Самостоятельная работа 2.

Инвариантная часть.

Задание 2.1: Визуализация примера для моделей и подходов к организации данных. Для каждой модели и подходу к организации данных предложить соответствующую предметную область и описать взаимоотношения ее объектов.

Иерархическая (англ. hierarchical), конец 1960-х и 1970-е

Предметная область: Образовательное учреждение.

Описание взаимоотношений объектов:

В иерархической модели данных объекты организованы по принципу иерархической структуры, где каждый элемент имеет ровно одного родителя, за исключением корневого элемента, и может иметь несколько дочерних элементов. Эти связи формируются однонаправленно, от родительского элемента к дочерним, образуя древовидную структуру данных. Таким образом, каждый элемент находится в определенном уровне иерархии и связан с другими элементами в соответствии с этой структурой.

Сетевая (англ. network), 1970-е

Предметная область: Система управления проектами.

Описание взаимоотношений объектов:

В сетевой модели данные организованы как набор узлов и ребер. Узлы могут представлять задачи, ресурсы, этапы проекта и прочие сущности. Ребра определяют связи между узлами, такие как зависимости между задачами, выделение ресурсов для выполнения задачи, иерархические отношения между этапами проекта и т.д. Такая модель позволяет гибко описывать сложные структуры проектов и их взаимосвязи.

Реляционная (англ. relational), 1970-е и начало 1980-х

Предметная область: Онлайн-магазин.

Описание взаимоотношений объектов:

В реляционной модели данные организованы в виде таблиц, где каждая таблица представляет собой отдельный тип данных или сущность. Например, таблицы могут содержать информацию о заказах, клиентах, продуктах и т.д. Связи между данными устанавливаются с помощью ключей: первичных и внешних. Например, таблица заказов может иметь внешний ключ, который ссылается на таблицу клиентов, чтобы указать, к какому клиенту относится данный заказ. Такая структура данных обеспечивает эффективное хранение, управление и извлечение информации о клиентах, заказах и других аспектах работы магазина.

Сущность-связь (англ. entity-relationship), 1970-е

Предметная область: Университетская информационная система.

Описание взаимоотношений объектов:

В модели сущность-связь данные представлены в виде сущностей (entity) и их взаимосвязей (relationship). Например, сущности могут включать студентов, преподавателей, курсы и т.д., а связи между ними определяют отношения, такие как учеба

на курсе, преподавание определенного преподавателя, принадлежность студента к определенной группе и т.д. Ключевые атрибуты (attributes) определяют свойства каждой сущности, такие как имя студента или название курса. Эта модель помогает четко определить структуру данных и их взаимосвязи, что облегчает проектирование базы данных для управления информацией в университетской системе.

Расширенная реляционная (англ. extended relational), 1980-е

Предметная область: Корпоративная система управления ресурсами предприятия (ERP).

Описание взаимоотношений объектов:

Расширенная реляционная модель включает дополнительные концепции, такие как наследование, полиморфизм и ассоциации, чтобы расширить возможности модели. Например, помимо базовых таблиц для сущностей, таких как сотрудники и отделы, могут быть введены таблицы для управления правами доступа, аудита и т.д. Также можно использовать наследование для создания более общих сущностей, а затем специализировать их для конкретных целей. Например, можно создать общую сущность "работник", а затем специализировать ее в "менеджер", "разработчик" и т.д. Эта модель позволяет более гибко описывать сложные предметные области и управлять разнообразными аспектами работы предприятия в ERP-системе.

Семантическая (англ. semantic), конец 1970-х и 1980-е

Предметная область: Информационная поисковая система.

Описание взаимоотношений объектов:

В семантической модели данных основное внимание уделяется смыслу данных. Данные организованы в виде онтологий, где определяются сущности и их отношения, а также свойства сущностей. Например, для информационной поисковой системы сущностями могут быть статьи, документы, термины, пользователи и т.д., а отношениями - связь между терминами и статьями, ассоциации между пользователями и документами. Кроме того, в семантической модели могут использоваться онтологии для описания семантических отношений между данными, таких как "является частью", "содержит", "относится к категории" и т.д. Это позволяет эффективно организовывать и обрабатывать данные в информационной системе, учитывая их семантическую природу и контекст использования.

Объектно-ориентированная (англ. object-oriented), конец 1980-х – начало 1990-х

Предметная область: Разработка программного обеспечения.

