УТВЕРЖДЕН

ЦАУВ.17001-01 90 01-ЛУ

Система управления базами данных МСВСфера СУБД 6.0

Руководство пользователя

ЦАУВ.17001-01 90 01

Версия 1.0

2015

Инв. № подл.	Подпись и дата	Взам. инв №	Инв. № дубл.	Подпись и дата

Изм	Лист	№ докум	Подп	Дата

АННОТАЦИЯ

Настоящее руководство предназначено для пользователей системы управления базами данных МСВСфера СУБД 6.0.

В нем дано описание основных средств управления и интерфейсов доступа к базам данных, а также средств защиты баз данных, обеспечивающих их безопасность. Более подробная информация о возможностях средств управления базами данных представлена в документации, ссылки на которую приводятся по тексту настоящего руководства.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

МСВСфера СУБД 6.0 – высокопроизводительная, масштабируемая и отказоустойчивая система управления базами данных, ориентированная на использование при создании информационных систем корпоративного уровня с так называемой клиент-серверной архитектурой, в которых клиентские приложения выполняют от имени пользователей запросы к базе данных через сервер баз данных, а сервер базы данных выполняет обработку этих запросов и управление файлами, образующими базу данных.

Система управления базами данных МСВСфера СУБД 6.0 представляет собой программное обеспечение, функционирующее под управлением операционной системы с интегрированными серверными службами МСВСфера 6.3 Сервер, а также набор программных средств, предназначенных для поддержки клиентских приложений.

МСВСфера СУБД 6.0 разработана в рамках партнерского OEM-соглашения с компанией TmaxSoft на базе ее флагманского программного продукта DBMS Tibero и обладает следующими функциональными возможностями:

управление базами данных с помощью средств, поддерживающих открытые стандарты и реализующих язык структурированных запросов SQL, его процедурное расширение PSM, интерфейсы ODBC, JDBC, OLE DB, а также DB Links (для Oracle, DB2, MS SQL и Sybase);

обеспечение безопасности баз данных с помощью встроенных средств идентификации и аутентификации, управления доступом, контроля использования ресурсов, резервного копирования и восстановления, обеспечения доступности и целостности, аудита и управления средствами обеспечения безопасности;

обеспечение требуемой масштабируемости, отказоустойчивости и производительности за счет использования различных технологий буферизации данных в оперативной памяти, многопотоковой параллельной обработки транзакций, кластеризации, балансировки нагрузки и виртуализации хранения данных.

простота внедрения, наличие удобных средств разработки приложений и администрирования баз данных, в том числе загрузки, экспорта, импорта данных и мониторинга производительности.

2 СРЕДСТВА УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ

2.1 Язык запросов к базам данных

Язык структурированных запросов к базам данных SQL, реализованный в МСВСфера СУБД 6.0, поддерживает стандарты SQL-92 и SQL-99.

2.1.1 Типы данных

В МСВСфера СУБД 6.0 можно использовать следующие типы данных:

Строковый тип	CHAR, VARCHAR, VARCHAR2, NCHAR, NVARCHAR, NVARCHAR2, RAW, LONG, LONG RAW
Числовой тип	NUMBER, INTEGER, FLOAT, BINARY_FLOAT, BINARY_DOUBLE
Временной тип	DATE, TIME, TIMESTAMP, TIMESTAMP WITH TIME ZONE, TIMESTAMP WITH LOCAL TIME ZONE
Интервальный тип	INTERVAL YEAR TO MONTH, INTERVAL DAY TO SECOND
Тип больших объектов	CLOB, BLOB, XMLTYPE
Встроенный тип	ROWID
Пользовательский тип	ARRAY, NESTED TABLE

Подробные сведения о поддерживаемых в МСВСфера СУБД 6.0 типах данных можно найти в [2].

2.1.2 Операторы

В МСВСфера СУБД 6.0 могут быть использованы следующие операторы:

арифметические операторы, осуществляющие сложение, вычитание, умножение, деление и унарные операторы, обозначающие положительные и отрицательные числа («+», «−», «*», «/»);

строковый оператор «||» для конкатенации строк;

множественные операторы, реализующие операции с множествами: объединение «UNION», объединение с избытком «UNION ALL», пересечение «INTERSECTION», вычитание «MINUS»;

логические операторы, реализующие булевы операции: пересечение «AND», объединение «OR», отрицание «NOT» и значения истины «TRUE», лжи «FALSE» и неопределенности «UNKNOWN»;

операторы сравнения: равенство «=», неравенство «!=», «^=», «~=», «<>», отношение «<», «>», «<=», «>=».

Подробнее об операторах и возможностях их использования можно найти в [2].

2.1.3 Функции

В МСВСфера СУБД 6.0 имеется большое количество встроенных функций, часть из которых определена в стандарте SQL, другие добавлены в качестве дополнения. Все функции МСВСфера СУБД 6.0 могут быть разделены на три типа: однострочные, агрегирующие и аналитические.

Большинство функций имеют один или несколько параметров, однако существуют также функции, которые не имеют параметров вовсе.

Каждый параметр имеет определенный тип данных. В случае, если тип фактического параметра отличается от формального, то он будет сконвертирован согласно правилам преобразования. Если же тип не может быть сконвертирован, то возникнет ошибка, о которой будет сообщено пользователю.

Более подробная информация о функциях и их полный список представлены в [2].

2.1.4 Запросы

Запросы в МСВСфера СУБД 6.0 формируются на языке SQL и бывают двух типов: на изменение/добавление данных в базу данных и на чтение из нее. Различные виды взаимодействия с базой данных требуют разных привилегий для пользователя. Подробнее о запросах и требующихся для их исполнения привилегиях изложено в [2].

2.1.5 Представления

В МСВСфера СУБД 6.0 реализованы представления, позволяющие создавать виртуальные таблицы, содержимое которых (столбцы и строки) определяется запросом. Виртуальные таблицы могут объединять данные из одной или нескольких обычных таблиц. Представление можно использовать в следующих целях:

для упрощения и настройки восприятия информации в базе данных каждым пользователем;

в качестве механизма обеспечения безопасности, позволяющего пользователям обращаться к данным через представления, без выдачи им разрешений на непосредственный доступ к базовым таблицам;

для предоставления интерфейса обратной совместимости, моделирующего таблицу, схема которой изменилась.

Подробнее о представлениях написано в [2].

2.1.6 Подмножества языка SQL

В МСВСфера СУБД 6.0 реализованы следующие подмножества языка запросов к базам данных SQL:

язык определения данных (DDL), использующийся для описания структур баз данных и включающий операторы создания, изменения и удаления объектов базы данных;

язык манипулирования данными (DML), использующий для добавления, изменения, удаления и чтения информации из баз данных;

язык управления транзакциями и сессиями (TSML), используемый для администрирования баз данных

Подробное описание синтаксиса вышеперечисленных языков приведено в [2].

2.1.7 Зарезервированные слова

В МСВСфера СУБД 6.0 реализация языка запросов к базам данными SQL использует следующие зарезервированные слова, нецелевое использование которых недоступно пользователю: ACCESS, ADD, ALL, ALTER, AND, ANY, AS, ASC, AUDIT, BETWEEN, BY, CHAR, CHECK, CLUSTER, COLUMN, COMMENT, COMPRESS, CREATE, CURRENT, DATE, DECIMAL, DEFAULT, DELETE, DESC, DISTINCT, DROP, ELSE, EXCEPT, EXCLUSIVE, EXISTS, FILE, FLOAT, FOR, FOREIGN, FROM, GRANT, GROUP, HAVING, IDENTIFIED, IMMEDIATE, IN, INDEX,

INDEXES, INITIAL, INSERT, INTEGER, INTERSECT, INTO, INVALIDATE, IS, LEVEL, LESS, LIKE, LOCK, LONG, MAXEXTENTS, MINUS, MODE, MODIFY, NOAUDIT, NOCOMPRESS, NOT, NOWAIT, NULL, NUMBER, OF, OFFLINE, ON, ONLINE, OPTION, OR, ORDER, PRIMARY, PRIOR, PRIVILEGES, PUBLIC, RAW, RENAME, REVOKE, ROW, ROWID, ROWNUM, ROWS, SELECT, SESSION, SET, SHARE, SIZE, SMALLINT, START, SUCCESSFUL, SYNONYM, SYSDATE, TABLE, THAN, THEN, TO, TRIGGER, UID, UNION, UNIQUE, UPDATE, USER, VALIDATE, VALUES, VARCHAR, VARCHAR2, VIEW, WHENEVER, WHERE, WITH.

2.2 Расширенный язык запросов к базам данных

Язык запросов к базам данных SQL является непроцедурным языком. Несложный запрос на языке SQL состоит из простых операторов (команд) и тем самым позволяет обычному пользователю осуществлять взаимодействие с базой данных даже без глубоких знаний ее структуры и алгоритмов работы с объектами. Однако, разработка сложных приложений для баз данных на языке SQL для опытных программистов может быть осложнена отсутствием привычного и удобного для них функционала.

Для решения этой проблемы в МСВСфера СУБД 6.0 интегрировано процедурное расширение языка SQL - язык программирования tbPSM и среда его исполнения.

Язык tbPSM используется совместно с SQL и добавляет такие возможности, как контроль над потоком исполнения программы (например, логические условные операторы и циклы), переменные, пользовательские процедуры и функции.

Подробнее о возможностях tbPSM и полное описание этого языка изложено в [3].

2.3 Средства обработки текстовых данных

Одной из особенностей МСВСфера СУБД 6.0 является возможность обрабатывать большие объемы текстовой информации. Для этого используется модуль TEXT, ускоряющий выполнение запросов к базе данных, связанных с текстовой информацией. Система индексации CTXCAT позволяет производить ускоренный поиск по колонкам с такими данными, как, например, названия книг или продуктов, что актуально для различного рода поисковых систем. Подробнее о средствах обработки текстовых данных написано в [4].

2.4 Средства обработки геопространственных данных

В МСВСфера СУБД 6.0 интегрирован специальный модуль Spatial, предоставляющий функционал для работы с геопространственной информацией и поддерживающий следующие дополнительные типы данных:

Тип данных	**Описание**
`POINT`	Представление точки в пространстве.
`LINESTRING`	Последовательность двух и более точек.
`POLYGON`	Представление полигона.
`MULTIPOINT`	Представление одной и более `POINT`.
`MULTILINESTRING`	Представление одной и более `LINESTRING`.
`MULTIPOLYGON`	Представление одного и более `POLYGON`.
`GEOMETRYCOLLECTION`	Представление одного и более геометрического объекта.
`EMPTY`	Обозначение пустого геометрического объекта.

Подробнее об использовании модуля управление геопространственной информацией Spatial написано в [5].

3 СРЕДСТВА ЗАЩИТЫ БАЗ ДАННЫХ

В МСВСфера СУБД 6.0 предусмотрены средства, необходимые для защиты баз данных, реализующие идентификацию и аутентификацию, управление доступом, контроль использования ресурсов, аудит безопасности и управление безопасностью.

Доступ к базе данных возможен только для зарегистрированного в ней пользователя, успешно прошедшего идентификацию, аутентификацию и авторизацию.

Регистрация пользователя в базе данных осуществляется путем создания так называемого аккаунта (учетной записи) пользователя - набора регистрационных данных, включающих уникальное имя пользователя, начальное значение его пароля и назначаемый пользователю профиль.

