Харківський національний університет імені В.Н. Каразіна

Лекция

«Детальный контроль доступа в базах данных»

1. Средства детального контроля доступа в СУБД Oracle и преимущества их использования.
2. Реализация средств детального контроля доступа в СУБД Oracle.

Средства детального контроля доступа в СУБД Oracle и преимущества их использования

В различных публикациях средства детального контроля доступа имеют названия:

- тщательный (детальный) контроль доступа (*Fine Grained Access Control – FGAC*);
- виртуальная приватная база данных (*Virtual Private Database – VPD*);
- защита на уровне строк (*Row Level Security*), или PL/SQL пакет DBMS_RLS.

Средства детального контроля доступа

Таблица 1. – Варианты запросов, получаемые с использованием FGAC

Пользователь зарегистрирован как:	***Запрос переписывается так...***	***Комментарии***

Сотрудник	`select *`
	` from (select * from emp`
	`  where ename = USER)`
		Рядовые сотрудники могут просматривать только собственные записи.

Руководитель подразделения	`select *`
	` from (select * from emp`
	`  where mgr =`
	`  (select empno from emp`
	`  where ename = USER)`
	`  or ename = USER)`
		Руководители подразделений могут просматривать свои записи и записи сотрудников своего подразделения.

Сотрудники отдела кадров	`select *`
	` from (select * from emp`
	`  where deptno =`
	`SYS_CONTEXT('OurApp',`
	`'ptno')`
		Сотрудники отдела кадров могут видеть все записи в данном подразделении. В этом примере представлен способ получения значений переменных из контекста приложения с помощью встроенной функции `SYS_CONTEXT()`.

Средства детального контроля доступа

Преимущества использования средств детального контроля доступа:

- простота сопровождения;
- контроль доступа выполняется на сервере;
- упрощение разработки приложений;
- эволюционная разработка приложений;
- отказ от совместно используемых учетных записей;
- предоставление доступа к приложению как к службе:

возможны следующие варианты:

- установить, сконфигурировать и поддерживать отдельные экземпляры базы данных для каждого клиента;
- переписать все используемые приложением хранимые процедуры так, чтобы они работали с правами вызывающего, и создать для каждого клиента отдельную схему;
- использовать один экземпляр базы данных и одну схему со средствами детального контроля доступа.

Реализация средств детального контроля доступа

Средства тщательного контроля доступа, начиная с Oracle 8i, реализуются с помощью двух конструкций:

- *контекста*;
- *правил (политик) безопасности (защиты).*

Контекст

Контекст – это именованный набор пар «*параметр/значение*» (*атрибут/значение*). В Oracle каждый конкретный подобный набор называется *пространством имен* (*namespace*).

Элементы пространства имен называются *атрибутами* (не путать с атрибутами таблиц), способными принимать значения.

Идея, лежащая в основе использования контекстов, хотя и достаточно проста, но позволяет обеспечивать серьезную защиту.

Суть ее состоит в следующем: определяется *контекст* – список переменных в памяти, значения которых привязаны к сеансам, при этом сеанс может получить текущие значения этих переменных, вызывая функцию SYS_CONTEXT(), а переменные в контексте могут устанавливаться только путем вызова процедуры, связанной с этим контекстом.

Реализация средств детального контроля доступа

Системная функция SYS_CONTEXT имеет следующий формат:

```
SYS_CONTEXT (простр_имен, парам, [длина]),
```

где ПРОСТР_ИМЕН – имя контекста;
ПАРАМ – имя параметра контекста (любая строка длиной не более 30 байт);
ДЛИНА – позволяет указывать длину возвращаемых данных, если она превышает 250 байт. Максимальное значение аргумента длина – 4000 байт.

Один контекст, с названием USERENV, создавать явным образом не требуется. Он доступен любому сеансу связи с СУБД Oracle в виде готового набора значений, разрешающего только чтение, но не правку. Он позволяет узнать всевозможные сведения о сеансе, полезные для прикладного программирования.

