**Лекция 7. Структуры внешней памяти, методы организации индексов**

**Объекты реляционной базы данных**

**Иерархия объектов реляционной базы данных**

Одной из главных задач, которые обязан решить проектировщик на стадии ***проектирования******физической модели реляционной базы данных***, является задача превращения объектов *логической модели реляционной базы данных* в объекты реляционной базы данных.

Для решения этой задачи проектировщику базы данных необходимо знать:

а) какими объектами располагает реляционная база данных в принципе;

б) какие объекты поддерживает конкретная СУБД, которая выбрана для реализации базы данных.

Таким образом, мы предполагаем, что решение о выборе СУБД уже принято руководителем ИТ-проекта, и согласовано с заказчиком базы данных, т.е. СУБД задана. Проектировщик базы данных должен ознакомиться с документацией, в которой описан диалект SQL, поддерживаемый выбранной СУБД. В настоящей лекции предполагается, что была выбрана СУБД Oracle 9i, хотя подавляющая часть материала охватывает объекты в любой промышленной реляционной СУБД.

*Замечание. О выборе СУБД. Выбор СУБД относится к многокритериальной задаче выбора и в настоящем курсе не рассматривается. Следует помнить о том, что СУБД обычно поддерживает только одну модель данных: реляционную, иерархическую, сетевую, многомерную, объектно-ориентированную, объектно-реляционную. Исключение составляют небольшое число СУБД. Например, ADABAS, Software AG (сетевая и реляционная модели), или Oracle 9i, Oracle Inc. (реляционная и объектно-реляционная модели). Обычно при выборе СУБД при всех прочих равных возможностях стараются создать базу данных на СУБД, претендующей на промышленный стандарт.*

Иерархия объектов реляционной базы данных показана на рисунке ниже. **На самом нижнем уровне находятся наименьшие объекты, с которыми работает реляционная база данных, - столбцы (колонки) и строки. Они, в свою очередь, группируются в таблицы и представления.**

Таблицы и представления, которые представляют физическое отражение логической структуры базы данных, собираются в схему. Несколько схем собираются в каталоги, которые затем могут быть сгруппированы в кластеры.



Помимо указанных на рисунке объектов, в реляционной базе данных могут быть созданы *индексы, триггеры, события, хранимые команды, хранимые процедуры* и ряд других. Теперь перейдем к определению объектов реляционной базы данных.

**Основные объекты реляционной базы данных**

Кластеры, каталоги и схемы не являются обязательными элементами стандарта и, следовательно, программной среды реляционных баз данных.

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

На практике 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 базы данных 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 базы данных в каждой схеме полностью определяется проектировщиком базы данных.

Для создания таблиц и представлений наличие схемы не обязательно. Если у вас планируется инсталляция только одной логической базы данных, то ясно, что можно обойтись и без схемы. Но если планируется, что одна и та же СУБД будет использоваться для поддержки нескольких баз данных, то надлежащая организация объектов баз данных в схемы может значительно облегчить сопровождение этих баз данных. На практике схема часто ассоциируется с объектами определенного пользователя физической базы данных.

**Таблица** (Table) является базовой структурой реляционной базы данных. Она представляет собой единицу хранения данных - отношение. Таблица идентифицируется в базе данных 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.

Для упрощения идентификации и именования объектов в базе данных поддерживается такие объекты, как синоним, последовательность и *определенные пользователем типы данных*.

**Синоним** (*Synonym*) - это *альтернативное имя* объекта (псевдоним) реляционной базы данных, которое позволяет иметь доступ к данному объекту. Синоним может быть общим и частным. Общий синоним позволяет всем пользователям базы данных обращаться к соответствующему объекту по его псевдониму. Синоним позволяет скрыть от конечных пользователей полную квалификацию объекта в базе данных.

**Последовательност**ь (Sequence) - это объект базы данных, который позволяет генерировать последовательность уникальных чисел (номеров) в условиях многопользовательского асинхронного доступа. Обычно элементы последовательности используются для уникальной нумерации элементов таблиц (строк) в операциях модификации данных.

*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. *Определенные пользователем типы данных* образуют ту часть среды СУБД, которая организована в соответствии с объектно-ориентированной парадигмой.Для обеспечения эффективного доступа к данным в реляционных СУБД поддерживаются 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). Полностью индексные таблицы (index-organized tables) исполняют роль таблицы и индекса одновременно.

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

**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.

**Команда**(Command) - это поименованный оператор SQL, который заранее откомпилирован и сохраняется в базе данных. Скорость обработки команды выше, чем у соответствующего ему оператора SQL, т.к. при этом не выполняются фазы *синтаксического разбора* и компиляции.

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

**Таймер** (Timer) отличается от триггера тем, что запускающим событием для хранимой процедуры является событие таймера.

**Пакет** (Package) - это объект базы данных, который состоит из поименованного структурированного набора переменных, процедур и функций.

