Федеральное агентство по образованию
Российской Федерации

Государственное образовательное учреждение высшего
профессионального образования
Ульяновский государственный технический университет

А. М. Наместников

ПОСТРОЕНИЕ БАЗ ДАННЫХ В СРЕДЕ ORACLE
Практический курс

Учебное пособие по курсу
«Базы данных»

Ульяновск 2008

УДК 519.852 + 517.977.5 (075)

ББК 73я7

Н30

Рецензенты:

Утверждено редакционно-издательским советом университета
в качестве учебного пособия для вузов

Наместников А.М.

Н30 Построение баз данных в среде Oracle. Практический курс: Учеб. пособие для вузов. — Ульяновск: УлГТУ, 2008. — 146с.

ISBN 5-89146-000-0

Содержит основные сведения, необходимые для построения баз данных в среде Oracle 10g с практическими примерами.

Для студентов вузов, обучающихся по специальностям «Прикладная информатика(в экономике)» и другим, применяющим ЭВМ в задачах построения баз данных.

УДК 519.852 + 517.977.5 (075)
ББК 73я7

Наверное нет более распространенного приложения в мире информационных систем, чем базы данных. Они применяются как в составе сложных интегрированных информационных систем, так и в виде простых однопользовательских приложений. Инструментальных сред для реализации современных баз данных известно большое количество и одной из них является Oracle 10g.

В данном учебном пособии содержится материал, используя который можно приступить к реализации собственной базы данных в среде Oracle 10g Express Edition, которая является свободно распространяемым программным пакетом. Конечно, материал, содержащийся в учебном пособии, не раскрывает все тонкости практической работы с Oracle 10g. В частности, не рассмотрены вопросы администрирования, безопасности и применения стандартов доступа к базам данных. Для получение дополнительной информации следует обращаться к литературе из предложенного списка.

Ульяновск, *А.М. Наместников*
Январь 2008

Оглавление

Глава 1

Описание практического примера

Перед непосредственным рассмотрением языка SQL и системы управления базами данных Oracle 10g необходимо определиться с теми исходными данными, которые будем применять в примерах учебного пособия.

В качестве примера рассматривается база данных небольшой художественной галереи [1]. Перечень требований к приложению для галереи:

- Вести учет покупателей и их художественных интересов.
- Отслеживать приобретения, которые делает галерея.
- Отслеживать покупки клиентов.
- Вести список художников и произведений, когда-либо появлявшихся в галерее.
- Генерировать отчет о том, насколько быстро и с какой прибылью продаются произведения конкретного художника.
- Отображать на веб-странице список произведений, выставленных на продажу.

Когда галерея покупает произведение, сведения о нем, его авторе, дате и стоимости приобретения записываются в базу данных. В отдельных случаях галерея может выкупить произведение у клиента и вновь выставить его на продажу, так что одно и то же произведение может появляться в галерее неоднократно. При повторном приобретении информация о работе и ее авторе не вводится заново: записывается только дата и стоимость последнего приобретения. Когда работа продается, записываются дата совершения сделки, уплаченная сумма и сведения о покупателе.

Данные о предыдущих продажах необходимы продавцам для того, чтобы они могли уделять больше времени наиболее активным покупателям. Иногда эти записи используются для определения местонахождения ранее проданных произведений.

Для маркетинговых целей требуется, чтобы приложение базы данных выдавало список всех произведений, которые когда-либо появлялись в галерее, и их авторов. Владелец хотел бы также иметь возможность определять, насколько быстро продаются произведения каждого из художников и какова прибыль от их продажи. Наконец, приложение должно отображать список работ, имеющихся в наличии.

