**Тема 1 — Введение**

**1/2 Общие сведения о базах данных**

**1) Различия между данными и информацией**

Ответ: Данные – это собранные факты по какому-либо вопросу или теме, а информация – это результат объединения, сравнения и выполнения расчётов с данными

**2) Определение базы данных**

Ответ: База данных – это централизованный структурированный набор данных, хранящихся в компьютерной системе. Она (БД) предоставляет средства для извлечения, добавления, изменения и удаления данных по требованию. Также, база данных предоставляет средства для преобразования извлеченных данных в полезную информацию.

**3) Описание элементов системы управления базами данных (DBMS)**

Ответ: Определение: Система управления базами данных (DBMS) — это программное обеспечение, которое контролирует хранение, организацию и извлечение данных. Элементами СУБД являются: управление памятью и хранилищем, словарь данных и язык запросов

**4) Преобразования в сфере компьютерных технологий**

Ответ: Данные преобразования состоят из пяти этапов:

1 этап – Мейнфрейм. В 1970 году были попытки построить СУБД, путём интегрирования аппаратного и программного обеспечения. Так, компьютеры использовались для доступа к мощным мейнфреймам и выполнения команд. Эти компьютеры отображали результат только после завершения обработки мейнфреймом.

2 этап – Настольные компьютеры. В 1980-х обработка данных переместилась с мейнфреймов на клиентские компьютеры. Данные компьютеры, имеющие собственное программное обеспечение и самостоятельно выполняющие некоторую обработку, стали известны как «смарт-клиенты»

3 этап - Среда «клиент-сервер» 1990-е. В среде «клиент-сервер» используется Интернет и быстродействующие серверы для удовлетворения потребностей организаций в хранении данных и получении информации. Программное обеспечение, которое управляет данными, находится на сервере баз данных и выполняет операции по хранению и извлечению данных.

4 этап - Распределенные вычисления 2000-е. В модели распределенных вычислений (в частности, в грид-системе) все компьютеры в организации, находящиеся в разных местах, могли использоваться как пул вычислительных ресурсов.

5 этап - Облачные вычисления 2010-е. Облачные вычисления позволяют предоставлять вычислительные услуги через Интернет.

Три основные категории облачных услуг:

IaaS — позволяет вам арендовать сервера, хранилища, операционные системы и т. д. в облачной среде.

PaaS — предоставление доступа к онлайн-среде для разработки и тестирования ПО без затрат на установку или управление.

SaaS — предоставление ПО напрямую из Интернета. Пользователи обычно получают к нему доступ, используя веб-браузер.

**5) Определение бизнес-сценариев и отраслевых примеров применения баз данных**

Ответ:

Бизнес-сценарий — это совокупность обстоятельств, которые могут возникать в рамках бизнеса. Как правило, сценарий основан на сочетании событий и факторов, которые составляют текущую ситуацию в бизнесе, а затем дополняются некоторыми прогнозами возможных событий, которые могут произойти в будущем.

Примеры:

Организации используют базы данных для хранения информации о своих сотрудниках, зарплатах, премиях, налогах и создания платежных чеков.

Банки используют базы данных для хранения информации о клиентах, счетах, займах и транзакциях.

Школы и колледжи используют базы данных, чтобы хранить информацию об учебных курсах, учащихся и преподавателях.

**1/3 Типы моделей баз данных**

**6) Описание процесса разработки базы данных**

Ответ:

1. Стратегия и анализ – концептуальное моделирование данных;
2. Дизайн – проектирование базы данных;
3. Построение – построение базы данных.

**7) Описание основных типов моделей баз данных:**

* Модель плоских файлов

Базы данных на основе плоских файлов проектируются при использовании одной таблицы. Такие БД, как правило, имеют текстовый формат, где в каждой строке содержится только одна запись. - Поля в записи отделяются друг от друга разделителями, например знаками табуляции и запятыми.

* Иерархическая модель

В иерархической модели данные организованы в виде древовидной структуры. Данные хранятся как записи, и между ними существуют связи. Запись является набором полей и соответствует строке в реляционной модели БД.

* Сетевая модель

Сетевая модель — это модель БД, которая обеспечивает гибкий способ представления объектов и их связей. Сетевая база данных состоит из набора записей, между которыми существуют связи (прямоугольники = поля, линии = связи). Каждая запись представляет собой набор полей, каждое из которых содержит только одно значение данных. Связь указывает на ассоциацию двух записей

* Объектно-ориентированная модель

В данной модели БД объект моделируется в виде объекта. Для каждого объекта определено состояние (набор значений для атрибутов объекта) и поведение (набор методов, которые совершают операции с состоянием объекта). Связь между объектами осуществляется путем предоставления доступа. Объект должен принадлежать только к одному классу, как экземпляр этого класса. Также, вы можете наследовать новый класс (подкласс) из существующего класса (суперкласса).

* Реляционная модель

Это модель БД, в которой данные представлены в виде набора таблиц. Каждый столбец представляет атрибуты, относящиеся к таблице. Каждая строка представляет собой экземпляр таблицы. Каждая таблица может быть визуально представлена в виде столбцов и строк. Каждая таблица имеет поле или набор полей, которые однозначно идентифицируют строку. Порядок строк и столбцов не имеет значения. Каждая строка уникальна. Каждое поле может содержать только одно значение. Значения в столбце или поле из одного и того же домена (типа данных). Имена таблиц должны быть уникальными. Имена столбцов в каждой таблице должны быть уникальными.

**1/4 Бизнес-требования**

**8) Объяснение необходимости базы данных**

Ответ:

Использование БД необходимо, так как существует множество пользователей, много элементов данных, а также происходит интеграция с разными компонентами.

**9) Объяснение важности бизнес-правил**

Ответ:

Бизнес-правила крайне важны, так как:

1) Позволяют разработчику или архитектору понять, какие имеются связи и ограничения для участвующих объектов.

2) Помогают понять процедуру стандартизации, которую применяет организация при обработке огромных объемов данных.

3) Бизнес-правила служат для понимания бизнес-процессов, а также характера, роли и области применения данных. Следовательно, Все бизнес-правила должны быть простыми и понятными и поддерживаться в актуальном состоянии.

4) Бизнес-правила помогают классифицировать и проектировать таблицы базы данных.

Определение бизнес-правила:

Бизнес-правило — это высказывание, которое определяет или ограничивает какие-либо аспекты бизнеса. Оно используется для определения объектов, атрибутов, связей и ограничений. Оно содержит правила и политики, определяющие ведение бизнеса, и определяет повседневные операции.

Существует два типа бизнес-правил: структурные и процедурные.

Пример: все заказы должны иметь номер контактного лица.

Источниками бизнес-правил являются:

Руководители;

Лица, определяющие;

Документация и руководства по эксплуатации и т.д.

**10) Описание принципов и примеров создания бизнес-правил**

Ответ:

Принципы бизнес-правил:

Бизнес-правила прописаны в концептуальной модели для бизнеса, которая в свою очередь точно описывает информационные потребности бизнеса, учитывает правила и законы, регулирующие отрасль бизнеса, и способствует обсуждению тех или иных требований и предложений. Также, такая модель предотвращает возможные ошибки и неправильное понимание, формирует важную документацию, а также формирует основу для разработки физической базы данных. В данной модели бизнес-правила документируются как процессы деловой деятельности.

Пример:

Компания Теремок предоставляет быстрое питание в своих ресторанах и имеет множество заведений по всему миру. Так как компания имеет очень большую клиентскую базу, была изобретена система лояльности с четырьмя уровнями: бронзовый ключ, серебряный, золотой и платиновый. Клиенты сразу бесплатно получают бронзовый ключ, а чтобы перейти на другие, необходимо платить взносы. При этом перейдя на следующий ключ клиент получает определённые привилегии.

Из этого сценария можно выделить бизнес-правило и ограничение:

Бизнес-правило: клиенты должны платить взносы, чтобы перейти на новый уровень

Ограничение: клиенты должны относиться к одному из уровней (бронзовый, серебряный, золотой или платиновый ключ)

**11) Объяснение важности четких формулировок и требований к сбору точной информации**

Ответ:

Независимо от области изучения или предпочтений при определении данных (качественных или количественных), тщательный сбор данных является существенной составляющей для целостности исследования. Выбор подходящих инструментов сбора данных (существующие, модифицированные или специально разработанные), а также ясно определённые инструкции по правильному применению инструментов сокращают возможность возникновения ошибок

Формальный процесс сбора данных необходим, поскольку это обеспечивает определенность, точность полученных данных и верность выводов, основанных на этих данных. Процесс сбора обеспечивает как точку отсчёта для измерений, так и некоторые указания, которые можно улучшить.

**Тема 2 — Базы данных и моделирование данных**

**2/1 Реляционные базы данных**

**1) Описание функций отдельной таблицы**

Ответ:

Таблица представляет собой простую структуру, в которой осуществляется организация и хранение данных.

**2) Описание функций и правил реляционной базы данных**

Ответ:

- Каждая таблица должна иметь особое имя.

- Каждая таблица может содержать несколько строк.

- Каждая таблица должна иметь значение для однозначной идентификации строк.

- Каждый столбец должен иметь уникальное имя.

- Записи в столбцах должны быть однозначными.

- Записи в столбцах должны быть одинакового типа.

- Порядок строк и столбцов не имеет значения

**3) Описание преимуществ и недостатков типов баз данных**

Ответ:

Преимущества и недостатки баз данных на основе плоского файла:

- Преимущества:

1. Простота понимания
2. Простота реализации
3. Простота извлечения информации
4. Все записи хранятся в одном месте
5. Простота сортировки и фильтрации записей
6. Низкие требования к аппаратному и программному обеспечению

- Недостатки:

1. Низкая безопасность
2. Несогласованность данных
3. Избыточность данных
4. Обременительное предоставление информации
5. Медленные базы данных огромного объема

Преимущества реляционной базы данных (многотабличной):

1. Меньше избыточность
2. Исключение несогласованности
3. Эффективность
4. Целостность данных
5. Конфиденциальность

**4) Описание реляционных таблиц и основных понятии**

Ответ:

1. Таблица — базовая структура для хранения данных
2. Столбец — атрибут, который описывает информацию в таблице
3. Первичный ключ — уникальный идентификатор для каждой строки
4. Внешний ключ — столбец, который ссылается на столбец первичного ключа в другой таблице
5. Строка — данные для одного экземпляра таблицы
6. Поле — одно значение на пересечении строки и столбца

**2/2 Концептуальные и физические модели данных**

**1) Описание концептуальной модели данных**

Ответ:

Отражает функциональные и информационные потребности бизнеса.

Основана на текущих потребностях, но может отражать будущие потребности.

