**Проектирование информационных систем**

1. Атрибуты и типы атрибутов. Способы отображения атрибутов в диаграммах Чена и IDEF1Х. Понятие доменов атрибутов. Требования, предъявляемые для проектирования доменов на разных этапах проектирования БД.

**Определение атрибутов.**

Как правило, атрибуты указываются только для сущностей. Если у связи имеются атрибуты, то это указывает на тот факт, что связь является сущностью. Существенно помочь в определении атрибутов могут различные бумажные и электронные формы и документы, используемые в организации при решении задачи. Это могут быть формы, содержащие как исходную, так и результаты обработки данных (например, «Форма № 1»).

Выявленные атрибуты могут быть следующих видов:

· простой (атомарный, неделимый) – состоит из одного компонента с независимым существованием (например, «должность работника», «зарплата», «норма непогашенного ускорения», «радиус кривой» и т. д.);

· составной (псевдоатомарный) – состоит из нескольких компонентов (например, «ФИО», «адрес», и т. д.). Степень атомарности атрибутов, закладываемая в модель, определяется разработчиком. Если от системы не требуется выборки всех клиентов с фамилией Иванов или проживающих на улице Комсомольской, то составные атрибуты можно не разбивать на атомарные;

· однозначный – содержит только одно значение для одного экземпляра сущности (например, у кривой в плане может быть только одно значение радиуса, угла поворота, возвышения наружного рельса и т. д.);

· многозначный – содержит несколько значений (например, у одного отделения компании может быть несколько контактных телефонов);

· производный (вычисляемый) – значение атрибута может быть определено по значениям других атрибутов (например, «возраст» может быть определен по «дате рождения» и текущей дате, установленной на компьютере);

· ключевой – служит для уникальной идентификации экземпляра сущности (входит в состав первичного ключа);

· неключевой (описательный) – не входит в первичный ключ;

· обязательный – при вводе нового экземпляра в сущность или редактировании обязательно указывается допустимое значение атрибута, т. е. оно после редактирования не может быть неопределенным (NOT NULL).

После определения атрибутов задаются их домены (области допустимых значений), например:

· наименование участка – набор из букв русского алфавита длиной не более 60 символов;

· поворот кривой – допустимые значения «Л» (влево) и «П» (вправо);

· радиус кривой – положительное число не более 4 цифр.

Задание доменов определяет набор допустимых значений для атрибута (нескольких атрибутов), а также тип, размер и формат атрибута (атрибутов).

На основании выделенного множества атрибутов для сущности определяется набор ключей. Ключ – один или несколько атрибутов сущности, служащих для однозначной идентификации ее экземпляров или для их быстрого поиска. Выделяют следующие типы ключей:

· суперключ (superkey) – атрибут или множество атрибутов, которое единственным образом идентифицирует экземпляр сущности. Суперключ может содержать «лишние» атрибуты, которые необязательны для уникальной идентификации экземпляра. При правильном проектировании структуры БД суперключом в каждой сущности (таблице) будет являться полный набор ее атрибутов;

· потенциальный ключ (potential key) – суперключ, который не содержит подмножества, также являющегося суперключом данной сущности, т. е. суперключ, содержащий минимально необходимый набор атрибутов, единственным образом идентифицирующих экземпляр сущности. Сущность может иметь несколько потенциальных ключей. Если ключ состоит из нескольких атрибутов, то он называется составным ключом. Среди всего множества потенциальных ключей для однозначной идентификации экземпляров выбирают один, так называемый первичный ключ, используемый в дальнейшем для установления связей с другими сущностями;

· первичный ключ (primary key) – потенциальный ключ, который выбран для уникальной идентификации экземпляров внутри сущности;

· альтернативные ключи (alternative key) – потенциальные ключи, которые не выбраны в качестве первичного ключа.

Если потенциальных ключей несколько, то для выбора первичного ключа рекомендуется придерживаться следующих правил:

· количество атрибутов, входящих в ключ, должно быть минимальным (желательно, чтобы ключ был атомарным, т. е. состоял из одного атрибута);

· размер ключа в байтах должен быть как можно короче;

· тип домена ключа – числовой. При выборе символьных атрибутов в ключ часто возникают проблемы с вводом ошибочных значений (путают регистр букв; добавляют лишние пробелы; используют буквы, пишущиеся на разных языках одинаково). В числовых атрибутах вероятность ошибки при вводе значения меньше;

· вероятность изменения значений ключа была наименьшей

· с ключом проще всего работать пользователям

В нотации IDEF1X атрибуты изображаются в виде списка имен внутри блока сущности. Атрибуты, определяющие первичный ключ, размещаются наверху списка и отделяются от других атрибутов горизонтальной чертой..

 В стандарте IDEF0 функциональная модель описывает моделируемый бизнес-процесс с качественной точки зрения. Чтобы получить количественные показатели, пользователь должен расширить модель набором дополнительных атрибутов, определить значения атрибутов для модели и ее отдельных функциональных блоков. Атрибуты используются как для задания исходных данных, так и для получения выходных расчетных данных. Набор дополнительных атрибутов расширяет функциональную модель и позволяет пользователю провести количественный анализ как по отдельным функциональным блокам, так и по модели в целом.

По области определения атрибуты делятся на атрибуты модели, общие атрибуты и локальные атрибуты.

**При проектировании и эксплуатации БД к ней предъявляются следующие требования:**

* Адекватность отображения ПО (полнота, целостность, непро­тиворечивость, актуальность данных).
* Возможность взаимодействия пользователей разных категорий; обеспечение высокой эффективности доступа.
* Дружественность интерфейса.
* Обеспечение секретности и конфиденциальности.
* Обеспечение взаимной независимости программ и данных.
* Обеспечение надежности БД; защита данных от случайного и преднамеренного разрушения; возможность быстрого и полного восстановления данных в случае сбоев в системе.
1. Базовый принцип структурного метода проектирования. Понятия технологии и методов проектирования ИС. Требования, предъявляемые к современным технологиям проектирования ИС.

В качестве двух базовых принципов используются следующие: принцип "разделяй и властвуй" и принцип иерархического упорядочивания. Первый является принципом решения трудных проблем путем разбиения их на множество меньших независимых задач, легких для понимания и решения. Второй принцип в дополнение к тому, что легче понимать проблему, если она разбита на части, декларирует, что устройство этих частей также существенно для понимания. Понимание проблемы резко повышается при организации ее частей в древовидные иерархические структуры, т.е. система может быть понята и построена по уровням, каждый из которых добавляет новые детали.

Выделение двух базовых принципов инженерии программного обеспечения вовсе не означает, что остальные принципы являются второстепенными, игнорирование любого из них может привести к непредсказуемым последствиям (в том числе и к неуспеху всего проекта). Отметим основные из таких принципов.

1. *Принцип абстрагирования* - заключается в выделении существенных аспектов системы с целью представления системы в простом общем виде.

2. *Принцип формализации* - заключается в необходимости строгого мето­дического подхода к решению проблемы.

3. *Принцип инкапсуляции* - заключается в упрятывании несущественной на конкретном этапе информации.

4. *Принцип концептуальной общности* - заключается в следовании единой философии (структурный анализ - структурное проектирование - структурное программирование - структурное тестирование).

5. *Принцип полноты* - заключается в контроле на присутствие лишних элементов.

6. *Принцип непротиворечивости* - заключается в обоснованности и согла­сованности элементов.

7. *Принцип логической* *независимости* - обеспечение независимости логического проектирования от физического проектирования.

8. *Принцип независимости данных* - модели данных должны быть проанализированы и спроектированы независимо от процессов их логической обработки.

9. *Принцип структурирования данных* - заключается в том, что данные должны быть структурированы и иерархически организованы.

10. *Принцип доступа конечного пользователя* - пользователь должен иметь средства доступа к базе данных, которые он может использовать непосредственно

Осуществление проектирования ИС предполагает использование проектировщиками определенной технологии проектирования, соответствующей масштабу и особенностям разрабатываемого проекта. Определение. Технология проектирования – это совокупность концептуальных методов и средств (методологий) проектирования ИС, а также методов и средств организации проектирования, то есть управления процессом создания или модернизации проекта информационной системы.
Требования к технологии проектирования Технология проектирования предполагает возможность выбора различных методов и средств проектирования. Выбор оптимальной совокупности следует осуществлять с учетом следующих требований к технологии проектирования: Обеспечение создания ИС, отвечающей целям и задачам организации, а также предъявляемым требованиям по автоматизации производственных процессов заказчика; Гарантированное создание системы с заданным качеством в заданные сроки и в рамках установленного бюджета проекта (с минимизацией трудовых и стоимостных затрат); Поддержка удобной дисциплины сопровождения, модификации и наращивания системы; Обеспечение преемственности разработки, т.е. использование в разрабатываемой ИС существующей информационной инфраструктуры организации (задела в области информационных технологий); Обеспечение роста производительности труда проектировщика при использовании выбранной технологии; Обеспечение надежности процесса проектирования и эксплуатации проекта; Простота ведения проектной документации.

1. Задачи анализа транзакций на этапе логического проектирования и правила его проведения на примере одной транзакции.

Цель логического проектирования – развить концептуальное представление БД с учетом принимаемой модели БД (иерархической, сетевой, реляционной и т. д.).

Примем в качестве модели реляционную БД в третьей нормальной форме (набор нормализованных отношений с кратностью связей 1:N). Поэтому необходимо будет проверить концептуальную модель с помощью методов нормализации и контроля выполнения транзакций [1, 20, 21]. Транзакция – одно действие или их последовательность, выполняемых как единое целое одним или несколькими пользователями (прикладными программами) с целью осуществления доступа к БД и изменению ее содержимого.

