**Тема 1 — Введение**

**1-2 Общие сведения о базах данных**

o Различия между данными и информацией

*Данными* называют собранные факты по какому-то вопросу или теме. А *информацией* является результат объединения, сравнения и выполнения расчетов с данными. Например, мы вводим в базу данных данные о бюджете, что $1 000 000 в 2015г., а из базы данных мы получаем информацию: «$1 000 000 бюджет на 2015 год».

o Определение базы данных

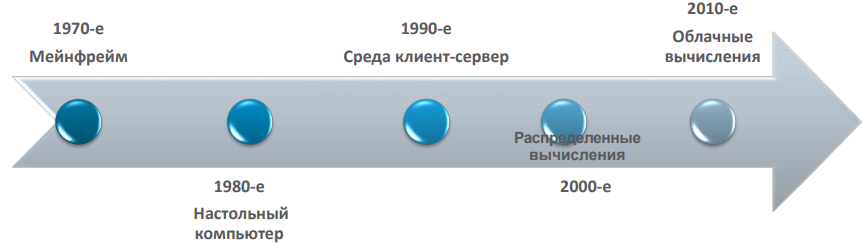
*База данных* – это централизованный структурированный набор данных, хранящихся в компьютерной системе. Она предоставляет средства для извлечения, добавления, изменения и удаления данных по требованию. База данных предоставляет средства для преобразования извлеченных данных в полезную информацию.

o Описание элементов системы управления базами данных (DBMS)

Система управления базами данных (DBMS) – это программное обеспечение, которое контролирует хранение, организацию и извлечение данных. Элементами DBMS являются:

* Управление памятью и хранилищем;
* Словарь данных («центральное хранилище информации о данных»);
* Язык запросов (искусственный язык, на котором делаются запросы к базам данных и информационно-поисковым системам).

o Преобразования в сфере компьютерных технологий



*1970-е: мейнфреймы (централизованная обработка):*

В 1970-е годы совершались попытки построить системы баз данных, интегрируя аппаратное и программное обеспечение. Небольшие компьютеры или «терминалы ввода-вывода» использовались для доступа к мощным мейнфреймам и выполнения команд. Эти терминалы зависели от мейнфрейма и отображали результаты только тогда, когда мейнфрейм завершал обработку. Сами они не обладали значимыми вычислительными мощностями для обработки данных.

*1980-е: настольные компьютеры (локализованная обработка):*

Обработка данных переместилась с мейнфреймов на клиентские компьютеры. ПК, имеющие собственное программное обеспечение и самостоятельно выполняющие некоторую обработку, стали известны как «смарт-клиенты» или «рабочие станции». Вычислительная мощность клиентской машины привела к появлению приложений с графическим интерфейсом пользователя. В эту эпоху были созданы многие приложения, широко распространенные сегодня (Word, Excel, PowerPoint).

*1990-е: среда «клиент-сервер» (централизованная и локальная обработка):*

В среде «клиент-сервер» используется Интернет и быстродействующие серверы для удовлетворения потребностей организаций в хранении данных и получении информации. Программное обеспечении, которое управляет данными, находится на сервере баз данных и выполняет операции по хранению и извлечению данных. Приложения для бизнес-операций находится на сервере приложений и служат для создания документов, разработки, взаимодействия или выполнения операций с данными. Клиенты могут иметь собственные приложения, но доступ к основным бизнес-приложениям осуществляется с клиентом с помощью интернет-браузера.

*2000-е: распределенные вычисления (совместная обработка):*

В модели распределенных вычислений все компьютеры в организации, находящиеся в разных местах, могли использоваться как пул вычислительных ресурсов. В грид-системах создается программная инфраструктура, которая может работать на большом числе сетевых серверов. Пользователи со своих рабочих станций отправляют запросы, чтобы получить информацию или выполнить вычисления, и эти запросы максимально эффективно обрабатываются на компьютерах в гриде-системе.

*2010-е: облачные вычисления (обработка в Интернете):*

Облачные вычисления позволяют предоставлять вычислительные услуги через Интернет. Три основные категории облачных услуг:

* IaaS – позволяет вам арендовать серверы, хранилища, операционные системе 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.

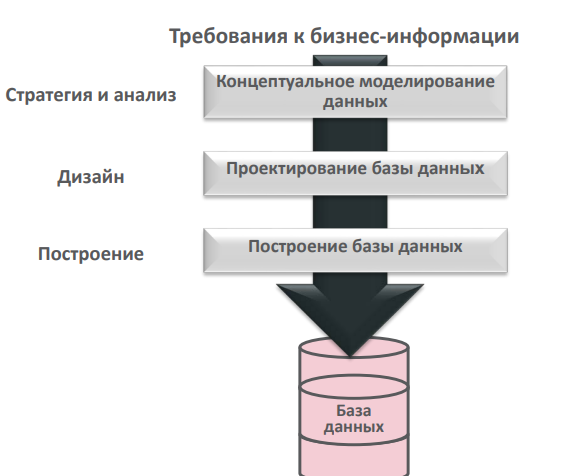
o Определение бизнес-сценариев и отраслевых примеров применения баз данных

Школы и колледжи используют базы данных, чтобы хранить информацию об учебных курсах, учащихся и преподавателях. Банки используют базы данных для хранения информации о клиентах, счетах, займах и транзакциях. Авиакомпании и железные дороги используют онлайн-базы данных для бронирования билетов и для отображения информации о расписании. Телекоммуникационные подразделения хранят в базах данных информацию о коммуникационных сетях, номера телефонов, сведения о вызовах и месячных счетах. В финансовой сфере и торговле базы данных используются для хранения информации, касающейся продаж и покупки акций и облигаций или интернет-трейдинга. Организации используют базы данных для хранения информации о своих сотрудниках, зарплатах, премиях, налогах и создания платежных чеков.

**1-3 Типы моделей баз данных**

o Описание процесса разработки базы данных

Процесс разработки базы данных:



o Описание основных типов моделей баз данных:

* Модель плоских файлов

Базы данных на основе плоских файлов проектируются при использовании одной таблицы. Базы данных на основе плоских файлов, как правило, имеют текстовый формат, где в каждой стоке 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. Запись в иерархической модели БД 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 полей содержит только одно значение данных. Связь указывает на ассоциацию двух записей.

* Объектно-ориентированная модель

Объект моделируется в виде объекта. Для каждого объекта определено состояние и поведение. Связь между объектами осуществляется путем предоставления доступа. Объект должен принадлежать только к одному классу, как экземпляр этого класса. Вы можете наследовать новый класс (подкласс) из существующего класса (суперкласса).

* Реляционная модель

Данные предоставлены 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**

o Объяснение необходимости базы данных

База данных важна для решений нескольких проблем. Таких как базой данных пользуются много пользователей, используется интеграция с разными компонентами и включено много элементов данных. База данных облегчают работу разработчику в несколько раз.

o Объяснение важности бизнес-правил

При проектировании базы данных крайне важно определить и задокументировать бизнес-правила. Бизнес-правила позволяют разработчику или архитектору понять, какие имеются связи и ограничения для участвующих объектов. Они помогают понять процедуру стандартизации, которую применяет организация при обработке огромных объемов данных. Бизнес-правила служат для понимания бизнес-процессов, а также характера, роли и области применения данных. Бизнес-правила помогают классифицировать и проектировать таблицы базы данных.

o Описание принципов и примеров создания бизнес-правил

Бизнес-правила простые и понятные и должны поддерживаться в актуальном состоянии. Разберем пример создания бизнес-правила. SunStar Online Book Rentals — успешная компания, сдающая в аренду книги. В связи с расширением бизнеса библиотеке SunStar потребовалось улучшить имеющуюся информационную систему, чтобы обеспечить поддержку организационных изменений. SunStar пользуется популярностью, и число клиентов быстро растет. Однако клиентская база нестабильна, и это вызывает затруднения. Основная идея — предложить клиентам три типа участия (Gold, Silver, Bronze) и иметь возможность в дальнейшем предлагать дополнительные варианты. Участие уровня Bronze — бесплатное. Участники уровня Silver и Gold должны платить взносы, но получают определенные привилегии. Тип участия не может меняться.

И мы выводим из этого бизнес-правило: участники должны платить взносы. И выводим ограничение: участники должны относиться к одному из трех уровней (Gold, Silver или Bronze).

o Объяснение важности четких формулировок и требований к сбору точной информации

Чтобы производить четкие формулировки и точную информацию, нужно делать концептуальную модель для бизнес-правил. Ведь она точно описывает информационные потребности бизнеса, предотвращает ошибки и неправильное понимание. Это все формирует важную документацию «идеальной системы» и прочную основу для разработки физической базы данных, а также учитывает правила и законы, регулирующие эту отрасль.