Удовлетворяет потребности бизнеса (что идеально теоретически), но не отвечает их реализации (что физически возможно)

Определяет:

Важные объекты (которые станут таблицами в базе данных)

Связи между объектами

Не определяет:

Атрибуты (которые станут столбцами или полями в базе данных)

Уникальные идентификаторы (атрибуты, которые станут первичными ключами в базе данных)

**2) Описание логической модели данных**

Ответ:

1. Содержит все объекты и связи между ними.
2. Называется моделью связей между объектами (ERM).
3. Иллюстрируется ERD-диаграммой.
4. Указывает все атрибуты и уникальные идентификаторы (UID) для каждого объекта.
5. Определяет обязательность атрибутов.
6. Определяет обязательность и кардинальность связей.

**3) Описание физической модели данных**

Ответ:

1. Является расширением логической модели данных.
2. Указывает определения таблиц, типы данных и точность.
3. Определяет представления, индексы и прочие объекты базы данных.
4. Описывает, как должны быть реализованы объекты в конкретной базе данных.
5. Показывает все структуры таблиц, включая столбцы, первичные ключи и внешние ключи.

**4) Анализ сходств и различий между концептуальными и физическими моделями данных**

Ответ:

Искусство планирования, разработки и коммуникации создает желаемый результат.

Моделирование данных — это процесс сбора важных концепций и правил, которые характеризуют бизнес, и их визуальное представление на диаграмме.

Эта диаграмма становится основой для проектирования физического продукта.

Мечты клиента (концептуальная модель) становятся физической реальностью (физическая модель).

Концептуальная модель является неким планом и основой для проектирования физической модели данных, иными словами, на концептуальном уровне происходит интеграция внешних представления в единую схему базы данных, а уже на физическом уровне происходит реализация базы данных на конкретной платформе

**2/3 Объекты и атрибуты**

**1) Определение объектов**

Ответ:

Объекты – это информация, которую требуется отслеживать. Наименование предметов, которые вы можете перечислить (обычно в форме существительного)

Типы объектов

Можно классифицировать объекты по следующим типам:

Первичный - существует независимо

Характеристика - существует благодаря другому(первичному) объекту

Пересечение - существует благодаря двум или более объектам

Объекты содержат экземпляры.

Экземпляр объекта — одно вхождение объекта.

Объекты представляют набор экземпляров, представляющих интерес для конкретной организации.

**2) Определение атрибутов**

Ответ:

Атрибуты описывают сущности и представляют собой конкретную информацию, которая должна быть известна. Также, атрибуты служат однозначными характеристиками объекта

Характеристики атрибута:

Атрибуты изображаются в ERD-диаграмме в пределах прямоугольника, определяющего объект.

Имена атрибутов должны быть существительными в единственном числе, содержащими строчные и прописные буквы.

В большинстве случаев имя атрибута не должно содержать в себе имя объекта, потому что атрибуты указываются с именем объекта.

Атрибуты классифицируются следующим образом:

Обязательные (не допускается значение null), обозначаются символом \*

Необязательные (допускается значение null), обозначаются строчной буквой o

Временные атрибуты — нестабильные.

Пример: Age

Постоянные атрибуты — стабильные.

Пример: Birth Date

**3) Определение обязательных, необязательных, временных и постоянных атрибутов**

Ответ:

Обязательные атрибуты должны иметь значение.

Необязательные атрибуты могут не иметь значения и оставаться пустыми (null).

Простые или атомарные атрибуты — это атрибуты, которые нельзя разделить на компоненты.

Составные атрибуты — это атрибуты, которые можно разделить на более мелкие компоненты, представляющие собой базовые атрибуты с самостоятельными значениями

Однозначные атрибуты могут иметь единственное значение в определенный момент времени.

Пример: Student Last Name

Многозначные атрибуты могут иметь более одного значения в определенный момент времени.

Пример: Address

**4) Описание обозначений Баркера, Бахмана и обозначений информационной инженерии**

Ответ:

Система обозначений Бракера:

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Правила представления объектов:

Объект представляется в виде прямоугольника с закругленными углами.

Следует присвоить объекту имя и поместить это имя в левом верхнем углу прямоугольника данного объекта.

Имя объекта должно быть в верхнем регистре, существительным в единственном числе.

Правила представления атрибутов:

Имена атрибутов должны быть понятны всем, а не только разработчикам.

Первая буква каждого слова атрибута должна быть прописной, а остальные буквы — строчными.

Рядом с каждым атрибутом следует указать символ, представляющий тип атрибута (\* — обязательный; o — необязательный; # — UID)

Правила представления связей:

Связь может существовать максимум между двумя объектами.

Может существовать связь объекта с самим собой (рекурсивная).

Связь имеет два направления.

Оба направления связи должны быть отмечены.

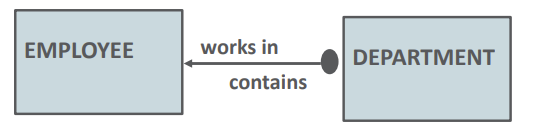
Система обозначений Бахмана:

Объект (представленный прямоугольником)

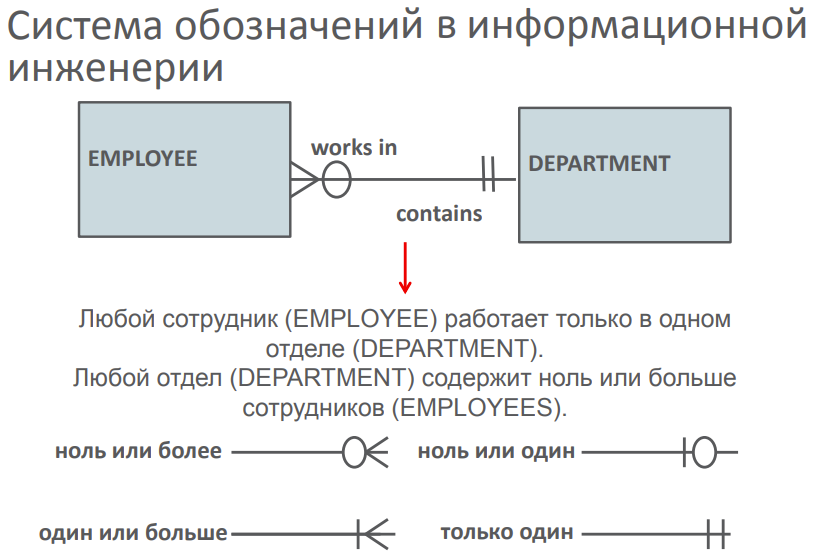
Изображение выглядит как текст

Автоматически созданное описание

Атрибуты

Линии связей

Система обозначений в информационной инженерии:



**2/4 Уникальные идентификаторы**

**1) Определение уникальных идентификаторов (UID)**

Ответ:

Уникальный идентификатор (UID) — это атрибут объекта, который соответствует следующим правилам:

1. Он уникален среди всех экземпляров данного объекта.
2. Его значение не равно NULL для каждого экземпляра объекта в течение времени существования экземпляра.
3. Его значение не изменяется в течение времени существования экземпляра.

UID — специальный атрибут или группа атрибутов, которая однозначно идентифицирует конкретный экземпляр объекта. При этом каждый объект должен иметь уникальный идентификатор. Иначе он не является объектом.

Простые и составные уникальные идентификаторы (UID)

UID из одного атрибута является простым UID

Иногда одного атрибута недостаточно, чтобы однозначно идентифицировать экземпляр объекта

Если UID представляет собой комбинацию атрибутов, его называют составным UID

**2) Определение искусственных уникальных идентификаторов**

Ответ:

Искусственный UID создается из данных, которые присваивает или генерирует система. Такие идентификаторы не встречаются в реальном мире, а создаются для идентификации в системе.

**3) Определение составных уникальных идентификаторов**

Ответ:

Простые и составные уникальные идентификаторы (UID)

UID из одного атрибута является простым UID

Иногда одного атрибута недостаточно, чтобы однозначно идентифицировать экземпляр объекта

Если UID представляет собой комбинацию атрибутов, его называют составным UID

**4) Определение потенциальных и дополнительных уникальных идентификаторов**

Ответ:

Потенциальные уникальные идентификаторы — это идентификаторы, в которых объект может иметь более одного UID. Например:

Номер бейджа (Badge number)

Номер платежной ведомости (Payroll number)

Только один из потенциальных UID можно выбрать в качестве первичного UID. Остальные потенциальные UID называются вторичными UID.

**5) Определение первичных ключей**

Ответ:

UID становится первичным ключом, когда логическая модель преобразуется в физическую базу данных.

Первичный ключ (PK) — это столбец или набор столбцов, которые однозначно идентифицируют каждую строку в таблице. Он не может содержать значения null.

PK — это либо существующий столбец таблицы, либо столбец, который специально генерируется базой данных в соответствии с определенной последовательностью. Он должен содержать уникальное значение для каждой строки данных.

**2/5 Связи**

**1) Определение и распознавание примеров связей и соответствующих внешних ключей**

Ответ:

Связь представляет собой двунаправленную значимую ассоциацию между двумя объектами и более объектами или объекта с самим собой.

Линия связи на диаграмме обозначается либо сплошной (обязательная), либо пунктирной линией (необязательная).

Эти линии заканчиваются «черточкой» (один экземпляр) или «птичьей лапкой» (один или более экземпляров).

Связи должны иметь имена, которые помогают описать связь между объектами. На диаграмме имя связи в любом направлении указано рядом с начальной точкой линии связи

Примеры:

ДЕПАРТАМЕНТ содержит РАБОТНИК.

РАБОТНИК назначен для ДЕПАРТАМЕНТ.

Внешний ключ

Связи в концептуальной модели данных сопоставляются с внешними ключами в таблице физической базы данных.

Внешний ключ (FK) — это столбец или комбинация столбцов в одной таблице, которая ссылается на первичный ключ в той же таблице или в другой таблице.

Компоненты связи

Компонентами связи являются:

Имя - метка, которая появляется рядом с объектом, для которого она назначена. Имена связи должны быть указаны в нижнем регистре.

Кардинальность - минимальное и максимальное число значений в связи. Например, одна и только одна соответствующая запись или одна или более соответствующих записей.

Обязательность - должна ли существовать связь. Подразделяется на:

Необязательная - ноль соответствующих записей

Обязательная - хотя бы одна соответствующая запись в каждом объекте

**2) Определение обязательности связей**

Ответ:

Связь является обязательной, если в данной связи должен участвовать каждый экземпляр сущности;

Необязательной — если не каждый экземпляр сущности должен участвовать в данной связи.

**3) Определение кардинальности связей**

Ответ:

Кардинальность служит мерой количества чего-либо. В связи она определяет степень, с которой один объект связан с другим, отвечая на вопрос «Сколько?»

