15 Выборка данных

Одной из важнейших характеристик PL/SQL является тесная интеграция с базой данных Oracle в отношении как изменения данных в таблицах, так и выборки данных из таблиц. В этой главе рассматриваются элементы PL/SQL, связанные с выборкой информации из базы данных и ее обработкой в программах PL/SQL.

При выполнении команды SQL из PL/SQL РСУБД Oracle назначает ей приватную рабочую область, а некоторые данные записывает в системную глобальную область (SGA, System Global Area). В приватной рабочей области содержится информация о команде SQL и набор данных, возвращаемых или обрабатываемых этой командой.

PL/SQL предоставляет программистам несколько механизмов доступа к этой рабочей области и содержащейся в ней информации; все они так или иначе связаны с определением курсоров и выполнением операций с ними.

- **Неявные курсоры.** Команда `SELECT...INTO` считывает одну строку данных и присваивает ее в качестве значения локальной переменной программы. Это простейший (и зачастую наиболее эффективный) способ доступа к данным, но он часто ведет к написанию сходных и даже одинаковых SQL-команд `SELECT` во многих местах программы.
- **Явные курсоры.** Запрос можно явно объявить как курсор в разделе объявлений локального блока или пакета. После этого такой курсор можно будет открывать и выбирать из него данные в одной или нескольких программах, причем возможности управления явным курсором шире, чем у неявного.
- **Курсорные переменные.** Курсорные переменные (в объявлении которых задается тип `REF CURSOR`) позволяют передавать из программы в программу *указатель* на результирующий набор строк запроса. Любая программа, для которой доступна такая переменная, может открыть курсор, извлечь из него необходимые данные и закрыть его.
- **Курсорные выражения.** Ключевое слово `CURSOR` превращает команду `SELECT` в набор `REF CURSOR`, который может использоваться совместно с табличными функциями для повышения производительности приложения.
- **Динамические SQL-запросы.** Oracle позволяет динамически конструировать и выполнять запросы с использованием либо встроенного динамического SQL (NDS — см. главу 16), либо программ пакета DMBS_SQL. Этот встроенный пакет описывается в документации Oracle, а также в книге *Oracle Built-in Packages* (O'Reilly).

В этой главе рассказано о неявных и явных курсорах, курсорных переменных и выражениях.

Основные принципы работы с курсорами

Курсор проще всего представить себе как указатель на таблицу в базе данных. Например, следующее объявление связывает всю таблицу `employee` с курсором `employee_cur`:

```
CURSOR employee_cur IS SELECT * FROM employee;
```

Объявленный курсор можно открыть:

```
OPEN employee_cur;
```

Далее из него можно выбирать строки:

```
FETCH employee_cur INTO employee_rec;
```

Завершив работу с курсором, его следует закрыть:

```
CLOSE employee_cur;
```

В этом случае каждая выбранная из курсора запись представляет строку таблицы `employee`. Однако с курсором можно связать любую допустимую команду `SELECT`. В следующем примере в объявлении курсора объединяются три таблицы:

```
DECLARE
   CURSOR joke_feedback_cur
   IS
      SELECT J.name, R.laugh_volume, C.name
        FROM joke J, response R, comedian C
       WHERE J.joke_id = R.joke_id
         AND R.joker_id = C.joker_id;
BEGIN
   ...
END;
```

В данном случае курсор действует не как указатель на конкретную таблицу базы данных — он указывает на виртуальную таблицу или неявное представление, определяемое командой `SELECT`. (Такая таблица называется виртуальной, потому что команда `SELECT` генерирует данные с табличной структурой, но эта таблица существует только временно, пока программа работает с возвращенными командой данными.) Если тройное объединение возвращает таблицу из 20 строк и 3 столбцов, то курсор действует как указатель на эти 20 строк.

Терминология

В PL/SQL имеется множество возможностей выполнения команд SQL, и все они реализованы в программах как курсоры того или иного типа. Прежде чем приступить к их освоению, необходимо познакомиться с методами выборки данных и используемой при этом терминологией.