Реализация средств детального контроля доступа

Пример информации, которую можно получить из контекста USERENV:

```
SELECT
SYS_CONTEXT ('userenv', 'AUTHENTICATION_TYPE') authent -- Как аутентифицирован пользователь: средствами ОС (OS) или СУБД (DATABASE)
,SYS_CONTEXT ('userenv', 'CURRENT_SCHEMA') curr_schema -- Используемая по умолчанию схема сеанса
,SYS_CONTEXT ('userenv', 'CURRENT_USER') curr_user -- Имя пользователя базы данных, чьи привилегии в настоящий момент активны
,SYS_CONTEXT ('userenv', 'DB_NAME') db_name -- Имя базы данных
,SYS_CONTEXT ('userenv', 'HOST' ) host      -- Имя хост-машины
,SYS_CONTEXT ('userenv', 'IP_ADDRESS') ip_address -- Адрес IP, с которого подключился пользователь
,SYS_CONTEXT ('userenv', 'OS_USER') os_user -- Учетная запись операционной системы, инициировавшая сеанс базы данных
,SYS_CONTEXT ('userenv', 'CURRENT_SCHEMAID') curr_schemaid -- Идентификатор используемой по умолчанию схема сеанса
FROM dual;
```

Результат запроса:

	AUTHENT	CURR_SCHEMA	CURR_USER	DB_NAME	HOST	IP_ADDRESS	OS_USER	CURRENT_SCHEMAID
1	DATABASE	AVT2	AVT2	vitaly	WORKGROUP\VIT	127.0.0.1	VIT\vitaly	208

Реализация средств детального контроля доступа

Установка контекста осуществляется с помощью процедуры `SET_CONTEXT` стандартного пакета `DBMS_SESSION`:

```
DBMS_SESSION.SET_CONTEXT (
namespace VARCHAR2
attribute VARCHAR2,
value    VARCHAR2,
username VARCHAR2,
client_id VARCHAR2);
```

Параметр	***Описание***
`namespace`	Имя контекста приложения (до 30 байт).
`attribute`	Атрибут контекста приложения (до 30 байт).
`value`	Значение контекста приложения (до 4 Кбайт).
`username`	Имя пользователя базы данных, которому доступен контекст приложения. По умолчанию: `NULL`.
`client_id`	Особый для приложения client_id атрибут прикладного контекста (максимум 64 байта). Значение по умолчанию: `NULL`.

Данная процедура не должна быть доступна произвольному пользователю!

Реализация средств детального контроля доступа

- Проверить и найти конкретный атрибут контекста можно с помощью системной функции **SYS_CONTEXT**.

- Для каждого контекста требуется указать специальную «доверительную» программную единицу: *процедуру*, *функцию* или *пакет*.
  Именно из тела этой программной единицы Oracle разрешит обращаться к процедуре **DBMS_SESSION.SET_CONTEXT**.

  Такое решение принято в целях безопасности, так как доступ к хранимым программным единицам регулируется уже готовым механизмом привилегий.

Реализация средств детального контроля доступа

Проверка:

```
SQL> CONNECT scott/tiger
Connected.

SQL> SELECT SYS_CONTEXT ('mycontext', 'sesame') FROM dual;

SYS_CONTEXT('MYCONTEXT','SESAME')
--------------------------------------------------------
```

Как видим результат – пустая строка.
Исполним следующие команды:

```
SQL> EXECUTE sys.set_mycontext_value ('sesame', '12345')
PL/SQL procedure successfully completed.
SQL> SELECT SYS_CONTEXT ('mycontext', 'sesame') FROM dual;

SYS_CONTEXT('MYCONTEXT','SESAME')
--------------------------------------------------------
12345
```

С помощью контекста MYCONTEXT и доступной ему процедуры пользователь SCOTT ввел значение, которое сможет читать и переустанавливать в собственном сеансе вплоть до завершения.
Другой сеанс пользователя SCOTT создаст и будет использовать с помощью этого же контекста свои значения, то есть *значения контекста являются собственностью сеанса*.

