Лабораторная работа № 5 по теме: «Физическое проектирование»

Цель: развитие навыков перевода из логической модели БД в физическую, учитывая выбранную СУБД

Ход работы:

1. Оформите титульный лист
2. Выполните задания (по вариантам) и оформите отчет
3. Сдайте лабораторную работу преподавателю (на следующей паре)

Отчет по лабораторной работе должен содержать (выполняется в MS Word):

1. ФИ, группа в верхнем колонтитуле на каждой странице, кроме первой
2. Первая страница – титульный лист
3. Каждая страница пронумерована, кроме первой
4. Выполненные задания начиная со второй страницы документа (каждое задание подписано, при необходимости присутствуют скриншоты)

Теоретическая часть:

Методы логического проектирования основываются на абстрактном рассмотрении данных. Логическая модель никак не связана с конкретной реализацией модели в БД СУБД.

Физическая модель данных, напротив, зависит от конкретной СУБД, в ней содержится информация обо всех объектах базы данных. Поскольку стандартов на объекты базы данных не существует (например, нет стандарта на типы данных), физическая модель зависит от конкретной реализации СУБД и ее диалекта SQL. Следовательно, одной и той же логической модели данных могут соответствовать несколько разных физических моделей.

Основными объектами логической модели данных являются сущности, атрибуты и взаимосвязи. Физическая модель данных, как правило, создается на основе логической, поэтому каждому объекту логической модели соответствует объект физической модели (хотя соответствие может быть неоднозначным). В физической модели данных сущности логической модели данных соответствует таблица, экземпляру сущности – строка в таблице, а атрибуту – колонка таблицы. Кроме перечисленных выше объектов, физическая модель может содержать объекты, тип которых зависит от СУБД: индексы, представления, последовательности, триггеры, процедуры и т.п. Если в логической модели данных не имеет большого значения, какой конкретно тип данных у атрибута, то в физической важно описать всю информацию о конкретных объектах.

При физическом проектировании решаются следующие задачи:

- физическое распределение памяти (размерности файлов, размер страницы, распределение по дискам);
- физическая реализация объектов базы;
- определение дополнительных объектов: индекс или, например, домен в IB, генераторы…
- определение типа и формата данных;
- определение реализации хранимых данных;
- определение прав доступа пользователей;
- настройка поддержки многопользовательского доступа;
- определение методов и средств для восстановления данных (параметры копий, зеркало, их размещение)
- создание представлений

Иерархия объектов реляционной БД прописана в стандартах по SQL, в частности, в стандарте SQL-92. Этот стандарт поддерживается практически всеми современными СУБД. Иерархия объектов БД показана на рис.1.

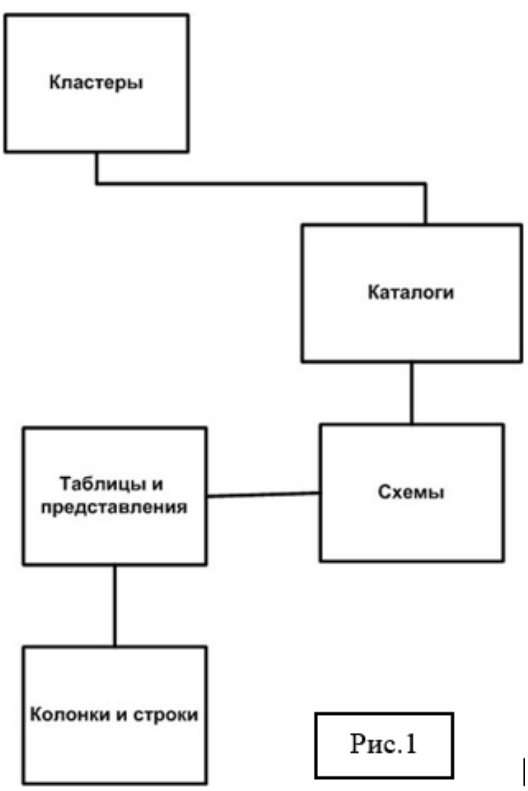


Рис.1

Основные объекты реляционной базы данных

Кластеры, каталоги > и схемы не являются обязательными элементами стандарта и, следовательно, программной среды БД.