- **Статический SQL.** Команда SQL называется *статической*, если она полностью определяется во время компиляции программы.
- **Динамический SQL.** Команда SQL называется *динамической*, если она строится и выполняется на стадии выполнения программы, так что в программном коде нет ее фиксированного объявления. Для динамического выполнения команд SQL могут использоваться программы встроенного пакета `DBMS_SQL` (имеющегося во всех версиях Oracle) или встроенный динамический SQL.
- **Результирующий набор строк.** Набор строк с результирующими данными, удовлетворяющими критериям, определяемым командой SQL. Результирующий набор кэшируется в системной глобальной области с целью ускорения чтения и модификации его данных.
- **Неявный курсор.** При каждом выполнении команды DML (`INSERT`, `UPDATE`, `MERGE` или `DELETE`) или команды `SELECT INTO`, возвращающей строку из базы данных прямо

в структуру данных программы, PL/SQL создает неявный курсор. Курсор этого типа называется *неявным*, поскольку Oracle автоматически выполняет многие связанные с ним операции, такие как открытие, выборка данных и даже закрытие.

- **Явный курсор.** Команда `SELECT`, явно определенная в программе как курсор. Все операции с явным курсором (открытие, выборка данных, закрытие и т. д.) в программе должны выполняться явно. Как правило, явные курсоры используются для выборки из базы данных набора строк с использованием статического SQL.
- **Курсорная переменная.** Объявленная программистом переменная, указывающая на объект курсора в базе данных. Ее значение (то есть указатель на курсор или результирующий набор строк) во время выполнения программы может меняться, как у всех остальных переменных. В разные моменты времени курсорная переменная может указывать на разные объекты курсора. Курсорную переменную можно передать в качестве параметра процедуре или функции. Такие переменные очень полезны для передачи результирующих наборов из программ PL/SQL в другие среды (например, Java или Visual Basic).
- **Атрибут курсора.** Атрибут курсора имеет форму *%имя_атрибута* и добавляется к имени курсора или курсорной переменной. Это что-то вроде внутренней переменной Oracle, возвращающей информацию о состоянии курсора — например о том, открыт ли курсор, или сколько строк из курсора вернул запрос. У явных и неявных курсоров и в динамическом SQL в атрибутах курсоров существуют некоторые различия, которые рассматриваются в этой главе.
- **SELECT FOR UPDATE.** Разновидность обычной команды `SELECT`, устанавливающая блокировку на каждую возвращаемую запросом строку данных. Пользоваться ею следует только в тех случаях, когда нужно «зарезервировать» запрошенные данные, чтобы никто другой не мог изменить их, пока с ними работаете вы.
- **Пакетная обработка.** В Oracle8i и выше PL/SQL поддерживает запросы с секцией `BULK COLLECT`, позволяющей за один раз выбрать из базы данных более одной строки.

Типичные операции с запросами и курсорами

Независимо от типа курсора процесс выполнения команд SQL всегда состоит из одних и тех же действий. В одних случаях PL/SQL производит их автоматически, а в других, как, например, при использовании явного курсора, они явно организуются программистом.

- **Разбор.** Первым шагом при обработке команды SQL должен быть ее *разбор* (синтаксический анализ), то есть проверка ее корректности и формирование плана выполнения (с применением оптимизации по синтаксису или по стоимости в зависимости от того, какое значение параметра `OPTIMIZER_MODE` задал администратор базы данных).
- **Привязка.** *Привязкой* называется установление соответствия между значениями программы и параметрами команды SQL. Для статического SQL привязка производится ядром PL/SQL. Привязка параметров в динамическом SQL выполняется явно с использованием переменных привязки.
- **Открытие.** При открытии курсора определяется результирующий набор строк команд SQL, для чего используются переменные привязки. Указатель активной или текущей строки указывает на первую строку результирующего набора. Иногда явное открытие курсора не требуется; ядро PL/SQL выполняет эту операцию автоматически (так происходит в случае применения неявных курсоров и встроенного динамического SQL).
- **Выполнение.** На этой стадии команда выполняется ядром SQL.

