Обработка данных

Цели

Изучив материал этого занятия, вы освоите следующие темы:

- Описание всех инструкций языка манипулирования данными (DML)
- Вставка строк в таблицу
- Обновление строк в таблице
- Удаление строк из таблицы
- Управление транзакциями

План занятия

- Добавление новых строк в таблицу
  - инструкция `INSERT`
- Изменение данных в таблице
  - инструкция `UPDATE`
- Удаление строк из таблицы:
  - инструкция `DELETE`
  - инструкция `TRUNCATE`
- Управление транзакциями базы данных с помощью инструкций `COMMIT`, `ROLLBACK` и `SAVEPOINT`
- Целостность чтения
- Предложение `FOR UPDATE` в инструкции `SELECT`

Язык манипулирования данными

- Инструкция DML выполняется в следующих ситуациях:
  - добавление новых строк в таблицу
  - изменение существующих строк в таблице
  - удаление существующих строк из таблицы
- *Транзакция* состоит из набора инструкций DML, образующих логический рабочий блок.

Добавление новой строки в таблицу

	70 Public Relations	100	1700

Новая строка

DEPARTMENTS

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	50	Shipping	124	1500
4	60	IT	103	1400
5	80	Sales	149	2500
6	90	Executive	100	1700
7	110	Accounting	205	1700
8	190	Contracting	(null)	1700

Вставка новой строки в таблицу DEPARTMENTS.

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	50	Shipping	124	1500
4	60	IT	103	1400
5	80	Sales	149	2500
6	90	Executive	100	1700
7	110	Accounting	205	1700
8	190	Contracting	(null)	1700

9	70	Public Relations	100	1700

Синтаксис инструкции INSERT

- Добавление нескольких строк в таблицу с помощью инструкции INSERT

```
INSERT INTO   table [(column [, column...])]
VALUES     (value [, value...]);
```

- При использовании этого синтаксиса вставляется только одна строка.

Вставка новых строк

- Вставьте новую строку, которая содержит значения для каждого столбца.
- Перечислите значения в соответствии со стандартным порядком столбцов в таблице.
- Перечислите столбцы в предложении `INSERT` (необязательно).

```
INSERT INTO departments(department_id,
       department_name, manager_id, location_id)
VALUES (70, 'Public Relations', 100, 1700);
```

```
1 rows inserted
```

- Символьные значения и даты заключаются в одиночные кавычки.

Вставка строк с пустыми значениями (Null)

- Неявный метод: исключение столбца из списка столбцов.

```
INSERT INTO  departments (department_id,
                          department_name)
VALUES     (30, 'Purchasing');

1 rows inserted
```

- Явный метод: задание ключевого слова `NULL` в предложении `VALUES`.

```
INSERT INTO  departments
VALUES     (100, 'Finance', NULL, NULL);

1 rows inserted
```

Вставка специальных значений

Функция SYSDATE записывает текущую дату и время.

```
INSERT INTO employees (employee_id,
                       first_name, last_name,
                       email, phone_number,
                       hire_date, job_id, salary,
                       commission_pct, manager_id,
                       department_id)
VALUES          (113,
                       'Louis', 'Popp',
                       'LPOPP', '515.124.4567',
                       SYSDATE, 'AC_ACCOUNT', 6900,
                       NULL, 205, 110);
```

```
1 rows inserted
```

Вставка конкретных значений даты и времени

- Добавьте нового работника.

```
INSERT INTO employees
VALUES      (114,
             'Den', 'Raphealy',
             'DRAPHEAL', '515.127.4561',
             TO_DATE('FEB 3, 1999', 'MON DD, YYYY'),
             'SA_REP', 11000, 0.2, 100, 60);
```

```
1 rows inserted
```