Под **кластером** понимается группа каталогов, к которым можно обращаться через одно соединение с сервером базы данных (программная компонента СУБД).

Обычно процедура создания каталога определяется реализацией СУБД на конкретной операционной платформе. Под каталогом понимается группа схем. На практике каталог часто ассоциируется с физической базой данных как набором физических файлов операционной системы, которые идентифицируются ее именем.

Для проектировщика БД **схема** – это общее логическое представление отношений законченной базы данных. С точки зрения SQL, схема – это контейнер для таблиц, представлений и других структурных элементов реляционной базы данных. Принцип размещения элементов БД в каждой схеме полностью определяется проектировщиком. На практике схема часто ассоциируется с объектами определенного пользователя — владельца физической БД.

Схема в PostgreSQL — это логическая структура, которая представляет собой набор таблиц, индексов, представлений, функций и других объектов базы данных, организованных в единую группу. Она помогает организовать данные в базе данных, делая их более удобными для управления и обеспечивая логическую разделяемость объектов базы данных.

Схемы также играют важную роль при управлении правами доступа к данным, так как позволяют назначать различные права на доступ к объектам базы данных в зависимости от их принадлежности к определённой схеме. Существуют схемы по умолчанию, созданные PostgreSQL, одна из таких схем называется public. Именно в неё будут попадать все созданные таблицы по умолчанию, если не указать какую-то другую схему. В дополнение к схеме public и схемам, создаваемым пользователями, любая база данных содержит схему pg_catalog, в которой находятся системные таблицы и все встроенные типы данных, функции и операторы.

К числу основных объектов реляционных БД относятся таблица, представление и пользователь.

Таблица (Table) является базовой структурой реляционной БД. Она представляет собой единицу хранения данных — отношение. Таблица представляет собой двумерный массив данных, в котором колонка (столбец, атрибут) определяет значение, а строки (записи, кортежи) содержат данные. Таблица идентифицируется в БД своим уникальным именем, которое включает в себя идентификацию пользователя. Таблица может быть пустой или состоять из набора строк.

Представление (View) — это поименованная динамически поддерживаемая СУБД выборка из одной или нескольких таблиц базы данных. Оператор выборки ограничивает видимые пользователем данные. Обычно СУБД гарантирует актуальность представления: его формирование производится каждый раз, когда представление используется. Иногда представления называют виртуальными таблицами.

Пользователь (User) — это объект, обладающий возможностью создавать или использовать другие объекты базы данных и запрашивать выполнение функций СУБД, таких как организация сеанса работы, изменение состояние базы данных и т.д.

Определенные пользователем **типы данных** (User-defined data types) представляют собой определенные пользователем типы атрибутов (домены), которые отличаются от поддерживаемых (встроенных) СУБД типов. Они определяются на основе встроенных типов.

Правила (Rules) – это декларативные выражения, ограничивающие возможные значения данных. Для формулировки правила используются допустимые предикатные выражения SQL.

Для обеспечения эффективного доступа к данным в реляционных СУБД поддерживается ряд других объектов: индекс, табличная область, кластер, секция.

Индекс (Index) — это объект базы данных, создаваемый для повышения производительности выборки данных и контроля уникальности первичного ключа (если он задан для таблицы).

Секция (Partition) — это объект базы данных, который позволяет представить объект с данными в виде совокупности подобъектов, отнесенных к различным табличным пространствам. Таким образом, секционирование позволяет распределять очень большие таблицы на нескольких жестких дисках.

Для обработки данных специальным образом или для реализации поддержки ссылочной целостности базы данных используются объекты: хранимая процедура, функция, команда, триггер. С помощью этих объектов базы данных можно выполнять так называемую построчную обработку (record processing) данных. С точки зрения приложений баз данных построчная обработка — это последовательная выборка данных по одной строке, ее обработка и переход к обработке следующей строки.

