спроектировать логичную, понятную и эффективную информационную систему до начала её программной реализации.

# **20 Билет**

### **1. Тестовые вопросы:**

1. Если спрос на товар **эластичный**, то снижение цены приводит к:  
    **а) повышению дохода от продажи данного товара**  
    При эластичном спросе снижение цены вызывает ***сильный рост объема продаж***, и общая выручка растёт.
2. К «провалам» рынка **НЕ** относят:  
    **в) производство частных благ**  
    Частные блага распределяются эффективно с помощью рынка, в отличие от общественных.
3. Банковский мультипликатор:  
    **б) уменьшается с ростом нормы обязательных резервов**  
    Чем больше банк обязан резервировать, тем меньше он может выдать кредитов, и тем ***ниже*** мультипликатор.
4. Величина экспорта на душу населения характеризует:  
    **г) степень открытости экономики**  
    Это показатель того, насколько страна вовлечена во внешнюю торговлю.
5. Излишек потребителя — это:  
    **б) разница между максимальной ценой, которую потребитель готов заплатить, и фактической ценой**  
    Это "экономия", которую получает покупатель благодаря тому, что товар стоит меньше, чем он был бы готов за него отдать.

### **2. SQL. Основные запросы SQL**

**SQL (Structured Query Language)** — это стандартный язык для работы с реляционными базами данных. Он позволяет **добавлять, извлекать, изменять и удалять данные**.

### **Основные типы SQL-запросов:**

#### **1. SELECT** — извлечение данных:

SELECT имя, возраст FROM сотрудники WHERE отдел = 'Бухгалтерия';

Выводит имена и возраст сотрудников из бухгалтерии.

#### **2. INSERT** — добавление данных:

INSERT INTO сотрудники (имя, возраст, отдел)

VALUES ('Иванов И.И.', 35, 'Отдел продаж');

1. **UPDATE** — изменение данных:

UPDATE сотрудники

SET возраст = 36

WHERE имя = 'Иванов И.И.';

1. **DELETE** — удаление строк:

DELETE FROM сотрудники WHERE возраст > 65;

### **Дополнительно:**

* **ORDER BY** — сортировка
* **GROUP BY** — группировка
* **JOIN** — объединение таблиц по общим полям
* **LIKE** — поиск по шаблону
* **IN**, **BETWEEN**, **IS NULL** — логические операторы

Пример запроса с объединением таблиц (JOIN):

SELECT заказы.id, клиенты.имя

FROM заказы

JOIN клиенты ON заказы.клиент\_id = клиенты.id;

**Итог:**  
SQL — это универсальный язык для работы с БД, позволяющий выполнять **все основные операции**: от просмотра до изменения и анализа данных.

### **3. Методы и средства проектирования информационных систем**

Проектирование информационной системы (ИС) — это процесс **планирования её структуры, функционала, интерфейсов и взаимодействия компонентов**, прежде чем начнётся программная реализация.

### **Методы проектирования ИС:**

#### **1. Структурный подход (DFD, ER):**

* Система представляется в виде **процессов и потоков данных**.
* Используются:
  + **DFD** — диаграммы потоков данных.
  + **ER-модели** — описание структуры данных.
* Пример: методология **SADT**, **IDEF0**, **IDEF1X**.

#### **2. Объектно-ориентированный подход (UML):**

* Система проектируется как **набор объектов** с методами и атрибутами.
* Используются **UML-диаграммы**: классов, прецедентов, последовательностей и др.
* Хорошо подходит для ПО, написанного на языках типа Java, C# и Python.

#### **3. Информационно-ориентированный подход:**

* Основной акцент — **данные**, их структура и отношения между ними.
* Часто используется в больших БД-проектах и BI-системах.

