Другие объекты БД

Реализация Oracle 10g

Темы

- Некоторые объекты базы данных и их использование
- Создание, сопровождение и использование последовательностей
- Создание и сопровождение индексов
- Создание частных и общедоступных синонимов

Объекты базы данных

Объект	Описание
Таблица	Основная единица хранения; состоит из строк и столбцов
Представление	Логически представляет подмножества данных из одной или нескольких таблиц
Последовательность	Генерирует значения первичных ключей
Индекс	Увеличивает производительность некоторых запросов
Синоним	Дает альтернативные имена некоторым объектам

Что такое последовательность?

- Автоматически генерирует уникальные номера
- Является совместно используемым объектом
- Обычно используется для генерации значений главного ключа
- Заменяет код в прикладной программе
- Ускоряет доступ к числам последовательности, если они находятся в сверхоперативной памяти (кэш-памяти).

Команда CREATE SEQUENCE: синтаксис

Определение последовательности для автоматической генерации чисел

```
CREATE SEQUENCE sequence
          [INCREMENT BY n]
          [START WITH n]
          [{MAXVALUE n | NOMAXVALUE}]
          [{MINVALUE n | NOMINVALUE}]
          [{CYCLE | NOCYCLE}]
          [{CACHE n | NOCACHE}];
```

Создание последовательности: пример

- Создание последовательности DEPT_DEPTNO для использования в качестве главного ключа таблицы DEPT.
- При генерации значений главных ключей не используйте опцию CYCLE.

```
SQL> CREATE SEQUENCE dept_deptno
  2       INCREMENT BY 1
  3       START WITH 91
  4       MAXVALUE 100
  5       NOCACHE
  6       NOCYCLE;
Sequence created.
```

Проверка параметров последовательности

- Проверить значения последовательности можно в представлении USER_SEQUENCES словаря данных.

```
SQL> SELECT   sequence_name, min_value, max_value,
  2           increment_by, last_number
  3  FROM     user_sequences;
```

- Столбец LAST_NUMBER содержит следующее свободное число.

Псевдостолбцы NEXTVAL и CURRVAL

- NEXTVAL возвращает следующий свободный номер в последовательности.

  Столбец возвращает уникальное значение при каждом обращении к нему - даже для разных пользователей.

- Псевдостолбец CURRVAL возвращает текущее значение последовательности.

  Чтобы псевдостолбец CURRVAL содержал значение, необходимо прежде выполнить NEXTVAL для этой последовательности.

Псевдостолбцы NEXTVAL и CURRVAL

- NEXTVAL возвращает следующий свободный номер в последовательности.

  Столбец возвращает уникальное значение при каждом обращении к нему - даже для разных пользователей.

- Псевдостолбец CURRVAL возвращает текущее значение последовательности.

  Чтобы псевдостолбец CURRVAL содержал значение, необходимо прежде выполнить NEXTVAL для этой последовательности.

Использование последовательности

- Вставка нового отдела MARKETING, расположенного в Сан-Диего.

```
SQL> INSERT INTO     dept(deptno, dname, loc)
  2  VALUES          (dept_deptno.NEXTVAL,
  3                  'MARKETING', 'SAN DIEGO');
1 row created.
```

- Вывод текущего значения в последовательности DEPT_DEPTNO.

```
SQL> SELECT   dept_deptno.CURRVAL
  2  FROM     dual;
```

Использование последовательности

- Запись значений последовательности в сверхоперативную память (кэш) ускоряет доступ к ним.
- Возможные причины пропусков значений в последовательности:
  - Откат транзакции
  - Отказ системы
  - Использование последовательности в другой таблице
- Если последовательность создана с параметром NOCACHE, увидеть следующее свободное число в ней можно в таблице USER_SEQUENCES.

Изменение последовательности

Изменение шага приращения, максимального и минимального значений, режима циклической генерации значений и кэширования.

```
SQL> ALTER SEQUENCE dept_deptno
  2          INCREMENT BY 1
  3          MAXVALUE 999999
  4          NOCACHE
  5          NOCYCLE;
Sequence altered.
```

Изменение последовательности: указания

- Для изменения параметров необходимо быть владельцем последовательности или иметь для нее привилегию ALTER.
- Команда влияет только на числа, генерируемые после изменения.
- Чтобы начать последовательность с другого числа, необходимо удалить ее из словаря данных и создать заново.
- Выполняются некоторые проверки.