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

**2-1 Реляционные базы данных**

o Описание функций отдельной таблицы

Основной и ключевой объект реляционной базы данных это – *таблица*. Базы данных на основе плоских файлов представляют собой тип баз данных, где данные хранятся в одной таблице. БД на основе плоских файлов, как правило, имеют текстовый формат, где в каждой строке содержится только одна запись. У таких баз данных есть свои преимущества – это простота понимания, простота реализации, простота извлечения информации, все записи хранятся в одном месте, простота сортировки и фильтрации записей, низкие требования к аппаратному и программному обеспечению. Но при всем этом есть и недостатки: низкая безопасность, несогласованность данных, избыточность данных, обременительное предоставление информации, медленные базы данных огромного объема.

o Описание функций и правил реляционной базы данных

Реляционная база данных представляет информацию в таблицах со строками и столбцами. Каждый столбец представляет конкретный тип информации, и каждая строка содержит одну запись. Таблицы ссылаются друг на друга с помощью общего поля. Уникальное поле, называемое *ключом*, используется для идентификации каждой записи в реляционной базе данных.

o Описание преимуществ и недостатков типов баз данных

Если говорить про преимущества – это меньше избыточность, мы можем один раз описать объект, и там в действительности будет это отображено. Мы исключаем несогласованность данных, увеличиваем эффективность за счет того, что данные у нас хранятся один раз. Гарантируется также целостность данных, и вопрос о конфиденциальности существенно выше. Но у нее есть и недостатки – это медленный доступ к данным и трудоемкость разработки.

o Описание реляционных таблиц и основных понятий

 Реляционная таблица представляет собой простую структуру, в которой осуществляется организация и хранение данных. Каждая таблица должна иметь особое имя, значение для однозначной идентификации строк, и таблица может содержать несколько строк. Каждый столбец должен иметь уникальное имя. А записи в столбцах должны быть однозначными и должны быть одинакового типа. Порядок строк и столбцов не имеет значения.

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

**2-2 Концептуальные и физические модели данных**

o Описание концептуальной модели данных

Концептуальная модель отражает функциональные и информационные потребности бизнеса, также она удовлетворяет потребности бизнеса, но не отвечает их реализации. Концептуальная модель основана на текущих потребностях, но может отражать будущие потребности. Она определяет важные объекты, которые станут таблицами в БД, и связи между объектами. Но она не определяет атрибуты, которые станут столбцами или полями в БД, и уникальные идентификаторы – атрибуты, которые станут первичными ключами в БД.

o Описание логической модели данных

*Логическая модель* – это модель связей между объектами (ERM), ведь она содержит все объекты и связи между ними. Она иллюстрируется ERD-диаграммой. Логическая модель указывает все атрибуты и уникальные идентификаторы (UID) для каждого объекта, определяет обязательность атрибутов и обязательность и кардинальность связей.

o Описание физической модели данных

*Физическая модель* является расширением логической модели данных. Она указывает определения таблиц, типы данных и точность, определяет представления, индексы и прочие объекты БД. Физическая модель описывает, как должны быть реализованы объекты в конкретной базе данных. Она показывает все структуры таблиц, включая столбцы, первичные ключи и внешние ключи.

Шаги для создания физической модели данных:

1. Моделирование объектов как таблиц
2. Моделирование связей как внешних ключей
3. Моделирование атрибутов как столбцов
4. Изменение модели данных с учетом физических ограничений и требований

o Анализ сходств и различий между концептуальными и физическими моделями данных

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

Чтобы примерно показать разницу у концептуальной и физической модели, рассмотрим следующее предложение. Мечты клиента (концептуальная модель) становятся физической реальностью (физическая модель).

В процессе моделирования физической модели из концептуальной, объекты сменяются на таблицы и связи на внешние ключи.

**2-3 Объекты и атрибуты**

o Определение объектов

*Объектом* можно назвать информацию, которую требуется отслеживать, а также можно назвать наименование предметов, которые вы можете перечислить. Существует несколько типов объектов:

1. Первичный (существует независимо);
2. Характеристика (существует благодаря другому объекту);
3. Пересечение (существует благодаря двум или более объектам).

Объекты содержат экземпляры. Экземпляр объекта – одно вхождение объекта. Объекты представляют набор экземпляров, представляющих интерес для конкретной организации.

o Определение атрибутов

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

o Определение обязательных, необязательных, временных и постоянных атрибутов

Атрибуты классифицируются на два типа: обязательные (не допускается значение null; обозначаются символом \*) и необязательные (допускается значение null; обозначаются строчной буквой o).

Бывают также атрибуты временные и постоянные. Временные атрибуты – нестабильные (пример: Age), а постоянные – стабильные (пример: Birth Date).

Еще атрибуты делятся на простые (атомарные) и составные. Простые атрибуты – это атрибуты, которые нельзя разделить на компоненты. А составные атрибуты – это атрибуты, которые можно разделить на более мелкие компоненты, представляющие собой базовые атрибуты с самостоятельными значениями.

Также атрибуты делятся на однозначные и многозначные. 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).

o Описание обозначений Баркера, Бахмана и обозначений информационной инженерии

Система обозначения Баркера:

1. 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 и должно быть существительным в единственном числе.

1. 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; о – необязательный; # - UID).

1. 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.

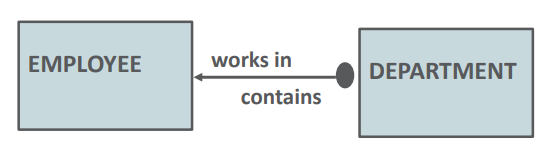
Система обозначения Бахмана:

1. Объект представлен прямоугольником
2. Атрибуты

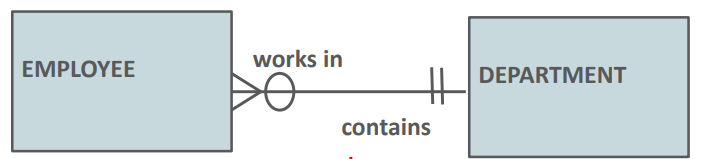
Изображение выглядит как текст

Автоматически созданное описание

1. Линии связей

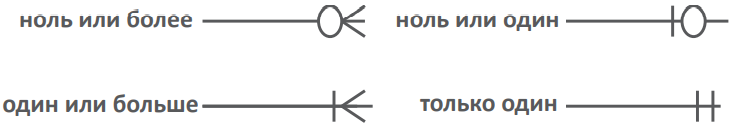


Система обозначений в информационной инженерии:



Любой сотрудник (EMPLOYEE) работает только в одном отделе (DEPARTMENT).

Любой отдел (DEPARTMENT) содержит ноль или больше сотрудников (EMPLOYEES).



Обозначения в модели данных:

Изображение выглядит как стол

Автоматически созданное описание

**2-4 Уникальные идентификаторы**

o Определение уникальных идентификаторов (UID)

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

– Он уникален среди всех экземпляров данного объекта.

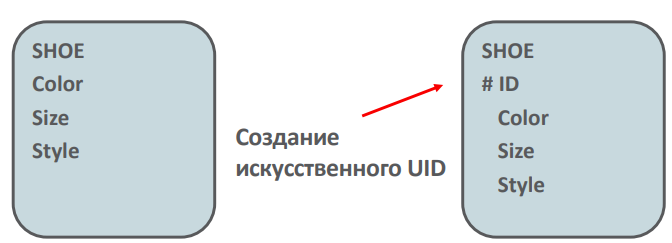
– Его значение не равно NULL для каждого экземпляра объекта в течение времени существования экземпляра.

– Его значение не изменяется в течение времени существования экземпляра.

*UID* — специальный атрибут или группа атрибутов, которая однозначно идентифицирует конкретный экземпляр объекта.

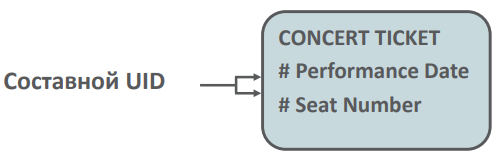
o Определение искусственных уникальных идентификаторов

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



o Определение составных уникальных идентификаторов

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*.



o Определение потенциальных и дополнительных уникальных идентификаторов

Объект может иметь более одного UID.

Изображение выглядит как текст

Автоматически созданное описание

Потенциальные уникальные идентификаторы на примере:

– 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.

o Определение первичных ключей

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**

o Определение и распознавание примеров связей и соответствующих внешних ключей

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

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.

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.

– Обязательность: должна ли существовать связь.

o Определение обязательности связей

Обязательность связей заключается в том, что определяют должна ли существовать связь. Связи бывают необязательные и обязательные. В системе обозначений Баркера обязательность помечается сплошной линией, а необязательность пунктирной линией. В прочтении бизнес-правил используется следующие фразы: необязательно – «может быть» или «может»; обязательно – «должно быть» или «должно».

o Определение кардинальности связей

Кардинальность связей служит мерой количества чего-либо. В связи она определяет степень, с которой один объект связан с другим, отвечая на вопрос «Сколько?». Бывают либо одна и только одна соответствующая запись, либо одна или более соответствующих записей. В системе обозначений Баркера минимальное значение отображается простой линией, а максимальное – птичьей лапкой. В прочтении бизнес-правил используется следующие фразы: линия – «один и только один»; птичья лапка – «один или более».

