

Создание объектов в Oracle

DDL – *Data Definition Language* – язык определения данных.

Команды:

create – создать,

drop – удалить,

alter – изменить,

grant – добавить привилегию,

revoke – убрать привилегию.

Создание таблиц в Oracle

CREATE TABLE имя_таблицы

(имя_столбца1 тип_данных (размер) [дополнительные ограничения],
[имя_столбца2 тип_данных (размер) [дополнительные ограничения],
имя_столбцаN тип_данных (размер) [дополнительные ограничения]],
[***CONSTRAINT*** имя_ограничения вид_ограничения])

Синтаксис для внешних и первичного ключей

CONSTRAINT pk_name **PRIMARY KEY** (имя_столбца1, имя_столбца2),
CONSTRAINT fk_name **FOREIGN KEY** (имя_столбца в данной таблице)
REFERENCES имя_таблицы на которую ссылаемся)

Основные ограничения

PRIMARY KEY (имя_столбца) – определение первичного ключа. Данное ограничение в рамках одной таблицы должно быть единственным и не допускает в столбце значений NULL;

FOREIGN KEY (имя_столбца) – определение внешнего ключа.

UNIQUE (имя_столбца1, имя_столбца2) – уникальные значения данных в указанных столбцах. При этом столбец может содержать значения NULL. Такого вида ограничений может быть несколько в рамках определения одной таблицы;

CHECK (условие) – проверка указанного условия;

NOT NULL – указание того, что столбец не должен содержать NULL – значений.

Пример создания таблицы

```
CREATE TABLE emp
(emp_number number not null,
emp_name varchar2 (50) not null,
department_id number,
email varchar2 (20) ,
salary number (6) check (salary >0),
CONSTRAINT emp_pk PRIMARY KEY (emp_number),
CONSTRAINT emp_un unique (email),
CONSTRAINT emp_fk FOREIGN KEY (department_id)
REFERENCES departments (department_id),
CONSTRAINT ch_up CHECK (email = UPPER(email)));
```

!!! таблица DEPARTMENTS должна быть создана ранее!!!

Создание таблицы на основе запроса

```
CREATE TABLE EMP as  
SELECT * FROM HR.employees;
```

Ключи не создаются!!!

В созданную таблицу переносятся только ограничения
CHECK (включая **NOT NULL**).



Удаление объектов в Oracle

Для удаления объектов используется команда **DROP**.

Удаление таблицы

DROP TABLE *имя_таблицы*;

DROP TABLE emp;

Удаление возможно если значения атрибутов не используются для других таблиц!

Изменение объектов в Oracle

Для удаления объектов используется команда **ALTER**.

Добавление ограничения целостности

```
ALTER TABLE имя_таблицы  
ADD CONSTRAINT ...
```

Добавление столбца

```
ALTER TABLE table_name  
ADD ( column_1 column-definition,  
      column_2 column-definition,  
      ...  
      column_n column_definition );
```

Переименование таблицы

```
ALTER TABLE table_name  
RENAME TO new_table_name;
```

Изменение типа данных

```
ALTER TABLE table_name  
MODIFY column_name column_type;
```

Удаление столбца

```
ALTER TABLE table_name  
DROP COLUMN column_name;
```

Переименование столбца

```
ALTER TABLE table_name RENAME COLUMN old_name TO new_name;
```


Модификация данных в Oracle

DML – *Data Manipulation Language* – язык манипулирования данными.

Команды:

insert – вставить,

update – обновить,

delete – удалить,

merge – вставить ИЛИ обновить
(**UPSERT**).

Добавление данных в таблицу

Команда **INSERT**.

INSERT INTO имя_таблицы

(имя_столбца1, имя_столбца2, ..., имя_столбцаN)

VALUES (значение1, значение2, ..., значениеN).

```
INSERT INTO emp (emp_number, emp_name ,  
                department_id, email , salary)  
VALUES (100, 'DEN', 90, upper('asd@gmail.com'),1000);
```

```
INSERT INTO emp  
VALUES (100, 'DEN', 90, upper('asd@gmail.com'),1000);
```

Варианты использования команды INSERT

Для вставки избранных полей
(обязательных)

```
INSERT INTO emp (emp_number, emp_name)  
VALUES (100, 'DEN');
```

Вставка данных на основе запроса

```
INSERT INTO emp  
SELECT emp_number, emp_name, department_id,  
upper(email), 2500 salary FROM hr.employees WHERE  
Department_id IN (20, 50, 80);
```

Обновление данных в таблице

Команда **UPDATE**.

