

13 장

데이터베이스 기반 앱 개발

Sejong Oh
Dankook University

목차

- 1. 개발 환경의 설정
- 2. 텍스트 모드 프로그래밍
- 3. GUI 프로그래밍
- 실습. 데이터베이스 GUI앱 개발

1. 개발 환경의 설정

● 개요

- 많은 경우 어플리케이션(앱)의 개발시 데이터베이스를 자료 관리의 기본 도구로 사용
- 앱 개발 언어로 파이썬을 선정하여 학습
- (C++, 자바 또는 JSP, PHP와 같은 웹 어플리케이션 개발 언어에서도 본 단원에서 배운 내용을 적용할 수 있음)

<표 13.1> 실습을 위해 필요한 요소들

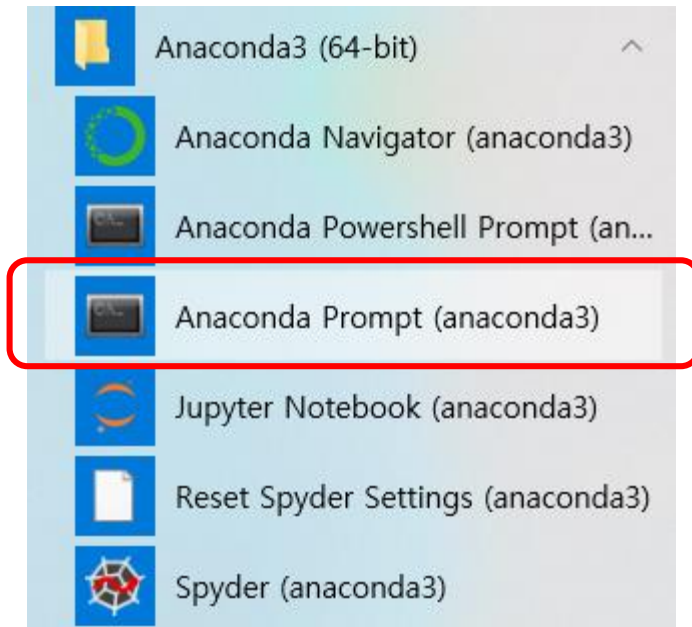
설치 요소	설명
오라클	DBMS
파이썬	앱 개발 언어
oracledb	오라클과 연동하기 위한 파이썬 라이브러리
PyQT5	GUI 프로그래밍을 하기 위한 파이썬 라이브러리 (아나콘다에 포함되어 있음)

* 아나콘다(anaconda)를 통해 파이썬을 설치 했다고 가정

1. 개발 환경의 설정

- **Oracledb의 설치**

- ① 윈도우 메뉴에서 다음과 같이 Anaconda Prompt를 선택한다.

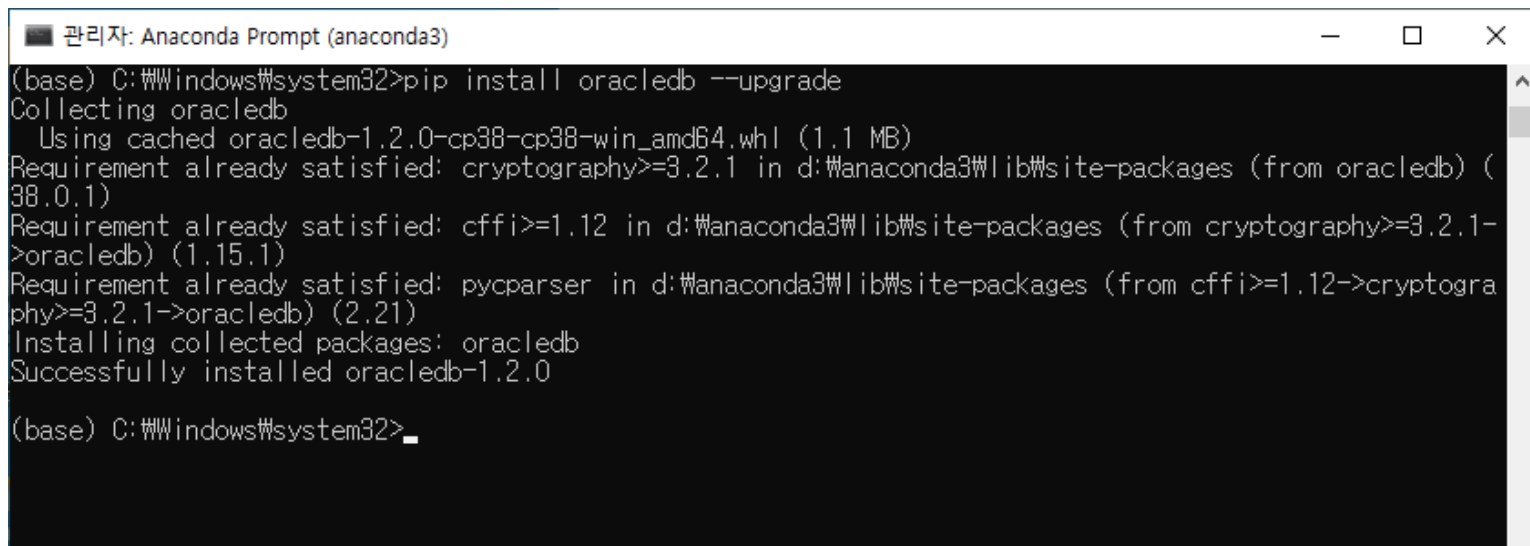


1. 개발 환경의 설정

- ② Anaconda Prompt 화면에서 다음의 명령어를 입력하여 실행한다.

```
pip install oracledb --upgrade
```

