Лабораторная работа №3

Часть 1. Реализация SQL-запросов для создания схемы базы данных

В лабораторной работе выполняется реализация схемы базы данных по ранее построенной реляционной схеме данных (см. лабораторную работу №2). Требуется сформировать SQL-запросы для создания таблиц базы данных и выполнить их в СУБД. Требуется заполнить таблицы данными с помощью оператора `INSERT`.

Порядок выполнения работы

1) Создать в СУБД новую схему данных для хранения пользовательских объектов (см. часть 2).
2) В этой новой схеме данных с помощью скрипта с запросами на языке DDL SQL реализовать таблицы, соответствующие реляционным отношениям схемы данных полученной в лабораторной работе №2, с помощью одного (желательно) оператора `CREATE TABLE` для каждой таблицы в следующем порядке:

- ~~реализовать простую структуру таблиц, включающую только набор столбцов с добавлением описаний первичного ключа;~~
- ~~дополнить описание таблицы реализацией ограничений для описания внешних ключей; для внешних ключей установить свойства контроля целостности данных (каскадное удаление и обновление), если это возможно в целевой СУБД;~~
- ~~дополнить описание таблицы реализацией ограничений для описания бизнес-правил;~~
- ~~дополнить описание таблицы реализацией комментариев для значимых элементов таблицы.~~

~~3) Заполнить с помощью SQL-скрипта с использованием оператора `INSERT` таблицы строками данных для проверки правильного выбора первичных ключей и работоспособности ссылок между таблицами:~~

- ~~строками данных сначала заполнять мастер-таблицы (или таблицы, которые НЕ ссылаются на другие таблицы);~~
- ~~в каждую таблицу добавить 30 строк осмысленных данных;~~
- если не удается добавить данные в таблицу по причине нарушения уникальности первичного ключа, то следует перепроверить описание этого первичного ключа и его смысл для реального мира;
- если не удается добавить данные в таблицу по причине нарушения ссылочной целостности, то следует убедиться, что целевые данные существуют, иначе перепроверить описание внешнего ключа.

~~4) Рассмотреть простые действия по изменению структуры таблицы (переименование столбца таблицы, добавление и удаление ограничений на столбец таблицы или всю таблицу) и реализовать их с помощью оператора~~

~~ALTER TABLE.~~

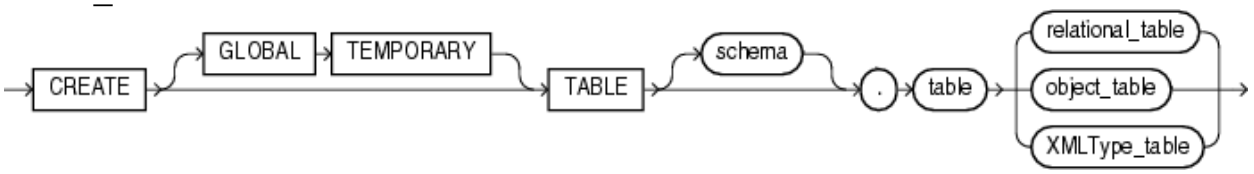
~~5) Создать временную таблицу с помощью оператора CREATE TABLE и удалить ее с помощью оператора DROP TABLE.~~

6) Экспортировать результаты работы в SQL-скрипт (см. часть 2), сравнить полученный скрипт со скриптами, созданными на этапах 2 и 3.

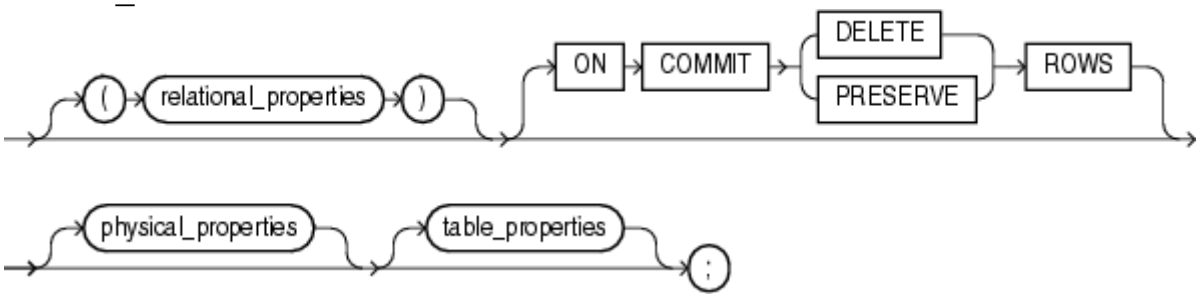
Работа с таблицами в SQL

Создание структуры базовой таблицы выполняется с помощью оператора CREATE TABLE, синтаксическая конструкция которого в виде значимых частей по формированию реляционных таблиц приведена ниже.

