Манипулирование данными

- Описание команд DML
- Вставка строк в таблицы
- Обновление строк в таблице
- Удаление строк из таблицы
- Управление транзакциями

Язык манипулирования данными (DML)

- • Команды DML выполняются при следующих операциях:
- Вставка новых строк в таблицу
- Изменение существующих строк в таблице
- Удаление существующих строк из таблицы
- * *Транзакция* - это совокупность команд DML, образующих логическую единицу работы.

Вставка новой строки в таблицу

50	DEVELOPMENT	DETROIT

Новая строка

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

"...вставка новой строки в таблицу DEPT ..."

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
50	DEVELOPMENT	DETROIT

Вставка новых строк в таблицу: пример

- Вставка новой строки, содержащей значения для каждого из столбцов.
- Значения указываются в стандартном порядке столбцов таблицы (используемом по умолчанию).
- Перечисление столбцов в предложении INSERT необязательно.

```
SQL> INSERT INTO      dept (deptno, dname, loc)
  2  VALUES           (50, 'DEVELOPMENT', 'DETROIT');
1 row created.
```

- Символьные значения и даты заключаются в апострофы.

Вставка строк с неопределенными значениями

- Неявный метод: столбец не указывается в списке столбцов.

```
SQL> INSERT INTO      dept (deptno, dname)
  2  VALUES           (60, 'MIS');
1 row created.
```

- Явный метод: использование ключевого слова NULL или пустой строки (‘’) в списке VALUES.

```
SQL> INSERT INTO      dept
  2  VALUES           (70, 'FINANCE', NULL);
1 row created.
```

Вставка специальных значений

Функция SYSDATE записывает текущие дату и время.

```
SQL> INSERT INTO     emp (empno, ename, job,
  2                  mgr, hiredate, sal, comm,
  3                  deptno)
  4  VALUES          (7196, 'GREEN', 'SALESMAN',
  5                  7782, SYSDATE, 2000, NULL,
  6                  10);
1 row created.
```

Вставка конкретных значений даты и времени

- Добавление нового служащего

```
SQL> INSERT INTO emp
  2  VALUES       (2296,'AROMANO','SALESMAN',7782,
  3                TO_DATE('FEB 3,97', 'MON DD, YY'),
  4                1300, NULL, 10);
1 row created.
```

- Проверка вставки.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
2296	AROMANO	SALESMAN	7782	03-FEB-97	1300		10

Вставка значений с помощью переменных подстановки

Создайте интерактивный скрипт-файл с помощью параметров подстановки SQL*Plus.

```
SQL> INSERT INTO      dept (deptno, dname, loc)
  2  VALUES           (&department_id,
  3                   '&department_name', '&location');
```

```
Enter value for department_id: 80
Enter value for department_name: EDUCATION
Enter value for location: ATLANTA

1 row created.
```

Команда UPDATE

- **Для обновления существующих строк используется команда UPDATE.**

```
UPDATE      table
SET         column = value [, column = value]
[WHERE      condition];
```

- **В случае необходимости можно одновременно обновлять несколько строк**

Обновление строк в таблице: пример

- **Предложение WHERE позволяет изменить конкретную строку или строки.**

```
SQL> UPDATE   emp
  2  SET      deptno = 20
  3  WHERE    empno = 7782;
1 row updated.
```

- **Если предложение WHERE отсутствует, обновляются все строки таблицы.**

```
SQL> UPDATE   employee
  2  SET      deptno = 20;
14 rows updated.
```

Обновление с помощью многостолбцового подзапроса

Изменение должности и номера отдела служащего под номером 7698, чтобы они стали такими же, как у служащего под номером 7499.

```
SQL> UPDATE   emp
  2  SET      (job, deptno) =
  3                          (SELECT job, deptno
  4                          FROM    emp
  5                          WHERE   empno = 7499)
  6  WHERE    empno = 7698;
1 row updated.
```

