COMP 3311 Database Management Systems

Lab 4. Simple DDLs and DMLs, and enforcing constraints

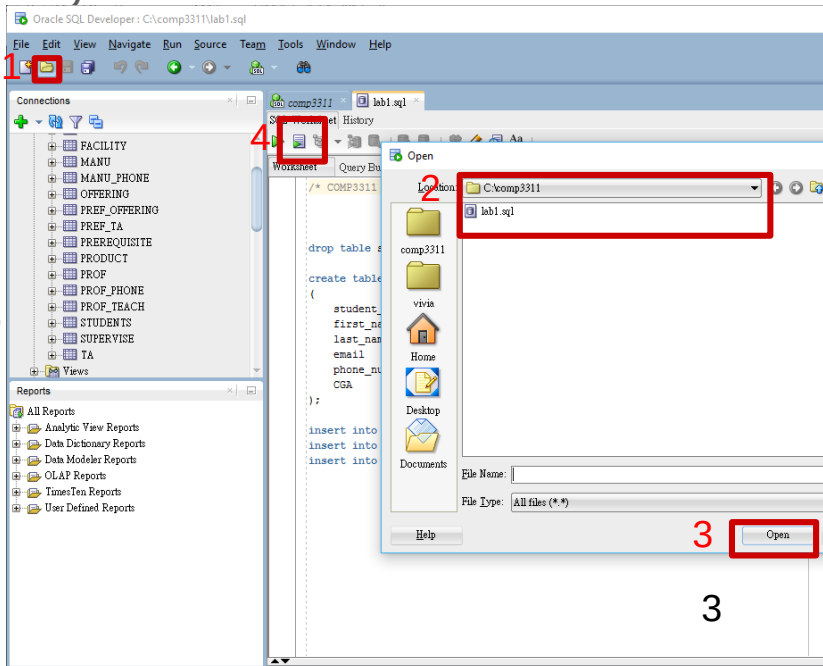
Objectives of the Lab

- After this lab you should be able to
  - Issue simple Data Definition Language commands,
  - create/ modify tables
  - Issue simple Data Manipulation Language commands,
  - insert/ delete/ modify data
  - Know to apply simple integrity constraints.

Downloading and running the lab SQL script file

- login Oracle database server using SQL Developer with your Oracle account
- Download (save) the lab4.sql file to local file system
  - http://course.cs.ust.hk/comp3311/labs/lab4.sql
- Open file
- Run script
- The tables created last time were dropped.
- Some new tables are created.
- Test statement can be download from

  lab4_test.sql



Simple Data Definition Language

- The Data Definition language (DDL) is a language for specifying the database schemes (in the form of tables). It enables operations to be made on creating tables and altering the tables.

- You will learn the following DDLs in this lab.

  - **CREATE**
  - **RENAME**
  - **DROP**
  - **ALTER**

Simple Data Definition Language

- Creating a new table
  CREATE TABLE table_name (column1 datatype, column2 datatype, …);
  create table department_facility (department_id varchar2(4) not null, name varchar2(40), no_of_projectors number(4), no_of_computers number(5));

- Renaming an existing table
  RENAME old_table TO new_table
  RENAME department_facility to test;

Simple Data Definition Language

- Dropping an existing table
  Drop TABLE table_name;
  DROP TABLE test;

- Adding new columns to an existing table
  ALTER TABLE table_name ADD (column1 datatype, column2 datatype,…);
  ALTER TABLE facility ADD (funding number(10));