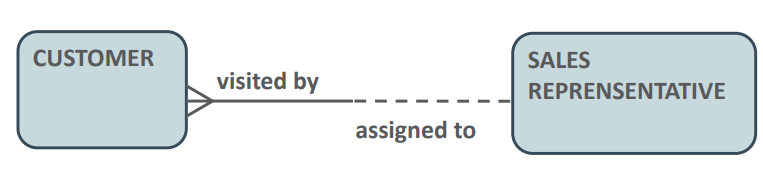
И общий синтаксис бизнес-правил выглядит следующим образом: Каждый объект entity1 {должен быть или может быть} имя связи {«один или более» или «один и только один»} объект entity2.

o Типы связей

Все связи представляют собой информационные требования и правила ведения бизнеса. Типы связей:

* «Многие к одному» (M:1) или «один ко многим» (1:M)

Связи «многие к одному» и «один ко многим» (M:1 и 1:M) имеют кардинальность «один или более» и «один и только один» в другом направлении.



Бизнес-правила:

– Каждого покупатель (CUSTOMER) должен посетить один и только один продавец-консультант (SALES REPRESENTATIVE).

– Каждый продавец-консультант (SALES REPRESENTATIVE) может быть назначен одному или более покупателям (CUSTOMER)

* «Многие ко многим» (M:M)

Связи «многие ко многим» (M:M) имеют кардинальность «один или более» в обоих направлениях.



Бизнес-правила:

– Каждому сотруднику (EMPLOYEE) может быть назначена одна или более работ (JOB).

– Каждую работу (JOB) может выполнять один или более сотрудников (EMPLOYEE).

* «Один к одному» (1:1)

Связи «один к одному» (1:1) имеют кардинальность «один и только один» в обоих направлениях.

Изображение выглядит как стол

Автоматически созданное описание

Бизнес-правила:

– Каждый компьютер (COMPUTER) должен содержать одну и только одну материнскую плату (MOTHERBOARD).

– Каждая материнская плата (MOTHERBOARD) должна содержаться в одном и только одном компьютере (COMPUTER).

* Рекурсивные

*Рекурсивные связи* — это связи объекта с самим собой.

Изображение выглядит как текст

Автоматически созданное описание

Бизнес-правила:

– Каждый сотрудник (EMPLOYEE) может руководить одним или более сотрудниками (EMPLOYEE).

– Каждым сотрудником (EMPLOYEE) должен руководить один и только один сотрудник (EMPLOYEE).

o Матрица связей

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

– Матрица связей показывает, состоит ли объект строки, расположенный слева, в каких-либо отношениях с объектом столбца в верхней части матрицы.

– Все объекты перечислены в левой и верхней частях матрицы.

– Если объект строки связан с объектом столбца, имя этой связи отображается в области пересечения.

– Если объект строки не связан с объектом столбца, область пересечения пуста.

– Каждая связь выше диагональной линии является обратной (служит зеркальным отражением) связи ниже диагональной линии.

– Рекурсивные связи представлены полями на диагонали.

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

o Описание моделирования данных

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

Концептуальная модель базы данных может использоваться для дальнейших обсуждений с разработчиками, администраторами баз данных и разработчиками приложений. Важность концептуальной модели для бизнеса обусловлена тем, что она:

– Точно описывает информационные потребности бизнеса

– Способствует обсуждению

– Предотвращает ошибки и неправильное понимание

– Формирует прочную основу для разработки физической базы данных

– Документирует процессы (также называемые бизнес-правилами) деловой деятельности

– Учитывает правила и законы, регулирующие эту отрасль

Концептуальная модель представляет собой формальную модель, которая:

– Описывает предметы, которые важны для организации (объекты).

– Определяет связи самого высокого уровня между различными объектами, но может содержать или не содержать сведения о кардинальности и допустимости неопределенного значения (null).

– Не определяет атрибуты или уникальный идентификатор для каждого объекта.

o Объяснение понятия «без реализации» (implementation-free) в отношении разработки моделей данных и

проектов баз данных

Логическая модель данных:

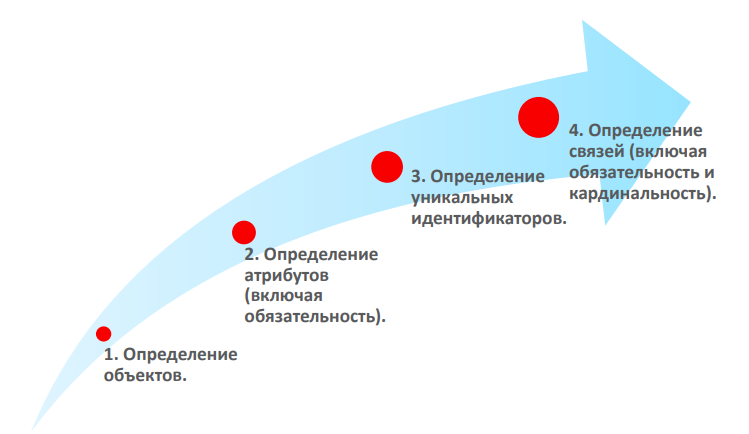
– Описывает данные максимально подробно, независимо от того, как это будет физически реализовано в базе данных.

– Обычно создается на базе концептуальной модели данных.

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

Логическая модель иллюстрируется с помощью ERD.

Этапы создания:



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).

o Перечисление четырех целей моделирования связей между объектами

Четыре цели моделирования связей между объектами:

• Собрать всю необходимую информацию

• Убедиться, что информация отображается только один раз

• Не тратить время на моделирование информации, производной от других, уже смоделированных данных

• Представить информацию в предсказуемой логической форме

o Определение диаграммы «объект-связь» (ERD)

*ERD* — модель, которая определяет понятия или объекты, существующие в системе, а также связи между этими объектами. Она преследует несколько целей:

– Аналитик/проектировщик базы данных в процессе построения ERD лучше понимает, какая информация должна содержаться в базе данных.

– Служит средством документирования.

– Используется для информирования пользователей о логической структуре базы данных. В частности, она эффективно информирует пользователей о логике базы данных.

ERD может использоваться для представления требований организации к данным независимо от типа используемой базы данных и даже при ее отсутствии.

o Сопоставление связей с помощью ERD-инструкций

*ERD-инструкции* — это словарь, который используется для четкого описания бизнес-правил, которые указаны в ERD. Используйте язык ERD-инструкций, чтобы установить связи между объектами в ERD.

o Создание компонентов ERD, представляющих собой объекты и атрибуты, в соответствии с правилами

построения диаграмм

Просто разбейте каждую ERD-инструкцию на ее компоненты.

Компоненты ERD-инструкций:

• EACH

• Entity A

• OPTIONALITY (должен быть/может быть)

• RELATIONSHIP NAME

• CARDINALITY (один и только один/один или более)

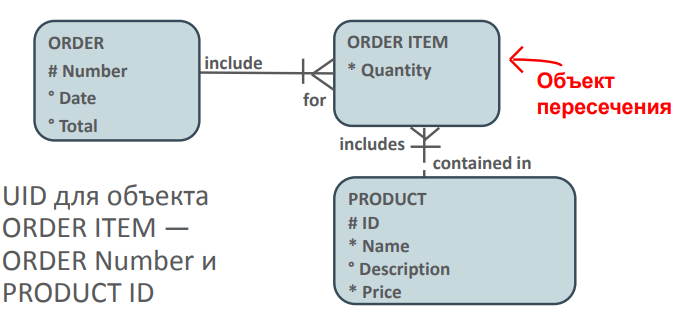
• Entity B

**Тема 3 — Уточнение модели данных**

**3-1 Дополнительные действия со связями**

o Решение связей типа M:M

Атрибуты описывают только объекты. Если атрибуты описывают связь, значит, эту связь необходимо решить. Решение связи M:M с новым объектом пересечения и двумя ассоциативными связями 1:M.



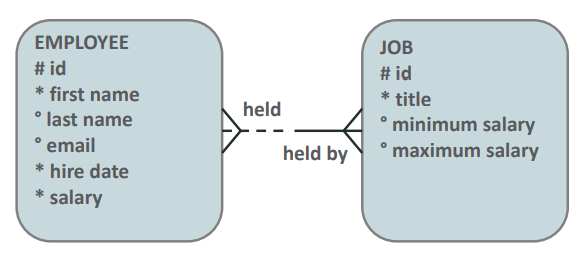
*Характеристики объекта пересечения:*

• Связи из объекта пересечения всегда обязательные.

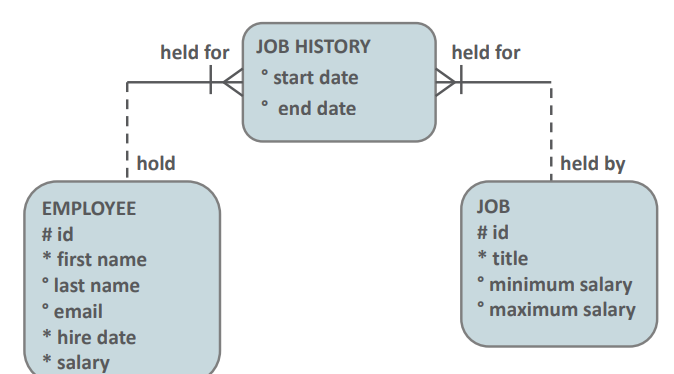
• Объекты пересечения обычно содержат расходные материалы, например использованное количество и даты. Обычно это временные объекты большого объема.