- ③ 다음과 같은 내용이 표시되면 정상 실행된 것이다.



```
관리자: Anaconda Prompt (anaconda3)
(base) C:\Windows\system32>pip install oracledb --upgrade
Collecting oracledb
  Using cached oracledb-1.2.0-cp38-cp38-win_amd64.whl (1.1 MB)
Requirement already satisfied: cryptography>=3.2.1 in d:\Wanaconda3\lib\site-packages (from oracledb) (38.0.1)
Requirement already satisfied: cffi>=1.12 in d:\Wanaconda3\lib\site-packages (from cryptography>=3.2.1->oracledb) (1.15.1)
Requirement already satisfied: pycparser in d:\Wanaconda3\lib\site-packages (from cffi>=1.12->cryptography>=3.2.1->oracledb) (2.21)
Installing collected packages: oracledb
Successfully installed oracledb-1.2.0

(base) C:\Windows\system32>_
```

1. 개발 환경의 설정

- **PyQT5 의 설치**

- PyQT5는 아나콘다를 설치하면 기본적으로 함께 설치가 되는 패키지중의 하나
- 아나콘다를 설치한 경우는 별도로 설치할 필요가 없다.
- 파이썬을 독립적으로 설치한 경우는 Command 콘솔에서 다음의 두 명령어를 순차적으로 실행하여 PyQT5를 설치

```
pip install pyqt5  
pip install pyqt5-tools
```

2. 텍스트 모드 프로그래밍

- (1) 오라클 접속 테스트

- 파이썬에서 오라클에 정상적으로 접속이 되는지를 테스트

```
1  import oracledb
2
3  # 필요한 기본 DB 정보
4  dsn = "localhost/emppdb" #접속할 db명
5  user = "scott"           #접속할 db의 user명
6  pw = "tiger"            #접속할 db의 password
7
8  # DB에 접속
9  conn = oracledb.connect(user = user, password = pw, dsn=dsn)
10 print(conn)
```

<코드 31-1> 오라클 접속 테스트

2. 텍스트 모드 프로그래밍

<코드설명>

```
1  import oracledb
```

오라클과 접속에 필요한 패키지를 로딩한다.

```
3  # 필요한 기본 DB 정보
```

```
4  dsn = "localhost/emppdb" #접속할 db명
```

```
5  user = "scott"           #접속할 db의 user명
```

```
6  pw = "tiger"            #접속할 db의 password
```

오라클 데이터베이스와 접속하기 위한 정보(데이터베이스명, 사용자ID, 비밀번호)를 각각 변수에 저장한다.

2. 텍스트 모드 프로그래밍

```
8  # DB에 접속
9  conn = oracledb.connect(user = user, password = pw, dsn=dsn)
```

오라클 데이터베이스에 접속하는 명령문이다. 성공하면 python 프로그램과 오라클 사이의 커뮤니케이션 통로가 열린다. 마치 전화를 걸어서 통화 연결이 된 것과 비슷하다.

```
10 print(conn)
```

접속결과를 출력한다. <코드 31-1>을 실행 했을 때 특별한 에러 메시지가 없고, print(conn) 실행시 다음과 같이 표시되면 정상적으로 접속이 된 것이다.

```
In [9]: print(conn)
<oracledb.Connection to scott@localhost/emppdb>
```

2. 텍스트 모드 프로그래밍

- (2) 사원정보 조회 (전체 조회)
 - emp 테이블의 내용을 파이썬에서 조회

```
1  import oracledb
2
3  # 필요한 기본 DB 정보
4  dsn = "localhost/emppdb" #접속할 db명
5  user = "scott"           #접속할 db의 user명
6  pw = "tiger"            #접속할 db의 password
7
8  # DB에 접속
9  conn = oracledb.connect(user = user, password = pw, dsn=dsn)
10
11 # 원하는 sql문 정의
12 sql = "SELECT * FROM emp WHERE ROWNUM <= 10"
13
```

2. 텍스트 모드 프로그래밍

```
14 # sql문 실행 / 데이터 받기
15 curs = conn.cursor()    # 커서 객체 생성
16 curs.execute(sql)       # SQL문 실행
17
18 data = curs.fetchall()  # sql 실행 결과 모두 가져오기
19 type(data)              # data 의 자료구조
20 data                    # data 의 내용출력
21 data[0]                 # 첫번째 행 출력
22 str(data[0][0])         # 첫번째 행의 첫번째 컬럼 출력
23
24 #db 접속 종료
25 curs.close()
26 conn.close()
```

<코드 13-2> 사원 정보 조회 (전체 결과 읽기)

2. 텍스트 모드 프로그래밍

<코드설명>

```
11  # 원하는 sql문 정의
12  sql = "SELECT * FROM emp WHERE ROWNUM <= 10"
```

emp 테이블을 읽는 SQL문을 작성하여 sql 변수에 저장한다. 튜플의 수가 많을 수 있으므로 10개만 읽도록 제한 하였다.

```
15  curs = conn.cursor()  # 커서 객체 생성
```

커서 객체를 생성한다. 커서 객체는 DBMS에게 작업을 요청하는데 사용되는 여러 함수를 가지고 있다.

```
16  curs.execute(sql)    # SQL문 실행
```

SQL문을 실행하도록 DBMS에게 요청한다. DBMS는 SQL문을 실행 하고 그 결과를 버퍼 (메모리) 에 저장한다.