Обновление строк на основе значений из другой таблицы

Для изменения строк таблицы на основе значений из другой таблицы используйте подзапросы в командах UPDATE.

```
SQL> UPDATE   employee
  2  SET      deptno =   (SELECT   deptno
  3                      FROM     emp
  4                      WHERE    empno = 7788)
  5  WHERE    job    =   (SELECT   job
  6                      FROM     emp
  7                      WHERE    empno = 7788);
2 rows updated.
```

Обновление строк: нарушение правила целостности данных

```
SQL> UPDATE   emp
  2  SET      deptno = 55
  3  WHERE    deptno = 10;
```

```
UPDATE emp
       *
ERROR at line 1:
ORA-02291: integrity constraint (USR.EMP_DEPTNO_FK)
violated - parent key not found
```

Отдела номер 55 не существует

Удаление строки из таблицы

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
50	DEVELOPMENT	DETROIT
60	MIS	
...		

"...удаление строки из таблицы DEPT ..."

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
60	MIS	
...		

Команда DELETE

Для удаления строк используется команда DELETE.

```
DELETE [FROM]   table
[WHERE          condition];
```

Удаление строк из таблицы: пример

- Конкретная строка или строки удаляются с помощью предложения WHERE.

```
SQL> DELETE FROM     department
  2  WHERE           dname = 'DEVELOPMENT';
1 row deleted.
```

- Если предложение WHERE отсутствует, удаляются все строки таблицы.

```
SQL> DELETE FROM     department;
4 rows deleted.
```

Удаление строк на основе значений из другой таблицы

Для удаления строк на основе значений из другой таблицы используйте подзапросы в командах DELETE.

```
SQL> DELETE FROM     employee
  2  WHERE            deptno =
  3                          (SELECT   deptno
  4                           FROM     dept
  5                           WHERE    dname ='SALES');
6 rows deleted.
```

Удаление строк: нарушение правила целостности

```
SQL> DELETE FROM     dept
  2  WHERE           deptno = 10;
```

```
DELETE FROM dept
            *
ERROR at line 1:
ORA-02292: integrity constraint (USR.EMP_DEPTNO_FK)
violated - child record found
```

Нельзя удалить строку, содержащую первичный ключ, который используется в качестве внешнего ключа в другой таблице.

Транзакции базы данных

- Сервер Oracle обеспечивает согласованность данных на основе транзакций.
- Транзакции обеспечивают большую гибкость, более широкий спектр средств управления при изменении данных, а также согласованность данных в случае ошибки в пользовательском процессе или сбоя системы.