Имя пользователя однозначно идентифицирует его и может быть максимальной длиной до 30-ти символов. Пароль пользователя может быть максимальной длиной до 63-х символов. Его начальное значение задается при регистрации и позднее может быть изменено самим пользователем. Назначенный пользователю профиль определяет совокупность правил и ограничений, связанных с использованием паролей, а именно:

количество неуспешных попыток предъявления значения пароля, которые пользователь может предпринять в процессе аутентификации, прежде, чем его аккаунт будет заблокирован;

период времени, в течение которого аккаунт пользователя будет заблокирован после достижения заданного числа неуспешных попыток аутентификации;

временной предел повторного использования конкретного значения пароля, по истечении которого, если значение пароля не будет изменено, срок его действия закончится;

период времени, в течение которого конкретное значение пароля не может быть использовано повторно;

количество изменений значения пароля прежде, чем это значение можно будет использовать повторно;

период времени, по истечении которого, если пользователь бездействует, его аккаунт будет заблокирован;

период времени, в течение которого пользователь будет предупреждаться о том, что срок действия текущего значения пароля закончился;

указание на функцию, с помощь которой предъявляемое значение пароля будет автоматически проверяться по реализованному данной функцией алгоритму на соответствие установленной метрике качества.

Разным пользователям может быть назначен один и тот же профиль. Назначить пользователю несколько профилей одновременно нельзя. Если при создании аккаунта пользователя профиль не указан, то ему назначается профиль по умолчанию, не накладывающий ограничений на использование паролей.

Когда пользователь с помощью клиентского приложения инициирует запрос на создание сессии доступа к базе данных, сначала производится его идентификация, т.е. поиск в словаре базы данных аккаунта с таким же именем пользователя. Если идентификация пользователя завершается неуспешно, то ему возвращается сообщение об ошибке и запрещаются любые дальнейшие действия. Если идентификация пользователя завершается успешно, то начинается процесс аутентификации, т.е. проверки подлинности пользователя с помощью предъявляемого им пароля с учетом связанных с использованием паролей правил и ограничений, определенных назначенным ему профилем.

Если аутентификация пользователя завершается неуспешно, то ему возвращается сообщение об ошибке и запрещаются любые дальнейшие действия. Если аутентификация пользователя завершается успешно, то начинается процесс авторизации, т.е. проверки наличия у пользователя привилегий, необходимых для создания сессии доступа к базе данных и выполнения запрошенных операций.

Если привилегия на создание сессии доступа к базе данных у пользователя имеется, то сессия доступа для него создается и на всем ее протяжении имя пользователя вместе со значениями атрибутов безопасности ассоциируются с действующим от имени пользователя клиентским приложением.

Привилегии могут быть системными и объектными.

Системные привилегии разрешают выполнение операций на системном уровне или над определенным типом объектов базы данных, они не назначаются для именованных объектов, например: разрешить создавать сессию доступа к базе данных, разрешить изменять информацию о пользователе, разрешить удалить табличное пространство и т.д.

Объектные привилегии разрешают выполнение конкретных операций над конкретными именованными объектами базы данных, например: разрешить вставку строк в

заданную таблицу, разрешить выборку строк из заданной таблицы, разрешить удаление строк из заданной таблицы и т. д.

Привилегии могут быть предоставлены пользователю двумя способами: либо явным образом другими пользователями, обладающими, в свою очередь, привилегиями на предоставление привилегий, которыми они обладают сами, либо с помощью создания и использования так называемых ролей - именованных наборов привилегий.

При использовании роли, привилегии сначала предоставляются роли, а затем эта роль назначается пользователю. Когда пользователю назначается роль, он получает все предоставленные этой роли привилегии. Привилегии, которые разрешают уполномоченным на это пользователям выполнять специальные административные задачи, могут предоставляться через специальные административные роли.

Кроме обеспечения конфиденциальности баз данных в МСВСфера СУБД 6.0 предусмотрено обеспечение их доступности, в том числе, путем предотвращения так называемого чрезмерного использования ресурсов, ставящего под угрозу возможность работы с базами данных. Осуществляется это путем ввода ограничений на размер предоставляемого для использования пространства памяти, а также на количество одновременных параллельных операций в рамках сессии доступа к базе данных.

Для осуществления контроля за безопасностью в МСВСфера СУБД 6.0 предусмотрены средства аудита - регистрации и анализа событий, связанных с безопасностью базы данных, возможности которых могут настраиваться гибким образом: по пользователям, по объектам базы данных, по привилегиям, по успешным или неуспешным результатам операций, для каждой операции или для каждой сессии доступа к базе данных.

В зависимости от настроек, в записях журнала аудита могут храниться: имя пользователя, идентификатор объекта, идентификатор субъекта, идентификатор сессии доступа, идентификатор и результат выполненной или заблокированной операции, дата и время события, другие параметры.

Журнал аудита хранится в специальной таблице словаря данных, что позволяет использовать различные представления для его выборочного просмотра. Если аудит включен, необходимо следить за объемом журнала аудита и периодически удалять просмотренные и ставшие ненужными записи. Для настройки, просмотра и чистки журнала аудита требуются административные привилегии.

Начальным инициатором регистрации пользователей, создания и назначения профилей и ролей, предоставления привилегий и ресурсов является так называемый системный администратор, т.е. пользователь, аккаунт которого создается при запуске сервера МСВСфера СУБД 6.0, после чего он может делегировать привилегии другим пользователям: разработчикам приложений для базы данных, конечным пользователям баз данных, администраторам базы данных, отвечающим за их использование.

Для управления режимами и параметрами безопасности в МСВСфера СУБД 6.0 предусмотрены все необходимые средства.

Так, в отношении аккаунтов, профилей и ролей пользователей предусмотрены команды создания, изменения и удаления. В отношении ролей дополнительно предусмотрены команды активации (включения) и деактивации (выключения).

В отношении привилегий предусмотрены команды предоставления и отмены (отзыва).

В отношении аудита предусмотрены команды включения и выключения.

Ограничения на использование ресурсов задаются при создании объектов базы данных.

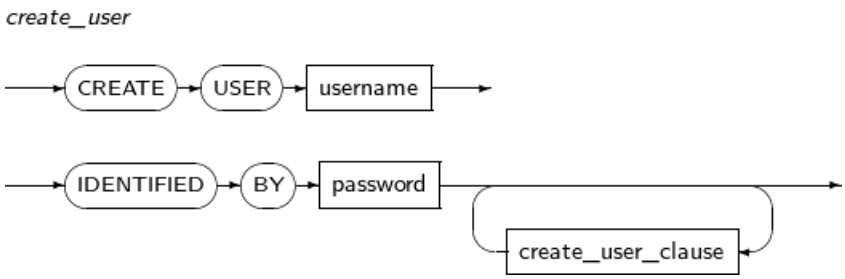
Ниже дано подробное описание вышеперечисленных команд. Соответствующая детальная информация имеется в [2].

3.1 Управление пользователями

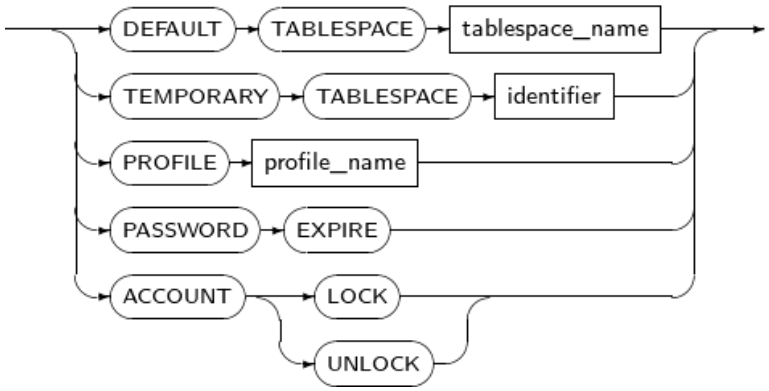
3.1.1 CREATE USER

Создание нового пользователя

Синтаксис:



create_user_clause



Привилегии:

Для создания аккаунта нового пользователя требуются права `CREATE USER`.

Пользователь, созданный командой `CREATE USER`, не имеет никаких привилегий. Кроме того, созданному пользователю необходимо выдать системную привилегию `CREATE SESSION` для доступа к базе данных.

Компоненты:

create_user:

Компонент	**Описание**
`username`	Определяет имя нового пользователя и может быть строкой длиной до 30-ти символов.
`IDENTIFIED BY password`	Определяет начальное значение пароля и может быть строкой длиной до 63-х символов При вставке специальных символов строку пароля необходимо заключить в одинарные или двойные кавычки. Для задания пароля, чувствительного к регистру, необходимо использовать одинарные кавычки.
`create_user_clause`	Определяет стандартные параметры создаваемого пользователя. Опциональный параметр.

create_user_clause:

Компонент	Описание
DEFAULT TABLESPACE	Определяет стандартное пространство таблиц для пользователя. Пространство таблиц должно существовать на момент выполнения команды. По умолчанию будет использовано системное пространство таблиц. Оно используется для внутренних настроек, поэтому рекомендуется создать отдельное пространство таблиц.
PROFILE	Задает профиль, который определяет для пользователя ограничения и правила использования паролей.
PASSWORD EXPIRE	Определяет необходимость смены значения пароля при первом входе пользователя в систему.
ACCOUNT LOCK	Создает аккаунт пользователя в заблокированном состоянии. По умолчанию аккаунт пользователя создается в разблокированном состоянии.
ACCOUNT UNLOCK	Создает аккаунт пользователя в разблокированном состоянии.

Примеры:

Следующий пример иллюстрирует создание аккаунта нового пользователя и назначение ему уже существующего пространства таблиц.

```
SQL> CREATE TABLESPACE t1 DATAFILE 't1.dbf' SIZE 10M
EXTENT MANAGEMENT LOCAL UNIFORM SIZE 256K;
Tablespace created.
SQL> CREATE USER u1 IDENTIFIED BY 'p1'
DEFAULT TABLESPACE t1;
User created.
SQL> SELECT username, default_tablespace
FROM dba_users WHERE username='U1';
USERNAME DEFAULT_TABLESPACE
------------------------------------------------------------
```

```
U1 T1
1 row selected
```

Следующий пример иллюстрирует создание пользователя с паролем, который требует смены при первом входе в систему.

```
SQL> CREATE USER u2 IDENTIFIED BY 'p2'
PASSWORD EXPIRE;
User created.
SQL> GRANT CREATE SESSION TO u2;
Granted.
SQL> CONN u2/p2
TBR-17002: password expired.
New password : ***
Retype new password : ***
Password changed
Connected.
```