**Правила определения имен**

Как и в любом языке, имена используются для идентификации элементов и объектов языка. В этом отношении имя есть *идентификатор объекта* SQL. Имена бывают длинными (до 18 символов) и короткими (до 8 символов). Также различают обыкновенный идентификатор, который начинается с буквы или символов #, @, $ и состоит из букв, цифр и символа \_, и идентификатор в апострофах (*Delimited* Identifier), который состоит из произвольных символов, заключенных в двойные кавычки.

Объекты SQL именуются в соответствии со своей иерархией, и могут иметь квалифицируемые имена, когда имя объекта квалифицируется именем охватывающего объекта, присоединенного к имени вложенного объекта через точку. По стандарту SQL охватывающим является имя схемы, которое есть практически во всех реализациях реляционных СУБД, в том числе и в Oracle. Нижеследующие объекты SQL должны иметь уникальное имя.

* Имя пользователя (Authorization ID), для идентификации которого используется короткий идентификатор, обозначающий *пользователя базы данных*.
* Колонки таблицы или представления базы данных, для идентификации которых используется, возможно, квалифицируемый длинный идентификатор. Имя колонки квалифицируется посредством либо имени таблицы, либо имени представления, либо алиасным (корреляционным) именем таблицы, назначенным в команде SQL.
* База данных, для идентификации которой используется короткий идентификатор, обозначающий базу данных. Имя базы данных может начинаться только с буквы и состоять из букв и цифр.
* Индексы таблиц, для идентификации которых используются, возможно, квалифицируемый длинный идентификатор. Имя индекса квалифицируется именем пользователя, который выдает команду, использующую данный индекс.
* Пароль *авторизации доступа*, для идентификации которого используется короткий идентификатор.
* Внутренние (связанные с командой SQL) переменные (Bind Variable), для идентификации которых используются обыкновенные идентификаторы или цифры с предшествующим им двоеточием.
* Команды SQL, для идентификации которых используются длинные идентификаторы. Имя команды определяется пользователем.
* Синонимы таблиц и представлений, для идентификации которых используются длинные идентификаторы. Синонимы сохраняются в системном каталоге и используются в качестве альтернативных имен таблиц и представлений.
* Таблицы базы данных, для идентификации которых используются, возможно, квалифицируемые длинные идентификаторы. В качестве квалификаторов применяются имена пользователей.
* Представления (*виртуальные таблицы*) базы данных, для идентификации которых используются, возможно, квалифицируемые длинные идентификаторы. В качестве квалификаторов применяются имена пользователей.
* События таймера, для идентификации которых используются, возможно, квалифицируемые длинные идентификаторы. В качестве квалификаторов применяются имена пользователей.
* Хранимые процедуры, для идентификации которых используются, возможно, квалифицируемые длинные идентификаторы. В качестве квалификаторов применяются имена пользователей.
* Триггеры, для идентификации которых используются, возможно, квалифицируемые длинные идентификаторы. В качестве квалификаторов применяются имена пользователей.

Таким образом, способ именования и идентификации объектов реляционной базы данных задается отчасти их иерархией и подчиняется следующим общим правилам:

1 имена столбцов должны быть уникальны в таблице;

2 имена таблиц должны быть уникальны в схеме;

3 имена схем должны быть уникальны в каталоге (базе данных);

4 для доступа к объекту базы данных используется квалификация имени;

5 для *идентификаторов объектов* используются буквы, цифры и символы подчеркивания.

**Домены и допустимые типы данных**

В информационной модели, создаваемой на этапе анализа, среда реализации не учитывается. 

*Аналитик* просто определяет атрибуты как строку, число или дату, в идеале он также назначает атрибуту *домен*. В контексте аналитика *домен* - это просто тип атрибута, например деньги или рабочий день. *Аналитик* может включить ряд проверок допустимости или правил обработки, например требование, что *значение* должно быть положительным, ненулевым и иметь *максимум* два десятичных разряда (это полезно для сумм долларовых трат, выставляемых банком на другой банк). Использование доменов аналитиком упрощает задачу обеспечения непротиворечивости. При переходе к проектированию физической модели 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 *базы данных* должен знать, какими типами данных он располагает при решении вышеуказанной задачи.

В контексте проектирования *физической модели реляционной базы данных* *домен* - это *выражение*, определяющее разрешенные значения для колонок (атрибутов) отношения. При описании таблицы реляционной *базы данных* 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.

**Допустимые типы данных**

Все допустимые типы данных описаны в *стандарте SQL-92*, но в большинстве диалектов поддерживается расширенный список типов данных. Однако любой диалект SQL поддерживают три общих типа данных: строковые, числовые и тип для представления даты и времени. Задание типа данных определяет значения и длину данных, а также формат их представления при визуализации.

Для всех типов данных определено так называемое **нуль-значение**, которое указывает на отсутствие данных в колонке указанного типа, т.е. то обстоятельство, что значение данных в текущий момент времени неизвестно.