Что такое индекс?

- Объект базы данных
- Используется сервером Oracle для ускорения выборки строк с помощью указателя
- Уменьшает количество операций ввода-вывода с диском за счет использования быстрого метода поиска данных
- Независим от таблицы, для которой был создан
- Автоматически используется и поддерживается сервером Oracle

Как создаются индексы?

- • Автоматически
- - Индекс уникальных ключей создается автоматически, если в определении таблицы задано ограничение PRIMARY KEY или UNIQUE.
- • Вручную
- - Для ускорения доступа к строкам пользователь может создавать неуникальные индексы по столбцам.

Создание индекса: синтаксис

- Создание индекса по одному или нескольким столбцам

```
CREATE INDEX index
ON table (column[, column]...);
```

- Увеличение скорости доступа в столбцу ENAME таблицы EMP

```
SQL> CREATE INDEX    emp_ename_idx
  2  ON              emp(ename);
Index created.
```

Когда создавать индекс

- Столбец часто используется в предложении WHERE или условии соединения.
- Столбец имеет большой диапазон значений.
- Столбец содержит большое количество неопределенных значений.
- Два или более столбцов часто используются вместе в предложении WHERE или условии соединения.
- Таблица большая, и предполагается, что каждый запрос будет выбирать менее 2–4% строк.

Когда не создавать индекс

Не создавайте индекс, если:

- Таблица небольшого размера
- Столбцы редко используются как условие в запросе
- Предполагается, что каждый запрос будет выбирать более 2–4% строк.
- Таблица часто обновляется

Проверка индексов

- Представление словаря данных USER_INDEXES содержит имя индекса и информацию о его уникальности.
- Представление словаря данных USER_IND_COLUMNS содержит имя индекса, имя таблицы и имя столбца.

```
SQL> SELECT   ic.index_name, ic.column_name,
  2           ic.column_position col_pos,ix.uniqueness
  3  FROM     user_indexes ix, user_ind_columns ic
  4  WHERE    ic.index_name = ix.index_name
  5  AND      ic.table_name = 'EMP';
```

Удаление индекса: синтаксис

- Удаление индекса из словаря данных.

```
SQL> DROP INDEX index;
```

- Удаление индекса EMP_ENAME_IDX из словаря данных.

```
SQL> DROP INDEX emp_ename_idx;
Index dropped.
```

- Для удаления индекса необходимо быть его владельцем или иметь привилегию DROP ANY INDEX.

Синонимы

Синонимы (альтернативные имена объектов) упрощают доступ к объектам:

- **Позволяют обращаться к таблицам других пользователей.**
- **Устраняют необходимость использования длинных имен объектов.**

```
CREATE [PUBLIC] SYNONYM synonym
FOR    object;
```

Создание и удаление синонимов

- Создание более короткого имени для представления DEPT_SUM_VU.

```
SQL> CREATE SYNONYM d_sum
  2  FOR              dept_sum_vu;
Synonym Created.
```