- **Выборка.** Выборка очередной строки из результирующего набора строк курсора осуществляется командой FETCH. После каждой выборки PL/SQL перемещает указатель на одну строку вперед. Работая с явными курсорами, помните, что и после завершения перебора всех строк можно снова и снова выполнять команду FETCH, но PL/SQL ничего не будет делать (и не станет инициировать исключение) — для выявления этого условия следует использовать атрибуты курсора.
- **Закрытие.** Операция закрывает курсор и освобождает используемую им память. Закрытый курсор уже не содержит результирующий набор строк. Иногда явное закрытие курсора не требуется, последовательность PL/SQL делает это автоматически (для неявных курсоров и встроенного динамического SQL).

На рис. 15.1 показано, как некоторые из этих операций используются для выборки информации из базы данных в программу PL/SQL.

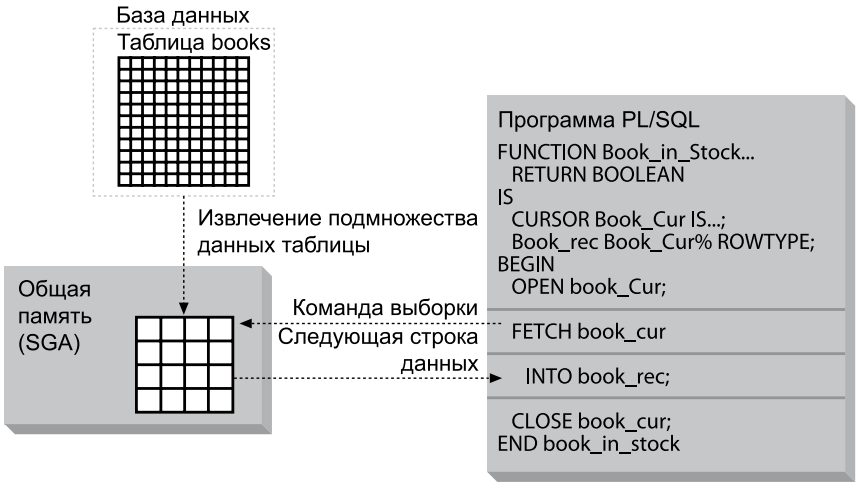

Рис. 15.1. Упрощенная схема выборки данных с использованием курсора

Знакомство с атрибутами курсоров

В этом разделе перечисляются и вкратце описываются атрибуты курсоров. Более подробные описания атрибутов для каждого вида курсоров приводятся в этой главе, а также в главах 14 и 16.

PL/SQL поддерживает шесть атрибутов курсоров, перечисленных в табл. 15.1.

Таблица 15.1. Атрибуты курсоров

Имя	Что возвращает
%FOUND	TRUE, если успешно выбрана хотя бы одна строка; в противном случае возвращает FALSE
%NOTFOUND	TRUE, если команда не выбрала ни одной строки; в противном случае возвращает FALSE
%ROWCOUNT	Количество строк, выбранных из курсора на данный момент времени
%ISOPEN	TRUE, если курсор открыт; в противном случае возвращает FALSE
%BULK_ROWCOUNT	Количество измененных записей для каждого элемента исходной коллекции, заданной в команде FORALL
%BULK_ EXCEPTIONS	Информация об исключении для каждого элемента исходной коллекции, заданной в команде FORALL

Чтобы обратиться к атрибуту курсора, укажите в виде его префикса имя курсора или курсорной переменной и символ %:

имя_курсора%имя_атрибута

В качестве имен неявных курсоров используется префикс SQL, например SQL%NOTFOUND.

Ниже приведены краткие описания всех атрибутов курсоров.