UPDATE *имя_таблицы*

SET *имя_столбца = новое_значение*

WHERE *условие;*

UPDATE *employees*

SET *salary = 21000*

WHERE *salary <= 20000;*

Условие **WHERE** может содержать подзапрос!!!

Удаление данных из таблицы

Команда **DELETE**.

DELETE FROM *имя_таблицы*
WHERE *условие*

DELETE FROM employees
WHERE emp_number=22304;

Удаление всех данных из таблицы

DELETE FROM employees;

Оператор MERGE

Оператор **MERGE** — DML-оператор вставки (**INSERT**)/обновления (**UPDATE**)/удаления (**DELETE**, начиная с **Oracle Database 10g**) данных при слиянии таблиц.

```
SELECT * FROM dept;
```

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

```
SELECT * FROM dept_online;
```

DEPTNO	DNAME	LOC
40	OPERATIONS	BOSTON
20	RESEARCH DEV	DALLAS
50	ENGINEERING	WEXFORD

Применение команды MERGE

```
MERGE INTO dept d
USING (SELECT deptno, dname, loc
      FROM dept_online) o
ON (d.deptno = o.deptno)
WHEN MATCHED THEN
  UPDATE SET d.dname = o.dname, d.loc = o.loc
WHEN NOT MATCHED THEN
  INSERT (d.deptno, d.dname, d.loc)
  VALUES (o.deptno, o.dname, o.loc);
```

```
SELECT * FROM dept;
```

DEPTNO DNAME

LOC

10	ACCOUNTING	NEW YORK
20	RESEARCH DEV	DALLAS
30	SALES	CHICAGO
50	ENGINEERING	WEXFORD
60	HELP DESK	PITTSBURGH
40	OPERATIONS	BOSTON

Транзакции в Oracle

Транзакция (*transaction*) – группа SQL-операторов, которые выполняются успешно или неуспешно как единое целое.

Транзакция является стандартным элементом реляционной базы данных и предназначена для обеспечения согласованности данных.

Транзакция начинается с первого SQL-оператора, который поступил после окончания предыдущей транзакции или после соединения с базой данных.

Завершение транзакции происходит с помощью операторов **COMMIT** (подтверждение) или **ROLLBACK** (откат/отмена).

Операторы DML требуют завершения транзакции!!!

Механизм согласованности транзакций в Oracle

время	Сеанс А (user 1)	Сеанс В (user 2)
	Insert into emp (emp_id) values (1000);	
		Select * from emp where emp_id=1000; <i>(не была извлечена ни одна строка)</i>
	Commit;	
		Select * from emp where emp_id=1000; <i>(извлечена одна строка)</i>

Оператор COMMIT

При выполнении оператора COMMIT транзакция оканчивается и:

1. Вся работа, проделанная этой транзакцией – сохраняется;
2. Другим сеансам становится доступна измененная информация;
3. Все блокировки, установленные этой транзакцией – снимаются.

Оператор ROLLBACK

При выполнении оператора ROLLBACK транзакция оканчивается и:

1. Вся работа, проделанная этой транзакцией – отменяется;
2. Все блокировки, установленные этой транзакцией – снимаются.

Если сеанс отключается от БД без окончания текущей транзакции оператором commit (rollback), то транзакция автоматически откатывается (отменяется).

Классический пример транзакции – банковская операция.

Оператор TRUNCATE

Оператор **TRUNCATE** предназначен для удаления всех данных из таблицы.

Команда **TRUNCATE** – необратима – **невозможно откатить изменения!!!**

TRUNCATE TABLE table_name;

Следовательно, не требует завершения транзакции!!!

Создание представлений VIEW

Представление – виртуальная таблица, которая создается на основе запроса к базовым таблицам БД, т.е. динамический результат одной или нескольких реляционных операций, которые выполняются над отношениями БД с целью получения нового отношения.

Простые представления создаются на основе запроса к одной таблице, сложные – к нескольким.

