1.4. SQLite

■ What is SQLite?

- ○ Open Source Database(Free)
- ○ Serverless(Direct I/O)
- ○ Self-contained(Embedded)
- ○ Single Disk File(Cross-platform)
- ○ Zero-configuration(No setup, No server)
- ○ Supports RDBMS Features(ACID, SQL Syntax, Transactions, etc)

http://sqlitebrowser.org

■ SQLite 확인

```
import sqlite3
```

`sqlite3.`**`version`**

The version number of this module, as a string. This is not the version of the SQLite library.

`sqlite3.`**`version_info`**

The version number of this module, as a tuple of integers. This is not the version of the SQLite library.

`sqlite3.`**`sqlite_version`**

The version number of the run-time SQLite library, as a string.

`sqlite3.`**`sqlite_version_info`**

The version number of the run-time SQLite library, as a tuple of integers.

■ Database 연결(생성)

```
conn = sqlite3.connect('경로/이름' or ':memory:')
```

sqlite3.**connect**(*database*[, *timeout*, *detect_types*, *isolation_level*, *check_same_thread*, *factory*, *cached_statements*, *uri*])

Opens a connection to the SQLite database file *database*. You can use `":memory:"` to open a database connection to a database that resides in RAM instead of on disk.

When a database is accessed by multiple connections, and one of the processes modifies the database, the SQLite database is locked until that transaction is committed. The *timeout* parameter specifies how long the connection should wait for the lock to go away until raising an exception. The default for the timeout parameter is 5.0 (five seconds).

■ Cursor 생성

```
cur = conn.cursor()
```

cursor(*factory=Cursor*)

The cursor method accepts a single optional parameter *factory*. If supplied, this must be a callable returning an instance of `Cursor` or its subclasses.

```
type(cur)
dir(cur)
```

```
'__setattr__',
'__sizeof__',
'__str__',
'__subclasshook__',
'arraysize',
'close',
'connection',
'description',
'execute',
'executemany',
'executescript',
'fetchall',
'fetchmany',
'fetchone',
'lastrowid',
'row_factory',
'rowcount',
'setinputsizes',
'setoutputsize']
```

■ Data Type

SQLite natively supports the following types: `NULL`, `INTEGER`, `REAL`, `TEXT`, `BLOB`.

Sr.No.	Storage Class & Description
1	**NULL** The value is a NULL value.
2	**INTEGER** The value is a signed integer, stored in 1, 2, 3, 4, 6, or 8 bytes depending on the magnitude of the value.
3	**REAL** The value is a floating point value, stored as an 8-byte IEEE floating point number.
4	**TEXT** The value is a text string, stored using the database encoding (UTF-8, UTF-16BE or UTF-16LE)
5	**BLOB** The value is a blob of data, stored exactly as it was input.

execute

```
cur.execute('SQL')
```

execute(*sql*[, *parameters*])

Executes an SQL statement. The SQL statement may be parameterized (i. e. placeholders instead of SQL literals). The `sqlite3` module supports two kinds of placeholders: question marks (qmark style) and named placeholders (named style).

```
cur.execute("create table people (name_last, age)")

who = "Yeltsin"
age = 72

# This is the qmark style:
cur.execute("insert into people values (?, ?)", (who, age))

# And this is the named style:
cur.execute("select * from people where name_last=:who and age=:age", {"who": who, "age": age})

print(cur.fetchone())
```

executemany

```
cur.executemany('SQL', params)
```

executemany(*sql*, *seq_of_parameters*)

Executes an SQL command against all parameter sequences or mappings found in the sequence *seq_of_parameters*. The `sqlite3` module also allows using an iterator yielding parameters instead of a sequence.

```
sql = "insert into people values (?, ?)"
curData = [('A', 1), ('B', 2), ('C', 3)]

cur.executemany(sql, curData)
```

executescript

```
cur.executescript('''SQL1; SQL2; ...''')
```

executescript(*sql_script*)

This is a nonstandard convenience method for executing multiple SQL statements at once. It issues a `COMMIT` statement first, then executes the SQL script it gets as a parameter.

```
cur.executescript("""
    create table person (
        first_name text primary key,
        last_name  text not null
    );

    insert into person values ('name', 'kim');
""")
```

```
cur.execute('select * from person')
print(cur.fetchall())
```

■ fectchone

```
cur.fetchone()
```

fetchone()

Fetches the next row of a query result set, returning a single sequence, or `None` when no more data is available.

■ fetchmanay

```
cur.fetchmany(size)
```

fetchmany(*size=cursor.arraysize*)

Fetches the next set of rows of a query result, returning a list. An empty list is returned when no more rows are available.

The number of rows to fetch per call is specified by the *size* parameter. If it is not given, the cursor's arraysize determines the number of rows to be fetched. The method should try to fetch as many rows as indicated by the size parameter. If this is not possible due to the specified number of rows not being available, fewer rows may be returned.

