

Projet de Base des Données Université



Rédigé par :

**ABDELLAH AIT
AHMED OUAAL**

TABLE DES MATIERES

INTRODUCTION	3
1 DESCRIPTION ET MODELISATION DE LA BDS.....	4
1.1. CAHIER DES CHARGES	4
1.2. MODELE LOGIQUE DES DONNEES MLD.....	5
1.3. MODELE PHYSIQUE DES DONNEES MPD.....	6
2 CREATION ET ALIMENTATION DE LA BDS.....	8
2.1. CREATION DU COMPTE UTILISATEUR « ABDEL »	8
2.1.1. Etape 1 : Connexion avec « system ».....	8
2.1.2. Etape 2 : Création du compte utilisateur « abdel ».....	10
2.2. CREATION DE LA BDS.....	11
2.2.1. Code SQL DDL	11
2.2.2. Administration de la BDs	16
DD - Afficher les informations à partir du DD	17
DD - Afficher les contraintes.....	18
Remarque très importante.....	18
2.3. ALIMENTATION DE LA BDS	18
3 MANIPULATION DE LA BDS.....	21
Etude de cas 1 : Inspiration du CM Slide 42 (Projection).....	21
Etude de cas 2 : Inspiration du CM Slide 55 (Sélection)	22
Etude de cas 3 : CM Partie 1, Séance 3 et 4, Page 10 (Jointure).....	23

Introduction

Ce projet de fin de module 'les bases des données avancées' consiste en la description, la modélisation, la création, l'implémentation, l'administration et la manipulation d'une Base de Données nommée « university » en utilisant le DBMS Oracle, cette BD modélise les différentes interactions d'un système universitaire qui est défini par son diagramme E-R ci-dessous :

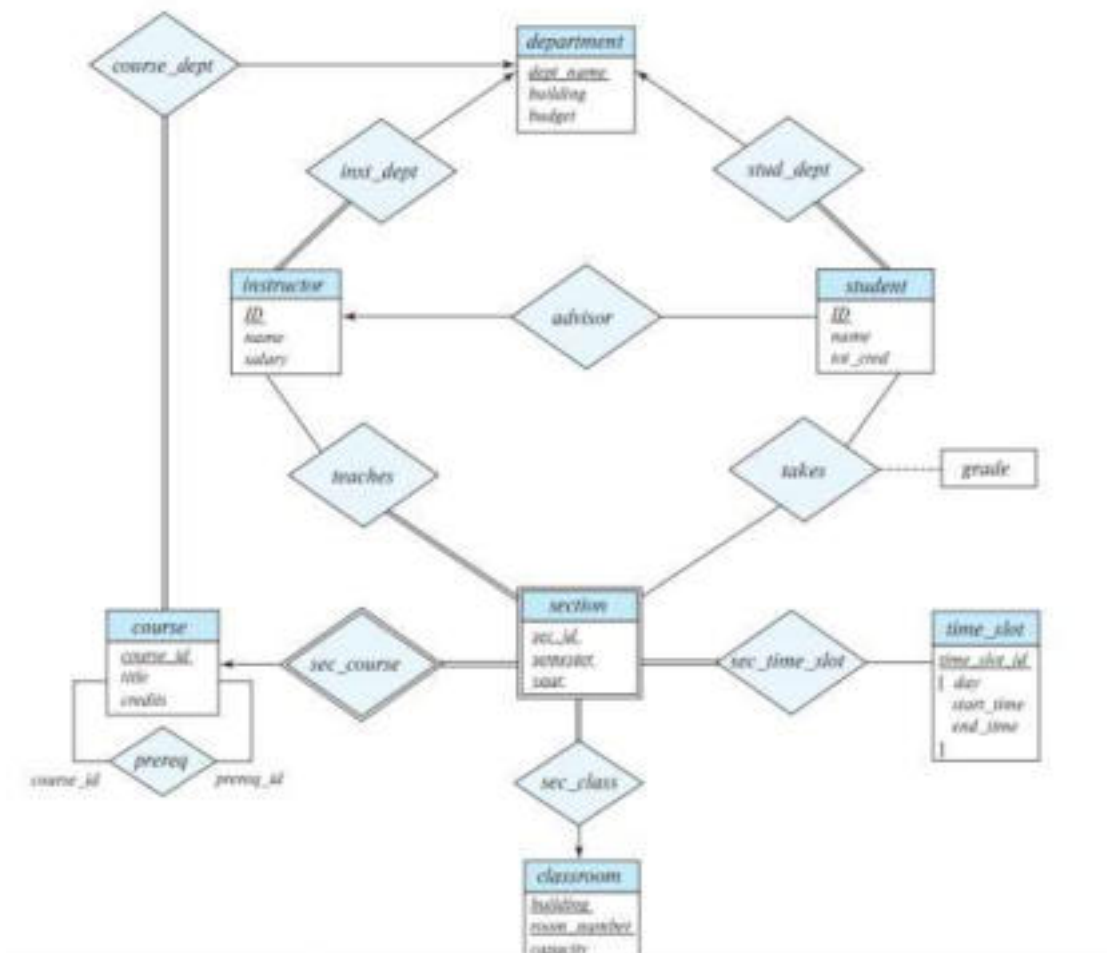


Figure 1 : Diagramme E-R de la BDs university

Notre étude est divisée en trois grandes parties :

- Partie 1 : Description et Modélisation de la BDs
- Partie 2 : Création et Alimentation de la BDs
- Partie 3 : Manipulation de la BDs

1 Description et Modélisation de la BDs

Dans cette première partie de notre étude on va décrire textuellement toutes les associations/rerelations possibles et les traduisent en un cahier des charges, puis on va construire le modèle logique et physique des données à l'aide du logiciel Oracle SQL DataModeler.

1.1. Cahier des charges

Le diagramme de la BDs « university » contient les entités suivants : « department », « instructor », « student », « section », « course », « time_slot » et « classroom », du même le diagramme contient les associations : « advisor », « takes », « teaches », « inst_dept », « stud_dept », « course_dept », « sec_course », « sec_time_slot », « sec_class » et « prereq ».

On va citer les associations entre les différentes entités (relations) du digramme E-R en spécifiant l'optionalité (**peut** ;; **doit**) et la cardinalité (**un seul** ;; **un ou plusieurs**) de chaque association liant deux relations comme suit :

1. Chaque étudiant (student) **doit** s'inscrire (stud_dept) dans **un seul** département (department).
2. Chaque département **peut** contenir **un ou plusieurs** étudiants.
3. Chaque instructeur (instructor) **doit** être affecté (inst_dept) à **un seul** département.
4. Chaque département **peut** contenir **un ou plusieurs** instructeurs.
5. Chaque étudiant **peut** avoir **un seul** instructeur conseiller.
6. Chaque instructeur **peut** être le conseiller (advisor) d'**un ou plusieurs** étudiants.
7. Chaque instructeur **peut** enseigner (teaches) dans **une ou plusieurs** sections (section).
8. Chaque section **doit** avoir **un ou plusieurs** instructeurs.
9. Chaque étudiant **peut** prendre (takes) **une ou plusieurs** sections.
10. Chaque section **peut** avoir **un ou plusieurs** étudiants.
11. Chaque section **doit** recevoir (sec_course) **un seul** cours (course).
12. Chaque cours **peut** être contenue dans **une ou plusieurs** sections.
13. Chaque cours **doit** être affecté (course_dept) à **un seul** département.
14. Chaque département **peut** recevoir **un ou plusieurs** cours.
15. Chaque section **doit** avoir (sec_class) **une seule** classe.
16. Chaque classe (classroom) **peut** être affectée à **une ou plusieurs** sections.
17. Chaque section **doit** avoir (sec_time_slot) **un ou plusieurs** créneaux horaires (time_slot).

18. Chaque créneau horaire **peut** être affectée à **une ou plusieurs** sections.
19. Chaque cours a une association récursive plusieurs à plusieurs avec chaque prérequis (prereq).