Синтаксис создания представлений

CREATE [OR REPLACE] [FORCE] VIEW

имя_представления

AS SELECT *текст_команды*

[WITH CHECK OPTION [CONSTRAINT *имя_ограничения*]

[WITH READ ONLY];

[OR REPLACE] – необязательный параметр, который позволяет изменить уже существующее представление;

[FORCE] – необязательный параметр, при указании которого представление будет создано даже если базовая таблица не существует или у созданного представления нет прав доступа к данным базовой таблицы;

[WITH CHECK OPTION] – необязательный параметр ограничивает операции INSERT и UPDATE, выполненные через представление, чтобы не дать этим операциям создать строки, которые само представление не может выбрать.

Ограничения на обновляемые представления

Для простых представлений возможно выполнения операций DML при условии:

1. представление должно включать первичный ключ таблицы;
2. не должно содержать полей, полученных в результате применения групповых функций;
3. не должно содержать distinct, having, group by
4. может быть определено на другом представлении, но оно должно быть обновляемым.
5. не может содержать литералы или выражения.

Пример создания представления

```
CREATE OR REPLACE VIEW emp_department  
AS SELECT DISTINCT department_name,  
ROUND (AVG (salary)) AS avg_salary  
FROM hr.employees, hr.departments  
WHERE  
employees.department_id = departments.department_id  
GROUP BY department_name  
ORDER BY department_name ASC;
```

Представления можно использовать для запросов как обычную таблицу БД.

```
SELECT * FROM emp_department;
```

Удаления объекта БД – представления

```
DROP view emp_department;
```


Создание синонимов для объектов БД

Синоним – альтернативное имя объекта БД.

Создание синонимов не дает дополнительных привилегий на объект!!!

CREATE SYNONYM *имя_синонима* **FOR** *ссылка*;

где «ссылка» имеет вид

[схема.]таблица[@ имя_БД]

Создание последовательностей

Последовательность (sequence) – это объект базы данных, который может быть использован многими пользователями для генерации уникальных целых чисел. Обычно последовательности используются для генерации значений первичного ключа.

Числа, создаваемые последовательностью, могут:

- возрастать постоянно;
- возрастать до определенного предела.

Синтаксис создания последовательностей

CREATE SEQUENCE [схема.]имя_последовательности
[**INCREMENT BY** приращение]
[**START WITH** начальное_значение]
[**MAXVALUE** наибольшее_значение | **NOMAXVALUE**]
[**MINVALUE** наименьшее_значение | **NOMINVALUE**]
[**CYCLE** | **NOCYCLE**]
[**CACHE** число_элементов | **NOCACHE**]
[**ORDER** | **NOORDER**]

Элементы последовательностей

INCREMENT BY определяет интервал между последовательными номерами. Если этот параметр имеет отрицательное значение, то последовательность убывающая, если положительное — то последовательность возрастающая. Значением по умолчанию является 1, параметр приращения может быть любым целым

START WITH параметром начальное_значение задает первый генерируемый номер. Если этот параметр отсутствует, то для возрастающих последовательностей первый генерируемый номер равен значению параметра MINVALUE, а для убывающей последовательности MAXVALUE.

Элементы последовательностей

MAXVALUE параметром *наибольшее_значение* задает максимальное значение, которое будет генерироваться последовательностью.

MINVALUE параметром *наименьшее_значение* задает минимальное число, которое будет генерироваться последовательностью.

NOCYCLE является значением, используемым по умолчанию, и означает завершение генерирования последовательности по достижении конца последовательности. Если при определении последовательности указан параметр **CYCLE**, то после достижения очередным членом последовательности значения *наибольшее_значение* (для возрастающих последовательностей) выдается значение параметра *наименьшее_значение*.

Элементы последовательностей

CACHE N – указывает, сколько значений последовательности ORACLE распределяет заранее и поддерживает в памяти для быстрого доступа. Минимальное значение этого параметра равно 2. Если кэширование нежелательно или не установлено явным образом, Oracle применяет значение по умолчанию – 20 значений.

ORDER — гарантирует, что номера последовательности генерируются в порядке запросов. Эта опция может использоваться, к примеру, когда номера последовательности предстают в качестве отметок времени. Гарантирование порядка обычно не существенно для тех последовательностей, которые используются для генерации первичных ключей. по умолчанию **NOORDER**.