**4) Типы связей**

Ответ:

Все связи представляют собой информационные требования и правила ведения бизнеса:

1. «Многие к одному» (M:1) или «один ко многим» (1:M)
2. «Многие ко многим» (M:M)
3. «Один к одному» (1:1)
4. Рекурсивные связи (связи объекта с самим собой)

**5) Матрица связей**

Ответ:

Матрицу связей можно использовать для сбора первоначальной информации о связях между объектами. Матрица связей имеет следующие характеристики:

1. Матрица связей показывает, состоит ли объект строки, расположенный слева, в каких-либо отношениях с объектом столбца в верхней части матрицы.
2. Все объекты перечислены в левой и верхней частях матрицы.
3. Если объект строки связан с объектом столбца, имя этой связи отображается в области пересечения.
4. Если объект строки не связан с объектом столбца, область пересечения пуста.
5. Каждая связь выше диагональной линии является обратной (служит зеркальным отражением) связи ниже диагональной линии.
6. Рекурсивные связи представлены полями на диагонали.

**2/6 Моделирование связей между объектами (ERD-диаграммы)**

**1) Описание моделирования данных**

Ответ:

Существуют два вида моделей: концептуальная и логическая.

Концептуальное моделирование подразумевает собой описание предмета, важного для бизнеса (объекта), определение связи самого высокого уровня между различными объектами. Однако, концептуальная модель может не содержать сведения о кардинальности и допустимости неопределённого значения (null). Также, концептуальная модель не определяет атрибуты или UID для каждого объекта.

Логическая модель данных:

Описывает данные максимально подробно, независимо от того, как это будет физически реализовано в базе данных.

Обычно создается на базе концептуальной модели данных.

Содержит все объекты, атрибуты, UID и связи, а также определяет обязательность и кардинальность этих элементов.

**2) Объяснение понятия «без реализации» (implementation-free) в отношении разработки моделей данных и проектов баз данных**

Ответ:

Хорошая логическая модель данных остается неизменной независимо от типа системы базы данных, где в итоге она будет реализована. Именно это означает термин «модель, не зависящая от реализации» (implementation-free)

**3) Перечисление четырех целей моделирования связей между объектами**

Ответ:

1. Собрать всю необходимую информацию;
2. Убедиться, что информация отображается только один раз;
3. Не тратить время на моделирование информации, производной от других, уже смоделированных данных;
4. Представить информацию в предсказуемой логической форме

**4) Определение диаграммы «объект-связь» (ERD)**

Ответ:

ERD — это модель, которая определяет понятия или объекты, существующие в системе, а также связи между этими объектами.

**5) Сопоставление связей с помощью ERD-инструкции**

Ответ:

ERD-инструкции — это словарь, который используется для четкого описания бизнес-правил, которые указаны в ERD. Просто разбейте каждую ERD-инструкцию на ее компоненты.

Компоненты ERD-инструкции:

1. EACH
2. Entity A
3. OPTIONALITY (должен быть/может быть)
4. RELATIONSHIP NAME
5. CARDINALITY (один и только один/один или более)
6. Entity B

**6) Создание компонентов ERD, представляющих собой объекты и атрибуты, в соответствии с правилами построения диаграмм**

Ответ:

Хронология построения:

1. Создание объектов и атрибутов
2. Выбор уникальных идентификаторов (UID)
3. Построение связей
4. Определение обязательности и кардинальности
5. Проверка модели

**Тема 3 — Уточнение модели данных**

**3/1 Дополнительные действия со связями**

**1) Решение связей типа M:M**

Ответ:

Решение связи типа М:М реализуется при помощи введения объекта пересечения и двух ассоциативных связей 1:М



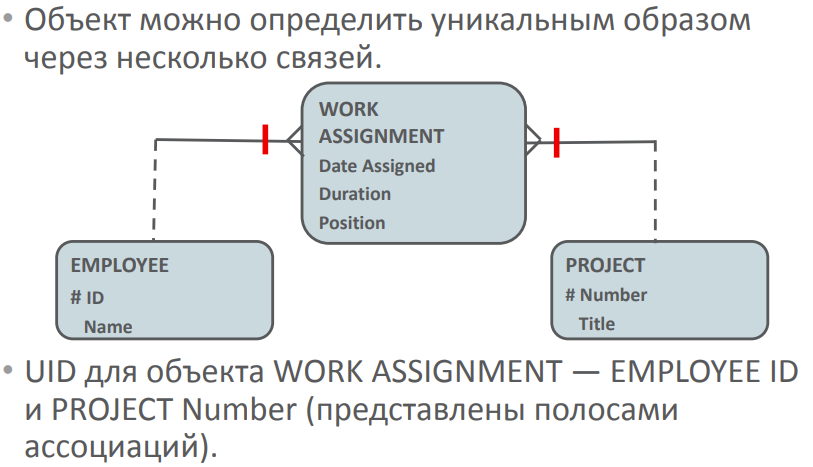
Характеристики объекта пересечения:

1. Связи из объекта пересечения всегда обязательные.
2. Объекты пересечения обычно содержат расходные материалы, например использованное количество и даты. Обычно это временные объекты большого объема.
3. Объект пересечения определяется по своим двум исходным связям (определяющим связям).

**2) Определение ассоциативных связей**

Ответ:

Ассоциативная связь подразумевает собой определение объекта, с помощью составления его уникального идентификатора из одной или более сущности. Пример:



**3) Определение и примеры непередаваемых связей**

Ответ:

Неперемещаемые(непередаваемые) связи характеризуются:

1. Перемещаемость — это возможность связи между двумя экземплярами объекта меняться со временем.
2. Неперемещаемую связь невозможно переместить между объектами, которые она соединяет.
3. Неперемещаемая связь обозначается ромбом.
4. Неперемещаемые связи могут быть только обязательными.

Например:

Место рождения (страна, город и т. д.) для человека представляет собой неперемещаемую связь.



**4) Определение и построение объектов с супертипом и подтипом**

Ответ:

ОПРЕДЕЛЕНИЕ

Объекты-супертипы и объекты-подтипы

Супертип – это класс объектов, который делится на взаимоисключающие подгруппы меньшего размера. Также, супертип имеет связь «родитель-потомок» с одним или несколькими подтипами.

Подтип — это подгруппировка объекта в типе объекта, имеющая атрибуты, которые отличаются от атрибутов в других подгруппировках.

Пример подтипа:

Изображение выглядит как текст

Автоматически созданное описание

Каждый подтип является конкретизацией супертипа, и поэтому должен быть заключен внутри объекта. Общие атрибуты и связи для всех подтипов должны быть перечислены только в супертипе, но они наследуются в каждом подтипе. Подтип может и, как правило, имеет собственные атрибуты и связи. Не может быть только одного подтипа; необходимо создать другой подтип, содержащий все остальное.

Характеристики подтипа:

1. Наследует все атрибуты супертипа.
2. Наследует все связи супертипа.
3. Обычно имеет собственные атрибуты или связи.
4. Извлекается внутри супертипа.
5. Никогда не существует один.
6. Может иметь собственные подтипы.
7. Имеет одинаковые первичные ключи супертипа и подтипа.

ПОСТРОЕНИЕ

Есть 2 подхода - обобщение и конкретизация

Обобщение — это подход «снизу вверх», при котором два или больше объектов нижнего уровня объединяются для формирования объекта более высокого уровня на основе общих черт.

Конкретизация — это подход «сверху вниз», при котором объект более высокого уровня разбивается на объекты нижнего уровня.

Правила для объекта-подтипа:

Исчерпывающие:

Все экземпляры супертипа также являются экземплярами одного из подтипов. Для категоризации экземпляров, которые не определяются одним из существующих подтипов необходимо добавить подтип OTHER.

Пример: у сотрудника должна быть полная занятость, частичная занятость или другой тип занятости.

Взаимоисключающие:

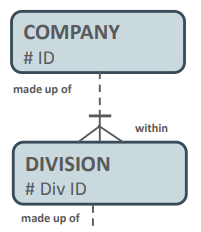
Каждый экземпляр супертипа принадлежит только одному подтипу.

Пример: у сотрудника не может быть одновременно и полная, и частичная занятость.

**5) Определение иерархических, рекурсивных и дуговых связей**

Ответ:

Иерархические связи — это связи, в которых один из элементов данных является родителем другого элемента.



Рекурсивная связь — это связь, при которой экземпляр объекта связан с другим экземпляром в этом же объекте. Рекурсивная связь всегда моделируется циклом.

Изображение выглядит как текст

Автоматически созданное описание

Дуговые связи:

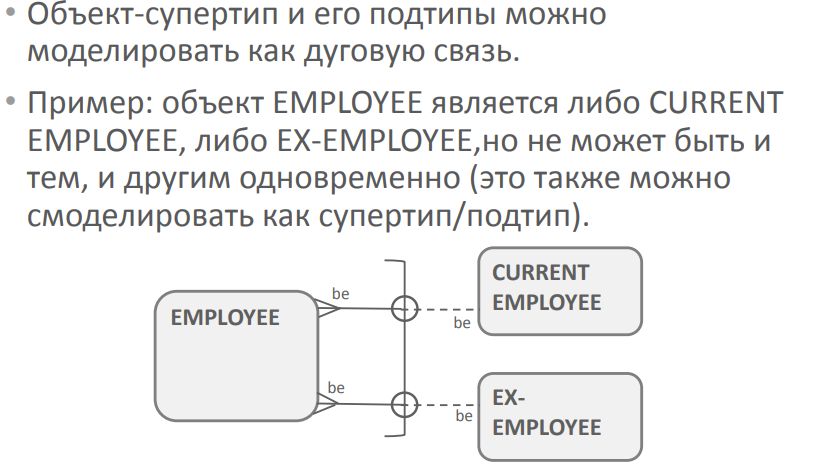
Дугой обводится группа эксклюзивных связей — в ней может существовать только одна из связей для любого экземпляра объекта.

Дуга, нарисованная между двумя связями, соединяет их и демонстрирует взаимоисключение. Такая связь предполагает наличие условия «или».

Дуга означает, что любой экземпляр этого объекта в заданный момент времени может иметь только одну действительную связь внутри дуги.

Характеристики дуговой связи

1. Связи в дуге часто имеют одинаковое название.
2. Связи в дуге должны быть либо все обязательные, либо все необязательные, и должны иметь одинаковую кардинальность.
3. Дуга принадлежит одному объекту и должна содержать только те связи, которые исходят из него.
4. Объект может иметь несколько дуг, но каждая связь может быть частью только одной дуги.
5. Дуговая связь представлена дугообразной линией, соединяющей две или более линий связи.
6. Связи, включенные в дугу, обозначены кружком на линии дуги связи.



**3-2 Отслеживание изменений данных**

**o Отслеживание изменений данных с течением времени**

Моделирование данных на протяжении времени:

1. Добавление исторических данных

• Добавьте в модель объекты и связи для размещения исторических данных.