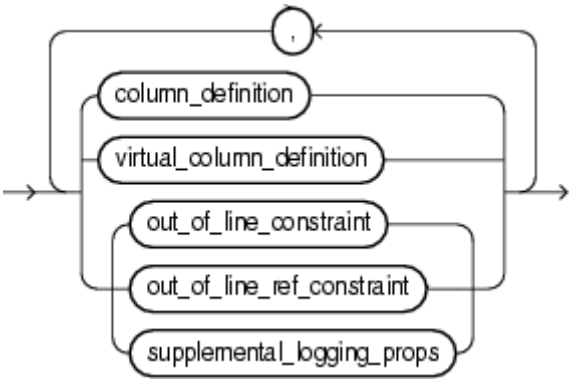
create_table::=



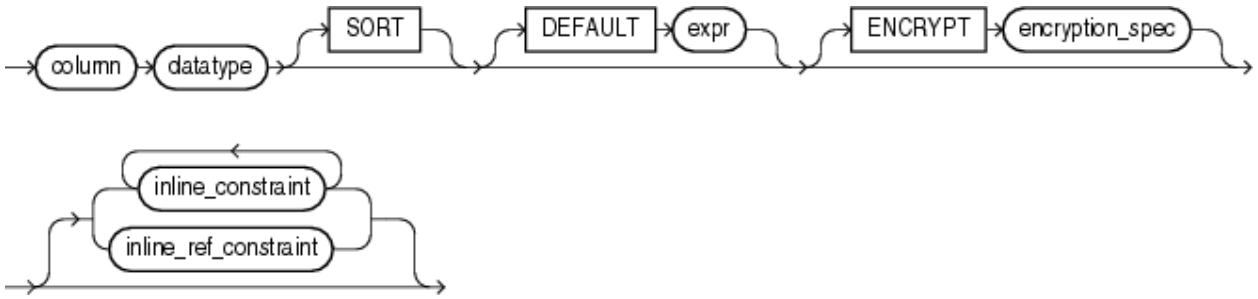
relational_table::=



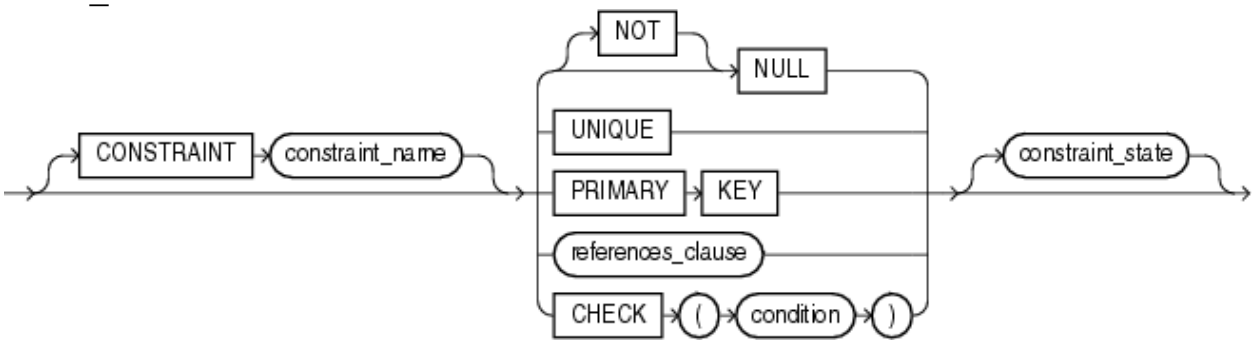
relational_properties::=



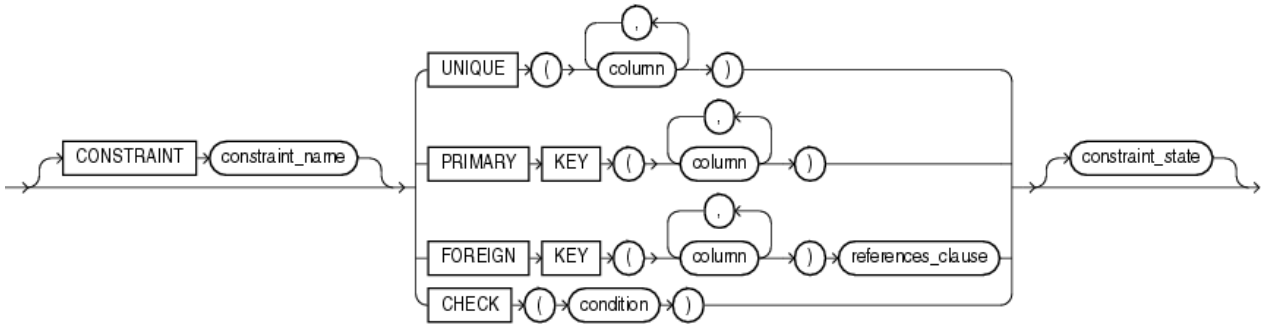
column_definition::=



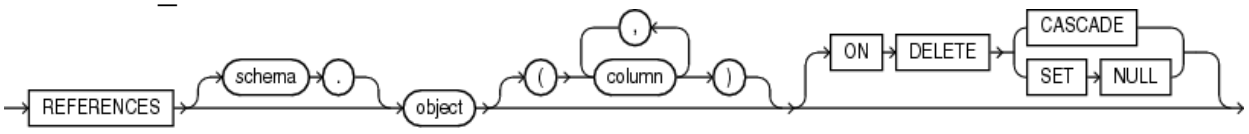
inline_constraint::=



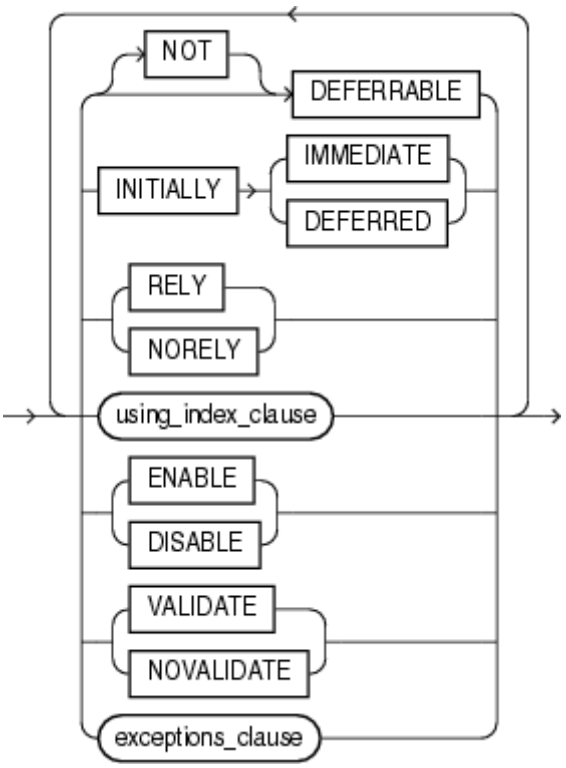
out_of_line_constraint::=



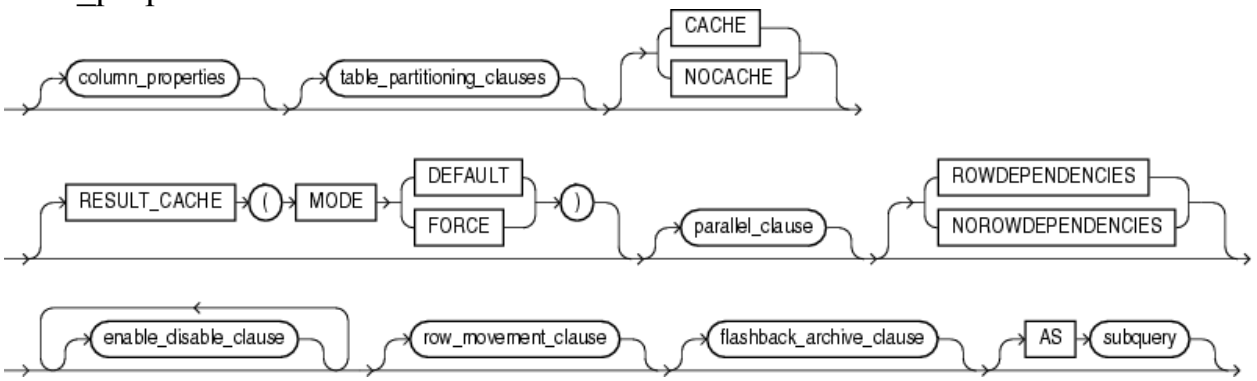
references_clause::=



constraint_state::=



table_properties::=

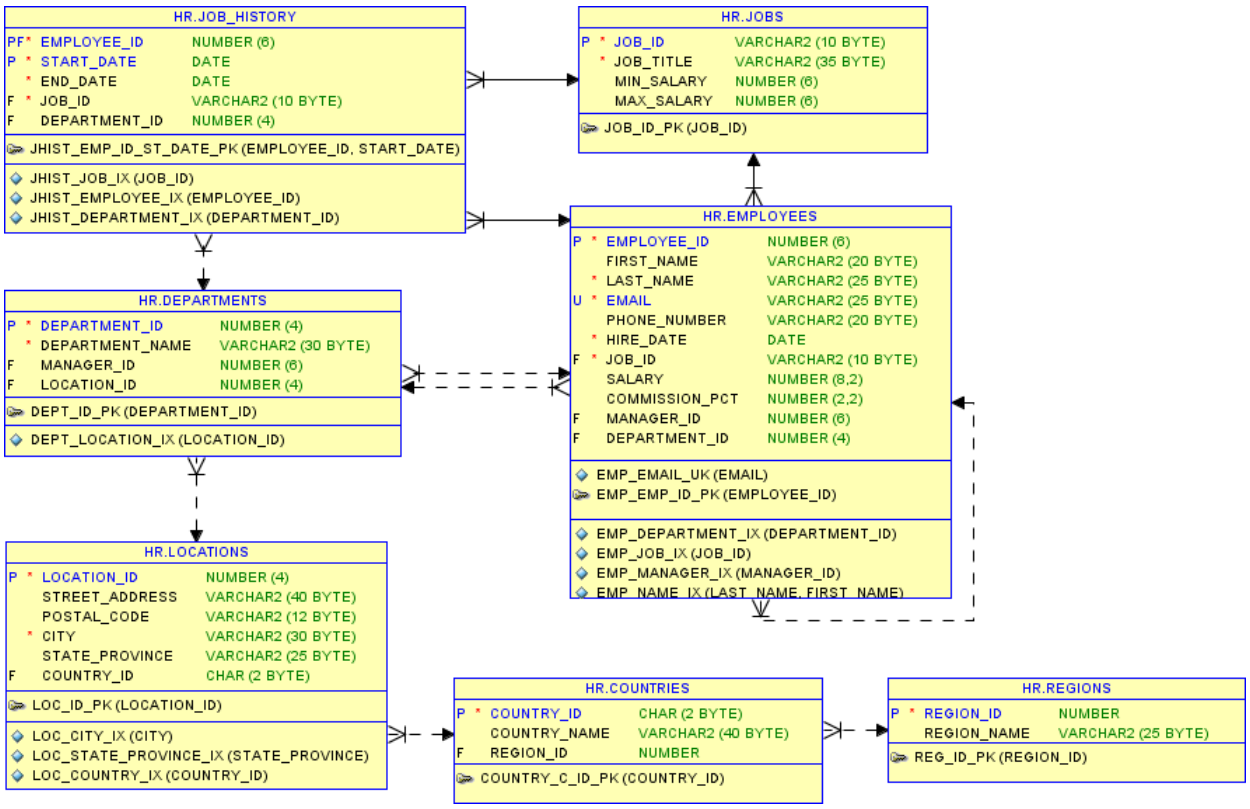


В операторе `CREATE TABLE` предложение `CONSTRAINT` позволяет задать возможные определения целостности при описании структуры таблицы:

- *constraint_name* – уникальное имя ограничения в схеме, в случае когда это имя явно не задано, оно будет сгенерировано автоматически с префиксом `SYS_C`;
- `NOT NULL` – запрет неопределенных значений в столбце (может быть задано только *inline*);
- `UNIQUE` – поддержка уникальных комбинаций значений (до 32 столбцов, значения `NULL` в столбце допустимы) – фактически описание потенциального ключа;
- `PRIMARY KEY` – указание первичного ключа (до 32 столбцов, `NULL` значения не допустимы во всех столбцах первичного ключа);
- `FOREIGN KEY` – указание внешнего ключа;

- `CHECK` – задание бизнес-правила по некоторому условию;
- по умолчанию ссылка устанавливается на первичный ключ объекта *object*;
- список столбцов, на который ссылаемся должен соответствовать по числу и типам списку столбцов ссылок;
- `ON DELETE CASCADE` – при удалении строки ссылки будет удаляться и сама ссылка;
- `ON DELETE SET NULL` – при удалении строки ссылки, ссылка будет установлена как неопределенная (`NULL`);
- `DEFERRABLE` – проверка ограничения может быть отложена до конца транзакции с помощью оператора `SET CONSTRAINTS`;
- `ENABLE VALIDATE` – все данные соответствуют ограничениям;
- `ENABLE NOVALIDATE` – только новые данные соответствуют ограничениям;
- `DISABLE VALIDATE` – только новые данные не соответствуют ограничениям;
- `DISABLE NOVALIDATE` – ограничения полностью отключены (но не удалены – могут вызвать отказ в операции удаления объекта и т.п.).