Реализация средств детального контроля доступа

Пример создания контекста

Положим, доверительной программной единицей должна быть процедура SET_MYCONTEXT_VALUE:

```
CONNECT / as sysdba
SQL> CREATE OR REPLACE CONTEXT mycontext USING set_mycontext_value;
Context created
```

Процедура не обязана существовать в момент создания контекста. Ее можно создать позже:

```
SQL> CREATE OR REPLACE PROCEDURE set_mycontext_value (
  2    par IN VARCHAR2
  3  , val IN VARCHAR2
  4  )
  5  AS
  6  BEGIN DBMS_SESSION.SET_CONTEXT('mycontext', par, val);
  7  END;
  8  /
 Procedure created

SQL> GRANT EXECUTE ON set_mycontext_value TO scott;
Grant succeeded
```

Реализация средств детального контроля доступа

Глобальный контекст сеанса

«Обычный» контекст сеанса имеет своею областью действия отдельный сеанс. Иногда этого разработчику приложения мало.

Как, например, запретить сеансу самостоятельно выставлять значение атрибута и предоставить ему только чтение, а значение задавать из другого сеанса?

Такую возможность обеспечивает *глобальный* контекст сеанса, называемый еще *контекстом приложения* (как и «обычный» контекст, тоже).

Пример использования глобального контекста:

```
CONNECT / AS SYSDBA

CREATE OR REPLACE CONTEXT globalcontext
USING globalcontext_pckg ACCESSED GLOBALLY
/
CREATE OR REPLACE PACKAGE globalcontext_pckg AS
   PROCEDURE set_value (
     par VARCHAR2
   , val VARCHAR2
   , usr VARCHAR2
   , usrid VARCHAR2
   );
END;
/
```

Реализация средств детального контроля доступа

```
CREATE OR REPLACE PACKAGE BODY globalcontext_pckg AS
   PROCEDURE set_value (
     par VARCHAR2
   , val VARCHAR2
   , usr VARCHAR2
   , usrid VARCHAR2
   )
   AS
   BEGIN
    DBMS_SESSION.SET_CONTEXT (
     'globalcontext'
   , par
   , val
   , usr
   , usrid
   );
   END;
END;
/

EXECUTE globalcontext_pckg.set_value ('sesame','123','SCOTT','XYZ32A6')
```

Реализация средств детального контроля доступа

Проверка:

```
SQL> CONNECT scott/tiger
Connected.
SQL> SELECT SYS_CONTEXT('globalcontext','sesame') as valconext FROM dual;
VALCONEXT
--------------------------------------------------------------------------------

SQL> EXECUTE DBMS_SESSION.SET_IDENTIFIER ('XYZ32A6');
PL/SQL procedure successfully completed.
SQL> SELECT SYS_CONTEXT ('globalcontext', 'sesame') as valconext FROM
dual;
VALCONEXT
--------------------------------------------------------------------------------
123

SQL> EXECUTE DBMS_SESSION.SET_IDENTIFIER ('XYZ32A6ZZZ');
PL/SQL procedure successfully completed.
SQL> SELECT SYS_CONTEXT('globalcontext','sesame') as valconext FROM dual;
VALCONEXT
--------------------------------------------------------------------------------
```

Реализация средств детального контроля доступа

Этот пример демонстрирует несколько важных моментов.

1. То, что контекст глобальный, было указано словами **`ACCESSED GLOBALLY`** при его создании.
2. В процедуре **`DBMS_SESSION.SET_CONTEXT`** *именно для глобального контекста существуют два дополнительных параметра*. *Первый* сообщает, сеансам какому пользователю будет доступен этот контекст (для каждого такого пользователя нужно будет выполнить отдельный вызов `SET_CONTEXT`), а *второй* – условное значение, которое необходимо будет сообщить для возможности прочитать установленное другим сеансом значения атрибута, *своего рода пароль*.
3. Сообщение этого условного значения выполняется специальной процедурой **`DBMS_SESSION.SET_IDENTIFIER`**.

Таким образом, мало войти в СУБД под «правильным» пользователем. Для того чтобы получить в сеансе значение желаемого атрибута (глобального контекста), нужно будет еще сообщить условную строку.