• Объект пересечения определяется по своим двум исходным связям (определяющим связям).

Решение связей M:M с помощью объекта пересечения с ассоциативными связями

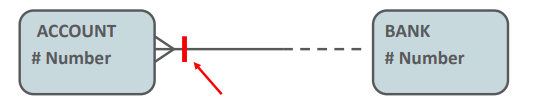


Объекты EMPLOYEE и JOB не хранят историю работ сотрудника.



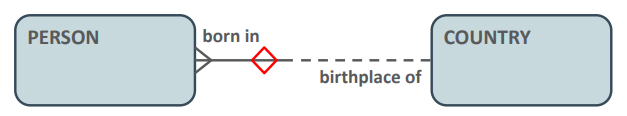
o Определение ассоциативных связей

У ассоциативных связей один объект заимствует как дополнительный уникальный идентификатор один из UID у другого объекта. Ассоциативная связь отображается как параллельная черточка на связи перед заимствующим объектом, и связь должна быть обязательная. Пример:



o Определение и примеры непередаваемых связей

*Перемещаемость* — это возможность связи между двумя экземплярами объекта меняться со временем. Неперемещаемую связь невозможно переместить между объектами, которые она соединяет. Неперемещаемая связь обозначается ромбом. Неперемещаемые связи могут быть только обязательными. Например, страна рождения для человека представляет собой неперемещаемую связь.



o Определение и построение объектов с супертипом и подтипом

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. 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.

Пример:Изображение выглядит как текст

Автоматически созданное описание

Характеристика подтипа:

– Наследует все атрибуты супертипа.

– Наследует все связи супертипа.

– Обычно имеет собственные атрибуты или связи.

– Извлекается внутри супертипа.

– Никогда не существует один.

– Может иметь собственные подтипы.

– Имеет одинаковые первичные ключи супертипа и подтипа.

*Правильное определение подтипов:*

• Является ли этот подтип разновидностью супертипа?

• Учтены все возможные случаи? (исчерпывающий)

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

• Каждый экземпляр соответствует только одному подтипу? (взаимоисключающий)

Каждый экземпляр супертипа принадлежит только одному подтипу.

o Определение иерархических, рекурсивных и дуговых связей

Моделирование иерархических данных:

Изображение выглядит как текст, внешний, знак, снимок экрана

Автоматически созданное описание Один объект включает в себя один объект, а другой – в этот объект и т.д. Связь между такими должна быть от одной обязательная и с птичьей лапкой. У каждого такого объекта должна быть связь намертво с предыдущим объектом.

*Иерархическая модель данных* – это модель данных, где используется представление базы данных в виде древовидной структуры, состоящей из объектов различных уровней.

*Рекурсивные связи* – это связь, при которой экземпляр объекта связан с другим экземпляром в этом же объекте. Она 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. Связи, включенные в дугу, обозначены кружком на линии дуги связи.

Характеристики дуговой связи:

– Связи в дуге часто имеют одинаковое название.

– Связи в дуге должны быть либо все обязательные, либо все необязательные, и должны иметь одинаковую кардинальность.

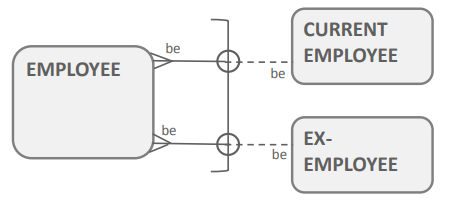
– Дуга принадлежит одному объекту и должна содержать только те связи, которые исходят из него.

– Объект может иметь несколько дуг, но каждая связь может быть частью только одной дуги.

– Дуговая связь представлена дугообразной линией, соединяющей две или более линий связи.

Объект-супертип и его подтипы можно моделировать как дуговую связь.

Пример: объект EMPLOYEE является либо CURRENT EMPLOYEE, либо EX-EMPLOYEE, но не может быть и тем, и другим одновременно (это также можно смоделировать как супертип/подтип).



**3-2 Отслеживание изменений данных**

o Отслеживание изменений данных с течением времени

Моделирование данных на протяжении времени:

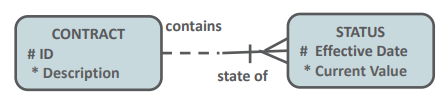
* Добавление исторических данных

Большинству компаний необходимо отслеживать исторические данные, чтобы выявлять тенденции и шаблоны, которые могут послужить основой для бизнес-инноваций или улучшения процессов. 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. Нужно добавить в модель объекты и связи для размещения исторических данных.

* Отслеживание значений атрибута

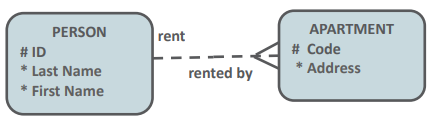
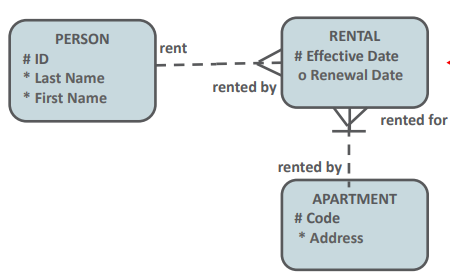
Нужно создать дополнительные объекты, чтобы отслеживать значения атрибутов на протяжение времени.

Изображение выглядит как текст

Автоматически созданное описание> 

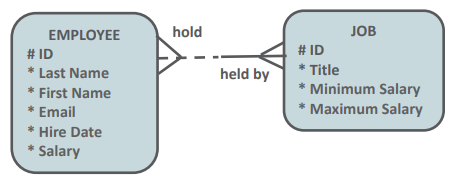
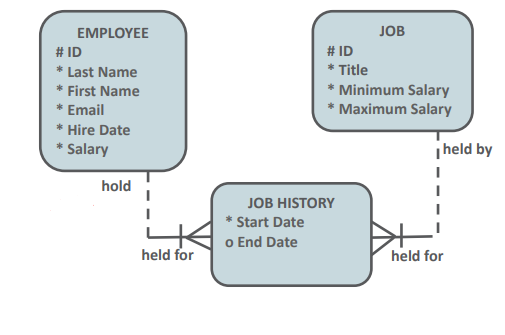
* Изменение связи

Нужно добавить объект для добавления связи, которая может со временем измениться.

> 

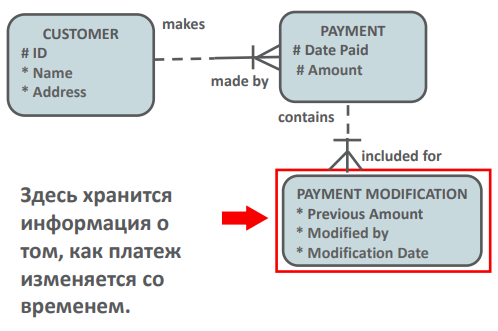
* Создание объектов пересечения

Решение связей M:M с помощью объекта пересечения часто используется для отслеживания информации о связи, которая меняется со временем.

> 

* Ведение журнала измененной информации

Дополнительные объекты и атрибуты хранят измененную информацию.



**3-3 Нормализация и бизнес-правила o Объяснение нормализации**

o Описание нормальных форм

*Нормализация* — концепция реляционной базы данных, но ее принципы применимы к моделированию данных. Целью является нормализация данных до 3NF перед преобразованием модели в реляционный проект. Основные типы нормальных форм по Э.Ф. Кодду:

• Первая нормальная форма (1NF)

Для первой нормальной формы необходимо наличие атрибутов без нескольких значений. Чтобы проверить пригодность для 1NF, нужно проверить, что каждый атрибут имеет только одно значение для каждого экземпляра объекта.

• Вторая нормальная форма (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.

• Третья нормальная форма (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.

o Проверка данных с использованием нормализации

Первая нормальная форма (1NF) - Если атрибут имеет несколько значений, создайте дополнительный объект и свяжите его с исходным с помощью связи 1:M. Каждый атрибут должен иметь только одно значение для каждого экземпляра объекта.

Вторая нормальная форма (2NF) - Если атрибут не зависит от всего UID, нужно создать дополнительный объект с частичным UID. Атрибут должен зависеть от полного UID его объекта.

Третья нормальная форма (3NF) - Необходимо переместить в новый объект все атрибуты, не являющиеся UID, которые зависят от другого атрибута, не являющегося UID. Каждый атрибут зависит только от UID своего объекта. Нужно переместить в новый объект все атрибуты, не являющиеся UID, которые зависят от другого атрибута, не являющегося UID.