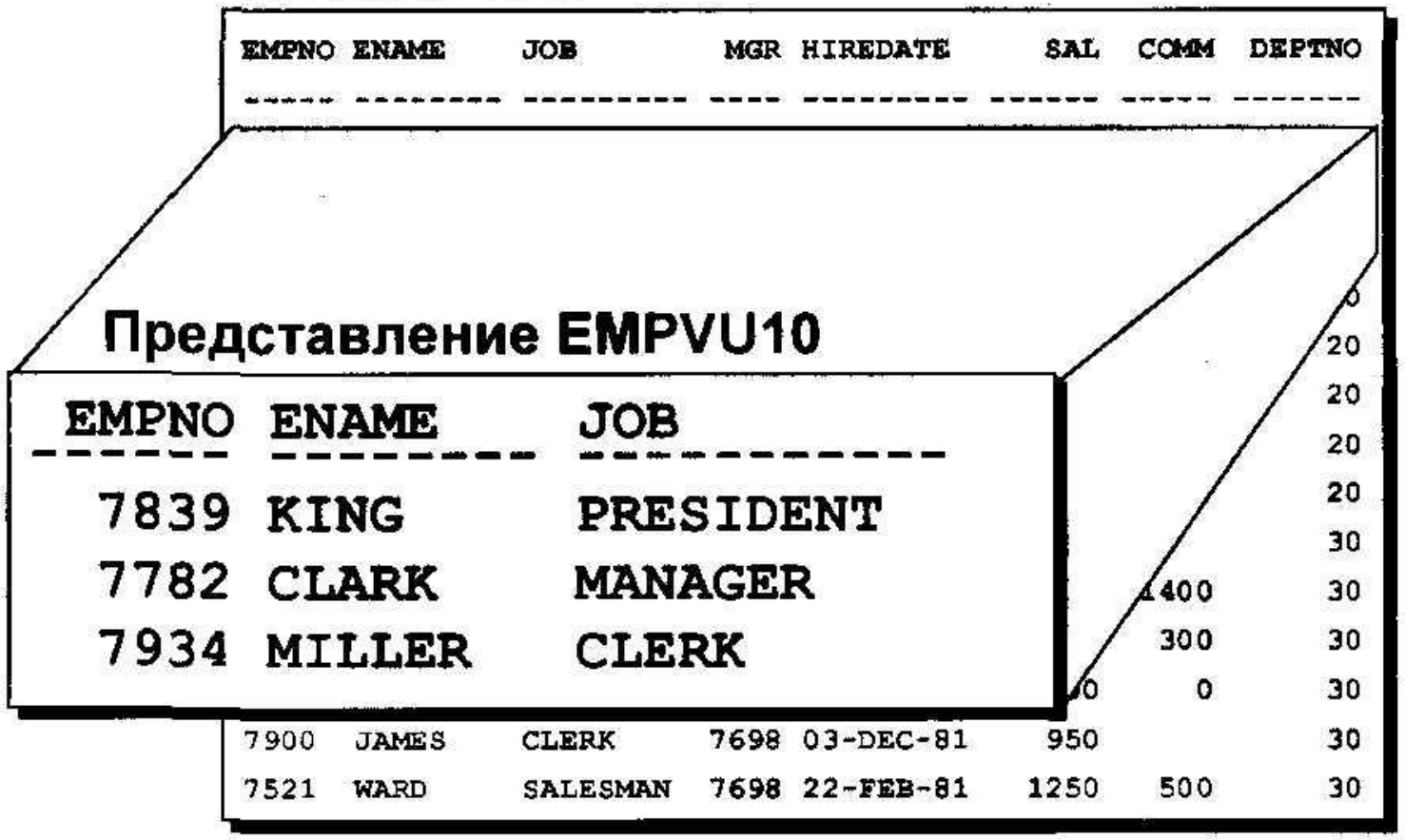
- Удаление синонима.

```
SQL> DROP SYNONYM d_sum;
Synonym dropped.
```

Заключение

- Для автоматической генерации чисел используется генератор последовательностей.
- Информация о последовательностях содержится в таблице словаря данных USER_SEQUENCES.
- Индексы ускоряют выборку данных по запросам.
- Информация об индексах содержится в таблице словаря данных USER_INDEXES.
- Синонимы позволяют объектам иметь альтернативные имена.

Что такое представление?



Зачем нужны представления?

- Для ограничения доступа к базе данных
- Для упрощения сложных запросов
- Для обеспечения независимости от данных
- Для представления одних и тех же данных в разных видах

Простые и сложные представления

Характеристика	Простые	Сложные
Количество таблиц	Одна	Одна или более
Содержат функции	Нет	Да
Содержат группы данных (предложение DISTINCT или групповые функции)	Нет	Да
Выполнение операций DML с представлениями	Да	Не всегда

Создание представления

- В команду CREATE VIEW включается подзапрос.

```
CREATE [OR REPLACE] [FORCE|NOFORCE] VIEW view
  [(alias[, alias]...)]
AS subquery
[WITH CHECK OPTION [CONSTRAINT constraint]]
[WITH READ ONLY]
```

- Подзапрос может содержать сложную команду SELECT.
- Подзапрос не может содержать предложение ORDER BY.