Все приводимые примеры базируются на готовой учебной схеме данных HR (СУБД ORACLE 11g):

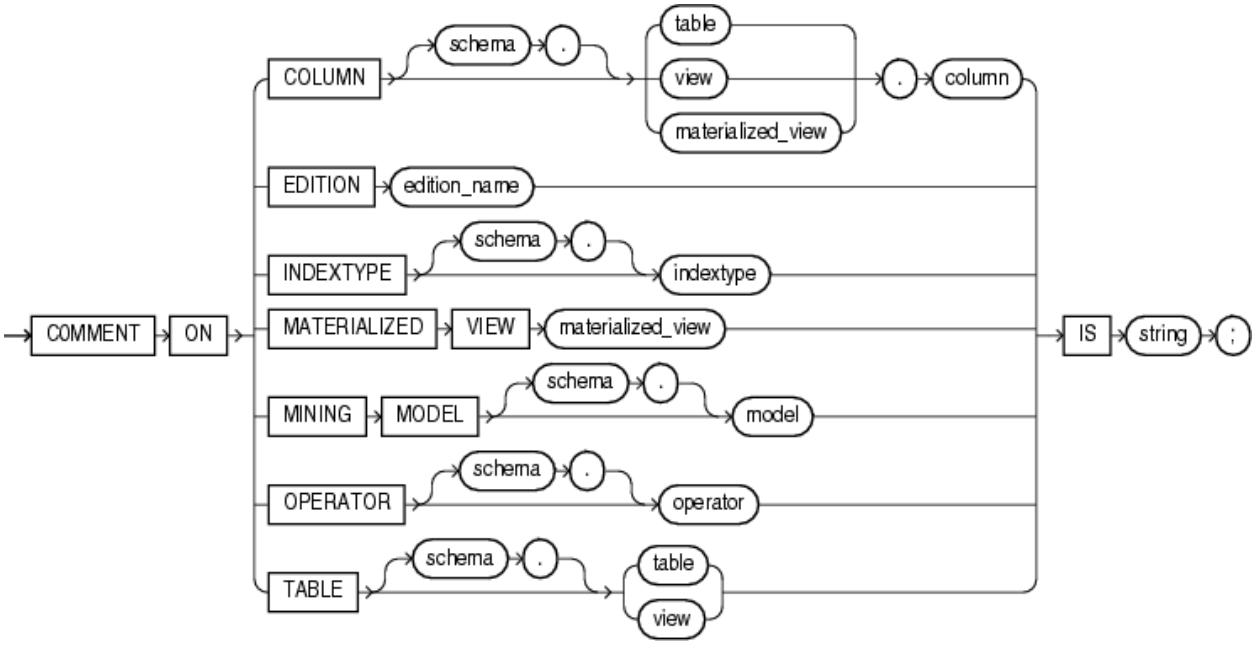


Ниже приведенный оператор `CREATE TABLE` выполняет создание

новой таблицы с именем my_employees в схеме данных HR, причем создаваемая таблица по своей структуре практически аналогична таблице HR.EMPLOYEES:

```
CREATE TABLE my_employees
( employee_id    NUMBER(6),
  first_name     VARCHAR2(20),
  last_name      VARCHAR2(25)
     CONSTRAINT my_emp_last_name_nn NOT
  NULL, email    VARCHAR2(25)
     CONSTRAINT my_emp_email_nn NOT NULL,
  hire_date      DATE
     CONSTRAINT my_emp_hire_date_nn NOT
  NULL, job_id   VARCHAR2(10)
     CONSTRAINT my_emp_job_nn NOT NULL,
  salary         NUMBER(8,2),
  manager_id     NUMBER(6),
  department_id NUMBER(4),
CONSTRAINT my_emp_salary_min CHECK (salary > 0),
  CONSTRAINT my_emp_email_uk UNIQUE (email),
  CONSTRAINT my_emp_emp_id_pk PRIMARY KEY (employee_id),
  CONSTRAINT my_emp_dept_fk FOREIGN KEY (department_id)
     REFERENCES departments,
  CONSTRAINT my_emp_job_fk FOREIGN KEY (job_id)
     REFERENCES jobs (job_id),
  CONSTRAINT my_emp_manager_fk FOREIGN KEY (manager_id)
     REFERENCES my_employees
);
```

Для описания таблиц и столбцов используется оператор COMMENT:
comment::=

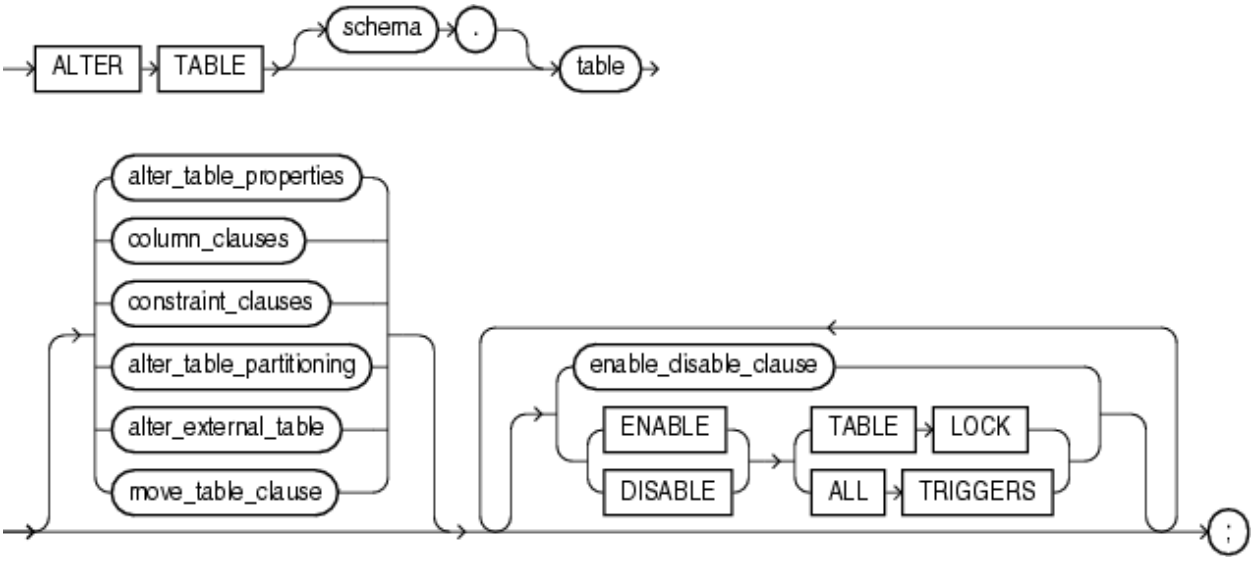


Ниже приведены примеры оператора COMMENT:

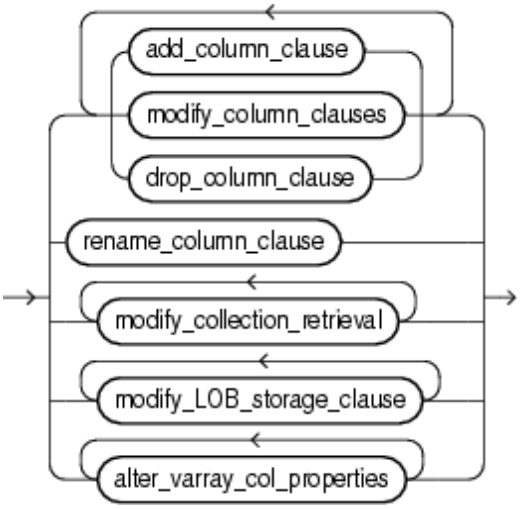
```
Worksheet  Query Builder
-- add comments
COMMENT ON TABLE my_employees IS 'Employees data';
COMMENT ON COLUMN my_employees.employee_id IS 'Employee''s unique number. Primary key column';
COMMENT ON COLUMN my_employees.first_name IS 'Employee''s first name';
COMMENT ON COLUMN my_employees.last_name IS 'Employee''s last name';

Script Output
Task completed in 0.033 seconds
COMMENT on table my_employees 'EMPLOYEES DATA' succeeded.
COMMENT on column my_employees.employee_id 'EMPLOYEE''S UNIQUE NUMBER. PRIMARY KEY COLUMN' succeeded.
COMMENT on column my_employees.first_name 'EMPLOYEE''S FIRST NAME' succeeded.
COMMENT on column my_employees.last_name 'EMPLOYEE''S LAST NAME' succeeded.
```

Изменение структуры таблицы выполняется оператором `ALTER TABLE`.