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)

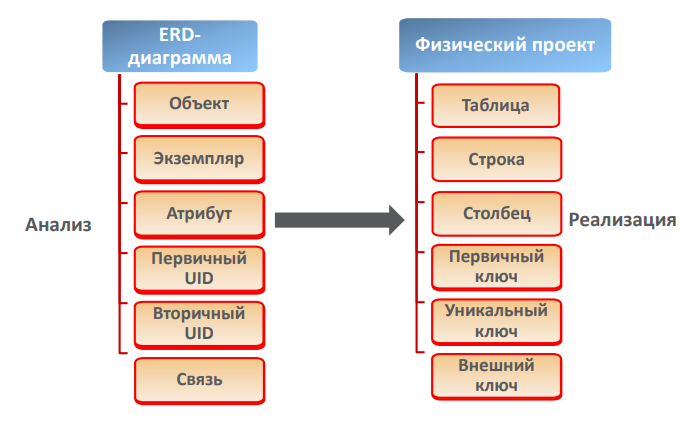
• Прямые собеседования с конечными пользователями

Структурные бизнес-правила 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 правила определяют предварительные требования, шаги, процессы или требования 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

*Диаграмма таблицы* — это дополнительная документация, которая часто используется для более подробного объяснения ключей и столбцов в физической базе данных.

**Правила именования таблиц:**

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

Уникальное краткое имя для каждой таблицы полезно при именовании столбцов внешних ключей.

Создайте краткие имена на основе следующего:

– Имена объектов, содержащие более одного слова

– Имена объектов, содержащие одно слово, состоящее более чем из одного слога

– Имена объектов, содержащие один слог, состоящий более чем из одного символа

**Правила именования столбцов:**

Имена столбцов аналогичны именам атрибутов, за тем исключением, что специальные символы и пробелы заменяются на подчеркивания, что было сказано ранее. 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

Связи сопоставляются между первичными ключами и внешними ключами, чтобы одна таблица могла ссылаться на другую. Связь создает в таблице один или несколько столбцов внешнего ключа на стороне связи «многие». Для именования столбца внешнего ключа используется краткое имя таблицы.

Ассоциативная связь сопоставлена со столбцом внешнего ключа на стороне «многие», как любая другая связь M:1. В этом случае столбец внешнего ключа играет двойную роль, так как он также входит в состав первичного ключа.

Связь M:M решается объектом пересечения, который сопоставляется с таблицей пересечения. Эта таблица пересечения будет содержать столбцы внешнего ключа, связанные с исходными таблицами.

При преобразовании связи 1:1 создается внешний ключ и уникальный ключ. Все столбцы этого внешнего ключа также входят в состав уникального ключа. Если связь обязательная с одной стороны, в соответствующей таблице создается внешний ключ.

Объект, имеющий дугу, будет сопоставлен с таблицей, содержащей внешние ключи из таблиц на конце связей «один». Даже хотя связи в дуге обязательные на стороне «многие», итоговые внешние ключи должны быть необязательными (так как один из них всегда будет пустым). Это может обеспечить ограничение проверки значений, хранящихся в базе данных.

Объекты-супертипы и объекты-подтипы можно сопоставлять несколькими способами:

– Реализация в одной таблице: создается одна таблица, независимо от количества подтипов. Используется, когда большинство атрибутов и связей общие и поэтому находятся на уровне супертипа.

Одна таблица содержит по одному столбцу для каждого атрибута супертипа, а также исходный необязательный характер атрибута. Также таблица содержит столбец для каждого атрибута, принадлежащего подтипу, но все столбцы становятся необязательными. Кроме того, необходимо создать обязательный столбец, который будет выступать столбцом-дискриминатором для различения разных типов подтипов объекта.

– Реализация в двух таблицах: для каждого подтипа создается по таблице (поэтому может быть больше двух таблиц). Используется, когда у подтипов мало общего, мало общих атрибутов и связей.

Одна таблица предназначена для одного подтипа первого уровня. Каждая таблица содержит по одному столбцу для каждого атрибута супертипа, а также его исходный необязательный характер. Каждая таблица также содержит по одному столбцу для каждого атрибута подтипа и его исходный необязательный характер.

**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:

*Oracle SQL Developer Data Modeler* – это инструментарий проектирования баз данных. С его помощью разработчики смогут быстро и просто проектировать модели данных для Oracle Database.

* Объекты, атрибуты и UID с правильной обязательностью и кардинальностью

Чтобы создать объект в Data Modeler, нужно на панели инструментов нажать на New Entity и выбрать, где разместить объект на листе Logical. Далее появится окно Entity Properties, где нужно ввести имя объекта. Чтобы добавить атрибуты в объекте, нужно том же окне Entity Properties перейти на Attributes, нажать на Add an Attribute, далее нужно вести имя, в поле Data Type выбрать Logical и выбрать нужный тип. Дальше вводим размер атрибута и, если атрибут обязательный, то устанавливаем соответствующий флажок. Чтобы установить первичный UID, нужно в панели Attributes выбрать атрибут в качестве первичного UID и установить флажок Primary UID. А чтобы создать вторичный UID, нужно в Unique Identifiers нажать на Add, далее нажать на Properties, ввести имя вторичного UID, далее в окне Key Properties выбрать Attributes and Relations, там нужно выбрать необходимый атрибут и нажать на значок со стрелкой.

* Объекты с супертипом и подтипом

Чтобы определить объект, как подтип, необходимо нажать дважды на этот объект. Далее в списке Super Type выбрать его объект-супертип.

* Дуговые, иерархические, ассоциативные и рекурсивные связи

Чтобы создать дуговые связи, нужно зажать клавишу Ctrl и выбрать объект пересечения и обе связи, на которых хотим установить дуговую связь. Если хотим создать иерархическую связь, то ее можно представить в виде набора связей 1:N. Чтобы добавить ассоциативную связь, нужно на панели инструментов Identifying Relationship и нажать на исходный и целевой объекты, чтобы добавить связь между ними. А чтобы создать рекурсивную связь, нужно выбрать нужную связь на панели инструментов, нажать на объект и затем снова на него нажать.

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

o Описание преобразования логической модели в реляционную с помощью Oracle SQL Developer Data Modeler

*Прямое проектирование* — это процесс преобразования логической модели данных в реляционную модель. В Oracle SQL Developer Data Modeler физическая модель представлена реляционной.

*Преимущества создания реляционной модели:*

– Ближе к решению внедрения.

– Способствует обсуждению.

– Формирует основу для разработки физической базы данных.

Идеальную модель можно адаптировать в модель системы управления реляционными базами данных (RDBMS).

|  |  |
| --- | --- |
| АНАЛИЗ  Логическая модель  Объект  Атрибут  Экземпляр  Первичный UID  Вторичный UID  Связь  Бизнес-ограничения | ПРОЕКТИРОВАНИЕ  Реляционная модель  Таблица  Столбец  Строка  Первичный ключ  Уникальное ограничение  Внешний ключ  Ограничения проверки |

Здесь представлено сопоставление терминологий.

o Перечисление этапов преобразования логической модели в реляционную

Существует три способа моделирования данных:

• *Моделирование сверху вниз* — это один из способов проектирования новой базы данных.

• *Моделирование снизу вверх* — это способ создания базы данных на основе извлечения метаданных из существующей базы данных или использования кода языка описания данных (DDL), полученного из внедрения существующей базы данных.

• *Целевое моделирование* лучше всего подходит для адаптации базы данных к новым требованиям.

o Перечисление этапов преобразования логической модели в реляционную с помощью Oracle SQL Developer

Data Modeler

Чтобы создать реляционную модель, нужно выбрать логическую модель, нажать на значок Engineer to Relational Model, далее нужно принять все значения по умолчанию и нажать на Engineer. Затем во вкладке Relational смотрим нашу спроектированную реляционную модель. Чтобы создать логическую модель, нужно выбрать реляционную, нажать на Engineer to Logical Model, также принять все значения по умолчанию и нажать на Engineer.

**Тема 5 — Сопоставление с физической моделью**

**5-1 Сопоставление объектов и атрибутов**

o Объяснение правил именования, используемых в реляционной базе данных

*Необходимость создания физической модели:*

• Является прототипом фактической реализации базы данных.

• Может использоваться в качестве основы для реализации любых типов DBMS.

• Идеальную модель можно адаптировать для модели RDBMS.

В Oracle SQL Developer Data Modeler физическая модель называется *реляционной моделью*.

*Правила именования:*

– В качестве имени таблицы, как правило, используется имя соответствующего объекта во множественном числе.

– Имена столбцов часто совпадают с именами атрибутов, но есть некоторые исключения. Заменяем специальные символы (%, \*, #, -, пробел, …) нижним подчеркиванием. Имена столбцов часто сокращаются.

– Уникальное сокращение для каждой таблицы очень удобно использовать при именовании столбцов внешнего ключа или ограничений внешнего ключа.

*Ограничения именования в базе данных Oracle:*

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 и имена столбцов в таблице должны быть уникальными.

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

Определить стандарты именования в Oracle SQL Developer Data Modeler можно несколькими способами.

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, в том числе при перепроектировании логической модели в реляционную. Можно создать новый глоссарий, использовать существующие глоссарии или создать глоссарий на основе существующей логической модели.

Чтобы создать глоссарий из логической модели, нужно нажать правой кнопкой мыши Logical Model в обозревателе объектов, далее выбрать Create Glossary from Logical Model. В редакторе глоссариев представлены рекомендуемые формы множественного числа и сокращения. Чтобы внести изменения, нажмите в поле и отредактируйте или удалите текст. Имена объектов должны иметь форму множественного числа, а для атрибутов должны быть добавлены требуемые сокращения. После внесения всех необходимых изменений нужно нажать Save, дальше нужно ввести имя файла для глоссария и перейти в папку для сохранения файла, и снова нажать Save.