1. Удаление и проверка элементов, не отвечающих принятой модели данных.

1.1. Удаление связей N:M.

Если в концептуальной модели присутствуют связи N:M, то их следует устранить путем определения промежуточной сущности. Связь N:M заменяется двумя связями типа 1:M, устанавливаемыми со вновь созданной сущностью.

 1.2. Удаление связей с атрибутами.

Связи с атрибутами должны быть преобразованы в сущности.

В разработанной концептуальной модели существовала связь N:M («Раздельные пункты» : «Пути»), которая имела собственные атрибуты. После устранения связь N:M, ее атрибуты перешли в сущность «Раздельные пункты на пути»

В графической нотации IDEF1X не предусмотрено отображение связей с атрибутами, хотя некоторые методологии ERD допускают их наличие и отображение.

1.3. Удаление сложных связей (со степенью участия более 2).

Сложную связь заменяют необходимым количеством бинарных связей 1:N со вновь созданной сущностью, которая и показывает эту связь.

1.4. Удаление рекурсивных связей (со степенью участия 1).

Рекурсивную связь заменяют, определив дополнительную сущность и необходимое количество связей

1.5. Удаление многозначных атрибутов (атрибутов имеющих несколько значений).

Многозначность устраняется путем введения новой сущности и связи 1:N

Данное преобразование, помимо соответствия реляционной модели данных, также позволяет хранить любое количество телефонов по одному филиалу.

 1.6. Удаление избыточных связей.

Связь является избыточной, если одна и та же информация может быть получена не только через нее, но и с помощью другой связи

В приведенном примере одну из связей «Руководит» можно смело удалить (лучше между «Руководителем филиала» и «Сотрудником»).

1.7. Перепроверка связей 1:1.

2. Проверка модели с помощью правил нормализации.

3. Проверка выполнения транзакций.

4. Определение требований поддержки целостности данных.

1. Задачи анализа транзакций на этапе физического проектирования и правила его проведения на примере одной транзакции.

Физическое проектирование базы данных - процесс подготовки описания реализации базы данных на вторичных запоминающих устройствах; на этом этапе рассматриваются основные отношения, организация файлов и индексов, предназначенных для обеспечения эффективного доступа к данным, а также все связанные с этим ограничения целостности и средства защиты.

 Физическое проектирование является третьим и последним этапом создания проекта базы данных, при выполнении которого проектировщик принимает решения о способах реализации разрабатываемой базы данных. Во время предыдущего этапа проектирования была определена логическая структура базы данных (которая описывает отношения и ограничения в рассматриваемой прикладной области). Хотя эта структура не зависит от конкретной целевой СУБД, она создается с учетом выбранной модели хранения данных, например реляционной, сетевой или иерархической. Однако, приступая к физическому проектированию базы данных, прежде всего необходимо выбрать конкретную целевую СУБД. Поэтому физическое проектирование неразрывно связано с конкретной СУБД. Между логическим и физическим проектированием существует постоянная обратная связь, так как решения, принимаемые на этапе физического проектирования с целью повышения производительности системы, способны повлиять на структуру логической модели данных.

 Как правило, основной целью физического проектирования базы данных является описание способа физической реализации логического проекта базы данных.

Проектирование физического представления базы данных

Определение оптимальной файловой структуры для хранения базовых отношений и индексов, необходимых для достижения приемлемой производительности. Иными словами, определение способа хранения отношений и кортежей во вторичной памяти.

Анализ транзакций

Определение функциональных характеристик транзакций, которые будут выполняться в проектируемой базе данных, и выделение наиболее важных из них.

Выбор файловой структуры

Определение наиболее эффективной файловой структуры для каждого базового отношения.

Выбор индексов

Определение того, будет ли добавление индексов способствовать повышению производительности системы.

Определение требований к дисковому пространству

Оценка объема дискового пространства, необходимого для размещения базы данных.

1. Макет пользовательского интерфейса. Анализ макета на этапе логического проектирования.

Разработчики интерфейсов стремятся к отображению реального мира , а по некоторым свойствам даже превосходят возможности реальности . Сказанное можно отнести прежде всего к воспроизведению и моделированию процесса во времени .

Следует отметить , что многое в содержании этапов предполагает имено обоснованный выбор свойств , непосредственно участвующих в деятельности пользователя . При таких возможностях от многого приходится скорее отказываться , чем включать в интерфейс . В связи с этим приходит в голову известная оппозиция недопрограммированных и перепрограммированных компьютерных систем , каждая из которых обладает своими недостатками . Задача же проектировщика интерфейса часто состоит в поиске того оптимума , который позволил бы освободиться от недостатков обоих родов . Понятно , что освободиться от недостатков пользовательский интерфейс может лишь в том случае , если процесс его разработки и настройки превратится в технологию и будет оснащен собственными средствами автоматизации . Это может означать , что традиционная архитектура пользовательского интерфейса дополнится новым слоем . Общепринято , нынешняя его архитектура содержит четыре слоя программного обеспечения (11): 1) графический интерфейс для управления аппаратурой дисплея ; 2) инструментальные средства (toolkits) для построения стандартных компонентов интерфейса , которые называют кубиками (widgets); 3) сам набор кубиков ; 4) программа интерфейса высокого уровня для процедур с операционной средой

|  |  |
| --- | --- |
| Принцип | Описание |
| Учет знаний пользователя | В интерфейсе необходимо использовать термины и понятия, взятые из опыта будущих пользователей системы |
| Согласованность | Интерфейс должен быть согласованным в том смысле, что однотипные (но различные) операции должны выполняться одним и тем же способом |
| Минимум неожиданностей | Поведение системы должно быть прогнозируемым |
| Способность к восстановлению | Интерфейс должен иметь средства, позволяющие пользователям восстановить данные после ошибочных действий |
| Руководство пользователя | Интерфейс должен предоставлять необходимую информацию в случае ошибок пользователя и поддерживать средства контекстно-зависимой справки |
| Учет разнородности пользователей | В интерфейсе должны быть средства для удобного взаимодействия с пользователями, имеющими разный уровень квалификации и различные возможности |

1. Нежелательные элементы при проведении анализа на этапе логического проектирования.

Временные оценки СУБД наиболее популярных тестов последнее время даются в виде числа транзакций определенного стандартизованного вида в единицу времени. Распределенная обработка строится на основе мониторов транзакций.

Нужно будет обнаруживать пределы возможностей такого деления работ на достаточно мелкие порции. Здесь отметим очень важный эффект: практика ориентации на "транзакционный подход" тесно связана с классической методологией проектирования БД, которая развивалась, в основном, как методология проектирования так называемы "операционных" БД, то есть баз данных, которые должны фиксировать отдельные совершаемые операции и хранить модель текущего фактического состояния объекта или ПрО.

Логическое проектирование базы данных. Процесс создания модели используемой на предприятии информации на основе выбранной модели организации данных, но без учета типа целевой СУБД и других физических аспектов реализации. В процессе разработки логическая модель данных постоянно тестируется и проверяется на соответствие требованиям пользователей. Для проверки правильносги логической модели данных используется метод нормализации. Нормализация гарантирует, что отношения, выведенные из существующей модели данных, не будут обладать избыточностью данных, способной вызвать нарушения в процессе обновления данных после их физической реализации. Помимо всего прочего, логическая модель данных должна обеспечивать поддержку всех необходимых пользователям транзакций. Созданная логическая модель данных является источником информации для этапа физического проектирования и обеспечивает разработчика физической базы данных средствами поиска компромиссов, необходимых для достижения поставленных целей, что очень важно для эффективного проектирования. Логическая модель данных играет также важную роль на этапе эксплуатации и сопровождения уже готовой системы. При правильно организованном сопровождении поддерживаемая в актуальном состоянии модель данных позволяет точно и наглядно представить любые вносимые в базу данных изменения, а также оценить их влияние на прикладные программы и использование данных, уже имеющихся в базе.

1. Основные подходы к проектированию ИС. Основные принципы структурного и объектно-ориентированного методов проектирования.

Процесс проектирования сложного программного обеспечения начинают с уточнения его структуры, т. е. определения структурных компонентов и связей между ними. Результат уточнения структуры может быть представлен в виде структурной и/или функциональной схем и описания (спецификаций) компонентов.

Структурная схема разрабатываемого программного обеспечения. Структурной называют схему, отражающую состав и взаимодействие по управлению частей разрабатываемого программного обеспечения.

Самый простой вид программного обеспечения - программа, которая в качестве структурных компонентов может включать только подпрограммы и библиотеки ресурсов. Разработку структурной схемы программы обычно выполняют методом пошаговой детализации.

Структурными компонентами программной системы или программного комплекса могут служить программы, подсистемы, базы данных, библиотеки ресурсов и т. п.

Структурная схема программной системы, как правило, показывает наличие подсистем или других структурных компонентов. В отличие от программного комплекса отдельные части (подсистемы) программной системы интенсивно обмениваются данными между собой и, возможно, с основной программой. Структурная же схема программной системы этого обычно не показывает.

Более полное представление о проектируемом программном обеспечении с точки зрения взаимодействия его компонентов между собой и с внешней средой дает функциональная схема.

1. Перечень элементов и их назначение для создания пользовательского интерфейса.