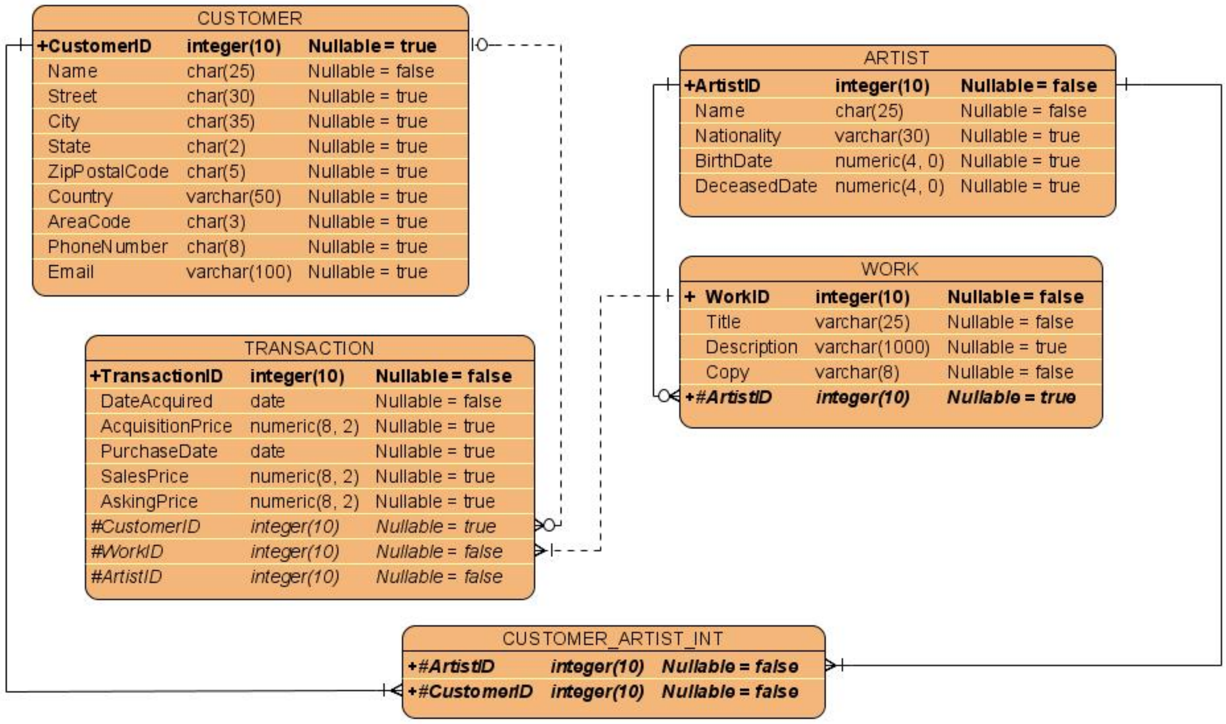

Рис. 1.1. Модель данных для практического примера

Модель данных такого примера приведена на рис. 1.1. В ней есть две сильных сущностей — CUSTOMER (клиент) и ARTIST (художник). Кроме того, имеется сущность WORK (произведение), идентификационно-зависимая от сущности ARTIST, и сущность TRANSACTION (транзакция), идентификационно-зависимая от сущности WORK. Между сущностями CUSTOMER и WORK имеется неидентифицирующая связь принадлежности.

Сведения о художнике могут присутствовать в базе данных, даже если ни одна из его работ не появлялась в галерее. Это сделано для того, чтобы можно было регистрировать интерес клиентов к художникам, чьи работы галерея может приобрести в будущем. Таким образом, с художником может быть связано любое количество произведений, в том числе ноль.

Идентификатором сущности WORK является группа (Title, Copy) (название, номер копии), поскольку в случае литографий и фотографий произведение может существовать в нескольких экземплярах. Кроме того, в требованиях к приложению указано, что одно и тоже произведение может неоднократно появляться в галерее, поэтому с каждым произведением потенциально может быть связано много транзакций. Каждый раз, когда произведение появляется в галерее, необходимо записывать дату и стоимость приобретения. Таким образом, каждой работе должна соответствовать по меньшей мере одна транзакция.