Note there are performance considerations involved with the *size* parameter. For optimal performance, it is usually best to use the arraysize attribute. If the *size* parameter is used, then it is best for it to retain the same value from one `fetchmany()` call to the next.

■ fectchall

```
cur.fetchall()
```

fetchall()

Fetches all (remaining) rows of a query result, returning a list. Note that the cursor's arraysize attribute can affect the performance of this operation. An empty list is returned when no rows are available.

■ DDL

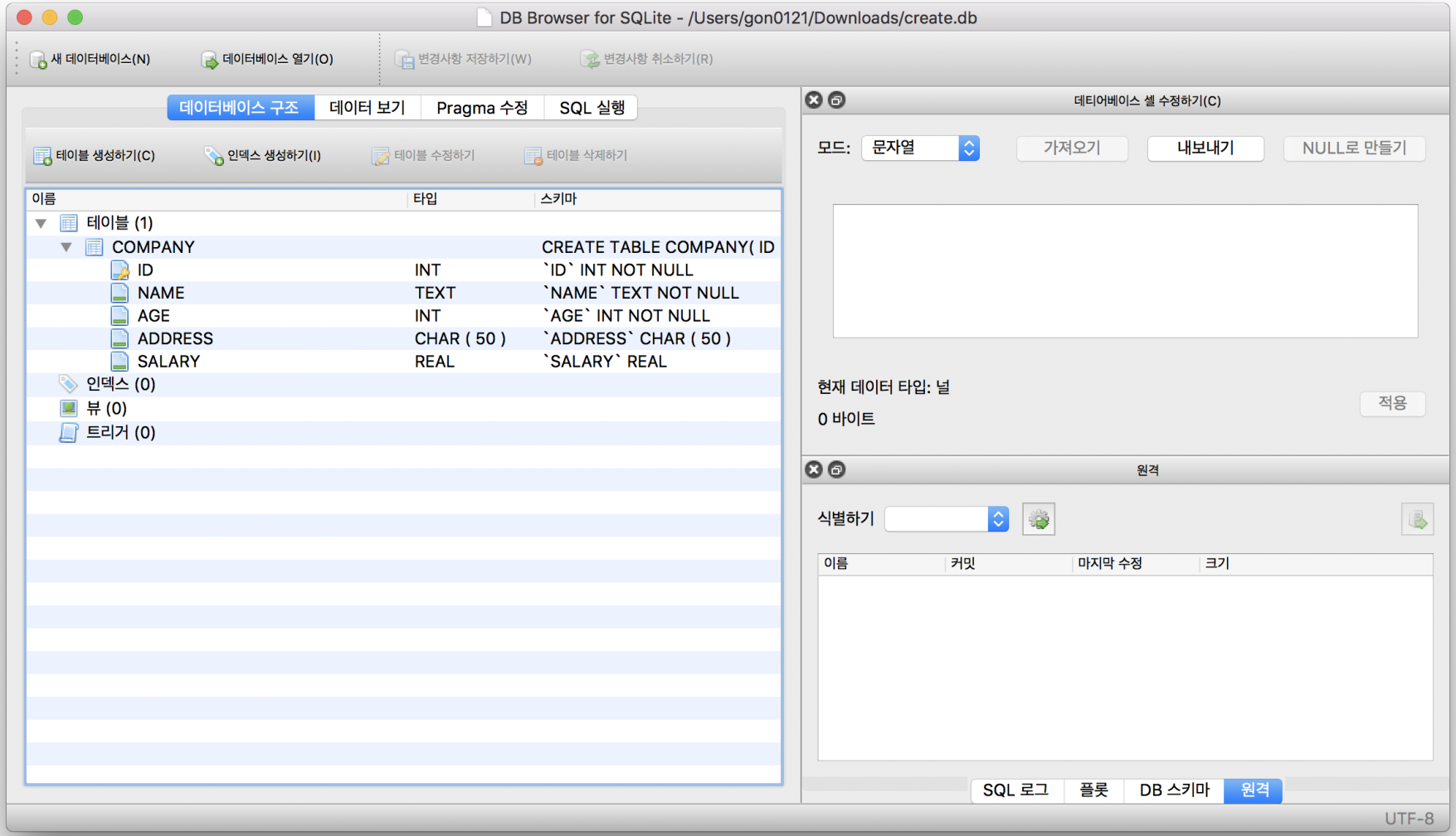
○ CREATE

```
conn = sqlite3.connect('create.db')
print("Opened database successfully")

conn.execute('''
    CREATE TABLE COMPANY(
        ID INT PRIMARY KEY     NOT NULL,
        NAME           TEXT    NOT NULL,
        AGE            INT     NOT NULL,
        ADDRESS        CHAR(50),
        SALARY         REAL);
    ''')

print("Table created successfully")
```