Атрибут %FOUND

Атрибут %FOUND возвращает информацию о состоянии последней операции FETCH с курсором. Если последний вызов FETCH для курсора вернул строку, то возвращается значение TRUE, а если строка не была получена, возвращается FALSE.

Если курсор еще не был открыт, база данных инициирует исключение INVALID_CURSOR.

В следующем примере я перебираю все строки курсора caller_cur, присваиваю все звонки, введенные до сегодняшнего дня для конкретного позвонившего, после чего перехожу к следующей записи. При достижении последней записи атрибут %FOUND явного курсора возвращает FALSE, и выполнение простого цикла прерывается. Атрибут %FOUND неявного курсора также проверяется после команды UPDATE:

```
FOR caller_rec IN caller_cur
LOOP
      UPDATE call
      SET caller_id = caller_rec.caller_id
    WHERE call_timestamp < SYSDATE;

   IF SQL%FOUND THEN
      DBMS_OUTPUT.PUT_LINE ('Calls updated for ' || caller_rec.caller_id);
   END IF;
END LOOP;
```

Атрибут %NOTFOUND

Атрибут %NOTFOUND по смыслу противоположен %FOUND: он возвращает TRUE, если последняя операция FETCH с курсором не вернула строки (как правило, из-за того, что последняя строка уже была прочитана). Если курсор не может вернуть строку из-за ошибки, инициируется соответствующее исключение.

Если курсор еще не был открыт, база данных инициирует исключение INVALID_CURSOR.

В каких случаях следует использовать %FOUND, а когда предпочтение отдается %NOTFOUND? Используйте ту формулировку, которая более естественно подходит для вашего кода. В предыдущем примере для выхода из цикла использовалась следующая команда:

```
EXIT WHEN NOT caller_cur%FOUND;
```

Альтернативная — и возможно, более понятная — формулировка могла бы использовать %NOTFOUND:

```
EXIT WHEN caller_cur%NOTFOUND;
```

Атрибут %ROWCOUNT

Атрибут %ROWCOUNT возвращает количество записей, прочитанных из курсора к моменту запроса атрибута. При исходном открытии курсора атрибут %ROWCOUNT равен 0. При обращении к атрибуту %ROWCOUNT курсора, который еще не был открыт, инициируется исключение INVALID_CURSOR. После выборки каждой записи %ROWCOUNT увеличивается на единицу.

Используйте атрибут %ROWCOUNT для проверки того, что программа прочитала (или обновила в случае DML) нужное количество строк, или для прерывания выполнения после заданного числа итераций. Пример:

```
BEGIN
   UPDATE employees SET last_name = 'FEUERSTEIN';
   DBMS_OUTPUT.PUT_LINE (SQL%ROWCOUNT);
END;
```

Атрибут %ISOPEN

Атрибут `%ISOPEN` возвращает `TRUE`, если курсор открыт; в противном случае возвращается `FALSE`.

Приведем типичный пример использования этого атрибута, который гарантирует, что курсор не остается открытым, если в программе произойдет что-то непредвиденное:

```
DECLARE
   CURSOR happiness_cur IS SELECT simple_delights FROM ...;
BEGIN
   OPEN happiness_cur;
   ...
   IF happiness_cur%ISOPEN THEN ...
EXCEPTION
      WHEN OTHERS THEN
           IF happiness_cur%ISOPEN THEN
                 close happiness_cur;
           END IF;
END;
```

Атрибут %BULK_ROWCOUNT

Атрибут `%BULK_ROWCOUNT`, предназначенный для использования с командой `FORALL`, возвращает количество строк, обработанных при каждом выполнении DML. Этот атрибут обладает семантикой ассоциативного массива (см. главу 21).

Атрибут %BULK_EXCEPTIONS

Атрибут `%BULK_EXCEPTIONS`, также предназначенный для использования с командой `FORALL`, возвращает информацию об исключении, которое может быть инициировано при каждом выполнении DML. Этот атрибут (см. главу 21) обладает семантикой ассоциативного массива записей.