Пример создания последовательностей

```
CREATE SEQUENCE sequence_2  
START WITH 20  
INCREMENT BY -1  
MAXVALUE 20  
MINVALUE 0  
CYCLE;
```

После создания последовательности к ней можно обращаться через псевдостолбцы **CURRVAL** (возвращает текущее значение последовательности) и **NEXTVAL** (выполняет приращение последовательности и возвращает ее следующее значение). Текущее и следующее значения последовательности пользователи базы данных получают, выполняя команду **SELECT**.

Значения CURRVAL и NEXTVAL

используются в:

в списке **SELECT** предложения **SELECT**;
в фразе **VALUES** предложения **INSERT**;
в фразе **SET** предложения **UPDATE**.

Нельзя использовать значения **CURRVAL** и **NEXTVAL** в:

в подзапросе;
в предложении **SELECT** с оператором **DISTINCT**;
в предложении **SELECT** с фразой **GROUP BY** или **ORDER BY**;
в предложении **SELECT**, объединенном с другим предложением **SELECT** оператором множеств **UNION**;
в фразе **WHERE** предложения **SELECT**;
в умалчиваемом (**DEFAULT**) значении столбца в предложении **CREATE TABLE** или **ALTER TABLE**
в условии ограничения **CHECK**.

Особенности CURRVAL и NEXTVAL

Прежде чем обращаться к **CURRVAL** в текущем сеансе работы, необходимо хотя бы один раз выполнить обращение к **NEXTVAL**.

В одном предложении **SQL** приращение последовательности может быть выполнено только один раз. Если предложение содержит несколько обращений к **NEXTVAL** для одной и той же последовательности, то **ORACLE** наращивает последовательность один раз, и возвращает одно и то же значение для всех вхождений **NEXTVAL**.

Если предложение содержит обращения как к **CURRVAL**, так и к **NEXTVAL**, то **ORACLE** наращивает последовательность и возвращает одно и то же значение как для **CURRVAL**, так и для **NEXTVAL**, независимо от того, в каком порядке они встречаются в предложении.

Примеры работы с последовательностью

```
SELECT sequence_2.NEXTVAL FROM dual;
```

```
SELECT sequence_2.CURRVAL FROM dual;
```

```
INSERT INTO emp  
SELECT sequence_2.NEXTVAL, emp_name , department_id ,  
upper(email) , 2500 salary FROM hr. employees WHERE  
Department_id IN (20, 50, 80);
```

```
DROP SEQUENCE sequence_2;
```

Загрузка данных из внешних источников

Утилита SQL*Loader

SQL*Loader - это продукт для загрузки данных из внешних файлов в таблицы БД ORACLE. SQL*Loader загружает данные в различных форматах, выполняет фильтрацию (выборочную загрузку записей в зависимости от значений данных. Во время выполнения SQL*Loader формирует детальный файл отчета (Log File) со статистикой загрузки и может также создавать файл отвергнутых (Discard File) записей (записи, отвергнутые из-за ошибок в данных) и файл пропущенных записей (Bad File) (записи, которые не соответствуют критерию выбора).

Простой пример загрузки данных

1. Создайте файл с данными формата **.txt** или **.dat** или **.csv** (данные в файле должны быть разделены, например « ; »). Каждая строка в файле с данными будет загружена как отдельная строка в таблицу.
2. Создайте управляющий файл с разрешением **.ctl**. В нем будет содержаться информация о том, где находятся загружаемые данные, в какую таблицу их загружать и с какими параметрами.
3. Оба файла должны находиться в каталогах, названия которых не содержат русских букв.