Описание взаимоотношений объектов:

В объектно-ориентированной модели данных основной упор делается на объекты и их взаимосвязи. Объекты представляют реальные или абстрактные сущности, а классы определяют типы этих объектов и их поведение. Например, для системы управления библиотекой объектами могут быть книги, читатели, библиотекари и т.д., а классами - Book, Reader, Librarian и т.д. Взаимосвязи между объектами определяются через методы и свойства классов. Например, у книги может быть метод "получить информацию о доступности", который возвращает информацию о том, доступна ли книга для выдачи, и свойство "название", содержащее название книги. Эта модель позволяет описывать

программные системы в терминах объектов и их взаимодействия, что облегчает проектирование, разработку и поддержку программного обеспечения.

Объектно-реляционная (англ. object-relational), конец 1980-х – начало 1990-х

Предметная область: Информационная система для управления персоналом.

Описание взаимоотношений объектов:

В объектно-реляционной модели данных объединяются преимущества объектно-ориентированной и реляционной моделей. Данные организованы в виде объектов, каждый из которых соответствует записи в базе данных, а классы определяют структуру этих объектов и их поведение. Например, для системы управления персоналом объектами могут быть сотрудники, отделы, проекты и т.д., а классами - Employee, Department, Project и т.д. В то же время, для хранения данных используются реляционные таблицы, а для установления связей между объектами - внешние ключи. Например, таблица сотрудников может содержать информацию о каждом сотруднике, а таблица проектов - информацию о каждом проекте, а затем через внешний ключ можно установить связь между сотрудниками и проектами, указывая, кто работает над каким проектом. Это позволяет объединить гибкость объектно-ориентированной модели с эффективностью хранения данных реляционной модели, обеспечивая удобство использования и эффективность работы с данными в информационной системе.

Вариативная часть.

Задание 2.2: Заполните таблицу "Преимущества и недостатки моделей данных"

№	Модель данных	Преимущества	Недостатки
1	Иерархическая	1. Простота структуры: Иерархическая модель представляет данные в виде древовидной структуры, что делает её легкой для понимания и использования. 2. Быстрый доступ к данным: Запросы к данным в иерархической модели обычно быстрее из-за её упорядоченной структуры. Это особенно полезно, когда необходимо быстро получать данные из больших наборов. 3. Эффективность при хранении древовидных данных: Иерархическая модель хорошо подходит для хранения данных, которые естественным образом представляются в виде иерархии, таких как организационные структуры или файловые системы.	1. Жёсткость структуры: Одним из основных недостатков иерархической модели является её жёсткая структура. Изменение структуры данных может быть сложным и требует много времени и усилий. 2. Ограничения в отношениях между данными: Иерархическая модель не всегда подходит для моделирования сложных отношений между данными, так как она ориентирована на иерархические связи. 3. Дублирование данных: Повторяющиеся данные могут стать проблемой в иерархической модели, особенно если один и тот же элемент данных используется в нескольких ветвях иерархии.
2	Сетевая	1. Гибкость: Сетевая модель позволяет описывать сложные отношения между данными, такие как множественные связи между записями, что делает её более гибкой по сравнению с иерархической моделью. 2. Эффективность при обработке связей: Эта модель хорошо подходит для приложений, где связи между данными играют важную роль, таких как социальные сети или сети поставщиков и потребителей. 3. Относительная простота добавления новых связей: Добавление новых типов связей или расширение существующих связей может быть сравнительно простым в сетевой модели без необходимости изменения всей структуры данных.	1. Сложность запросов: Поскольку данные в сетевой модели организованы в виде сложной сети связей, запросы к этим данным могут быть сложными и требовать глубокого понимания структуры данных. 2. Зависимость от структуры: Изменение структуры данных может потребовать изменения множества запросов и приложений, использующих эти данные, что может привести к сложностям в управлении и поддержке. 3. Возможность появления циклических зависимостей: Сетевая модель может привести к появлению циклических зависимостей между данными, что может усложнить обработку и анализ данных.
3	Реляционная	1. Простота использования: Реляционная модель представляет	1. Сложность моделирования некоторых типов данных: Для