### **Средства (инструменты) проектирования ИС:**

* **CASE-средства:**
  + Enterprise Architect
  + Visual Paradigm
  + Erwin Data Modeler
  + Oracle SQL Developer Data Modeler
* **Онлайн-редакторы:**
  + Draw.io
  + Lucidchart
  + Creately
* **IDE/платформы разработки:**
  + Visual Studio
  + JetBrains
  + Power BI (для визуализации и анализа данных)

### **Вывод:**

Проектирование ИС требует выбора **методологии**, соответствующей целям системы, и **инструментов**, которые облегчат моделирование, описание и автоматизацию. Правильное проектирование — основа эффективной разработки.

# **21 Билет**

### **1. Тестовые вопросы:**

1. Кардиналистский подход к полезности предполагает:  
    **г) возможность определения количественной величины полезности**  
    Кардиналисты считают, что полезность можно ***измерить в числовом выражении*** (например, "10 единиц удовольствия"). Ординалисты лишь сравнивают предпочтения (без чисел).
2. Затраты третьих лиц, не отраженные в ценах:  
    **г) отрицательные внешние эффекты**  
    Пример: завод загрязняет воздух, жители болеют, но цена продукции не включает их убытки.
3. Повышение учетной ставки Центральным банком направлено на:  
    **б) стимулирование роста величины сбережений населения**  
    Повышение ставки делает кредиты дорогими и стимулирует людей и бизнес ***меньше тратить и больше сберегать***.
4. Автор теории сравнительных преимуществ:  
    **б) Д. Риккардо**  
    Он доказал, что даже менее производительные страны могут ***выигрывать от торговли***, специализируясь на том, что у них получается лучше относительно других товаров.
5. Альтернативные издержки посещения хоккея при зарплате 700 руб./час и 3-х часах отсутствия:  
    **а) 2100 рублей**  
    700 × 3 = 2100 — это упущенный доход, ***альтернативная стоимость*** времени.

### **2. Назначение реляционной алгебры и реляционного исчисления. Начальная реляционная алгебра Э. Кодда**

**Реляционная алгебра** и **реляционное исчисление** — это **формальные методы работы с реляционными базами данных**, предложенные **Эдгаром Коддом**, основателем реляционной модели данных. Они легли в основу языка SQL.

### **Реляционная алгебра:**

Это **процедурный язык запросов**, в котором описывается **как получить результат** — пошагово, через операции над таблицами.

#### Основные операции:

1. **SELECT (σ)** — выбор строк по условию
2. **PROJECT (π)** — выбор нужных столбцов
3. **UNION (∪)** — объединение результатов двух таблиц
4. **INTERSECT (∩)** — пересечение двух таблиц
5. **SET DIFFERENCE (-)** — разность (исключение записей одной из таблиц)
6. **JOIN (⨝)** — соединение таблиц по общим значениям
7. **CARTESIAN PRODUCT (×)** — все возможные пары строк двух таблиц

#### Пример:

π\_ФИО(σ\_отдел='Продажи'(Сотрудники))

Выбрать ФИО сотрудников из отдела "Продажи".

### **Реляционное исчисление:**

Это **декларативный язык** — в нём описывается **что нужно получить**, а не как.

Бывает:

* **Кортежное** — работа с целыми строками.
* **Доменное** — работа с отдельными значениями.

Пример:

Найти все X, такие что X — сотрудник и X работает в "Продажах".

### **Зачем это нужно:**

* Даёт теоретическую основу языка SQL.
* Позволяет описывать сложные запросы математически.
* Используется в **оптимизаторах запросов** внутри СУБД.

**Итог:**  
Реляционная алгебра — это **основа SQL**. Исчисление — альтернатива для описания запросов на логическом уровне. Вместе они формируют **формальный язык работы с данными**.

### **3. Классификация средств проектирования информационных систем**

**Средства проектирования ИС** — это программы и платформы, которые помогают **анализировать, моделировать и документировать** структуру и логику будущей информационной системы.