2. 텍스트 모드 프로그래밍

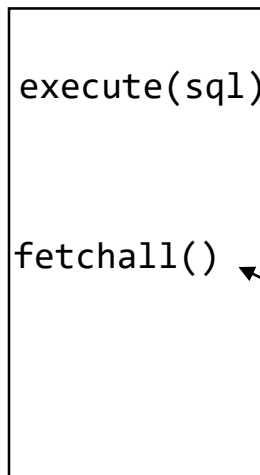
```
18 data = curs.fetchall() # sql 실행 결과 모두 가져오기
```

버퍼에 저장된 SQL문의 실행 결과를 프로그램 안으로 불러와서 data 변수에 저장한다.

버퍼에 저장된 결과를 프로그램으로 읽어오는 함수는 다음과 같이 두가지가 있다.

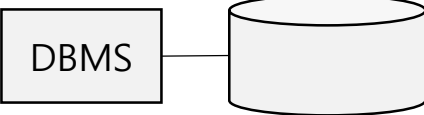
- fetchall() : 버퍼에 저장된 결과(10개의 튜플)를 한꺼번에 읽어온다.
- fetchone() : 버퍼에 저장된 결과로부터 하나의 튜플을 읽어온다.

파이썬 프로그램



① SQL문 실행 요청

MySQL



버퍼

EMPNO	ENAME	JOB	MGR	HIREDATE
7886	SMITH	CLERK	7902	1980-12-17
7900	ALLEN	SALESMAN	7908	1981-02-20
7912	WARD	SALESMAN	7908	1981-02-22
7967	SCOTT	ANALYST	7902	1982-07-06
7994	TURNER	SALESMAN	7908	1981-09-08
7800	BLAKE	MANAGER	7902	1981-05-01
7802	CLARK	MANAGER	7902	1981-06-09
7803	MILLER	CLERK	7902	1982-02-23
7804	JONES	MANAGER	7902	1981-04-02
7809	COOPER	CLERK	7902	1982-03-17
7811	ADAMS	CLERK	7902	1981-12-13
7820	MARTIN	SALESMAN	7908	1981-02-12
7821	WATSON	CLERK	7902	1981-12-21

② 실행 결과를 버퍼에 저장

③ 버퍼에 저장된 결과를 읽는다

2. 텍스트 모드 프로그래밍

```
19 type(data)          # data 의 자료구조
```

data 변수의 자료구조를 살펴본다. 파이썬 자료형은 리스트(list)이다.

```
In [14]: type(data)          # data 의 자료구조
Out[14]: list
```

```
20 data                # data 의 내용출력
```

data 변수의 내용을 출력해 본다. emp 테이블의 한 행(row)이 파이썬 튜플의 한 원소로 저장된다.

```
In [15]: data              # data 의 내용출력
Out[15]:
[(7839,
  'KING',
  'PRESIDENT',
  None,
  datetime.datetime(1981, 11, 17, 0, 0),
  5000.0,
  None,
  10),
 (7698,
  'BLAKE',
  'MANAGER',
  7839,
  datetime.datetime(1981, 5, 1, 0, 0),
  2850.0,
  None,
  30),
 (7782,
```

2. 텍스트 모드 프로그래밍

21	<code>data[0]</code>	# 첫번째 행 출력
----	----------------------	------------

`data` 변수에서 첫 번째 원소(실행 결과의 첫 번째 행)의 내용을 출력한다. 세 번째 행의 내용을 보고 싶으면 `data[2]` 와 같이 한다.

```
In [16]: data[0]                                # 첫번째 행 출력
Out[16]:
(7839,
 'KING',
 'PRESIDENT',
 None,
 datetime.datetime(1981, 11, 17, 0, 0),
 5000.0,
 None,
 10)
```

2. 텍스트 모드 프로그래밍

```
22 str(data[0][0]) # 첫번째 행의 첫번째 컬럼 출력
```

data 변수로부터 실행 결과의 첫 번째 행의 첫 번째 컬럼의 내용을 출력한다.

```
In [17]: str(data[0][0]) # 첫번째 행의 첫번째 컬럼 출력
Out[17]: '7839'
```

```
24 #db 접속 종료
```

```
25 curs.close()
```

```
26 conn.close()
```

커서 객체의 사용 및 DBMS와의 접속을 종료한다.