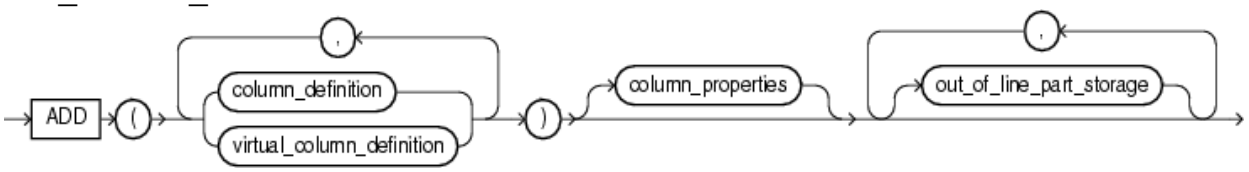
alter_table::=



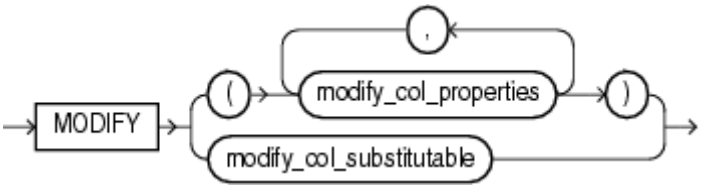
column_clauses::=



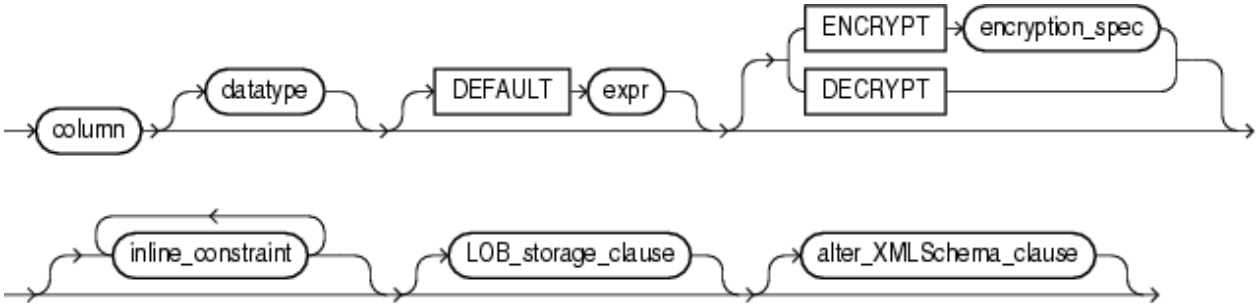
add_column_clause ::=



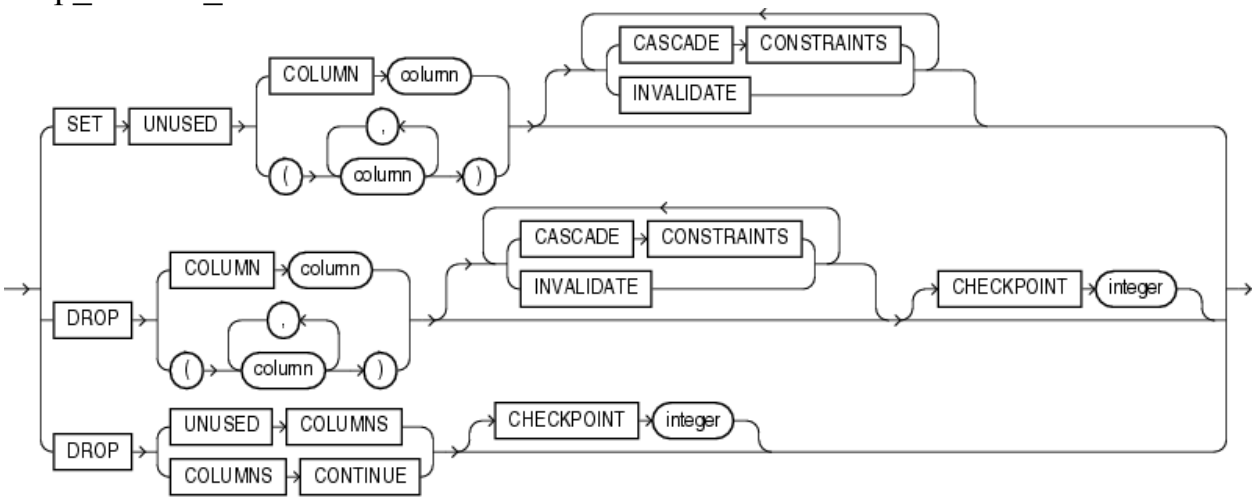
modify_column_clauses::=



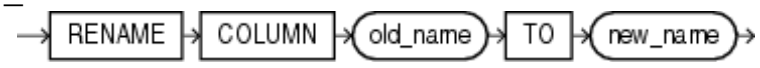
modify_col_properties::=



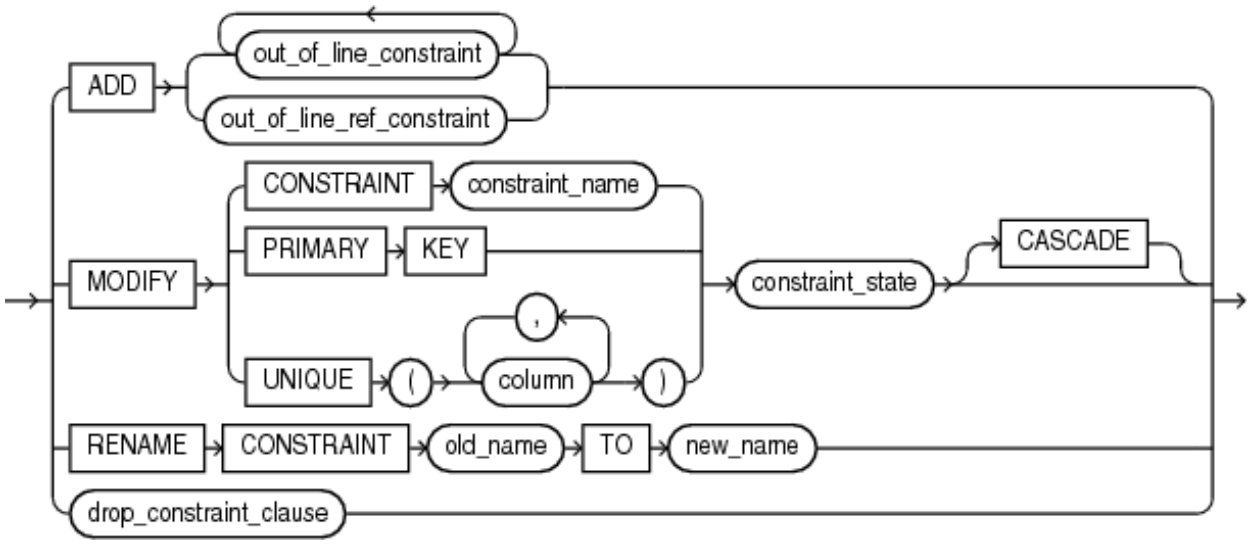
drop_column_clause ::=



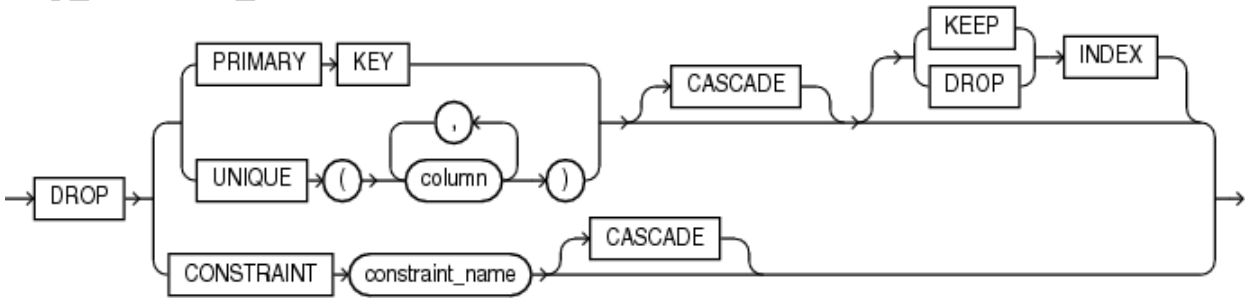
rename_column_clause ::=



constraint_clauses::=



drop_constraint_clause::=



Ниже приведены примеры оператора `ALTER TABLE`:

```
ALTER TABLE my_employees
ADD (commission_pct NUMBER(2,2) CHECK (commission_pct < 0.5),
     phone_number VARCHAR2(20));

ALTER TABLE my_employees
MODIFY (salary DEFAULT 1000);

ALTER TABLE my_employees
DROP (email);

ALTER TABLE my_employees
RENAME COLUMN hire_date TO hire;
```

```
ALTER TABLE my_employees
ADD CONSTRAINT check_commission
    CHECK (salary + (commission_pct*salary) <= 5000);

ALTER TABLE my_employees
MODIFY PRIMARY KEY DISABLE CASCADE;

ALTER TABLE my_employees
RENAME CONSTRAINT my_emp_salary_min TO my_emp_salary;

ALTER TABLE my_employees
DROP CONSTRAINT my_emp_job_fk;
```

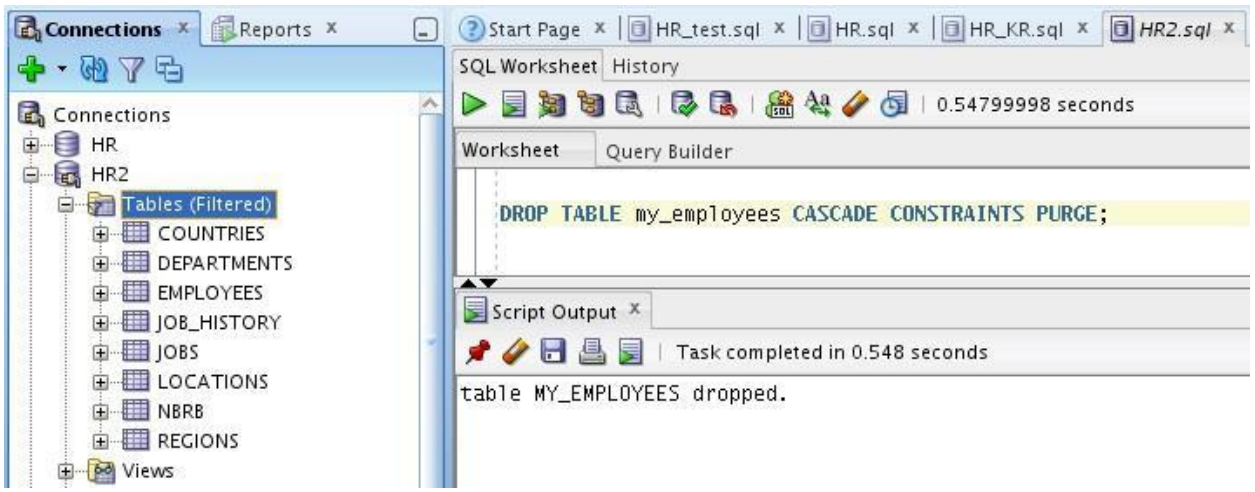
Удаление структуры таблицы выполняется оператором `DROP TABLE`:

drop_table::=

Для удаления можно использовать опции:

- `CASCADE CONSTRAINTS` – для каскадного удаления ограничений, которые описывают ссылки на первичный или потенциальный ключ удаляемой таблицы *table*.
- `PURGE` – непосредственное освобождение дискового пространства, связанного с удаляемой таблицей.

Ниже приведен пример удаления таблицы:



Оператор `INSERT`

Оператор `INSERT` предназначен для добавления (вставки) новых строк данных в таблицу.

insert::=

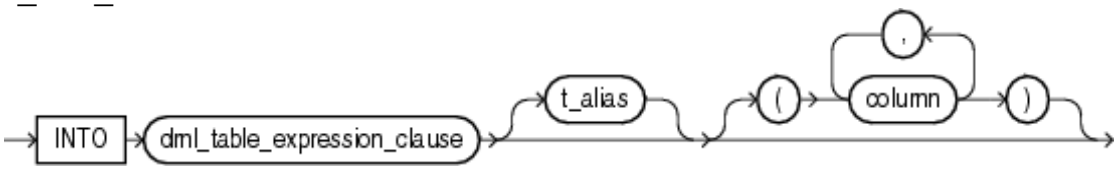
Оператор существует в двух вариантах:

- вставка данных только в одну таблицу (вставка одной строки или вставка нескольких строк данных);
- вставка данных в несколько таблиц (безусловная вставка или условная вставка – в этой лабораторной работе нам не понадобится).