Интерфейс пользователя - эта та часть программы, которая находится у всех на виду. Некоторые программисты склонны оставлять дизайн интерфейса пользователя на потом, считая, что реальное достоинство приложения - его программный код, который и требует большего внимания. Однако часто возникает недовольство пользователей из-за неудачно подобранных шрифтов, непонятного содержимого экрана и скорости его прорисовывания, поэтому работу над интерфейсом также нужно воспринимать серьезно. Проектирование форм ввода данных Особый вид форм - формы, предназначенные для ввода данных. Они позволяют пользователю идти в нужном ему темпе, не оглядываясь на программиста. Работа с несколькими формами Если интерфейс пользователя должен содержать несколько форм, вам предстоит принять самое важное решение: какой использовать вид интерфейса - однодокументный или многодокументный.Эффективные меню Еще одна важная часть разработки форм - создание содержательных и эффективных меню. Приведем некоторые важные рекомендации:• Следуйте стандартным соглашениям о расположении пунктов меню принятым • Группируйте пункты меню в логическом порядке и по содержанию.• Для группировки пунктов в раскрывающихся меню используйте разделительные линии• Избегайте избыточных меню.• Избегайте пунктов меню верхнего уровня, не содержащих раскрывающихся меню•

**Ощущение скорости**

Ощущение - это реальность. Здесь мы попытаемся объяснить, как ощущения пользователя могут повлиять на то, понравится ему ваша программа или нет. Простой пример - скорость работы приложения. У вас может быть самый быстродействующий программный код, но это ничего не значит, если с точки зрения пользователя он работает медленно. Когда пользователи жалуется на скорость, программисты защищаются, утверждая, что "пользователь не знает, что делает программа". Однако, если воспользоваться некоторыми уловками, то можно сделать та что будет казаться, будто программа работает быстрее.

Пользователь гораздо более расположен к ожиданию, если считает, что компьютер работает с максимальной скоростью. Хороший пример- загрузка Windows, которая обычно требует достаточно много времени. Однако вывод графики, сопровождающие звуки, шум жесткого диска отвлекают настолько, что пользователь не ощущает ожидания. Описанная ниже техника поможет в создании "более быстрых" приложений.

 **Информируйте пользователя о ходе процесса**

Когда есть видимость работы приложения, пользователи более легко переносят длительное ожидание в работе программы. Один из способов информирования пользователя о ходе выполнения работы - использовать в форме индикатор процесса. Если вы обновляете записи базы данных, можно использовать такой индикатор для отображения числа записей, над которыми операция уже произведена. Для этого добавьте пару строк кода обновляющих показания индикатора по мере перехода к следующим записям.

**Выводы** по проектированию пользовательского интерфейса

Хотя ни одно ухищрение не гарантирует создания удачного интерфейса пользователя, плохой интерфейс гарантирует отсутствие пользователей вашей программы. Однако при стремительном появлении новшеств, в сфере пользовательских интерфейсов, понятие "хорошего" интерфейса очень быстро изменяется. Возьмем, например, процесс настройки часов видеомагнитофона. Раньше часы программировали "вслепую" кнопками и переключателями, потом стали применяться дисплеи, для которых использовался экран телевизора, а теперь в некоторых моделях этот процесс выполняется автоматически по радиосигналу. Так и интерфейс пользователя программ будет "эволюционировать" по мере того как индустрия будет устанавливать новые стандарты, и вы, в свою очередь, должны быть всегда в курсе, как в наибольшей степени удовлетворить ожидания пользователя.

1. Показатель кардинальности. Правило нахождения и особенности связи с показателем кардинальности 1:1 Отражение связи с показателем кардинальности 1:1 в среде Erwin.

Для того чтобы указать количество возможных связей для каждого экземпляра участвующего в связи сущности, используют *показатель кардинальности.*

Для бинарных связей показатель кардинальности может иметь следующие значения:

*"один к одному"* (1 : 1), *"один ко многим"*(1 : N*), "многие ко многим"* (М : N).

Если максимальная мощность связи в обоих направлениях равна одному, мы называем ее связью *"один к одному"* (1 : 1).

Например, на факультете может быть один декан, и обратно, один и тот же декан может руководить только одним факультетом, что может быть обозначено следующим образом:

**ФАКУЛЬТЕТ <-----> ДЕКАН.**

Если максимальная мощность в одном направлении равна одному, а в другом — многим, то связь называется *"один ко многим"* (1 : N).

Например, в группе учится много студентов, но каждый студент учится только в одной группе:

**ГРУППА <----->> СТУДЕНТ**.

Количество классов сущностей, охваченных данной связью, выражает степень связи. Связь со степенью два наиболее распространена и называется бинарной. Например, в вышеприведенном примере - бинарная связь. Такие связи между сущностями наиболее распространены. многие ко многим (N : M). В этом случае связь типа N : M (многие ко многим). Правила, определяющие показатели кардинальности связей, называют бизнес-правилами организации. Они устанавливаются для определенной организации исходя из особенностей её функционирования.. Бизнес – правила организации являются обязательными для реализации в создаваемой БД. Имеется два варианта участия сущности в связи: полное и частичное. Степень участия является полной, если для существования некоторой сущности требуется существования другой сущности. На ER - диаграмме участник связи с полным участием соединяется со значком связи двойной линией, а участник связи с частичным участием – одинарной линией.

1. Показатель кардинальности. Правило нахождения и особенности связи с показателем кардинальности 1:м. Отражение связи с показателем кардинальности 1:м в среде Erwin.

Для нахождения показателя кардинальности необходимо использовать следующее правило, состоящее из трех шагов: Выделяется один экземпляр первой сущности и проверяется, со сколькими экземплярами второй сущности он может вступать в связь. При этом не учитывается временной фактор, другими словами - со сколькими экземплярами второй сущности он может вступать в связь в любое время. Выделяется один экземпляр второй сущности и проверяется, со сколькими экземплярами первой сущности он может вступать в связь в любое время. Результаты первого и второго шага сравниваются, затем выбирается показатель кардинальности данной связи:



Показатель кардинальности зависит от бизнес-правил описываемой организации, а также от набора атрибутов сущностей, вступающих в эту связь. Степень участия определяет, зависит ли существование некоторой сущности от участия в этой связи другой сущности. Для определения степени участия необходимо задать вопрос все ли экземпляры первой сущности принимают участие в заявленной связи? Если ответ - да, то степень участия первой сущности в этой связи – полная, если ответ - нет, то степень участия первой сущности в этой связи – частичная.


1. Понятие жизненного цикла ИС. Понятие модели жизненного цикла ИС. Типы моделей ЖЦ ИС. Особенности, преимущества, недостатки

Жизненный цикл программного обеспечения представляет собой модель его создания и использования. Модель отражает его различные состояния, начиная с момента возникновения необходимости в данном ПО и заканчивая моментом его полного выхода из употребления у всех пользователей.

Известны следующие модели жизненного цикла:

* Каскадная модель. Переход на следующий этап означает полное завершение работ на предыдущем этапе.
* Поэтапная модель с промежуточным контролем. Разработка ПО ведется итерациями с циклами обратной связи между этапами. Межэтапные корректировки позволяют уменьшить трудоемкость процесса разработки по сравнению с каскадной моделью; время жизни каждого из этапов растягивается на весь период разработки.
* Спиральная модель. Особое внимание уделяется начальным этапам разработки - выработке стратегии, анализу и проектированию, где реализуемость тех или иных технических решений проверяется и обосновывается посредством создания прототипов (макетирования). Каждый виток спирали предполагает создание некой версии продукта или какого-либо его компонента, при этом уточняются характеристики и цели проекта, определяется его качество и планируются работы следующего витка спирали.

Двумя классическими результатами анализа являются:

* иерархия функций, которая разбивает процесс обработки на составные части (что делается и из чего это состоит);
* модель "сущность-связь" (Entry Relationship model, ER-модель), которая описывает сущности, их атрибуты и связи (отношения) между ними.

Эти результаты являются необходимыми, но не достаточными. К достаточным результатам следует отнести диаграммы потоков данных и диаграммы жизненных циклов сущностей. Довольно часто ошибки анализа возникают при попытке показать жизненный цикл сущности на диаграмме ER.

1. Понятие и классификация CASE-средств. Особенности CASE-средства Erwin.

CASE-технология представляет собой методологию проектирования ИС, а также набор инструментальных средств, позволяющих в наглядной форме моделировать предметную область, анализировать эту модель на всех этапах разработки и сопровождения ИС и разрабатывать приложения в соответствии с информационными потребностями пользователей. Большинство существующих CASE-средств основано на методологиях структурного (в основном) или объектно-ориентированного анализа и проектирования, использующих спецификации в виде диаграмм или текстов для описания внешних требований, связей между моделями системы, динамики поведения системы и архитектуры программных средств

CASE-средства позволяют создавать не только продукт, практически готовый к применению, но и обеспечить “правильный” процесс его разработки. Основная цель технологии – отделить проектирование программного обеспечения от его кодирования, сборки, тестирования и максимально “скрыть” от будущих пользователей все детали разработки и функционирования ПО. При этом значительно повышается эффективность работы проектировщика: сокращается время разработки, уменьшается число программных ошибок, программные модули можно использовать при следующих разработках.

Большинство CASE-средств основано на парадигме “методология/метод/нотация/структура/средство”.

Методология задает руководящие указания для оценки и выбора проекта разработки ПО, этапы и последовательность работ, правила применения тех или иных методов.

Метод – систематическая процедура или технология генерации описаний компонент ПО (например, описание потоков и структур данных).

Нотации предназначены для описания системы в целом, ее элементов, таких как графы, диаграммы, таблица, блок-схемы, алгоритмы, формальные языки и языки программирования.

Структуры являются средством для реализации структурного анализа и построения структуры конкретной системы.

Средства – технологические и программные инструменты для поддержки и усиления методов.