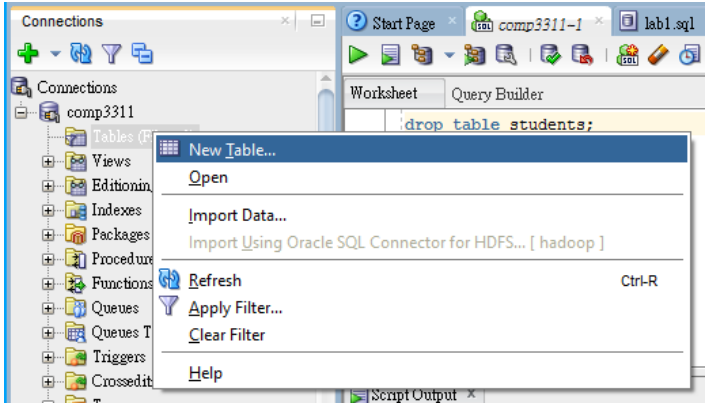
Simple Data Definition Language

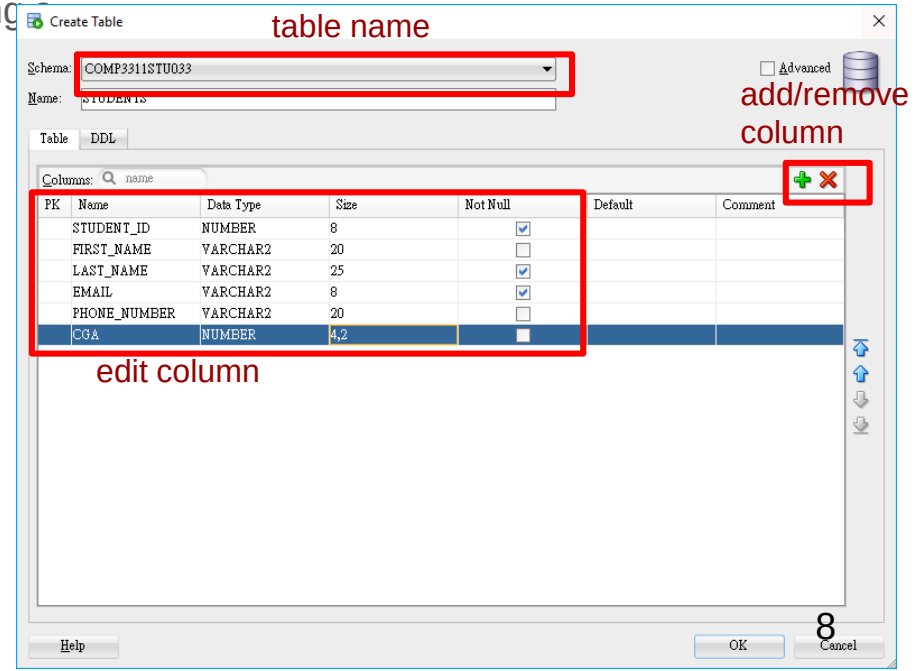
- Changing the data type of the column
  ALTER TABLE table_name MODIFY (column1 datatype, column2 datatype,…);
  ALTER TABLE facility MODIFY (funding varchar2(10));

- Deleting a column from an existing table
  ALTER TABLE table_name DROP (column1,column2,…);
  ALTER TABLE facility DROP (funding);

Appendix: Create Table with SQL Developer

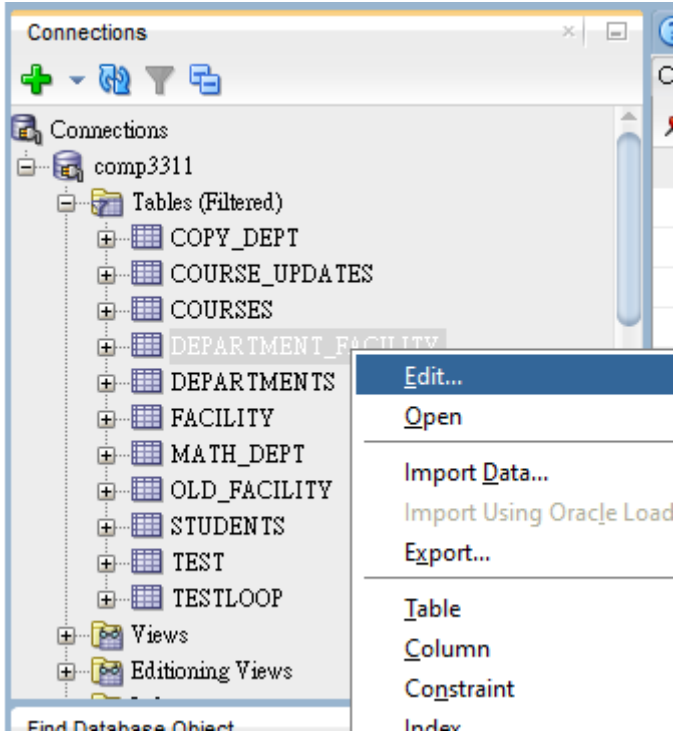
- Right click on “Tables(Filter)” to display its context menu
- Select “New Table” to display a panel for creating a table
- Select “add column(+)” and input attributes
- edit attritutes





Appendix: Defining Database Schemas with SQL Developer

- Right click target table and select edit



Edit Table

Schema: COMP3311STU033

Name: DEPARTMENT_FACILITY

rename table

Table Type: Normal

Search

- Columns
- Constraints
- Indexes
- In-Memory
- Storage
- Comment
- DDL

add/remove column

Columns: name

PK	Name	Data Type	Size	Not Null	Default	Comment
	DEPARTMEN...	VARCHAR2	4	☑		
	NAME	VARCHAR2	40	☐		
	NO_OF_PROJ...	NUMBER	4	☐		
		NUMBER	5	☐		