– Требуется ли журнал аудита?

– Могут ли со временем изменяться значения атрибутов?

– Могут ли со временем изменяться связи?

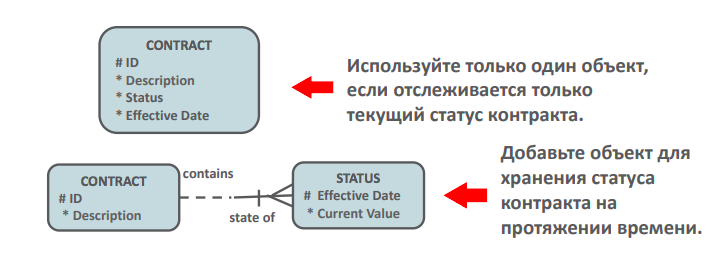
– Нужно ли запрашивать более старые данные?

Зачем это нужно?

1. Большинству компаний необходимо отслеживать исторические данные, чтобы выявлять тенденции и шаблоны, которые могут послужить основой для бизнес-инноваций или улучшения процессов.
2. Каждое обновление атрибута или перемещение связи означает потерю информации. Часто информация уже бесполезна, но некоторые системы должны продолжать отслеживать некоторые или все исторические значения атрибута. Кроме того, некоторые системы должны хранить журнал аудита для каждой транзакции.
3. Хранение ненужных исторических данных может быть затратным.

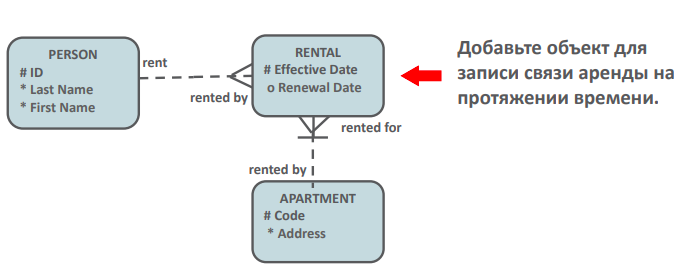
2. Отслеживание значений атрибута

Создайте дополнительные объекты, чтобы отслеживать значения атрибутов на протяжение времени.



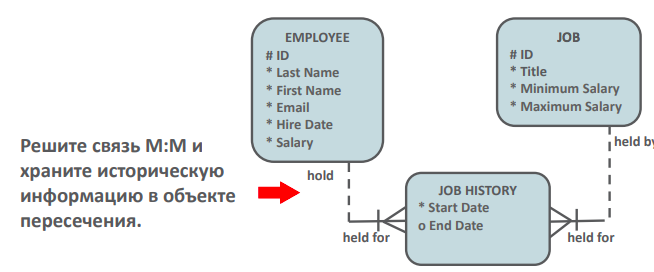
3) Изменение связи

Добавьте объект для добавления связи, которая может со временем измениться.



1. ​​: Создание объектов пересечения

• Объект пересечения часто используется для отслеживания информации о связи, которая меняется со временем.



1. Ведение журнала измененной информации

Дополнительные объекты и атрибуты хранят измененную информацию.



**3-3 Нормализация и бизнес-правила**

**o Объяснение нормализации**

Нормализация

* Процесс организации атрибутов и таблиц реляционной базы данных, позволяющий свести к минимуму избыточность.
* Помогает обрабатывать аномалии вставки, обновления и удаления, обеспечивая повышение производительности базы данных.
* Нормализация — концепция реляционной базы данных, но ее принципы применимы к моделированию данных.
* Целью является нормализация данных до 3NF перед преобразованием модели в реляционный проект

Изображение выглядит как стол

Автоматически созданное описание

**o Описание нормальных форм**

1NF

* Для первой нормальной формы необходимо наличие атрибутов без нескольких значений.
* Чтобы проверить пригодность для 1NF, проверьте, что каждый атрибут имеет только одно значение для каждого экземпляра объекта.
* Если атрибут имеет несколько значений, создайте дополнительный объект и свяжите его с исходным с помощью связи 1:M.

2NF

* Вторая нормальная форма (2NF) требует, чтобы любой атрибут, не являющийся UID, зависел (был свойством или характеристикой) от всего UID.
* Если UID составной, то каждый атрибут должен зависеть от всех частей составного UID.
* Если атрибут не зависит от всего UID, создайте дополнительный объект с частичным UID.

3NF:

* Правило третьей нормальной формы (3NF) гласит, что никакой атрибут, не являющийся UID, не может зависеть от другого атрибута, не являющегося UID.
* Третья нормальная форма запрещает транзитивные зависимости.
* Транзитивная зависимость существует в том случае, когда какой-либо атрибут в объекте зависит от другого атрибута, не являющегося UID, в этом объекте.
* Необходимо переместить в новый объект все атрибуты, не являющиеся UID, которые зависят от другого атрибута, не являющегося UID.

**o Проверка данных с использованием нормализации**

**o Описание бизнес-правил**

* Бизнес-правило — это высказывание, которое определяет или ограничивает какие-либо аспекты бизнеса.
* Оно используется для определения объектов, атрибутов, связей и ограничений.
* Оно содержит правила и политики, определяющие ведение бизнеса, и определяет повседневные операции

Существует два типа бизнес-правил:

– Структурные

– Процедурные

Источники бизнес-правил

* Руководство и руководители высшего звена
* Письменная документация
* Процедуры
* Стандарты
* Руководства по эксплуатации
* Прямые собеседования с конечными пользователями

Структурные бизнес-правила

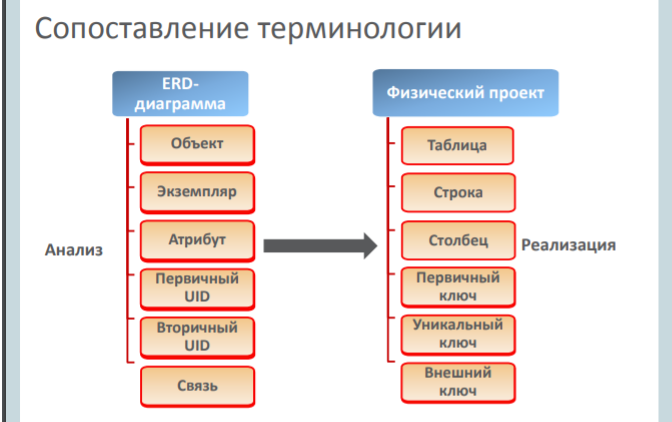
* Структурные бизнес-правила обозначают тип информации, которая должна храниться, и взаимодействие элементов информации.
* Структурное бизнес-правило определяет конкретный статичный аспект бизнеса.
* Эти правила всегда можно представить в виде диаграммы ERD.

Процедурные бизнес-правила

* Процедурные правила определяют предварительные требования, шаги, процессы к рабочим процессам.
* Многие процедурные бизнес-правила связаны со временем: событие A должно произойти до события Б.
* Некоторые процедурные правила невозможно изобразить в виде диаграммы, но их необходимо задокументировать, чтобы впоследствии запрограммировать.

**3-4 Терминология моделирования данных и сопоставление**

**o Применение сопоставления терминологии между логическими и физическими моделями**

****

**o Изучение и применение правил именования Oracle для таблиц и столбцов, используемых в физических**

**моделях**

Правила именования таблиц

* Имя таблицы — это множественное число от имени объекта. – Пример: STUDENT становится STUDENTS.
* Имена столбцов аналогичны именам атрибутов, за тем исключением, что специальные символы и пробелы заменяются на подчеркивания.

Правила именования столбцов

* Имена столбцов аналогичны именам атрибутов, за тем исключением, что специальные символы и пробелы заменяются на подчеркивания.
* В именах столбцов часто используется больше сокращений, чем в именах атрибутов.

Ограничения именования в Oracle

* Имена таблиц и столбцов:
* Должны начинаться с буквы
* Могут содержать до 30 символов
* Не могут содержать пробелы или специальные символы, например «!» (но «$», «#» и «\_» разрешены)
* Не могут быть «зарезервированными словами» в БД Oracle или SQL
* Имена таблиц должны быть уникальными в пределах одной учетной записи пользователя в базе данных Oracle.
* Имена столбцов в таблице должны быть уникальными.

**o Применение правил сопоставления связей для правильного преобразования отношений**

**Тема 4 — Oracle SQL Developer Data Modeler**

**4-1 — Oracle SQL Developer Data Modeler**

**o Создание следующих элементов с помощью Oracle SQL Developer Data Modeler:**

* Объекты, атрибуты и UID с правильной обязательностью и кардинальностью

Чтобы создать объект, выполните следующие действия.

a. На панели инструментов выберите инструмент New Entity, затем нажмите на любое свободное место панели Logical. Появится окно Entity Properties.

b. В окне Entity Properties введите имя объекта. В примере на слайде имя объекта «BOOK». Не закрывайте окно после ввода имени.

Чтобы добавить к объекту атрибуты, выполните следующие действия.

a. Выберите Attributes в навигаторе окна Entity Properties.

b. Нажмите на значок Add an Attribute.

c. В поле Name введите имя атрибута.

d. В поле Data type выберите Logical, затем в раскрывающемся списке выберите нужный тип.

e. Введите размер атрибута.

f. Если атрибут обязательный, установите соответствующий флажок. g. Нажмите OK.

Хотя типы данных не требуются в логической модели (они не будут показаны в ERD-диаграмме), добавление их на этом этапе позволит Data Modeler преобразовать их в фактические типы данных SQL при проектировании физической (реляционной) модели.

Можно задать часто используемые логические типы данных как предпочтительные, выполнив следующие действия.

a. Выберите Tools > Preferences > Data Modeler, затем выберите узел Model.

b. Выберите типы в разделе All Logical Types и переместите их в область предпочтительных типов, нажав на стрелку.

c. Нажмите OK.

– Установка первичных и вторичных UID. Чтобы задать для объекта первичный UID, выполните следующие действия.

a. Выберите Attributes в левой части навигатора окна Entity Properties.

b. Выберите атрибут, который вы хотите назначить в качестве первичного UID.

c. Установите флажок Primary UID.

Атрибут, назначаемый в качестве первичного UID, также автоматически становится обязательным.

Чтобы задать для объекта вторичные UID, выполните следующие действия.

a. Выберите Unique Identifiers в левой части навигатора окна Entity Properties.

b. Нажмите на значок Add, чтобы добавить еще один UID.

c. Нажмите на значок Properties.

d. Введите имя для идентификации вторичного UID.

e. Выберите Attributes and Relations в левой части навигатора окна Key Properties.

f. Выберите атрибут, который хотите назначить в качестве вторичного UID.