2. 텍스트 모드 프로그래밍

- (3) 사원정보 조회 (한행씩 조회)
 - 버퍼의 결과를 한번에 한행씩 가져오는 예제

```
1  ... (<코드 13-2>의 1~11행과 동일)
2
3  # 원하는 sql문 정의
4  sql = "SELECT * FROM emp WHERE ROWNUM <= 10"
5
6  #sql문 실행 / 데이터 받기
7  curs = conn.cursor()
8  curs.execute(sql)
9
10 row = curs.fetchone()          # 버퍼에서 하나의 행 읽기
11 while(row):
12     print(row)
13     row = curs.fetchone()       # 버퍼에서 하나의 행 읽기
14
```

2. 텍스트 모드 프로그래밍

15	#db 접속 종료
16	curs.close()
17	conn.close()

<코드 13-3> 사원 정보 조회 (한번에 한 행씩 읽기)

2. 텍스트 모드 프로그래밍

<코드설명>

```
10 row = curs.fetchone() # 버퍼에서 하나의 행 읽기
```

버퍼의 저장된 결과에서 첫번째 행을 읽어서 변수 row에 저장한다. 그리고 데이터를 읽을 위치가 다음번 행으로 이동한다. (fetchone()은 버퍼에서 한 행의 정보를 읽어온다.)

```
11 while(row):  
12     print(row)  
13     row = curs.fetchone() # 버퍼에서 하나의 행 읽기
```

버퍼의 결과로부터 모든 행을 다 읽을 때까지 반복문 while()을 수행한다. 변수 row에 읽어온 값이 없으면 반복문은 종료
반복문 안의 내용은 읽어온 row의 내용을 출력하고 버퍼에서 다음번 행을 읽어서 row에 저장하는 것

2. 텍스트 모드 프로그래밍

반복문 실행 결과 (앞쪽 일부)

```
In [26]: while(row):
...:     print(row)
...:     row = curs.fetchone()          # 버퍼에서 하나의 행 읽기
# 첫번째 행의 첫번째 컬럼 출력
(7839, 'KING', 'PRESIDENT', None, datetime.datetime(1981, 11, 17, 0, 0),
5000.0, None, 10)
(7698, 'BLAKE', 'MANAGER', 7839, datetime.datetime(1981, 5, 1, 0, 0),
2850.0, None, 30)
(7782, 'CLARK', 'MANAGER', 7839, datetime.datetime(1981, 6, 9, 0, 0),
2450.0, None, 10)
(7566, 'JONES', 'MANAGER', 7839, datetime.datetime(1981, 4, 2, 0, 0),
2975.0, None, 20)
(7654, 'MARTIN', 'SALESMAN', 7698, datetime.datetime(1981, 8, 28, 0, 0),
1250.0, 1400.0, 30)
(7499, 'ALLEN', 'SALESMAN', 7698, datetime.datetime(1981, 2, 20, 0, 0),
1600.0, 300.0, 30)
```

2. 텍스트 모드 프로그래밍

- (4) 부서정보 입력

- 파이썬 프로그램에서 데이터를 테이블에 저장하는 예제
- dept 테이블에 새로운 부서정보를 입력한다.

```
1  ... (<코드 13-2>의 1~11행과 동일)
2
3  # input values
4  deptno = 60
5  dname = "DEVELOP"
6  loc = "SEOUL"
7
8  # 원하는 sql문 정의
9  sql = "INSERT INTO dept VALUES (:1, :2, :3)"
10 vals = (deptno, dname, loc)
11
```

2. 텍스트 모드 프로그래밍



```
12  #sql문 실행
13  curs = conn.cursor()
14  curs.execute(sql, vals)
15  conn.commit()
16
17  #db 접속 종료
18  curs.close()
19  conn.close()
```

<코드 13-4> 부서정보 입력

2. 텍스트 모드 프로그래밍

<코드설명>

```
3  # input values
4  deptno = 60
5  dname = "DEVELOP"
6  loc = "SEOUL"
```

dept 테이블에 저장할 데이터를 준비한다. 각 컬럼별 데이터를 각각의 변수에 저장하였다.

```
9  sql = "INSERT INTO dept VALUES (:1, :2, :3)"
```

요청할 SQL문을 정의 한다. SQL문의 :1, :2, :3는 이 위치에 어떤 값이 입력될 것임을 표시하는 역할을 한다.

2. 텍스트 모드 프로그래밍

```
10 vals = (deptno, dname, loc)
```

dept 테이블에 입력할 값들을 파이썬 튜플 자료구조를 이용해 하나로 묶는다. 입력된 값들의 순서에 주의하여 묶어주도록 한다.

```
12 #sql문 실행
```