Les attributs des relations ainsi que des associations sont données par les schémas des relations en intention alléger :

```
instructor(ID, name, salary)
student(ID, name, tot_cred)
section(sec_id, semester, year)
department(dept_name, building, budget)
course(course_id, title, credits)
time_slot(time_slot_id, day, start_hr, start_min, end_hr, end_min)
classroom(building, room_number, capacity)
advisor(student.ID, instructor.ID)
teaches(instructor.ID, sec_id, semester, year)
takes(student.ID, sec_id, semester, year, grade)
sec_time_slot(sec_id, semester, year, time_slot_id, day, start_hr, start_min)
prereq(prereq_id, course_id)
```

1.2. Modèle Logique des Données MLD

À l'aide du logiciel Oracle DataModeler on va construire le modèle logique des données, le résultat final de MLD est montré en bas :

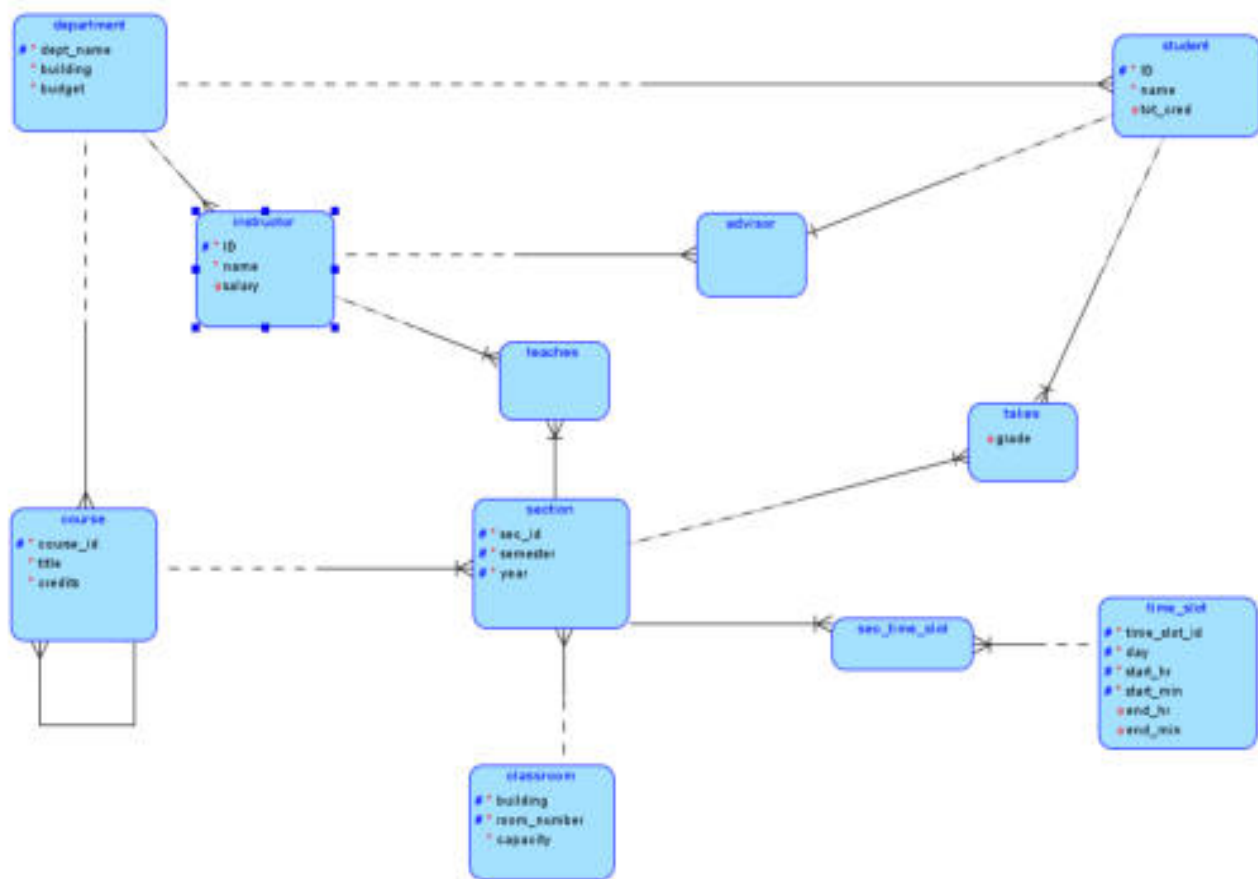


Figure 1.2 : Le MLD

1.3. Modèle Physique des Données MPD

De la même manière on va donner aussi le modèle physique des données à l'aide de l'option « Engineer to Relational Model » qui est disponible à partir du MLD sur DataModeler :