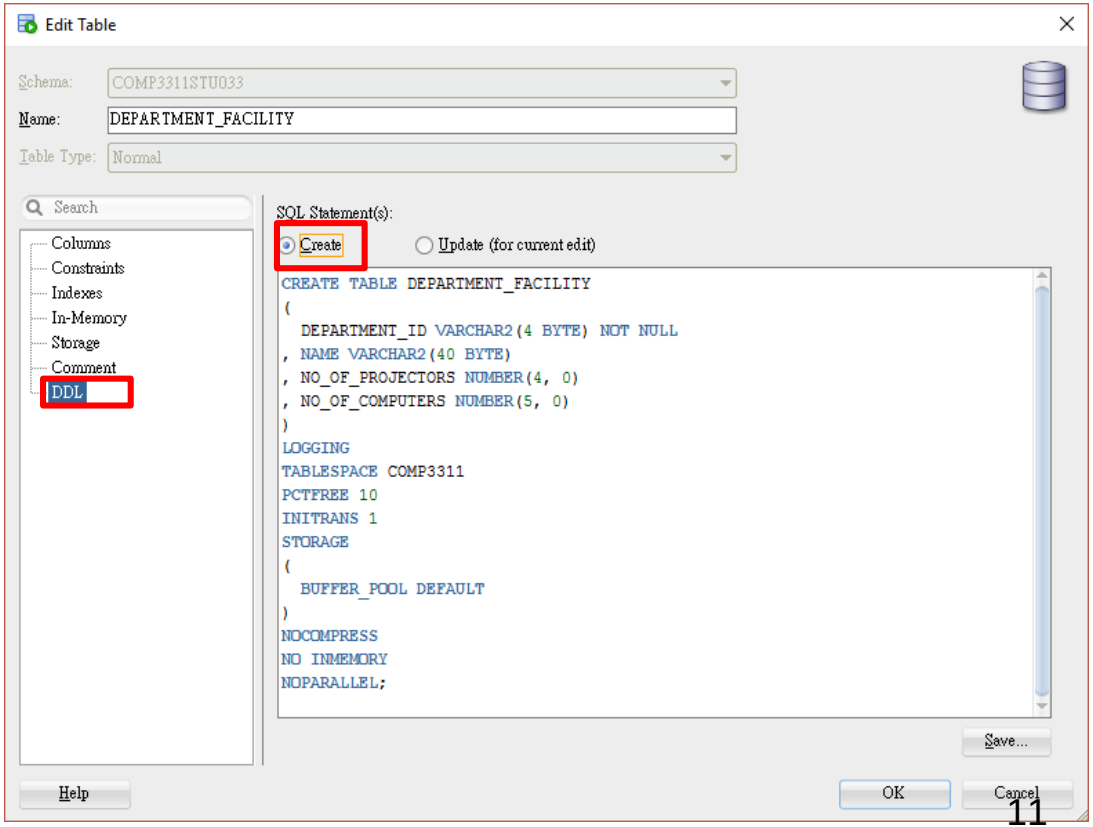
edit column

Data Type | Constraints | Indexes | LOB Parameters | Identity Column

Help | OK | Cancel

Get DDL from SQL Developer

- Open Edit Table dialog
- Select DDL
- Select Create



Simple Data Manipulation Language

- The Data Manipulation language (DML) is a language for manipulating data in a database.
- You will learn the following DML in this lab.
  - INSERT
  - DELETE
  - UPDATE

Simple Data Manipulation Language

- Insert records to an existing table
  INSERT INTO table_name (column1, column2,…) VALUES (value1, value2,…)
  INSERT INTO facility (department_id, name, no_of_projectors, no_of_computers) VALUES ('COMP', 'Computer Science', 5, 150);

- You can omit the column names, if you are inserting records with all the columns present.
  INSERT INTO facility VALUES ('COMP', 'Computer Science', 5, 150);

Simple Data Manipulation Language

- By stating explicitly the columns, you can insert partial records with some of the columns being absent, as long as these columns do not have the “NOT NULL” constraint (will cover the “NOT NULL” constraint in details in the following lab).
  INSERT INTO facility (department_id) VALUES ('test');

Simple Data Manipulation Language

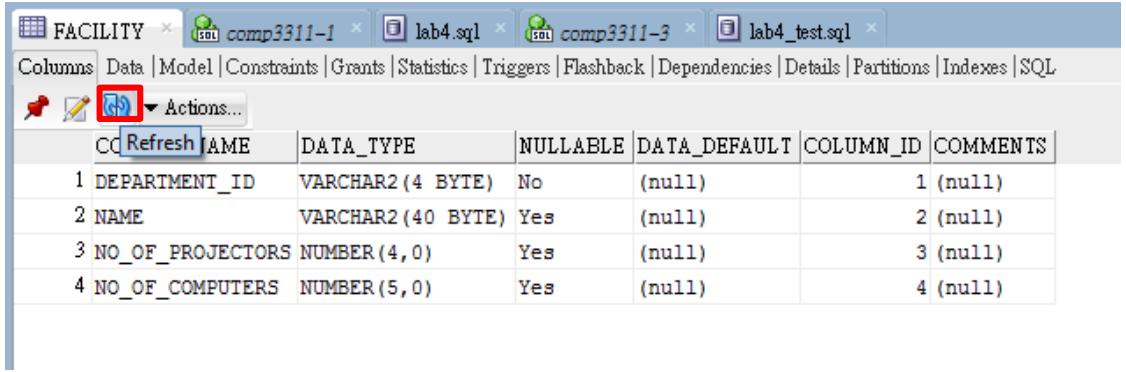
- Removing a record from an existing table
  DELETE FROM table_name [WHERE conditions]
  DELETE FROM facility WHERE department_id='test' ;