Реализация средств детального контроля доступа

Технология RLS

Технология *RLS*, появившись в Oracle 8i, позволяла задавать *правила* (*политики*) *безопасности* (*защиты*) для таблиц базы данных (и отдельных типов операций над таблицами), ограничивающие для пользователя возможность чтения или изменения определенных строк в этих таблицах. Эта технология стала очень полезным инструментом для администратора баз данных, поэтому в старших версиях ее возможности были расширены. Функциональность *RLS* реализована в основном с помощью встроенного пакета **`DBMS_RLS`**.

Технология RLS включает в себя три основных элемента:

- ***политику*** (*policy*) − декларативная команда, которая определяет, как и когда следует применять ограничения пользовательского доступа для выборки, вставок, удалений, изменений или комбинаций перечисленных операций (например, можно запретить пользователю выполнять операцию UPDATE, не ограничивая его возможности по выборке, или ограничить ему доступ к выбору данных из определенного столбца, не ограничивая выборку из остальных столбцов);
- ***функцию политики безопасности*** (*policy function*) − хранимая функция, которая вызывается в случае, когда выполняются условия, заданные в политике безопасности;
- ***предикат*** (*predicate*) − строка, которая генерируется функцией политики безопасности, и которую Oracle автоматически присоединяет в конец предложения WHERE выполняемых пользователем операторов SQL.

Ключевым моментом, обеспечивающим высокую надежность и полноту технологии *RLS*, является то, что Oracle автоматически применяет предикат к пользовательскому SQL-оператору.

Реализация средств детального контроля доступа

Замечания.

1. Следует помнить, что для пользователя SYS (или INTERNAL) соответствующие функции политик безопасности не вызываются. Эти пользователи всегда могут читать и изменять все данные (если не используется *Oracle Database Vault*).
2. Чтобы использовать средства детального контроля доступа, разработчику, помимо стандартных ролей **CONNECT** и **RESOURCE** (или соответствующих им привилегий), необходимы следующие привилегии:

- **EXECUTE_CATALOG_ROLE**. Эта роль позволяет разработчику выполнять подпрограммы пакета `DBMS_RLS`. Достаточно также, подключившись как **SYS**, предоставить пользователю привилегию на выполнение пакета `DBMS_RLS`.
- **CREATE ANY CONTEXT**. Эта привилегия позволяет разработчику создавать контексты приложений.

Реализация средств детального контроля доступа

Пример. Пусть необходимо, чтобы пользователи могли видеть данные только тех сотрудников, чья заработная плата не превышает 1500 у.е.

Предположим, что пользователь вводит запрос:

```
select * from emp;
```

Хотелось бы, чтобы средства *RLS* прозрачно преобразовывали этот запрос в такой:

```
select * from emp where sal <= 1500;
```

От имени пользователя **SCOTT** создадим функцию **authorized_emps**.

```
SQL> CREATE OR REPLACE FUNCTION authorized_emps (
  2     p_schema_name    IN   VARCHAR2,
  3     p_object_name    IN   VARCHAR2
  4  )
  5  RETURN VARCHAR2
  6  IS
  7  BEGIN
  8  RETURN 'SAL <= 1500';
  9  END;
 10  /

Function created
```

Реализация средств детального контроля доступа

При выполнении функция возвращает предикат: SAL <= 1500.
Проверим это, используя тестовый сценарий:

```
SQL > set serveroutput on
SQL>
SQL> DECLARE
  2     l_return_string   VARCHAR2 (2000);
  3  BEGIN
  4     l_return_string := authorized_emps ('X', 'X');
  5     DBMS_OUTPUT.put_line ('Return String = ' || l_return_string);
  6  END;
  7  /
Return String = SAL <= 1500
PL/SQL procedure successfully completed
```

Имея функцию, возвращающую предикат, можно перейти к следующему шагу: созданию *политики безопасности*, также *называемой политикой RLS* или просто *политикой*. Эта политика определяет, когда и как предикат будет применяться к командам SQL.