Клиент может приобрести множество работ; этот факт обозначен связью вида 1:N между сущностями CUSTOMER и TRANSACTION. Кроме того, между сущностями CUSTOMER и ARTIST существует связь вида N:M.

Удаление строк в таблицах CUSTOMER и ARTIST вызывает каскадное удаление в таблице CUSTOMER_ARTIST_INT. Это имеет смысл, поскольку когда сведения о клиенте или художнике удаляются из базы данных, нет нужды сохранять информацию о предпочтениях данного клиента или интересе к данному художнику. Если с клиентом связана хотя бы одна транзакция, этот клиент не может быть удален из базы данных. Аналогично, если с художником связана хотя бы одна картина, удалить его будет нельзя. Кроме того, записи о работах, по которым имели место какие-либо транзакции, удалению также не подлежат.

Данные для рассматриваемого примера приведены в табл. 1.1 -1.3.

Таблица 1.1. Данные для таблицы ARTIST

ArtistID	**Name**	**Nationality**	**BirthDate**	**DeceasedDate**
3	Miro	Spanish	1870	1950
4	Kandinsky	Russian	1854	1900
5	Frings	US	1700	1800
6	Klee	German	1900	<NULL>
8	Moos	US	<NULL>	<NULL>
14	Tobey	US	<NULL>	<NULL>
15	Matisse	French	<NULL>	<NULL>
16	Chagall	French	<NULL>	<NULL>

Таблица 1.2. Данные для таблицы WORK

WorkID	**Title**	**Description**	**Copy**	**ArtistID**
505	Mystic Fabric	One of the only pr	99/135	14
506	Mi Vida	Very black, but ve	7/100	3
507	Slow Embers	From the artist's	HC	14
525	Mystic Farbic	Some water damage	105/135	14
530	Northwest by Night	Wonderful, moody	37/50	16

Таблица 1.3. Данные для таблицы CUSTOMER_ARTIST_INT

ArtistID	**CustomerID**	**ArtistID**	**CustomerID**
3	1036	14	1015
5	1015	14	1033
5	1034	14	1034
5	1041	14	1036
5	1051	14	1040
8	1034	14	1041
8	1041	14	1051
14	1001	16	1015

Глава 2

Введение в язык SQL

Язык SQL был разработан фирмой IBM в конце 1970-х годов и был принят Американским национальным институтом стандартов (ANSI) в качестве национального стандарта США в 1992 году [1].

Язык SQL ориентирован на текст. Он был разработан задолго до появления графических интерфейсов пользователя, так что для работы с ним требуется лишь текстовый редактор. Разумеется, в таких СУБД как Oracle имеются графические средства для выполнения многих из тех задач, которые ранее могли быть выполнены только с помощью SQL. Но не все из того, что позволяет делать SQL, можно осуществить с помощью графических средств; более того, в ряде случаев, например для динамической генерации операторов SQL в программном коде, SQL использовать необходимо.

С помощью SQL можно определять структуры базы данных, а также запрашивать и обновлять информацию в базе данных. Совокупность команд, служащих для определения данных, называют иногда *языком определения данных* (data definition language, DDL), а совокупность команд для обновления и запроса данных — *языком манипулирования данными* (data manipulation language, DML). Далее в учебном пособии будут рассмотрены оба эти подмножества языка SQL на примере, приведенном в предыдущей главе.

2.1 Средства определения данных языка SQL

Оператор CREATE TABLE

Основная функция этого оператора — создание новой таблицы и описание ее столбцов и типов данных. Кроме того, этот оператор позволяет определять первичные ключи, альтернативные ключи и внешние ключи с некоторыми ограничениями ссылочной целостности, а также задавать ограничения на столбцы и таблицы.