		данные в виде таблиц, что делает её легкой для понимания и использования как для разработчиков, так и для конечных пользователей. 2. Гибкость структуры: Таблицы в реляционной модели могут быть легко изменены, добавлены или удалены без больших сложностей. Это облегчает адаптацию модели к изменяющимся потребностям бизнеса. 3. Целостность данных: Реляционная модель предоставляет механизмы для обеспечения целостности данных, таких как ограничения, индексы и транзакции, что помогает поддерживать точность и надёжность данных. 4. Мощные языки запросов: SQL (Structured Query Language) является стандартным языком для работы с данными в реляционной модели, предоставляя широкий спектр возможностей для выполнения запросов и анализа данных. 5. Поддержка множественных пользователей: Реляционные СУБД обычно обладают хорошей поддержкой многопользовательского доступа, что позволяет нескольким пользователям одновременно работать с данными без конфликтов.	некоторых типов данных, таких как иерархические или графовые структуры, реляционная модель может быть неэффективной или неудобной в использовании. 2. Избыточность данных: В реляционной модели иногда происходит дублирование данных между таблицами, что может привести к избыточности и усложнению поддержки целостности данных. 3. Производительность при сложных запросах: Некоторые сложные запросы в реляционной модели могут быть медленными из-за необходимости объединения большого количества таблиц или использования сложных операций. 4. Сложность обновления больших объёмов данных: При обновлении больших объёмов данных в реляционной модели может потребоваться значительное время из-за необходимости выполнения множества операций.
4	Сущность-связь	1. Простота визуализации: Модель сущность-связь позволяет легко визуализировать структуру данных с помощью сущностей (таблиц) и их связей, что упрощает понимание модели как разработчикам, так и конечным пользователям. 2. Гибкость в описании отношений: Эта модель позволяет описывать сложные отношения между различными сущностями, включая один к одному, один ко многим и многие ко многим, что делает её эффективным инструментом для моделирования реальных бизнес-сценариев.	1. Не всегда эффективна для сложных структур данных: Для некоторых типов данных, особенно связанных с графовыми или иерархическими структурами, модель сущность-связь может быть неэффективной или неудобной в использовании. 2. Сложность в описании сложных отношений: В некоторых случаях описание сложных отношений между сущностями может быть сложным и запутанным, особенно когда требуется учесть множество условий или вариантов.

		3. Наглядность при проектировании баз данных: Модель сущность-связь помогает разработчикам лучше понять требования к базе данных и проектировать её таким образом, чтобы она соответствовала бизнес-логике и потребностям приложения.	3. Ограничения визуализации при больших объёмах данных: При работе с большими объёмами данных модель сущность-связь может стать сложной для визуализации и анализа, особенно если сущностей и связей становится слишком много.
5	Расширенная реляционная	1. Поддержка объектов и типов данных: Расширенная реляционная модель обычно включает поддержку более разнообразных типов данных, включая геометрические, графовые, текстовые и другие специализированные типы данных. 2. Способность описывать более сложные отношения: Эта модель позволяет описывать более сложные типы отношений между данными, такие как множественные значения атрибутов, наследование и полиморфизм. 3. Поддержка расширенных функций и операторов: Расширенная реляционная модель может включать более широкий набор функций и операторов для работы с данными, таких как аналитические функции, операции с геоданными и т.д. 4. Интеграция с другими типами данных: Эта модель может включать возможности для интеграции с другими системами и форматами данных, такими как XML, JSON и другими структурированными форматами.	1. Сложность проектирования и использования: Дополнительные возможности и концепции могут увеличить сложность проектирования и использования базы данных, особенно для разработчиков, не знакомых с расширенными концепциями. 2. Требования к производительности: Некоторые расширенные функции и операции могут иметь более высокие требования к производительности, что может потребовать более мощного оборудования или оптимизации запросов. 3. Необходимость дополнительного обучения: Использование расширенных возможностей реляционной модели может потребовать дополнительного обучения для разработчиков и администраторов баз данных. 4. Ограничения совместимости: Некоторые расширенные функции могут быть специфичны для определённых СУБД и не поддерживаться другими системами, что может создавать проблемы с переносимостью кода.
6	Семантическая	1. Более глубокое понимание данных: Семантическая модель позволяет описывать данные с более высоким уровнем абстракции, что способствует лучшему пониманию их значения и контекста. 2. Большая гибкость при моделировании данных: Семантическая модель не привязана к определенной структуре данных, что делает её более	1. Сложность в разработке и понимании: Построение семантической модели требует глубокого понимания данных и их отношений, что может быть сложно для неподготовленных пользователей. 2. Требования к обучению и навыкам: Использование семантической модели требует