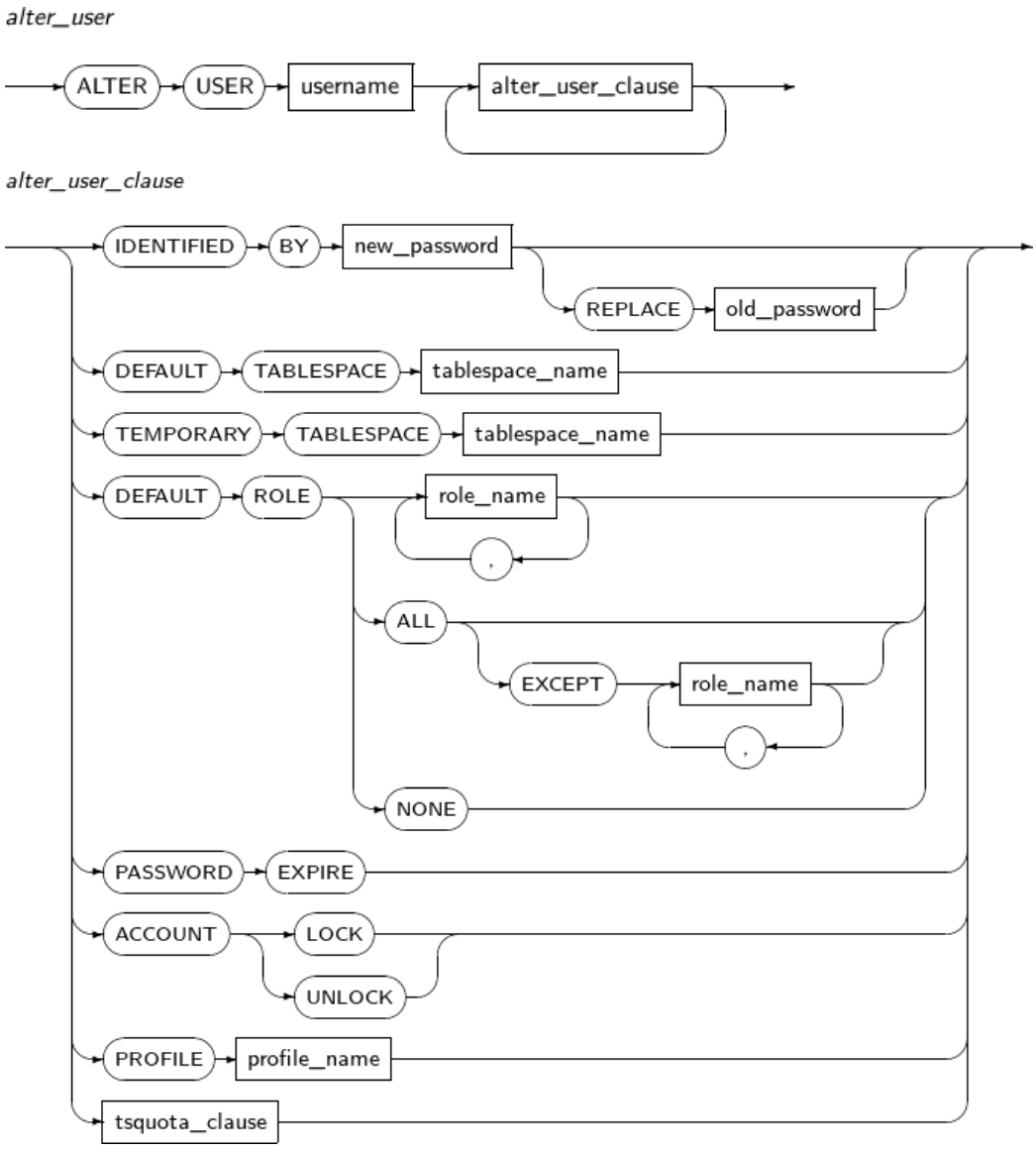
Следующий пример иллюстрирует создание пользователя с переводом его аккаунта в заблокированное состояние.

```
SQL> CREATE USER u3 IDENTIFIED BY 'p3'
ACCOUNT LOCK;
User created.
SQL> SELECT username, account_status
FROM dba_users WHERE username='U3';
USERNAME ACCOUNT_STATUS
------------------------------ ----------------
U3 LOCKED
1 row selected.
SQL> GRANT CREATE SESSION TO u3;
Granted.
SQL> CONN u3/p3
TBR-17006: account is locked.
```

3.1.2 ALTER USER

Изменение данных пользователя.

Синтаксис:



Привилегии:

Для выполнения команды требуется системная привилегия `ALTER USER`. Пользователь не нуждается в данной привилегии для изменения значения собственного пароля.

Изм	Лист	№ докум	Подп	Дата

Компоненты:

alter_user:

Компонент	Описание
username	Имя пользователя, чья информация будет изменена. Пользователь должен быть предварительно создан командой CREATE USER.
alter_user_clause	Информация о пользователе, которую необходимо изменить. В отличие от CREATE USER хотя бы одно выражение alter_user_clause должно быть задано.

alter_user_clause:

Компонент	Описание
IDENTIFIED BY	Изменение значения пользовательского пароля. Изменение пользователем пароля другого пользователя возможно только при наличии у него системной привилегии ALTER USER. При изменении пароля другого пользователя выражения REPLACE игнорируется.
new_password	Новое значение пароля, которое может быть строкой длиной до 63 символов.
REPLACE	Данное выражение используется совместно с IDENTIFIED BY в случаях, когда пользователь меняет собственный пароль. Пользователь может изменить собственный пароль без системной привилегии ALTER USER. В этом случае требуется указать REPLACE, а значение old_password должно совпадать с текущим значением пароля.
old_password	Текущее значение пароля.
DEFAULT TABLESPACE	Изменение текущего пространства таблиц. В случаях, если

	таблица создана командой CREATE TABLE и пространство таблиц не определено, то используется пространство таблиц по умолчанию.
DEFAULT ROLE	Роль по умолчанию, используемая при подсоединении пользователя к базе данных. Доступные роли должны быть выданы явным образом командой GRANT.
PASSWORD EXPIRE	Помечает текущий пароль пользователя как просроченный и требующий смены при его следующем подсоединении к базе данных.
ACCOUNT LOCK	Изменяет состояние аккаунта пользователя на заблокированное. Заблокированный пользователь не может использовать базу данных.
ACCOUNT UNLOCK	Изменяет состояние аккаунта пользователя на разблокированное.

Существует три способа для задания роли по умолчанию:

№	Способ	Описание
1	role_name	Перечисление ролей в качестве ролей по умолчанию из списка доступных ролей. Используется в случаях, когда надо задать небольшое количество ролей.
2	ALL (EXCEPT)	Данная опция используется в ситуациях, когда должно быть задано много ролей. Для исключения ролей из списка для назначения используется ключевое слово EXCEPT следующее за ALL.
3	NONE	Деактивирует все роли за исключением необходимых. Роли, выданные пользователю, могут быть динамически активированы и деактивированы командой SET ROLE.

Примеры:

Следующий пример иллюстрирует возможности команды по смене пароля.

```
SQL> CONN sys/tibero
Connected.
SQL> ALTER USER u1 IDENTIFIED BY 'p1';
User altered.
SQL> CONN u1/p1
Connected.
SQL> ALTER USER u1 IDENTIFIED BY 'p2';
TBR-7053: Invalid old password.
SQL> ALTER USER u1 IDENTIFIED BY 'p2' REPLACE 'p1';
User altered.
```

Следующий пример показывает, как изменить доступное пространство таблиц.

```
SQL> SELECT username, default_tablespace
FROM dba_users
WHERE username='U1';
USERNAME DEFAULT_TABLESPACE
------------------------------ ------------------------------
U1 SYSTEM
1 row selected.
SQL> ALTER USER u1 DEFAULT TABLESPACE t1;
Altered.
SQL> SELECT username, default_tablespace
FROM dba_users
WHERE username='U1';
USERNAME DEFAULT_TABLESPACE
------------------------------ ------------------------------
U1 T1
1 row selected.
```

Следующий пример иллюстрирует возможности команды по изменению ролей.

```
SQL> CREATE ROLE a;
Role created.
SQL> CREATE ROLE b;
Role created.
SQL> CREATE ROLE c;
Role created.
```

```
SQL> GRANT CREATE SESSION TO a;
Granted.
SQL> GRANT a TO b;
Granted.
SQL> GRANT b, c TO u1;
Granted.
SQL> SELECT grantee, granted_role, default_role
FROM dba_role_privs
WHERE grantee='U1';
GRANTEE GRANTED_ROLE DEF
------- ------------ ---
U1 B YES
U1 C YES
2 rows selected.
SQL> ALTER USER u1 DEFAULT ROLE a;
TBR-7172: cannot enable role 'a' you have not been granted
SQL> GRANT a TO u1;
granted.
SQL> ALTER USER u1 DEFAULT ROLE NONE;
User altered.
SQL> SELECT grantee, granted_role, default_role
FROM dba_role_privs
WHERE grantee='U1';
GRANTEE GRANTED_ROLE DEF
------- ------------ ---
U1 B NO
U1 C NO
U1 A NO
3 rows selected.
SQL> ALTER USER u1 DEFAULT ROLE ALL EXCEPT a, c;
User altered.
SQL> SELECT grantee, granted_role, default_role
FROM dba_role_privs
WHERE grantee='U1';
GRANTEE GRANTED_ROLE DEF
------- ------------ ---
U1 B YES
U1 C NO
```

```
U1 A NO
3 rows selected.
```

Следующий пример иллюстрирует способ указания пароля как требующего изменения.

```
SQL> ALTER USER u1 PASSWORD EXPIRE;
User altered.
SQL> CONN u1/p1
TBR-17002 : password expired.
New password : ***
Retype new password : ***
Password changed.
Connected.
```

Следующий пример показывает, как заблокировать и разблокировать аккаунт пользователя.

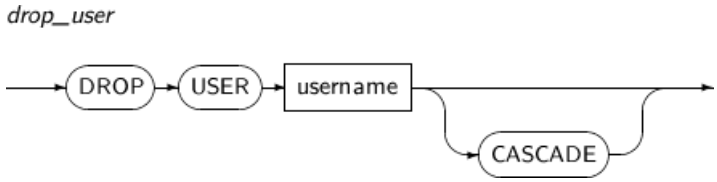
```
SQL> ALTER USER u1 ACCOUNT LOCK;
User altered.
SQL> SELECT username, account_status
FROM dba_users
WHERE username='U1';
USERNAME ACCOUNT_STATUS
------------------------------ ----------------
U1 LOCKED
1 row selected.
SQL> CONN u1/p1;
TBR-17006: account is locked.
SQL> CONN sys/tibero
Connected.
SQL> ALTER USER u1 ACCOUNT UNLOCK;
User altered.
SQL> SELECT username, account_status
FROM dba_users
WHERE username='U1';
USERNAME ACCOUNT_STATUS
------------------------------ ----------------
U1 OPEN
1 row selected.
```

```
SQL> CONN U1/p1;
Connected.
```

3.1.3 DROP USER

Удаление пользователя.

Синтаксис:



Привилегии:

Для выполнения команды требуется системная привилегия `DROP USER`.

Компоненты:

Компонент	**Описание**
`username`	Имя пользователя, который должен быть удален.
`CASCADE`	В случае, если пользователь, который должен быть удален, обладает какими-либо объектами, то данные объекты будут удалены вместе с ним. При удалении пользователя, обладающего какими-либо объектами, без ключевого слова `CASCADE`, возникнет ошибка, и удаление не произойдет.

Примеры:

Следующий пример иллюстрирует возможности команды по удалению пользователя.

```
SQL> CREATE USER u1 IDENTIFIED BY 'p1';
User created.
SQL> DROP USER u1;
User dropped.
SQL> CREATE USER u2 IDENTIFIED BY 'p2';
User created.
SQL> GRANT CREATE SESSION TO u2;
Granted.
SQL> GRANT RESOURCE TO u2;
Granted.
```

```
SQL> CONN U2/p2
Connected.
SQL> CREATE TABLE t1 (a NUMBER, b NUMBER);
Table created.
SQL> CONN sys/tibero
Connected.
SQL> DROP USER u2;
TBR-7134: cascade is required to remove this user from the system
SQL> DROP USER u2 CASCADE;
User dropped.
```

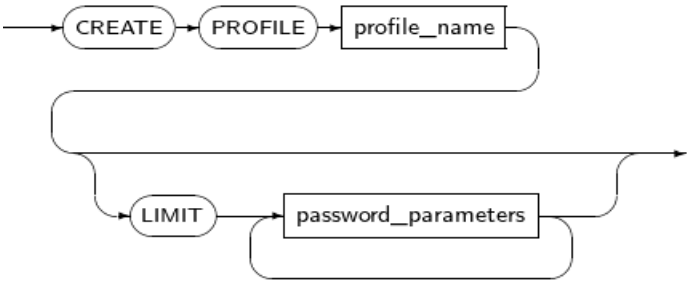
3.2 Управление профилями

3.2.1 CREATE PROFILE

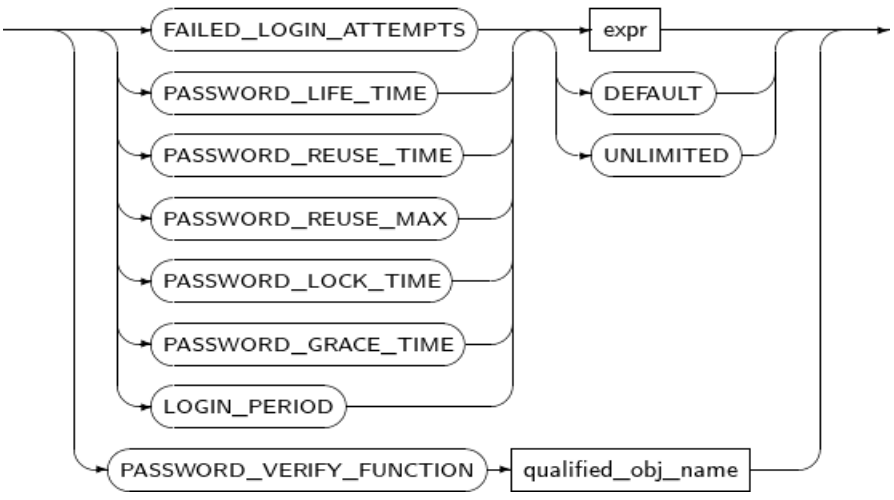
Создание нового профиля.