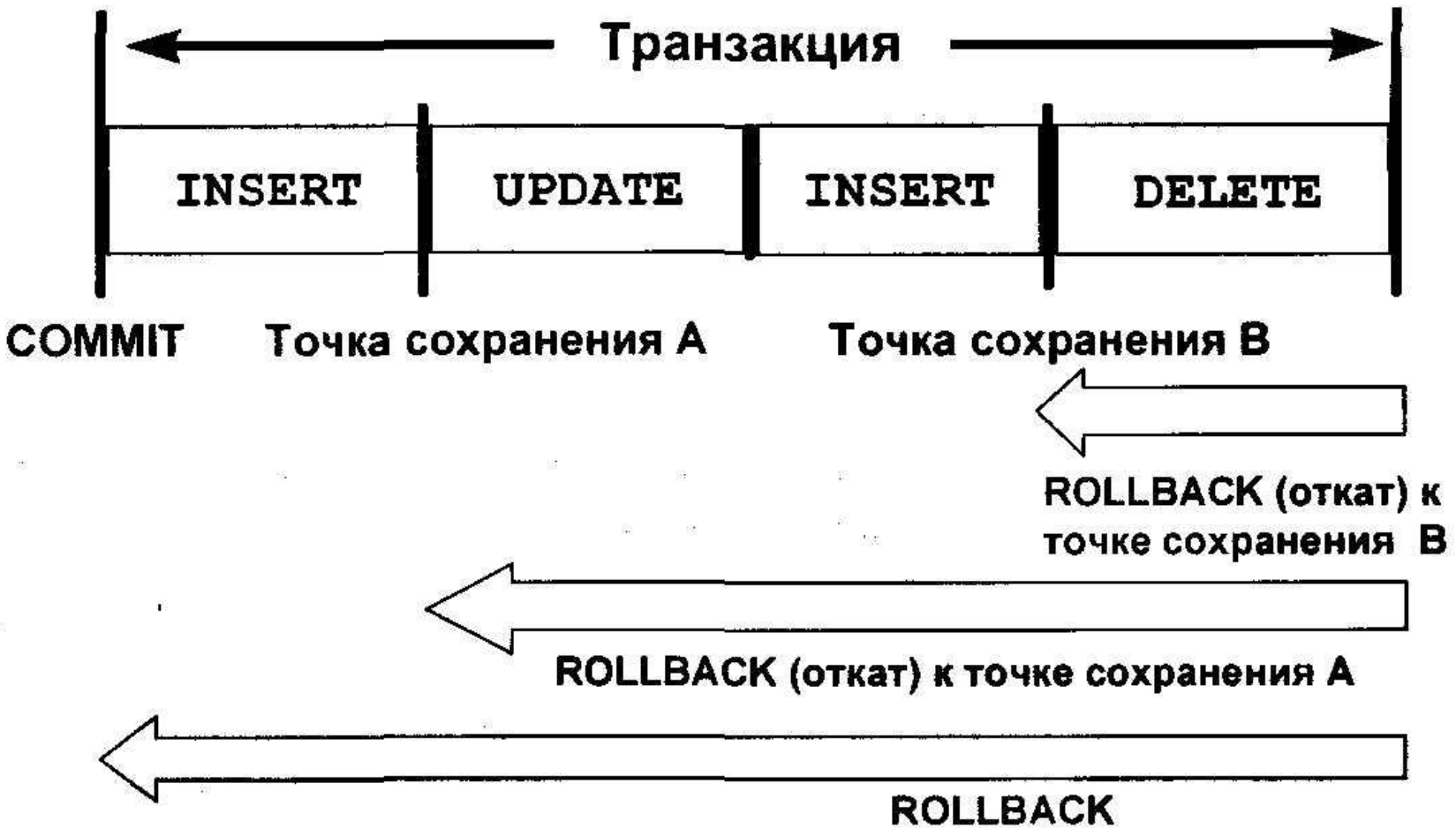
Транзакции базы данных

- Начинаются с выполнения первой исполняемой команды SQL.
- Заканчиваются одним из следующих событий:
  - Выполнение команды COMMIT или ROLLBACK
  - Выполнение команды DDL или DCL (автоматическая фиксация транзакции)
  - Выходы в подпрограммы пользователя
  - Отказы системы

Преимущества команд COMMIT и ROLLBACK

- Обеспечивают согласованность данных.
- Позволяют проверить изменения в данных прежде, чем сделать их постоянными.
- Логически группируют взаимосвязанные операции.
-

Управление транзакциями



Неявная обработка транзакций

- Автоматическая фиксация изменений (COMMIT) происходит в следующих случаях:
  - Выполнение команды DDL
  - Выполнение команды DCL
  - Нормальный выход из SQL*Plus без явной посылки команды COMMIT или ROLLBACK
- Автоматический откат (ROLLBACK) выполняется в случае аварийного прекращения сеанса работы в SQL*Plus или отказа системы

Состояние данных до выполнения команды COMMIT или ROLLBACK

- Предыдущее состояние данных может быть восстановлено, т.к. изменения производятся в буфере базы данных.
- Текущий пользователь может просмотреть результаты своих операций DML с помощью команды SELECT.
- Другие пользователи не *могут* видеть , результаты команд DML, выполняемых текущим пользователем.
- Измененяемые строки *блокируются,* и другие пользователи не могут обновлять их содержимое.

Состояние данных после выполнения команды COMMIT

- Измененные данные записываются в базу данных.
- Предшествующее состояние данных теряется.
- Все пользователи могут видеть результаты.
- Измененные строки разблокируются, и другие пользователи получают доступ к ним для обработки данных.
- Все точки сохранения стираются.