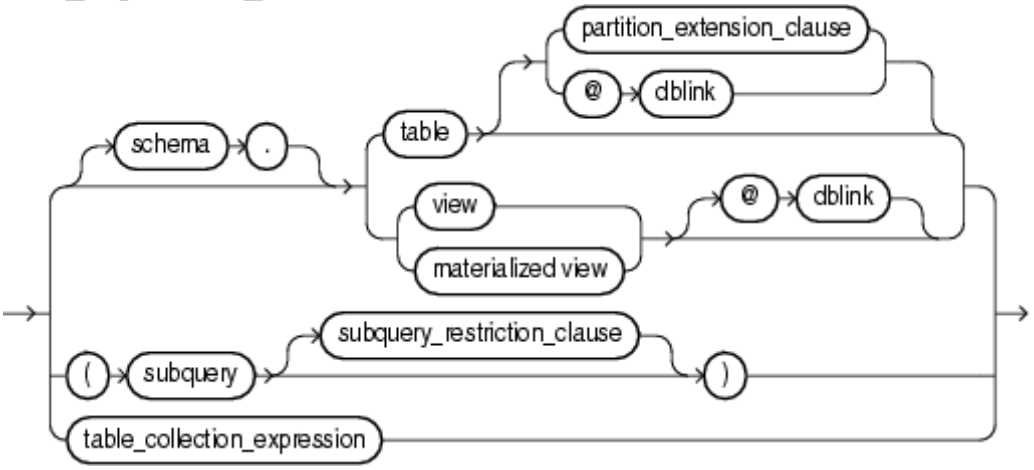
Для вставки данных только в одну таблицу применим ниже приведенный синтаксис элементов оператора `INSERT`:

single_table_insert ::=

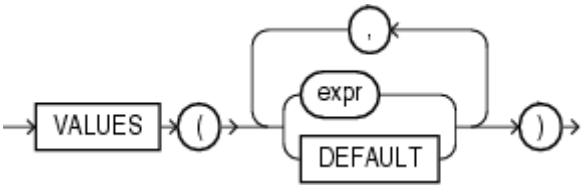
insert_into_clause ::=



DML_table_expression_clause::=



values_clause ::=



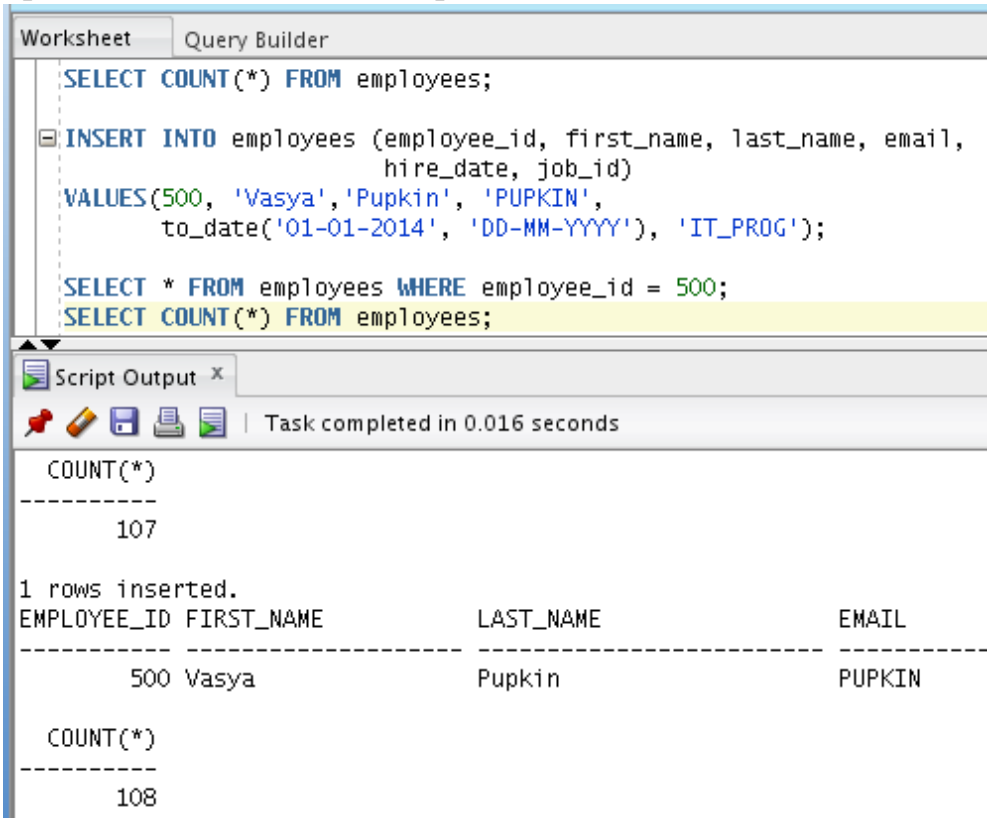
Особенности оператора `INSERT` для вставки данных только в одну таблицу:

- `INSERT INTO` – предложение для указания приемника данных, в который будет производиться вставка строк;
- приемник данных – обычно базовая таблица, но также может применяться обновляемое представление или подзапрос, допускающий обновление;
- *t_alias* – псевдоним таблицы для ссылок на нее при работе (например, в

коррелированном подзапросе);

- *column* – список столбцов данных приемника, которым будут назначены конкретные значения при вставке (если список не указан, то будут использованы все столбцы таблицы); столбцы отсутствующие в этом списке получат значения автоматически (по умолчанию, `null` и т.п.), если это возможно для конкретного столбца приемника;
- `VALUES` – указание списка данных для вставки только одной строки (внешние данные); этот список по числу параметров и их типам данных должен соответствовать заданному списку столбцов приемника;
- *subquery* – строки данных можно сформировать также с помощью подзапроса из других объектов базы данных (внутренние данные), при этом требуется соблюдение соответствия списка столбцов приемника и списка столбцов, возвращаемых подзапросом, как по числу параметров, так и по их типам – в данной лабораторной работе этот фрагмент также вряд ли будет применим.

Пример вставки только одной строки:



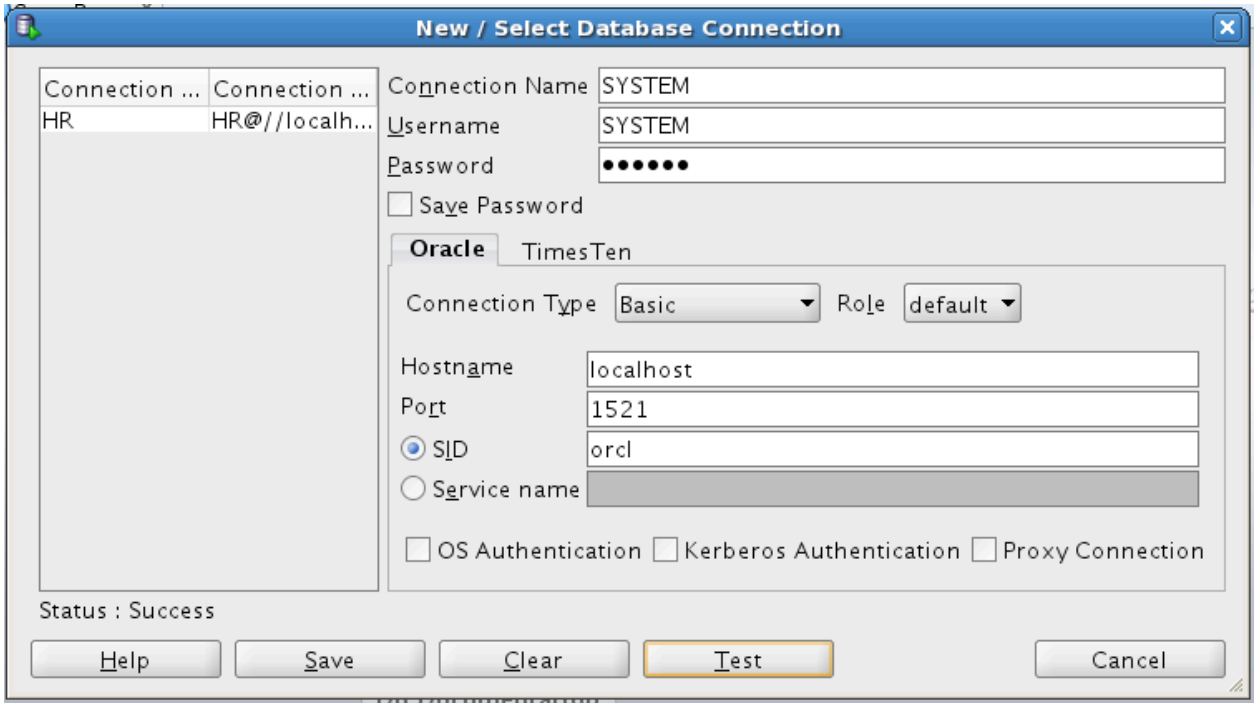
Часть 2. Использование Oracle SQL Developer

В этой части приводятся рекомендации по использованию Oracle SQL Developer для решения ряда задач данной лабораторной работы.