Чтобы применить глоссарий во время проектирования, необходимо добавить его на вкладку Naming Standard в диалоговом окне Preferences. Чтобы добавить глоссарий, нужно нажать правой кнопкой мыши на модель проекта в обозревателе объектов, выбрать Properties, развернуть Settings и выбрать Naming Standard, далее нужно нажать на зеленый значок «+» и перейти в папку, где находится глоссарий.

Чтобы редактировать глоссарий, нужно нажать на Tools, далее на Glossary Editor и перейти в папку, где находится файл глоссария, и нажать Open.

Чтобы применить глоссарий в виде стандарта именования, нужно перепроектировать логическую модель еще раз и выбрать вкладку Relational, чтобы увидеть 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

Если имена атрибутов объекта включают в себя название объекта, то после применения стандартов именования из глоссария к именам столбцов применяются сокращения, заданные для данного объекта.

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

o Выбор правил именования для следующих элементов:

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

▪ 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.

*Создание сокращенных имен:*

Чтобы применить стандарты именования к именам ограничений, создайте файл .csv в приложении для работы с электронными таблицами. • В первом столбце укажите полные имена таблиц, а во втором столбце — используемые сокращения. Сохраните файл с расширением .csv и запомните его расположение.

Чтобы применить сокращения к модели, необходимо задать образец в шаблоне именования. Для этого необходимо нажать правой кнопкой мыши на проект в обозревателе объектов, выбрать Properties, развернуть Settings и Naming Standard и выбрать Templates. Там можно отредактировать шаблон, чтобы использовать сокращенные имена для первичных и внешних ключей и внешнего ключа столбца. Далее нужно добавить в шаблон элемент abbr.

Чтобы применить созданные сокращенные имена, нужно выбрать Tools, далее выбрать Name Abbreviations, перейти к файлу .csv, в котором хранятся сокращения, и снять флажок Tables, чтобы сохранить существующие имена из глоссария. Теперь в реляционной модели есть имена ограничений, созданные в соответствии со стандартами именования.

**Тема 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

Несмотря на то, что компьютеры появились очень давно, они не использовались в рабочих и личных целях до тех пор, пока не были разработаны компьютерные программы. Прикладные программы позволяют конечным пользователям приобретать полностью готовые к работе программы. Сейчас уже не обязательно знать, как работает программа. Главное, чтобы она работала и выполняла нужные задачи. Yahoo.com использует базу данных Oracle для хранения данных. Чтобы каждому, кто хочет найти информацию в базе данных или получить электронное письмо, не приходилось изучать SQL, в приложении запрограммированы все элементы SQL. Пользователь может получить доступ к нужной информации всего в несколько кликов.

Приложение можно сравнить с автомобилем. Чтобы управлять автомобилем, необходимо знать, как это делается. Автомобиль имеет удобную «оболочку», которая скрывает все, что не нужно знать водителю.

o Вход в Oracle Application Express для практической работы

*Oracle Application Express* — это средство разработки, развертывания и обслуживания веб-приложений. 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

Чтобы выполнить простой запрос для извлечения информации из базы данных, нужно просто ввести команду SELECT \*, она возвращает все строки таблицы. Ее синтаксис:

SELECT \*

FROM <имя таблицы>;

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

заданными критериями

Чтобы SELECT возвращал набор данных, нужно будет ввести:

SELECT <имя столбца 1, имя столбца 2, и т.д.>

FROM <имя таблицы>

WHERE <условие>;

При вводе команд SQL важно соблюдать правила написания, иначе появится сообщение об ошибке. Чтобы устранить ошибку, нужно исправить написание и запустить команду снова.

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

o Описание структуры данных в реляционной базе данных

Для хранения данных используется двухмерная матрица, известная как таблица. Для управления чтением данных и манипуляциями с данными используется программное обеспечение DBMS.

o Описание различных терминологий реляционных баз данных

Терминология реляционных баз данных:

*Строка* – определяет экземпляр объекта, хранит информацию;

*Уникальный идентификатор* – первичный ключ, однозначно определяет запись

*Поля таблицы* - хранят информацию

*Внешний ключ* – поле ссылается на другую таблицу

*Ячейка* – на пересечении строки и столбца, содержит одно значение, может быть пустым (null)

o Определение структурированного языка запросов и его функций

В реляционной базе данных не указывается маршрут для доступа к таблицам, и вам не нужно знать физическую структуру данных. Чтобы получить доступ к базе данных, необходимо выполнить инструкцию SQL. *SQL* — это язык стандарта American National Standards Institute (ANSI), используемый для работы с реляционными базами данных. Структурированный язык запросов (SQL) — это декларативный язык на основе наборов, используемый для доступа к базе данных Oracle. SQL предоставляет интерфейс для реляционной базы данных и инструкции для работы с базой данных. У нее есть свои преимущества – это эффективность, простота в освоении и использовании, функциональность.

*Функции SQL:*

* 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.

*Типы команд SQL:*

• DDL (язык описания данных) — определяет структуры баз данных

• DML (язык манипулирования данными) — позволяет манипулировать данными (INSERT, UPDATE, DELETE)

• DQL (язык запросов данных) — позволяет выбирать данные (SELECT)

• DCL (язык управления данными) — позволяет контролировать доступ пользователей

• TCL (язык управления транзакциями) — позволяет управлять транзакциями базы данных

o Описание процесса обработки SQL

*Этапы обработки SQL:*

1. Разбор
2. Оптимизация
3. Создание источника строк
4. Выполнение

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.

*Программы*: Oracle Application Express, SQL\*Plus, Oracle SQL Developer

**6-3 Язык описания данных (DDL)**

o Определение действий для создания таблиц базы данных

Таблица - (Представляет собой базовую единицу хранения; состоит из строк);

Представление - (Логически представляет подмножества данных из одной или нескольких таблиц);

Последовательность - (Создает числовые значения);

Индекс - (Повышает эффективность некоторых запросов);

Синоним - (Присваивает альтернативное имя объекту).

*Имена таблиц и столбцов должны:*

• Начинаться с буквы

• Содержать от 1 до 30 символов

• Содержать только символы A–Z, a–z, 0–9, \_, $ и #

• Не дублировать имена других объектов, созданных тем же пользователем

• Не являться словами, зарезервированными для сервера Oracle

Чтобы создать таблицу базы данных, нужно использовать команду CREATE TABLE. Ее синтаксис:

CREATE TABLE [schema.]table

(column datatype [DEFAULT expr][, ...]);

Чтобы проверить, создана ли таблица, выполните команду DESCRIBE.

*Пример создания таблицы:*

CREATE TABLE dept

(deptno NUMBER(2),

dname VARCHAR2(14),

loc VARCHAR2(13),

create\_date DATE DEFAULT SYSDATE);

*Пример подтверждения создания таблицы:*

DESCRIBE dept;

Параметр DEFAULT указывает значение по умолчанию для столбца во время выполнения инструкции CREATE TABLE. Этот параметр предотвращает добавление значений null в столбцы, когда при вставке строки не указано значение столбца. Имя другого столбца или псевдостолбца — недопустимые значения. Тип данных по умолчанию должен совпадать с типом данных столбца.

*Пример:*

CREATE TABLE hire\_dates

(id NUMBER(8),

hire\_date DATE DEFAULT SYSDATE);

*Добавление ограничений:*

Ограничения применяют правила на уровне таблицы. Ограничения обеспечивают согласованность и целостность базы данных. Допускаются следующие типы ограничений:

– NOT NULL (Столбец не может содержать значение null)

– UNIQUE (Значения столбца или комбинации столбцов должны быть уникальными для всех строк в таблице.)

– PRIMARY KEY (Столбец (или комбинация столбцов) должны содержать уникальное значение AND IS NOT NULL для всех строк.)

– FOREIGN KEY (Столбец (или комбинация столбцов) должны создавать и применять ссылки на столбец или комбинацию столбцов в другой (или той же) таблице)

– CHECK (Условие должно иметь значение true.)

Ограничению можно присвоить имя. Рекомендуется давать ограничениям понятные имена, чтобы на них было проще ссылаться. Определять ограничение нужно на уровне столбца или таблицы. Ограничения на уровне столбца включаются при создании столбца. Ограничения на уровне таблицы определяются на последнем этапе создания таблицы. Эти ограничения должны ссылаться на столбец (или столбцы), которым они принадлежат.

o Описание цели языка описания данных (DDL)

Создание таблиц относится к языку описания данных SQL.

Язык определения данных (Data Definition Language) предназначен для создания, изменения и удаления объектов базы данных. *Инструкции DDL:*

– ALTER: изменение структуры объекта

– DROP: удаление объекта из базы данных

– RENAME: переименование объекта базы данных

o Перечисление операций DDL для создания и обслуживания таблиц базы данных

Инструкция ALTER TABLE позволяет изменить структуру таблицы:

• Добавить столбец

• Изменить определение существующего столбца

• Указать значение по умолчанию для нового столбца

• Удалить столбец

• Переименовать столбец

• Присвоить таблице статус «только чтение»

С помощью инструкции ALTER TABLE можно добавить, изменить или удалить столбцы:

ALTER TABLE table

ADD (column data type [DEFAULT expr]

[, column data type]...);

ALTER TABLE table

MODIFY (column data type [DEFAULT expr]

[, column data type]...);

ALTER TABLE table

DROP (column [, column] …);

Для добавления столбцов используется фраза ADD:

ALTER TABLE dept

ADD job\_id VARCHAR2(9);

Можно изменить тип данных, размер данных и значение по умолчанию для столбца:

ALTER TABLE dept

MODIFY dname VARCHAR2(30);

Измененное значение по умолчанию применяется только при последующих вставках в таблицу.