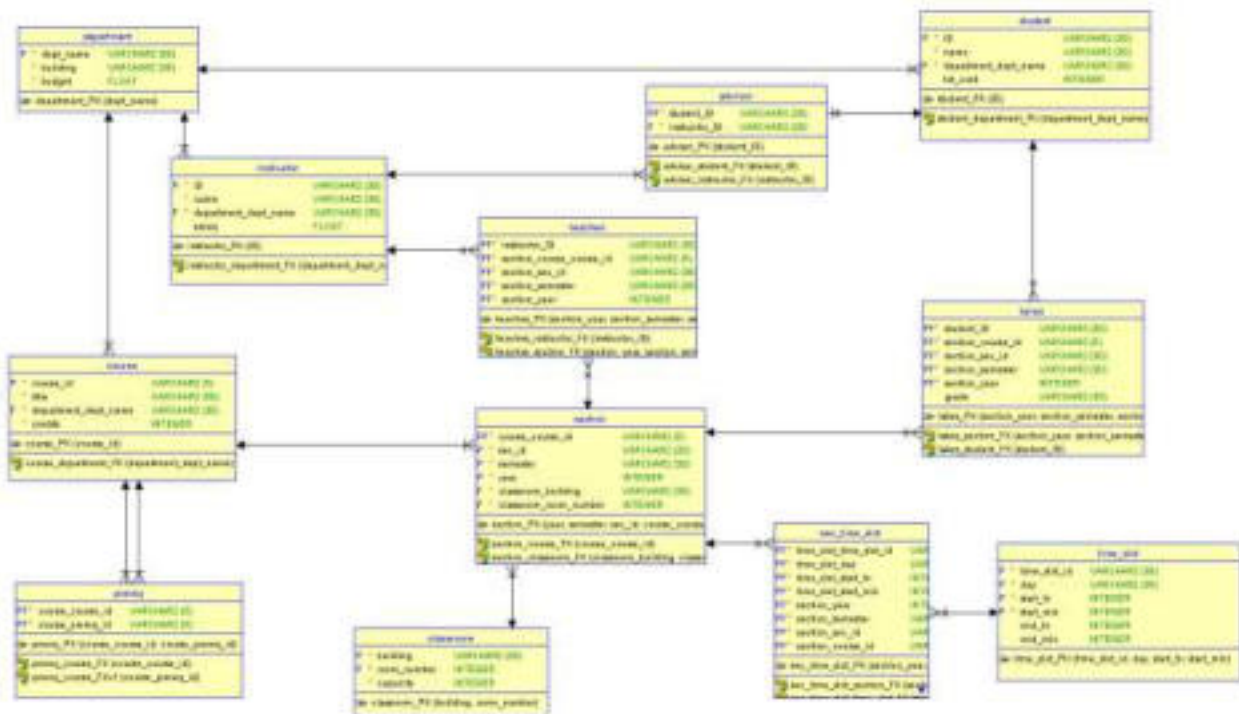


Figure 1.3 : Le MPD

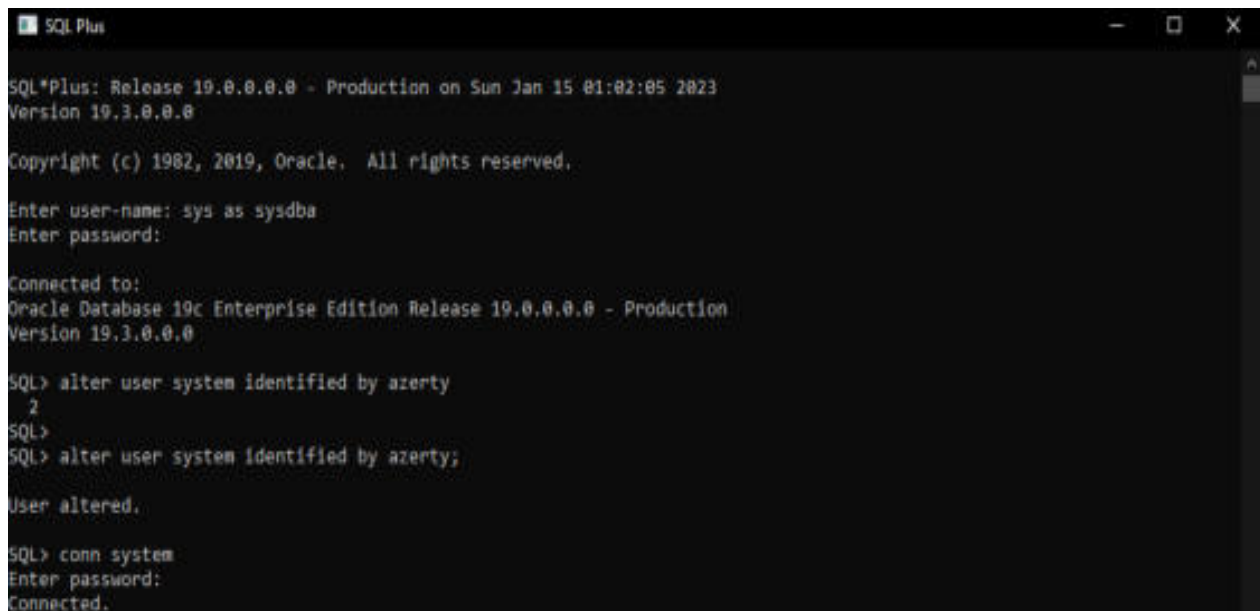
2 Création et Alimentation de la BDs

Dans la deuxième partie on va utiliser les deux outils d'Oracle SQL Plus et SQL Developer pour créer une connexion avec le compte « system » de la BDs « university », puis on va créer un compte utilisateur « abdel » et lui garantir le privilège DBA, ensuite on va générer et donner le code SQL DDL pour créer toutes les tables de BDs, finalement on va alimenter et administrer la BDs en utilisant le fichier *insert.sql*.

2.1. Création du compte utilisateur « abdel »

2.1.1. Etape 1 : Connexion avec « system »

La première des choses est d'utiliser SQL Plus pour se connecter en tant que « sysdba » puis créer un utilisateur « system » à l'aide des commandes suivantes :



```
SQL*Plus
SQL*Plus: Release 19.0.0.0.0 - Production on Sun Jan 15 01:02:05 2023
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle. All rights reserved.

Enter user-name: sys as sysdba
Enter password:

Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0

SQL> alter user system identified by azerty
  2
SQL>
SQL> alter user system identified by azerty;

User altered.

SQL> conn system
Enter password:
Connected.
```

Figure 2.1.1.1 : Connexion avec le compte "system"

Après on va utiliser SQL Developer pour créer une nouvelle connexion avec la base des données qu'on va la nommée « university » et en utilisant le compte « system » :

2 - Création et Alimentation de la BDs – Création du compte utilisateur



Figure 2.1.1.2 : Création de la connexion "university"

Ensuite on va montrer l'utilisateur actuel qui doit être « system » pour vérifier la connexion :



Figure 2.1.1.2 : Montrer l'utilisateur

2.1.2. Etape 2 : Création du compte utilisateur « abdel »

Maintenant sur le même logiciel SQL Developer et l'aide de connexion avec le compte « system » on va créer un nouvel utilisateur qui va porter le nom « abdel » et lui garantir le privilège DBA puis on va valider ces modifications :

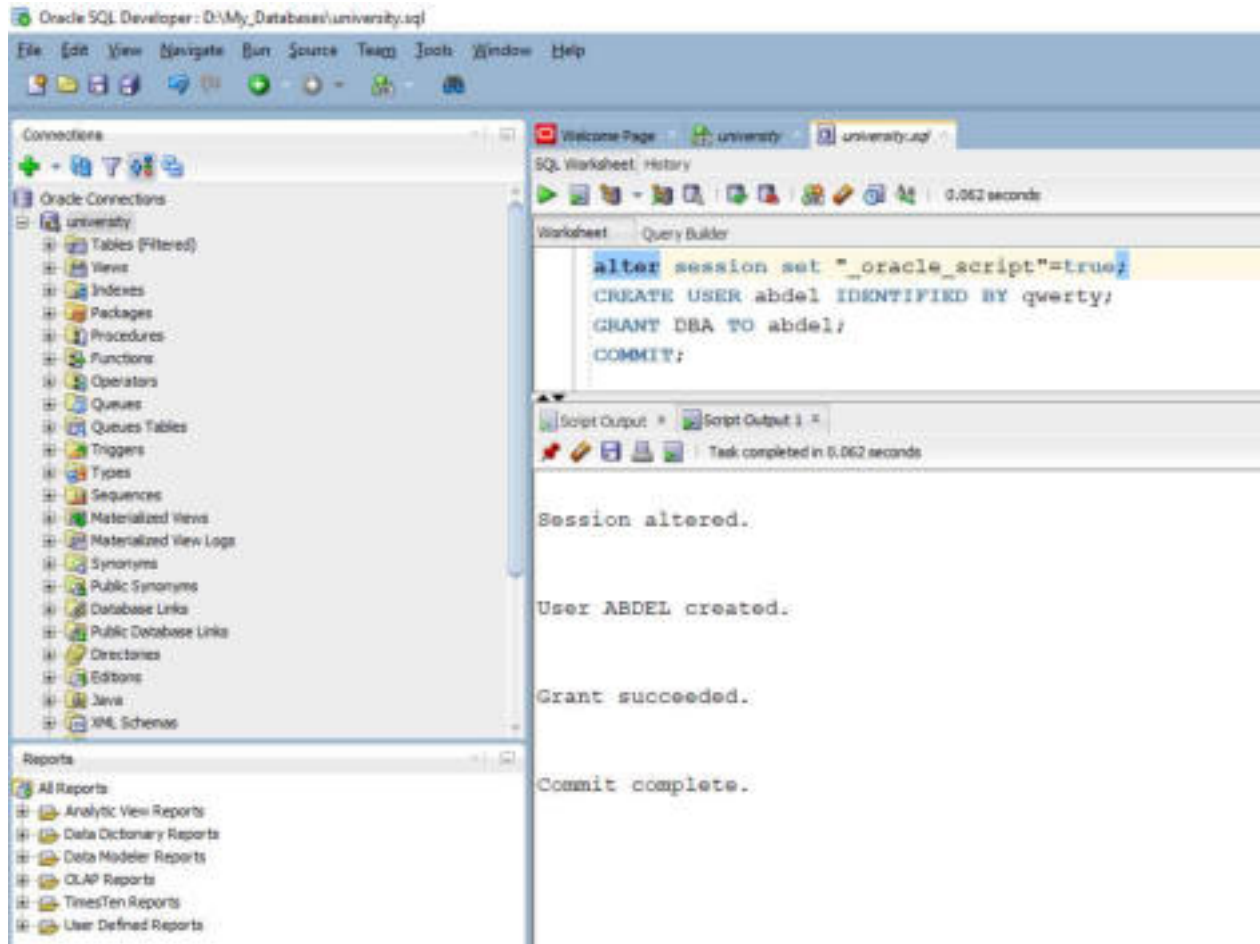


Figure 2.1.2.1 : Création du compte "abdel"

Finalement on doit déconnecter du compte « system » et reconnecter avec « abdel » comme utilisateur, puis vérifier la connexion :