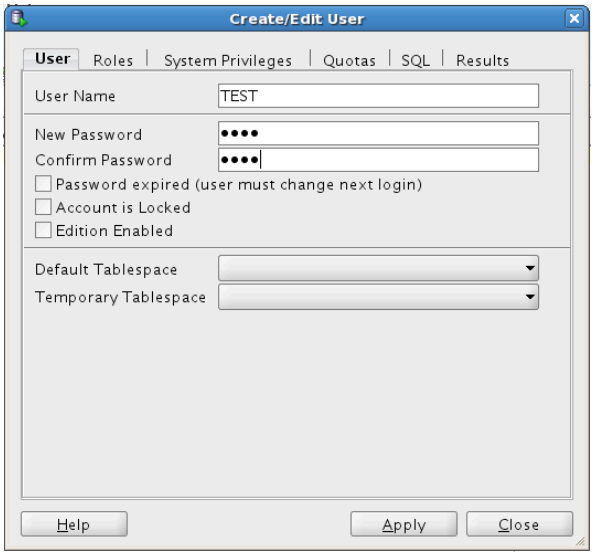
Порядок выполнения работы

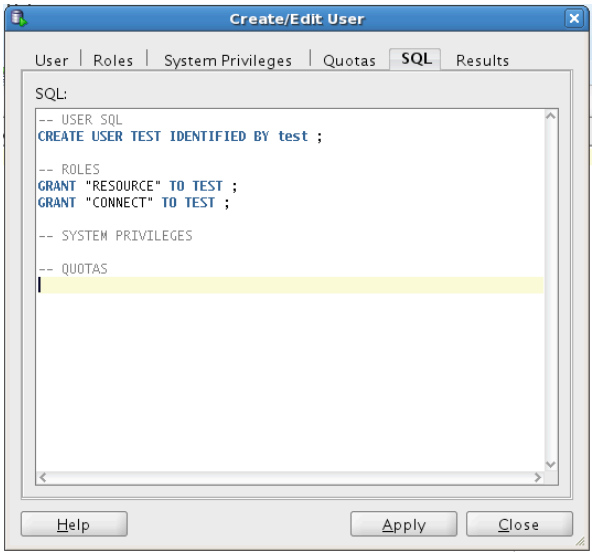
1) Создание новой схемы для хранения новой базы данных:

- Запустить SQL Developer.
- Выполнить вход под *администратором базы данных*, для этого в закладке Connections создать новое соединение: имя соединения – SYSTEM, пользователь – system, пароль – oracle.

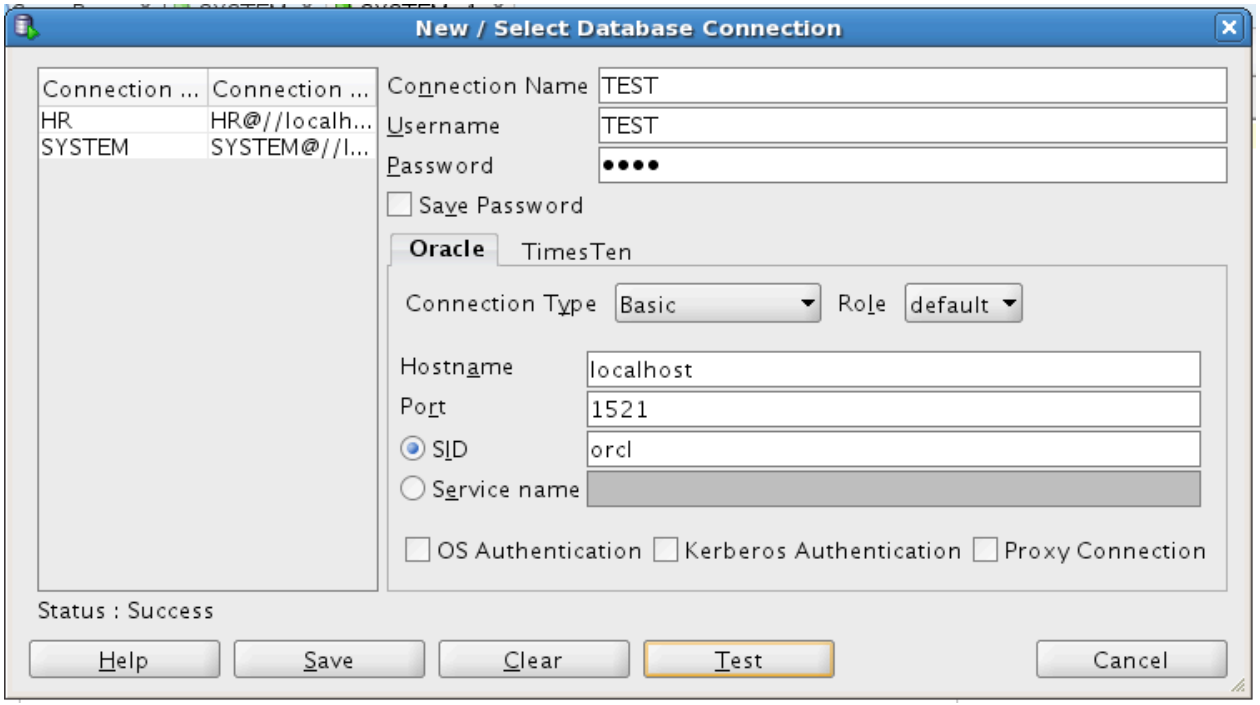


- В соединении SYSTEM в списке элементов выбрать пункт «Other Users» и вызвать на нем контекстное меню. Выбрать в контекстном меню пункт «Create User …». Заполняем поля закладки «User» во всплывшем окне (имя и пароль выбираем по своему вкусу), затем заполняем поля закладки «Roles» (в столбце «Granted» выбираем пункты «CONNECT» и «RESOURCE») – выбранные действия отображаются в закладке «SQL» в виде SQL-операторов - и нажимаем кнопку «Apply».





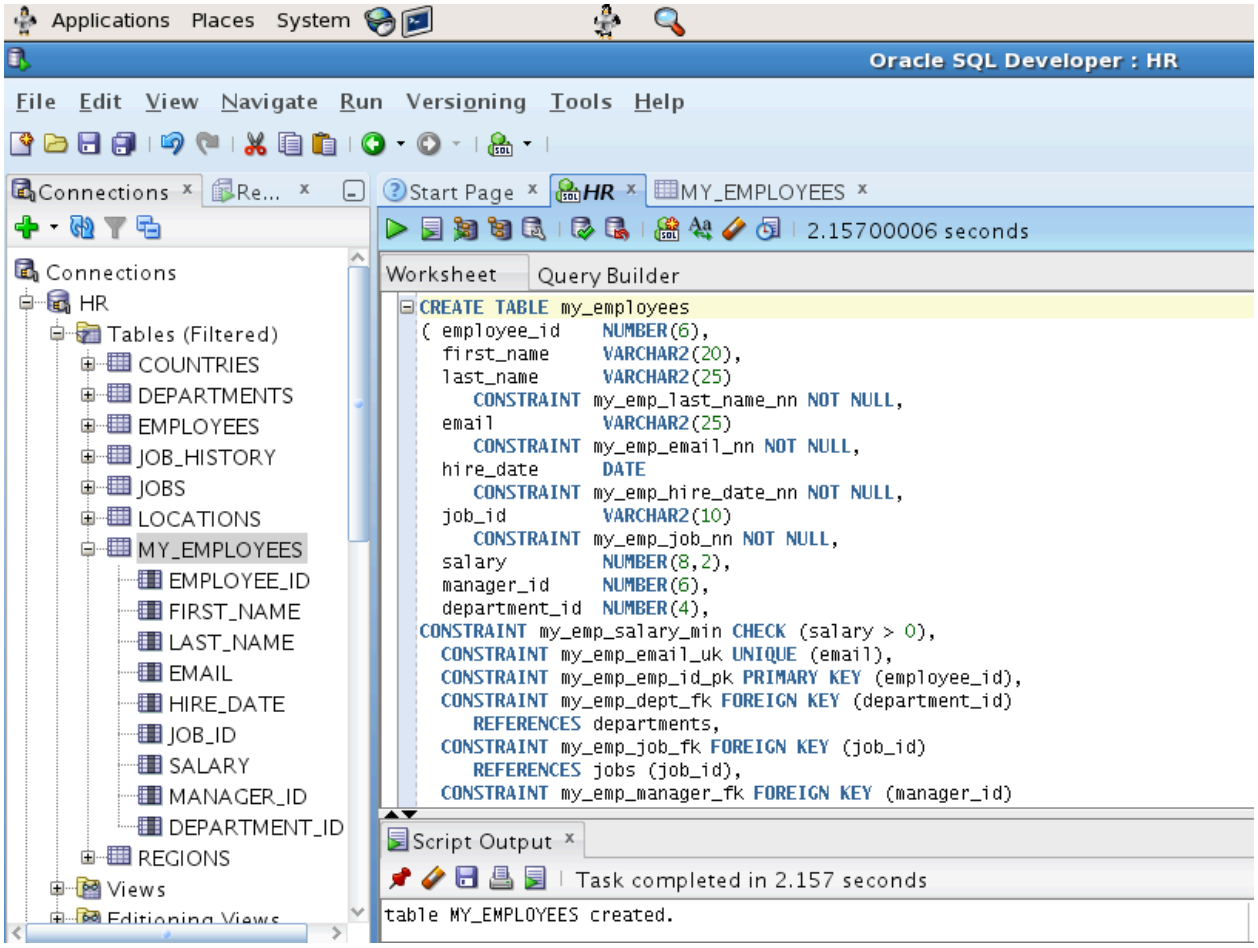
- Теперь можно загрузиться под созданным пользователем – создаем новое соединение и подключаемся к нему:



2) Реализация реляционной модели с помощью команд DDL SQL (желательно использовать указанный ниже порядок; также желательно использование только операторов на языке SQL, набираемых в закладке редактора; при использовании контекстного меню на объектах списка «Table» можно также работать с конструктором таблиц, при этом в закладке DDL будет отображаться текст оператора на языке SQL):

- реализация структуры таблиц в виде набора столбцов с добавлением описаний только первичных ключей;
- реализация ограничений таблиц с добавлением описаний внешних ключей;