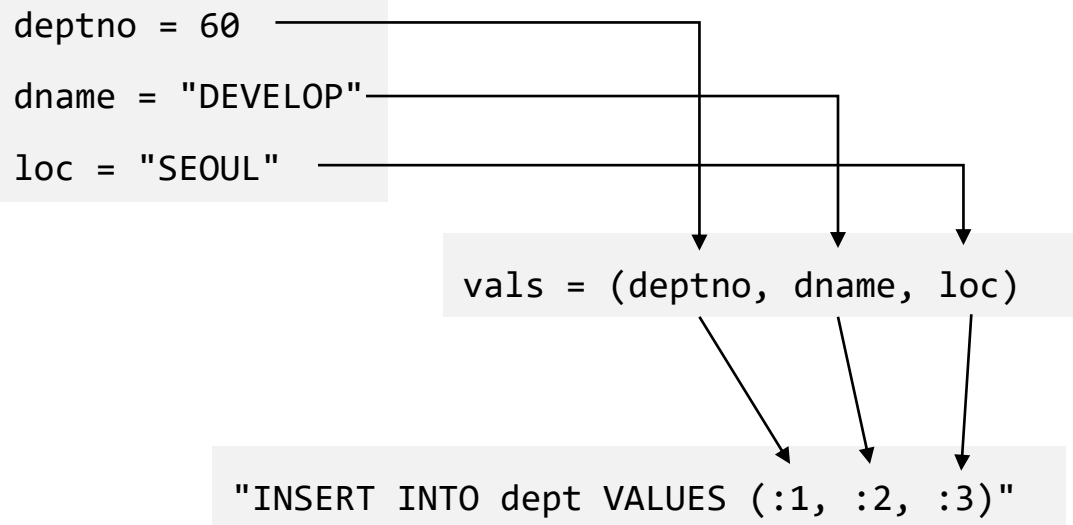
```
13 curs = conn.cursor()
```

커서 객체를 생성한다.

```
14 curs.execute(sql, vals)
```

SQL문의 실행을 DBMS에게 요청한다. 이때 SQL문의 :1, :2, :3 자리에 vals에 저장된 값들이 차례로 입력된다. (다음 슬라이드 그림 참조)

2. 텍스트 모드 프로그래밍



2. 텍스트 모드 프로그래밍

```
15 conn.commit()
```

SQL문의 실행 결과를 데이터베이스에 영구 저장하도록 요청한다. 그 결과 dept 테이블에는 새로운 행이 추가된다.

	DEPTNO	DNAME	LOC
1	10	ACCOUNTING	NEW YORK
2	20	RESEARCH	DALLAS
3	30	SALES	CHICAGO
4	40	OPERATIONS	BOSTON
5	60	DEVELOP	SEOUL

```
17 #db 접속 종료
```

```
18 curs.close()
```

```
19 conn.close()
```

DB접속을 종료하고 작업을 마친다.

3. GUI 프로그래밍 예제

- 개요

- 사원ID를 입력하고 [Query] 버튼을 클릭하면 emp 테이블에서 해당 사원의 정보를 읽어와서 윈도우 화면상에 보여주는 예제
- [Exit] 버튼을 클릭하면 프로그램을 종료한다.

The screenshot shows a window titled "My Program" with standard Windows window controls (minimize, maximize, close). Inside the window, there are four input fields for employee information, each with a label to its left:

- ID**: The input field contains the text "7839". To its right is a button labeled "Query".
- ename**: The input field contains the text "KING". To its right is a button labeled "Exit".
- job**: The input field contains the text "PRESIDENT".
- department**: The input field contains the text "ACCOUNTING".

3. GUI 프로그래밍 예제

윈도우 디자인 및
동작 정의

7	def connectDB():	→ 데이터베이스 커넥션 구현
18	def disconnectDB(conn):	→ 데이터베이스 커넥션 종료
22	class MyApp(QWidget):	
23	def __init__(self):	→ 윈도우 실행시 초기화 내용 구현
28	def initUI(self):	→ 사용자 인터페이스 정의 (화면 디자인)
73	def btn_1_clicked(self):	→ [Query] 버튼이 클릭되었을 때 실행할 내용
96	# END Class #####	
99	if (__name__ == '__main__'):	→ 윈도우의 시작과 종료 관련 함수 호출

<그림 13-2> 사원정보 조회 GUI 프로그램의 구조

3. GUI 프로그래밍 예제

```
1  import sys
2  import oracledb
3  from PyQt5.QtWidgets import *
4  from PyQt5.QtCore import QApplication
5
6  # 데이터베이스 연결 함수
7  def connectDB():
8      # DB 접속에 필요한 기본정보
9      dsn = "localhost/emppdb" #접속할 db명
10     user = "scott" #접속할 db의 user명
11     pw = "tiger" #접속할 db의 password
12
13     # DB에 접속
14     conn = oracledb.connect(user = user, password = pw, dsn=dsn)
15     return(conn)
16
```

3. GUI 프로그래밍 예제

```
17 # 데이터베이스 연결 해제 함수
18 def disconnectDB(conn):
19     conn.close()
20
21 # 윈도우 생성 클래스 #####
22 class MyApp(QWidget):
23     def __init__(self):
24         super().__init__()
25         self.initUI()
26
27     # UI 디자인 함수
28     def initUI(self):
29         label1 = QLabel('ID')
30         label2 = QLabel('ename')
31         label3 = QLabel('job')
32         label4 = QLabel('department')
```

3. GUI 프로그래밍 예제

```
34     self.text_id = QTextEdit()
35     self.text_id.setFixedWidth(200)      # 텍스트 박스의 폭
36     self.text_id.setFixedHeight(30)     # 텍스트 박스의 높이
37     btn_1 = QPushButton('Query')
38     btn_1.clicked.connect(self.btn_1_clicked)
39
40     btn_2 = QPushButton('Exit', self)
41     btn_2.clicked.connect(self.close)
42     btn_2.clicked.connect(QCoreApplication.instance().quit)
43
44     self.text_ename = QTextEdit()
45     self.text_ename.setFixedWidth(200)
46     self.text_ename.setFixedHeight(30)
47     self.text_job = QTextEdit()
48     self.text_job.setFixedWidth(200)
49     self.text_job.setFixedHeight(30)
```