### **Классификация по назначению:**

#### **1. CASE-средства (Computer-Aided Software Engineering):**

* Предназначены для **автоматизации проектирования и документации**.
* Функции:
  + Построение **ER-диаграмм**, **DFD**, **UML**;
  + Генерация SQL-кода;
  + Формирование ТЗ.
* Примеры:
  + Enterprise Architect
  + Visual Paradigm
  + Erwin Data Modeler
  + PowerDesigner

#### **2. Средства прототипирования и интерфейсного проектирования:**

* Используются на стадии проектирования пользовательского интерфейса.
* Примеры:
  + Figma
  + Adobe XD
  + Axure

#### **3. Средства моделирования процессов:**

* Ориентированы на бизнес-процессы (BPMN, IDEF0, EPC).
* Примеры:
  + Bizagi Modeler
  + ARIS Express
  + Draw.io

#### **4. Средства интеграции и управления архитектурой предприятия:**

* Применяются в крупных организациях.
* Позволяют строить связи между ИТ-системами, бизнес-функциями и процессами.
* Примеры:
  + Sparx EA
  + IBM Rational System Architect

**Итог:**  
Средства проектирования ИС выбираются **в зависимости от цели**: моделировать данные, интерфейс, процессы или всё вместе. Современные CASE-средства позволяют ускорить проектирование и уменьшить число ошибок за счёт визуализации и автоматизации.

# **22 Билет**

### **1. Тестовые вопросы:**

1. Предельная полезность — это:  
    **б) полезность, которую потребитель получает от потребления дополнительной единицы блага**  
    Это прирост удовлетворения от каждой следующей единицы товара. Обычно она снижается с каждой новой единицей (закон убывающей предельной полезности).
2. Кривая Лоренца за три года сдвинулась дальше от биссектрисы. Это означает:  
    **г) усиление дифференциации доходов населения**  
    Чем дальше кривая от идеального равенства (биссектрисы), тем выше неравенство.
3. М1 включает в себя:  
    **а) металлические и бумажные наличные деньги, денежные средства на текущих счетах**  
    Это наиболее «ликвидные» деньги — наличность и средства, доступные сразу.
4. Демпинг — это продажа товара на внешнем рынке по цене:  
    **в) ниже цены аналогичного товара на внутреннем рынке страны – импортера**  
    Цель — захватить рынок за рубежом, часто за счёт временного снижения прибыли.
5. Сдвиг кривой производственных возможностей вправо означает:  
    **а) увеличение ресурсов**  
    Такой сдвиг показывает, что страна может производить больше благодаря росту ресурсов или улучшению технологий.

### **2. Этапы проектирования баз данных, основанных на различных моделях данных**

Проектирование базы данных проходит несколько **этапов**, которые зависят от модели хранения: реляционной, объектно-ориентированной, иерархической и др. Основная цель — создать **структуру, в которой данные хранятся эффективно, логично и безопасно.**

### **Общие этапы проектирования БД:**

#### **1. Концептуальное проектирование**

* Определяются основные сущности (объекты) и связи между ними.
* Используются **ER-диаграммы**.
* Независимо от типа БД — общая логика.

#### **2. Логическое проектирование**

* Перевод модели в формат конкретной БД:
  + для реляционной — таблицы с первичными/внешними ключами;
  + для объектной — классы и их отношения;
  + для иерархической — древовидные структуры.
* Уточняется тип данных, связи, ограничения.

#### **3. Физическое проектирование**

* Настройка **хранения** данных: индексирование, партиционирование, размещение на дисках.
* Оптимизация для быстродействия и отказоустойчивости.

### **Особенности для разных моделей:**

#### **Реляционная модель**

* Строгая структура: таблицы, ключи, нормализация.
* Основной инструмент — **SQL**.
* Подходит для большинства бизнес-приложений.

#### **Объектно-ориентированная модель**

* Данные хранятся в виде объектов, близко к логике программирования.
* Поддержка вложенных структур и методов.
* Хорошо подходит для инженерных и графических систем.

#### **Иерархическая и сетевая модели**

* Устаревшие, но применяются в некоторых промышленных/гос. системах.
* Требуют чёткой структуры "родитель—потомок".