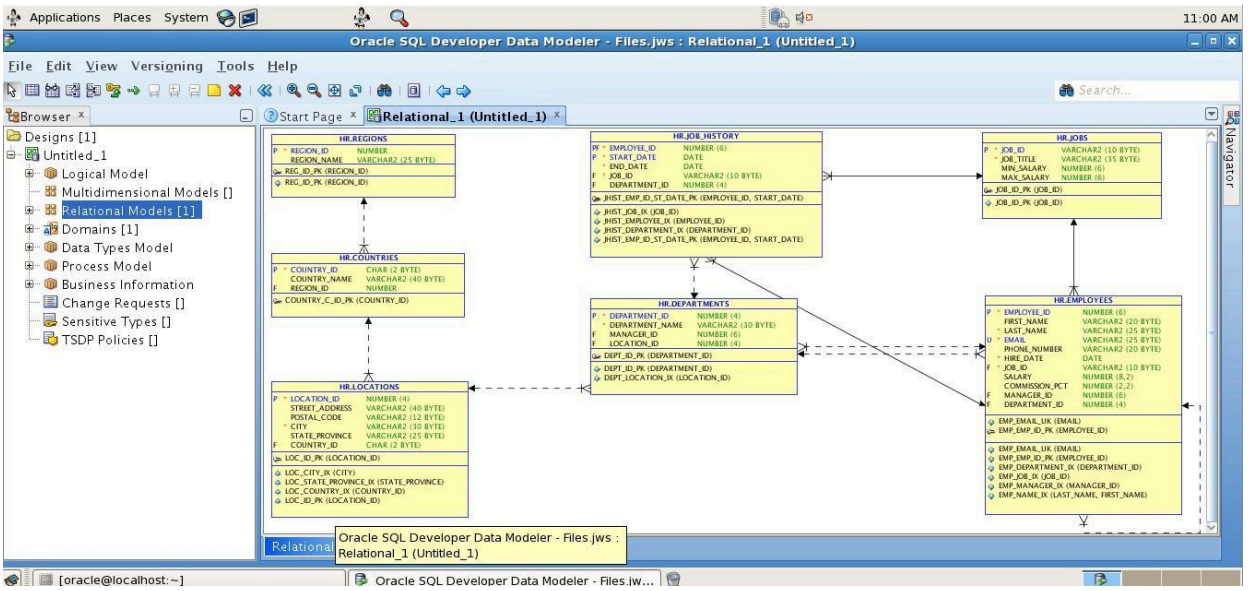
- реализация ограничений таблиц с добавлением бизнес-правил;
- реализация документирования (комментариев к элементам таблицы).



3) Изучение созданной схемы данных:

- Создать реляционную диаграмму:
  - Запустить SQL Developer Data Modeler;
  - File → Import → Data Dictionary → Add (если не видно кнопки – нажать Alt-A);
  - Создать соединение со схемой (например, со схемой HR):
    - username: HR
    - password: oracle
    - SID: orcl

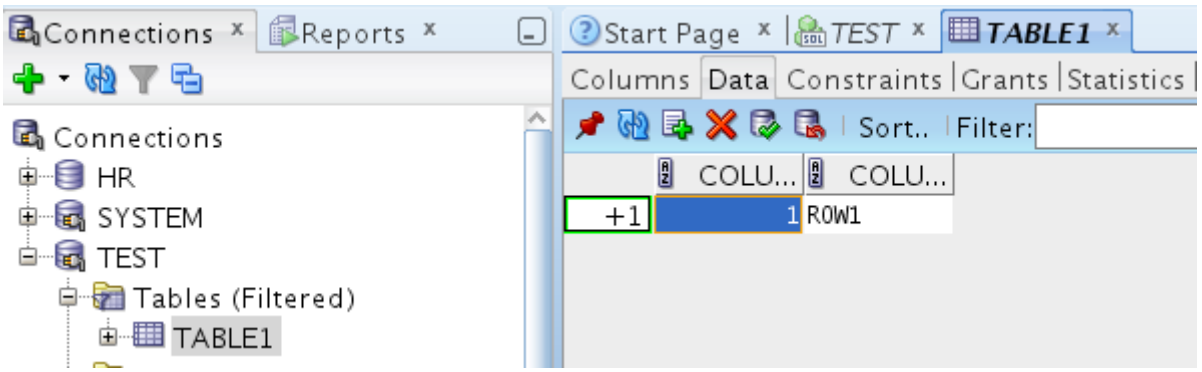
    (желательно сначала Test и только потом Save)
  - Выбрать требуемую схему, выбрать все таблицы, и т.д. до построения реляционной схемы данных (Next = Alt-N, …, Finish = Alt-F);
  - Сохранить полученную схему для изучения (File → Print Diagram → To Image File (в формат .png), файл затем можно скопировать на флешку (FAT)).



- Сравнить структуру таблиц и ссылок для полученной диаграммы и схемы реляционной модели из отчета по лабораторной работе №2.
- При различии в структуре этих диаграмм, надо разобраться, почему так получилось, и выполнить коррекцию (скорее всего проблема заключена в построении SQL-команд, но проблемы могут также появиться и на более раннем этапе – некорректном переводе ER- модели в реляционную модель).

4) Заполнение созданных таблиц строками для проверки правильного выбора первичных ключей и работоспособности ссылок между таблицами можно выполнить следующими путями:

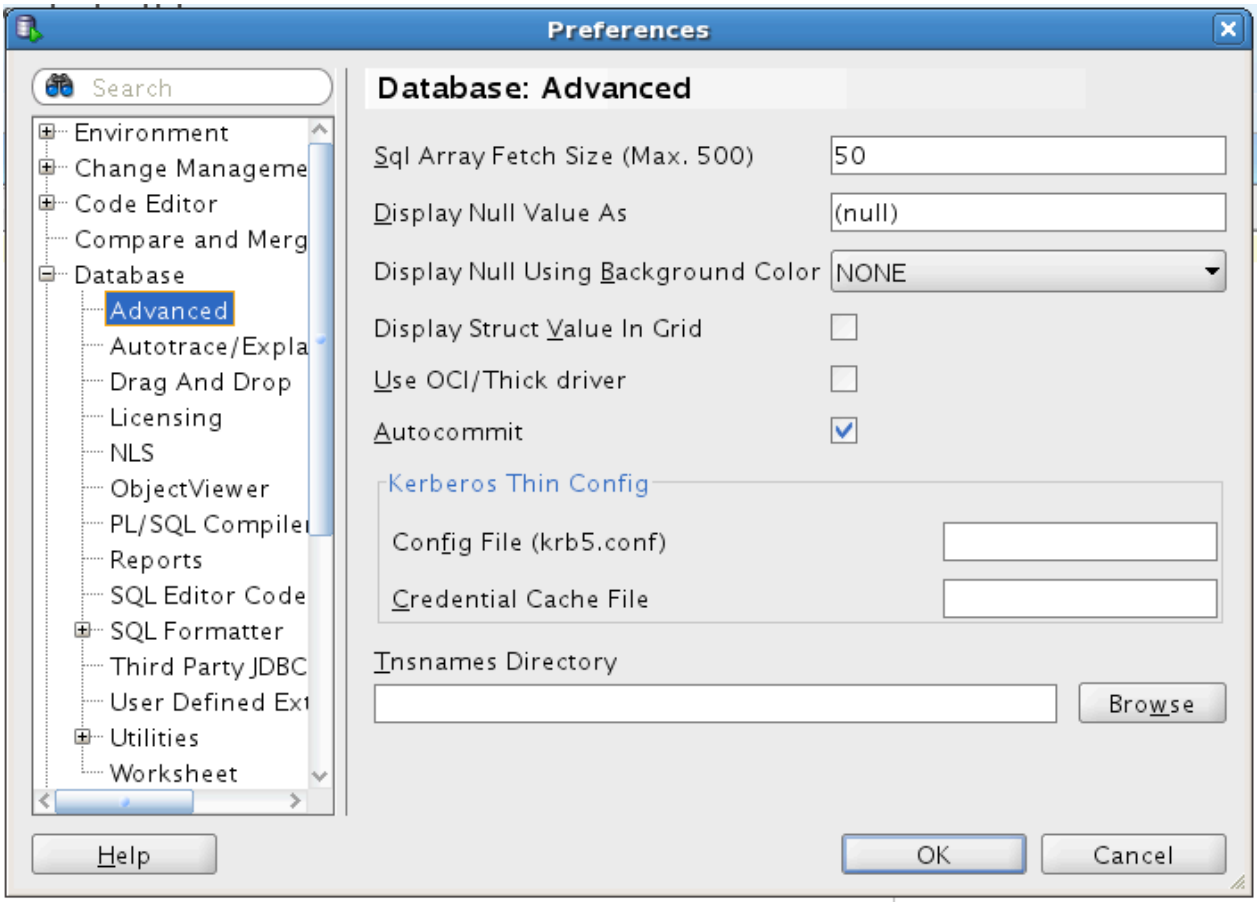
- заполнение таблицы возможно при использовании закладки «Data» при редактировании таблицы (выбрать конкретную таблицу или в контекстном меню таблицы выбрать пункт «Open»), но этот путь не рекомендован к использованию в данной лабораторной работе;



- создание и выполнение SQL-скрипта с операторами `INSERT`, которые сформированы на основе списка данных для вставки одной строки и предназначены для вставки данных только в одну таблицу (см. часть 1).