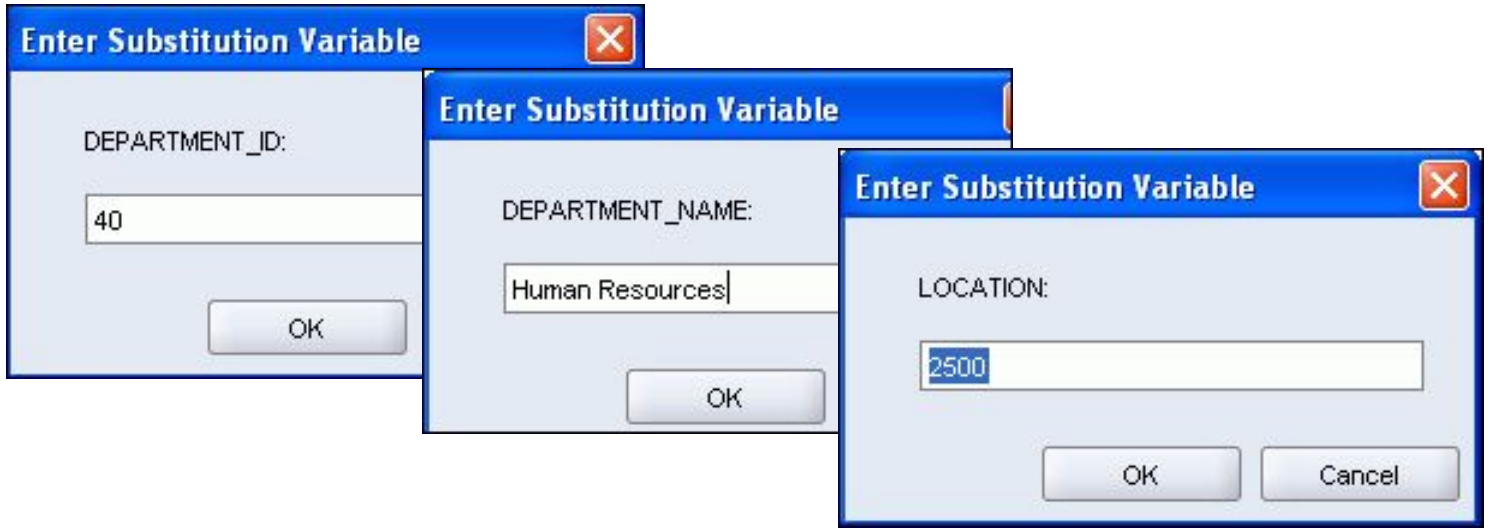
- Проверьте добавление.

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT
114	Den	Raphealy	DRAPHEAL	515.127.4561	03-FEB-99	SA_REP	11000	0.2

Создание сценария

- Подстановочный символ & в инструкции SQL используется для запроса значений.
- Символ & является фиктивным значением переменной.

```
INSERT INTO departments
            (department_id, department_name, location_id)
VALUES      (&department_id, '&department_name',&location);
```



Копирование строк из другой таблицы

- Запишите инструкцию `INSERT` с подзапросом:

```
INSERT INTO sales_reps(id, name, salary, commission_pct)
  SELECT employee_id, last_name, salary, commission_pct
  FROM   employees
  WHERE  job_id LIKE '%REP%';

4 rows inserted
```

- Не используйте предложение `VALUES`.
- Число столбцов в предложении `INSERT` и подзапросе должно совпадать.
- Вставьте все строки, возвращенные по подзапросу, в таблицу `sales_reps`.

План занятия

- Добавление новых строк в таблицу
  - инструкция `INSERT`
- Изменение данных в таблице
  - инструкция `UPDATE`
- Удаление строк из таблицы:
  - инструкция `DELETE`
  - инструкция `TRUNCATE`
- Управление транзакциями базы данных с помощью инструкций `COMMIT`, `ROLLBACK` и `SAVEPOINT`
- Целостность чтения
- Предложение `FOR UPDATE` в инструкции `SELECT`

Изменение данных в таблице

EMPLOYEES

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	MANAGER_ID	COMMISSION_PCT	DEPARTMENT_ID
100	Steven	King	24000	(null)	(null)	90
101	Neena	Kochhar	17000	100	(null)	90
102	Lex	De Haan	17000	100	(null)	90
103	Alexander	Hunold	9000	102	(null)	60
104	Bruce	Ernst	6000	103	(null)	60
107	Diana	Lorentz	4200	103	(null)	60
124	Kevin	Mourgos	5800	100	(null)	50

Обновите строки в таблице EMPLOYEES:

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	MANAGER_ID	COMMISSION_PCT	DEPARTMENT_ID
100	Steven	King	24000	(null)	(null)	90
101	Neena	Kochhar	17000	100	(null)	90
102	Lex	De Haan	17000	100	(null)	90
103	Alexander	Hunold	9000	102	(null)	80
104	Bruce	Ernst	6000	103	(null)	80
107	Diana	Lorentz	4200	103	(null)	80
124	Kevin	Mourgos	5800	100	(null)	50

Синтаксис инструкции UPDATE

- Измените существующие значения в таблице с помощью инструкции UPDATE:

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

- Обновите сразу несколько строк (при необходимости).