Данные объекты реляционной базы данных представляют собой программы, т.е. исполняемый код. Этого код обычно называют серверным кодом (server-side code), поскольку он выполняется компьютером, на котором установлено ядро реляционной СУБД. Планирование и разработка такого кода является одной из задач проектировщика реляционной базы данных.

Хранимая процедура (Stored procedure) — это объект базы данных, представляющий поименованный набор команд SQL и/или операторов специализированных языков обработки программирования базы данных.

Функция (Function) — это объект базы данных, представляющий поименованный набор команд SQL и/или операторов специализированных языков обработки программирования базы данных, который при выполнении возвращает значение — результат вычислений.

Триггер (Trigger) — это объект базы данных, который представляет собой специальную хранимую процедуру. Эта процедура запускается автоматически, когда происходит связанное с триггером событие (например, вставка строки в таблицу).

Для эффективного управления разграничением доступа к данным поддерживается объект роль.

Роль (Role) – объект базы данных, представляющий собой поименованную совокупность привилегий, которые могут назначаться пользователям, категориям пользователей или другим ролям.

Домены в физической модели данных

В логической модели данных среда реализации не учитывается. В ней определяются атрибуты и их возможные значения, такие как строка, число или дата, в идеале атрибуту может назначаться домен. Домен — это просто тип атрибута, например, "Деньги" или "Рабочий день". Проектировщик может включить ряд проверок допустимости или правил обработки, например, требование, что значение должно быть положительным, ненулевым и иметь максимум два десятичных разряда.

Использование доменов упрощает задачу обеспечения непротиворечивости на стадии логической модели данных. При переходе к проектированию физической модели данных проектировщику необходимо знать возможности выбранной СУБД по назначению типов данных колонок. В логической модели данных значения, которые может принимать атрибут отношения, также задаются доменом, который наследуется из информационной модели. В физической модели базы данных требуется, чтобы каждый атрибут отношения в базе данных обладал рядом свойств, которые диктуют, что в нем может храниться и что не может. Этими свойствами являются **тип, размер и ограничения**, которые могут еще более ограничивать допустимый набор значений столбца. Задача состоит в преобразовании домена в подходящий тип данных, поддерживаемый СУБД. Таким образом, проектировщик должен знать, какими типами данных он располагает при решении вышеуказанной задачи.

В контексте проектирования физической модели реляционной БД **домен** – это выражение, определяющее разрешенные значения для колонок (атрибутов) отношения. При описании таблицы реляционной БД каждой колонке назначается определенный тип данных. Практически, основу определения домена составляет тип данных, содержащихся в колонке, поскольку большинство встроенных типов задают разрешенный интервал значений данных.

Пример типа, размера и ограничения

```
amount NUMBER (8,2) NOT NULL CONSTRAINT cc_limit_amnt CHECK
(amount > 0)
```

В колонке "Сумма платежа" (`Amount`) можно размещать только числовые данные; точность этого значения — два значащих десятичных разряда (`NUMBER (8,2)`); она должна быть заполнена для каждой строки *таблицы* (`NOT NULL`); ее значение должно быть положительным `CONSTRAINT cc_limit_amnt CHECK (amount > 0)` . Максимальное значение, которое может храниться в этом столбце, — 999999.99. В этом простом определении колонки мы фактически определили ряд неявных правил, проверку которых MS SQL Server принудительно включает при вводе данных в БД.

Как видно, дальнейшее определение домена колонки (после присвоения ей типа) выполняется проектировщиком с помощью уточнений правил изменения значений. Такие уточнения поддерживаются в SQL с помощью механизма *ограничений* в *спецификации колонки* в *таблице*.

Ограничения (constraint) — это правила, которые накладываются на данные в таблицах. Они определяют условия, которым должны соответствовать данные при вставке, обновлении или удалении полей в таблице.

Есть два вида ограничений: ограничения целостности данных и ограничения целостности ссылок.

Ограничения ссылочной целостности обеспечивают целостность ссылок между связанными таблицами. К ним относится FOREIGN KEY constraint, или ограничение внешнего ключа. Оно гарантирует, что значения в столбцах одной таблицы ссылаются на существующие значения в другой таблице.