- The following statement removes all records from the table facility
  DELETE FROM facility;

Simple Data Manipulation Language

- Updating tuples of an existing table
  UPDATE table_name SET column= value [WHERE conditions];
  UPDATE facility SET no_of_computers=200 WHERE department_id= 'COMP';

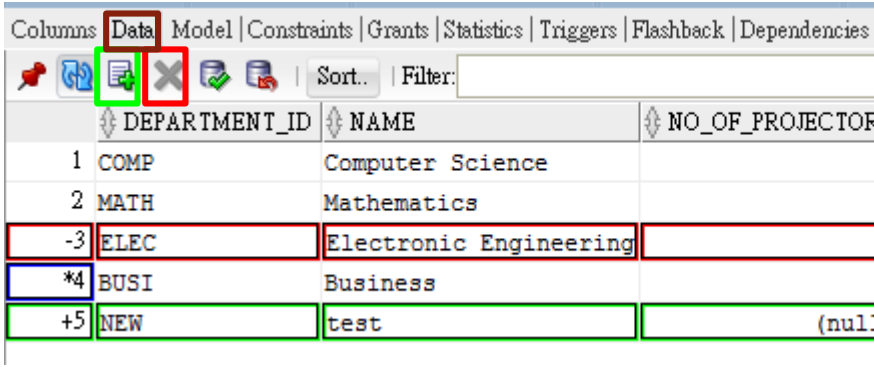
  - Reminder 1 : to test this statement, insert deleted data back before update
  - Reminder 2 : to view the result from table, refresh data



	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	DEPARTMENT_ID	VARCHAR2(4 BYTE)	No	(null)	1	(null)
2	NAME	VARCHAR2(40 BYTE)	Yes	(null)	2	(null)
3	NO_OF_PROJECTORS	NUMBER(4,0)	Yes	(null)	3	(null)
4	NO_OF_COMPUTERS	NUMBER(5,0)	Yes	(null)	4	(null)

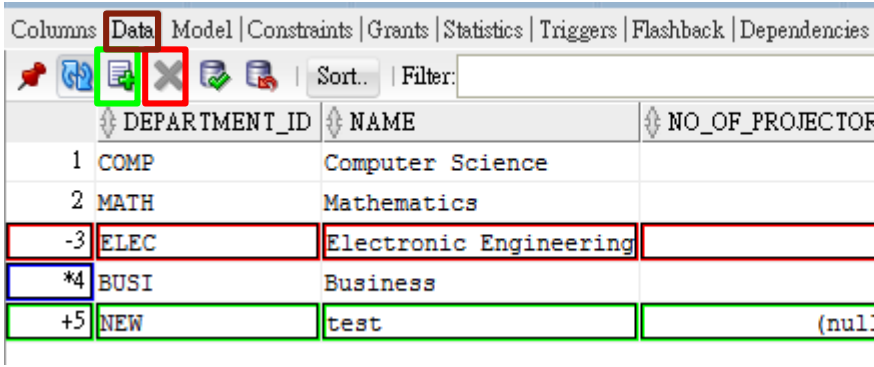
Data Manipulation with SQL Developer 1

- Select target table, Select “Data”
- Insert: click insert row, empty row will be created after selected row
  - example: row +5 :NEW
- Delete: select target record(s), click delete selected row; click again to undo deletion
  - example: row -3 : ELEC



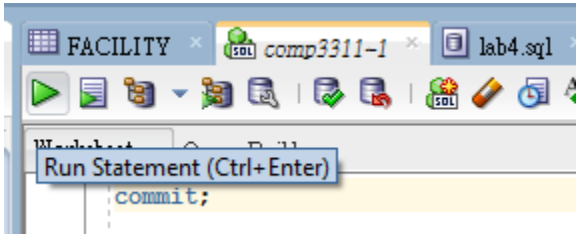
Data Manipulation with SQL Developer 2

- Update: double click on target record
  - example: row *4 : BUS -> BUSI
- Updated rows will be marked (as example on right-hand side), until commitment



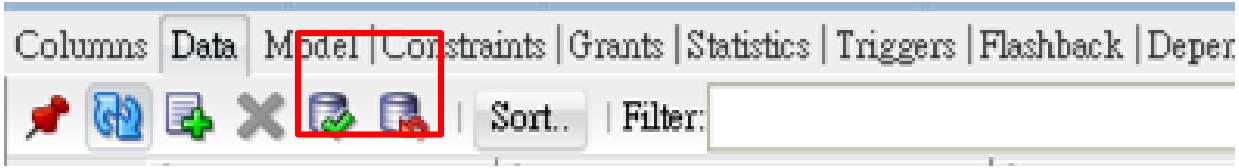
Commit

- Method 1: Commit with SQL statement
  - open SQL worksheet
  - enter “commit;”
  - run statement



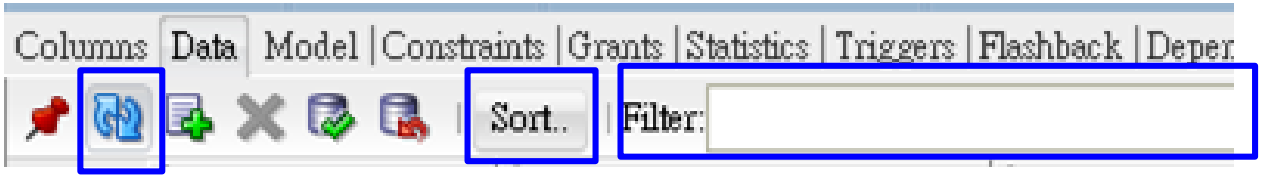
Appendix: Commit alt.