Реализация средств детального контроля доступа

Для определения безопасности на уровне строк для таблицы EMP используем код:

```
SQL> BEGIN
  2  DBMS_RLS.add_policy (object_schema   => 'SCOTT',
  3  object_name          => 'EMP',
  4  policy_name          => 'EMP_POLICY',
  5  function_schema      => 'SCOTT',
  6  policy_function      => 'AUTHORIZED_EMPS',
  7  statement_types      => 'INSERT, UPDATE, DELETE, SELECT'
  8  );
  9  END;
 10  /

PL/SQL procedure successfully completed.
```

В нем добавляется политика EMP_POLICY (строка 4) для таблицы EMP (строка 3), принадлежащей схеме SCOTT (строка 2). Эта политика будет применять фильтр, задаваемый функцией AUTHORIZED_EMPS (строка 6), принадлежащей схеме SCOTT (строка 5), при выполнении любым пользователем операции INSERT, UPDATE, DELETE или SELECT (строка 7).

Процедуры стандартного пакета ADD_POLICY имеет следующие параметры:

- **object_schema** – это имя схемы (в нашем случае – 'SCOTT'), которая содержит таблицу 'EMP' (параметр – **object_name**), подлежащую защите с помощью политики;
- **policy_name** – название политики, которая добавляется к таблице (в нашем случае – 'EMP_POLICY');
- **function_schema** – имя схемы, владеющей функцией правил (в нашем случае – 'SCOTT');
- **policy_function** – имя функции, генерирующей предикат для стратегии относительно object_name, (в нашем случае – 'AUTHORIZED_EMPS');
- **statement_types** – типы оператора (в нашем случае – 'INSERT, UPDATE, DELETE, SELECT'), к которым применима функция правил (допустимыми значениями для этого параметра являются любые комбинации операторов SELECT, INSERT, UPDATE, DELETE и INDEX, разделенных запятыми; по умолчанию применяемыми считаются все типы за исключением INDEX);
- **update_check** – данный параметр напоминает конструкцию *"with check option"* для представлений, он гарантирует невозможность вставки или изменения строки так, что после вставки или изменения нельзя будет ее увидеть (для типов INSERT и UPDATE этот параметр является необязательным и по умолчанию принимает значение FALSE; если он установлен в TRUE, стратегия также проверяется для операторов INSERT и UPDATE при проверке операций SELECT или DELETE);

Параметры процедуры стандартного пакета ADD_POLICY (продолжение):

- **ENABLE** – данный параметр указывает будет ли политика активирована непосредственно после своего добавления (по умолчанию принимает значение TRUE);
- **STATIC_POLICY** – если этот параметр равен TRUE, политика генерирует одну и ту же строку предиката для любого пользователя, пытающегося обратиться к объекту, за исключением пользователя SYS или любого пользователя с привилегией EXEMPT ACCESS POLICY (значение по умолчанию FALSE);
- **POLICY_TYPE** – тип политики (доступные типы политик: `STATIC`, `SHARED_STATIC`, `CONTEXT_SENSITIVE`, `SHARED_CONTEXT_SENSITIVE`, `DYNAMIC`); по умолчанию является пустым, что означает, POLICY_TYPE принимает значение STATIC_POLICY. Указание любого из любого доступного типа политик переопределяет значение STATIC_POLICY;
- **LONG_PREDICATE** – длина предиката. Значение по умолчанию – FALSE, означающее, что функция, задающая политику, может возвращать предикат с длиной до 4000 байт. Значение TRUE означает, что длина текстовой строки предиката может составлять до 32K байтов. Политики, существовавшие до наличия этого параметра, сохраняли максимум 32K байтов текстовой строки предиката;
- **SEC_RELEVANT_COLS** – позволяет создавать VPD на уровне столбцов. *Применяется к таблицам и представлениям, но не синонимам*. Список столбцов, защищаемых политикой безопасности, указывается перечислением. В качестве разделителя списка перечислений используется запятая или пробел. Политика применяется только тогда, когда столбец указывается в перечислении. Значение по умолчанию – все определяемые пользователем столбцы для объекта;
- **SEC_RELEVANT_COLS_OPT** – используется совместно с SEC_RELEVANT_COLS для отображения всех строк, удовлетворяющих VPD на уровне столбцов (фильтрация только запросов SELECT), но там, где чувствительные столбцы отображаются как NULL. Значение по умолчанию NULL, что говорит о необходимости осуществлять фильтрацию, в соответствии с SEC_RELEVANT_COLS.