3. GUI 프로그래밍 예제

```
51     self.text_dept = QTextEdit()
52     self.text_dept.setFixedWidth(200)
53     self.text_dept.setFixedHeight(30)
54
55     gbox = QGridLayout()
56     gbox.addWidget(label1, 0, 0)
57     gbox.addWidget(self.text_id, 0, 1)
58     gbox.addWidget(btn_1, 0, 2)
59     gbox.addWidget(btn_2, 1, 2)
60     gbox.addWidget(label2, 1, 0)
61     gbox.addWidget(self.text_ename, 1, 1)
62     gbox.addWidget(label3, 2, 0)
63     gbox.addWidget(self.text_job, 2, 1)
64     gbox.addWidget(label4, 3, 0)
65     gbox.addWidget(self.text_dept, 3, 1)
66
```


3. GUI 프로그래밍 예제

```
67         self.setLayout(gbox)
68         self.setWindowTitle('My Program')
69         self.setGeometry(300,300, 480,250)
70         self.show()
71
72         # 버튼 클릭시 처리 함수
73         def btn_1_clicked(self):
74
75             empno = [self.text_id.toPlainText()]
76
77             # sql쿼리 문
78             sql = "SELECT ename, job, dname \
79                   FROM emp e, dept d \
80                   WHERE e.deptno = d.deptno \
81                   AND empno = :1"
82
```

3. GUI 프로그래밍 예제

```
83     conn = connectDB()
84     curs = conn.cursor()
85     curs.execute(sql, empno)
86
87     result = curs.fetchone() # sql 실행 결과 가져오기
88
89     self.text_ename.setText(result[0])
90     self.text_job.setText(result[1])
91     self.text_dept.setText(result[2])
92
93     curs.close()
94     disconnectDB(conn)
95
96 # END Class #####
97
```

3. GUI 프로그래밍 예제

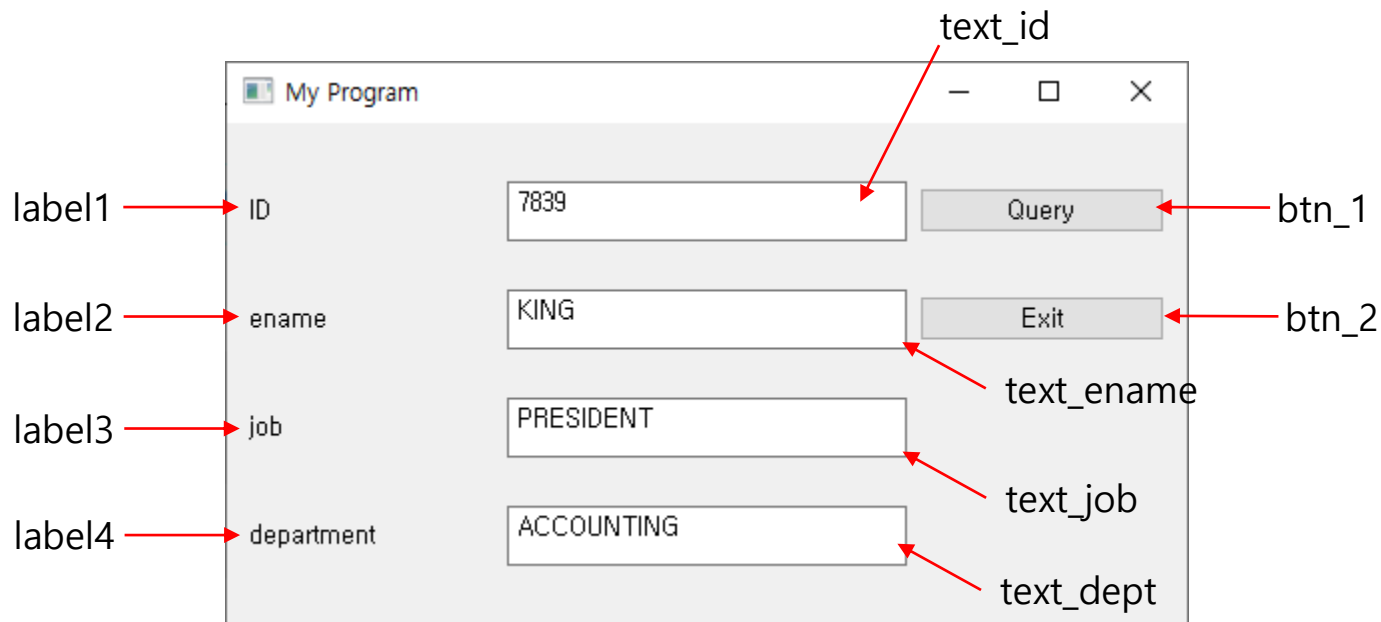
```
98  # 프로그램 실행
99  if (__name__ == '__main__'):
100  app = QApplication(sys.argv)
101  ex = MyApp()
102  sys.exit(app.exec_())
```