Ограничения целостности (integrity constraints) определяют отношения между данными в разных таблицах и обеспечивают согласованность данных в соответствии с установленными правилами.

К целостным ограничениям относятся:

- **Ограничение NOT NULL** гарантирует, что столбец обязательно будет иметь значение для каждой записи, то есть значение будет не нулевым. Таким образом программа не позволит хранить в столбцах пустые значения (Ограничение NOT NULL может быть полезно для столбцов с контактными данными, когда нам нужно обязать пользователя, например, ввести свою электронную почту или номер телефона. Поэтому такие обязательные поля нередко используют ограничение NOT NULL, чтобы гарантировать, что пользователь введет определенное значение)
- **Ограничение UNIQUE.** Unique значит «уникальный», и это название полностью отражает суть ограничения. Таким образом, ограничение UNIQUE гарантирует, что никакие два значения в определяемом столбце не будут одинаковыми (Ограничение UNIQUE идеально подходит для столбцов, которые не должны содержать повторяющихся значений. Например, у каждого из нас уникальные номера паспорта и полиса социального страхования (СНИЛС). Таким образом, если таблица содержит столбцы, в которых хранятся номера паспорта и СНИЛС, эти столбцы должны использовать ограничение UNIQUE. Это необходимо, чтобы избежать того, что у двух человек будут одни и те же номера, которые могут быть вставлены по ошибке или намеренно)
- **Ограничение CHECK.** Check в переводе с английского значит «проверять», и ограничение CHECK служит для проверки значений по определенному условию. Инструкцию CHECK можно использовать для реализации пользовательских ограничений. Так, если в таблице должны храниться только данные взрослых, мы могли бы использовать ограничение CHECK для столбца «Возраст покупателя. Рассмотрим следующий пример, в нем введённый покупатель не может быть моложе 18 лет:

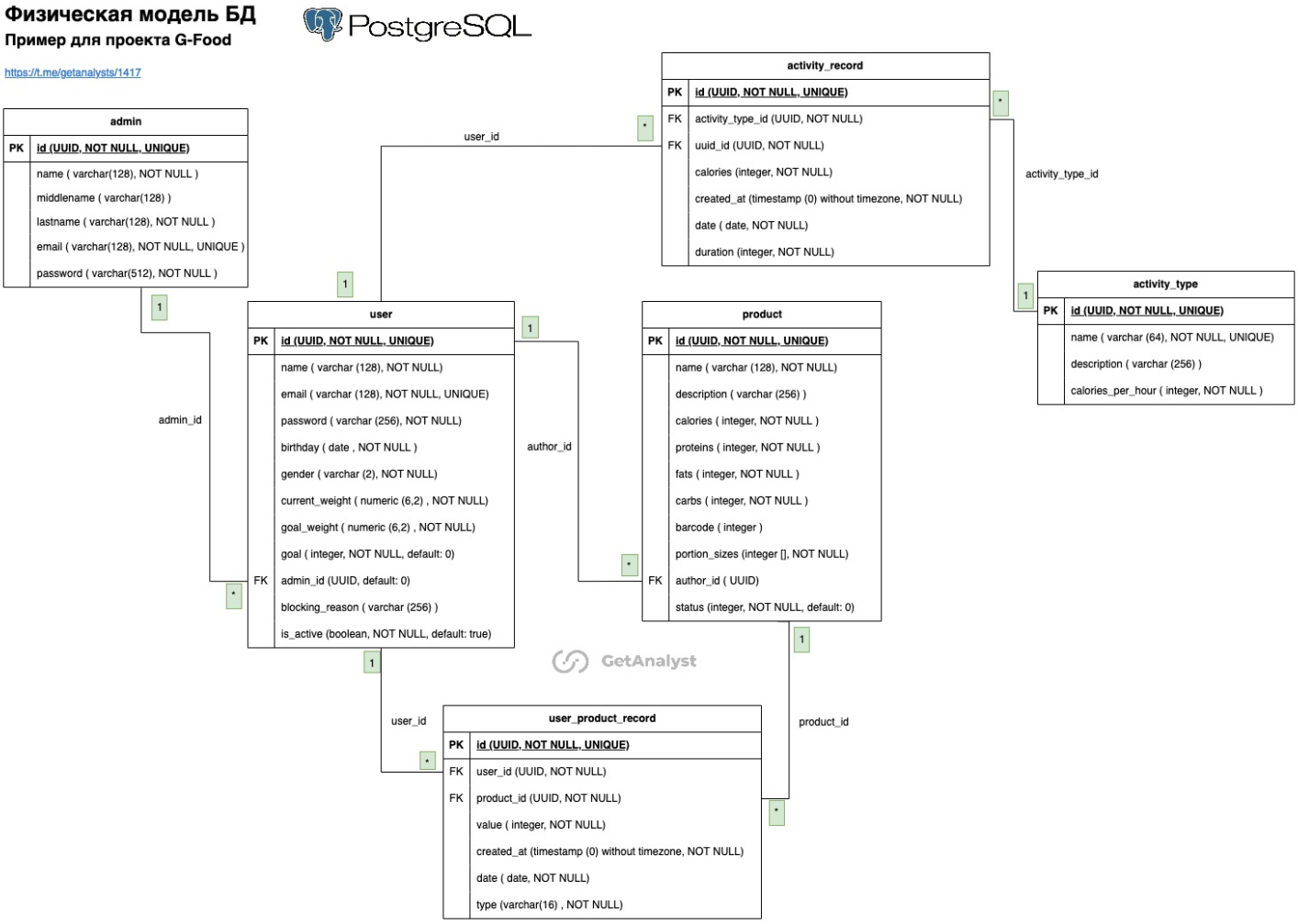
```
CREATE TABLE Customers1 (
CustomerName1 VARCHAR(46),
CustomerName2 VARCHAR(46),
CustomerEmail VARCHAR(56),
CustomerAge INTEGER CHECK (CustomerAge>17)
)
```