На атрибуты курсоров можно ссылаться в коде PL/SQL, как показано в приведенных примерах, но вы не сможете напрямую обращаться к ним в команде SQL. Например, при попытке использовать атрибут %ROWCOUNT в секции WHERE команды SELECT:

```
SELECT caller_id, company_id FROM caller
WHERE company_id = company_cur%ROWCOUNT;
```

компилятор выдает ошибку PLS-00229: Attribute expression within SQL expression.

Ссылки на переменные PL/SQL в курсорах

Поскольку курсор должен быть связан с командой `SELECT`, в нем всегда присутствует хотя бы одна ссылка на таблицу базы данных; по ней (и по содержимому условия `WHERE`) Oracle определяет, какие строки будут возвращены в составе активного набора. Однако это не означает, что команда `SELECT` может возвращать информацию только из базы данных.

Список выражений, задаваемых между ключевыми словами `SELECT` и `FROM`, называется *списком выборки*. Во встроенном SQL список выборки может содержать столбцы и выражения (вызовы SQL-функций для этих столбцов, константы и т. д.). В PL/SQL список выборки обычно содержит переменные PL/SQL и сложные выражения.

На локальные данные программы PL/SQL (переменные и константы), а также на переменные привязки хост-среды можно ссылаться в предложениях `WHERE`, `GROUP BY` и `HAVING` команды `SELECT` курсора. Ссылки на переменные PL/SQL можно (и должно) *уточнять* именем ее области видимости (именем процедуры, именем пакета и т. д.), особенно в командах SQL.

Выбор между явным и неявным курсорами

Все последние годы знатоки Oracle (включая и авторов данной книги) убежденно доказывали, что для однострочной выборки данных никогда не следует использовать неявные курсоры. Это мотивировалось тем, что неявные курсоры, соответствуя стандарту ISO, всегда выполняют две выборки, из-за чего они уступают по эффективности явным курсорам.

Так утверждалось и в первых двух изданиях этой книги, но пришло время нарушить эту традицию (вместе со многими другими). Начиная с Oracle8, в результате целенаправленных оптимизаций неявные курсоры выполняются даже *эффективнее* эквивалентных явных курсоров.

Означает ли это, что теперь всегда лучше пользоваться неявными курсорами? Вовсе нет. В пользу применения явных курсоров существуют убедительные доводы.

- В некоторых случаях явные курсоры эффективнее неявных. Часто выполняемые критические запросы лучше протестировать в обеих формах, чтобы точно выяснить, как лучше выполнять каждый из них в каждом конкретном случае.
- Явными курсорами проще управлять из программы. Например, если строка не найдена, Oracle не инициирует исключение, а просто принудительно завершает выполняемый блок.

Поэтому вместо формулировки «явный или неявный?» лучше спросить: «инкапсулированный или открытый?» И ответ будет таким: всегда инкапсулируйте однострочные запросы, скрывая их за интерфейсом функции (желательно пакетной) и возвращая данные через `RETURN`.

Не жалейте времени на инкапсуляцию запросов в функциях, желательно пакетных. Это позволит вам и всем остальным разработчикам вашей группы просто вызвать функцию, когда появится необходимость в данных. Если Oracle изменит правила обработки запросов, а ваши предыдущие наработки станут бесполезными, достаточно будет изменить реализацию всего одной функции.

Работа с неявными курсорами

При каждом выполнении команды DML (`INSERT`, `UPDATE`, `MERGE` или `DELETE`) или команды `SELECT INTO`, возвращающей строку из базы данных в структуру данных программы, PL/SQL автоматически создает для нее курсор. Курсор этого типа называется неявным, поскольку Oracle автоматически выполняет многие связанные с ним операции, такие как выделение курсора, его открытие, выборку строк и т. д.

Выполнение команд DML с помощью неявных курсоров рассматривается в главе 14. В этой главе рассматриваются только неявные запросы SQL.