Данные **строкового типа** представляют собой последовательность строк символов. Строковые данные могут быть заданы как с предопределенной длиной (ключевые слова char или varchar (длина строки)), так и без указания длины (ключевое слово long varchar ) для представления строк произвольной длины.

Тип данных varchar2 определяет строку символов переменной длины, имеющую максимальный размер size. В отличие от строкового типа с предопределенной длиной, со строками long varchar не допускаются операции сравнения, и они не могут быть использованы в выражениях и как аргументы большинства встроенных функций. Строки последнего типа могут применяться для сохранения битовых образов. 

Существует **два типа числовых данных**.

1. Целые и вещественные значения (например, сальдо банковского счета или *ставка процента*). Они являются объектом математической обработки.
2. Строковые числовые данные, в которых единственно допустимыми символами являются цифры (например, номера банковских счетов).

Числовые типы данных предназначены для представления целых чисел, чисел с десятичной точкой и чисел с плавающей точкой. Любое представление чисел задается своей точностью и масштабом. Точность определяет допустимое представлением количество значащих цифр числа, а масштаб - количество значащих цифр после десятичной точки.

Для представления целых чисел используются типы interger (точность 10 значащих цифр) и *smallint* (точность 5 значащих цифр).

Для представления **чисел с фиксированной десятичной точкой** используются типы number (точность, масштаб) (для чисел с точностью до 15 значащих цифр) и decimal (точность, масштаб) (для чисел заданной точности до 15 значащих цифр). Если указать для колонки тип number без задания масштаба, максимальное число значащих цифр для Oracle будет 105. Вместо задания точности и масштаба может быть указан символ \*. Это будет эквивалентно заданию просто типа number. Различие между этими типами данных состоит в том, что для типа number нет необходимости следить за точностью при выполнении операций.

Для представления **чисел с плавающей точкой** в SQL предусмотрены следующие типы данных:

* *Double Precision* - для чисел с точностью от 22 до 53 значащих цифр;
* Float (точность) - для представления чисел с точностью от 1 до 21 значащей цифры;
* Real - для чисел с точностью по умолчанию (зависит от конкретной реализации).

**Тип данных** **для представления даты и времени** отсутствует в стандарте SQL. Обычно в конкретных диалектах SQL используются три типа для представления таких данных:

* datestamp (*timestamp*) - для представления даты и времени;
* date - для представления даты;
* time - для представления времени.

**Константы, выражения, системные переменные**

Константы обычно специфицируют единственное значение и, в соответствии с типом представляемых данных, могут быть строковыми, числовыми и представлять дату/время. Строковые константы должны быть заключены в одинарные кавычки.

В SQL существует ряд предопределенных системных переменных, которые можно использовать в выражениях вместо имен колонок и констант. К таким переменным относятся следующие:

* NULL - для представления неопределенных значений;
* ROWID - (в SQLBase) внутренний системный номер строки в таблице;
* USER - имя пользователя, активного в данный момент;
* SYSDATETIME - системное текущее время и дата;
* SYSDATE - системная текущая дата;
* SYSTIME - системное текущее время;
* SYSTIMEZONE - временной пояс, установленный в системе.

**Выражением в SQL** является итем или комбинация итемов с допустимыми для них операциями, которая дает единственное значение. В качестве итемов могут выступать имена колонок, константы, *связанные переменные*, результаты вычислений функций, системные переменные и другие выражения. При этом если один из итемов имеет нуль-значение, то результат выражения также имеет нуль-значение.

**КОНТРОЛЬНЫЕ ВОПРОСЫ К ЛЕКЦИИ**

1. Назовите главную задачу проектировщика на стадии проектирования физической модели реляционной базы данных
2. Перечислите объекты реляционной БД
3. Назовите объекты, находящиеся на самом нижнем уровне иерархии
4. Дайте определение понятию «таблица»
5. Дайте определение понятию «кластер»
6. Дайте определение понятию «представление»
7. Дайте определение понятию «пользователь»
8. Дайте определение понятию «синоним»
9. Дайте определение понятию «последовательность»
10. Дайте определение понятию «индекс»
11. Дайте определение понятию «хранимая процедура»
12. Дайте определение понятию «функция»
13. Дайте определение понятию «команда»
14. Дайте определение понятию «триггер»
15. Дайте определение понятию «таймер»
16. Дайте определение понятию «пакет»
17. Перечислите общие правила именования и идентификации объектов реляционной базы данных
18. Дайте определение понятию «домен» в рамках контекста аналитика, логической и физической модели данных
19. Для чего предназначены данные строкового типа? Какие типы данных для строк вы знаете в рамках языка SQL?
20. Для чего предназначены числовые типы данных? Приведите примеры типов данных для чисел с фиксированной и плавающей точкой.
21. Приведите пример двух-трех предопределенных системных переменных.