Обновление строк в таблице

- При использовании предложения `WHERE` изменяются значения конкретных строк:

```
UPDATE employees
SET    department_id = 50
WHERE  employee_id = 113;
1 rows updated
```

- При пропуске предложения `WHERE` изменяются значения всех строк в таблице:

```
UPDATE     copy_emp
SET        department_id = 110;
22 rows updated
```

- Укажите `SET` *`column_name`*`= NULL`, чтобы изменить значение столбца на `NULL`.

Обновление двух столбцов с помощью подзапроса

Обновите должность и оклад работника 113, чтобы они совпадали с аналогичными значениями для работника 205.

```
UPDATE    employees
SET       job_id   = (SELECT   job_id
                     FROM     employees
                     WHERE    employee_id = 205),
          salary   = (SELECT   salary
                     FROM     employees
                     WHERE    employee_id = 205)
WHERE     employee_id     =  113;
```

```
1 rows updated
```

Обновление строк на основе другой таблицы

Использование подзапросов в инструкциях `UPDATE` позволяет обновлять значения строк в таблице на основе значений из другой таблицы:

```
UPDATE   copy_emp
SET      department_id  =  (SELECT department_id
                           FROM employees
                           WHERE employee_id = 100)
WHERE    job_id         =  (SELECT job_id
                           FROM employees
                           WHERE employee_id = 200);
1 rows updated
```

План занятия

- Добавление новых строк в таблицу
  - инструкция `INSERT`
- Изменение данных в таблице
  - инструкция `UPDATE`
- Удаление строк из таблицы:
  - инструкция `DELETE`
  - инструкция `TRUNCATE`
- Управление транзакциями базы данных с помощью инструкций `COMMIT`, `ROLLBACK` и `SAVEPOINT`
- Целостность чтения
- Предложение `FOR UPDATE` в инструкции `SELECT`

Удаление строки из таблицы

DEPARTMENTS

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	50	Shipping	124	1500
4	60	IT	103	1400
5	80	Sales	149	2500
6	90	Executive	100	1700
7	110	Accounting	205	1700
8	190	Contracting	(null)	1700

Удаление строки из таблицы DEPARTMENTS:

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	50	Shipping	124	1500
4	60	IT	103	1400
5	80	Sales	149	2500
6	90	Executive	100	1700
7	110	Accounting	205	1700

Инструкция DELETE

Инструкция DELETE позволяет удалить существующие строки из таблицы:

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

Удаление строк из таблицы

- Использование предложения `WHERE` позволяет удалить заданные строки:

```
DELETE FROM departments
WHERE  department_name = 'Finance';
```

```
1 rows deleted
```

- При отсутствии предложения `WHERE` удаляются все строки в таблице:

```
DELETE FROM  copy_emp;
```

```
22 rows deleted
```

Удаление строк на основе другой таблицы

Использование подзапросов в инструкциях `DELETE` для удаления строк в таблице на основе значений из другой таблицы:

```
DELETE FROM employees
WHERE  department_id =
                (SELECT department_id
                 FROM   departments
                 WHERE  department_name
                        LIKE '%Public%');
```

```
1 rows deleted
```

Инструкция TRUNCATE

- Удаляет все строки из таблицы, оставляя ее пустой и сохраняя структуру таблицы
- Является инструкцией языка определения данных (DDL), а не DML; практически не подлежит отмене
- Синтаксис:

```
TRUNCATE TABLE table_name;
```

- Пример:

```
TRUNCATE TABLE copy_emp;
```

План занятия

- Добавление новых строк в таблицу
  - инструкция `INSERT`
- Изменение данных в таблице
  - инструкция `UPDATE`
- Удаление строк из таблицы:
  - инструкция `DELETE`
  - инструкция `TRUNCATE`
- Управление транзакциями базы данных с помощью инструкций `COMMIT`, `ROLLBACK` и `SAVEPOINT`
- Целостность чтения
- Предложение `FOR UPDATE` в инструкции `SELECT`

Транзакции базы данных

Состав транзакции базы данных:

- инструкции DML, составляющие одно согласованное изменение данных
- одна инструкция DDL
- одна инструкция языка управления данными (DCL)

Транзакции базы данных: начало и завершение

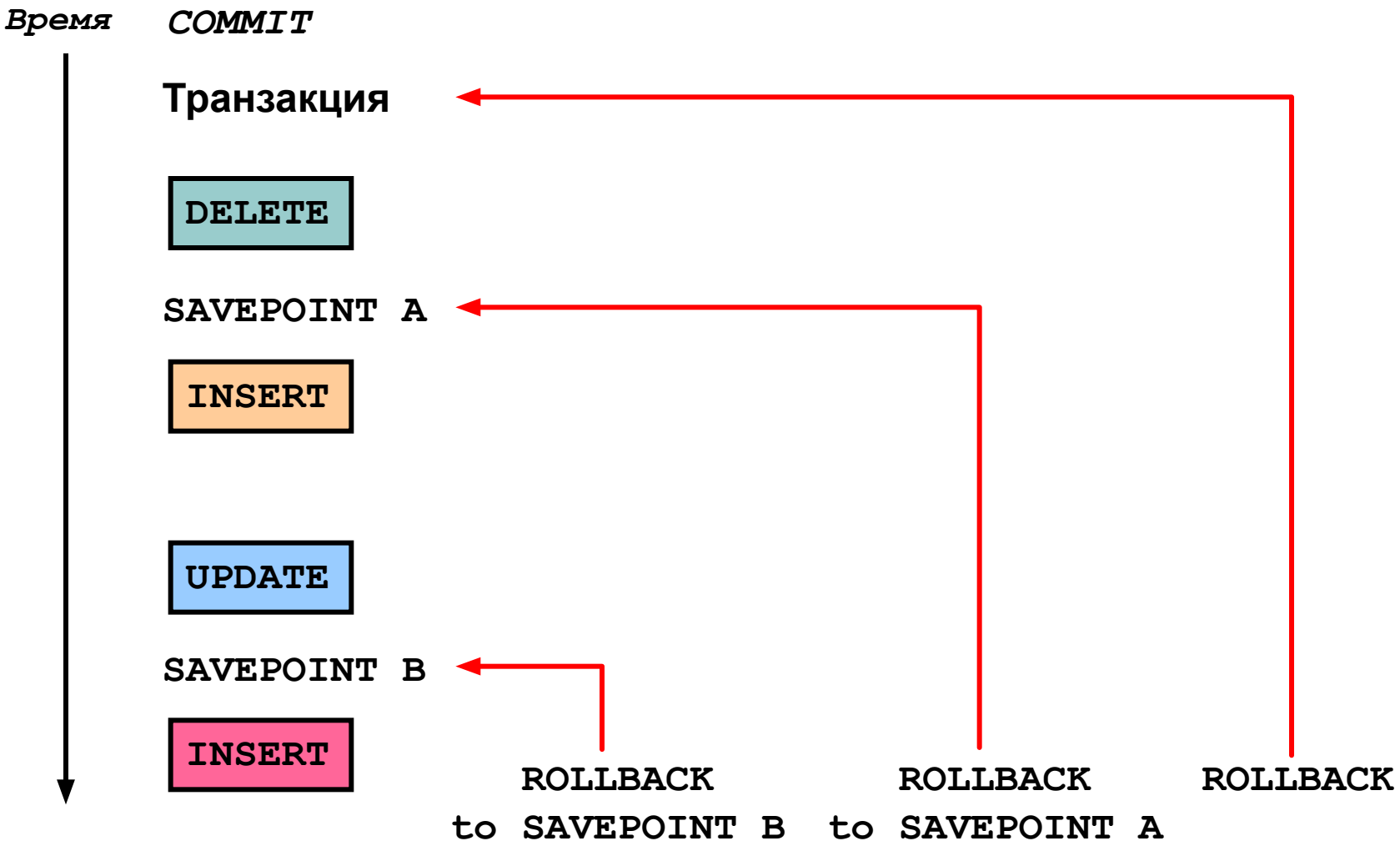
- Начинается при выполнении первой инструкции SQL DML.
- Завершается одним из следующих событий:
  - Запуск инструкции `COMMIT` или `ROLLBACK`.
  - Выполнение инструкции DDL или DCL (автоматическая фиксация).
  - Завершение пользователем работы SQL Developer или SQL*Plus.
  - Отказ системы.