- **Ограничение PRIMARY KEY** — это одно из ограничений ключа таблицы SQL, в данном случае — первичного ключа. PRIMARY KEY используется для создания идентификатора, с которым соотносится каждая строка в таблице. PRIMARY KEY в таблице может относиться только к одному столбцу (и это понятно, так как это идентификатор). Соответственно, каждое значение PRIMARY KEY обязательно должно быть уникальным, при этом нулевые значения в столбце, определенном с помощью PRIMARY KEY, не допускаются.

- **Ограничение FOREIGN KEY** (внешний ключ) создает ссылку на PRIMARY KEY из другой таблицы. Таким образом, столбец, в котором есть FOREIGN KEY, ссылается на столбец с PRIMARY KEY из другой таблицы, и текущая таблица связывается с ней через это ограничение
- **DEFAULT**. Устанавливает предопределённое значение в случае, если другое не было указано

Этапы перевода логической модели БД в физическую модель:

1. Определиться с используемой СУБД
2. Изучить типы данных, используемые в СУБД (Для PostgreSQL тут: https://postgrespro.ru/docs/postgresql/9.6/datatype)
3. Если отношения и атрибуты названы по-русски, то переименовать на английский или написать транслитом
4. Для каждого поля в скобках указать тип данных и ограничения (если есть)
5. Указать первичные и внешние ключи



Ход работы.

ЗАДАНИЕ.

Для своего варианта из 3 и 4 лабораторных работ создать физическую модель данных (в любом case-средстве). В отчете указать выбранную СУБД. Объяснить выбранные ограничения и типы данных.

Критерии оценки:

Зачтено	Все задания лабораторной работы самостоятельно, студент может объяснить, как он выполнил то или иное действие; отчет содержит все необходимые сведения, студент уверенно отвечает на вопросы по теории
Не зачтено	Лабораторная работа выполнена не самостоятельно (студент не может ответить ни на один вопрос преподавателя), или выполнена не до конца, или не выполнена совсем