- Method 2: Commit with Controls Under the Data tab
  - Commit Changes ends the current transaction and makes permanent all changes performed in the transaction.
  - Rollback Changes undoes any work done in the current transaction.



Appendix: Other Controls Under the Data tab

- Refresh queries the database to update the data display. If a filter is specified, the refresh operation uses the filter.
- Sort displays a dialog box for selecting columns to sort by.
- Filter enables you to enter a SQL predicate (WHERE clause text without the WHERE keyword) for limiting the display of data.
  For example, NO_OF_COMPUTERS=60



Integrity Constraints 1

- We need to ensure that changes made to the database do not disrupt data consistency.
- One of the methods is to enforce integrity constraints on the database.
- Integrity constraints can be declared at the column level or at the table level.

Integrity Constraints 2

- Column level constraints apply to the columns only. Each constraint involves one column.
- Table level constraints apply to the whole table. Usually involved multi-columns.
- In Oracle, column level constraints are placed right after the column definitions. Table level constraints are placed after all the definitions of the columns.

Constraint commands

- The Constraint commands
  - PRIMARY KEY: specifies the column(s) that are used to uniquely indentify the rows(records) in a table.
  - FOREIGN KEY: specifies the column(s) that is/are being “borrowed” from another table and must be present in that table.
  - UNIQUE: indicates the column has unique values.
  - NOT NULL: indicates the column must have a value.
  - CHECK: place conditions (in the form of a predicate) on the column.
- Oracle Database allows applying the above constraints at the column level or at the table level (except for the NOT NULL constraint which can only be applied as a column level constraint).

Enforcing Integrity Constraints 1

- Examples 1:

```
CREATE TABLE staff (
id number(10) PRIMARY KEY,
age number(3) CHECK
                (age between 0 and 65),
salary number(10)  CHECK
                (salary>0));

CREATE TABLE work (
id number(10) REFERENCES staff(id),
firm_name VARCHAR2(100) NOT NULL,
Primary Key(id, firm_name));
```

Column Level Constraints

Table Level Constraint

Enforcing Integrity Constraints 2

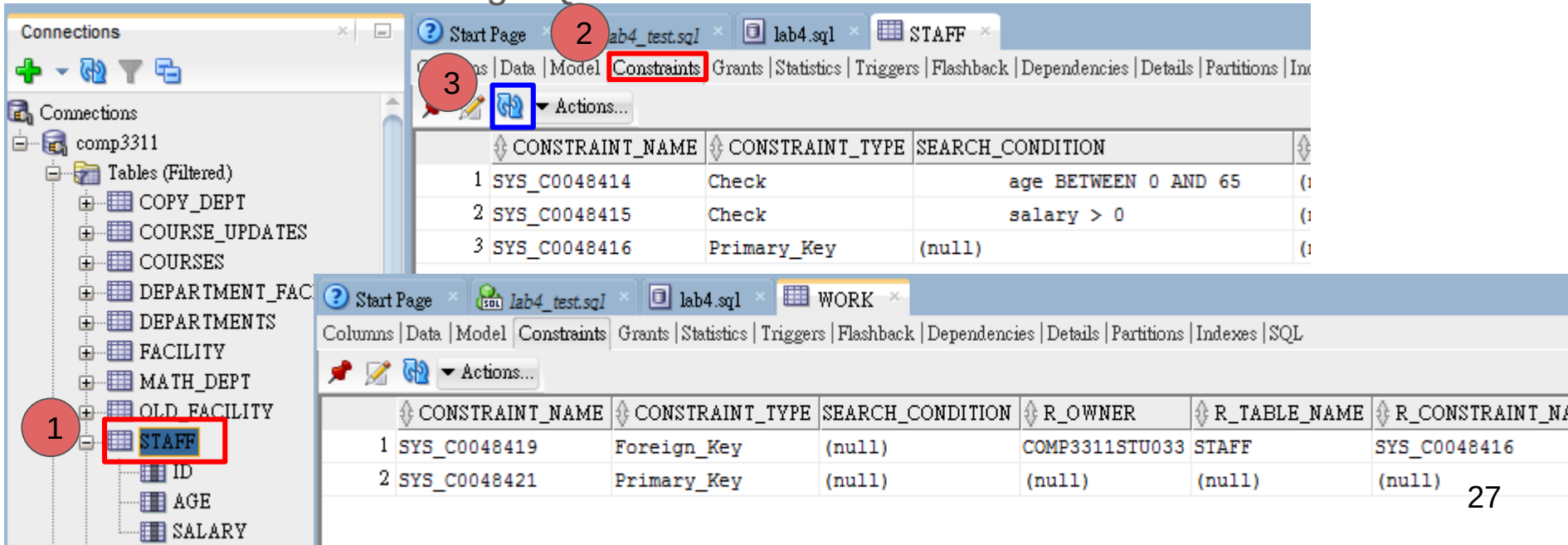
- Examples 2: The following two statements are identical. Note that all the constraints in the second CREATE statement were given *names*.

```
CREATE TABLE work (
id number(10) REFERENCES staff (id),
firm_name VARCHAR2(100) NOT NULL,
  Primary Key (id, firm_name)
);
```

```
CREATE TABLE work(
id number(10),
firm_name VARCHAR2(100)
CONSTRAINT not_null NOT NULL,
CONSTRAINT f_key FOREIGN KEY (id) REFERENCES staff (id),
CONSTRAINT p_key Primary Key(id, firm_name)
);
```