CASE-технологии обладают следующими основными достоинствами, которые позволяют широко использовать их при разработке информационных систем:

* ускоряют процесс коллективного проектирования и разработки;
* позволяют за короткий срок создать прототип заказанной системы с заданными свойствами;
* освобождают разработчика от рутинной работы, оставляя время для творчества;
* обеспечивают эффективность и качество разрабатываемого ПО за счет автоматизации контроля всего процесса разработки;
* поддерживают сопровождение и развитие системы на высоком уровне.

**Для успешного внедрения CASE-средств** организация должна обладать нижеследующими качествами.

Культура. Готовность к внедрению новых процессов и взаимоотношений между разработчиками и пользователями, ИТ/ИС-управленцами и пользователями.

Управление. Четкое руководство и организованность по отношению к наиболее важным этапам и процессам внедрения.

Технология. Понимание ограниченности существующих возможностей и способность принять новую технологию.

Если организация не обладает хотя бы одним из перечисленных качеств, то внедрение CASE-средств может закончиться неудачей независимо от степени тщательности следования различным рекомендациям по внедрению.

В качестве примеров популярных CASE-средств укажем программные средства компании Computer Associates, IBM-Rational Software и Oracle:

* BPwin – моделирование бизнес-процессов;
* ERwin – моделирование баз данных и хранилищ данных;
* ERwin Examiner – проверка структуры СУБД и моделей, созданных в Erwin;
* ModelMart – среда для командной работы проектировщиков;
* Paradigm Plus – моделирование приложений и генерация объектного кода;
* Rational Rose – моделирование бизнес-процессов и компонентов приложений;
* Rational Suite AnalystStudio – пакет для аналитиков данных;
* Oracle Designer (входит в Oracle9i Developer Suite) – высокофункциональное средство проектирования программных систем и баз данных, реализующее технологию CASE и собственную методологию Oracle – CDM. Позволяет команде разработчиков полностью провести проект, начиная от анализа бизнес-процессов через моделирование к генерации кода и получению прототипа, а в дальнейшем и окончательного продукта. Сложное CASE-средство, его имеет смысл использовать при ориентации на линейку продуктов Oracle.
1. Понятие информационной системы. Требования, предъявляемые к информационной системе. Классификация информационных систем

Информационная система (ИС) — это система, реализующая информационную модель предметной области, чаще всего — какой-либо области человеческой деятельности. ИС должна обеспечивать: получение (ввод или сбор), хранение, поиск, передачу и обработку (преобразование) информации.

Информационной системой (или информационно-вычислительной системой) называют совокупность взаимосвязанных аппаратно-программных средств для автоматизации накопления и обработки информации. В информационную систему данные поступают от источника информации. Эти данные отправляются на хранение либо претерпевают в системе некоторую обработку и затем передаются потребителю.

Между потребителем и собственно информационной системой может быть установлена обратная связь. В этом случае информационная система называется замкнутой. Канал обратной связи необходим, когда нужно учесть реакцию потребителя на полученную информацию.

Информационная система состоит из баз данных, в которых накапливается информация, источника информации, аппаратной части ИС, программной части ИС, потребителя информации.

По мнению одних авторов, ИС включает в себя персонал, её эксплуатирующий, по мнению других — нет.

Состав информационных систем:

·        Данные

·        Информация

·        Знания

·        Базы данных

·        База знаний

·        программное обеспечение

·        экспертные системы

·        локальные сети

·        защита информации

·        информационная безопасность

**Классификация информационных систем по степени автоматизации**

* Ручные информационные системы характеризуются отсутствием современных технических средств переработки информации и выполнением всех операций человеком. Например, о деятельности менеджера в фирме, где отсутствуют компьютеры, можно говорить, что он работает с ручной ИС.
* Автоматизированные информационные системы (АИС) — наиболее популярный класс ИС. Предполагают участие в процессе накопления, обработки информации баз данных, программного обеспечения, людей и технических средств.
* Автоматические информационные системы выполняют все операции по переработке информации без участия человека, различные роботы. Примером автоматических информационных систем являются некоторые поисковые машины Интернет, например Google, где сбор информации о сайтах осуществляется автоматически поисковым роботом и человеческий фактор не влияет на ранжирование результатов поиска.

Обычно термином ИС в наше время называют автоматизированные информационные системы.

**Классификация информационных систем по характеру использования информации**

* Информационно-поисковые системы — система для накопления, обработки, поиска и выдачи интересующей пользователя информации.
* Информационно-аналитические системы — класс информационных систем, предназначенных для аналитической обработки данных с использованием баз знаний и экспертных систем.
* Информационно-решающие системы — системы, осуществляющие накопление, обработку и переработку информации с использованием прикладного программного обеспечения.
	+ управляющие информационные системы с использованием баз данных и прикладных пакетов программ.
	+ советующие экспертные информационные системы, использующие прикладные базы знаний,
* Ситуационные центры (информационно-аналитические комплексы)

**Классификация информационных систем по архитектуре**

* Локальные ИС (работающие на одном электронном устройстве, не взаимодействующем с сервером или другими устройствами)
* Клиент-серверные ИС (работающие в локальной или глобальной сети с единым сервером)
* Распределенные ИС (децентрализованные системы в гетерогенной многосерверной сети)

**Классификация информационных систем по сфере применения**

* Информационные системы организационного управления — обеспечение автоматизации функций управленческого персонала.
* Информационные системы управления техническими процессами — обеспечение управления механизмами, технологическими режимами на автоматизированном производстве.
* Автоматизированные системы научных исследований — программно-аппаратные комплексы, предназначенные для научных исследований и испытаний.
* Информационные системы автоматизированного проектирования — программно-технические системы, предназначенные для выполнения проектных работ с применением математических методов.
* Автоматизированные обучающие системы — комплексы программно-технических, учебно-методической литературы и электронные учебники, обеспечивающих учебную деятельность.
* Интегрированные информационные системы — обеспечение автоматизации большинства функций предприятия.
* Экономическая информационная система — обеспечение автоматизации сбора, хранения, обработки и выдачи необходимой информации, предназначенной для выполнения функций управления.

**Классификация информационных систем по признаку структурированности решаемых задач**

* Модельные информационные системы позволяют установить диалог с моделью в процессе ее исследования (предоставляя при этом недостающую для принятия решения информацию), а также обеспечивает широкий спектр математических, статистических, финансовых и других моделей, использование которых облегчает выработку стратегии и объективную оценку альтернатив решения. Пользователь может получить недостающую ему для принятия решения информацию путем.
* Использование экспертных информационных систем связано с обработкой знаний для выработки и оценки возможных альтернатив принятия решения пользователем. Реализуется на двух уровнях:

*Первый уровень* (концепция «типового набора альтернатив») — сведение проблемных ситуаций к некоторым однородным классам решений. Экспертная поддержка на этом уровне реализуется созданием информационного фонда хранения и анализа типовых альтернатив. *Второй уровень* — генерация альтернативы на основе правил преобразования и процедур оценки синтезированных альтернатив, используя базу имеющихся в информационном фонде данных.

Экспертные системы представляют совокупность фактов, сведений и данных с системой правил логического вывода информации на основании логической модели баз данных и баз знаний. Базы данных содержат совокупность конкретных данных, а базы знаний — совокупность конкретных и обобщенных сведений в рамках логической модели базы знаний.

1. Понятие локальной логической модели данных. Способы создания глобальной логической модели данных.

Логическая модель данных описывает понятия предметной области, их взаимосвязь, а также ограничения на данные, налагаемые предметной областью. Примеры понятий - "сотрудник", "отдел", "проект", "зарплата". Примеры взаимосвязей между понятиями - "сотрудник числится ровно в одном отделе", "сотрудник может выполнять несколько проектов", "над одним проектом может работать несколько сотрудников". Логическая модель данных является начальным прототипом будущей базы данных. Логическая модель строится в терминах информационных единиц, но без привязки к конкретной СУБД. Более того, логическая модель данных необязательно должна быть выражена средствами именно реляционной модели данных. Основным средством разработки логической модели данных в настоящий момент являются различные варианты ER-диаграмм (Entity-Relationship, диаграммы сущность-связь). Одну и ту же ER-модель можно преобразовать как в реляционную модель данных, так и в модель данных для иерархических и сетевых СУБД, или в постреляционную модель данных. Однако, т.к. мы рассматриваем именно реляционные СУБД, то можно считать, что логическая модель данных для нас формулируется в терминах реляционной модели данных.

Ниже описан подход, который можно использовать для выполнения слияния локальных моделей в глобальную и устранения любых возникающих при этом несовместимостей.

1. *Анализ имен и содержимого сущностей/отношений и их потенциальных ключей*

Может оказаться целесообразным предварительно проанализировать имена сущностей, присутствующих в локальных моделях данных; эти сведения можно найти в словаре данных. Проблемы имеют место в следующих случаях:

* + если две или несколько сущностей/отношений имеют одно и то же имя, но на самом деле отличаются друг от друга (проблема омонимов);
	+ если две или несколько сущностей/отношений являются одинаковыми, но имеют разные имена (проблема синонимов).

Для решения этих проблем может потребоваться сравнить содержимое данных каждой сущности/отношения. В частности, обнаружить эквивалентные сущности/отношения с разными именами, которые применяются в тех или иных представлениях, поможет сравнение их потенциальных ключей.