Неявный курсор — это команда `SELECT`, обладающая следующими характеристиками:

- Команда `SELECT` определяется в исполняемом разделе блока, а не в разделе объявлений, как явные курсоры.

- В команде содержится предложение INTO (или `BULK COLLECT INTO`), которое относится к языку PL/SQL (а не SQL) и представляет собой механизм передачи данных из базы в локальные структуры данных PL/SQL.
- Команду `SELECT` не нужно открывать, выбирать из нее данные и закрывать; все эти операции осуществляются автоматически.

Общая структура неявного запроса выглядит так:

```
SELECT список_столбцов
  [BULK COLLECT] INTO PL/SQL список_переменных
  ...продолжение команды SELECT...
```

Если в программе используется неявный курсор, Oracle автоматически открывает его, выбирает строки и закрывает; над этими операциями программист не властен. Однако он может получить информацию о последней выполненной команде SQL, анализируя значения атрибутов неявного курсора SQL, как рассказывается далее в этой главе.

В следующих разделах, говоря о неявных курсорах, мы будем иметь в виду команду SELECT INTO, которая извлекает (или пытается извлечь) одну строку данных. В главе 21 обсуждается ее разновидность SELECT BULK COLLECT INTO, которая позволяет с помощью одного неявного запроса получить несколько строк данных.

Примеры неявных курсоров

Неявные курсоры часто используются для поиска данных на основе значений первичного ключа. В следующем примере выполняется поиск названия книги по ее коду ISBN:

```
DECLARE
   l_title books.title%TYPE;
BEGIN
   SELECT title
     INTO l_title
     FROM books
    WHERE isbn = '0-596-00121-5';
```

После того как название книги будет выбрано и присвоено локальной переменной `l_title`, с последней можно работать как с любой другой переменной: изменять ее значение, выводить на экран или передавать для обработки другой программе PL/SQL.

Пример неявного курсора, извлекающего из таблицы строку данных и помещающего ее в запись программы:

```
DECLARE
   l_book books%ROWTYPE;
BEGIN
   SELECT *
     INTO l_book
     FROM books
    WHERE isbn = '0-596-00121-5';
```

Из запроса также можно получить информацию уровня групп. Например, следующий запрос вычисляет и возвращает сумму окладов по отделу. И снова PL/SQL создает для него неявный курсор:

```
SELECT SUM (salary)
  INTO department_total
  FROM employees
 WHERE department_id = 10;
```

Благодаря тесной интеграции PL/SQL с базой данных Oracle с помощью запросов из базы данных можно извлекать и сложные типы данных, такие как объекты и коллекции. Все эти примеры демонстрируют применение неявных запросов для выборки данных

одной строки. Если вы хотите получить более одной строки, используйте либо явный курсор, либо конструкцию BULK COLLECT INTO (см. главу 21).

Как упоминалось ранее, я рекомендую всегда «скрывать» однострочные запросы за функциональным интерфейсом. Эта концепция подробно рассматривается в разделе «Выбор между явным и неявным курсорами» этой главы.

Обработка ошибок при использовании неявных курсоров

Команда SELECT, выполняемая как неявный курсор, представляет собой «черный ящик». Она передается в базу данных и возвращает одну строку информации. Что происходит с курсором, как он открывается, получает данные и закрывается, вы не знаете. Также приходится смириться с тем, что в двух стандартных ситуациях при выполнении команды SELECT автоматически инициируются исключения:

- По запросу не найдено ни одной строки. В этом случае Oracle инициирует исключение NO_DATA_FOUND.
- Команда SELECT вернула несколько строк. В этом случае Oracle инициирует исключение TOO_MANY_ROWS.

В каждом из этих случаев (как и при возникновении любого другого исключения, инициированного при выполнении команды SQL) выполнение текущего блока прерывается, и управление передается в раздел исключений. Этим процессом вы не управляете; у вас даже нет возможности сообщить Oracle, что данный запрос может не вернуть ни одной строки, и это не является ошибкой. Таким образом, если вы используете неявный курсор, в раздел исключений обязательно нужно включить код перехвата и обработки этих двух исключений (а возможно, и других исключений, как того требует логика программы).

