Основы СУБД ORACLE

Лекция №3

Язык описания данных ORACLE. Типы данных ORACLE. Таблицы. Представления.

Язык описания данных

Data Definition Language (DDL) (язык описания данных) – это семейство компьютерных языков, используемых в компьютерных программах для описания структуры баз данных. В базах данных DDL является подмножеством SQL, используемым для определения и модификации различных структур данных.

К данной группе относятся команды, предназначенные для создания, изменения и удаления различных объектов базы данных. Команды CREATE (создание), ALTER (модификация) и DROP (удаление) работают с большинством типов объектов баз данных (таблиц, представлений, процедур, триггеров, табличных областей, пользователей и др.). Также к этой группе относятся операторы RENAME и TRUNCATE.

Создание таблиц с помощью оператора CREATE.

Базы данных предназначены для хранения информации, а она содержится в таблицах. Таблицы, создаваемые в ORACLE обычно:

- хранят данные различных типов, например, текст, числа и даты;
- ограничивают длину вводимых данных;
- запрещают ввод записей, в которых не заполнены определенные столбцы;
- гарантируют, что значения, введенные в определенные столбцы, находятся в допустимом диапазоне;
- имеют логическую связь с другими таблицами.

Правила именования таблиц

1. Максимальная длина имени таблицы или столбца равна 30 символам.
2. Имена таблиц и столбцов могут содержать буквы, цифры и символ подчеркивания (_). (Есть еще пара специальных символов, которые можно использовать в случае острой необходимости, но в обычной работе это не принесет ничего, кроме проблем, поэтому лучше ограничиться буквами, цифрами и символом подчеркивания.)
3. Имена таблиц и столбцов должны начинаться с буквенного символа.
4. Имя может содержать цифры или символы подчеркивания, но в любом случае должно начинаться с буквы.
5. Символы верхнего и нижнего регистров в именах таблиц и столбцов считаются одинаковыми.

Правила именования таблиц

6. Имя таблицы или столбца не должно содержать пробелы.
7. В ORACLE таблицы присваиваются пользователям; по умолчанию они присваиваются тому пользователю, который их создал. Каждая из таблиц должна иметь имя, отличное от имен других таблиц этого пользователя. Иными словами, у пользователя не может быть двух таблиц с одним и тем же именем. (Однако разные пользователи могут без проблем создавать таблицы с одинаковыми именами.) Все столбцы в пределах таблицы должны иметь уникальные имена.
8. Некоторые слова представляют собой команды и параметры ORACLE, а следовательно, не могут использоваться в качестве имен таблиц или столбцов.

Создание таблиц

```
CREATE TABLE [schema.] table
( column datatype [DEFAULT expr]
  [column_constraint]
  [, { column datatype [DEFAULT expr]
  [column_constraint] ...
 | CONSTRAINT constraint }]...)
[TABLESPACE tab_space];
```

Описание column_constraint:

```
[NOT NULL]
[PRIMARY KEY]
[FOREIGN KEY (columns) REFERENCES table
  (columns) [ON {UPDATE | DELETE} { CASCADE |
  SET DEFAULT | SET NULL |NO ACTION }]]
[UNIQUE]
[CHECK (condition)]
```

Создание таблиц

- Эта версия оператора CREATE TABLE включает средства определения ограничений ссылочной целостности и других ограничений. В результате выполнения этого оператора будет создана таблица, имя которой определяется параметром table, состоящая из одного или нескольких столбцов column типа datatype.
- Схема schema это набор таблиц и других объектов, которым владеет один пользователь, чье имя совпадает с именем схемы. Если при выполнении данного запроса не будет указана схема, то таблица будет создана в схеме пользователя, выполняющего запрос. В дополнение к этому пользователь выполняющий запрос CREATE TABLE должен иметь соответствующие привилегии для создания таблиц в чужих схемах.

Создание таблиц

- Таблица создается в табличном пространстве tab_space, но если в запросе опущено упоминание о табличном пространстве, то по умолчанию таблица создается табличном пространстве владельца таблицы.

```
CREATE TABLE EMP_HOURLY (
EMPNO NUMBER (4) NOT NULL,
ENAME VARCHAR2 (10),
JOB VARCHAR2 (9),
MGR NUMBER (4),
HIREDATE DATE,
HOURRATE NUMBER (5,2) NOT NULL DEFAULT 6.50,
DEPTNO NUMBER (2),
CONSTRAINT PK_EMP PRIMARY KEY ( EMPNO ) );
```

- Для просмотра структуры таблицы очень удобно использовать команду DESCRIBE.