Листинг 2.1.

```
CREATE TABLE CUSTOMER(
    CustomerID      int             PRIMARY KEY,
    Name            char(25)        NOT NULL,
    Street          char(30)        NULL,
    City            char(35)        NULL,
    State           char(2)         NULL,
    ZipPostalCode   char(5)         NULL,
    Country         varchar(50)     NULL,
    AreaCode        char(3)         NULL,
    PhoneNumber     char(8)         NULL,
    Email           varchar(100)    NULL);
```

В SQL имеется пять типов ограничений: PRIMARY KEY, NULL/NOT NULL, UNIQUE, FOREIGN KEY и CHECK. В листинге 2.1 столбец CustomerID принадлежит к типу данных Integer (целочисленный) и имеет свойство Primary Key. Следующий столбец, Name, имеет тип данных Character (строковый) с максимальной длиной 25 символов. Ключевые слова Not Null означают, что этот столбец обязан иметь значение.

В SQL первичные ключи ни при каких условиях не могут иметь пустых значений. Именно поэтому для столбца CustomerID можно просто указать свойство Primary Key, не уточняя, что он не должен быть пустым (Not Null). Ключевые слова Primary Key сами по себе уже говорят, что столбец CustomerID не будет иметь пустых занчений. Для уникальных столбцов, однако, пустые значения возможны.

Седьмой столбец, Country, принадлежит к типу данных VarChar(50) и имеет свойство Null. VarChar обозначает строку переменной длины. Таким образом, в разных строках значения столбца Country могут различаться по длине, и максимально возможная длина строки равна 50 символам. Ключевое слово Null указывает на то, что пустые значения допустимы.

Значения типа Char имеют фиксированную длину. Если столбец Name определен как Char(25), это означает, что каждое значение столбца Name будет храниться в виде строки длиной 25 символов, независимо от того, какова реальная длина имени клиента. При необходимости имена будут дополняться пробелами до 25 символов. Значения типа VarChar могут иметь разную длину. Если название страны состоит всего из четырех символов, то только эти четыре символа и будут храниться в столбце Country.

В табл. 2.1 перечислены некоторые из типов данных, поддерживаемых в СУБД Oracle

Таблица 2.1. Типы данных SQL в СУБД Oracle

Тип данных	**Описание**
BLOB	Большой двоичный объект. Может быть длиной до 4 Гбайт
CHAR(n)	Текстовое поле фиксированной длины n. Максимум 2000 символов
DATE	Поле длиной 7 байт, содержащее дату и время
INT	Целое число длиной 38 знаков
NUMBER(n,d)	Число длиной n с d знаками после запятой
VARCHAR(n) или VARCHAR2(n)	Текстовое поле переменной длины до n символов. Максимальное значение n=4000

Определение первичных и альтернативных ключей с помощью оператора ALTER

После того как таблица определена, ее структуру, свойства и ограничения можно изменить, используя оператор ALTER. Так, в листинге 2.2 представлен альтернативный способ определения первичного ключа, при котором сначала определяется таблица, а потом ее определение модифицируется оператором ALTER. Оператор CREATE TABLE определяет все столбцы таблицы CUSTOMER, но ни один из них не указывается в качестве первичного ключа. Затем при помощи оператора ALTER TABLE вводится новое ограничение под названием CustomerPK, которое определяет столбец CustomerID как первичный ключ.

Листинг 2.2.

```
CREATE TABLE CUSTOMER(
    CustomerID      int             NOT NULL,
    Name            char(25)        NOT NULL,
    Street          char(30)        NULL,
```

```
    City            char(35)        NULL,
    State           char(2)         NULL,
    ZipPostalCode   char(5)         NULL,
    Country         varchar(50)     NULL,
    AreaCode        char(3)         NULL,
    PhoneNumber     char(8)         NULL,
    Email           varchar(100)    NULL);

    ALTER TABLE CUSTOMER
        ADD CONSTRAINT CustomerPK PRIMARY KEY (CustomerID);

CREATE TABLE ARTIST(
    ArtistID        int             NOT NULL,
    Name            char(25)        NOT NULL,
    Nationality     varchar(30)     NULL,
    BirthDate       numeric(4,0)    NULL,
    DeceasedDate    numeric(4,0)    NULL,
    CONSTRAINT  ArtistPK    PRIMARY KEY (ArtistID));
```

Имя для ограничения может выбираться произвольно. Однако есть смысл придерживаться некоторым стандартным соглашениям об именовании. В этом пособии, например, имена первичных ключей образуются путем присоединения аббревиатуры PK (Primary Key - первичный ключ) к названию таблицы.