С помощью фразы DROP COLUMN можно удалить ненужные столбцы:

ALTER TABLE dept

DROP (job\_id);

Параметр SET UNUSED позволяет пометить столбец (или столбцы) как неиспользуемые. Эти столбцы можно удалить, когда уменьшится потребность в ресурсах системы. Параметр SET UNUSED позволяет пометить один или несколько столбцов как неиспользуемые. Параметр DROP UNUSED COLUMNS позволяет удалить столбцы, помеченные как неиспользуемые.

ALTER TABLE <table\_name>

SET UNUSED(<column\_name> [ , <column\_name>]);

С помощью синтаксиса ALTER TABLE можно:

• Перевести таблицу в режим «только чтение». Этот режим не позволяет вносить изменения DDL или DML во время обслуживания таблицы.

• Перевести таблицу обратно в режим «чтение/запись».

ALTER TABLE dept READ ONLY;

ALTER TABLE dept READ WRITE;

Инструкция DROP перемещает таблицу в корзину, удаляет таблицу вместе с данными, если указана фраза PURGE и делает недействительными зависимые объекты и аннулирует полномочия объектов для таблицы.

DROP TABLE dept;

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

o Описание цели языка манипулирования данными (DDL)

Инструкция DML выполняется в следующих случаях:

– Добавление новых строк в таблицу (INSERT)

– Изменение существующих строк в таблице (UPDATE)

– Удаление существующих строк из таблицы (DELETE)

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

▪ Вставка

Добавьте строки в таблицу с помощью инструкции INSERT. Этот синтаксис позволяет вставлять строки только по одной.

INSERT INTO table [(column [, column...])]

VALUES (value [, value...]);

Прежде чем выполнять любые операции DML с таблицей, создайте копию таблицы (копируется только ограничение NOT NULL):

CREATE TABLE copy\_departments

AS (SELECT \* FROM departments);

Если вы вставляете строку, содержащую значения для каждого столбца, во фразе INSERT не нужно указывать список столбцов. Нужно перечислять значения в том же порядке, в каком по умолчанию располагаются столбцы таблицы. Для каждого столбца необходимо указать значение.

INSERT INTO copy\_departments

VALUES (40, 'Advertising', 201, 1800);

Также можно указать список столбцов во фразе INSERT. А символы и значения дат заключаются в одинарные кавычки.

Вставка строк со значениями Null:

• Явный метод: пропустить один столбец в списке столбцов.

INSERT INTO copy\_departments(department\_id,department\_name)

VALUES(30,'Purchasing');

• Неявный метод: указать ключевое слово NULL во фразе VALUES.

INSERT INTO copy\_departments

VALUES (100, 'Finance', NULL, NULL);

Для ввода специальных значений в таблице можно использовать функции. Функция SYSDATE регистрирует текущую дату и время.

INSERT INTO copy\_employees (hire\_date)

VALUES (SYSDATE);

Вставка определенных значений даты и времени

INSERT INTO copy\_employees

VALUES (TO\_DATE('Dec 7, 2002', 'MON DD, YYYY'));

▪ Обновление

Изменить существующие значения в таблице можно с помощью инструкции UPDATE:

UPDATE table

SET column = value [, column = value, ...]

[WHERE condition];

Как правило, столбец первичного ключа во фразе WHERE используется как идентификатор одной обновляемой строки.

Если указана фраза WHERE, изменяются значения в определенной строке или строках:

UPDATE copy\_employees

SET department\_id = 50

WHERE employee\_id = 113;

Если фраза WHERE пропущена, изменяются значения во всех строках таблицы:

UPDATE copy\_employees

SET department\_id = 110;

Укажите SET column\_name= NULL, чтобы изменить значение столбца на NULL.

▪ Удаление

Удалить существующие строки из таблицы можно с помощью инструкции DELETE:

DELETE [FROM] table

[WHERE condition];

Если указана фраза WHERE, удаляются определенные строки:

DELETE FROM copy\_departments

WHERE department\_name = 'Purchasing';

Если фраза WHERE пропущена, удаляются все строки таблицы:

DELETE FROM copy\_departments;

Нельзя удалить строку, содержащую первичный ключ, который используется в качестве внешнего ключа в другой таблице.

Инструкция TRUNCATE удаляет все строки из таблицы, оставляя таблицу пустой, а ее структуру без изменений. Эта инструкция относится к языку DDL, а не DML, и ее непросто отменить.

Синтаксис: TRUNCATE TABLE table\_name;

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

o Описание цели языка управления транзакциями (TCL)

Транзакции состоят из инструкций DML, представляющих собой одно согласованное изменение данных. Сервер Oracle обеспечивает согласованность данных на основе транзакций. Транзакции обеспечивают дополнительную гибкость и возможности управления при изменении данных, а также гарантируют согласованность данных в случае сбоя обработки пользователем или системного сбоя.

Транзакция начинается при выполнении первой SQL-инструкции DML. Она завершается одним из следующих событий:

– Выполнение инструкции COMMIT или ROLLBACK.

– Выполнение инструкции DDL или TCL (автоматическая фиксация).

– Пользователь выполняет выход из используемого ПО SQL.

– Сбой системы.

o Описание операций TCL для управления следующими транзакциями:

▪ COMMIT

COMMIT завершает текущую транзакцию, делая все отложенные изменения данных постоянными.

COMMIT;

Состояние данных после COMMIT:

• Изменения данных сохранены в базе данных.

• Предыдущее состояние данных перезаписано.

• Все сеансы могут просматривать результаты.

• Блокировка затронутых строк снята; доступна обработка данных строк другими сеансами.

• Все точки сохранения удалены.

▪ SAVEPOINT

SAVEPOINT помечает точку сохранения в текущей транзакции.

UPDATE…

SAVEPOINT update\_done;

▪ ROLLBACK

ROLLBACK завершает текущую транзакцию, отменяя все отложенные изменения данных.

Состояние данных после ROLLBACK:

• Отмените все отложенные изменения с помощью инструкции ROLLBACK.

• Изменения данных отменены.

• Предыдущее состояние данных восстановлено.

• Блокировка затронутых строк снята.

DELETE FROM copy\_employees;

ROLLBACK ;

ROLLBACK TO SAVEPOINT name – выполняет откат текущей транзакции до указанной точки сохранения, тем самым отменяя все изменения и/или точки сохранения, созданные после точки сохранения, с которой выполняется откат.

INSERT…

ROLLBACK TO update\_done;

Если во время выполнения происходит сбой отдельной инструкции DML, выполняется откат только этой инструкции. Сервер Oracle реализует неявную точку сохранения. Все остальные изменения сохраняются. Пользователь должен явным образом прервать транзакции, выполнив инструкцию COMMIT или ROLLBACK.

o Описание необходимости согласованности чтения

Согласованность чтения гарантирует согласованный просмотр данных в любое время — каждый пользователь видит данные в том виде, в котором они существовали на момент последнего выполнения команды COMMIT. Изменения, выполненные одним пользователем, не конфликтуют с изменениями, выполненными другим пользователем.

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

o Список возможностей инструкций SQL SELECT

SELECT определяет столбцы для отображения. FROM определяет таблицу, содержащую данные столбцы.

SELECT {\*|[DISTINCT] column|expression [alias],...}

FROM table;

o Создание и выполнение инструкции SELECT, которая:

▪ Возвращает все строки и столбцы таблицы

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

SELECT \*

FROM departments;

▪ Возвращает определенные столбцы таблицы

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

SELECT department\_id, location\_id

FROM departments;

▪ Использует псевдонимы столбцов для отображения описательных заголовков столбцов

*Псевдоним столбца:*

• переименовывает заголовок столбца;

• полезен при вычислениях;

• следует за именем столбца (между именем столбца и псевдонимом может использоваться дополнительное ключевое слово AS);

• требует использования двойных кавычек, если содержит пробелы или специальные символы или чувствителен к регистру; по умолчанию указывается в верхнем регистре.

SELECT last\_name AS name, commission\_pct comm

FROM employees;

Ключевое слово AS является необязательным.

▪ Использует арифметические операторы и операторы соединения

Создать 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.

SELECT last\_name, salary, salary + 300 FROM employees;

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

Определение значения null:

*Null* — это значение, которое недоступно, не назначено, неизвестно или неприменимо. Null не равно нулю и не является пробелом. 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

SELECT last\_name||job\_id

AS "Employees"

FROM employees;

▪ Использует 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.

SELECT last\_name ||' is a '||job\_id

AS "Employee Details"

FROM employees;

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

Альтернативный оператор кавычки (q):

SELECT department\_name || q'[ Department's Manager Id: ]' || manager\_id

AS "Department and Manager"

FROM departments;

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 [].