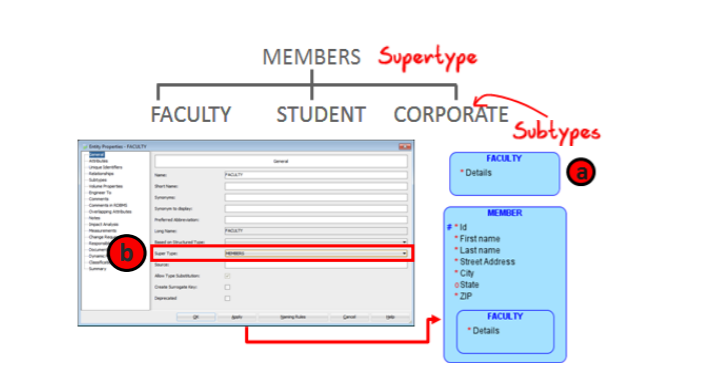
g. Нажмите на значок со стрелкой, чтобы переместить атрибут на правую панель.

h. Нажмите OK

* Объекты с супертипом и подтипом

Чтобы определить объект, как подтип, в Oracle SQL Developer Data Modeler, должен существовать супертип.

В этом примере супертипом является MEMBER. Чтобы создать подтип FACULTY, выполните следующие действия. a. Дважды нажмите на объект, который хотите сделать подтипом. Для примера на слайде нужно сделать FACULTY подтипом супертипа MEMBER. Дважды нажмите на объект FACULTY. b. Выберите объект-супертип MEMBER в списке Super Type и нажмите OK. Теперь объект FACULTY стал подтипом супертипа MEMBER и унаследует все атрибуты супертипа.

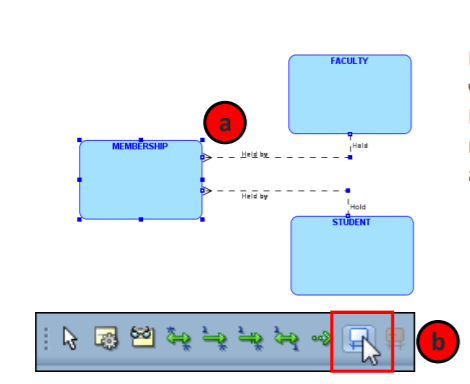


* Дуговые, иерархические, ассоциативные и рекурсивные связи

Создание дуговой связи:

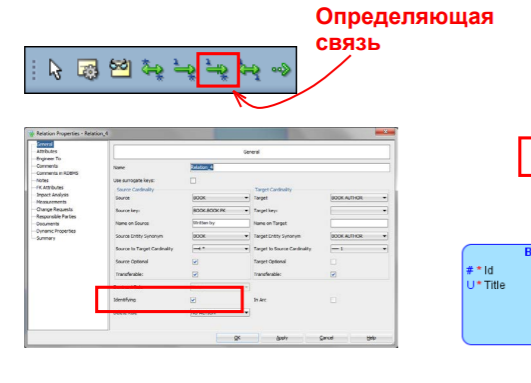
Выполните следующие действия, чтобы создать в Oracle SQL Developer Data Modeler исключающую связь. a. Зажмите клавишу Ctrl и выберите объект пересечения и обе связи, на которых хотите создать дуговую связь.

b. Нажмите на значок New Arc на панели инструментов. Исключающая связь будет создана с дугой.



Создание иерархической связи:

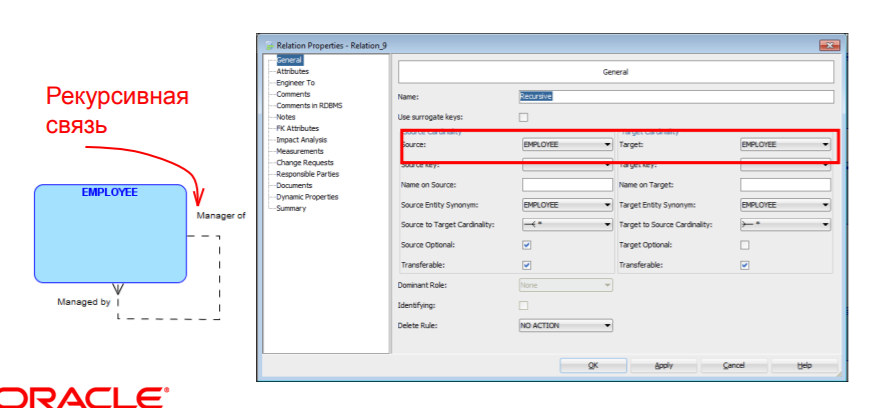
Чтобы добавить ассоциативную связь, выберите на панели инструментов Identifying Relationship и нажмите на исходный и целевой объекты, чтобы добавить связь между ними.



Создание ассоциативной связи:

Иерархические данные можно представить в виде набора связей 1:N (или определяющей связи 1:N). В этом примере университет (University) имеет иерархическую структуру: • Факультет (FACULTY) может состоять из одного или нескольких отделений (DEPARTMENT). • Отделение (DEPARTMENT) может предлагать один или несколько курсов (COURSE). UID для набора иерархических объектов можно подставить через несколько связей, сделав их определяющими.

Создание рекурсивной связи:

Чтобы добавить рекурсивную связь, выберите нужную связь в панели инструментов, как обычно, нажмите на объект, чтобы сделать его исходным, затем повторно нажмите на него, чтобы сделать его целевым.

**4-2 Преобразование логической модели в реляционную**

* **Описание преобразования логической модели в реляционную с помощью Oracle SQL Developer Data Modeler**
* **Перечисление этапов преобразования логической модели в реляционную**
* **Перечисление этапов преобразования логической модели в реляционную с помощью Oracle SQL Developer Data Modeler**

**Ниже перечислены действия по прямому проектированию логической модели в реляционную с помощью Oracle SQL Developer Data Modeler. a. Выберите логическую модель.**

**b. Нажмите на значок Engineer to Relational Model.**

**c. Примите все значения по умолчанию и нажмите Engineer.**

**d. Выберите вкладку Relational, чтобы просмотреть спроектированную реляционную модель.**

**Тема5 — Сопоставление с физической моделью**

**5-1 Сопоставление объектов и атрибутов**

**o Объяснение правил именования, используемых в реляционной базе данных**

**Правила именования для следующих элементов: –**

**Имена таблиц: • В качестве имени таблицы, как правило, используется имя соответствующего объекта во множественном числе. Имена столбцов и специальные символы (%, \*, #, -, пробел, …):**

**• Имена столбцов часто совпадают с именами атрибутов, но есть некоторые исключения. Следует заменять специальные символы нижним подчеркиванием, так как SQL не позволяет использовать большинство специальных символов в именах столбцов. Имена столбцов часто сокращаются.**

**Краткие имена таблиц (сокращения) • Уникальное сокращение для каждой таблицы очень удобно использовать при именовании столбцов внешнего ключа или ограничений внешнего ключа.**

**Ограничения:**

**Имена таблиц и столбцов: – Должны начинаться с буквы – Могут содержать до 30 буквенно-числовых символов – Не должны содержать пробелы и некоторые специальные символы – Не должны содержать зарезервированные слова**

**• Имена таблиц в схеме должны быть уникальными.**

**• Имена столбцов в таблице должны быть уникальными.**

**o Создание глоссария с помощью Oracle SQL Developer Data Modeler и применение стандартов именования**

**По умолчанию имена таблиц создаются непосредственно из имен объектов и не имеют формы множественного числа. • Имена столбцов создаются непосредственно из имен атрибутов, при этом пробелы заменяются нижними подчеркиваниями. • Чтобы применить правила именования к именам таблиц и столбцов, необходимо создать глоссарий.**

**• Глоссарий в Oracle SQL Developer Data Modeler — это набор утвержденных терминов, которые можно использовать при проектировании.**

**• Он гарантирует, что модель соответствует заданным стандартам именования.**

**• Можно создать новый глоссарий, использовать существующие глоссарии или создать глоссарий на основе существующей логической модели.**

**Чтобы создать глоссарий из логической модели, (самый удобный способ) - нужно выполнить следующие действия: a. нажать на кнопку “логическая модель” в обозревателе объектов. b.Создать глоссарий из логической модели. В редакторе глоссариев представлены рекомендуемые формы множественного числа и сокращения, при желании их можно отредактировать. • Имена объектов должны иметь форму множественного числа, а для атрибутов должны быть добавлены требуемые сокращения. Ненужные рекомендации можно удалить. После внесения всех необходимых изменений следует сохранить глоссарий как файл в определенную папку.**

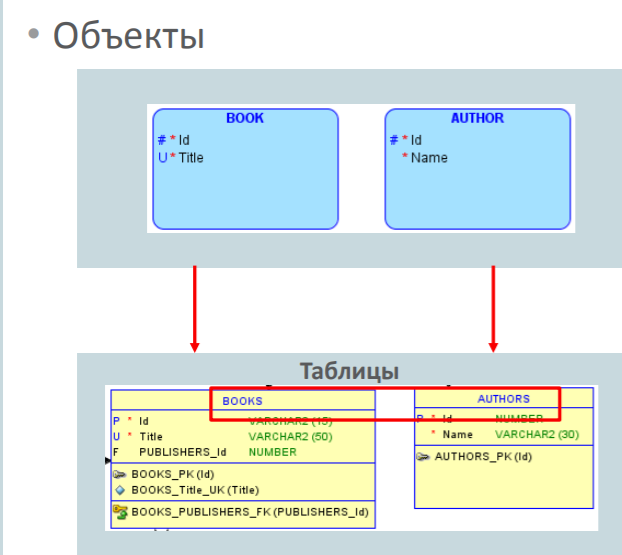
**Чтобы применить глоссарий во время проектирования, необходимо добавить его на одну из вкладок в диалоговом окне. Действия по добавлению глоссария: a. Нажать на модель проекта в обозревателе объектов. b. Выбрать “Свойства”. c. Развернуть настройки- Settings и выбрать узел Naming Standard. d. Нажать на «+» и перейти в папку, где находится глоссарий.**

**Редактирование существующего глоссария осуществляется в разделе “настройки” - редактор глоссария. Оттуда следует просто перейти в папку с нужным файлом глоссария и открыть его.**

**Чтобы применить глоссарий в качестве стандарта именования, нужно: a. Перепроектировать логическую модель еще раз. b. Выбрать вкладку Relational, чтобы посмотреть результаты.**