Appendix: Check Integrity Constraints with SQL Developer

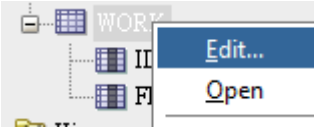
- You can check the constraints through SQL table Object
- Refresh after running SQL statement

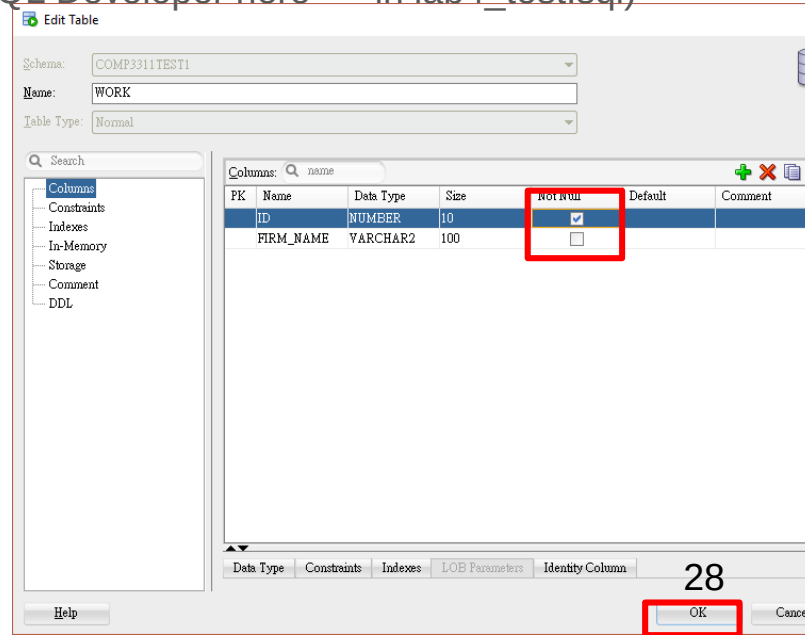


Appendix: Example of Create IC with SQL Developer 1

- To create IC as work table in Example 2,
  - Reminder: drop the previous work table, create a new one without IC (run SQL statements above --**Try to create IC with SQL Developer here**-- in lab4_test.sql) refresh

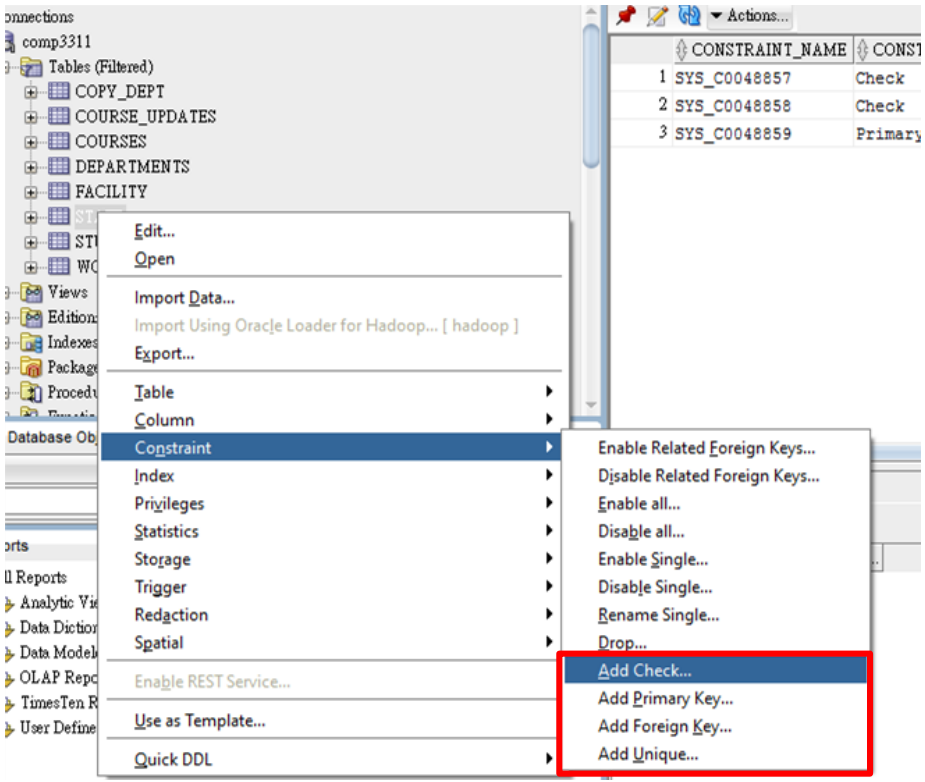
- Create Column level Constraints:
  - Right click on target table (WORK)
  - Select “Edit…”
  - “Edit Table” Dialog opened
  - Select “Column”
  - Tick “Not Null” of ID
  - Click “OK”