**Итог:**  
Выбор модели и этап проектирования зависит от задач. Однако последовательность **концепция → логика → физика** — универсальна для всех типов баз данных.

### **3. Специфика построения клиент-серверных приложений**

**Клиент-серверная архитектура** — это структура приложения, где функции разделены между двумя уровнями:

* **Клиент** — пользовательский интерфейс.
* **Сервер** — логика обработки данных и взаимодействие с базой.

### **Особенности архитектуры:**

#### **1. Чёткое разделение обязанностей:**

* Клиент отвечает за **отображение и ввод данных**.
* Сервер — за **хранение, обработку, проверку данных**.

#### **2. Централизация:**

* Все данные хранятся на сервере — это **повышает безопасность** и **облегчает резервное копирование**.

#### **3. Масштабируемость:**

* Сервер можно улучшить или масштабировать без замены клиентов.
* В многопользовательской среде (интернет-магазины, CRM) это особенно важно.

### **Компоненты клиент-серверного приложения:**

* **Клиент** — веб-браузер, мобильное приложение, десктоп-программа.
* **Сервер приложений** — логика (на Python, Java, PHP, .NET).
* **База данных** — PostgreSQL, MySQL, MS SQL и др.
* **API** — интерфейс общения между клиентом и сервером (обычно REST или GraphQL).

### **Преимущества:**

* Безопасность и контроль.
* Разгрузка клиента (вся обработка на сервере).
* Централизованные обновления.

### **Недостатки:**

* Требуется стабильная сеть.
* Сложнее реализовать, чем файл-серверную архитектуру.

**Итог:**  
Клиент-серверная архитектура — **универсальное и современное решение**, применимое как для локальных, так и распределённых систем. Она обеспечивает стабильную работу, масштабируемость и безопасность.

# **23 Билет**

### **1. Тестовые вопросы:**

1. Изменение цены одного из товаров при прочих равных условиях:  
    **б) вызовет изменение наклона бюджетной линии**  
    Если цена одного товара изменяется, а доход и цена другого остаются прежними — изменяется наклон.
2. Положительный внешний эффект производства может быть следствием:  
    **г) производства лекарственных средств**  
    Такой эффект — это ***выгода для общества***, не оплаченная и не включённая в цену.
3. Прямое влияние на увеличение количества денег в обращении оказывает:  
    **г) рост цен выпускаемых товаров**  
    Рост цен увеличивает потребность в большем количестве денег для покупок — это влияет на количество денег в обращении.
4. Торговый баланс:  
    **а) является частью платежного баланса**  
    Он учитывает разницу между экспортом и импортом товаров и входит в текущий счёт платежного баланса.
5. Форма собственности: государство — владелец, коллектив использует здание за плату:  
    **а) государственная**  
    Собственность остаётся за государством, а коллектив — лишь временный пользователь.

### **2. Сравнительная характеристика моделей данных**

**Модель данных** — это способ представления информации в базе данных. Существует несколько типов моделей, каждая со своими преимуществами и областью применения.

### **Основные модели данных:**

#### **1. Иерархическая модель:**

* Данные организованы по принципу **дерева**: один родитель — много потомков.
* Пример: структура каталогов на компьютере.

**Плюсы:**

* Быстрый доступ при фиксированной структуре.
* Простая навигация по иерархии.

**Минусы:**

* Жёсткие связи, трудно перестраивать.
* Неэффективна при изменении связей.

#### **2. Сетевая модель:**

* Каждая запись может иметь **несколько связей** с другими.
* Подходит для сложных взаимосвязей (например, проекты и исполнители).

**Плюсы:**

* Гибче иерархической.
* Поддерживает M:N связи.

**Минусы:**

* Сложность в управлении.
* Неудобна для визуализации и понимания.

#### **3. Реляционная модель (наиболее распространена):**

* Данные хранятся в **таблицах**.
* Используются ключи, индексы, связи между таблицами.