▪ 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.

SELECT DISTINCT department\_id

FROM employees;

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.

DESC[RIBE] tablename

DESCRIBE employees

**6-7 Ограничение данных с помощью WHERE o Ограничение строк с помощью:**

* Фразы WHERE

Ограничение возвращенных строк с помощью фразы WHERE:

SELECT \*|{[DISTINCT] column|expression [alias],...}

FROM table

[WHERE logical expression(s)];

Если логическое выражение выдает значение true, возвращается строка, отвечающая условию. Фраза WHERE следует за фразой FROM.

*Пример:*

SELECT employee\_id,last\_name,job\_id, department\_id

FROM employees

WHERE department\_id = 90;

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

Строки символов и значения дат заключены в одинарные кавычки. Значения символов чувствительны к регистру; значения даты чувствительны к формату.

SELECT last\_name, job\_id, department\_id

FROM employees

WHERE last\_name = 'Whalen' ;

Форматом отображения даты по умолчанию является DD-Mon-YYYY.

SELECT last\_name

FROM employees

WHERE hire\_date = '29-Jan-2000' ;

Операторы сравнения:

= (Равно);

> (Больше);

>= (Больше или равно);

< (Меньше);

<= (Меньше или равно);

<> (Нe равно);

BETWEEN…AND… (Между двумя значениями (включительно));

IN (set) (Соответствует любому значению в списке);

LIKE (Соответствует шаблону символа);

IS NULL (Является значением null)

Извлеките записи из таблицы EMPLOYEES, где заработная плата меньше или равна $3,000:

SELECT last\_name, salary

FROM employees

WHERE salary <= 3000 ;

Используйте оператор BETWEEN для отображения строк на основе диапазона значений:

SELECT last\_name, salary

FROM employees

WHERE salary BETWEEN 2500 AND 3500 ;

Используйте оператор IN для тестирования значений в списке:

SELECT employee\_id, last\_name, salary, manager\_id

FROM employees

WHERE manager\_id IN (100, 101, 201) ;

Используйте оператор NOT IN для тестирования значений в списке:

SELECT employee\_id, last\_name, salary, manager\_id

FROM employees

WHERE department\_id NOT IN (60, 90, 100) ;

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.

SELECT first\_name

FROM employees

WHERE first\_name LIKE 'S%' ;

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

SELECT last\_name

FROM employees

WHERE last\_name LIKE '\_o%' ;

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

SELECT last\_name, manager\_id

FROM employees

WHERE manager\_id IS NULL ;

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

AND (Возвращает значение TRUE, если оба компонентных условия имеют значение TRUE);

OR (Возвращает значение TRUE, если любое из компонентных условий имеют значение TRUE);

NOT (Возвращает значение TRUE, если условие имеет значение FALSE; Возвращает значение FALSE, если условие имеет значение TRUE)

AND требует, чтобы выполнялись оба компонентных условия:

SELECT employee\_id, last\_name, job\_id, salary

FROM employees

WHERE salary >= 10000

AND job\_id LIKE '%MAN%' ;

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

SELECT employee\_id, last\_name, job\_id, salary

FROM employees

WHERE salary >= 10000

OR job\_id LIKE '%MAN%' ;

NOT изменяет значение условия на противоположное

SELECT last\_name, job\_id

FROM employees

WHERE job\_id

NOT IN ('IT\_PROG', 'ST\_CLERK', 'SA\_REP');

o Описание правил приоритета для операторов в выражении

Правила приоритета:

1 Арифметические операторы

2 Оператор объединения

3 Условия сравнения

4 IS [NOT] NULL, LIKE, [NOT] IN

5 [NOT] BETWEEN

6 Не равно

7 Логический оператор NOT

8 Логический оператор AND

9 Логический оператор OR

Чтобы переопределить правила приоритета, необходимо использовать круглые скобки.

**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:

SELECT last\_name, job\_id, department\_id, hire\_date

FROM employees

ORDER BY hire\_date ;

o Определение правильного положения для фразы ORDER BY в инструкции SELECT

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.

*Сортировка в порядке убывания:*

SELECT last\_name, job\_id, department\_id, hire\_date

FROM employees

ORDER BY hire\_date DESC ;

*Сортировка по псевдонимам столбцов:*

SELECT employee\_id, last\_name, salary\*12 annsal

FROM employees

ORDER BY annsal;

*Сортировка с помощью номера позиции столбца:*

SELECT last\_name, job\_id, department\_id, hire\_date

FROM employees

ORDER BY 3;

*Сортировка по нескольким столбцам:*

SELECT last\_name, department\_id, salary

FROM employees

ORDER BY department\_id, salary DESC;

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. В APEX запросы Top-N выполняются с помощью ROWNUM.

SELECT ROWNUM As "Longest employed",last\_name,hire\_date

FROM (SELECT last\_name, hire\_date

FROM employees

ORDER BY hire\_date)

WHERE ROWNUM <=3;

o Применение подстановочных переменных во фразе WHERE

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. Используют подстановочные переменные для запроса значений. Нужно использовать переменную с префиксом в виде двоеточия (:) для запроса пользовательского значения:

SELECT employee\_id, last\_name, salary, department\_id

FROM employees

WHERE employee\_id = :employee\_num ;

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.

Значения символов и дат с подстановочными переменными

Можно также использовать значения даты или символов:

SELECT last\_name, department\_id, salary\*12

FROM employees

WHERE job\_id = :job\_title ;

**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

Иногда требуется использовать данные из различных таблиц.

*Типы соединений:*

– 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

Соединение для запроса данных из нескольких таблиц:

SELECT table1.column, table2.column

FROM table1

[NATURAL JOIN table2] |

[JOIN table2 USING (column\_name)] |

[JOIN table2

ON (table1.column\_name = table2.column\_name)]|

[LEFT|RIGHT|FULL OUTER JOIN table2

ON (table1.column\_name = table2.column\_name)]|

[CROSS JOIN table2];

Используйте псевдонимы таблиц для различия столбцов с одинаковыми именами, содержащихся в различных таблицах:

SELECT e.first\_name,d.department\_name,d.manager\_id,

FROM employees e JOIN departments d

USING(deparment\_id)

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.

SELECT department\_id, department\_name,location\_id, city

FROM departments NATURAL JOIN locations;

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.

SELECT employee\_id, last\_name, location\_id, department\_id

FROM employees JOIN departments

USING (department\_id) ;

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 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.

Можно также использовать фразу ON для соединения столбцов с различными именами или типами данных.

SELECT e.employee\_id, e.last\_name, e.department\_id, d.department\_id, d.location\_id

FROM employees e JOIN departments d

ON (e.department\_id = d.department\_id);

При соединении 3 таблиц необходимо наличие 2 инструкций соединения:

SELECT employee\_id, city, department\_name

FROM employees e

JOIN departments d

ON d.department\_id = e.department\_id

JOIN locations l

ON d.location\_id = l.location\_id;

В данном примере выполняется создание соединения не по эквивалентности для оценки категории оклада сотрудника. Уровень заработной платы должен находиться в пределах любой пары верхнего и нижнего значений диапазона заработной платы:

SELECT e.last\_name, e.salary, j.grade\_level

FROM employees e JOIN job\_grades j

ON e.salary

BETWEEN j.lowest\_sal AND j.highest\_sal;

o Присоединение таблицы к самой себе путем самосоединения

Фразу ON можно также использовать для соединения столбцов с различными именами, расположенных в одной или различных таблицах.

SELECT worker.last\_name emp, manager.last\_name mgr

FROM employees worker JOIN employees manager

ON (worker.manager\_id = manager.employee\_id);

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

В SQL:1999 соединение двух таблиц, возвращающих только совпадающие строки, называется соединением INNER. 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.

Здесь нам необходимо просмотреть все записи сотрудников (таблица слева), даже если они не назначены подразделению:

SELECT e.last\_name, e.department\_id, d.department\_name

FROM employees e LEFT OUTER JOIN departments d

ON (e.department\_id = d.department\_id) ;

Здесь нам необходимо просмотреть все записи подразделений (таблица справа), даже если в них отсутствуют сотрудники:

SELECT e.last\_name, d.department\_id, d.department\_name

FROM employees e RIGHT OUTER JOIN departments d

ON (e.department\_id = d.department\_id) ;

Здесь нам необходимо просмотреть все записи сотрудников и все записи подразделений:

SELECT e.last\_name, d.department\_id, d.department\_name

FROM employees e FULL OUTER JOIN departments d

ON (e.department\_id = d.department\_id) ;

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.

SELECT last\_name, department\_name

FROM employees, departments

Декартово произведение образуется, когда условие соединения опущено или недопустимо. Чтобы предотвратить образования декартова произведения, всегда указывайте допустимое условие соединения.

SELECT last\_name, department\_name

FROM employees e, departments d

WHERE e.department\_id = d.department\_id

Фраза CROSS JOIN создает векторное произведение двух таблиц. Оно также называется декартовым произведением между двумя таблицами.

SELECT last\_name, department\_name

FROM employees

CROSS JOIN departments ;