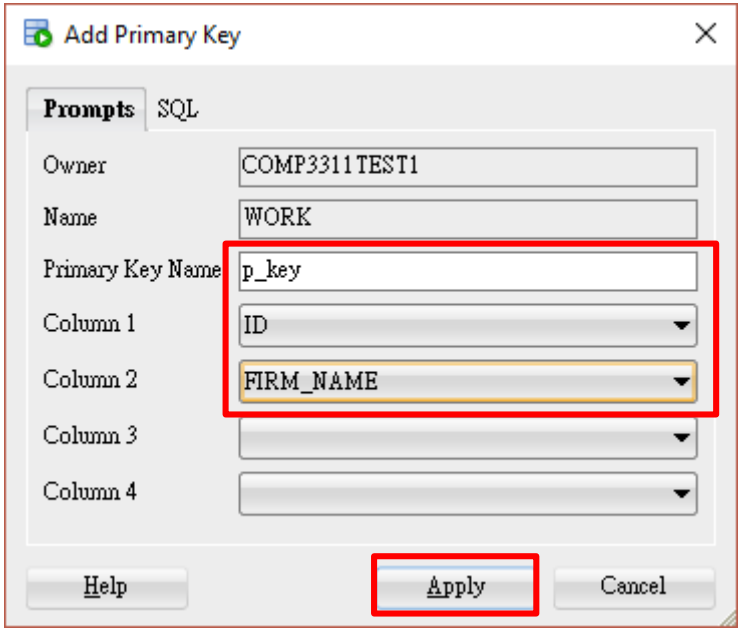
Appendix: Create Integrity Constraints with SQL Developer

- Add table level constraints
  - right click on targe table
  - select “Constraint"
  - select Add *target constraint*



Appendix: Example of Create IC with SQL Developer 2

- Add table level constraints
  - right click on table WORK
  - select “Constraint" (as P.25)
  - select Add Primary Key
    - Primary Key Name: p_key
    - Column 1 : ID
    - Column 2 : FIRM_NAME
  - Click “Apply”
  - Refresh



Appendix: Example of Create IC with SQL Developer 3

- foreign key is similar to primary key, try to add the following foreign key back yourself
  CONSTRAINT *f_key* FOREIGN KEY (id) REFERENCES staff (id)

- Reminder: add table work back before continue alter IC

Modify Integrity Constraints 1

- add constraints in an existing table by their names, using the ALTER TABLE statement.
  - ALTER TABLE staff ADD CONSTRAINT test CHECK (age between 20 and 40);
- modify constraints in an existing table by their names, using the ALTER TABLE statement.
  - ALTER TABLE work MODIFY (firm_name null);

Modify Integrity Constraints 2

- We can drop a non-primary key constraint
- ALTER TABLE staff DROP CONSTRAINT test;

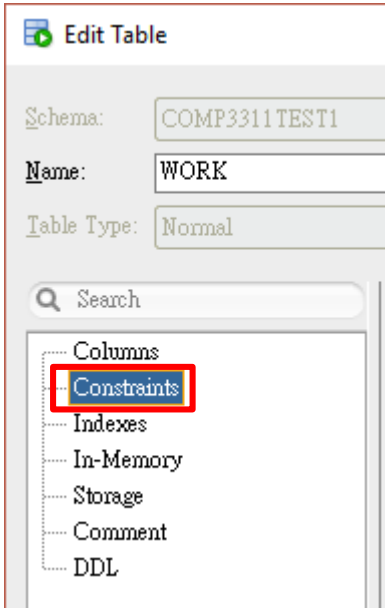
- We can also drop a primary key
- ALTER TABLE work DROP PRIMARY KEY;