1. *Анализ имен и содержимого связей/внешних ключей*
2. *Слияние сущностей/отношений, соответствующих локальным моделям данных*



1. *Объединение сущностей/отношений с разными именами с использованием одинаковых или разных первичных ключей*
2. *Включение (без слияния) сущностей/отношений, характерных только для отдельных локальных моделей данных*
3. *Слияние связей/внешних ключей из отдельных локальных моделей данных*
4. *Включение (без слияния) связей/внешних ключей, характерных только для отдельных локальных моделей данных*
5. *Проверка того, нет ли пропущенных сущностей/отношений и связей/внешних ключей*
6. *Проверка внешних ключей*
7. *Проверка ограничений целостности*
8. *Формирование глобальной ER-диаграммы/схемы отношений*
9. *Обновление документации*
10. Понятие ограничения целостности. Типы требований по ограничению целостности.Стратегии при ограничении ссылочной целостности. Назначение стратегии в среде Erwin.

**Целостность базы данных** (DATABASE INTEGRITY) — соответствие имеющейся в базе данных информации её внутренней логике, структуре и всем явно заданным правилам. Каждое правило, налагающее некоторое ограничение на возможное состояние базы данных, называется ограничением целостности (integrity constraint).

Примеры ограничений: вес детали должен быть положительным; возраст родителей не может быть меньше возраста их ребёнка и т.д.

Ограничения целостности обладают следующими свойствами:

* Ограничения целостности навязывают правила на уровне таблицы.
* Ограничения целостности предотвращают удаления строк в случаях, если на эти строки в таблицы наложены ограничения.
* Ограничения к таблице могут быть наложены на момент создания таблицы, а также после создания таблицы
* ORACLE автоматически генерирует имена ограниениям при их создании при помощи добавлению SYS\_C уникального числа
* Существует следующие типы ограничений:

v      NOT NULL

v      UNIQUE

v      CHECK

v      PRIMARY KEY

v      FOREIGN KEY

#### В базе существуют связи между таблицами к примеру таблицы DEPT и EMP связаны через столбец deptno. Логически это означает:

#### ·         все сотрудники должны быть распределены по имеющимся отделам

#### ·         невозможно взять сотрудника на работу, но при этом зарегистрировав его на неизвестный отдел

#### ·         можно создать отдел и в нем могут быть пока не зарегистрированые сотрудники

#### ·         при закрытие(удаление) отдела – сперва нужно проверить нет ли сотрудников, привязанных к этому отделу. Если нет то просто удаляем отдел, а если есть то:

#### а) не разрешать удаление этого отдела, пока есть хотя бы один сотрудник привязанный к этому отделу

#### б) удалить отдел и всех сотрудников, которые в нем зарегистрированы

#### в) удалить отдел, при этом сотрудником назначить пустой (null) отдел

#### ·         при увольнение сотрудника – просто удаляем запись про него

#### Надо отметить, что при проектировки таблиц часто одни таблицы по определенным столбцам являются родительскими по отношению одним таблицам и одновременно по другим столбцам являются дочерними по отношению к другим таблицам.

#### FOREIGN KEY это ссылочная ограничение, устанавливается на дочерний таблицы, который ссылается на столбец в родительской таблице.

#### ·         FOREIGN KEY может принимать значение из родительского столбца или NULL

#### ·         В свою очередь родительский столбец должен иметь ограничение PRIMARY KEY или UNIQUE.

Назначать FOREIGN KEY  можно как на уровне таблице, так и на уровне столбца.

1. Понятие пользовательского интерфейса. Типы ПИ. Состав документации по пользовательскому интерфейсу.

**Пользовательский интерфейс** - это программных и аппаратных средств, обеспечивающих взаимодействие пользователя с компьютером. Основу такого взаимодействия составляют диалоги. Под диалогом в данном случае понимают регламентированный обмен информацией между человеком и компьютером, осуществляемый в реальном масштабе времени и направленный на совместное решение конкретной задачи. Каждый диалог состоит из отдельных процессов ввода/вывода, которые физически обеспечивают связь пользователя и компьютера. Обмен информацией осуществляется передачей сообщения.

 В основном пользователь генерирует сообщения следующих типов:

-запрос информации

-запрос помощи

-запрос операции или функции

-ввод или изменение информации

В ответ пользователь получает подсказки или справки; информационные сообщения, требующие ответа; приказы, требующие действия; сообщения об ошибках и другую информацию.

# Типы интерфейсов:

Интерфейсы пользователя бывают двух типов:

**1)**процедурно-ориентированные:

-примитивные

-меню

-со свободной навигацией

**2)**объектно-ориентированные:

-прямого манипулирования.

**Процедурно ориентированный интерфейс** использует традиционную модель взаимодействия с пользователем, основанную на понятиях «процедура» и «операция». В рамках этой модели программное обеспечение предоставляет пользователю возможность выполнения некоторых действий, для которых пользователь определяет соответствие данных и следствием выполнения которых является получение желаемого результата.

**Объектно-ориентированные интерфейсы** используют модель взаимодействия с пользователем, ориентированную на манипулирование объектами предметной области. В рамках этой модели пользователю предоставляется возможность напрямую взаимодействовать с каждым объектом и инициировать выполнение операций, в процессе которых взаимодействуют несколько объектов. Задача пользователя формулируется как целенаправленное изменение некоторого объекта. Объект понимается в широком смысле слова - модель БД, системы и т.д.

**Процедурно-ориентированные интерфейсы:**

**1)**Обеспечивает пользователю функции, необходимые для выполнения задач;

**2)**Акцент делается на задачи;

**3)**Пиктограммы представляют приложения, окна или операции;

**4)**Содержание папок и справочников отражается с помощью таблицы-списка.

**Объектно-ориентированные интерфейсы:**

**1)**Обеспечивает пользователю возможность взаимодействия с объектами;

**2)**Акцент делается на входные данные и результаты;

**3)**Пиктограммы представляют объекты;

**4)**Папки и справочники являются визуальными контейнерами объектов.

**Интерфейс Меню.** В отличие от примитивного интерфейса, позволяет пользователю выбирать операцию из специального списка, выводимого ему программой. Эти интерфейсы предполагают реализацию множества сценариев работы, последовательность действий в которых определяется пользователями. Древовидная организация меню предполагает строго ограниченную реализацию. При этом возможны два варианта организации меню:

-каждое окно меню занимает весь экран

-на экране одновременно присутствуют несколько разноуровневых меню (Windows).

В условиях ограниченной навигации, независимо от варианта реализации, поиск пункта более чем двух уровневого меню оказывается довольно сложной задачей.

**Объектно-ориентированный интерфейс**предполагает, что взаимодействие с пользователем осуществляется посредством выбора и перемещения пиктограмм соответствующей объектно-ориентированной области.

Различают одно-документные (SDI) и многодокументные (MDI) интерфейсы.

1. Понятие пользовательского интерфейса. Типы ПИ Требования, предъявляемые к проектированию пользовательского интерфейса.

**Пользовательский интерфейс** - это программных и аппаратных средств, обеспечивающих взаимодействие пользователя с компьютером. Основу такого взаимодействия составляют диалоги. Под диалогом в данном случае понимают регламентированный обмен информацией между человеком и компьютером, осуществляемый в реальном масштабе времени и направленный на совместное решение конкретной задачи. Каждый диалог состоит из отдельных процессов ввода/вывода, которые физически обеспечивают связь пользователя и компьютера. Обмен информацией осуществляется передачей сообщения.

 В основном пользователь генерирует сообщения следующих типов:

-запрос информации

-запрос помощи

-запрос операции или функции

-ввод или изменение информации

В ответ пользователь получает подсказки или справки; информационные сообщения, требующие ответа; приказы, требующие действия; сообщения об ошибках и другую информацию.

# Типы интерфейсов:

Интерфейсы пользователя бывают двух типов:

**1)**процедурно-ориентированные:

-примитивные

-меню

-со свободной навигацией

**2)**объектно-ориентированные:

-прямого манипулирования.

Главным требованием к пользовательскому интерфейсу коммерческих АБД является его дружественность (friendliness) под которой подразумевается обеспечение удобного и естественного для пользователя способа общения АБД. Степень такой дружественности зависит от ориентации на определенный рыночный сегмент.

Повышают дружественность пользовательского интерфейса такие методы, как:

- естественное наименование команд, сходные с аналогичными наименованиями в других БД (например, команды поиска целесообразно называть командами «find» «search" и т.д. команды выдачи - "print" и т.п.);

- наличие развитой системы подсказок (help);

- удобный порядок выдачи информации.

1. Понятие пользователя, типа пользователя, требования пользователя. Типы пользовательских требований. Методы сбора требований для описания предметной области.

**1. Интервью**
1) Подготовка – планирование процесса опроса и выработка стратегии управления этим процессом. -выбор нужного собеседника; договоренность о встрече; формирование предварительной программы встречи; изучение сопутствующей информации; согласование плана опроса с группой проектирования.
2) Проведение опроса.
3) Завершение. Опрос нужно завершать, если:получен достаточно большой объем информации; поступает большой объем неподходящей информации; информация перестает усваиваться; эксперт начинает уставать; с экспертом возник конфликт.)

**2. Анкетирование** (Анкетирование проводится при условии готовности опрашиваемых к правдивым ответам. Преимущество: наименее затратный способ извлечения информации.Недостаток: наименее эффективный способ сбора данных.В анкетах могут использоваться следующие виды вопросов:Многоальтернативные вопросы. Предполагает множественные ответы на вопросы; может расширяться комментариями респондента в свободной форме. Рейтинговые вопросы. Предполагает использование лингвистических переменных: "абсолютно согласен", "согласен", "отношусь нейтрально", "не согласен", "абсолютно не согласен", "не знаю". Вопросы с ранжированием. Предусматривает ранжирование (упорядочивание) ответов путем присваивания им порядковых номеров, процентных значений и т.п.)