Пример управляющего файла

LOAD DATA

INFILE 'D:\HP_D\LMchick\academy\SQL\test.csv'

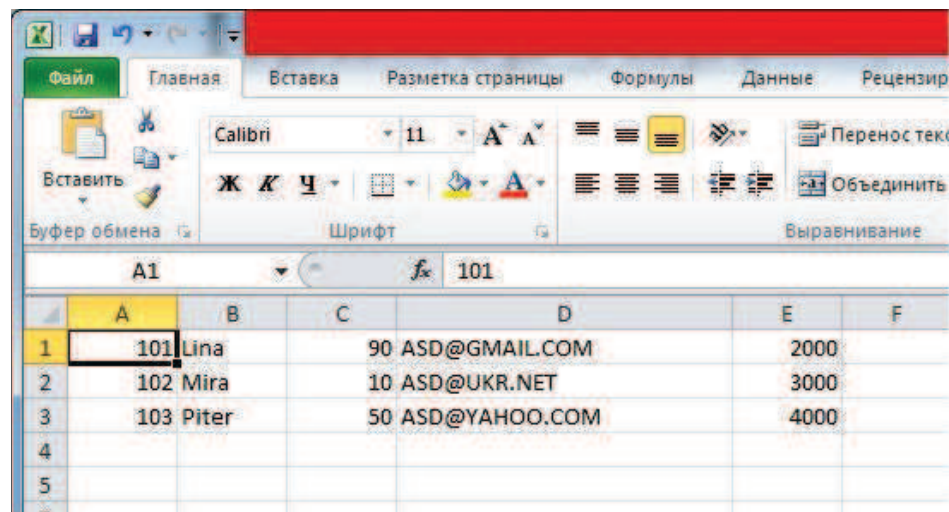
APPEND

INTO TABLE emp

FIELDS TERMINATED BY ";" OPTIONALLY ENCLOSED BY "'"
TRAILING NULLCOLS

(emp_number,emp_name ,department_id ,email,salary)

Пример файла данных

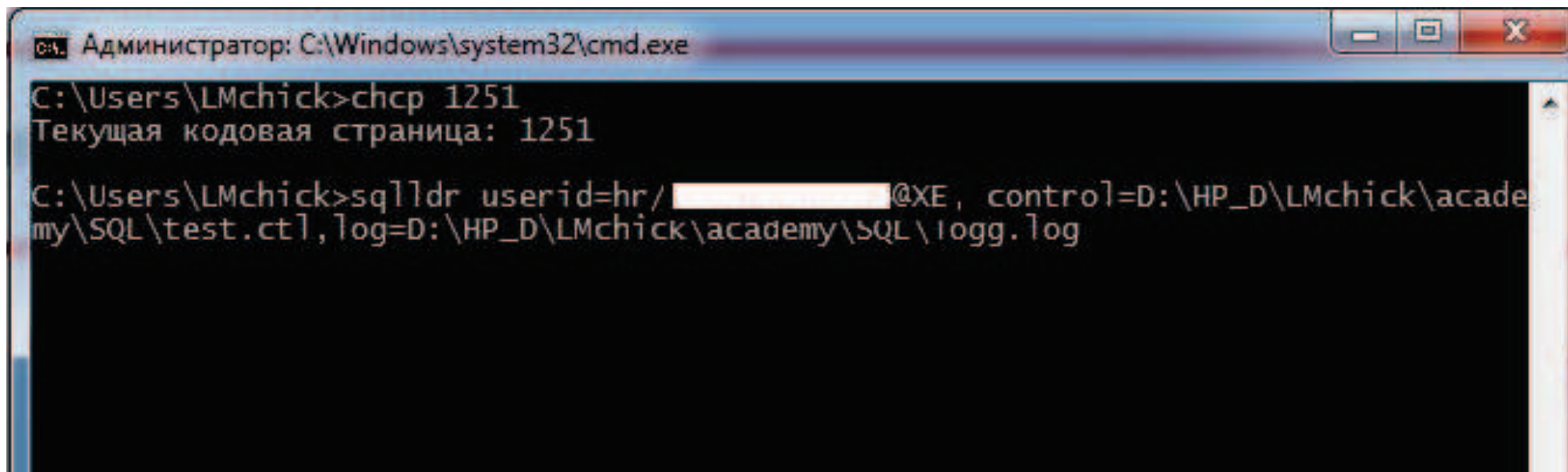


The screenshot shows a Microsoft Excel spreadsheet with a table containing employee data. The table has six columns labeled A through F. The data is as follows:

	A	B	C	D	E	F
1	101	Lina	90	ASD@GMAIL.COM	2000	
2	102	Mira	10	ASD@UKR.NET	3000	
3	103	Piter	50	ASD@YAHOO.COM	4000	
4						
5						