Создание представления: пример

- Создание представления EMPVU10 с информацией о служащих отдела 10.

```
SQL> CREATE VIEW     empvu10
  2  AS SELECT       empno, ename, job
  3  FROM            emp
  4  WHERE           deptno = 10;
View created.
```

- Вывод структуры представления с помощью команды DESCRIBE SQL*Plus.

```
SQL> DESCRIBE empvu10
```

Создание представления: пример

- Создайте представление с псевдонимами столбцов в подзапросе.

```
SQL> CREATE VIEW     salvu30
  2  AS SELECT       empno EMPLOYEE_NUMBER, ename NAME,
  3                  sal SALARY
  4  FROM            emp
  5  WHERE           deptno = 30;
View created.
```

- Производите выборку столбцов из представления по этим псевдонимам.

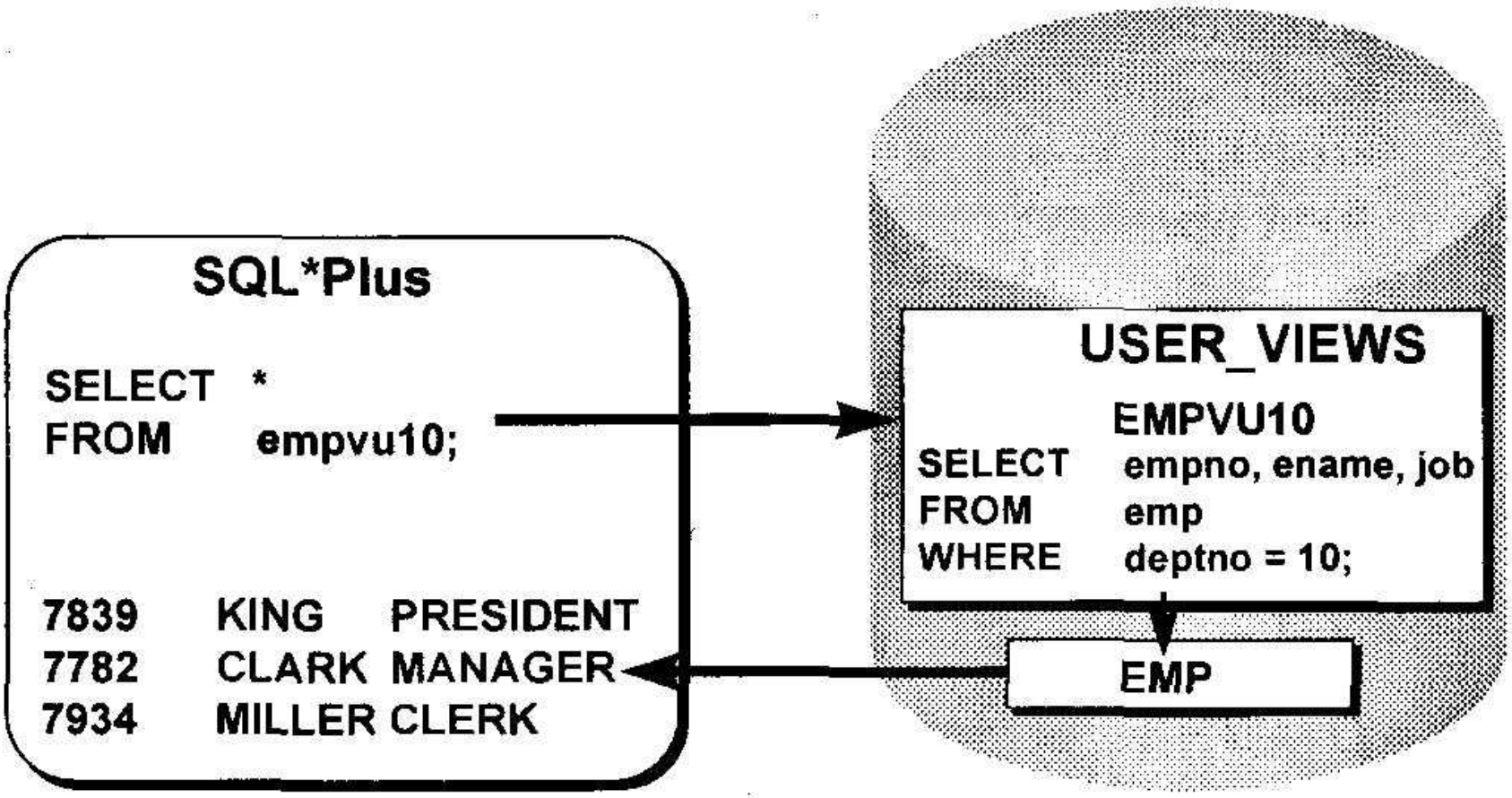
Выборка данных из представления: пример

```
SQL>   SELECT *
  2    FROM    salvu30;
```

```
EMPLOYEE_NUMBER NAME                 SALARY
--------------- ---------- ---------
           7698 BLAKE                2850
           7654 MARTIN               1250
           7499 ALLEN                1600
           7844 TURNER               1500
           7900 JAMES                 950
           7521 WARD                 1250

6 rows selected.
```