<코드 13-5> 사원정보 조회 GUI 프로그램

3. GUI 프로그래밍 예제

<코드설명>

- 사원정보 조회 GUI 프로그램에서 핵심은 사용자 인터페이스를 정의하는 `initUI()` 함수와 버튼 클릭 이벤트를 처리하는 `btn_1_clicked()` 함수
- `initUI()` 함수에서는 윈도우에 표시될 레이블, 텍스트 에디트, 버튼 (이를 위젯이라고 한다) 등을 정의하고 이를 윈도우상의 적절한 위치에 배치

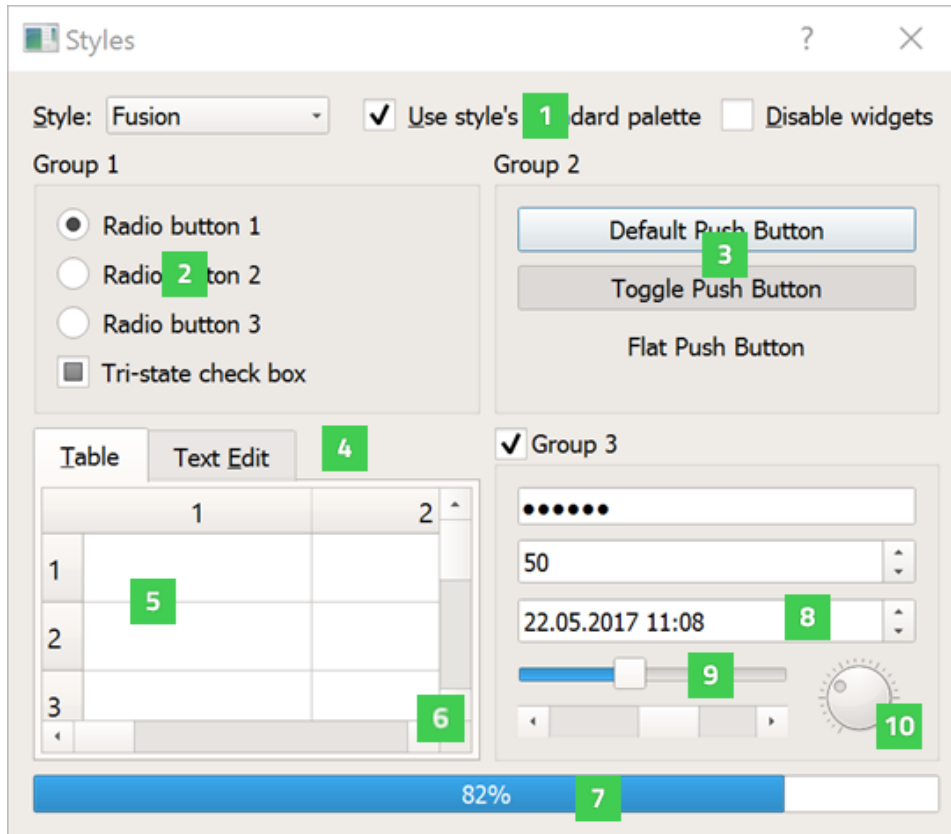


3. GUI 프로그래밍 예제

- 라인 29~53 : 레이블, 텍스트 에디트, 버튼 위젯의 정의
- 라인 55~65 : 정의한 위젯을 윈도우의 적절한 위치에 배치
- 라인 67~70 : 윈도우의 타이틀, 크기, 화면 표시 위치를 정의

3. GUI 프로그래밍 예제

- pyqt5 위젯(widget)



- (1) QCheckBox
- (2) QRadioButton
- (3) QPushButton
- (4) QTabWidget
- (5) QTableWidget
- (6) QScrollBar
- (7) QProgressBar
- (8) QDateTimeEdit
- (9) QSlider
- (10) QDial

(출처: <https://doc.qt.io/qt-6/gallery.html>)

3. GUI 프로그래밍 예제

- 버튼 클릭 이벤트의 처리

```
38      btn_1.clicked.connect(self.btn_1_clicked)
```

btn_1 버튼을 클릭하면 btn_1_clicked() 함수가 실행되도록 지정됨

- btn_1_clicked() 함수 (Line 73~96)

```
75      empno = self.text_id.toPlainText()
```

사용자가 text_id에 입력한 값을 추출하여 텍스트 형태로 empno 변수에 저장한다.

```
77      # sql쿼리 문
78      sql = "SELECT ename, job, dname \
79      FROM emp e, dept d \
80      where e.deptno = d.deptno \
81      and empno = :1"
```

DBMS에게 요청할 SQL문을 정의 한다. 입력한 사원번호에 대해 사원의 이름, 담당업무, 부서명을 조회한다.