**▪ Сопоставление объектов и имен таблиц**

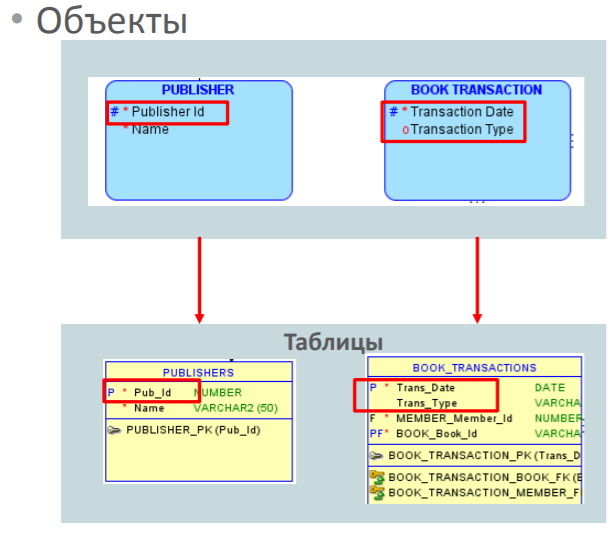
**При применении стандартов именования из глоссария имена объектов логической модели в единственном числе сопоставляются с именами таблиц реляционной модели во множественном числе.**

****

**▪ Сопоставление атрибутов и имен столбцов**

**Имена атрибутов в логической модели сопоставляются с названием столбцов в реляционной модели**

**Если имена атрибутов объекта включают в себя Publisher и Transaction, то после применения стандартов именования из глоссария к именам столбцов применяются сокращения, заданные для Publisher и Transaction, на рисунке это будет выглядеть так:**



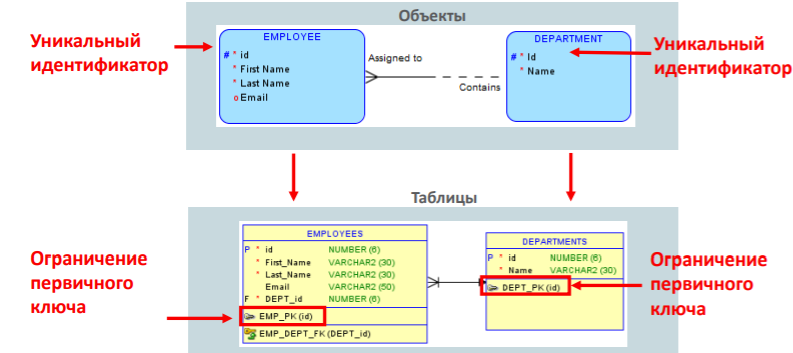
**5-2 Сопоставление первичных и внешних ключей**

**Правила именования • Ограничения обеспечивают применение правил, согласованность и целостность базы данных. Ограничениям следует давать понятные имена, чтобы на них было проще ссылаться.**

**o Выбор правил именования для следующих элементов:**

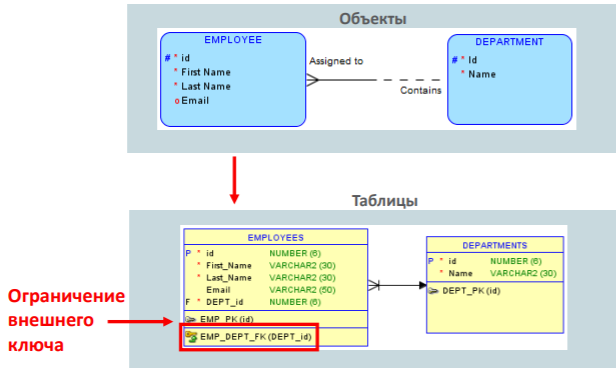
**▪ Имена ограничений первичного ключа**

**Имя ограничения первичного ключа состоит из краткого имени таблицы, нижнего подчеркивания и суффикса PK**

****

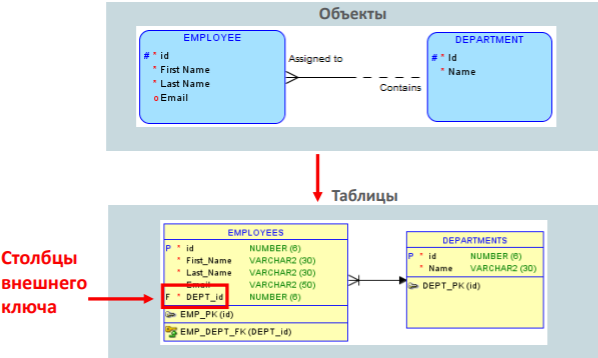
**▪ Имена ограничений внешнего ключа**

**Имя ограничения внешнего ключа состоит из кратких имен обоих таблиц, нижнего подчеркивания и суффикса FK.**

****

**▪ Имена столбцов внешнего ключа**

**Имя столбца внешнего ключа состоит из краткого имени таблицы и имени столбца в таблице, на который ссылается этот внешний ключ.**

****

**По умолчанию имена ограничений создаются из полного имени таблицы в Oracle SQL Developer Data Modeler. По этой причине имена ограничений могут быть очень длинными, сложными и могут превышать максимально допустимое число символов для SQL. Чтобы применить стандарты именования к именам ограничений, можно дополнительно создать и в дальнейшем использовать csv файл, где в 1 столбце следует указать полные имена таблиц, а во втором столбце — используемые сокращения.**

**Раздел 6 — Основы языка SQL**

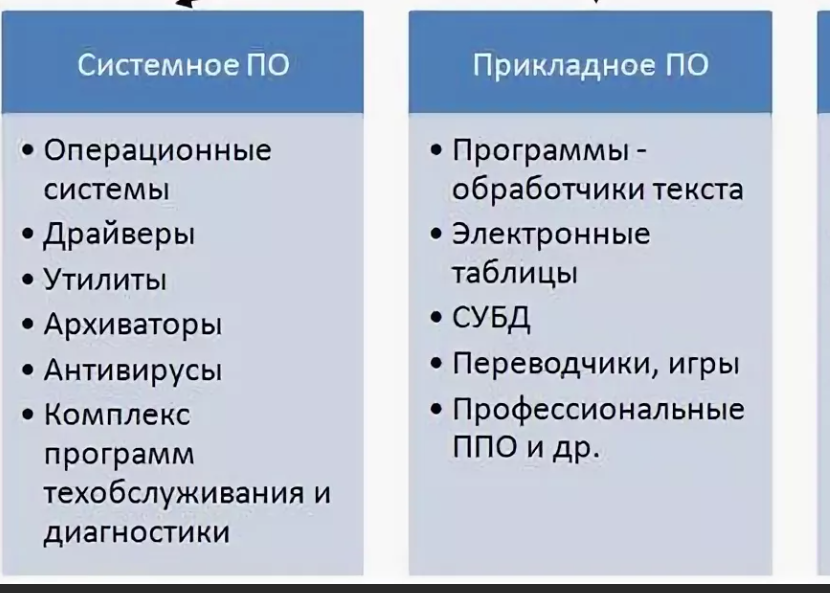
**6-1 Основные сведения об Oracle Application Express**

**o Различия между прикладным и системным программным обеспечением и примеры этих типов программного**

**обеспечения**

**Системное ПО обеспечивает работу компьютера (загрузка ядра, драйверов, проверка портов, синхронизацию устройств и протоколов передачи данных, чтение, передачу данных, осуществляют слежение за устройствами ввода-вывода и реагируют на команды с них и т. д.). Прикладные программы используют системное ПО, но выполняют действия и функции, не влияющие на систему, а имеющие прикладное значение (просмотр и обработка видео, математические вычисления, запись на внешние носители, игры...).**

**Примеры**

****

**o Вход в Oracle Application Express для практической работы**

**Компоненты Oracle Application Express Oracle Application Express включает в себя следующие компоненты: – SQL Workshop – Application Builder – Object Browser • Для изучения SQL используется компонент SQL Workshop. • Для разработки приложений используется Application Builder.**

**o Выполнение простого запроса для извлечения информации из базы данных**

**SELECT \* FROM <имя таблицы>;**

**o Применение правил SQL для отображения всех столбцов и подмножеств столбцов в соответствии с**

**заданными критериями**

**SELECT <имя столбца 1, имя столбца 2, и т. д.> FROM <имя таблицы> WHERE <условие>;**

**6-2 Структурированный язык запросов (SQL)**

**o Описание структуры данных в реляционной базе данных**

**Для управления чтением данных и манипуляциями с данными используется программное обеспечение DBMS.**

**o Описание различных терминологий реляционных баз данных**

**o Определение структурированного языка запросов и его функций**

**SQL — декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных**

**-Создание, замена, изменение и удаление объектов базы данных -Вставка, обновление и удаление строк в таблице**

**- Запрос данных, хранящихся в базе данных**

**- Контроль доступа к базе данных и объектам базы данных -Обеспечение согласованности и целостности базы данных**

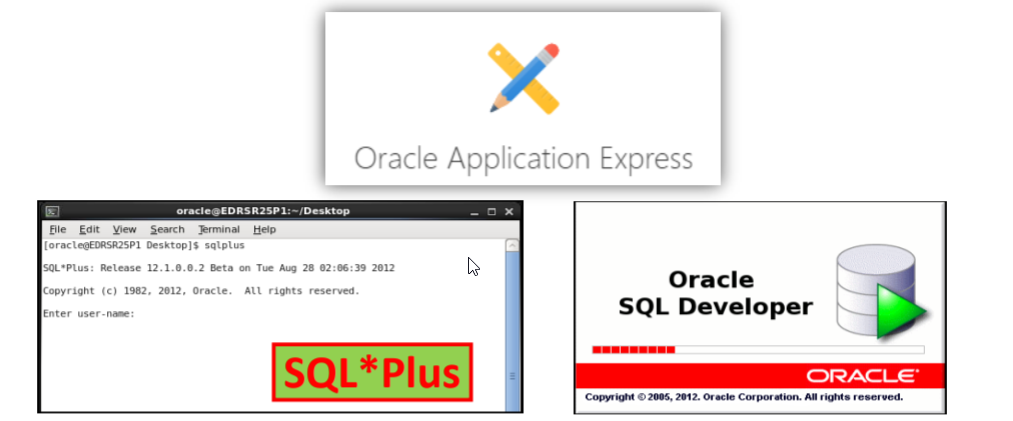
**o Описание процесса обработки SQL**

**Изображение выглядит как стрела

Автоматически созданное описание**

**o Определение средств, используемых для доступа к реляционной базе данных**

**Пользователи могут получать доступ к данным на сервере баз данных Oracle с помощью специализированных программ, установленных на компьютере. Эти программы называются клиентами и используются для отправки инструкций SQL (команд) на сервер**