**3. Наблюдение** (Применяется для сбора сведений о параметрах, признаках и объектах в соответствующей предметной области. Важные для изучения параметры, признаки и объекты точно оцениваются сотрудниками и регистрируются в карточках или в формулярах (например, по частоте, количеству, продолжительности, затратам). Различают пассивное и активное наблюдение. При активном наблюдении аналитик работает, как участник команды, что позволяет улучшить понимание процессов.Достоинство: сбор информации, которую невозможно получить путем опроса или изучения документации.Недостаток: наблюдатель «вносит помехи» в результаты измерений.)

**4. Самостоятельное описание требований** (Используется при наличии:хорошо структурированной документации, описывающей устоявшиеся в организации бизнес-процессы;большого опыта разработки ИС в схожих предметных областях.Достоинство: предварительное формирование требований происходит в удобном для аналитика режиме.
Недостаток: возможность пропуска важной информации, связанной с выполнением бизнес-процессов в реальной жизни и не вошедшей в документы. )

**5. Совместные семинары** (Групповое обсуждение проводится проектировщиками совместно с заказчиками, включая пользователей с целью обобщения и обсуждение важных для решения проблем вопросов.Недостаток: одна из наиболее затратныхДостоинство: быстрота принятия решений, снижение количества ошибок, выработка нетривиальных идей.)

**6. Прототипирование** (Программный прототип – это «зеркало», в котором видно отражение того, как понял исполнитель требования заказчика. Прототипирование является ключевым компонентом методологии быстрой разработки приложений (RAD – Rapid Application Development).RAD базируется на следующих принципах:
- Эволюционное прототипирование;
- использование CASE-средств, обладающих возможностями прямого и обратного проектирования и автоматической генерации кода;
- высококвалифицированные специалисты;
- совмещение живого общения с разработкой в режиме online;
- жесткие временные рамки.

1. Понятие предметной области. Способы описания предметной области. Способ выделения сущностей из описания предметной области.

Совокупность реалий (объектов) внешнего мира - объектов, о которых можно задавать вопросы, - образует объектное ядро предметной области, которое имеет онтологический статус. Нельзя получить в ИС ответ на вопрос о том, что ей неизвестно. Термин объект является первичным, неопределяемым понятием. Синонимами термина "объект" являются "реалия, сущность, вещь". Однако термин сущность понимается нами несколько уже, как компонент модели предметной области, т.е. как уже выделенный на концептуальном уровне объект для базы данных. Таким образом, выделяемые в предметной области объекты превращаются аналитиками (а не проектировщиками базы данных) в сущности. Сущность предметной области является результатом абстрагирования реального объекта путем выделения и фиксации набора его свойств. Сущность является результатом абстрагирования реального объекта, т.е. в нашем контексте имеет гносеологический статус. Хотя далее в контексте сущность нередко отождествляется с объектом.

С объектами связано две проблемы: идентификация и адекватное описание. Для идентификации используют имя. При этом предполагается, что происходит отказ от его смысла, который присущ естественному языку. Используется только указательная функция имени. Имя - это прямой способ идентификации объекта. К косвенным способам идентификации объекта относят определение объекта через его свойства (характеристики или признаки).

     Объекты взаимодействуют между собой через свои свойства, что порождает ситуации. Ситуации - это взаимосвязи, выражающие взаимоотношения между объектами. Ситуации в предметной области описываются посредством высказываний о предметной области с использованием исчисления высказываний и исчисления предикатов, т.е. формальной, математической логики. Например, высказывание "Программист и менеджер есть служащие компании" описывает отношение включения. Таким образом, вся информация об объектах и сущностях предметной области описывается с помощью утверждений на естественном языке.

Выделяют два подхода формирования перечня сущностных объектов предметной области и их атрибутов.

**В первом подходе** (дедуктивном – от общего к частному) на основе формирования общего представления о предметной области, функций, а также информационных потребностей основных абонентов-пользователей ***выделяются основные понятия и категории, которыми оперируют фрагменты предметной области.***

*Данные понятия и категории принимаются за основу списка объектов – сущностей предметной области.*

*Далее на основе анализа служебной и технологической документации, а также консультаций с заказчиком формируются атрибуты, характеризующие выделенные объекты-сущности.*

*Перечень объектов-сущностей и набор их атрибутов должен быть достаточным для решения всех частных задач системы и удовлетворять информационным потребностям абонентов-пользователей системы, однако он не должен быть избыточным, так как это приводит к дополнительным расходам при накоплении информации и эксплуатации АИС.*

***Во втором подходе*** (индуктивном – от частного к общему) на основе анализа служебной и технической документации ***выделяются все необходимые для решения частных задач ИС сведения, их характеристики и параметры, и на этой основе формируется общий перечень атрибутов предметной области***. Далее ***на основе анализа производится агрегация (группирование) атрибутов в группы, образующие объекты-сущности предметной области.***

*Часть атрибутов и понятий предметной области выражают процессы-отношения между объектами-сущностями. Такие атрибуты выделяются, и анализируются параметры и характер связей, которые они выражают – структурность, направленность, множественность, обязательность наличия для экземпляров объектов.*

*Выделение объектов-сущностей, их атрибутов и отношений связей осуществляется на итерационной основе и комбинированным способом. Список объектов многократно уточняется. Атрибуты агрегируются в группы и т.д.*

1. Понятие связи и типы связей. Степень связи. Рекурсивная связь. Способы отображения связи и ограничений связи в диаграммах Чена. Типы связей и отражение связей в среде Erwin. Окно «Свойства связи». Использование ролевых имен в моделях в среде Erwin.

Определение связей.

Наиболее характерными типами связей между сущностями являются:

· связи типа «часть–целое», определяемые обычно глаголами «состоит из», «включает» и т.п.;

· классифицирующие связи (например, «тип – подтип», «множество – элемент», «общее – частное» и т. п.);

· производственные связи (например, «начальник–подчиненный»);

· функциональные связи, определяемые обычно глаголами «производит», «влияет», «зависит от», «вычисляется по» и т. п.

Среди них выделяются только те связи, которые необходимы для удовлетворения требований к разработке БД.

Связь характеризуется следующим набором параметров:

· именем – указывается в виде глагола и определяет семантику (смысловую подоплеку) связи;

· кратностью (кардинальность, мощность): один-к-одному (1:1), один-ко-многим (1:N) и многие-ко-многим (N:M, N = M или N <> M). Кратность показывает, какое количество экземпляров одной сущности определяется экземпляром другой. Например, на одном участке (описывается строкой таблицы «Участки») может быть один, два и более путей (каждый путь описывается отдельной строкой в таблице «Пути»). В данном случае связь 1:N. Другой пример: один путь проходит через несколько раздельных пунктов и через один раздельный пункт может проходить несколько путей – cвязь N:M;

· типом: идентифицирующая (атрибуты одной сущности, называемые внешним ключом, входят в состав дочерней и служат для идентификации ее экземпляров, т.е. входят в ее первичный ключ) и неидентифицирующая (внешний ключ имеется в дочерней сущности, но не входит в состав первичного ключа);

· обязательностью: обязательная (при вводе нового экземпляра в дочернюю сущность заполнение атрибутов внешнего ключа обязательно и для введенных значений должен существовать экземпляр в родительской сущности) и необязательная (заполнение атрибутов внешнего ключа в экземпляре дочерней сущности необязательно или введенным значениям не соответствует экземпляр в родительской сущности);

· степенью участия – количеством сущностей, участвующих в связи. В основном между сущностями существуют бинарные связи, т. е. ассоциации, связывающие две сущности (степень участия равна 2). Например, «Участок» состоит из «Путей». В то же время по степени участия возможны следующие типы связей:

o унарная (рекурсивная) – сущность может быть связана сама с собой. Например, в таблице «Работники» могут быть записи и по подчиненным, и по их начальникам. Тогда возможна связь «начальник» – «подчиненный», определенная на одной таблице;

o тернарная – связывает три сущности. Например, «Студент» на «Сессии» получил «Оценку по дисциплине»;

o кватернарная и т.д.

В методологии IDEF1X степень участия может быть только унарной или бинарной. Связи большей степени приводятся к бинарному виду.

Внешний вид связи на диаграммах IDEF1X указывает на ее мощность, тип и обязательность

Ролевые имена связей – используются для указания ролей сущностей в данной связи.

1. Понятие степени участия. Правило нахождения степени участия. Отражение степени связи в среде Erwin.

Связь характеризуется следующим набором параметров:

· степенью участия – количеством сущностей, участвующих в связи. В основном между сущностями существуют бинарные связи, т. е. ассоциации, связывающие две сущности (степень участия равна 2). Например, «Участок» состоит из «Путей».

В то же время по степени участия возможны следующие типы связей:

* унарная (рекурсивная) – сущность может быть связана сама с собой. Например, в таблице «Работники» могут быть записи и по подчиненным, и по их начальникам. Тогда возможна связь «начальник» – «подчиненный», определенная на одной таблице;
* тернарная – связывает три сущности. Например, «Студент» на «Сессии» получил «Оценку по дисциплине»;
* кватернарная и т.д.

В методологии IDEF1X степень участия может быть только унарной или бинарной. Связи большей степени приводятся к бинарному виду.

Внешний вид связи на диаграммах IDEF1X указывает на ее мощность, тип и обязательность.