3. GUI 프로그래밍 예제

```
83      conn = connectDB()
84      curs = conn.cursor()
85      curs.execute(sql, empno)
```

DBMS에 연결을 하고 커서 객체를 생성한다. 그리고 정의한 SQL문의 실행을 요청한다. 이때 SQL문의 %s 를 empno 변수의 내용으로 대체한다.

```
87      result = curs.fetchone() # sql 실행 결과 가져오기
```

SQL문의 실행 결과를 버퍼에서 가져온다. 결과가 하나의 행이므로 fetchone() 함수를 이용하였다.

```
89      self.text_ename.setText(result[0])
90      self.text_job.setText(result[1])
91      self.text_dept.setText(result[2])
```

가져온 결과 행으로부터 컬럼들을 잘라서 각각 text_ename, text_job, text_dept에 저장한다. 그러면 윈도우 위에 사원이름, 담당업무, 부서이름이 표시된다.

3. GUI 프로그래밍 예제

```
93     curs.close()  
94     disconnectDB(conn)
```

작업이 끝났으므로 커서와 데이터베이스 연결을 종료한다.

- 원도우의 종료

```
41 btn_2.clicked.connect(self.close)  
42 btn_2.clicked.connect(QCoreApplication.instance().quit)
```

원도우를 종료 하기 위해서는 btn_2 버튼을 클릭하면 된다.

실습. 데이터베이스 GUI 앱 개발

- 과제

- emppdb의 country 테이블과 city 테이블로부터 국가를 선택하면 해당 국가의 도시명, 인구수를 화면에 표시하는 GUI 프로그램을 작성

City info

국가명 선택 Austria

Query

	도시명	인구수
1	Wien	1608144
2	Graz	240967
3	Linz	188022
4	Salzburg	144247
5	Innsbruck	111752
6	Klagenfurt	91141

Exit

3. GUI 프로그래밍 예제

```
import sys
import oracledb
from PyQt5.QtWidgets import *
from PyQt5.QtCore import QApplication
from PyQt5.QtCore import Qt

# 데이터베이스 연결 함수 #####
def connectDB():
    #필요한 기본 DB 정보
    dsn = "localhost/emppdb" #접속할 db명
    user = "scott" #접속할 db의 user명
    pw = "tiger" #접속할 db의 password

    # DB에 접속
    conn = oracledb.connect(user = user, password = pw, dsn=dsn)
    return(conn)
```

3. GUI 프로그래밍 예제

```
# 데이터베이스 연결 해제 함수 #####  
def disconnectDB(conn):  
    conn.close()  
  
class MyApp(QWidget):  
    def __init__(self):  
        super().__init__()  
        self.initUI()  
  
    # UI 디자인 함수 #####  
    def initUI(self):  
        label1 = QLabel('국가명 선택')  
        self.contry = QComboBox()  
        self.contry.setFixedWidth(200)  
        self.contry.setFixedHeight(30)
```

3. GUI 프로그래밍 예제

```
# combobox에 국가 리스트 입력
sql = "SELECT name \
FROM country "

conn = connectDB()
curs = conn.cursor()
curs.execute(sql)

cname = curs.fetchone()
while(cname):
    self.contry.addItem(cname[0])
    cname = curs.fetchone()

curs.close()
disconnectDB(conn)
```

3. GUI 프로그래밍 예제

```
btn_1 = QPushButton('Query')
btn_1.clicked.connect(self.btn_1_clicked)

btn_2 = QPushButton('Exit', self)
btn_2.clicked.connect(self.close)
btn_2.clicked.connect(QCoreApplication.instance().quit)

self.city_info = QTableWidget()

# 생성한 위젯 배치
gbox = QGridLayout()
gbox.addWidget(label1, 0, 0)
gbox.addWidget(self.contry, 0, 1)
gbox.addWidget(btn_1, 0, 2)
gbox.addWidget(btn_2, 1, 2)
```

3. GUI 프로그래밍 예제

```
gbox.addWidget(self.city_info, 1, 1)

self.setLayout(gbox)
self.setWindowTitle('City info')
self.setGeometry(300,300, 550,300)
self.show()

# 버튼 클릭시 처리 함수 #####
def btn_1_clicked(self):
    country_name = self.contry.currentText()

    # sql쿼리 문
    sql = "SELECT city.name, city.population \
FROM city , country \
WHERE city.countrycode = country.code \
AND country.name = :var1"
```

3. GUI 프로그래밍 예제

```
conn = connectDB()
curs = conn.cursor()
curs.execute(sql, var1=country_name)

self.city_info.setColumnCount(2)
self.city_info.setHorizontalHeaderItem(0,
                                     QTableWidgetItem("도시명"))
self.city_info.setHorizontalHeaderItem(1,
                                     QTableWidgetItem("인구수"))

result = curs.fetchone()
i = 0                                # no of row
while(result):
    rowPosition = self.city_info.rowCount()
    self.city_info.insertRow(rowPosition)
```


3. GUI 프로그래밍 예제

```
        self.city_info.setItem(i, 0, QTableWidgetItem(result[0]))
        pop = QTableWidgetItem(str(result[1])) # create the item
        pop.setTextAlignment(Qt.AlignRight) # change the alignment
        self.city_info.setItem(i, 1, pop)
        result = curs.fetchone()
        i = i+1

    curs.close()
    disconnectDB(conn)
# END Class
# 프로그램 실행
if (__name__ == '__main__'):
    app = QApplication(sys.argv)
    ex = MyApp()
    sys.exit(app.exec_())
```

3. GUI 프로그래밍 예제