**Примеры:**

**6-3 Язык описания данных (DDL)**

**o Определение действий для создания таблиц базы данных**

**• Чтобы отправить инструкцию CREATE TABLE, требуются**

**• Полномочие CREATE TABLE**

**• Область хранения**

**Указать:**

**Имя таблицы**

**• Имя столбца, тип данных столбца, размер столбца**

**• Ограничения целостности (необязательно)**

**• Значения по умолчанию (необязательно)**

**Изображение выглядит как текст

Автоматически созданное описание**

**o Описание цели языка описания данных (DDL)**

**определяет структуры баз данных**

**o Перечисление операций DDL для создания и обслуживания таблиц базы данных**

**Изображение выглядит как текст

Автоматически созданное описание**

* rename

**6-4 Язык манипулирования данными (DML)**

**o Описание цели языка манипулирования данными (DML)**

**позволяет манипулировать данными (INSERT, UPDATE, DELETE)**

**o Объяснение операций DML для управления данными в таблице базы данных:**

**▪ Вставка**

**▪ Обновление**

**▪ Удаление**

**Изображение выглядит как текст

Автоматически созданное описание**

**6-5 Язык управления транзакциями (TCL)**

**o Описание цели языка управления транзакциями (TCL)**

**позволяет управлять транзакциями базы данных, чтобы Изменения, выполненные одним пользователем, не конфликтуют с изменениями, выполненными другим пользователем**

**o Описание операций TCL для управления следующими транзакциями:**

**▪ COMMIT**

**▪ SAVEPOINT**

**▪ ROLLBACK**

**Изображение выглядит как текст

Автоматически созданное описание**

**o Описание необходимости согласованности чтения**

**-каждый пользователь видит данные в том виде, в котором они существовали на момент последнего выполнения команды COMMIT**

**Согласованность чтения гарантирует для одних и тех же данных, что: • читатели не ждут писателей;**

**• писатели не ждут читателей;**

**• писатели ждут других писателей.**

**6-6 Извлечение данных с помощью SELECT**

**o Список возможностей инструкций SQL SELECT**

• SELECT определяет столбцы для отображения.

• FROM определяет таблицу, содержащую данные столбцы.

Изображение выглядит как текст

Автоматически созданное описание

**o Создание и выполнение инструкции SELECT, которая:**

**▪ Возвращает все строки и столбцы таблицы**

Можно отобразить все столбцы таблицы с помощью ввода \* после SELECT

Изображение выглядит как стол

Автоматически созданное описание

**▪ Возвращает определенные столбцы таблицы**

Можно использовать инструкцию SELECT для отображения определенных столбцов таблицы, указав имена столбцов в необходимом порядке отображения и разделив их запятой.

Изображение выглядит как стол

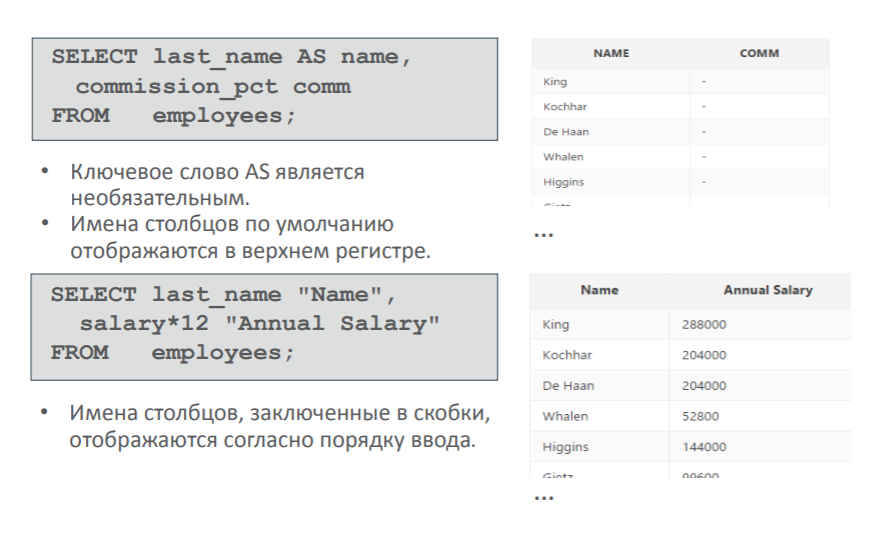
Автоматически созданное описание

**▪ Использует псевдонимы столбцов для отображения описательных заголовков столбцов**

Ключевое слово AS является необязательным.

• Имена столбцов по умолчанию отображаются в верхнем регистре.

• Имена столбцов, заключенные в скобки, отображаются согласно порядку ввода.

****

**▪ Использует арифметические операторы и операторы соединения**

Создайте выражения с числовыми данными и данными дат с помощью арифметических операторов. • В арифметическом выражении можно использовать имена столбцов, числовые константы и арифметические операторы. • Арифметические операторы можно использовать в любой фразе инструкции SQL, кроме FROM.

Изображение выглядит как стол

Автоматически созданное описание

Здесь оператор сложения используется для вычисления повышения заработной платы на $300 для всех сотрудников. В качестве заголовка столбца отображается SALARY + 300.

**Изображение выглядит как стол

Автоматически созданное описание**

Любое арифметическое выражение, содержащее значение null, будет принимать значение null.

Оператор объединения

• Связывает столбцы или строки символов с другими столбцами

• Представлен двумя вертикальными полосами (||)

• Создает столбец, являющийся символьным выражением

• Объединение NULL с символом приводит к созданию строки символов

Изображение выглядит как текст

Автоматически созданное описание

**▪ Использует строки литеральных символов**

Строки литеральных символов

• Литерал — это символ, число или дата, заключенные в инструкцию SELECT.

• Литеральные значения даты и символа должны быть заключены в одинарные кавычки.

• Вывод каждой строки выполняется один раз для каждой возвращаемой строки.

• В этом примере фамилия и job\_id каждого сотрудника объединены с литералом, чтобы придать возвращаемым строкам дополнительное значение.

Изображение выглядит как текст

Автоматически созданное описание

• Многие инструкции SQL используют в выражениях или условиях символьные литералы. Если литерал включает в себя одинарную кавычку, можно использовать оператор кавычки (q) и выбрать собственный разделитель кавычки — в данном случае скобки [].Изображение выглядит как текст

Автоматически созданное описание

**▪ Устраняет дублирование строк**

По умолчанию в запросах отображаются все строки, включая повторяющиеся.

• Чтобы удалить из результатов повторяющиеся строки, включите ключевое слово DISTINCT во фразу SELECT сразу после ключевого слова SELECT

**Изображение выглядит как текст

Автоматически созданное описание**

**o Описание структуры таблицы**

Чтобы отобразить структуру таблицы, включая имя столбца, тип данных и допустимость неопределенного значения, используйте команду DESCRIBE.

• Или выберите Object Browser APEX в SQL Workshop для просмотра структуры таблицы.

• Для просмотра структур таблиц можно также нажать кнопку «Find Tables» в SQL Commands APEX.

Изображение выглядит как стол

Автоматически созданное описание

**6-7 Ограничение данных с помощью WHERE**

**o Ограничение строк с помощью:**

* **Фразы WHERE**

Ограничение возвращенных строк с помощью фразы WHERE:

Изображение выглядит как текст

Автоматически созданное описание

• Если логическое выражение выдает значение true, возвращается строка, отвечающая условию.

• Фраза WHERE следует за фразой FROM.

Изображение выглядит как стол

Автоматически созданное описание

* **операторов сравнения, использующих условия =, <=, >=, <>, >, <, !=, ^=, BETWEEN, IN, LIKE и NULL**

Изображение выглядит как стол

Автоматически созданное описание

Изображение выглядит как стол

Автоматически созданное описание

Используйте оператор BETWEEN для отображения строк на основе диапазона значений:Изображение выглядит как стол

Автоматически созданное описание

\*\* Примечание. При использовании BETWEEN сначала необходимо указывать нижнее значение.

Используйте оператор IN для тестирования значений в списке:

Изображение выглядит как стол

Автоматически созданное описание

\*\* Примечание. Элементы в списке могут быть расположены в любом порядке.

Используйте оператор NOT IN для тестирования значений в списке:

Изображение выглядит как стол

Автоматически созданное описание

Соответствие шаблону: Оператор LIKE

• Используйте оператор LIKE для поиска с подстановочными знаками допустимых значений строки поиска.

Условия поиска могут содержать символы или числа литерала:

– % обозначает ноль или более символов.

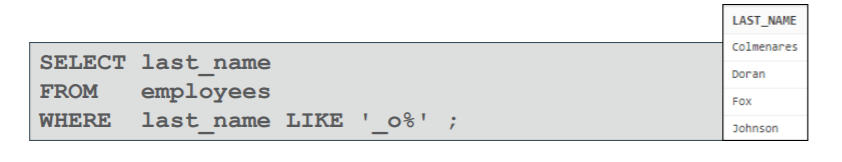
– \_ обозначает один символ.

Изображение выглядит как текст

Автоматически созданное описание

Можно объединить два подстановочных знака (%, \_) с символами литерала для соответствия шаблону:

• Можно использовать идентификатор ESCAPE для поиска фактических символов % и \_.



• При этом будут возвращены записи с SA\_ в job\_id.

Изображение выглядит как текст

Автоматически созданное описание

Использование условий NULL Проверьте наличие значений null с помощью операторов IS NULL или IS NOT NULL:

Изображение выглядит как текст

Автоматически созданное описание

• Проверка с помощью = недоступна, так как значение null не может быть равно или не равно любому значению.

* **логических условий с операторами AND, OR и NOT**

Логическое условие объединяет результат двух компонентных условий для получения единого результата на основании данных условий или, если используется NOT, инвертирует результат единого условия.

Изображение выглядит как текст

Автоматически созданное описание

AND требует, чтобы выполнялись оба компонентных условия:

Изображение выглядит как текст

Автоматически созданное описание

• Примечание. Все поиски символов чувствительны к регистру и должны быть заключены в кавычки.

OR требует, чтобы одно из компонентных условий имело значение true:

Изображение выглядит как стол

Автоматически созданное описание

NOT изменяет значение условия на противоположное:

Изображение выглядит как стол

Автоматически созданное описание

**o Описание правил приоритета для операторов в выражении**

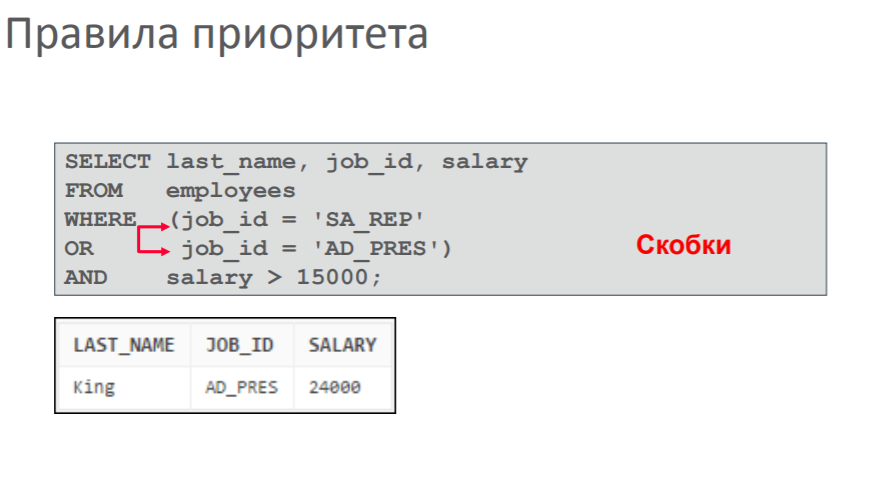
Правила приоритета Чтобы переопределить правила приоритета, необходимо использовать круглые скобки.