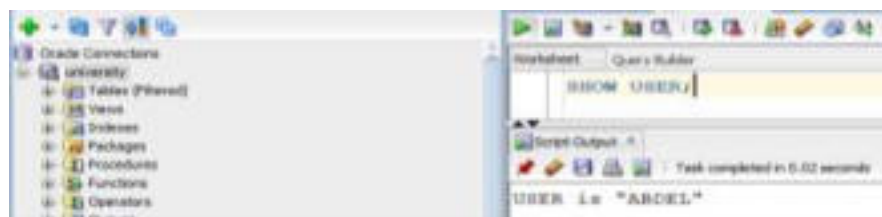


Figure 2.1.2.2 : Montrer le nouvel utilisateur

2.2. Création de la BDs

2.2.1. Code SQL DDL

On va servir de l'option « generate DDL » qui est disponible à partir du MPD de notre BDs sur le DataModeler pour générer le code SQL DDL de notre schéma et qui inclut la création de toutes les tables avec leurs contraintes, on obtient comme résultat le script suivant :

```
CREATE TABLE advisor (  
    student_id VARCHAR2(30) NOT NULL,  
    instructor_id VARCHAR2(30) NOT NULL  
);  
  
ALTER TABLE advisor ADD CONSTRAINT advisor_pk PRIMARY KEY ( student_id  
);  
  
CREATE TABLE classroom (  
    building VARCHAR2(30) NOT NULL,  
    room_number INTEGER NOT NULL,  
    capacity INTEGER NOT NULL  
);  
  
ALTER TABLE classroom ADD CONSTRAINT classroom_pk PRIMARY KEY (  
building,  
room_number );  
  
CREATE TABLE course (  
    course_id VARCHAR2(5) NOT NULL,  
    title VARCHAR2(50) NOT NULL,  
    department_dept_name VARCHAR2(30) NOT NULL,  
    credits INTEGER NOT NULL  
);  
  
ALTER TABLE course ADD CONSTRAINT course_pk PRIMARY KEY ( course_id );  
  
CREATE TABLE department (  
    dept_name VARCHAR2(30) NOT NULL,  
    building VARCHAR2(30) NOT NULL,  
    budget FLOAT NOT NULL  
);  
  
ALTER TABLE department ADD CONSTRAINT department_pk PRIMARY KEY (  
dept_name );
```

```
CREATE TABLE instructor (  
    id          VARCHAR2(30) NOT NULL,  
    name        VARCHAR2(30) NOT NULL,  
    department_dept_name VARCHAR2(30) NOT NULL,  
    salary      FLOAT  
);  
  
ALTER TABLE instructor ADD CONSTRAINT instructor_pk PRIMARY KEY ( id );  
  
CREATE TABLE prereq (  
    course_course_id VARCHAR2(5) NOT NULL,  
    course_prereq_id  VARCHAR2(5) NOT NULL  
);  
  
ALTER TABLE prereq ADD CONSTRAINT prereq_pk PRIMARY KEY (  
course_course_id,  
                                course_prereq_id );  
  
CREATE TABLE sec_time_slot (  
    time_slot_time_slot_id VARCHAR2(30) NOT NULL,  
    time_slot_day          VARCHAR2(30) NOT NULL,  
    time_slot_start_hr     INTEGER NOT NULL,  
    time_slot_start_min    INTEGER NOT NULL,  
    section_year           INTEGER NOT NULL,  
    section_semester       VARCHAR2(30) NOT NULL,  
    section_sec_id         VARCHAR2(30) NOT NULL,  
    section_course_id      VARCHAR2(5) NOT NULL  
);  
  
ALTER TABLE sec_time_slot  
    ADD CONSTRAINT sec_time_slot_pk PRIMARY KEY ( section_year,  
                                                section_semester,  
                                                section_sec_id,  
                                                section_course_id,  
                                                time_slot_time_slot_id,  
                                                time_slot_day,  
                                                time_slot_start_hr,  
                                                time_slot_start_min );  
  
CREATE TABLE section (  
    course_course_id  VARCHAR2(5) NOT NULL,  
    sec_id            VARCHAR2(30) NOT NULL,  
    semester          VARCHAR2(30) NOT NULL,  
    year              INTEGER NOT NULL,  
    classroom_building VARCHAR2(30) NOT NULL,  
    classroom_room_number INTEGER NOT NULL  
);
```

```
ALTER TABLE section
  ADD CONSTRAINT section_pk PRIMARY KEY ( year,
                                         semester,
                                         sec_id,
                                         course_course_id );

CREATE TABLE student (
  id          VARCHAR2(30) NOT NULL,
  name        VARCHAR2(30) NOT NULL,
  department_dept_name VARCHAR2(30) NOT NULL,
  tot_cred    INTEGER
);

ALTER TABLE student ADD CONSTRAINT student_pk PRIMARY KEY ( id );

CREATE TABLE takes (
  student_id      VARCHAR2(30) NOT NULL,
  section_course_id VARCHAR2(5) NOT NULL,
  section_sec_id   VARCHAR2(30) NOT NULL,
  section_semester VARCHAR2(30) NOT NULL,
  section_year     INTEGER NOT NULL,
  grade           VARCHAR2(30)
);

ALTER TABLE takes
  ADD CONSTRAINT takes_pk PRIMARY KEY ( section_year,
                                         section_semester,
                                         section_sec_id,
                                         section_course_id,
                                         student_id );

CREATE TABLE teaches (
  instructor_id      VARCHAR2(30) NOT NULL,
  section_course_course_id VARCHAR2(5) NOT NULL,
  section_sec_id      VARCHAR2(30) NOT NULL,
  section_semester    VARCHAR2(30) NOT NULL,
  section_year        INTEGER NOT NULL
);

ALTER TABLE teaches
  ADD CONSTRAINT teaches_pk PRIMARY KEY ( section_year,
                                         section_semester,
                                         section_sec_id,
                                         section_course_course_id,
                                         instructor_id );
```

```
CREATE TABLE time_slot (  
    time_slot_id VARCHAR2(30) NOT NULL,  
    day          VARCHAR2(30) NOT NULL,  
    start_hr     INTEGER NOT NULL,  
    start_min    INTEGER NOT NULL,  
    end_hr       INTEGER,  
    end_min      INTEGER  
);  
  
ALTER TABLE time_slot  
    ADD CONSTRAINT time_slot_pk PRIMARY KEY ( time_slot_id,  
                                              day,  
                                              start_hr,  
                                              start_min );  
  
ALTER TABLE advisor  
    ADD CONSTRAINT advisor_instructor_fk FOREIGN KEY ( instructor_id )  
        REFERENCES instructor ( id );  
  
ALTER TABLE advisor  
    ADD CONSTRAINT advisor_student_fk FOREIGN KEY ( student_id )  
        REFERENCES student ( id )  
        ON DELETE CASCADE;  
  
ALTER TABLE course  
    ADD CONSTRAINT course_department_fk FOREIGN KEY ( department_dept_name  
)  
        REFERENCES department ( dept_name );  
  
ALTER TABLE instructor  
    ADD CONSTRAINT instructor_department_fk FOREIGN KEY ( department_dept_name )  
        REFERENCES department ( dept_name );  
  
ALTER TABLE prereq  
    ADD CONSTRAINT prereq_course_fk FOREIGN KEY ( course_course_id )  
        REFERENCES course ( course_id );  
  
ALTER TABLE prereq  
    ADD CONSTRAINT prereq_course_fkv1 FOREIGN KEY ( course_prereq_id )  
        REFERENCES course ( course_id );  
  
ALTER TABLE sec_time_slot  
    ADD CONSTRAINT sec_time_slot_section_fk FOREIGN KEY ( section_year,  
                                                         section_semester,  
                                                         section_sec_id,  
                                                         section_course_id )
```

```
REFERENCES section ( year,
                    semester,
                    sec_id,
                    course_course_id )
ON DELETE CASCADE;

ALTER TABLE sec_time_slot
ADD CONSTRAINT sec_time_slot_time_slot_fk FOREIGN KEY (
time_slot_time_slot_id,
                    time_slot_day,
                    time_slot_start_hr,
                    time_slot_start_min )
REFERENCES time_slot ( time_slot_id,
                    day,
                    start_hr,
                    start_min )
ON DELETE CASCADE;

ALTER TABLE section
ADD CONSTRAINT section_classroom_fk FOREIGN KEY ( classroom_building,
                    classroom_room_number )
REFERENCES classroom ( building,
                    room_number );

ALTER TABLE section
ADD CONSTRAINT section_course_fk FOREIGN KEY ( course_course_id )
REFERENCES course ( course_id );

ALTER TABLE student
ADD CONSTRAINT student_department_fk FOREIGN KEY (
department_dept_name )
REFERENCES department ( dept_name );

ALTER TABLE takes
ADD CONSTRAINT takes_section_fk FOREIGN KEY ( section_year,
                    section_semester,
                    section_sec_id,
                    section_course_id )
REFERENCES section ( year,
                    semester,
                    sec_id,
                    course_course_id )
ON DELETE CASCADE;

ALTER TABLE takes
ADD CONSTRAINT takes_student_fk FOREIGN KEY ( student_id )
REFERENCES student ( id )
```

```
ON DELETE CASCADE;
```

```
ALTER TABLE teaches
```

```
ADD CONSTRAINT teaches_instructor_fk FOREIGN KEY ( instructor_id )  
REFERENCES instructor ( id )  
ON DELETE CASCADE;
```

```
ALTER TABLE teaches
```

```
ADD CONSTRAINT teaches_section_fk FOREIGN KEY ( section_year,  
                                                section_semester,  
                                                section_sec_id,  
                                                section_course_course_id )  
REFERENCES section ( year,  
                    semester,  
                    sec_id,  
                    course_course_id )  
ON DELETE CASCADE;
```

2.2.2. Administration de la BDs

Après l'exécution de script SQL ci-dessus sur SQL Developer et pour afficher les tables créées, il faut rafraîchir le bouton « Tables » au-dessous de la connexion :