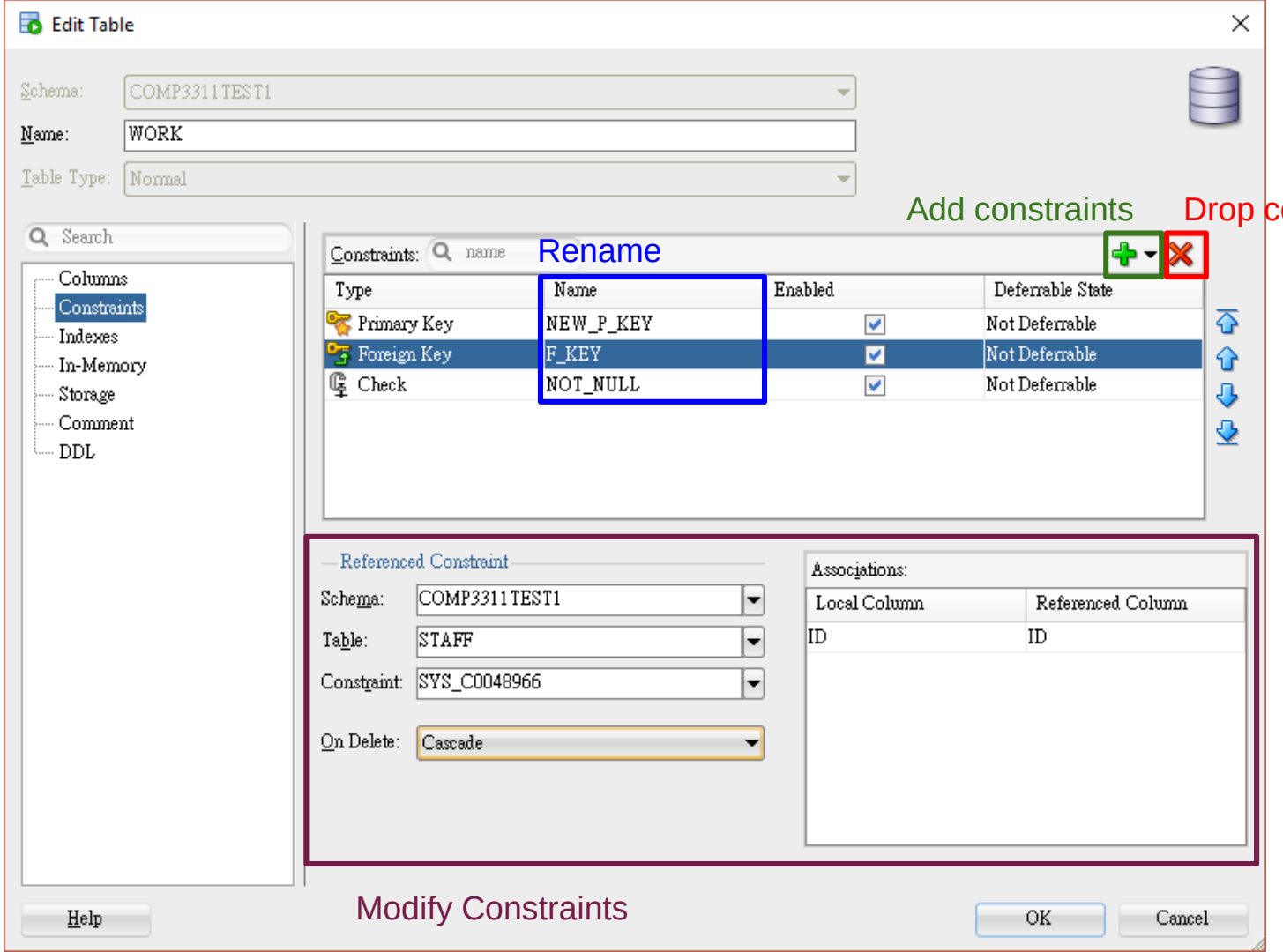
- or add it back
- ALTER TABLE work MODIFY (PRIMARY KEY (id,firm_name));

- We need to remember the constraint name in order to drop it. The following query returns all the declared constraints.
- SELECT constraint_name FROM user_constraints;

Appendix: Modify Integrity Constraints with SQL Developer

- right click on target table
- select "Edit”
- Select “Constraints”
  - Drop Constraints
  - Rename Constraints
    - Example: P_KEY → NEW_P_KEY
  - Modify Constraints
    - F_KEY on delete: “No Action → Cascade”




Edit Table
Schema: COMP3311TEST1
Name: WORK
Table Type: Normal
Add constraints
Drop constraints
Search
Columns
Constraints
Indexes
In-Memory
Storage
Comment
DDL
Constraints: name
Rename
Type
Name
Enabled
Deferrable State
Primary Key
NEW_P_KEY
Not Deferrable
Foreign Key
F_KEY
Not Deferrable
Check
NOT_NULL
Not Deferrable
Referenced Constraint
Schema: COMP3311TEST1
Table: STAFF
Constraint: SYS_C0048966
On Delete: Cascade
Associations:
Local Column
Referenced Column
ID
ID
Help
Modify Constraints
OK
Cancel

Conclusion

- We covered the following topics in this lab:
  - Simple DDLs and DMLs,
  - Enforcing integrity constraints

    Create constraints in table and column level

    modify constraints by alter table or edit Table Dialog