Синтаксис:

create_profile



password_parameters



Привилегии:

Для выполнения команды требуется системная привилегия `CREATE PROFILE`.

Компоненты:

create_user:

Компонент	Описание
`profile_name`	Имя создаваемого профиля длиной до 255 символов.
`password_parameters`	Указывает совокупность правил и ограничений по использованию паролей.

password_parameters:

Компонент	Описание
`FAILED_LOGIN_ATTEMPTS`	Задает количество возможных неуспешных попыток аутентификации с неправильным паролем.
`LOGIN_PERIOD`	Задает время блокировки после `FAILED_LOGIN_ATTEMPTS`.
`PASSWORD_LIFE_TIME`	Задает время жизни пароля. Значение может быть числовым или задано формулой. Числовое значение задает количество дней. Например, значение 30 задает срок жизни пароля в 30 дней. Значение, заданное формулой, используется для исключительных ситуаций. Например, 1/1440 задаст срок жизни пароля равный одной минуте.
`PASSWORD_REUSE_TIME`	Задает период времени, в пределах которого пароль не может быть использован повторно. Например, значение 30 означает, что значение пароля не может быть повторно использовано в течение 30 дней.
`PASSWORD_REUSE_MAX`	Задает количество изменений значения пароля, после которого он может быть использован снова. Например, если задано значение 10, то пароль может быть использован

	повторно только после его 10 изменений.
PASSWORD_LOCK_TIME	Задает время для автоматической блокировки аккаунта пользователя по бездействию. Значение может числовым или задано формулой (аналогично с PASSWORD_LIFE_TIME).
PASSWORD_GRACE_TIME	Задает время, за которое происходит предупреждение об истечении срока действия текущего значения пароля. Значение может числовым или задано формулой (аналогично с PASSWORD_LIFE_TIME). Предупреждающее сообщение будет показано при первом входе пользователя в систему после истечения срока действия пароля. Например, если PASSWORD_LIFE_TIME имеет значение 30, а PASSWORD_GRACE_TIME имеет значение 3, то срок действия пароля истечет через 30 дней, пользователю будет выводиться сообщение в течение 3 дней с момента его первого входа в систему. После истечения этого времени аккаунт пользователя будет заблокирован.
PASSWORD_VERIFY_FUNCTION	Задает PSM функцию, которая проверяет, является ли предъявляемое значение пароля удовлетворяющим заданным требованиям. Если функция не указана, то по умолчанию никаких требований к паролю не предъявляется.

Примеры:

Следующий пример иллюстрирует процесс создания нового профиля. В нем создаются правила, при которых пользователь будет заблокирован на одну минуту в случае трех попыток ввода неправильного пароля. Срок действия пароля истечет через 90 дней и 10 последних паролей не могут быть повторены при его смене. Пользователь будет получать сообщение о необходимости сменить пароль в течение 10 дней с момента истечения его срока годности.

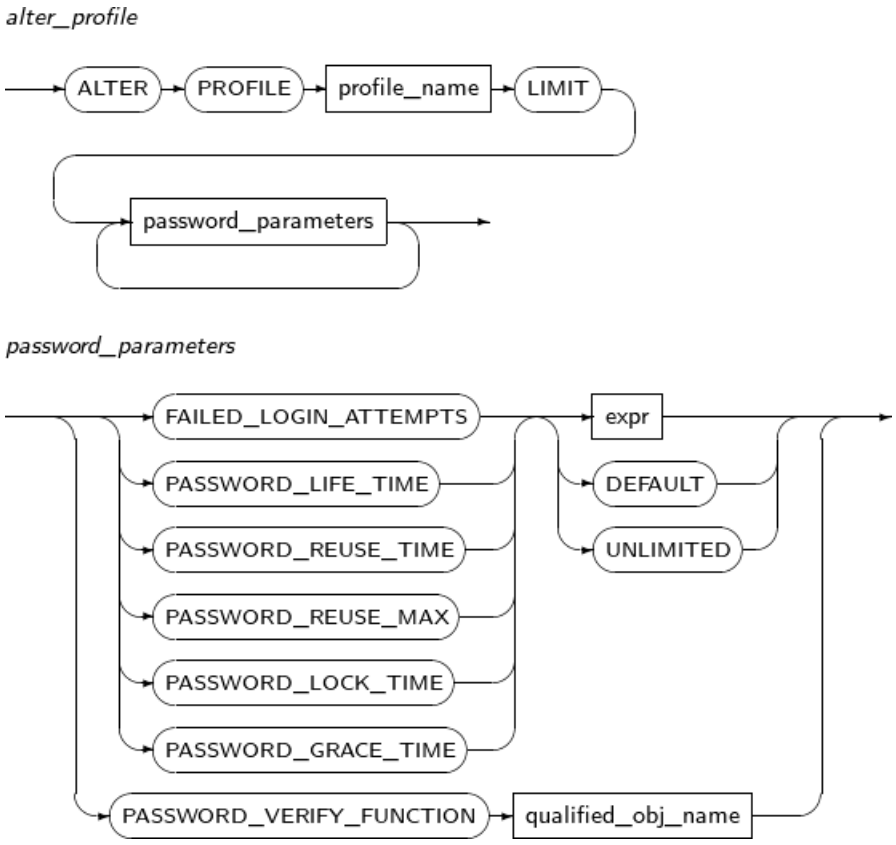
```
SQL> CREATE PROFILE prof LIMIT
failed_login_attempts 3
```

```
password_lock_time 1/1440
password_life_time 90
password_reuse_time unlimited
password_reuse_max 10
password_grace_time 10
password_verify_function verify_function;
Profile 'PROF' created.
```

3.2.2 ALTER PROFILE

Изменение свойств профиля.

Синтаксис:



Привилегии:

Для выполнения команды требуется системная привилегия `ALTER PROFILE`.

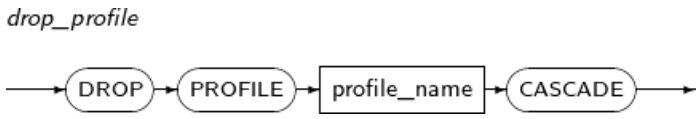
Компоненты:

Компонент	Описание
`profile_name`	Задает имя профиля, который должен быть изменен.
`password_parameters`	Задает свойства профиля, которые должны быть изменены.

3.2.3 DROP PROFILE

Удалить профиль

Синтаксис:



Привилегии:

Для выполнения команды требуется системная привилегия `DROP PROFILE`.

Компоненты:

Компонент	Описание
`username`	Задает имя профиля, который должен быть удален.
`CASCADE`	В случае, если профиль, который должен быть удален, назначен одному или нескольким пользователям, то при задании параметра `CASCADE` всем таким пользователям будет назначен профиль по умолчанию. В случае же, если пользователю назначен профилю, который должен быть удален, но параметр `CASCADE` не задан, то будет выведена ошибка и сообщение о том, что профиль не может быть удален.

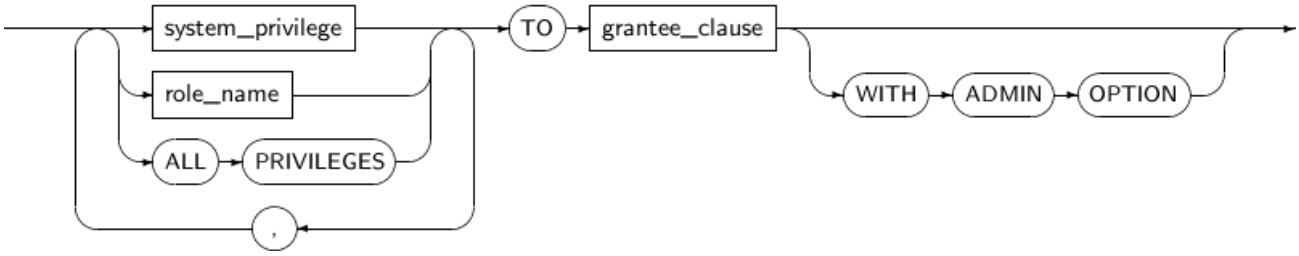
3.3 Управление привилегиями

3.3.1 GRANT

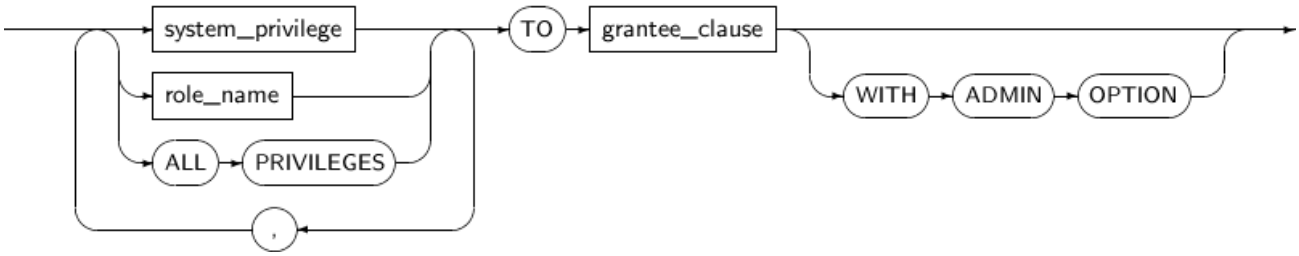
Выдает привилегии или назначает роли.