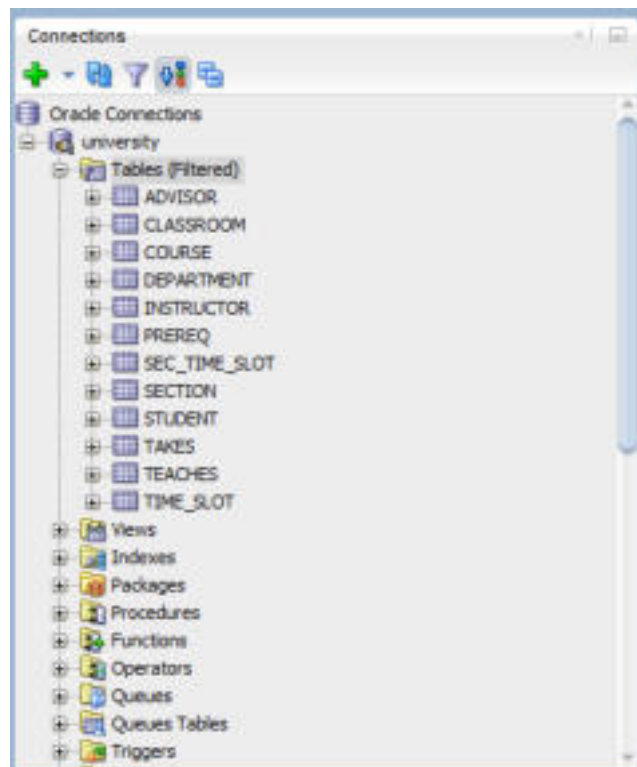


Figure 2.2.2.1 : Affichage des tables graphiquement

DD - Afficher les informations à partir du DD

On peut aussi afficher la liste de toutes les table à travers la requête ci-dessous et on obtient le résultat suivant :

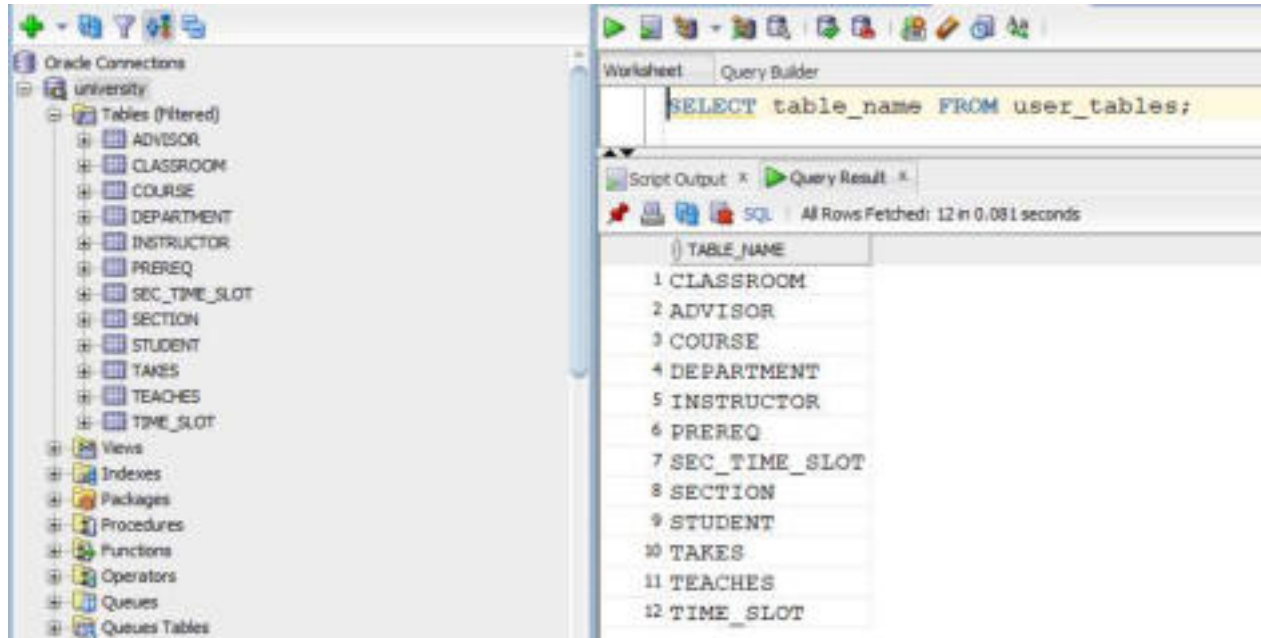


Figure 2.2.2.2 : Affichage des tables à partir du DD

Si on veut par exemple afficher les informations sur les colonnes de la table « student », on exécute l'instruction suivante :

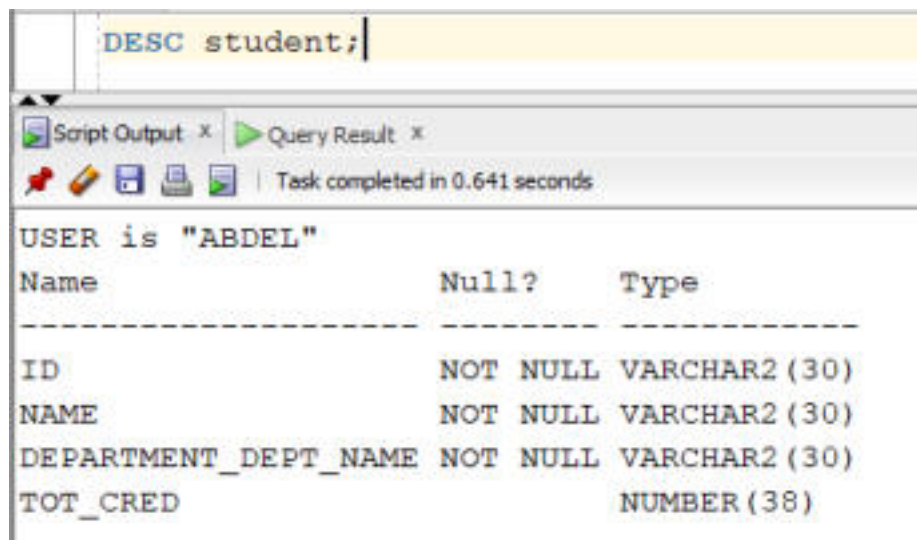
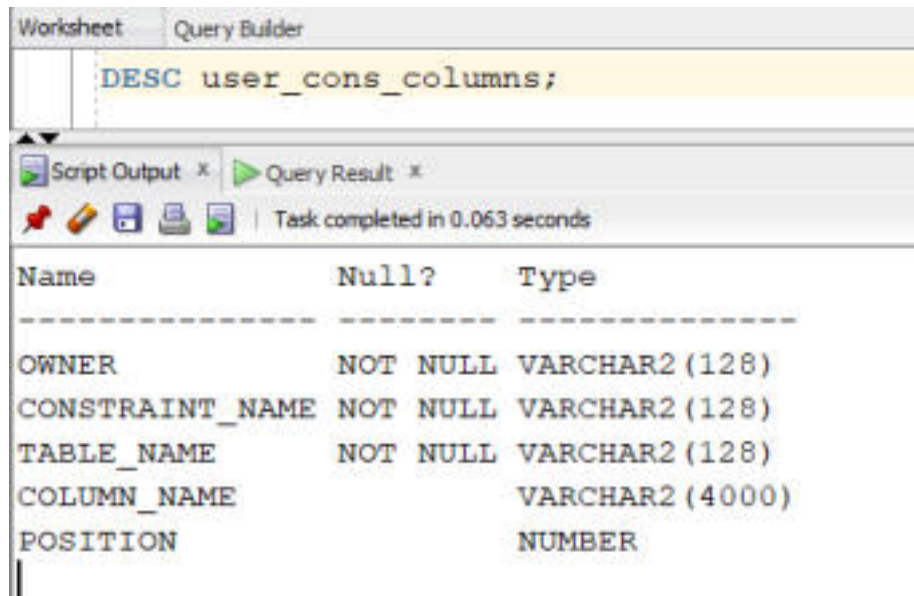


Figure 2.2.2.3 : Décrire la table "student"

DD - Afficher les contraintes

Pour décrire les colonnes qui appartiennent à l'utilisateur actuel « abdel » et qui sont spécifiées dans les définitions de contraintes, il suffit de faire comme montrer ci-dessous :



The screenshot shows a SQL Query Builder window with the command `DESC user_cons_columns;` entered. Below the command, the results are displayed in a table format. The table has three columns: Name, Null?, and Type. The results are as follows:

Name	Null?	Type
OWNER	NOT NULL	VARCHAR2 (128)
CONSTRAINT_NAME	NOT NULL	VARCHAR2 (128)
TABLE_NAME	NOT NULL	VARCHAR2 (128)
COLUMN_NAME		VARCHAR2 (4000)
POSITION		NUMBER

Figure 2.2.2.4 : Affichage des contraintes

2.3. Alimentation de la BDs



Remarque très importante

Il est utile de mentionner ici les problèmes que j'ai eu lors de la phase de l'alimentation de la BDs en utilisant le fichier *insert.sql*, et les solutions que j'ai apporté à ces problèmes :

1 L'erreur ORA-01722 : invalid number

Cette erreur est survenue lors d'un échec de la conversion d'une chaîne de caractères en un nombre valide, après plusieurs tentatives j'ai constaté qu'avant de générer le code DDL en MPD, il faut mettre tous les attributs de toutes les tables sur le même ordre qui figure sur les arguments des fonctions VALUES () de fichier *insert.sql*, **il faut respecter le même ordre.**

2 L'erreur ORA-00913 : too many values

Cette erreur se produit lorsque le deuxième ensemble contient plus d'éléments que le premier ensemble. La table « section » contient en total seulement 6 colonnes (attributs), mais dans le fichier *insert.sql* le nombre des arguments dans l'instruction INSERT INTO section VALUES () est égal à 7 donc c'est une colonne de plus, après l'étude de tous les attributs j'ai constaté que cette colonne n'est pas porteuse de aucune information, **d'où j'ai la supprimée.**

2 - Création et Alimentation de la BDs – Alimentation de la BDs

Après les modifications mentionnées en remarque, on exécute aisément notre fichier *insert.sql*, le temps d'exécution (dans mon cas 105s) vient du fait que notre fichier contient plus de 34000 lignes :

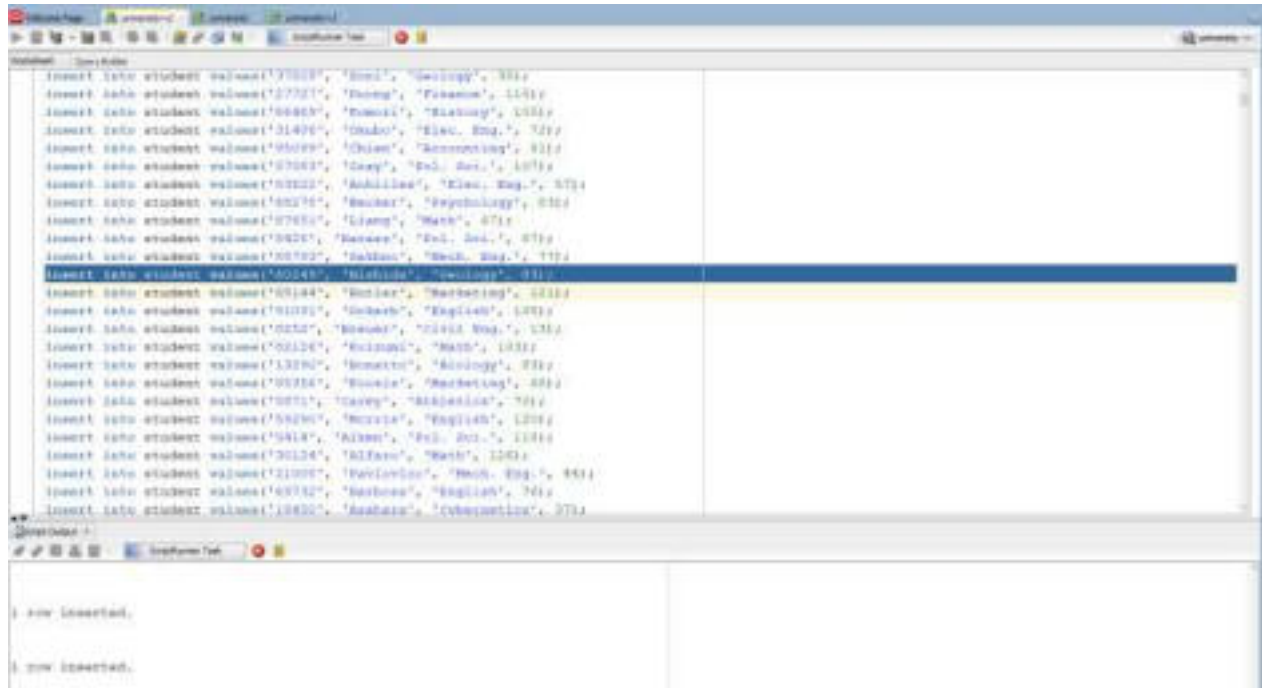


Figure 2.3.1 : Alimentation de la BDs en cours d'exécution

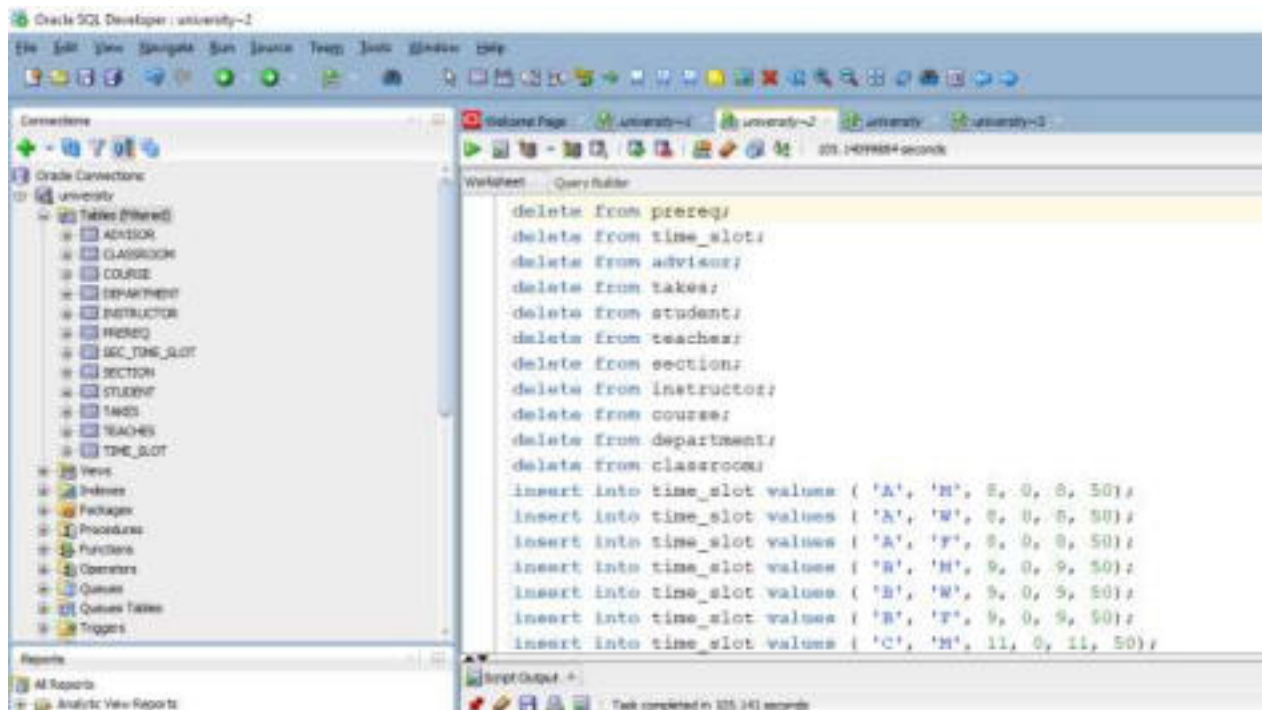
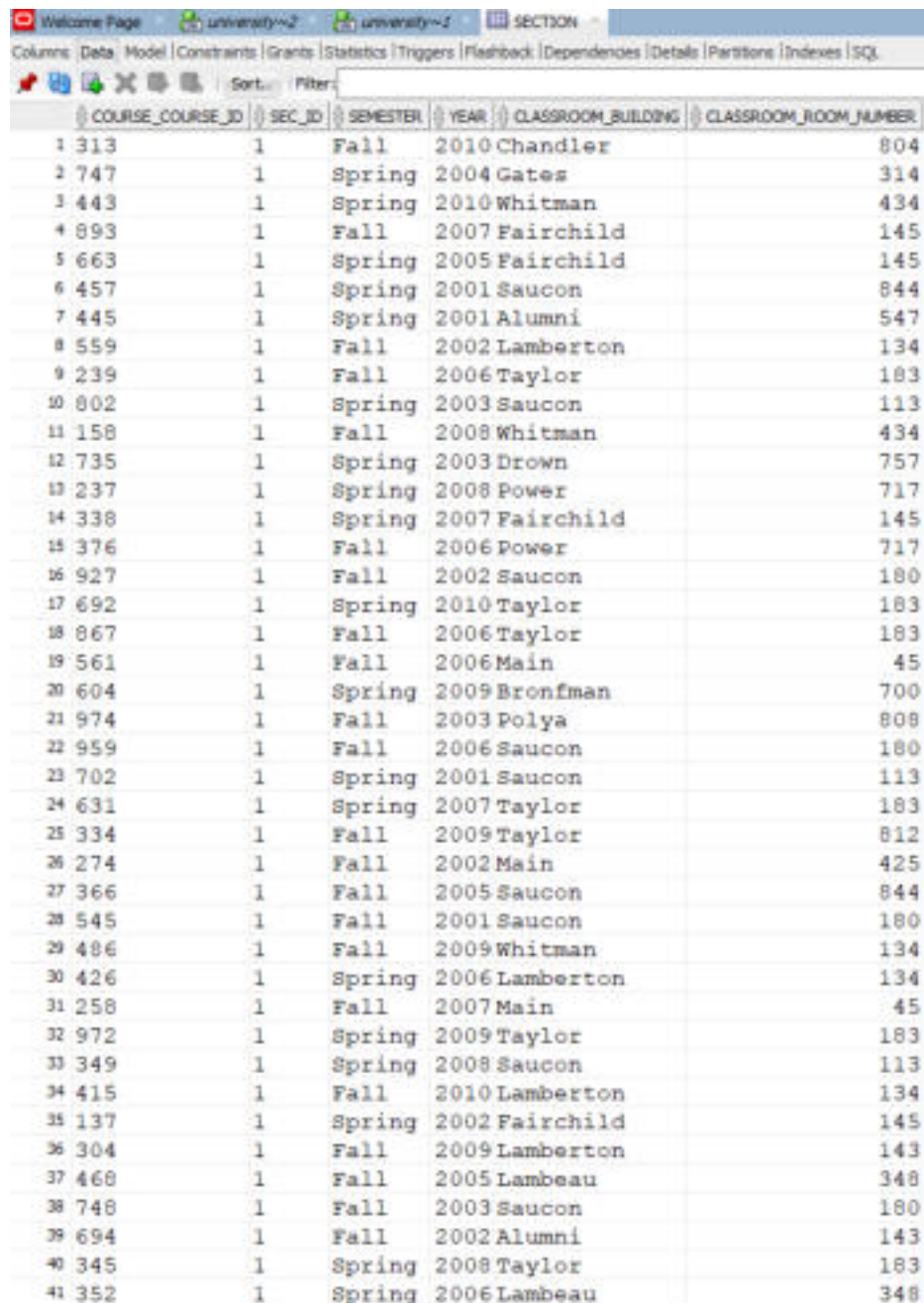