**Изображение выглядит как стол

Автоматически созданное описание**

**Изображение выглядит как стол

Автоматически созданное описание**

****

**6-8 Сортировка данных с помощью ORDER BY**

**o Применение фразы ORDER BY для сортировки результатов SQL-запроса**

Сортировка извлеченных строк с помощью фразы ORDER BY :

– ASC: по возрастанию (по умолчанию)

– DESC: по убыванию

Фраза ORDER BY расположена в инструкции SELECT последней: \

Изображение выглядит как текст, снимок экрана, устройство

Автоматически созданное описание

Фраза ORDER BY

• Числовые значения отображаются в порядке убывания или возрастания.

• Отображение значений дат начинается с наиболее ранней даты.

• Значения символов отображаются в алфавитном порядке.

• Значения null отображаются последними в порядке возрастания и первыми в порядке убывания.

• NULLS FIRST указывает, что значения NULL должны быть возвращены перед значениями, не являющимися значениями NULL.

• NULLS LAST указывает, что значения NULL должны быть возвращены после значений, не являющихся значениями NULL.

**o Определение правильного положения для фразы ORDER BY в инструкции SELECT**

Сортировка в порядке убывания:

Изображение выглядит как текст, устройство, датчик, счетчик

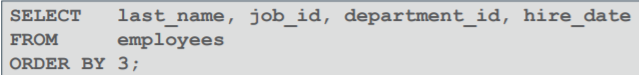
Автоматически созданное описание

Сортировка по псевдонимам столбцов:

Изображение выглядит как текст

Автоматически созданное описание

Сортировка с помощью номера позиции столбца:



Сортировка по нескольким столбцам:

Изображение выглядит как текст, устройство, счетчик

Автоматически созданное описание

**Порядок выполнения**

Предусмотрен следующий порядок выполнения инструкции SELECT:

– Фраза FROM: находит таблицу с данными

– Фраза WHERE: ограничивает возвращаемые строки

– Фраза SELECT: выбирает запрашиваемые столбцы в уменьшенном наборе данных

– Фраза ORDER BY: упорядочивает набор результатов

**o Применение ROWNUM для Top-N-Analysis**

Top-n-analysis представляет собой операцию SQL, выполняемую для ранжирования результатов.

Изображение выглядит как текст

Автоматически созданное описание

**ROWNUM - это виртуальный столбец (не настоящий столбец)**, Данный псевдостолбец возвращает порядковый номер, под которым Oracle выбирает строку из таблицы. Для первой строки значение ROWNUM будет равно 1, для второй - 2, и т.д.

Один из классических примеров использования ROWNUM - ограничение количества получаемых строк из таблицы:

select d.\*

from dishes d

where rownum < 3

Изображение выглядит как стол

Автоматически созданное описание

**o Применение подстановочных переменных во фразе WHERE**

Подстановочные переменные

• При создании отчета пользователям часто требуется динамическое ограничение возвращаемых данных.

• Подстановочные переменные позволяют создать отчеты с запросами на указание пользователями собственных значений, чтобы ограничить диапазон возвращаемых данных.

• Можно встроить подстановочные переменные в файл команд или в отдельную инструкцию SQL. Переменная может быть представлена как контейнер, в котором временно хранятся значения. При выполнении инструкции подставляется сохраненное значение.

Применение подстановочной переменной

• Используйте подстановочные переменные для запроса значений.

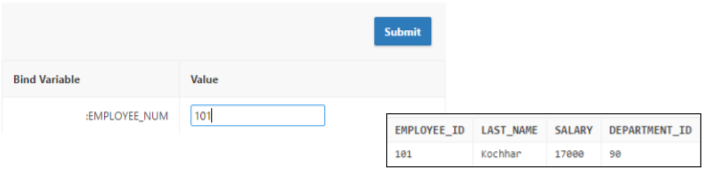
• Используйте переменную с префиксом в виде двоеточия (:) для запроса пользовательского значения:

Изображение выглядит как текст

Автоматически созданное описание

• Когда APEX определяет наличие постановочной переменной в инструкции SQL, отобразится запрос на указание ее значения.

• После ввода значения и нажатия кнопки Submit результаты отобразятся на вкладке Results сеанса APEX.



Можно также использовать значения даты или символов:



**6-9 Соединение таблиц с помощью JOIN**

**o Создание инструкций SELECT для доступа к данным из более чем одной таблицы, используя соединения по**

**эквивалентности и неэквивалентности**

Изображение выглядит как стол

Автоматически созданное описание

Типы соединений

Соединения, совместимые со стандартом SQL:1999:

– Естественное соединение с помощью фразы NATURAL JOIN

– Соединение с помощью фразы USING

– Соединение с помощью фразы ON

– Соединения OUTER:

• LEFT OUTER JOIN

• RIGHT OUTER JOIN

• FULL OUTER JOIN

– CROSS JOIN

Изображение выглядит как текст

Автоматически созданное описание

Определение неоднозначных имен столбцов

• Используйте префиксы таблицы для определения имен столбцов, содержащихся в нескольких таблицах, при устранении неоднозначности.

• Используйте префиксы таблицы для повышения скорости разбора инструкции.

• Используйте псевдонимы таблиц вместо префиксов полного имени таблицы.

• Псевдоним таблицы указывает на краткое имя таблицы, поддерживает краткий код SQL, использует меньший объем памяти.

Изображение выглядит как текст

Автоматически созданное описание

Создание естественных соединений

• Фраза NATURAL JOIN основана на всех столбцах в двух таблицах, имеющих одинаковое имя и тип данных.

• Она выбирает из двух таблиц строки с одинаковыми значениями во всех совпадающих столбцах.

• Если столбцы с одинаковыми именами имеют различные типы данных, возвращается ошибка.

Изображение выглядит как стол

Автоматически созданное описание

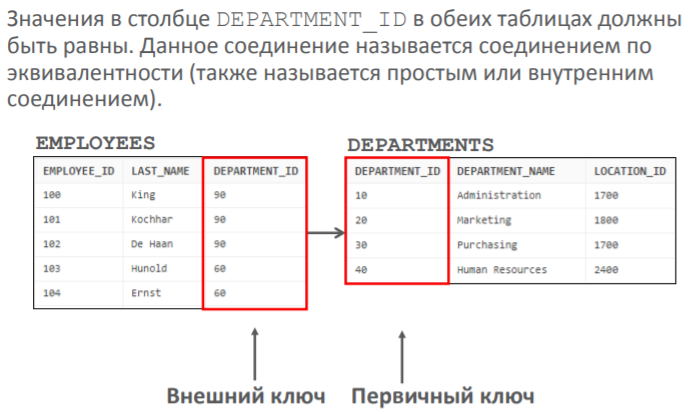
Создание соединений с помощью фразы USING

• Если соединяемые таблицы совместно используют несколько столбцов, все общие поля используются в соединении.

• Используйте фразу USING для указания отдельного столбца для JOIN вместо NATURAL JOIN.

• Фразу USING также можно использовать для сопоставления столбцов с одинаковым именем, но разными типами данных.

• Фразы NATURAL JOIN и USING являются взаимоисключающими.



Изображение выглядит как стол

Автоматически созданное описание

Использование псевдонимов таблиц с фразой USING   
• Не используйте имя таблицы или псевдоним во фразе USING.

• Если псевдоним также используется в других частях инструкции SQL, не присваивайте ему псевдоним.

Изображение выглядит как текст

Автоматически созданное описание

Создание соединений с помощью фразы ON

• NATURAL JOIN создает соединение по эквивалентности для всех столбцов с одинаковым именем и типом данных.

• Используйте фразу ON для указания произвольных условий или указания столбцов для соединения.

• Условие соединения отделяется от других условий поиска.

• Фраза ON упрощает понимание кода.

• Фраза USING создает соединение по

эквивалентности между двумя таблицами с

помощью одного столбца с одинаковым именем,

независимо от типа данных.

• Фраза ON создает соединение по эквивалентности

между двумя таблицами с помощью одного столбца

из каждой таблицы, независимо от имени или типа

данных.

Изображение выглядит как стол

Автоматически созданное описание

Изображение выглядит как стол

Автоматически созданное описание

Изображение выглядит как стол

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как стол

Автоматически созданное описание

Изображение выглядит как стол

Автоматически созданное описание

**o Присоединение таблицы к самой себе путем самосоединения**

**Изображение выглядит как стол

Автоматически созданное описание**

**Изображение выглядит как стол

Автоматически созданное описание**

**o Применение внешних соединений (OUTER JOIN) для просмотра данных, которые обычно не соответствуют**

**условию соединения**

**Изображение выглядит как стол

Автоматически созданное описание**

Соединения INNER и OUTER

• В SQL:1999 соединение двух таблиц, возвращающих

только совпадающие строки, называется соединением

INNER. (фразы NATURAL JOIN, USING, ON)

• Соединение между двумя таблицами, возвращающее

результаты соединения INNER и несовпадающие строки

из таблицы слева (или справа), называется соединением

LEFT OUTER (или RIGHT OUTER).

• Соединение между двумя таблицами, возвращающее

результаты соединения INNER и результаты соединения

OUTER слева и справа, является полным соединением

FULL OUTER.

Изображение выглядит как стол

Автоматически созданное описание

Изображение выглядит как стол

Автоматически созданное описание

Изображение выглядит как стол

Автоматически созданное описание

**o Создание декартова произведения (перекрестного соединения) всех строк из двух и более таблиц**

В декартовом произведении отображаются все комбинации строк. Все строки в первой таблице связаны со всеми строками во второй таблице.

Изображение выглядит как текст

Автоматически созданное описание

• Декартово произведение образуется, когда условие соединения опущено или недопустимо.

• Чтобы предотвратить образования декартова произведения, всегда указывайте допустимое условие соединения.

Изображение выглядит как текст, устройство, закрыть, счетчик

Автоматически созданное описание

Изображение выглядит как стол

Автоматически созданное описание

Фраза CROSS JOIN создает векторное произведение двух таблиц.

• Оно также называется декартовым произведением между двумя таблицами.

Изображение выглядит как текст, устройство, закрыть

Автоматически созданное описание