Изменения таблиц с помощью оператора ALTER TABLE

Команда ALTER TABLE позволяет пользователю производить изменение объектов в базе данных. Возможности команды ALTER TABLE можно разделит три категории.

- Добавление изменение удаление столбца.
- Добавление и удаление ограничений.
- Включение и выключение ограничений.

Для изменения таблицы можно упростить синтаксис до одного из трех выражений:

- `ALTER TABLE tablename ADD (column1 datatype1 [DEFAULT expression] [, ...]);`
- `ALTER TABLE tablename MODIFY (column1 datatype1 [DEFAULT expression] [,...]);`
- `ALTER TABLE tablename DROP COLUMN column1;`

Пример

Например: политика, которая применяется в компании Скотта, имеет новое значение часовой оплаты в 7,25$. Таблица EMP_HOURLY должна быть изменена таким образом, чтобы отражать данную политику. Мы можем использовать вторую форму запроса ALTER TABLE, рассмотренную выше, чтобы выполнить поставленную задачу. Также выясняется, что один из менеджеров управляет всеми сотрудниками с почасовой оплатой, следовательно, нам не нужен столбец MGR в таблице EMP_HOURLY. Мы можем использовать третью форму для выполнения дополнительной задачи:

- `ALTER TABLE EMP_HOURLY MODIFY (HOURRATE NUMBER(5,2) DEFAULT 7.25);`
- `ALTER TABLE EMP_HOURLY DROP COLUMN MGR;`

Ограничения

Для добавления или удаления ограничений необходимо знать имя этого ограничения. Синтаксис команды ALTER TABLE выглядит следующим образом:

- `ALTER TABLE table_name ADD CONSTRAINT constraint_name PRIMARY KEY (column [, column1, ..]);`
- `ALTER TABLE table_name ADD CONSTRAINT constraint_name UNIQUE (column [, column1, ..]);`
- `ALTER TABLE table_name ADD CONSTRAINT constraint_name FOREIGN KEY (column [, column1, ..]) REFERENCES new_table_name (new_column [, new_column1, …]);`

Ограничения

Можно также установить ограничение CHECK.

- `ALTER TABLE table_name ADD CONSTRAINT constraint_name CHECK (column condition);`

Где condition – условие, на которое проверятся значение столбца.

Чтобы удалить ограничение используется выражение DROP CONSTRAINT:

- `ALTER TABLE table_name DROP CONSTRAINT constraint_name;`

Ограничение на столбец, возможно отключить при помощи выражения DISABLE CONSTRAINT:

- `ALTER TABLE table_name DISABLE CONSTRAINT constraint_name;`

Удаление таблицы

- Когда таблица больше не нужна, ее можно удалить. Удаляются и сама таблица и все записи, хранящиеся в ней, а пространство, выделенное для таблицы, становится доступным для других объектов базы данных. Синтаксис DROP TABLE очень прост:
- DROP TABLE tablename;

Как и большинство DDL операторов, чтобы таблица была удалена, пользователь, выполняющий данную операцию, должен иметь соответствующие привилегии.

Операторы RENAME и TRUNCATE.

Оператор RENAME очень прост. Имя таблицы должно быть изменено на новое имя при этом связи с данной таблицей других объектов, таких как индексы, автоматически корректируются. Синтаксис выглядит следующим образом:

```
RENAME old_tablename TO new_tablename;
```

- С точки зрения пользователей оператор TRUNCATE удаляет записи из таблицы. Основная разница состоит в том, что при использовании DELETE можно удалять строки выборочно. Также оператор TRUNCATE работает намного быстрее. Оператор TRUNCATE очищает все пространство удаленных строк. Пространство записей, удаленных оператором DELETE, останется распределенным за таблицей, и в будущем может быть повторно использовано операторами INSERT.

```
TRUNCATE TABLE tablename;
```

Типы данных в ORACLE делятся на:

- простые
- комплексные
- объектные

Простые типы

- VARCHAR2(size [BYTE | CHAR])
- NVARCHAR2(size)
- NUMBER [(p [, s])]
- FLOAT [(p)]
- DATE
- BINARY_FLOAT
- BINARY_DOUBLE
- TIMESTAMP [(fractional_seconds_precision)]
- TIMESTAMP [(fractional_ seconds)] WITH TIME ZONE
- TIMESTAMP [(fractional_ seconds)] WITH LOCAL TIME ZONE