Figure 2.3.2 : Fin d'exécution

2 - Création et Alimentation de la BDs – Alimentation de la BDs

Aussi on peut graphiquement afficher par exemple la table « section » alimenté au sein du logiciel :



	COURSE_COURSE_ID	SEC_ID	SEMESTER	YEAR	CLASSROOM_BUILDING	CLASSROOM_ROOM_NUMBER
1	313	1	Fall	2010	Chandler	804
2	747	1	Spring	2004	Gates	314
3	443	1	Spring	2010	Whitman	434
4	893	1	Fall	2007	Fairchild	145
5	663	1	Spring	2005	Fairchild	145
6	457	1	Spring	2001	Saucon	844
7	445	1	Spring	2001	Alumni	547
8	559	1	Fall	2002	Lamberton	134
9	239	1	Fall	2006	Taylor	183
10	802	1	Spring	2003	Saucon	113
11	158	1	Fall	2008	Whitman	434
12	735	1	Spring	2003	Drown	757
13	237	1	Spring	2008	Power	717
14	338	1	Spring	2007	Fairchild	145
15	376	1	Fall	2006	Power	717
16	927	1	Fall	2002	Saucon	180
17	692	1	Spring	2010	Taylor	183
18	867	1	Fall	2006	Taylor	183
19	561	1	Fall	2006	Main	45
20	604	1	Spring	2009	Bronfman	700
21	974	1	Fall	2003	Polya	808
22	959	1	Fall	2006	Saucon	180
23	702	1	Spring	2001	Saucon	113
24	631	1	Spring	2007	Taylor	183
25	334	1	Fall	2009	Taylor	812
26	274	1	Fall	2002	Main	425
27	366	1	Fall	2005	Saucon	844
28	545	1	Fall	2001	Saucon	180
29	486	1	Fall	2009	Whitman	134
30	426	1	Spring	2006	Lamberton	134
31	258	1	Fall	2007	Main	45
32	972	1	Spring	2009	Taylor	183
33	349	1	Spring	2008	Saucon	113
34	415	1	Fall	2010	Lamberton	134
35	137	1	Spring	2002	Fairchild	145
36	304	1	Fall	2009	Lamberton	143
37	468	1	Fall	2005	Lambeau	348
38	748	1	Fall	2003	Saucon	180
39	694	1	Fall	2002	Alumni	143
40	345	1	Spring	2008	Taylor	183
41	352	1	Spring	2006	Lambeau	348

Figure 2.3.3 : Une partie de la table "section"

3 Manipulation de la BDs

Dans cette dernière partie de notre projet, on va étudier 3 cas en s'inspirant du cours magistral et en donnant pour chaque cas la requête en SQL, l'équivalente en algèbre relationnel, l'explication de la requête et le résultat obtenu après l'exécution.

Etude de cas 1 : Inspiration du CM Slide 42 (Projection)

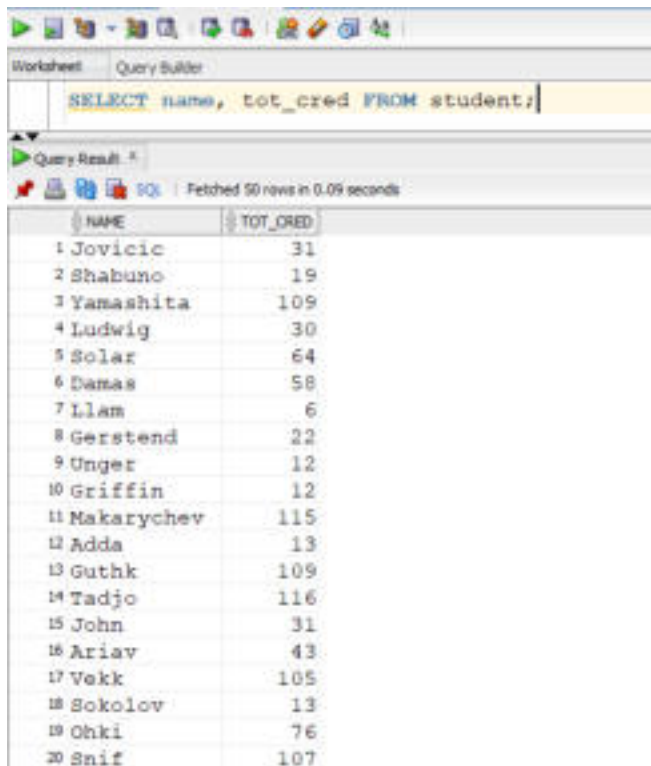
Requête en SQL : **SELECT** name, tot_cred **FROM** student;

Requête en AR : $\pi_{name, tot_cred}(student)$

Explication :

En termes d'AR, il s'agit d'une projection sur la table « student » en ne conservant que les attributs cités en opérande ici le nom et le crédit total de l'étudiant, qui est équivalent en SQL à SELECT name, tot_cred.