Синтаксис:

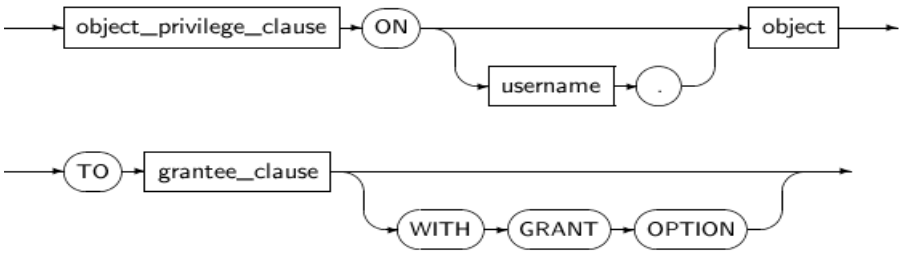
grant_sysprivs



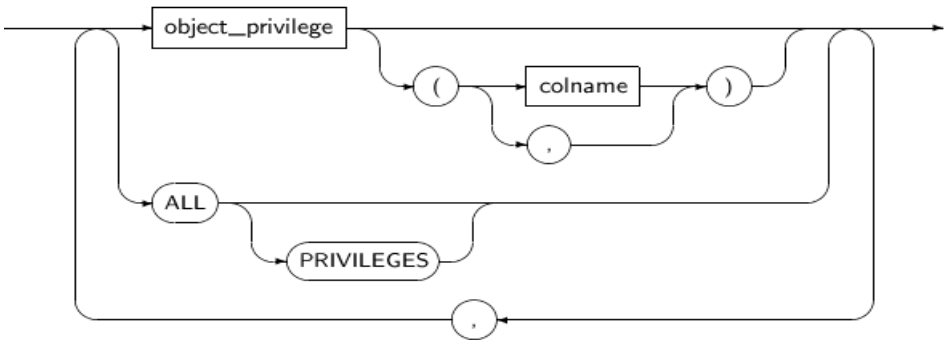
grant_sysprivs

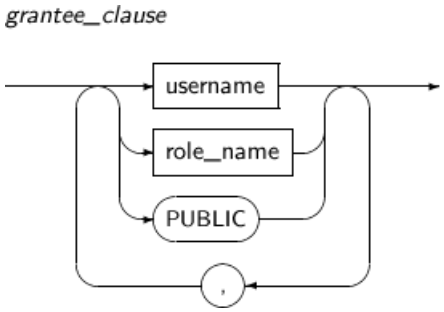


grant_objprivs



object_privilege_clause





Привилегии:

Для выдачи привилегий при помощи команды GRANT пользователь должен обладать следующими привилегиями:

Привилегия	**Описание**
System Privilege	Для выдачи системной привилегии пользователь должен иметь аналогичную системную привилегию. Однако не все системные привилегии могут быть выданы пользователю. Выдаваться могут только те привилегии, которые пользователь получил при помощи WITH ADMIN OPTION.
Role	Пользователь может назначать только те роли, которые были выданы ему с использованием WITH ADMIN OPTION или созданы им самостоятельно. Пользователь, имеющий привилегию GRANT ANY ROLE, может выдавать любую роль.
Schema Object Privilege	Пользователь может выдавать объектные привилегии в случаях, если сам является владельцем данного объекта или получил привилегии с указанием WITH GRANT OPTION. Пользователь, которому выдана системная привилегия GRANT ANY OBJECT PRIVILEGE, может выдавать любую объектную привилегию.

Компоненты:

grant:

Компонент	Описание
grant_sysprivs	Выдает системную привилегию или роль.
grant_objprivs	Выдает объектную привилегию.

grant_sysprivs:

Компонент	Описание
system_privilege	Задает системную привилегию.
role_name	Задает роль.
ALL PRIVILEGES	Выдает все доступные системные привилегии. Только пользователь, обладающий системной привилегией GRANT ANY PRIVILEGE, может использовать опцию ALL RPIVILEGES.
TO grantee_clause	Указывает субъект, которому будет выдана системная привилегия или роль. Субъектами являются обычные пользователи, PUBLIC пользователи и роли. Стоит заметить, что после выдачи роли или системной привилегии пользователю PUBLIC, любой обычный пользователь получает данную роль или системную привилегию автоматически.
WITH ADMIN OPTION	Системные привилегии делятся на два типа: использование привилегии или роли и выдача аналогичной привилегии или роли. Без опции WITH ADMIN OPTION пользователь может использовать выданную ему системную привилегию или роль, но не может выдавать ее другим пользователям. С опцией WITH ADMIN OPTION пользователь может выдавать аналогичные системные привилегии и роли другим пользователям.

object_objprivs:

Компонент	Описание
`object_privilege_clause`	Задает объектную привилегию.
`(username.)object`	Задает субъект для выдачи объектной привилегии. Если `username` опущено, то будет использоваться схема пользователя от имени которого выполняется команда. Если субъект является синонимом, то будет использован базовый объект синонима.
`TO grantee_clause`	Задает субъект, которому будет выдана привилегия или роль. Субъектами являются обычные пользователи, `PUBLIC` пользователи и роли.
`WITH GRANT OPTION`	Аналогично опции `WITH ADMIN OPTION`.

object_privilege_clause:

Компонент	Описание
`object_privilege`	Задает объектную привилегию.
`colname`	Объектные привилегии могут быть сконфигурированы более детально, нежели системные привилегии. Пользователь может выборочно выдавать привилегии схем объектов вплоть до отдельных колонок таблиц. Данный параметр задает список колонок, если это требуется.
`ALL PRIVILEGES`	Выдает все объектные привилегии. Только пользователь с системной привилегией `GRANT ANY PRIVILEGE` может использовать опцию `ALL PRIVILEGES`.

grantee_clause:

Компонент	Описание
user_name	Задает имя целевого пользователя для объектной привилегии.
role_name	Задает имя целевой роли для объектной привилегии.
PUBLIC	Задает целевого PUBLIC пользователя для объектной привилегии. Стоит отметить, если привилегия или роль выдана PUBLIC пользователю, то она автоматически становится доступна всем пользователям.

Примеры:

Следующий пример показывает, как выдать только что созданному пользователю системные привилегии:

```
SQL> CONN sys/tibero
Connected.
SQL> CREATE USER u1 IDENTIFIED BY 'a';
User created.
SQL> SELECT grantee, privilege FROM dba_sys_privs
     WHERE grantee='U1';
GRANTEE                    PRIVILEGE
------------------------------------------------------------
0 row selected.
SQL> GRANT CREATE SESSION TO u1;
Granted.
SQL> SELECT grantee, privilege FROM dba_sys_privs
     WHERE grantee='U1';
GRANTEE                    PRIVILEGE
------------------------------------------------------------
U1                         CREATE SESSION
1 row selected.
```

Следующий пример показывает, как выдать роль пользователю:

```
SQL> SELECT grantee, granted_role FROM dba_role_privs
     WHERE grantee='U1';
GRANTEE                        GRANTED_ROLE
-------------------------------------------------------------
0 row selected.
SQL> SELECT * FROM dba_roles;
ROLE                           PAS
-------------------------------------------------------------
DBA                            NO
CONNECT                        NO
RESOURCE                       NO
3 rows selected.
SQL> GRANT RESOURCE TO u1;
Granted.
SQL> SELECT grantee, granted_role FROM dba_role_privs
     WHERE grantee='U1';
GRANTEE                        GRANTED_ROLE
-------------------------------------------------------------
U1                             RESOURCE
1 row selected.
```

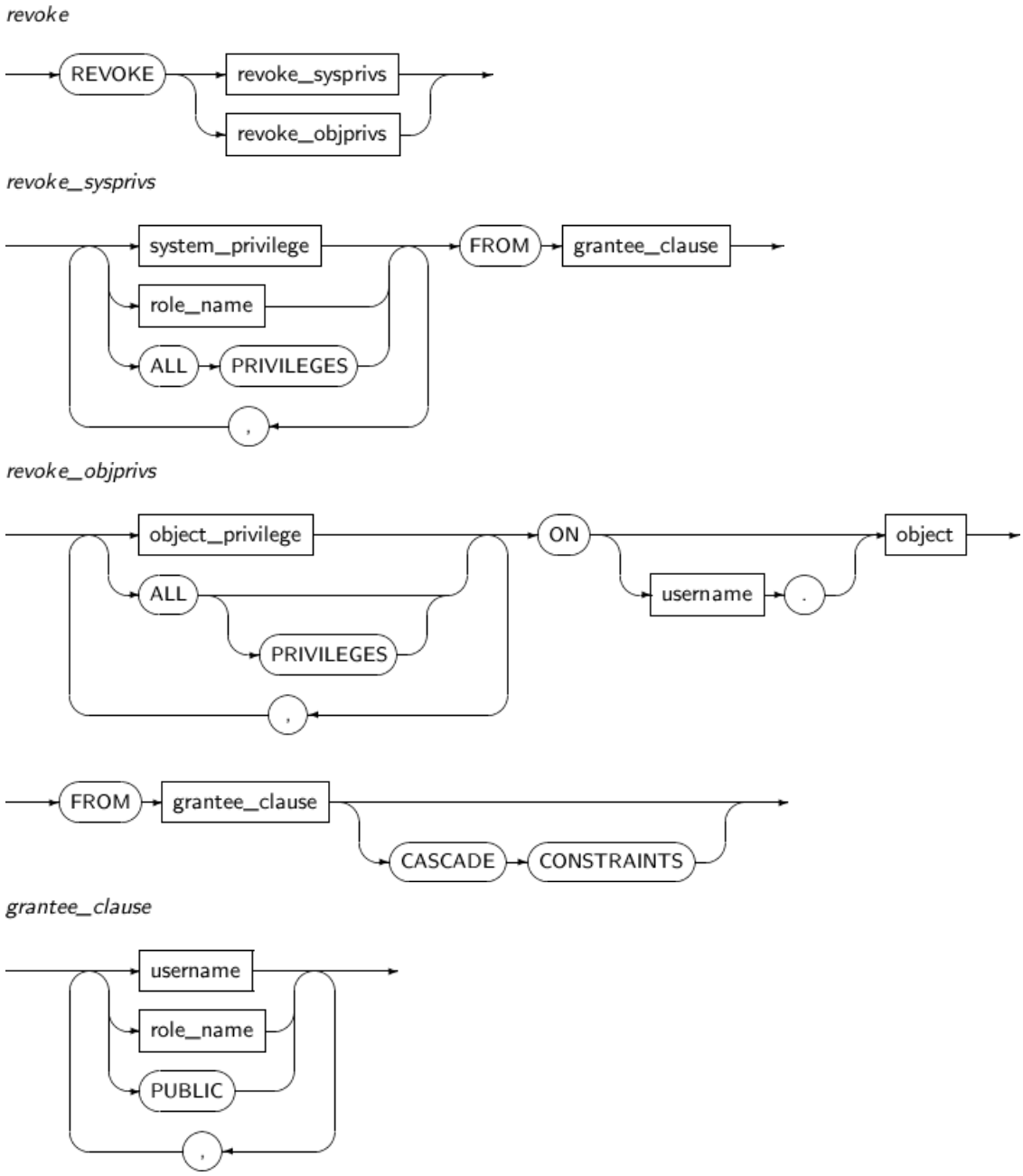
Следующий пример показывает, как вывести список системных привилегий для создания объекта схемы:

```
SQL> SELECT grantee, privilege FROM dba_sys_privs
     WHERE grantee='RESOURCE';
GRANTEE                    PRIVILEGE
-------------------------------------------------------------
RESOURCE                   CREATE TABLE
RESOURCE                   CREATE SEQUENCE
RESOURCE                   CREATE PROCEDURE
RESOURCE                   CREATE TRIGGER
4 rows selected.
```

3.3.2 REVOKE

Удаляет привилегии или снимает роли.

Синтаксис:



Привилегии:

Аналогично команде `GRANT`.

Изм	Лист	№ докум	Подп	Дата

Компоненты:

Аналогично команде GRANT.