		гибкой и адаптируемой к различным видам данных и их изменениям. 3. Повышение интероперабельности: Описание семантической модели в терминах значений и отношений позволяет лучше интегрировать данные из различных источников и обеспечивает более эффективное взаимодействие между системами. 4. Поддержка автоматической обработки и анализа данных: Семантическая модель способствует автоматизации обработки данных и проведению анализа, так как позволяет компьютерам лучше понимать смысл информации.	определенных навыков и знаний, как в области данных, так и в области семантики и онтологий. 3. Возможные проблемы с масштабируемостью: При работе с большими объемами данных и сложными отношениями между ними семантическая модель может столкнуться с проблемами масштабируемости. 4. Неоднозначность интерпретации: Интерпретация семантической модели может быть различной для разных пользователей или систем, что может приводить к неоднозначности в понимании данных.
7	Объектно-ориентированная	1. Модульность и повторное использование кода: Объектно-ориентированное программирование (ООП) способствует созданию модульного кода, который легко поддерживать и модифицировать. Это позволяет повторно использовать классы и объекты в различных частях программы или даже в разных проектах. 2. Упрощение разработки и сопровождения: ООП позволяет абстрагироваться от сложных деталей реализации, фокусируясь на моделировании объектов и их взаимодействии. Это упрощает разработку нового кода и облегчает поддержку и сопровождение существующего. 3. Использование наследования и полиморфизма: Наследование позволяет создавать иерархии классов, что способствует повторному использованию кода и организации его в логические группы. Полиморфизм позволяет обрабатывать объекты различных типов с использованием общего интерфейса. 4. Улучшенная абстракция и моделирование реального мира: ООП позволяет создавать модели, более	1. Перегрузка понятий и сложность: Для непривычных пользователей объектно-ориентированный подход может показаться сложным из-за большого количества понятий, таких как наследование, полиморфизм, инкапсуляция и другие. 2. Потребление ресурсов: Использование объектно-ориентированного подхода может требовать больше памяти и процессорного времени из-за дополнительных накладных расходов, связанных с созданием объектов и управлением ими. 3. Сложность тестирования и отладки: При использовании ООП тестирование и отладка могут быть сложными из-за сложных взаимосвязей между объектами и классами, а также из-за возможных проблем с наследованием и полиморфизмом. 4. Возможность возникновения проблем при проектировании: Неправильное использование ООП или недостаточное понимание концепций может привести к плохому проектированию

		точно отражающие структуру и поведение объектов в реальном мире, что делает код более понятным и легко поддерживаемым.	программы, что может привести к проблемам с производительностью, поддержкой и расширяемостью.
8	Объектно-реляционная	1. Упрощение разработки: ORM позволяет разработчикам использовать объектно-ориентированный подход к программированию при работе с реляционными базами данных, что упрощает процесс разработки. 2. Более высокий уровень абстракции: ORM скрывает сложности написания SQL-запросов, предоставляя более высокоуровневый интерфейс для работы с данными, что делает код более понятным и легко поддерживаемым. 3. Уменьшение времени разработки: ORM позволяет быстрее создавать приложения, так как уменьшает необходимость вручную писать SQL-запросы и маппинг объектов на таблицы базы данных. 4. Повышение переносимости кода: Используя ORM, можно легко переключаться между различными системами управления базами данных без изменения прикладного кода, так как ORM обеспечивает абстракцию от конкретной СУБД.	1. Производительность: Использование ORM может привести к снижению производительности в сравнении с написанием ручных SQL-запросов, особенно в случае сложных запросов или больших объемов данных. 2. Ограничения ORM: ORM может иметь ограничения в возможностях работы с базой данных, что может привести к тому, что определенные функции или оптимизации недоступны. 3. Сложность отладки и оптимизации: При использовании ORM отладка и оптимизация запросов могут быть сложными из-за того, что ORM скрывает детали внутреннего выполнения SQL-запросов. 4. Недостаточная гибкость: В некоторых случаях ORM может оказаться недостаточно гибким для реализации сложных бизнес-логик или отображения сложных структур данных.