Фиксация изменений в данных

- **Внесение изменений.**

```
SQL> UPDATE   emp
  2  SET      deptno = 10
  3  WHERE    empno = 7782;
1 row updated.
```

- **Фиксация изменений.**

```
SQL> COMMIT;
Commit complete.
```

Состояние данных после выполнения команды ROLLBACK

Команда ROLLBACK отменяет все незавершенные изменения.

- **Изменения в данных отменяются.**
- **Данные возвращаются в прежнее состояние.**
- **Блокировка строк, над которыми выполнялись операции, отменяется.**

```
SQL> DELETE FROM    employee;
14 rows deleted.
SQL> ROLLBACK;
Rollback complete.
```

Откат к точке сохранения

- Маркеры (точки сохранения) для отката в текущей транзакции создаются с помощью команды SAVEPOINT.
- Откат до такого маркера выполняется с помощью команды ROLLBACK TO SAVEPOINT.

```
SQL> UPDATE...
SQL> SAVEPOINT update_done;
Savepoint created.
SQL> INSERT...
SQL> ROLLBACK TO update_done;
Rollback complete.
```

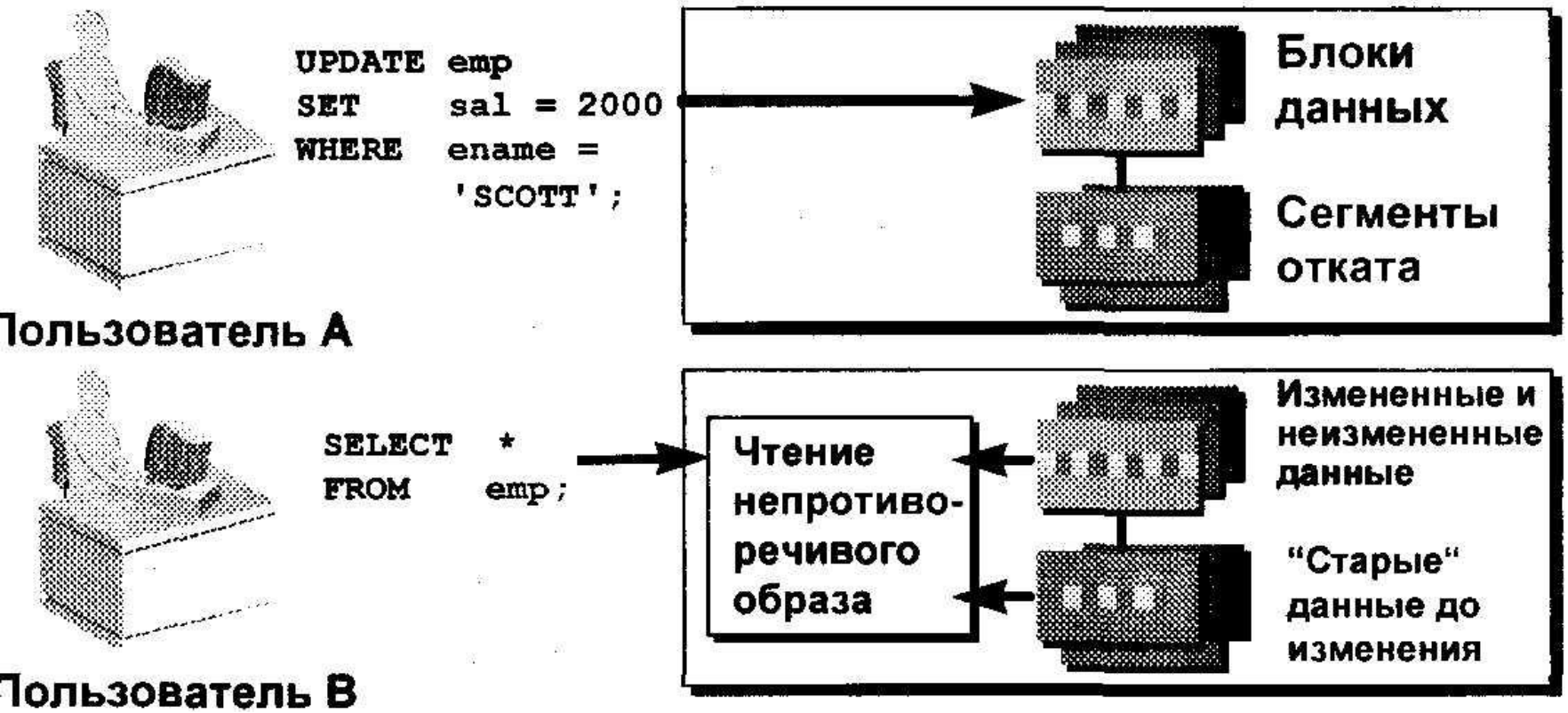
Откат на уровне команды

- Если ошибка возникла при выполнении одной конкретной команды DML, отменяются только результаты этой команды.
- Сервер Oracle использует неявную точку сохранения.
- Все прочие изменения сохраняются.
- Пользователю следует завершать транзакции явно командой COMMIT или ROLLBACK.

Согласованность чтения

- Согласованность чтения гарантирует непротиворечивое представление данных в любой момент времени.
- Изменения, сделанные одним пользователем, не вступают в противоречие с изменениями, сделанными другим пользователем.
- Гарантируется, что для одних и тех же данных:
  - “Читатели“ никогда не блокируют “писателей“.
  - “Писатели“ никогда не блокируют “читателей“.

Реализация согласованности чтения



Блокировка данных

Блокировка данных в Oracle:

- **Предотвращает деструктивное взаимодействие между одновременными транзакциями**
- **Не требует действий со стороны пользователя**
- **Автоматически использует ограничение самого низкого уровня**