Запуск SQL*Loader

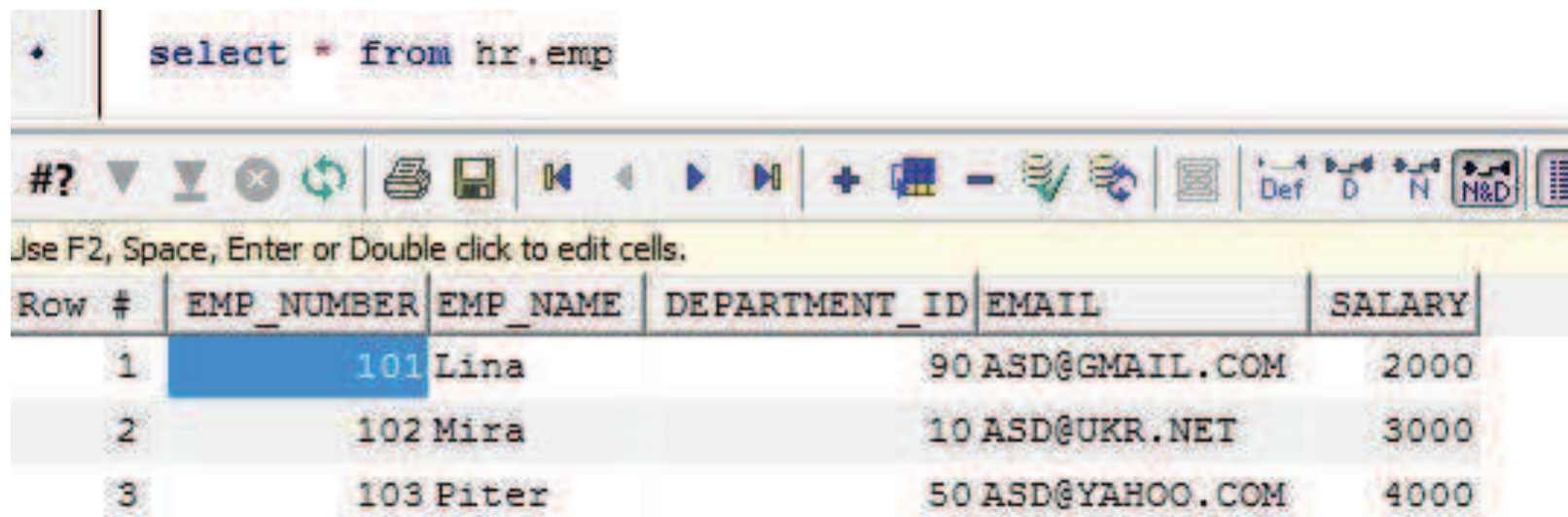
Запускаем из командной строки
операционной системы!!!



The screenshot shows a Windows command prompt window titled "Администратор: C:\Windows\system32\cmd.exe". The user is logged in as LMchick. The first command entered is `chcp 1251`, which sets the active code page to 1251. The second command is `sqlldr userid=hr/[REDACTED]@XE, control=D:\HP_D\LMchick\academy\SQL\test.ctl, log=D:\HP_D\LMchick\academy\SQL\logg.log`, which initiates the SQL*Loader process with specific user, control file, and log file parameters.

```
Администратор: C:\Windows\system32\cmd.exe
C:\Users\LMchick>chcp 1251
Текущая кодовая страница: 1251
C:\Users\LMchick>sqlldr userid=hr/[REDACTED]@XE, control=D:\HP_D\LMchick\academy\SQL\test.ctl, log=D:\HP_D\LMchick\academy\SQL\logg.log
```

Результаты загрузки



```
select * from hr.emp
```

Row #	EMP_NUMBER	EMP_NAME	DEPARTMENT_ID	EMAIL	SALARY
1	101	Lina	90	ASD@GMAIL.COM	2000
2	102	Mira	10	ASD@UKR.NET	3000
3	103	Piter	50	ASD@YAHOO.COM	4000

При возникновении ошибок необходимо просмотреть
log –файл!!!

Дополнительную информацию по использованию
SQL*Loader вы найдете в файле
SQL_Loader_Help.docx