**Плюсы:**

* Универсальность, поддержка SQL.
* Хорошая нормализация и целостность данных.

**Минусы:**

* Неудобна для вложенных или гибких структур.

#### **4. Объектно-ориентированная модель:**

* Данные хранятся в виде **объектов** (как в ООП).
* Поддержка сложных типов: списки, вложенные объекты.

**Плюсы:**

* Близка к логике программирования.
* Удобна для сложных приложений.

**Минусы:**

* Низкая совместимость с реляционными СУБД.
* Меньшая распространённость.

#### **5. Документно-ориентированная (NoSQL):**

* Данные хранятся в виде документов (JSON, BSON).
* Пример: MongoDB.

**Плюсы:**

* Гибкость, не требует жёсткой схемы.
* Хорошо масштабируется.

**Минусы:**

* Сложности с транзакциями.
* Нет строгих связей между объектами.

**Вывод:**  
Выбор модели зависит от требований к структуре, скорости, типу данных и способу доступа. **Реляционная модель** — стандарт в большинстве бизнес-приложений, **NoSQL и объектные** — хороши для гибких, распределённых систем.

### **3. Объектно-ориентированные концепции построения информационных систем**

**Объектно-ориентированная концепция** (ОО-концепция) — это метод построения информационных систем (ИС), в основе которого лежит **объектно-ориентированное программирование (ООП)** и представление компонентов системы как **объектов** с состоянием и поведением.

### **Основные принципы ОО-концепции:**

#### **1. Инкапсуляция**

* Объект скрывает внутреннюю реализацию, открывая только нужные данные и методы.
* Обеспечивает надёжность и упрощает сопровождение.

#### **2. Наследование**

* Возможность создавать новые объекты на основе уже существующих, переиспользуя код.

#### **3. Полиморфизм**

* Одинаковый интерфейс может иметь разную реализацию.
* Пример: метод print() у разных классов работает по-разному, но вызывается одинаково.

#### **4. Абстракция**

* Модель выделяет только существенные характеристики объектов, игнорируя неважные.

### **Преимущества объектно-ориентированного подхода:**

* Хорошая масштабируемость и модульность.
* Возможность переиспользовать компоненты.
* Подходит для сложных ИС с разнообразной логикой.
* Легко перейти от **анализа к реализации** — классы в модели становятся классами в коде.

### **Применение:**

* ERP-системы, CRM, облачные сервисы.
* Пример: объект «Пользователь» может содержать логин, пароль, методы входа, выхода, обновления профиля.

### **Инструменты:**

* **UML (Unified Modeling Language)** — основной язык моделирования ОО-систем.
* CASE-средства:
  + Enterprise Architect
  + Visual Paradigm
  + StarUML

**Вывод:**  
Объектно-ориентированный подход делает проектирование ИС **более гибким, логичным и приближенным к реальной структуре программного кода**, что упрощает как разработку, так и поддержку.

# **24 Билет**

### **1. Тестовые вопросы:**

1. Метод познания, предполагающий разделение целого на части:  
    **г) анализ**  
    Анализ — это разложение сложного объекта на составляющие с целью лучшего понимания.
2. От потребительских предпочтений зависит:  
    **б) положение и наклон кривых безразличия**  
    Кривые безразличия отражают ***личные предпочтения*** между различными наборами благ.
3. Благо, которым могут одновременно пользоваться многие:  
    **б) общественное благо**  
    Примеры: уличное освещение, оборона — нельзя исключить из потребления и нет соперничества.
4. Деньги служат:  
    **г) все ответы верны**  
    Деньги выполняют функции: меры стоимости, средства обращения, платежа, накопления и др.
5. К микроэкономическому утверждению **НЕ** относится:  
    **б) с 2000 г. в России наблюдается экономический рост**  
    Это макроэкономика, так как охватывает экономику в целом. Остальные варианты — про отдельные рынки.