Особенности добавления данных в таблицы:

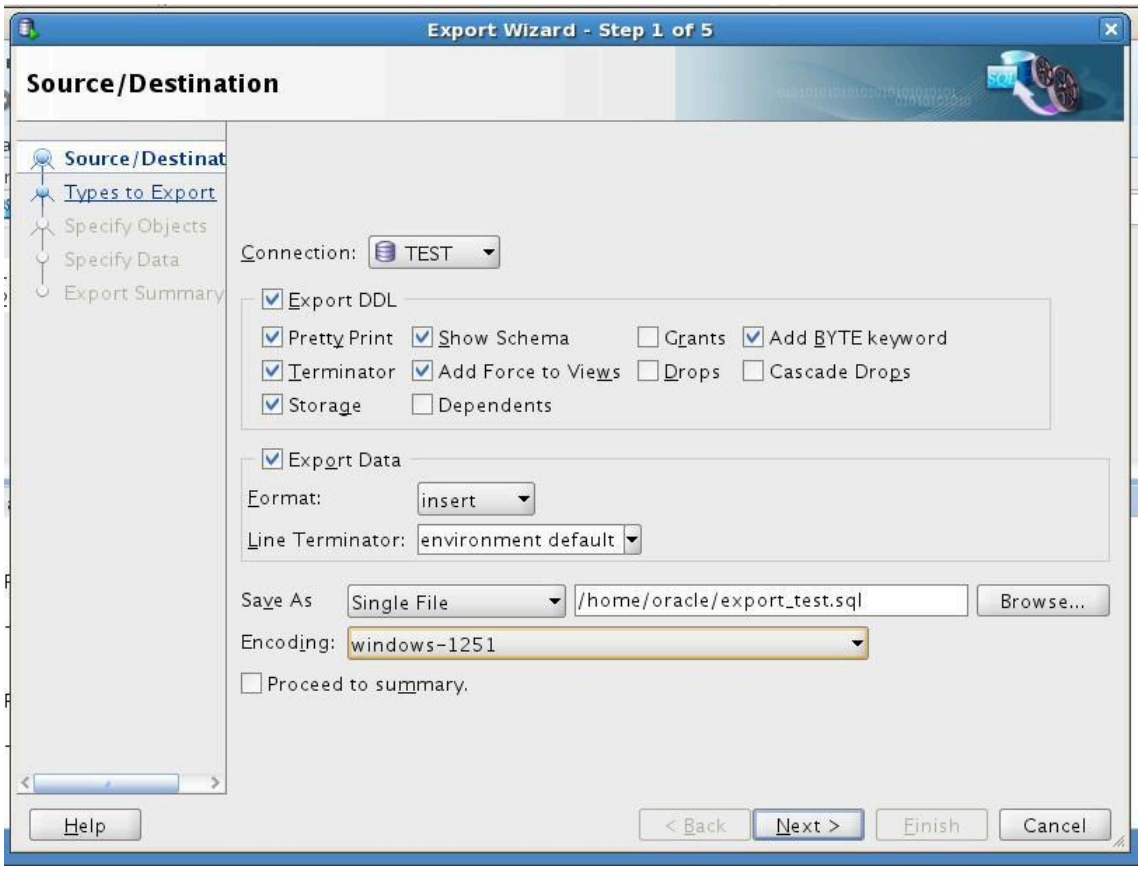
- для работы в режиме автоматической фиксации транзакций (*autocommit*) следует включить автоматическое завершение транзакции (Tools → Preferences → Database → Advanced …);



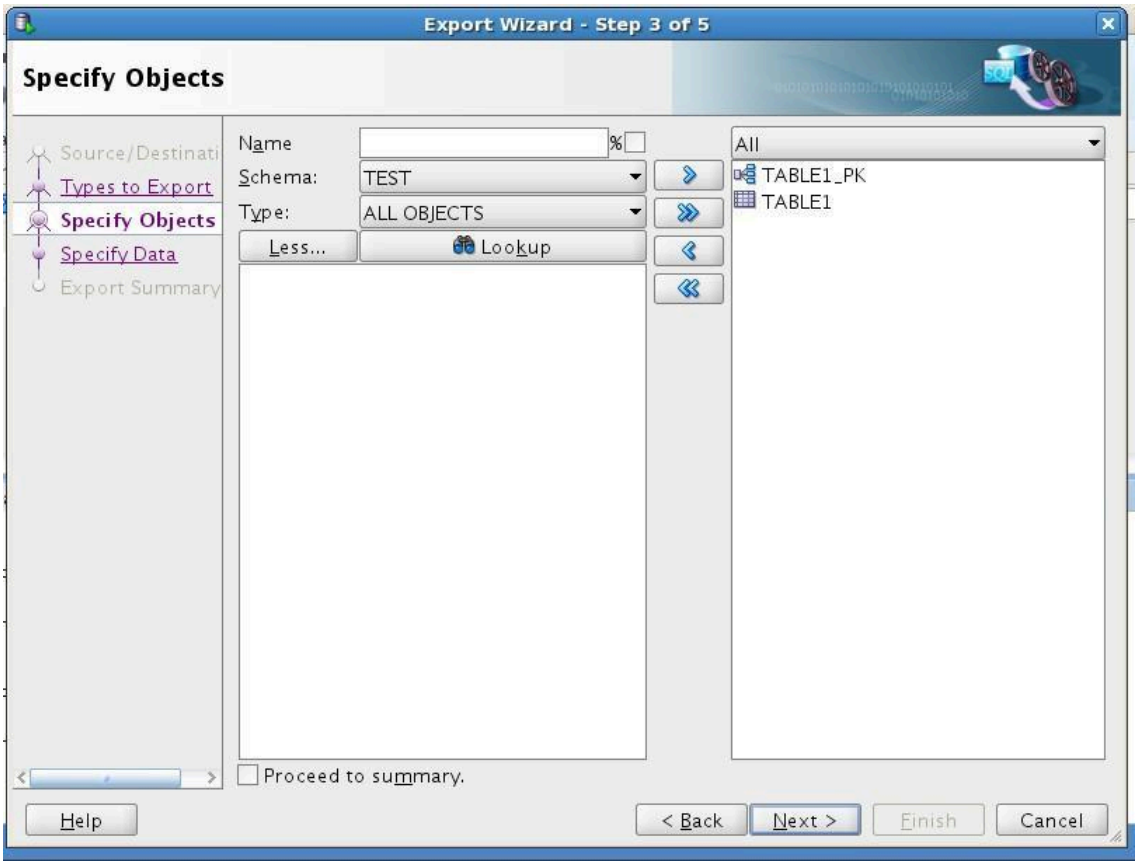
- строками данных желательно сначала заполнять мастер-таблицы (или таблицы, которые НЕ ссылаются на другие таблицы);
- если не удается добавить данные в таблицу по причине нарушения уникальности первичного ключа, то следует перепроверить описание этого первичного ключа;
- если не удается добавить данные в таблицу по причине нарушения ссылочной целостности, то следует убедиться, что целевые данные существуют, иначе перепроверить описание внешнего ключа;
- при добавлении строк желательно добавлять данные не беспорядочно, а по некоторой системе (например, проверить добавление данных во всех допустимых в реальном мире комбинациях, но на небольшом множестве объектов, а не только в одном каком-то направлении).

5) Выгрузка (экспорт) состояния схемы БД в отдельный скрипт, если утеряны скрипты для создания структуры и заполнения ее данными, созданные ранее:

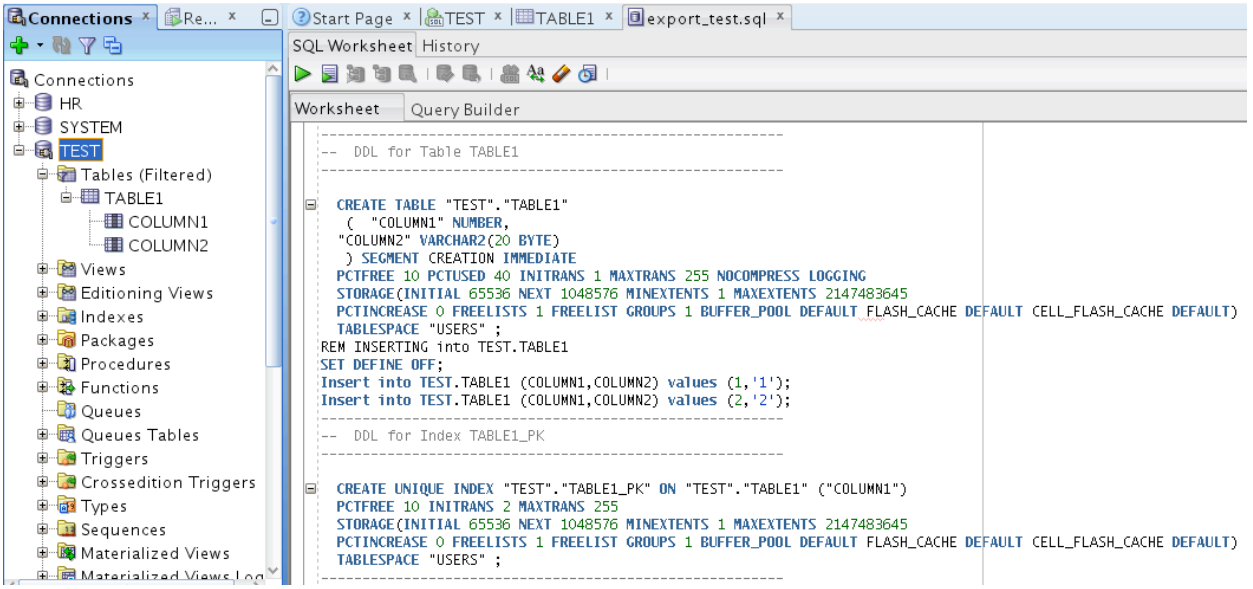
- можно воспользоваться экспортом через (Tools → Database Export) – указать требуемое соединение;



Next … - выбрать требуемые для экспорта объекты



Next … Next … Next … Finish



Минусы такого подхода – скрипт достаточно долго создается, операторы создания таблиц плохо читаются, т.к. утяжелены и размазаны по скрипту. Полученный таким образом скрипт желательно доработать – добавить в его начало команды для удаления создаваемых таблиц, которые уже могут существовать в схеме при его запуске, для того, чтобы не выполнять такие операции вручную каждый раз до запуска скрипта.