Преимущества инструкций COMMIT и ROLLBACK

Инструкции COMMIT и ROLLBACK позволяют выполнять следующие задачи:

- обеспечивать согласованность данных
- просматривать изменения данных перед их сохранением
- группировать логически связанные операции

Явные инструкции управления транзакциями



Откат изменений к маркеру

- Создайте в текущей транзакции маркер с помощью инструкции `SAVEPOINT`.
- Выполните откат к этому маркеру с использованием инструкции `ROLLBACK TO SAVEPOINT`.

```
UPDATE...
SAVEPOINT update_done;
SAVEPOINT update_done succeeded.
INSERT...
ROLLBACK TO update_done;
ROLLBACK TO succeeded.
```

Обработка неявной транзакции

- Автоматическая фиксация происходит в следующих ситуациях:
  - запуск инструкции DDL
  - запуск инструкции DCL
  - обычное завершение работы SQL Developer или SQL*Plus без задания инструкций `COMMIT` или `ROLLBACK` явным образом
- Автоматический откат выполняется при аварийном завершении работы SQL Developer или SQL*Plus или системном сбое.

Состояние данных перед использованием инструкций COMMIT или ROLLBACK

- Возможно восстановление предшествующего состояния данных.
- Текущий пользователь может просматривать операции DML с помощью инструкции SELECT.
- Другие пользователи *не могут* просматривать результаты инструкций DML, отправленных текущим пользователем.
- Затронутые строки *блокируются*; другие пользователи не могут изменять данные в затронутых строках.

Состояние данных после фиксации

- Изменения данных сохраняются в базе данных.
- Предыдущее состояние данных перезаписывается.
- Все пользователи могут просматривать результаты.
- Затронутые строки разблокируются и становятся доступными другим пользователям для обработки.
- Все точки отката стираются.

Фиксация данных

- Внесите изменения:

```
DELETE FROM employees
WHERE  employee_id = 99999;
1 rows deleted

INSERT INTO departments
VALUES (290, 'Corporate Tax', NULL, 1700);
1 rows inserted
```

- Зафиксируйте изменения:

```
COMMIT;
COMMIT succeeded.
```

Состояние данных после отката

Отмена всех отложенных изменений с помощью инструкции отката `ROLLBACK`:

- Изменения данных отменяются.
- Восстанавливается предыдущее состояние данных.
- Затронутые строки разблокируются.

```
DELETE FROM copy_emp;
ROLLBACK ;
```

Пример состояния данных после отката

```
DELETE FROM test;
25000 rows deleted.

ROLLBACK;
Rollback complete.

DELETE FROM test WHERE  id = 100;
1 row deleted.

SELECT * FROM   test WHERE  id = 100;
No rows selected.

COMMIT;
Commit complete.
```

Откат на уровне инструкции

- При ошибке выполнения одной инструкции DML выполняется откат только этой инструкции.
- Сервер Oracle реализует неявную точку отката.
- Все прочие изменения сохраняются.
- Пользователь должен завершить транзакцию явным образом, выполнив инструкцию `COMMIT` или `ROLLBACK`.

План занятия

- Добавление новых строк в таблицу
  - инструкция `INSERT`
- Изменение данных в таблице
  - инструкция `UPDATE`
- Удаление строк из таблицы:
  - инструкция `DELETE`
  - инструкция `TRUNCATE`
- Управление транзакциями базы данных с помощью инструкций `COMMIT`, `ROLLBACK` и `SAVEPOINT`
- Целостность чтения
- Предложение `FOR UPDATE` в инструкции `SELECT`

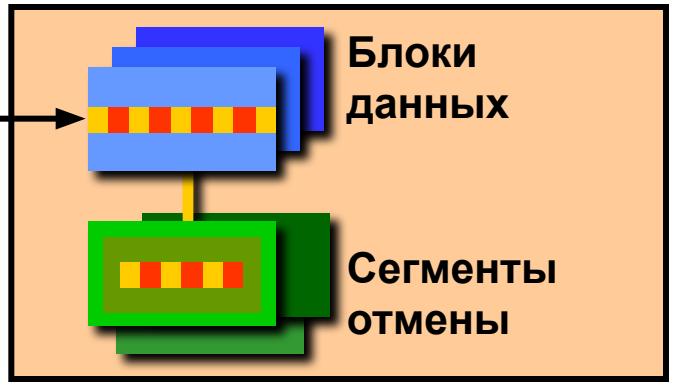
Целостность чтения

- Целостность чтения гарантирует постоянное согласованное представление данных.
- Изменения, выполненные разными пользователями, не должны конфликтовать.
- Целостность чтения гарантирует, что для одних и тех же данных:
  - операции считывания не ожидают завершения операций записи
  - операции записи не ожидают завершения операций считывания
  - одни операции записи ожидают завершения других