Степень участия – количество сущностей, участвующих в связи. В основном между сущностями существуют бинарные связи, т. е. ассоциации, связывающие две сущности (степень участия равна 2). В то же время по степени участия возможны следующие типы связей:

* унарная (рекурсивная) – сущность может быть связана сама с собой. Например, в таблице «Работники» могут быть записи и по подчиненным, и по их начальникам. Тогда возможна связь «начальник» – «подчиненный», определенная на одной таблице;
* тернарная – связывает три сущности. Например, «Студент» на «Сессии» получил «Оценку по дисциплине»;
* кватернарная и т.д.

1. Понятие сущности и типы сущностей. Способы отражения сущностей в диаграммах Чена и IDEF1Х. Признаки сущности. Понятие потенциального и первичного ключа. Роль первичного ключа для проектирования БД.

В IDEF1X различают зависимые и независимые сущности.

Тип сущности определяется ее связью с другими сущностями. Идентифицирующая связь устанавливается между независимой (родительский конец связи) и зависимой (дочерний конец связи) сущностями. Когда рисуется идентифицирующая связь, ERwin автоматически преобразует дочернюю сущность в зависимую. Зависимая сущность изображается прямоугольником со скругленными углами (сущность ***Заказ*** на рис. 2.21). Экземпляр зависимой сущности определяется только через отношение к родительской сущности, т. е. в структуре на рис. 2.21 информация о заказе не может быть внесена и не имеет смысла без информации о клиенте, который его размещает. При установлении идентифицирующей связи атрибуты первичного ключа родительской сущности автоматически переносятся в состав первичного ключа дочерней сущности. Эта операция дополнения атрибутов дочерней сущности при создании связи называется миграцией атрибутов. В дочерней сущности новые атрибуты помечаются как внешний ключ - (FK).

Многообразие ИС можно классифицировать по различным признакам:

-         по признаку структурированности задач;

-         по функциональному признаку и уровню управления;

-         по временной зависимости;

-         по сложности;

-         по степени автоматизации процесса обработки информации;

-         по сфере применения

-         по другим признакам.

По *признаку структурированности задач* различают ИС:

1.     Структурированные,

2.     Не структурированные,

3.     Частично структурированные.

Атрибутов может быть много. Каждый атрибут хранит информацию об определенном свойстве сущности. Каждый экземпляр сущности должен быть уникальным, атрибут или группа атрибутов, которые идентифицируют сущность называются первичным ключом. При определении первичного ключа может быть рассмотрено несколько наборов атрибутов. Эти наборы называются потенциальными ключами. К первичным ключам представляются определенные требования:

1.     Первичный ключ должен однозначно идентифицировать экземпляр сущности.

2.     Первичный ключ должен быть компактен, то есть удаление любого атрибута из составного первичного ключа приводит к потере уникальности экземпляра сущности.

3.     Каждый атрибут из состава первичного ключа не должен принимать нулевого значения.

4.     Каждый атрибут первичного ключа не долженменять своего значения в течении всего времени существования экземпляра сущности.

Потенциальные ключи не ставшие первичными называются альтернативными.

1. Понятия списка требований пользователя и спецификации транзакций. Создание спецификации транзакций. Функциональные характеристики транзакций.

Транзакция базы данных - это логическая единица работы, которая переводит базу данных из одного завершенного состояния в другое завершенное состояние.

Поскольку любая операция изменения данных в базе данных несет в себе потенциальную возможность нарушения целостности данных, необходимо строго определять транзакции и идентифицировать информацию, которая обычно включается в определение транзакции.

Определение транзакции может иметь различные формы. Иногда для определения транзакций используется репозиторий данных CASE -средств проектирования базы данных. Очень часто определение транзакций выполняется посредством текстовых описаний. Независимо от выбранного подхода любое хорошее определение транзакции включает несколько важных элементов. К таким элементам относятся:

* имя транзакции;
* номер транзакции;
* описание транзакции;
* характер транзакции и ее сложность;
* объем транзакции;
* требования к производительности транзакции ;
* относительный приоритет;
* время выполнения транзакции.

Первым шагом в определении транзакции является уникальная идентификация каждой транзакции базы данных. Это можно сделать назначением имени и номера каждой транзакции базы данных. Имена транзакций должны позволять пользователям отличать их друг от друга. Описание транзакций включает перечень операций предметной области, которые выполняются транзакцией. Что касается описания транзакций, то оно должно быть выполнено в терминах предметной области, понятных пользователю. Здесь нужно иметь в виду следующее: а) описание транзакции должно описывать, что транзакция делает для пользователя, а не как она выполняется; и б) описание должно быть понятно пользователю, что не исключает использование технологического жаргона.

Сложность транзакции является условной мерой трудоемкости работы проектировщика базы данных при достижении требований производительности базы данных.

Высокая сложность приписывается обычно транзакции, которая имеет две из следующих характеристик:

* содержит от 8 до 10 команд SQL;
* содержит предложение WHERE с большим количеством предикатов;
* содержит предложение WHERE с более чем тремя соединениями или под запросами;
* обрабатывает более чем 100 строк.

Низкая сложность приписывается транзакции со следующими характеристиками:

* содержит до трех команд SQL;
* содержит предложение WHERE с одним или двумя предикатами;
* обрабатывает менее чем 25 строк.

Транзакция со средней сложностью имеет характеристики между нижней и высокой сложностью

1. Понятия суперкласс и подкласс. Свойства подкласса. Свойства связи «суперкласс-подкласс». Отображение связи «суперкласс-подкласс» в среде Erwin.

**Суперкласс** – тип сущности, включающий разные подклассы, которые необходимо представить в модели данных.

**Подкласс** - тип сущности, который исполняет отдельную роль, а также является членом суперкласса.

Связь между суперклассом и подклассом относится к типу 1:1 и называется связью «суперкласс/подкласс».

Типы сущностей делятся на суперклассы и подклассы, расширяющие свои суперклассы. Допускается множественное наследование (shared subclass). Полное и частичное участие в специализации. Полное – каждая сущность суперкласса является членом какого-либо подкласса по этой специализации. Частичная – наоборот, сущности суперкласса не обязаны быть специализированными. Пересечение/непересечение и полное/частичное участие – ортогональные характеристики.

Категоризация – моделирование одного подкласса-категории со связью, которая охватывает несколько разных суперклассов. При этом каждый экземпляр сущности класса-категории настедует атрибуты только одного суперкласса.

К категориям применимо понятие полного/частичного участия.

Процесс моделирования:

Спецификации требований

Определение типов сущностей

Определение типов связей

Определение показателей кардинальности и степеней участия сторон для типов связей

Определение атрибутов и связывание их с типами сущностей и связей

Определение потенциальных и первичных ключей

Специализация (выделение подклассов) / генерализация (выделение суперклассов) типов сущностей

Категоризация типов сущностей

Программные средства для построения ER-диаграмм: ERwin, Visio

Преобразование ER-модели в реляционную:

Типы сущностей - отношения (атрибуты и ключи переходят практически без изменений)

Суперклассы/подклассы – связь на основе внешнего ключа, указывающего на первичный ключ суперкласса.

Связь One-To-One: внешний ключ + ограничение целостности, либо пара внешних ключей.

Связь One-To-Many: внешний ключ.

Связь Many-To-Many: специальная таблица пересечения, состоящая из 2 внешних ключей, каждый кортеж которой представляет собой экземпляр связи.

Атрибуты связей – уходят во внешние ключи.

При использовании CASE-средств обычно можно произвести подобное преобразование автоматически.

1. Правило нахождения и особенности связи с показателем кардинальности M:N. Признаки ассоциативной таблицы.

Каждую связь характеризуют два структурных ограничения: показатель кардинальности; степень участия. Показатель кардинальности описывает количество возможных связей для каждой сущности-участницы связи. Для нахождения показателя кардинальности необходимо использовать следующее правило, состоящее из трех шагов: Выделяется один экземпляр первой сущности и проверяется, со сколькими экземплярами второй сущности он может вступать в связь. При этом не учитывается временной фактор, другими словами - со сколькими экземплярами второй сущности он может вступать в связь в любое время. Выделяется один экземпляр второй сущности и проверяется, со сколькими экземплярами первой сущности он может вступать в связь в любое время. Результаты первого и второго шага сравниваются, затем выбирается показатель кардинальности данной связи:


Показатель кардинальности зависит от бизнес-правил описываемой организации, а также от набора атрибутов сущностей, вступающих в эту связь. Степень участия определяет, зависит ли существование некоторой сущности от участия в этой связи другой сущности.
ля определения степени участия необходимо задать вопрос все ли экземпляры первой сущности принимают участие в заявленной связи? Если ответ - да, то степень участия первой сущности в этой связи – полная, если ответ - нет, то степень участия первой сущности в этой связи – частичная. Такой же вопрос задается и для второй сущности-участнице этой связи. Ответы заносятся в таблицу № 2. Пример. Результатом данной лабораторной работы является заполненная таблица.


1. Способы реализации транзакций. Работа по проектированию производных атрибутов. Виды реализации производных атрибутов.

Связь – некоторая ассоциация между двумя сущностями, значимая для рассматриваемой предметной области. Примерами связей могут являться родственные отношения «отец–сын», производственные – «начальник-подчиненный» или произвольные – «иметь в собственности», «обладать свойством».

Атрибут (столбец, поле) – свойство сущности или связи.

Связь между сущностями характеризуется:

типом связи (1:1, 1:N, N:М);

классом принадлежности. Класс может быть обязательным и необязательным. Если каждый экземпляр сущности участвует в связи, то класс принадлежности – обязательный, иначе – необязательный.

Неключевыми являются атрибуты, значения которых зависят от значений первичного ключа или составного первичного ключа. Эти не ключевые атрибуты должны зависеть от значения ключа, полного ключа, и ни от чего кроме ключа.