Простые типы

- INTERVAL YEAR [(year_ precision)] TO MONTH
- INTERVAL DAY [(day_precision)] TO SECOND [(fractional_ seconds)]
- ROWID
- UROWID [(size)]
- CHAR [(size [BYTE | CHAR])]
- NCHAR[(size)]
- BFILE

Составные типы

- LONG
- CLOB
- NCLOB
- BLOB
- RAW(size)
- LONG RAW

Таблицы

Таблицы представляют собой механизм сохранения информации в базе данных ORACLE. Они содержат фиксированный набор столбцов, в которых описываются атрибуты объекта, с которым эта таблица работает. У каждого столбца есть имя и уникальные характеристики.

Временные таблицы

Подобно регулярной таблице временная таблица является механизмом хранения данных в базе данных ORACLE. Временная таблица состоит из столбцов, имеющих типы данных и длину. В отличие от регулярной таблицы описание временной таблицы сохраняется, но данные, внесенные в таблицу, остаются в ней во время сеанса или во время транзакции. Создание временной таблицы в качестве глобальной временной таблицы обеспечивает для всех сеансов, поддерживающих соединение с базой данных, возможность видеть данную таблицу и пользоваться ею. Во время коллективных сеансов во временные таблицы можно вставлять строки данных. Однако каждая строка данных в таблице видна только для того сеанса, который вставил эту строку.

Представления в ORACLE

Идея представления (view) проста: определить запрос, который предполагается часто использовать, сохранить его в базе данных Oracle и разрешить пользователям обращаться к нему по имени, как к обычной таблице. Когда пользователь выбирает данные из представления, Oracle выполняет соответствующий запрос, организует результаты так, как определено в представлении, и выдает их пользователю. Для пользователя представление выглядит как таблица, из которой поступают данные. Однако на самом деле данные поступают через представление, из одного или нескольких других источников.

Представления часто используют:

- Чтобы поддерживать безопасность, поскольку ни позволяют ограничивать диапазон строк и столбцов, возвращаемых пользователям. Если вы не хотите, чтобы пользователи видели столбец с зарплатой из таблицы личных данных, просто не включайте его в определение представления. Для пользователей представления этот столбец не будет существовать. То же самое справедливо и для строк: включите в представление конструкцию WHERE, и возвращаемые записи будут отфильтрованы любым нужным вам образом.
- Чтобы скрывать сложность данных. Например, представление может быть определено соединением – количеством столбцов или строк в различных таблицах. Однако представление скрывает тот факт, что информация на самом деле происходит из нескольких таблиц.

Представления часто используют:

- Для упрощения использования для пользователей. Например, представления позволяют пользователям выбирать информацию из нескольких таблиц без необходимости знания как производить соединение.
- Для представления данных в проекции, не так как они расположены в основной базе данных. Например, столбцы или представление могут быть переименованы, не оказывая влияния на таблицы, на основе которых создано представление.
- Для изолирования приложения от возможных изменений в базовых таблицах. Например, представление определено запросом выбирающим три столбца из четырех столбцов таблицы, и в таблицу добавляется пятый столбец, на представление это не оказывает влияния и все приложения, работающие с представлением, также не меняются.

Представления часто используют:

- Для представления запросов, которые просто не могут быть сделаны без использования представлений. Например, представления которые соединяют представление основанное на предикате GROUP BY с таблицей, или представление которое объединяет с помощью оператора UNION представление и таблицу.
- Позволяет хранить сложнее запросы. Например, запрос может производить пространственные вычисления с информацией из таблиц. Сохраняя такой запрос в качестве представления, вы производите вычисления каждый раз при запросе к представлению.

Создание и удаление представления

- Метод создания представления – это сама простота. Нужно указать только имя представления и оператор SELECT, который будет выполняться при обращении к представлению. Вот соответствующий синтаксис:

```
CREATE [OR REPLACE] VIEW view_name AS
  оператор SELECT;
```

- Удалить представление так же легко, как и таблицу (но это действие менее разрушительно, поскольку представление не содержит никаких данных; самое худшее, к чему может привести случайное удаление представления – это к необходимости создавать его заново). Команда, удаляющая представление, имеет следующий синтаксис:

```
DROP VIEW view_name;
```