Примеры:

Следующий пример показывает, как выдать и удалить системную привилегию:

```
SQL> CONN sys/tibero
Connected.

SQL> CREATE USER u1 IDENTIFIED BY a;
User created.

SQL> GRANT CREATE SESSION TO u1;
Granted.

SQL> SELECT grantee, privilege FROM dba_sys_privs
     WHERE grantee='U1';

GRANTEE                        PRIVILEGE
-------------------------------------------------------------
U1                             CREATE SESSION

1 row selected.

SQL> REVOKE CREATE SESSION FROM u1;
Revoked.

SQL> SELECT grantee, privilege FROM dba_sys_privs
     WHERE grantee='U1';

GRANTEE                        PRIVILEGE
-------------------------------------------------------------

0 row selected.
```

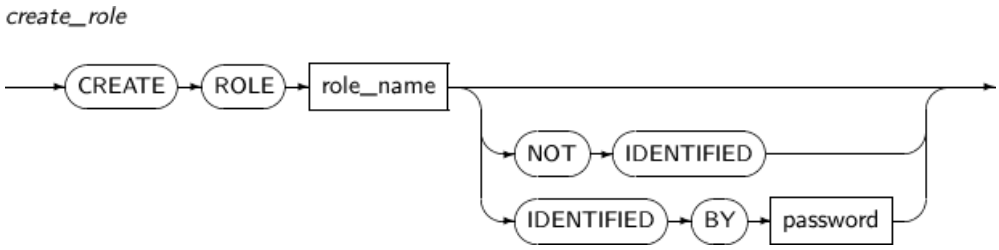
3.4 Управление ролями

3.4.1 CREATE ROLE

Создание роли.

Роль может состоять из набора привилегий или других ролей. Роль, созданная командой `CREATE ROLE`, не будет содержать никаких других привилегий или ролей за исключением явно указанных. Для добавления к существующей роли привилегий или ролей используется команда `GRANT`.

Синтаксис:



Привилегии:

Требуется системная привилегия `CREATE ROLE`.

Компоненты:

Компонент	**Описание**
`role_name`	Задаёт имя создаваемой роли, которое должно быть длиной до 30 символов. Имя роли не должно совпадать ни с каким другим именем роли или пользователя.
`NOT IDENTIFIED`	Задаёт отсутствие пароля для создаваемой роли. Это значение используется по умолчанию.
`IDENTIFIED BY`	Задаёт пароль для создаваемой роли, который используется в команде `SET ROLE`.
`password`	Задаёт пароль для роли.

Примеры:

Следующий пример показывает, как создавать новую роль без пароля.

```
SQL> CONN sys/tibero
Connected.

SQL> CREATE ROLE a;
Role created.

SQL> CREATE ROLE b NOT IDENTIFIED;
Role created.

SQL> CREATE USER u1 IDENTIFIED BY 'p1';
User created.

SQL> GRANT CREATE SESSION TO a;
Granted.

SQL> GRANT a, b TO u1;
Granted.

SQL> CONN u1/p1
Connected.

SQL> SET ROLE a;
Set.

SQL> SET ROLE b;
Set.
```

Следующий пример показывает, как создавать роль с заданным паролем.

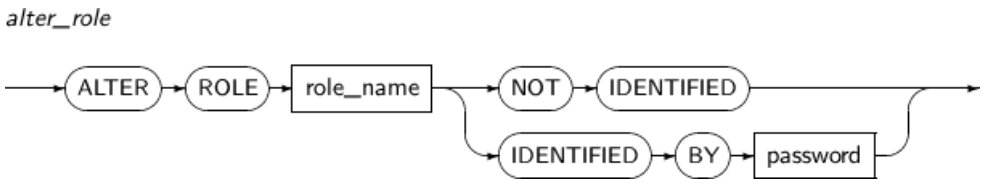
```
SQL> CONN sys/tibero
Connected.
SQL> CREATE ROLE c IDENTIFIED BY 'abc';
Role created.
SQL> GRANT CREATE SESSION TO c;
Granted.
```

```
SQL> GRANT c TO u1;
Granted.
SQL> CONN u1/p1
Connected.
SQL> SET ROLE c;
TBR-7181: need password to enable the role
SQL> SET ROLE c IDENTIFIED BY 'abc';
Set.
```

3.4.2 ALTER ROLE

Изменяет пароль роли.

Синтаксис:



Привилегии:

Изменение пароля роли с использованием команды `ALTER ROLE` доступно только для пользователя, который создал данную роль, или которому она была выдана с привилегией `WITH ADMIN OPTION`. Пользователь с системной привилегией `ALTER ANY ROLE` может изменять пароли даже не у созданных им ролей.

Компоненты:

Компонент	Описание
`role_name`	Задаёт имя изменяемой роли.
`NOT IDENTIFIED`	Задаёт отсутствие пароля для изменяемой роли.
`IDENTIFIED BY`	Задаёт наличие пароля для изменяемой роли.
`password`	Задаёт пароль для роли.

Примеры:

Следующий пример показывает, как изменять пароль у созданной ранее роли.

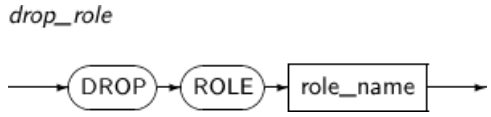
```
SQL> CONN sys/tibero
Connected
SQL> CREATE ROLE a;
Role created.
SQL> SELECT role, password_required FROM dba_roles
     WHERE role='A';
ROLE                           PAS
-------------------------------------------------------
A                              NO
1 row selected.
SQL> ALTER ROLE a IDENTIFIED BY 'xxx';
Role altered.
SQL> SELECT role, password_required FROM dba_roles
     WHERE role='A';
ROLE                       PAS
-------------------------------------------------------
A                              YES
1 row selected.
SQL> ALTER ROLE a NOT IDENTIFIED;
Role altered.
SQL> SELECT role, password_required FROM dba_roles
     WHERE role='A';
ROLE                           PAS
-------------------------------------------------------
A                              NO
1 row selected.
```

3.4.3 DROP ROLE

Удаляет роль.

Удаленная роль отзывается у всех обладающих ей пользователей и ролей. Текущие активные сессии не лишаются удаляемой роли, но при этом они больше не могут её назначать или изменять.

Синтаксис:



Привилегии:

Для выполнения команды требуется системная привилегия DROP ROLE.

Компоненты:

Компонент	**Описание**
role_name	Имя удаляемой роли. Роль должна быть предварительно создана командой CREATE ROLE.

Пример:

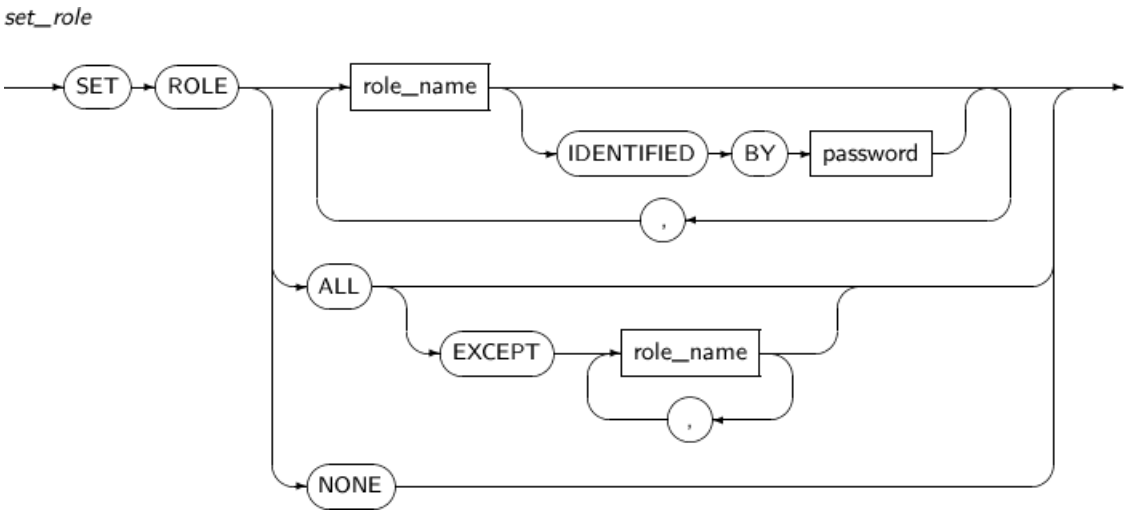
```
SQL> CONN sys/tibero
Connected.

SQL> CREATE USER u1 IDENTIFIED BY 'p1';
User created.

SQL> CREATE ROLE a;
Role created.

SQL> GRANT a TO u1;
Granted.

SQL> SELECT grantee, granted_role
     FROM dba_role_privs
     WHERE granted_role='A';

GRANTEE                        GRANTED_ROLE
---------------------------------------------------------------
```

```
SYS                            A
U1                             A

2 rows selected.

SQL> DROP ROLE a;
Role dropped.

SQL> SELECT grantee, granted_role
     FROM dba_role_privs
     WHERE granted_role='A';

GRANTEE                        GRANTED_ROLE
--------------------------------------------------------------
0 row selected.
```

3.4.4 SET ROLE

Активирует или деактивирует роль

Синтаксис:



Привилегии:

Для выполнения команды `SET ROLE` требуется, чтобы пользователь имел доступ к роли, которая будет активироваться или деактивироваться.

Компоненты:

Компонент	Описание
role_name	Имена ролей, которые должны быть активированы. Все роли, которые не будут перечислены в списке, будут деактивированы.
IDENTIFIED BY	В случае, если для роли задан пароль, то он должен быть введен.
ALL	Указывает, что все выданные роли будут активированы. В случае, если роль требует пароль, то ALL не может быть использован, пока соответствующая роль не будет перечислена в списке EXCEPT.
EXCEPT	При использовании ALL для активирования ролей список исключений может быть передан в EXCEPT. То есть будут активированы все роли пользователя за исключением тех, что перечислены в EXCEPT.

Пример:

```
SQL> CONN sys/tibero
Connected.
SQL> CREATE USER u1 IDENTIFIED BY xxx; -- Creates a user u1.
User created.
SQL> GRANT CREATE SESSION TO u1;
Granted.
SQL> CREATE ROLE a; -- Creates a role.
Role created.
SQL> CREATE ROLE b;
Role created.
SQL> CREATE ROLE c;
Role created.
SQL> CREATE ROLE d IDENTIFIED BY aaa; --Some roles use a password.
Role created.
SQL> CREATE ROLE e IDENTIFIED BY bbb;
Role created.
```