В следующем блоке кода производится поиск названия книги по ее коду ISBN с обработкой возможных исключений:

```
DECLARE
   l_isbn books.isbn%TYPE := '0-596-00121-5';
   l_title books.title%TYPE;
BEGIN
   SELECT title
     INTO l_title
     FROM books
    WHERE isbn = l_isbn;
EXCEPTION
   WHEN NO_DATA_FOUND
   THEN
      DBMS_OUTPUT.PUT_LINE ('Неизвестная книга: ' || l_isbn);
   WHEN TOO_MANY_ROWS
   THEN
      /* Пакет определяется в errpkg.pkg */
      errpkg.record_and_stop ('Нарушение целостности данных для: ' || l_isbn);
      RAISE;
END;
```

При работе с неявным курсором программист может сделать некоторые рискованные предположения об извлекаемых данных. Приведем пару примеров:

- В таблице не может быть более одной книги с указанным кодом ISBN — ведь это гарантируется заданным для таблицы ограничением.
- В таблице обязательно имеется строка с данными о содержащейся в ней книге, поэтому не стоит беспокоиться об исключении NO_DATA_FOUND.

Последствия таких предположений часто оказываются плачевными, поскольку программисты не включают в программы соответствующие обработчики исключений для неявных запросов.

Конечно, в настоящий момент и при текущем состоянии данных запрос может вернуть ровно одну строку. Но если данные изменятся, запрос, к примеру, команда SELECT может вернуть две строки вместо одной, программа выдаст исключение, которое не будет обработано — и это может создать проблемы в коде.

Как правило, при использовании неявных курсоров желательно всегда включать в программу обработчики исключений NO_DATA_FOUND и TOO_MANY_ROWS. В обобщенной формулировке можно сказать, что в программе должны присутствовать обработчики всех исключений, которые в ней могут произойти. А вот действия, выполняемые этими обработчиками, могут быть самыми разными. Возьмем все тот же код, извлекающий название книги по заданному коду ISBN. Ниже он реализован в виде функции с двумя обработчиками ошибок. Обработчик NO_DATA_FOUND возвращает значение, а обработчик TOO_MANY_ROWS записывает ошибку в журнал и повторно инициирует исключение, прерывая работу функции. (О пакете errpkg.pkg более подробно рассказано в главе 6.)

```
FUNCTION book_title (isbn_in   IN   books.isbn%TYPE
)
   RETURN books.title%TYPE
IS
   return_value   book.title%TYPE;
BEGIN
  SELECT title
    INTO return_value
    FROM books
    WHERE isbn = isbn_in;

    RETURN return_value;
EXCEPTION
    WHEN NO_DATA_FOUND
    THEN
      RETURN NULL;
    WHEN TOO_MANY_ROWS
    THEN
      errpkg.record_and_stop ('Нарушение целостности данных для: '
                || isbn_in);
      RAISE;
END;
```

Почему эти два обработчика ведут себя по-разному? Дело в том, что функция должна вернуть название книги, которое никогда не может быть представлено значением NULL. Для проверки используется код ISBN («существует ли книга с данным кодом ISBN?»), поэтому если книга по ISBN не найдена, функция не должна инициировать исключение. В этом нет ничего плохого. Например, логика программы может быть такой: «если книги с заданным ISBN не существует, используем его для новой книги», а возможная реализация может выглядеть так:

```
IF book_title ('0-596-00121-7') IS NULL
THEN ...
```

Иначе говоря, то, что запрос не вернул ни одной строки, не всегда свидетельствует об ошибке.

С другой стороны, если запрос сгенерировал исключение TOO_MANY_ROWS, мы сталкиваемся с настоящей проблемой: в базе не может быть двух книг с одинаковыми кодами ISBN. Поэтому в данном случае информация об ошибке записывается в журнал, а программа прекращает свою работу.