Оба способа определения первичных ключей для таблиц CUSTOMER (листинг 2.1 и листинг 2.2) являются правильными. Различие состоит в том, что во втором листинге разработчик указал явно имя органичения первичного ключа. Ограничению первичного ключа таблицы CUSTOMER (листинг 2.1) также будет дано имя, но выберет его уже СУБД, что очень часто затрудняет задачи администрирования данных. Поэтому второй метод более предпочтителен.

Определение таблицы ARTIST в листинге 2.2 демонстрирует еще один способ определения первичного ключа. Здесь ограничение определяется в теле оператора CREATE, после того как определены все столбцы. Используя этот подход, разработчик может присваивать имена ограничениям в момент создания таблиц, не прибегая к использованию оператора ALTER.

Листинг 2.3.

```
CREATE TABLE CUSTOMER_ARTIST_INT(
    ArtistID        int         NOT NULL,
    CustomerID      int         NOT NULL,
    CONSTRAINT  CustomerArtistPK
                PRIMARY KEY (ArtistID, CustomerID),
    CONSTRAINT  Customer_Artist_Int_ArtistFK
                FOREIGN KEY (ArtistID)
                REFERENCES ARTIST (ArtistID)
                ON DELETE CASCADE,
    CONSTRAINT  Customer_Artist_Int_CustomerFK
                FOREIGN KEY (CustomerID)
                REFERENCES CUSTOMER (CustomerID)
                ON DELETE CASCADE);
```

Композитный ключ определяется путем перечисления имен атрибутов в скобках. Первичный ключ таблицы CUSTOMER_ARTIST_INT представляет собой сочетание {ArtistID, CustomerID}. Композитный первичный ключ не может быть определен с помощью того метода, который представлен в листинге 2.1.

Посредством оператора ALTER можно также определять внешние ключи (см. листинг 2.3). Ограничение Customer_Artist_Int_ArtistFK указывает, что столбец ArtistID является внешний ключом, который указывает на столбец ARTIST.ArtistID. Аббревиатура FK означает внешний ключ (foreign key). В определении внешнего ключа можно указать процедуру обеспечения ссылочной целостности при удалении: ON DELETE CASCADE.

Операторы DROP

Одним из самых полезных операторов определения данных SQL является оператор DROP TABLE. Но он одновременно является и одним из самых опасных, поскольку удаляет таблицу из базы данных *вместе со всеми содержащимися в ней данными*. Если необходимо удалить из базы данных таблицу CUSTOMER и все содержащиеся в ней данные, то можно использовать следующий оператор:

```
DROP TABLE CUSTOMER
```

Оператор DROP TABLE не выполняется, если таблица содержит или может содержать значения, необходимые для соблюдения ограничений ссылочной целостности. Например, столбец ArtistID таблицы ARTIST может содержать значения, необходимые для соблюдения ограничения внешнего ключа Customer_Artist_Int_ArtistFK. Попытка передать СУБД на выполнение оператор DROP TABLE ARTIST окончится неудачей и система выдаст сообщение об ошибке.

Если необходимо удалить таблицу ARTIST, нужно сначала удалить либо ограничение внешнего ключа Customer_Artist_Int_ArtistFK, либо всю таблицу CUSTOMER_ARTIST_INT. Удалить ограничение можно при помощи следующего оператора:

```
ALTER TABLE CUSTOMER_ARTIST_INT
DROP CONSTRAINT Customer_Artist_Int_ArtistFK
```

В другой главе будут рассмотрены дополнительные применения оператора ALTER.