Селективные атрибуты - атрибуты, используемые для идентификации единственного экземпляра сущности, когда ключ не уникален. Также называются вторичными ключами .

Групповой атрибут - атрибут, объединяющий группу более детальных атрибутов.

Атрибуты повторяющейся группы - атрибуты, представляющие несколько включений одного атрибута в рамках сущности.

Производные атрибуты - атрибуты, чьи значения определяются из значений других атрибутов.

Атрибуты основных данных - атрибуты, которые не являются селективными, групповыми, атрибутами повторяющейся группы или производными.

Селективные, групповые и атрибуты повторяющейся группы не должны присутствовать в логической модели, приведенной к третьей нормальной форме. Селективные атрибуты должны стать частью первичного ключа, если они нужны для идентификации единственного экземпляра сущности. Групповые атрибуты являются многозначными. По моему мнению, групповые атрибуты лучше всего представлять в модели кодовыми или классификационными сущностями. В третьей нормальной форме не ключевые атрибуты должны быть простыми (основными) или производными атрибутами.

1. Структура информационной системы. Предназначение и состав основных компонент ИС.

Структуру информационной системы составляет совокупность отдельных ее частей, называемых подсистемами*.*



***Подсистема*** – это часть системы, выделенная по какому-либо признаку.

**Функциональная часть информационной системы** обеспечивает выполнение задач и назначение информационной системы. Фактически здесь содержится модель системы управления организацией. В рамках этой части происходит трансформация целей управления в функции, функций – в подсистемы информационной системы. Подсистемы реализуют задачи. Обычно в информационной системе функциональная часть разбивается на подсистемы по функциональным признакам:

·     уровень управления (высший, средний, низший);

·     вид управляемого ресурса (материальные, трудовые, финансовые и т.п.);

·     сфера применения (банковская, фондового рынка и т.п.);

·     функции управления и период управления.

Например, информационная система управления технологическими процессами – компьютерная информационная система, обеспечивающая поддержку принятия решений по управлению технологическими процессами с заданной дискретностью и в рамках определенного периода управления.

В табл. 2.3 указаны некоторые из возможных информационных систем, однако их достаточно для иллюстрации связи функций систем и функций управления.

***Функциональный признак*** определяет назначение подсистемы, а также ее основные цели, задачи и функции. Структура информационной системы может быть представлена как совокупность ее функциональных подсистем, а функциональный признак может быть использован при классификации информационных систем.

Например, информационная система производственной фирмы имеет следующие подсистемы: управление запасами, управление производственным процессом и др.

В хозяйственной практике производственных и коммерческих объектов типовыми видами деятельности, которые определяют функциональный признак классификации информационных систем, являются: производственная, маркетинговая, финансовая, кадровая.

1. Этапы проектирования БД и пользовательских приложений. Цель и виды работ на этапе концептуального проектирования базы данных и пользовательских приложений.

Законы проектирования  баз данных:
1.Системный анализ предметной области
2.Инфологическое проектирование
3.Выбор СУБД
4.Датологическое проектирование
5.Физическое проектирование

На первом этапе проектирования баз данных рассматриваются цели и задачи с помощью которой они будут решатся. Анализируются информационные потребности будущих пользователей баз данных. Рассматриваются формы входных и выходных потоков, которые будут составлять основу баз данных. Затем уточняются алгоритмы и процедуры обработки данных хранимой в базе данных. Формируются требования, которым должна удовлетворять проектируемая база данных и определяется примерный список объектов  предметной области, свойства которых будут использоваться при разработке базы данных.

II Инфологическое проектирование

На второй стадии проектирования выполняется моделирование данных. **Моделирование данных** – это процесс создания логической структуры данных. Существует два подхода к моделированию данных:
Модель «Сущность-связь»
Семантическая объектная модель
Эти модели представляют собой языки для описания структуры данных и их связей в представлениях пользователей. Моделирование данных, подобно блок-схемам, отражают логику программы.
**Модель «Сущность-Связь»**
Сущность – это объект, идентифицируемый в рабочей среде пользователя за которым пользователь хотел бы наблюдать. Класс сущностей – это совокупность сущностей, которая описывается структурой, либо форматом сущностей, составляющих этот класс.
Экземпляр сущности – представляет собой конкретную сущность.
Атрибуты сущности – это свойства сущности, которые описывают характеристики сущности.
Идентификаторы – это атрибуты, с помощью которых экземпляры именуются или идентифицируются.
Если идентификатор указывает на один экземпляр сущности, то его значение называется уникальным. Если идентификатор не является уникальным, то его значение определяется некоторым множеством экземпляров сущности.
Связи – это взаимоотношения сущностей выраженная связями.
Модель «Сущность-Связь» включает в себя классы связей и экземпляры связей. Классы связей – это взаимоотношения между классами сущностей. Экземпляры связей – это взаимоотношения между экземплярами сущностей.Типы связей:
**Связь один к одному (1:1)** – одиночный экземпляр сущности одного типа связан с одиночным экземпляром сущности другого типа.
**Связь один ко многим (1:М)** – один экземпляр сущности связан со многими экземплярами другой сущности.
**Связь многие ко многим (М:N)** – несколько экземпляров одной сущности связаны с несколькими экземплярами другой сущности.
Модель «Сущность-Связь» или ER-диаграммы включают в себя изображения сущностей в виде прямоугольников (или прямоугольников с закругленными углами), а связей в виде ромбиков (или ромбиков с закругленными углами).
На ER-диаграммах атрибуты обозначаются эллипсами. Если атрибутов у сущности много, то чтобы не загружать ER-диаграмму, атрибуты помещают в прямоугольник, в котором идет перечисление всех атрибутов сущности.

1. Этапы проектирования БД и пользовательских приложений. Цель и виды работ на этапе логического проектирования базы данных и пользовательских приложений.

 1.Системный анализ предметной области
2.Инфологическое проектирование
3.Выбор СУБД
4.Датологическое проектирование
5.Физическое проектирование

На первом этапе проектирования баз данных рассматриваются цели и задачи с помощью которой они будут решатся. Анализируются информационные потребности будущих пользователей баз данных. Рассматриваются формы входных и выходных потоков, которые будут составлять основу баз данных. Затем уточняются алгоритмы и процедуры обработки данных хранимой в базе данных. Формируются требования, которым должна удовлетворять проектируемая база данных и определяется примерный список объектов  предметной области, свойства которых будут использоваться при разработке базы данных.

При выборе СУБД руководствуются следующими соображениями:
аппаратное обеспечение, на котором в дальнейшем будет работать проектируемая база данных;
 системное программное обеспечение, с которым будет в последствии работать проектируемая база данных и соответствующее ей приложения;
методология и подходы, к программированию реализованные в той или иной СУБД;
модель данных, которая встроена в конкретную СУБД;
 Выбор СУБД полностью определяется на II этапе построения базы данных, т. к. оно зависит от той модели данных, которая встроена в выбранную СУБД.

IV Датологическое проектирование

После того, как выбор СУБД завершён, необходимо приступить к проектированию датологической модели базы данных. При формировании датологической схемы, каждая из определённых в концептуальной схеме сущностей отображается в таблицу, которая является одним  отношением. При этом следует учитывать ограничения на размер таблиц, которые накладывает конкретная СУБД.

На этапе логического проектирования разрабатывается логическая структура БД, соответствующая инфологической модели ПО. Решение этой задачи существенно зависит от модели данных, поддерживаемой выбранной СУБД. Результатом выполнения этого этапа являются схемы БД концептуального и внешнего уровней архитектуры, составленные на языках определения данных (DDL) выбранной СУБД.

1. Этапы проектирования БД и пользовательских приложений. Цель и виды работ на этапе физического проектирования базы данных и пользовательских приложений.

1.Системный анализ предметной области
2.Инфологическое проектирование
3.Выбор СУБД
4.Датологическое проектирование
5.Физическое проектирование

**Физическое проектирование базы данных** - процесс подготовки описания реализации базы данных на вторичных запоминающих устройствах; на этом этапе рассматриваются основные отношения, организация файлов и индексов, предназначенных для обеспечения эффективного доступа к данным, а также все связанные с этим ограничения целостности и средства защиты.

Физическое проектирование является третьим и последним этапом создания проекта базы данных, при выполнении которого проектировщик принимает решения о способах реализации разрабатываемой базы данных. Во время предыдущего этапа проектирования была определена логическая структура базы данных (которая описывает отношения и ограничения в рассматриваемой прикладной области). Хотя эта структура не зависит от конкретной целевой СУБД, она создается с учетом выбранной модели хранения данных, например реляционной, сетевой или иерархической. Однако, приступая к физическому проектированию базы данных, прежде всего необходимо выбрать конкретную целевую СУБД. Поэтому физическое проектирование неразрывно связано с конкретной СУБД. Между логическим и физическим проектированием существует постоянная обратная связь, так как решения, принимаемые на этапе физического проектирования с целью повышения производительности системы, способны повлиять на структуру логической модели данных.

Как правило, основной целью физического проектирования базы данных является описание способа физической реализации логического проекта базы данных.

**Этапы физического проектирования баз данных:**

1. Перенос глобальной логической модели данных в среду целевой СУБД.
2. Проектирование основных отношений.
3. Разработка способов получения производных данных.
4. Реализация ограничений предметной области.
5. Проектирование физического представления базы данных.
6. Анализ транзакций.
7. Выбор файловой структуры.
8. Определение индексов.
9. Определение требований к дисковой памяти.
10. Проектирование пользовательских представлений.
11. Разработка механизмов защиты.
12. Обоснование необходимости введения контролируемой избыточности.
13. Текущий контроль и настройка операционной системы.