Реализация средств детального контроля доступа

Пример:

```
BEGIN
dbms_rls.add_policy(object_schema => 'hr',
object_name => 'employee',
policy_name => 'hr_policy',
function_schema =>'hr',
policy_function => 'hrfun',
statement_types =>'select,index',
policy_type => dbms_rls.CONTEXT_SENSITIVE,
sec_relevant_cols=>'salary,birthdate,ssn');
END;
```

Реализация средств детального контроля доступа

Определив политику, можно сразу ее протестировать, выполнив запрос к таблице EMP:

```
SQL> select * from emp;

EMPNO ENAME      JOB         MGR HIREDATE         SAL       COMM DEPTNO
----- ---------- --------- ----- ----------- --------- --------- ------
 7369 SMITH      CLERK      7902 17.12.1980     800.00                20
 7521 WARD       SALESMAN   7698 22.02.1981    1250.00    500.00     30
 7654 MARTIN     SALESMAN   7698 28.09.1981    1250.00   1400.00     30
 7844 TURNER     SALESMAN   7698 08.09.1981    1500.00      0.00     30
 7876 ADAMS      CLERK      7788 23.05.1987    1100.00                20
 7900 JAMES      CLERK      7698 03.12.1981     950.00                30
 7934 MILLER     CLERK      7782 23.01.1982    1300.00                10

7 rows selected
```

Как видите, выбрано только 7 строк, а не все 14. Можно заметить, что во всех выбранных строках значение столбца SAL меньше или равно 1500, то есть соответствует функции предиката.

Аналогично, если пользователи попытаются удалить или обновить все строки таблицы, им удастся выполнить операцию только для тех строк, видимость которых обеспечивает политика *RLS*:

```
SQL> DELETE emp;

7 rows deleted
SQL> UPDATE emp SET comm = 100;

7 rows updated
```

Реализация средств детального контроля доступа

Политики не являются объектами схемы базы данных.

Другими словами, *они не принадлежат никакому пользователю*. Любой пользователь, обладающий привилегией **EXECUTE** на пакет DBMS_RLS, может создать политику. Аналогично любой пользователь с привилегией **EXECUTE** может удалить любую политику. Поэтому необходимо очень внимательно подходить к выдаче прав на работу с пакетом DBMS_RLS.

Если кто-то выдаст привилегию **EXECUTE** *на пакет для* PUBLIC, *ее надо немедленно отозвать!*

Вы можете создавать функции политики безопасности любой сложности, описывающие практически любые требования к приложению. Однако все эти функции должны следовать нескольким правилам:

- функция политики безопасности должна быть самостоятельной функцией в схеме или в составе пакета, но ни в коем случае не процедурой;
- она должна возвращать значение типа VARCHAR2, которое будет использоваться как предикат;
- функция должна иметь ровно два входных параметра, следующих в определенном порядке:
  - имя схемы, которой принадлежит таблица, для которой определена политика;
  - имя объекта (таблицы или представления), к которому применяется политика.

Реализация средств детального контроля доступа

Для просмотра политик, определенных для таблицы, можно обратиться к представлению словаря данных DBA_POLICIES, которое отображает имя политики, имя объекта, для которого она определена (и его владельца), имя функции политики (и ее владельца) и многое другое.
Если вы хотите удалить существующую политику *RLS*, то можете использовать программу DROP_POLICY из пакета DBMS_RLS:

```
SQL> BEGIN
  2  DBMS_RLS.drop_policy (object_name => 'EMP',
  3  policy_name => 'EMP_POLICY');
  4  END;
  5  /

PL/SQL procedure successfully completed.
```

Литература

1. Кайт Т. Oracle для профессионалов: Пер. с англ. / Т. Кайт. – СПб. : ООО «ДиаСофтЮП», 2003. – 672 с.
2. Нанда А. Oracle PL/SQL для администраторов баз данных. – Пер. с англ. / А. Нанда, С. Фейерштейн. – СПб : Символ-Плюс, 2008. – 496 с.
3. Пржиялковский В. Контекст сеанса в Oracle. [Электронный ресурс] / В. Пржиялковский. – Режим доступа : *http://citforum.ru/database/oracle/session_context/part2.shtml*