2.2 Средства запроса данных языка SQL

Создав таблицу, можно записывать в нее данные, изменять их значения и удалять данные из таблицы. Предполагается, что данные, приведенные в табл. 1.1-1.3 уже введены в базу данных. Операторы, которые позволяют добавлять, модифицировать и удалить строки в таблицах будут рассмотрены в разделе 2.3.

Чтение заданных столбцов из одиночной таблицы

Следующий оператор запросит (прочитает) три из пяти столбцов таблицы ARTIST:

```
SELECT Name, Nationality, BirthDate
FROM ARTIST;
```

Имена запрашиваемых столбцов перечисляются после ключевого слова SELECT, а имя отношения, из которого считываются данные, указываются после ключевого слова FROM. Результатом этого оператора при использовании данных из таблицы ARTIST (см. табл. 1.1) будет следующая таблица:

Miro	Spanish	1870
Kandinsky	Russian	1854
Frings	US	1700
Klee	German	1900
Moos	US	<NULL>
Tobey	US	<NULL>
Matisse	French	<NULL>
Chagall	French	<NULL>

Результатом работы оператора SELECT всегда является отношение. Этот оператор берет одно или несколько отношений, манипулирует ими определенным образом и выдает на выходе одно отношение. Даже если результатом является всего лишь одно число, это число рассматривается как отношение, имеющее одну строку и один столбец.

Порядок столбцов в результирующей таблице определяется порядком следования их имен после ключевого слова SELECT. Предположим, что порядок имен столбцов в операторе SELECT будет следующим:

```
SELECT Nationality, Name, BirthDate
FROM ARTIST;
```

Результат работы оператора будет следующим:

Spanish	Miro	1870
Russian	Kandinsky	1854
US	Frings	1700
German	Klee	1900
US	Moos	<NULL>
US	Tobey	<NULL>
French	Matisse	<NULL>
French	Chagall	<NULL>

Следующий оператор SELECT извлекает из таблицы ARTIST только столбец Nationality:

```
SELECT Nationality
FROM ARTIST;
```

Результатом будет таблица:

Spanish
Russian
US
German
US
US
French
French

Следует обратить внимание, что в последней таблице есть одинаковые строки. Согласно определению отношения, повторения строк в отношении недопустимо. Однако процесс поиска и удаления таких повторений отнимает много времени. Таким образом, на практике все же приходится сталкиваться с одинаковыми строками.

Если нужно, чтобы СУБД нашла и удалила повторяющиеся строки, при запросе необходимо использовать ключевое слово DISTINCT:

```
SELECT DISTINCT Nationality
FROM ARTIST;
```

Результат будет иметь вид:

Spanish
Russian
US
German
French

2.3 Средства модификации данных языка SQL

Глава 3

Применение SQL в приложениях

3.1 SQL представления

3.2 Триггеры

3.3 Хранимые процедуры

Глава 4

Установка Oracle

Здесь будем писать при инсталляцию Oracle

4.1 Системные и программные требования

4.2 Инсталляция Oracle 10g Express Edition

Глава 5

Работа с базами данных в Oracle 10g Express Edition

5.1 Создание базы данных Oracle 10g Express Edition

5.2 Определение логики приложения

Библиографический список

1. Крёнке Д. Теория и практика построения баз данных. 9-е изд. — СПб.: Питер, 2005 — 859 с.

Учебное издание
Наместников Алексей Михайлович
Построение баз данных в среде Oracle
Практический курс

Редактор Н. А. Евдокимова

Оригинал-макет изготовлен в системе $\LaTeX 2_\varepsilon$.

Изд. лиц. 020640 от 22.10.97. Подписано в печать .
Формат 60×84/16. Бумага писчая. Усл. печ. л. 7,67.
Уч.-изд. л. 6,85. Гарнитура Computer Modern.
Тираж 200 экз. Заказ .

Ульяновский государственный технический университет
432027, Ульяновск, Сев. Венец, 32.

Типография УлГТУ, 432027, Ульяновск, Сев. Венец, 32.