### **2. Методы и средства логического проектирования реляционных баз данных**

**Логическое проектирование** — это этап, на котором **концептуальная модель данных** (ER-диаграммы и связи) преобразуется в **структуру таблиц**, пригодную для реализации в реляционной СУБД.

### **Цель логического проектирования:**

Создать **реляционную схему базы данных**, оптимизированную для хранения, связи и обработки информации.

### **Основные методы:**

#### **1. Нормализация:**

* Разделение данных на таблицы так, чтобы избежать дублирования и обеспечить целостность.
* Происходит в несколько **нормальных форм** (1НФ, 2НФ, 3НФ и т. д.).
* Пример: таблицу "Заказы" делят на "Заказы" и "Покупатели", если покупательские данные повторяются.

#### **2. Определение ключей:**

* **Первичный ключ** — уникальный идентификатор строки (например, id).
* **Внешний ключ** — поле, ссылающееся на первичный ключ в другой таблице.

#### **3. Установление связей между таблицами:**

* Один ко многим (1:N) — например, один клиент может иметь много заказов.
* Многие ко многим (M:N) — реализуется через промежуточную таблицу.

#### **4. Определение ограничений:**

* NOT NULL, UNIQUE, CHECK, DEFAULT — обеспечивают правильность и согласованность данных.

### **Средства:**

* **SQL DDL (Data Definition Language)**:  
   CREATE TABLE, ALTER TABLE, DROP TABLE.
* **CASE-средства**:  
   Erwin Data Modeler, SQL Developer Data Modeler, Visual Paradigm.

**Итог:**  
Логическое проектирование — важнейший шаг, где концептуальные идеи превращаются в **структурированную, реализуемую схему таблиц**, соответствующую требованиям бизнеса и обеспечивающую надёжную работу БД.

### **3. Базовая архитектура сервера баз данных**

**Сервер баз данных** — это программно-аппаратный комплекс, обеспечивающий **хранение, обработку и защиту данных**, а также обслуживание запросов от пользователей и приложений.

### **Основные компоненты архитектуры:**

#### **1. Ядро СУБД:**

* Отвечает за **выполнение SQL-запросов**, управление транзакциями, блокировками и обработку данных.
* Включает оптимизатор запросов, планировщик выполнения и ядро хранения.

#### **2. Механизм управления данными:**

* **Хранилище данных** — файлы, где физически сохраняется информация.
* **Кэш (буферный пул)** — ускоряет доступ к часто используемым данным.
* **Журнал транзакций (лог)** — обеспечивает восстановление данных в случае сбоя.

#### **3. Подсистема безопасности:**

* Аутентификация пользователей (логин/пароль, роли).
* Ограничения доступа к данным и операциям.

#### **4. Сетевой интерфейс:**

* Обеспечивает обмен данными между клиентом и сервером по сетевым протоколам (TCP/IP, HTTP, ODBC, JDBC).
* Примеры: PostgreSQL, MySQL, SQL Server.

#### **5. Механизмы репликации и резервного копирования:**

* Позволяют создавать копии данных, обеспечивая отказоустойчивость и защиту от потерь.

### **Дополнительно:**

* **Триггеры, процедуры и функции** — логика, выполняемая на сервере.
* **Индексы** — ускоряют поиск по таблицам.

**Итог:**  
Базовая архитектура сервера БД обеспечивает **устойчивую, защищённую и высокопроизводительную среду**, в которой хранятся и обрабатываются данные приложений. Она состоит из ядра, хранилищ, механизмов безопасности и сетевого взаимодействия.