- **Сохраняется до конца транзакции**
- **Имеет два основных режима:**
  - **Исключительная блокировка (*Exclusive*)**
  - **Разделяемая блокировка (*Share*)**

Заключение

Команда	Описание
INSERT	Вставляет новую строку в таблицу.
UPDATE	Изменяет существующие строки таблицы.
DELETE	Удаляет существующие строки из таблицы.
COMMIT	Делает все незафиксированные изменения постоянными.
SAVEPOINT	Позволяет произвести откат до определенной точки сохранения.
ROLLBACK	Отменяет все незафиксированные изменения данных.

Создание таблиц и управление ими

- Главные объекты базы данных
- Создание таблиц
- Типы данных, которые могут использоваться в определениях столбцов
- Изменение определений таблиц
- Удаление, переименование и усечение таблиц

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

Правила присвоения имен

- Имя должно начинаться с буквы
- Может быть длиной до 30 символов
- Должно содержать только символы A–Z, a–z, 0–9, _, $ и #
- Не должно совпадать с именем другого объекта, принадлежащего этому же самому пользователю
- Не должно совпадать со словом, *зарезервированным сервером Oracle*

Команда CREATE TABLE

- Необходимо иметь :
  - привилегию CREATE TABLE
  - Область хранения

```
CREATE TABLE [schema.]table
             (column datatype [DEFAULT expr]);
```

- Вы задаете:
  - Имя таблицы
  - Имя столбца, тип данных столбца и размер столбца, значение по умолчанию

Ссылки на таблицы других пользователей

- Таблицы, принадлежащие другим пользователям, не входят в схему пользователя.
- В качестве префикса в имени таблицы
  следует указать имя владельца.

Опция DEFAULT

- Задает значение по умолчанию, если при вставке данных значение не указано явно.

```
... hiredate DATE DEFAULT SYSDATE, ...
```

- В качестве значения допускается литерал, выражение или функция SQL.
- Не может использоваться имя другого столбца или псевдостолбец.
- Тип данных, используемый по умолчанию, должен совпадать с типом данных столбца.