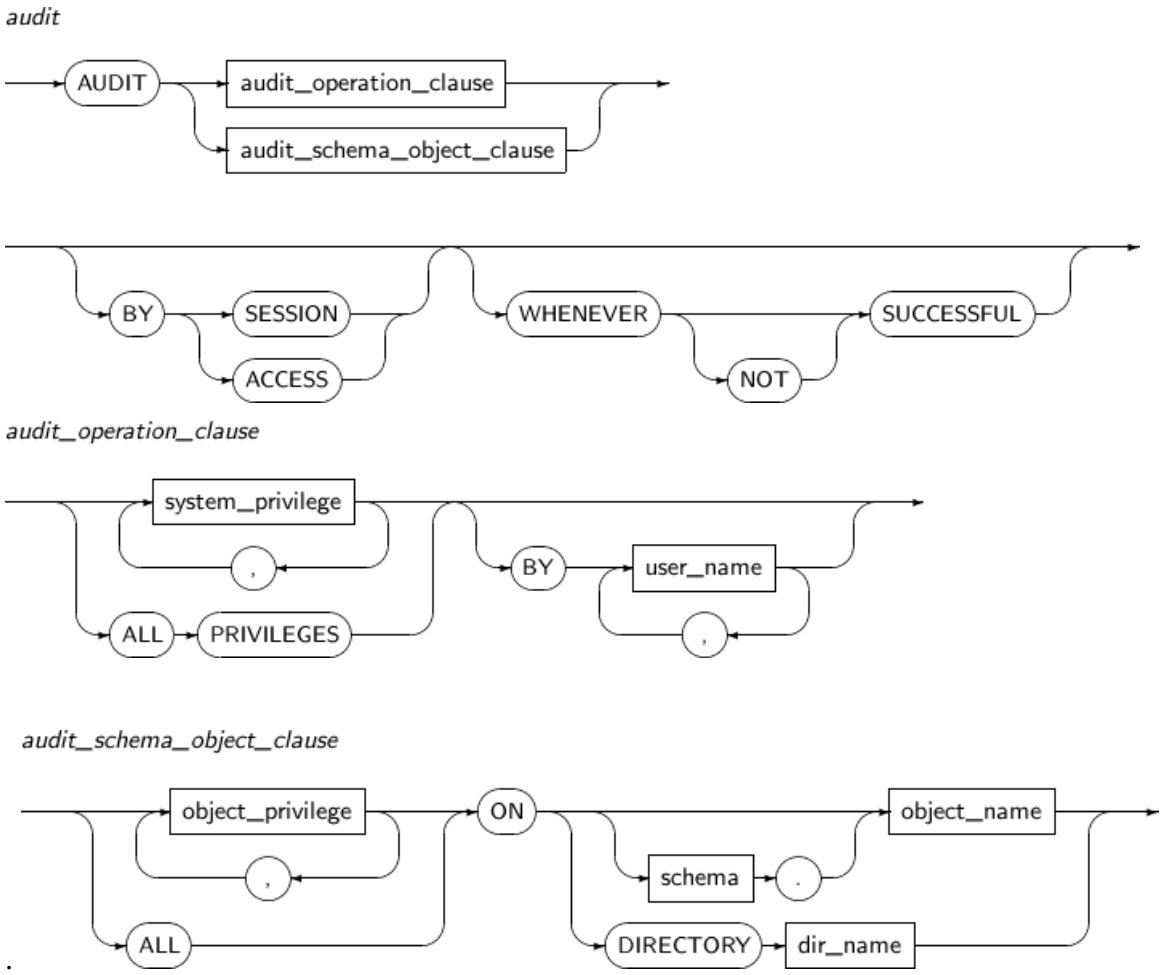
```
SQL> GRANT a, b, c, d, e TO u1; -- Grants roles to user U1.
Granted.
SQL> CONN u1/xxx
Connected.
SQL> SET ROLE a, b, c;
Set.
SQL> SELECT * FROM session_roles;
ROLE
------------------------------
A
B
C
3 rows selected.
SQL> SET ROLE a, b, c, d;
TBR-7181: need password to enable the role
SQL> SET ROLE c, d IDENTIFIED BY aaa, e IDENTIFIED BY bbb;
Set.
SQL> SELECT * FROM session_roles;
ROLE
------------------------------
C
D
E
3 rows selected
```

3.5 Управление аудитом

3.5.1 AUDIT

Включение аудита.

Синтаксис:



.

Привилегии:

Для выполнения аудита системных привилегий требуется привилегия `AUDIT SYSTEM`. При проведении аудита объекта схемы или объекта директории владельцем

Которых является другой пользователь, выполняющий данную команду

пользователь должен обладать системной привилегией `AUDIT ANY`.

Пользователь может записывать (фиксировать) в журнале аудита любую

информацию при помощи указания параметра AUDIT_TRAIL с опцией отличной от NONE в файле $TB_SID.tip. В случае если параметр установлен в NONE, аудит будет осуществляться, но информация о нем не будет фиксироваться в журнале аудита. Пользователь SYS по умолчанию находится вне контроля аудита. Для аудита пользователя SYS необходимо установить параметр AUDIT_SYS_OPERATION в значение Y в файле $TB_SID.tip. После этого все операции, совершаемые пользователем SYS, будут фиксироваться.

Компоненты:

audit:

Компонент	**Описание**
audit_operation_clause	Аудит системной привилегии.
audit_schema_object_clause	Аудит объектной привилегии на конкретном объекте.
BY SESSION	Указывает, как фиксировать ход аудита в случае, если пользователю были выданы системные привилегии для проведения аудита. Параметр BY SESSION указывает, что необходимо производить запись повторяющихся действий только один раз за сессию.
BY ACCESS	Параметр BY ACCESS записывает каждое действие, даже в случаях, когда они повторяются. Если значение параметра AUDIT_TRAIL в конфигурационном файле задано как OS, то система будет всегда себя вести, как будто задан параметр BY ACCESS, игнорируя места, где указано BY SESSION.

WHENEVER SUCCESSFUL	Указывает, как фиксировать ход аудита в зависимости от успешности выполнения команды в случае, если пользователю выданы системные привилегии для аудита. При указании параметра WHENEVER SUCCESSFUL будут записываться только успешно выполненные действия. В случае, если этот параметр не указан, то будут записываться все действия вне зависимости от успешности их выполнения.
WHENEVER NOT SUCCESSFUL	Параметр WHENEVER NOT SUCCESSFUL указывает, что необходимо фиксировать информацию только о выполнении команд, которые не закончились успешно.

audit_operation_clause:

Компонент	**Описание**
system_privilege	Указывает системную привилегию, которая должна подвергаться аудиту. Подробнее описано в команде GRANT.
ALL PRIVILEGES	Указывает необходимость осуществлять аудит всех системных привилегий.
BY user_name	Указывает пользователя, который должен подвергаться аудиту. Если параметр не указан, то аудиту будут подвергаться все пользователи.

audit_schema_object_clause:

Компонент	Описание
object_privilege	Указывает объект схемы, который должен подвергаться аудиту.
ALL	Проводит аудит всех объектных привилегий объекта схемы, доступных для объекта.
ON	Указывает объект, чья привилегия объекта схемы будет подвергаться аудиту.
schema	Указывает имя схемы, которой принадлежит объект. По умолчанию будет использована схема текущего пользователя.
object_name	Указывает имя объекта вместо имени директории.
DIRECTORY dir_name	Указывает имя директории.

Примеры:

Следующий пример указывает, как генерировать аудит для сессии:

```
SQL> AUDIT delete ON t BY SESSION WHENEVER SUCCESSFUL;
Audited.
```

Следующий пример показывает, как осуществлять аудит за конкретным пользователем:

```
SQL> AUDIT create table BY tibero;
Audited.
```

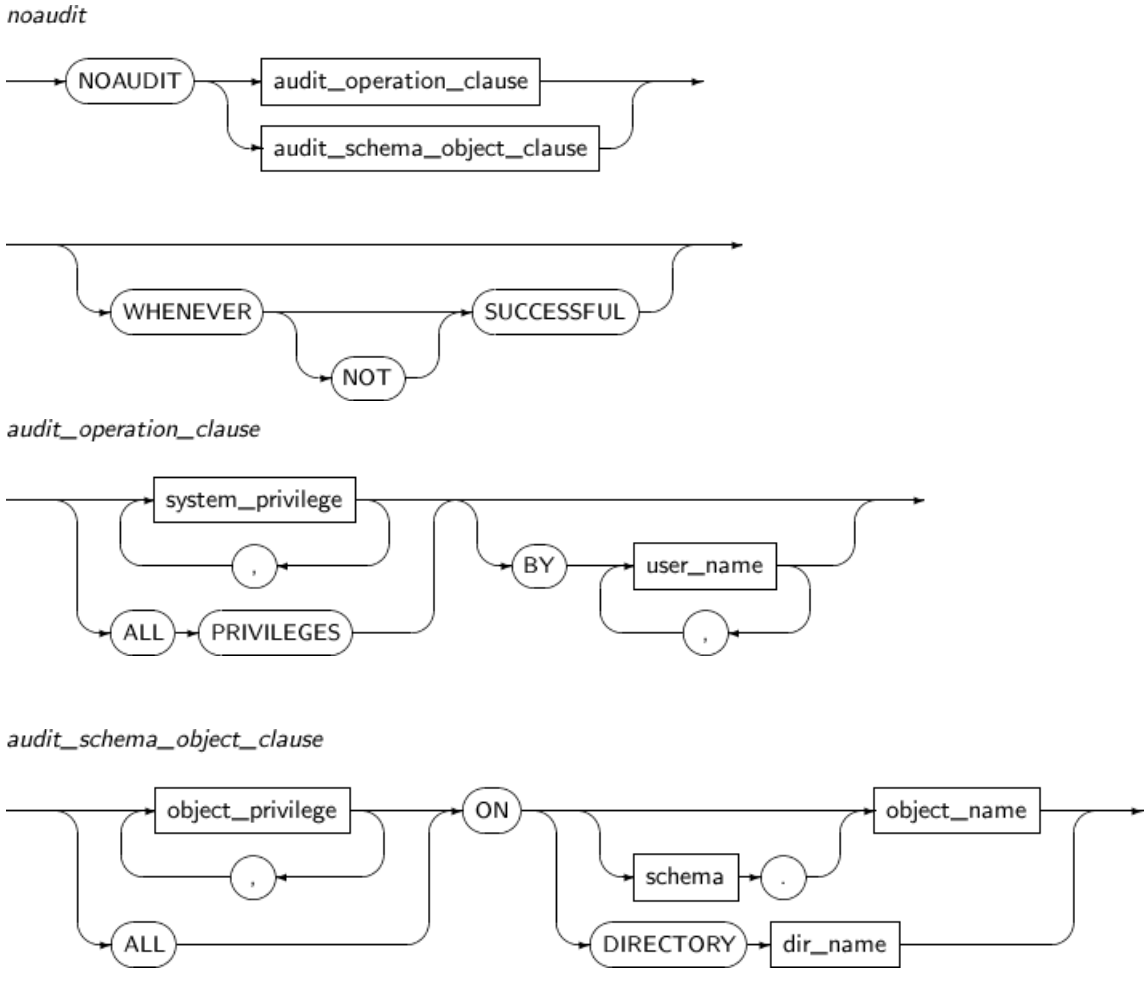
Следующий пример показывает, как осуществлять аудит над привилегией объекта схемы для конкретного объекта:

```
SQL> AUDIT insert ON t;
Audited.
```

3.5.2 NO AUDIT

Отмена аудита.

Синтаксис:



Привилегии:

Требуется системная привилегия `AUDIT SYSTEM`.

Для осуществления отмены (остановки) аудита объектов, не принадлежащих текущему пользователю, необходима привилегия `AUDIT ANY`.

Компоненты:

Аналогично команде `AUDIT`.

Примеры:

Следующий пример указывает, как отменить аудит для сессии:

```
SQL> NOAUDIT delete ON t BY SESSION WHENEVER SUCCESSFUL;
Noaudited.
```

Следующий пример показывает, как отменить аудит конкретного пользователя:

```
SQL> NOAUDIT create table BY tibero;
Noaudited.
```

Следующий пример показывает, как отменить аудит привилегии объекта схемы для конкретного объекта:

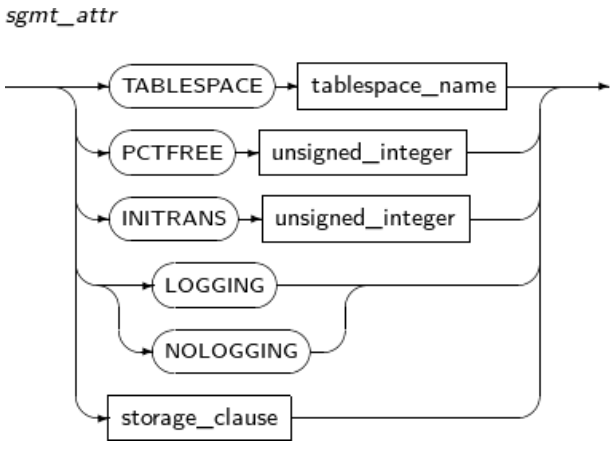
```
SQL> NOAUDIT insert ON t;
Noaudited.
```

3.6 Управление ресурсами

3.6.1 Управление характеристиками хранилища

Управление характеристиками объема хранилища осуществляется через дополнительные параметры команд `CREATE INDEX` и `CREATE TABLE`. Подробнее об их использовании изложено в [2].