# **25 Билет**

### **1. Тестовые вопросы:**

1. Потенциальный объем ВВП измеряется как:  
    **б) стоимость товаров и услуг, которые могут быть произведены, если экономика функционирует в условиях полной занятости**  
    Это максимальный возможный ВВП при эффективном использовании всех ресурсов.
2. Длительным периодом называется период, в котором:  
    **б) все факторы производства рассматриваются как переменные**  
    В долгосрочном периоде фирма может изменить объёмы всех ресурсов (капитал, труд и т. д.).
3. Разница между конечной стоимостью товара и стоимостью материалов, используемых для его производства — это:  
    **в) добавленная стоимость**  
    Это тот "вклад", который вносится на каждом этапе производства.
4. Наименее ликвидное средство из предложенных:  
    **в) недвижимость**  
    Продать недвижимость быстро за полную стоимость сложно, поэтому она наименее ликвидна.
5. Метод познания, предполагающий разложение целого на части:  
    **г) анализ**  
    Анализ позволяет глубже понять объект, изучая его составные элементы.

### **2. SQL. Основные запросы SQL**

**SQL (Structured Query Language)** — это язык для управления реляционными базами данных. Он позволяет **добавлять, извлекать, обновлять и удалять данные**, а также управлять их структурой.

### **Основные типы запросов SQL:**

#### **1. SELECT** — извлечение данных:

SELECT имя, зарплата FROM сотрудники WHERE отдел = 'IT';

**2. INSERT** — добавление строки:

INSERT INTO сотрудники (имя, должность, зарплата)

VALUES ('Иванов И.И.', 'Инженер', 50000);

**3. UPDATE** — изменение существующих данных:

UPDATE сотрудники SET зарплата = 55000 WHERE имя = 'Иванов И.И.';

#### **4. DELETE** — удаление строк:

DELETE FROM сотрудники WHERE зарплата < 30000;

### **Дополнительные конструкции:**

* ORDER BY — сортировка
* GROUP BY — группировка
* JOIN — объединение таблиц
* WHERE, IN, LIKE — условия
* AS — псевдонимы столбцов

### **Пример JOIN-запроса:**

SELECT заказы.id, клиенты.имя

FROM заказы

JOIN клиенты ON заказы.клиент\_id = клиенты.id;

**Итог:**  
SQL — это ключевой инструмент работы с базами данных. Он универсален, понятен и используется как в бизнесе, так и в науке.

### **3. Модели жизненного цикла информационной системы**

**Жизненный цикл информационной системы (ИС)** — это последовательность этапов, через которые проходит система **от момента зарождения идеи до её утилизации или замены**.

### **Основные модели жизненного цикла:**

#### **1. Каскадная модель (Waterfall)**

* Этапы идут строго друг за другом: анализ → проектирование → разработка → тестирование → внедрение → сопровождение.
* **Плюсы:** чёткость, контроль.
* **Минусы:** слабо гибкая, неустойчива к изменениям.

#### **2. V-модель**

* Расширение каскадной: каждому этапу разработки соответствует этап тестирования.
* **Применяется:** в критичных системах (авиация, медицина).

#### **3. Инкрементальная модель**

* Система создаётся **по частям** (инкрементам), каждая из которых добавляет новую функциональность.
* **Плюсы:** быстрая реализация рабочей версии, гибкость.

#### **4. Спиральная модель**

* Процесс разработки проходит в циклах (итерациях), каждая включает анализ рисков.
* **Плюсы:** отлично подходит для сложных проектов, позволяет гибко вносить изменения.

#### **5. Agile / Scrum**

* Современный гибкий подход, разработка идёт небольшими итерациями (**спринтами**), с регулярной обратной связью от заказчика.
* **Плюсы:** высокая гибкость, ориентация на результат.

### **Типовые стадии жизненного цикла ИС:**

1. **Инициирование проекта**
2. **Анализ требований**
3. **Проектирование (логическое и физическое)**
4. **Разработка (программирование)**
5. **Тестирование и отладка**
6. **Внедрение и обучение пользователей**
7. **Эксплуатация и сопровождение**
8. **Вывод из эксплуатации**

**Итог:**  
Модель жизненного цикла выбирается в зависимости от **масштаба, сложности проекта и требований к гибкости**. Agile популярен для ИТ-стартапов, каскадная — для предсказуемых проектов.