Создание таблиц

- Создание таблицы.

```
SQL> CREATE TABLE dept
  2          (deptno  NUMBER(2),
  3           dname   VARCHAR2(14),
  4           loc     VARCHAR2(13));
Table created.
```

- Проверка создания таблицы.

```
SQL> DESCRIBE dept
```

```
Name                            Null?     Type
------------------------------- --------- ---------
DEPTNO                          NOT NULL  NUMBER(2)
DNAME                                     VARCHAR2(14)
LOC                                       VARCHAR2(13)
```

Запрос к словарю данных

- **Вывод описаний таблиц, принадлежащих пользователю.**

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

- **Просмотр типов объектов, принадлежащих пользователю.**

```
SQL> SELECT   DISTINCT object_type
  2  FROM     user_objects;
```

- **Просмотр таблиц, представлений, синонимов и последовательностей, принадлежащих пользователю.**

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

Типы данных

Тип данных	Описание
VARCHAR2(*size*)	Символьные данные переменной длины
CHAR(*size*)	Символьные данные постоянной длины
NUMBER(*p*,*s*)	Числовые данные переменной длины
DATE	Значения даты и времени
LONG	Символьные данные переменной длины до 2 гигабайтов
CLOB	Однобайтовые символьные данные до 4 гигабайтов
RAW и LONG RAW	Необработанные (“сырые“) двоичные данные
BLOB	Двоичные данные до 4 гигабайтов
BFILE	Двоичные данные, которые хранятся во внешнем файле; длина до 4 гигабайтов

Создание таблицы с использованием подзапроса

- Создание таблицы и вставка строк путем совместного использования команды CREATE TABLE и опции "AS *subquery*".

```
CREATE TABLE table
        [column(, column...)]
AS subquery;
```

- Количество заданных столбцов должно совпадать с количеством столбцов в подзапросе.
- Столбцы могут быть определены с именами и значениями, используемыми по умолчанию.

Команда ALTER TABLE

Команда ALTER TABLE используется для следующих операций:

- **Добавление столбца**
- ***Изменение существующего столбца***
- **Задание значения по умолчанию для нового столбца**

```
ALTER TABLE table
ADD         (column datatype [DEFAULT expr]
            [, column datatype]...);
```

```
ALTER TABLE table
MODIFY      (column datatype [DEFAULT expr]
            [, column datatype]...);
```

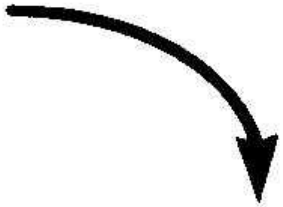
Добавление столбца

DEPT30

Новый столбец

EMPNO	ENAME	ANNSAL	HIREDATE	JOB
7698	BLAKE	34200	01-MAY-81	
7654	MARTIN	15000	28-SEP-81	
7499	ALLEN	19200	20-FEB-81	
7844	TURNER	18000	08-SEP-81	
...				

"...вставка столбца в таблицу DEPT30..."



DEPT30

EMPNO	ENAME	ANNSAL	HIREDATE	JOB
7698	BLAKE	34200	01-MAY-81	
7654	MARTIN	15000	28-SEP-81	
7499	ALLEN	19200	20-FEB-81	
7844	TURNER	18000	08-SEP-81	
...				

Добавление столбца

- Столбцы добавляются с помощью предложения ADD.

```
SQL> ALTER TABLE dept30
  2  ADD          (job VARCHAR2(9));
Table altered.
```

- Новый столбец становится в таблице последним.

```
     EMPNO ENAME          ANNSAL HIREDATE  JOB
---------- ---------- --------- --------- ----
      7698 BLAKE          34200 01-MAY-81
      7654 MARTIN         15000 28-SEP-81
      7499 ALLEN          19200 20-FEB-81
      7844 TURNER         18000 08-SEP-81
...
6 rows selected.
```

Изменение столбца

- Вы можете изменить тип данных столбца, размер и значение, используемое по умолчанию.

```
ALTER TABLE   dept30
MODIFY        (ename VARCHAR2(15));
Table altered.
```

- Новое значение по умолчанию влияет только на последующие вставки в таблицу.

Удаление таблицы с помощью команды DROP

- Удаляются все данные и структура таблицы.
- Все незафиксированные транзакции фиксируются
- Все индексы удаляются.
- Откат этой команды *невозможен*.

```
SQL> DROP TABLE dept30;
Table dropped.
```