Синтаксис:

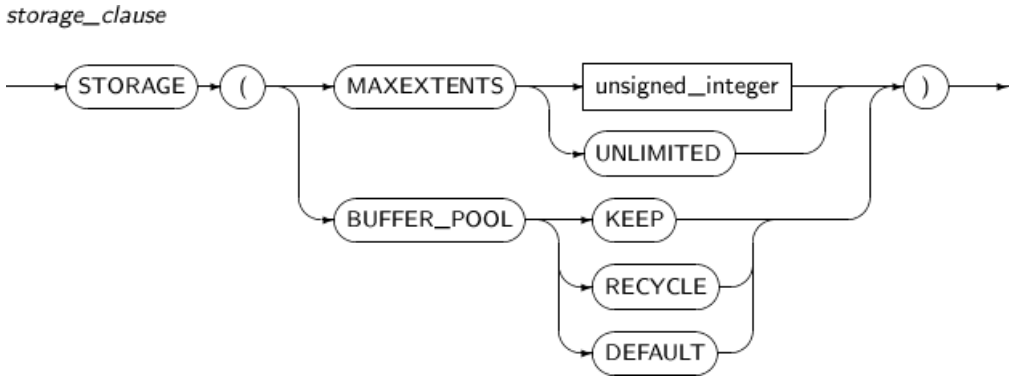


Компоненты:

Компонент	Описание
`TABLESPACE tablespace_name`	Имя пространства таблиц, где должны быть сохранены табличные данные. По умолчанию будет использовано базовое для пользователя пространство таблиц. Для временных таблиц будет использовано временное пространство таблиц, установленное по умолчанию.
`PCTFREE unsigned_integer`	Количество дополнительных пространств, использование которых возможно при увеличении объема данных. Указывается число в интервале от 1 до 99. По умолчанию используется значение 10.
`INITRANS unsigned_integer`	Процент от пространства таблиц, используемый для транзакций в каждом дисковом блоке. Транзакции будут автоматически расширяться на предоставленные ресурсы, пока доступно свободное место, поэтому не требуется указывать большие значения для данного параметра. Минимальное значение 1, максимальное значение варьируется в зависимости от размера блока на диске. По умолчанию используется значение 2.
`LOGGING / NOLOGGING`	В случаях использования `Direct-Path Loading` логи отмен действий не записываются. Однако в режиме `Archive Mode` логи записываются всегда. По умолчанию значение `LOGGING`.
`storage_clause`	Указание детальных параметров сегментов (подробнее далее по тексту).

`Storage_clause` задает детальные свойства сегмента.

Синтаксис:



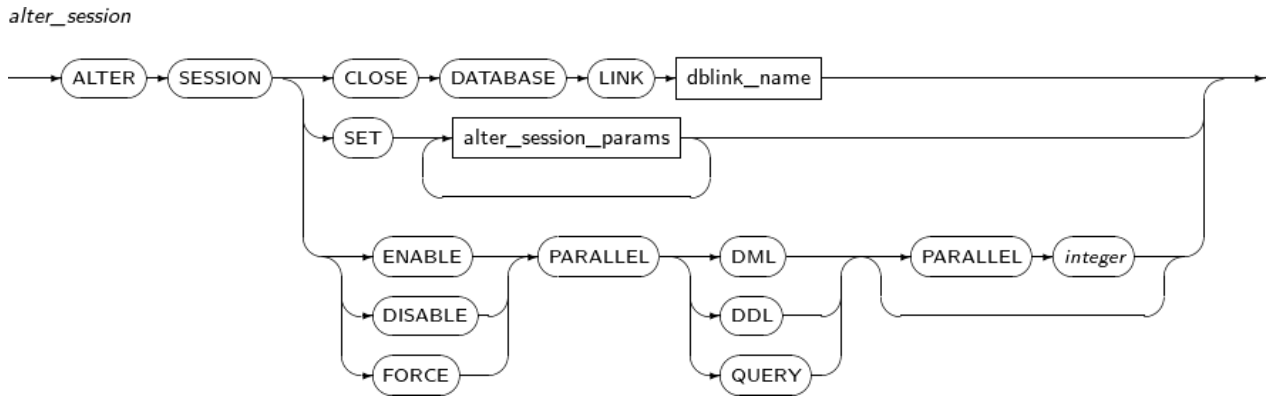
Компоненты:

Компонент	Описание
`MAXEXTENTS`	Число расширений на которое может быть увеличен сегмент.
`unsigned_integer`	Задает число возможных расширений сегмента.
`UNLIMITED`	Задает отсутствие ограничений, является значением по умолчанию.
`BUFFER_POOL`	Задает буфер в который будет помещен блок данных сегмента.
`KEEP`	Хранит сегментный блок в памяти располагая его в буферной зоне `KEEP`, что позволяет сократить число операций чтения/записи.
`RECYCLE`	Назначает сегментный блок на повторное использование в буферной зоне `RECYCLE`, что позволяет сократить объем хранимых буферных кэшей в зоне `DEFAULT`.
`DEFAULT`	Является значением по умолчанию, если не выбраны другие варианты. Задает использование буферной зоны `DEFAULT`.

3.6.2 Управление ресурсами сессии

Управление ресурсами сессии возможно при помощи дополнительных параметров команды `ALTER SESSION`.

Синтаксис:



В данном блоке указана возможность создания параллельных запросов DML и DDL, и указания их количества через ключевое слово `PARALLEL`, что влияет на производительность системы в целом. Подробнее об этом изложено в [2].

4 ИНТЕРФЕЙСЫ ДОСТУПА К БАЗАМ ДАННЫХ

4.1 Интерфейс уровня вызовов

МСВСфера СУБД 6.0 предоставляет средство tbCLI (Call Level Interface), которое работает в качестве SQL интерфейса между пользовательскими приложениями и сервером баз данных. Разработчики приложений могут использовать tbCLI при создании программ на языках C или C++ для доступа к базе данных.

Средство tbCLI разработано на основе ODBC (Open Database Connectivity) и X/Open Call Level Interface Standard. Помимо этого, tbCLI удовлетворяет всем требованиям Level 2 из ODBC и большинству требований из Level 2 из ODBC 3.0.

Средство tbCLI имеет обладает следующими характеристиками:

предварительная компиляция не является обязательной для создания исполняемого файла;

использование tbCLI для управления модулями позволяет достичь высокой производительности приложения;

нет необходимости в привязке к пакету приложения;

возможно использования статистики базы данных;

облегченное создание многопоточных приложений.

Подробнее о tbCLI написано в [6].

4.2 Интерфейс для приложений на языке C

МСВСфера СУБД 6.0 предоставляет средство tbESQL в качестве встраиваемой поддержки языка запросов SQL. В большинстве случаев языки программирования используются для обработки большого количества сложных и глубоко детализированных задач. Возможность использовать синтаксис SQL позволяет снизить сложность задач, относящихся к работе с базами данных. Интерфейс ESQL позволяет объединять операторы языка программирования с функциями языка запросов SQL.

МСВСфера СУБД 6.0 предоставляет средство tbESQL/C для написания программ на языке программирования C. Подробное описание этого интерфейса изложено в [7].

Программы, написанные с использованием технологии tbESQL/C, содержат одновременно программный код на языке C и логические выражения на tbESQL/C (например, SQL запросы, относящиеся непосредственно к работе с базами данных). Выражения tbESQL/C похожи по синтаксису на SQL, но отличаются следующим:

они всегда начинаются с ключевых слов `EXEC SQL` и заканчиваются точкой с запятой;

они содержат переменные входа и выхода для передачи значений;

могут содержать внутренние выражения, например, внутри `SELECT` может быть помещено выражение с `INTO`.

Ниже приведен пример выражения на tbESQL/C, встроенного в код программы на языке программирования C:

```
EXEC SQL UPDATE EMP SET SALARY = SALARY * 1.05
WHERE EMPNO = 5;
```

В отличие от обычных SQL выражений, этот пример иллюстрирует использование специального ключевого слова `EXEC SQL`, обозначающего начало выражения tbESQL/C.

4.3 Интерфейс для приложений на языке COBOL

МСВСфера СУБД 6.0 предоставляет также интерфейс tbESQL/COBOL для написания программ на языке программирования COBOL. Подробное описание интерфейса изложено в [8].

Программы, написанные с использованием технологии tbESQL/COBOL, содержат одновременно программный код на языке COBOL и логические выражения на tbESQL/COBOL (например, SQL запросы, относящиеся непосредственно к работе с базами данных). Выражения tbESQL/COBOL похожи по синтаксису на SQL, но отличаются следующим:

они всегда начинаются с ключевых слов `EXEC SQL` и заканчиваются точкой с запятой;

они содержат переменные входа и выхода для передачи значений;

они могут содержать внутренние выражения, например, внутри `SELECT` может быть помещено выражение с `INTO`.

Ниже приведен пример выражения на tbESQL/COBOL, встроенного в код программы на языке программирования COBOL:

```
EXEC SQL UPDATE EMP SET SALARY = SALARY * 1.05
WHERE EMPNO = 5
END-EXEC.
```

В отличие от обычных SQL выражений, этот пример иллюстрирует использование специального ключевого слова `EXEC SQL`, обозначающего начало выражения tbESQL/C, и ключевого слова `ENX-EXEC`, обозначающего его окончание.

4.4 Интерфейс для приложений на языке Java

МСВСфера СУБД 6.0 предоставляет также интерфейс JDBC для разработки программ на языке программирования Java, в коде которых требуется выполнять выражения на языке запросов SQL для работы с базами данных.

Преимущества использования интерфейса JDBC:

выражения SQL могут быть использованы для запросов к любым реляционным базам данных (нет необходимости в написании отдельных программ для работы с базами данных от разных производителей);

приложения, написанные на языке программирования Java, не нуждаются в переписывании под каждую конкретную платформу, где планируется их запуск;

могут быть использованы расширенные функции Java. Например, JDBC позволяет пользователям создавать апплеты с информацией, получаемой из удаленной базы данных, или подключаемые к одному или нескольким локальным хранилищам информации. JDBC также позволяет пользователям получать доступ к информации, хранимой в разных хранилищах, что снижает время на разработку новых приложений.

МСВСфера СУБД 6.0 следует стандарту JDBC и предлагает его собственную реализацию под названием tbJDBC. Реализация имеет следующие особенности:

согласованность классов и методов интерфейсов;

поддержка стандарта SQL-99 для выражений SQL;

возможность разработки программ вне зависимости от типа DBMS;

tbJDBC наследует пакеты `java.sql.*` и `javax.sql.*`.

Предоставляемая МСВСфера СУБД 6.0 версия JDBC базируется на следующих стандартах Java: JDBC 3.0 (J2SE 1.4), JDBC 4.0 (Java SE 6).

Подробное описание интерфейса JDBC изложено в [9].

СПИСОК ДОКУМЕНТОВ

[1] – МСВСфера СУБД 6.0. Руководство администратора

[2] – Tibero SQL Reference Guide

[3] – Tibero tbPSM Reference Guide

[4] – Tibero TEXT Reference Guide

[5] – Tibero Spatial Reference Guide

[6] – Tibero tbCLI Guide

[7] – Tibero tbESQLC Guide

[8] – Tibero tbESQLCOBOL Guide

[9] – Tibero JDBC Developers Guide

[10] – Интернет-ресурс www.technet.tmaxsoft.com

Лист регистрации изменений									
Номера листов (страниц)					Всего листов (страниц) в докум.	№ документа	Входящий № сопроводи-тельного докум. и дата	Подп.	Дата
Изм	измененных	замененных	новых	аннулиро-ванных					