Résultat obtenue :

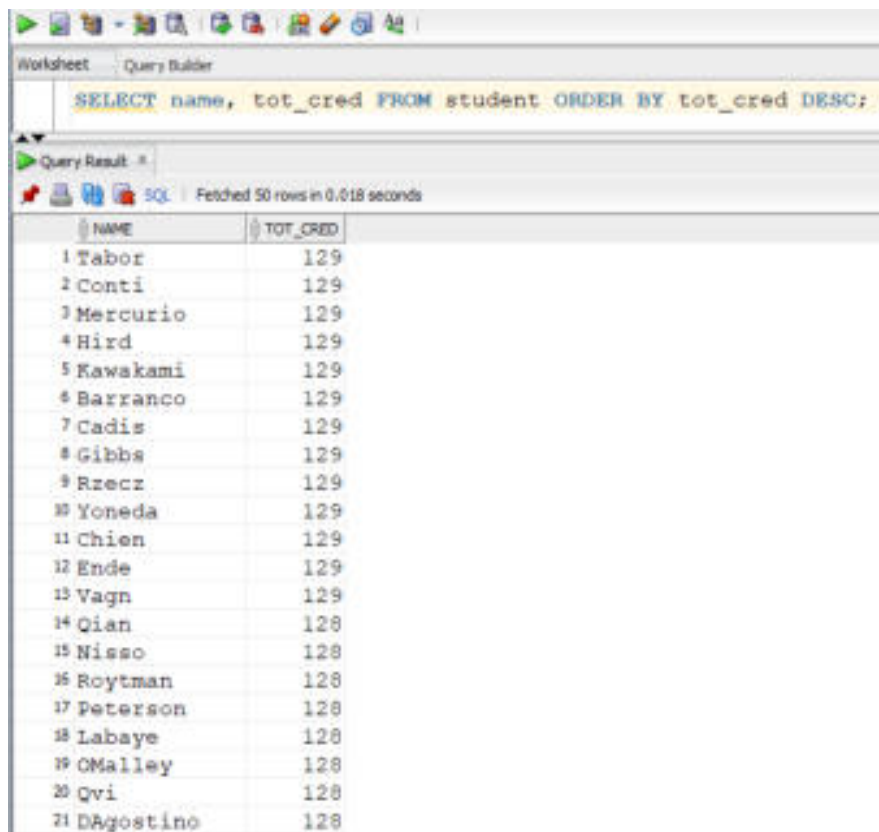


	NAME	TOT_CRED
1	Jovicic	31
2	Shabuno	19
3	Yamashita	109
4	Ludwig	30
5	Solar	64
6	Damas	58
7	Llam	6
8	Gerstend	22
9	Unger	12
10	Griffin	12
11	Makarychev	115
12	Adda	13
13	Guthk	109
14	Tadjo	116
15	John	31
16	Ariav	43
17	Vekk	105
18	Sokolov	13
19	Ohki	76
20	Snif	107

Figure 3.1 : Résultat de l'exécution de la requête

3 - Manipulation de la BDs – SQL DML et AR

On peut aussi ordonner le résultat précédent par ordre décroissant selon le crédit total de chaque étudiant à l'aide de ORDER BY :



The screenshot shows a SQL Query Builder window. The query text is: `SELECT name, tot_cred FROM student ORDER BY tot_cred DESC;`. The query result is displayed in a table with two columns: NAME and TOT_CRED. The results are ordered by TOT_CRED in descending order. The first 21 rows are shown, with the first 10 rows having a TOT_CRED of 129 and the remaining 11 rows having a TOT_CRED of 128.

	NAME	TOT_CRED
1	Tabor	129
2	Conti	129
3	Mercurio	129
4	Hird	129
5	Kawakami	129
6	Barranco	129
7	Cadis	129
8	Gibbs	129
9	Rzecz	129
10	Yoneda	129
11	Chien	129
12	Ende	129
13	Vagn	129
14	Qian	128
15	Nisso	128
16	Roytman	128
17	Peterson	128
18	Labaye	128
19	OMalley	128
20	Qvi	128
21	D'Agostino	128

Figure 3.2 : Résultat ordonné

Etude de cas 2 : Inspiration du CM Slide 55 (Sélection)

Requête en SQL :

`SELECT * FROM instructor WHERE salary > 50000 and department_dept_name = 'Statistics';`

Requête en AR :


$\sigma_{(salary > 50000) \wedge (department_dept_name \neq 'Statistics')}(instructor)$

Explication :

En AR c'est une sélection de la table « instructor » dont le prédicat est composé du salaire de l'instructeur qui est supérieur strictement à 50000 et au même temps l'instructeur doit appartenir au département de Statistiques, en SQL SELECT * va garder tous les attributs de la table « instructor » et le prédicat est cité après le mot clé WHERE.

3 - Manipulation de la BDs – SQL DML et AR

Résultat obtenue :



The screenshot shows a database query tool interface. The top pane displays the SQL query: `SELECT * FROM instructor WHERE (salary > 50000) AND (department_dept_name = 'Statistics');`. The bottom pane shows the query results in a table format. The table has four columns: ID, NAME, DEPARTMENT_DEPT_NAME, and SALARY. There are five rows of data.

ID	NAME	DEPARTMENT_DEPT_NAME	SALARY
178699	Pingr	Statistics	59303.62
237687	Arias	Statistics	104563.38
390643	Choll	Statistics	57807.09
495030	Azinb	Statistics	54805.11
528400	Atanassov	Statistics	84982.92

Figure 3.3 : Résultat de l'exécution de la requête

Etude de cas 3 : CM Partie 1, Séance 3 et 4, Page 10 (Jointure)

Requête en SQL :

```
SELECT department_dept_name, title, building, budget
FROM course JOIN department ON department_dept_name = dept_name
WHERE budget > 100000;
```

Requête en AR :

$\pi_{\text{department_dept_name, title, building, budget}} \left(\sigma_{\text{budget} > 100000} (\text{course} \bowtie \text{department}) \right)$

Explication :

En AR la requête représente une jointure naturelle qui va faire un produit cartésien des deux tables « course » et « student » et garder les tuples vérifiant la condition d'égalité des deux colonnes en communs (noms des départements), suivi par une sélection sur le résultat de jointure dont le prédicat est $\text{budget} > 100000$, enfin une projection sur le résultat de sélection en gardant que les attributs nom de département, titre de cours, le bâtiment et le budget.

En SQL la projection est faite en premier ligne après SELECT, la jointure en deuxième ligne avec la condition d'égalité des noms des départements est mets après ON, le dernier ligne contient le prédicat de sélection après WHERE.

Résultat obtenue :

3 - Manipulation de la BDs – SQL DML et AR

Worksheet Query Builder

```
SELECT department_dept_name, title, building, budget
FROM course JOIN department ON department_dept_name = dept_name WHERE budget > 100000;
```

Script Output Query Result

Fetched 50 rows in 0.004 seconds

	DEPARTMENT_DEPT_NAME	TITLE	BUILDING	BUDGET
1	Accounting	How to Groom your Cat	Saucon	441840.92
2	Accounting	Data Mining	Saucon	441840.92
3	Accounting	Image Processing	Saucon	441840.92
4	Accounting	Multivariable Calculus	Saucon	441840.92
5	Accounting	Existentialism	Saucon	441840.92
6	Accounting	Real-Time Database Systems	Saucon	441840.92
7	Accounting	Race Car Driving	Saucon	441840.92
8	Accounting	Astronautics	Saucon	441840.92
9	Accounting	Quantum Mechanics	Saucon	441840.92
10	Accounting	The Music of the Ramones	Saucon	441840.92
11	Accounting	Number Theory	Saucon	441840.92
12	Accounting	Bankruptcy	Saucon	441840.92
13	Astronomy	Sensor Networks	Taylor	617253.94
14	Astronomy	Journalism	Taylor	617253.94
15	Astronomy	Image Processing	Taylor	617253.94
16	Astronomy	Banking and Finance	Taylor	617253.94
17	Astronomy	Environmental Law	Taylor	617253.94
18	Astronomy	International Communication	Taylor	617253.94
19	Astronomy	Networking	Taylor	617253.94
20	Astronomy	Differential Geometry	Taylor	617253.94
21	Astronomy	The Monkeys	Taylor	617253.94
22	Astronomy	Recursive Function Theory	Taylor	617253.94
23	Athletics	International Finance	Bronfman	734550.7
24	Athletics	Sanitary Engineering	Bronfman	734550.7
25	Athletics	International Trade	Bronfman	734550.7
26	Athletics	Existentialism	Bronfman	734550.7
27	Athletics	C Programming	Bronfman	734550.7
28	Athletics	World History	Bronfman	734550.7
29	Athletics	Strength of Materials	Bronfman	734550.7
30	Athletics	Cat Herding	Bronfman	734550.7
31	Athletics	Aquatic Chemistry	Bronfman	734550.7
32	Biology	Virology	Candlestick	647610.55
33	Biology	Antidisestablishmentarianism in Modern America	Candlestick	647610.55
34	Biology	Composition and Literature	Candlestick	647610.55
35	Biology	Numerical Methods	Candlestick	647610.55

Figure 3.4 : Résultat de l'exécution de la requête