Запрос представления



Изменение представления: пример

- Изменение представления EMPVU10 с помощью предложения CREATE OR REPLACE VIEW. Добавление псевдонима для каждого столбца.

```
SQL> CREATE OR REPLACE VIEW empvu10
  2          (employee_number, employee_name, job_title)
  3   AS SELECT       empno, ename, job
  4   FROM            emp
  5   WHERE           deptno = 10;
View created.
```

- Порядок псевдонимов в предложении CREATE VIEW должен быть таким же, как порядок столбцов в подзапросе.

Создание сложного представления : пример

Создание сложного представления с групповыми функциями для вывода данных из двух таблиц.

```
SQL> CREATE VIEW     dept_sum_vu
  2                  (name, minsal, maxsal, avgsal)
  3  AS SELECT       d.dname, MIN(e.sal), MAX(e.sal),
  4                  AVG(e.sal)
  5  FROM            emp e, dept d
  6  WHERE           e.deptno = d.deptno
  7  GROUP BY        d.dname;
View created.
```

Правила выполнения DML операций с представлением

- Операции DML можно выполнять с простыми представлениями.
- Нельзя удалить строку, если представление содержит:
  - Групповые функции
  - Предложение GROUP BY
  - Ключевое слово DISTINCT

Правила выполнения DML операций с представлением

- Невозможно изменить данные в представлении, которое содержит:
  - Любое из указанного на предыдущем слайде
  - Столбцы, определяемые выражениями
  - Псевдостолбец ROWNUM
- Невозможно добавить данные, если:
  - Представление содержит любое из вышеуказанных условий на этом и предыдущем слайдах.
  - В базовых таблицах имеются столбцы с ограничением NOT NULL, которые не были определены при создании представления.

Использование предложения WITH CHECK OPTION

- Необходимо следить за тем, чтобы результаты DML операций оставались в пределах домена представления.

```
SQL> CREATE OR REPLACE VIEW empvu20
  2  AS SELECT       *
  3  FROM            emp
  4  WHERE           deptno = 20
  5  WITH CHECK OPTION CONSTRAINT empvu20_ck;
View created.
```

- Попытка изменить номер отдела для какой-либо строки в представлении закончится неудачей, т.к. при этом нарушится ограничение CHECK OPTION.

Запрет DML операций

- Использование опции WITH READ ONLY запрещает выполнять над представлением любые DML операции.

```
SQL> CREATE OR REPLACE VIEW empvu10
  2         (employee_number, employee_name, job_title)
  3  AS SELECT       empno, ename, job
  4  FROM            emp
  5  WHERE           deptno = 10
  6  WITH READ ONLY;
View created.
```

- Попытка выполнить команду DML для любой строки представления вызовет ошибку сервера Oracle ORA-01752.

Удаление представления

Удаление представления не вызывает потери данных, т.к. представление основано на реальных таблицах базы данных.

```
DROP VIEW view;
```

```
SQL> DROP VIEW empvu10;
View dropped.
```

Заключение

- Представление создается на основе данных из других таблиц или представлений.
- Преимущества представлений:
  - Ограничивают доступ к базе данных
  - Упрощают запросы
  - Обеспечивают независимость данных
  - Позволяют по-разному видеть одни и те же данные
  - Могут быть удалены без удаления данных из таблиц, лежащих в их основе