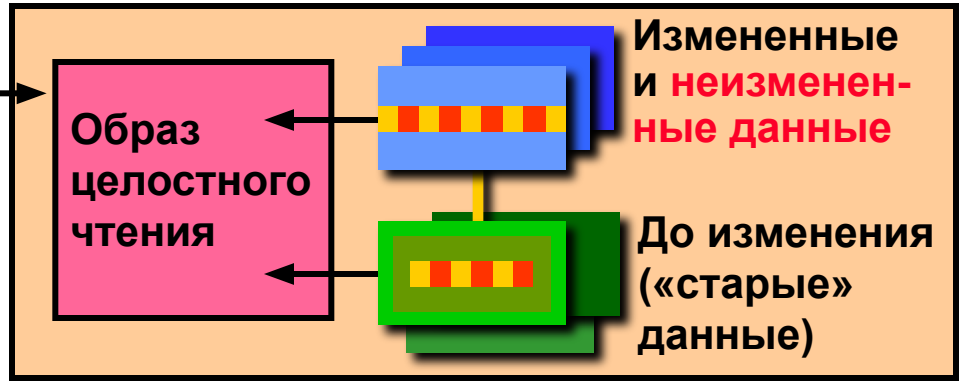
Реализация целостности чтения

Пользователь A

```
UPDATE employees
SET    salary = 7000
WHERE  last_name = 'Grant';
```



```
SELECT  *
FROM userA.employees;
```



Пользователь B

План занятия

- Добавление новых строк в таблицу
  - инструкция `INSERT`
- Изменение данных в таблице
  - инструкция `UPDATE`
- Удаление строк из таблицы:
  - инструкция `DELETE`
  - инструкция `TRUNCATE`
- Управление транзакциями базы данных с помощью инструкций `COMMIT`, `ROLLBACK` и `SAVEPOINT`
- Целостность чтения
- Предложение `FOR UPDATE` в инструкции `SELECT`

Предложение FOR UPDATE в инструкции SELECT

- Блокирует строки в таблице EMPLOYEES, в которых job_id имеет значение SA_REP.

```
SELECT employee_id, salary, commission_pct, job_id
FROM employees
WHERE job_id = 'SA_REP'
FOR UPDATE
ORDER BY employee_id;
```

- Разблокирование происходит только после отправки инструкций ROLLBACK или COMMIT.
- Если инструкция SELECT пытается заблокировать строку, уже заблокированную другим пользователем, база данных ожидает разблокирования строки и затем возвращает результаты инструкции SELECT.

Примеры предложения FOR UPDATE

- Предложение FOR UPDATE в инструкции SELECT можно использовать для нескольких таблиц.

```
SELECT e.employee_id, e.salary, e.commission_pct
FROM employees e JOIN departments d
USING (department_id)
WHERE job_id = 'ST_CLERK'
AND location_id = 1500
FOR UPDATE
ORDER BY e.employee_id;
```

- Блокируются строки в обеих таблицах EMPLOYEES и DEPARTMENTS.
- При использовании предложения FOR UPDATE OF *column_name* для определения столбца, который требуется изменить, блокируются только строки из указанной таблицы.

Заключение

На этом занятии были изучены следующие темы, касающиеся использования инструкций:

Функция	Описание
INSERT	Добавление новой строки в таблицу
UPDATE	Изменение существующих строк в таблице
DELETE	Удаление существующих строк из таблицы
TRUNCATE	Удаление всех строк из таблицы
COMMIT	Сохранение всех отложенных изменений
SAVEPOINT	Откат к маркеру точки сохранения
ROLLBACK	Отмена всех отложенных изменений данных
FOR UPDATE в инструкции SELECT	Блокировка строк, указанных в запросе SELECT

Упражнение 9: обзор

Это упражнение охватывает следующие темы:

- вставка строк в таблицы
- обновление и удаление строк в таблице
- управление транзакциями