○ DROP

```
conn.execute('drop table company')
```

○ CREAT 확인

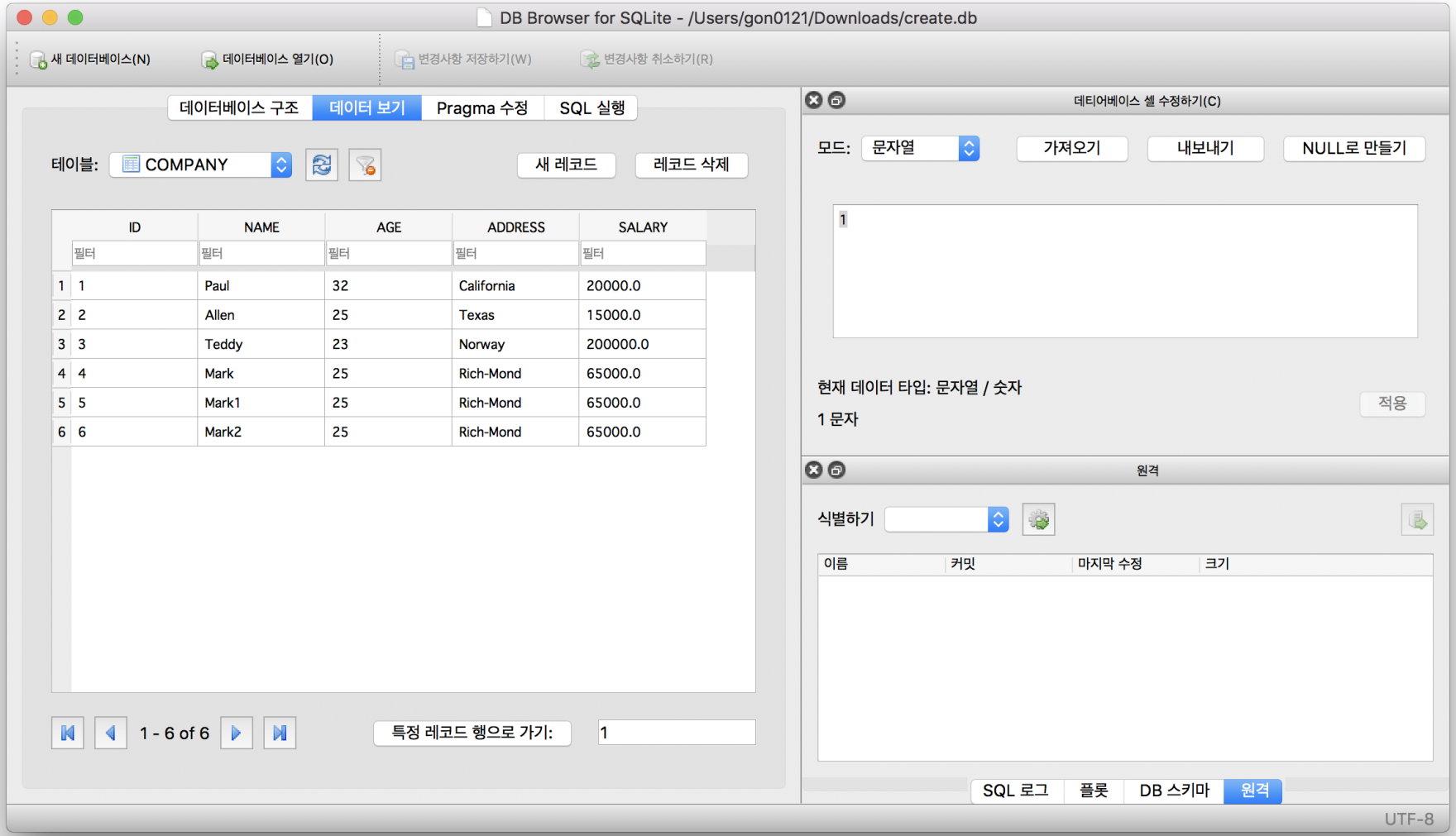

DB Browser for SQLite - /Users/gon0121/Downloads/create.db
새 데이터베이스(N)
데이터베이스 열기(O)
변경사항 저장하기(W)
변경사항 취소하기(R)
데이터베이스 구조
데이터 보기
Pragma 수정
SQL 실행
테이블 생성하기(C)
인덱스 생성하기(I)
테이블 수정하기
테이블 삭제하기
이름
타입
스키마
테이블 (1)
COMPANY
CREATE TABLE COMPANY(ID
ID
INT
`ID` INT NOT NULL
NAME
TEXT
`NAME` TEXT NOT NULL
AGE
INT
`AGE` INT NOT NULL
ADDRESS
CHAR (50)
`ADDRESS` CHAR (50)
SALARY
REAL
`SALARY` REAL
인덱스 (0)
뷰 (0)
트리거 (0)
데티어베이스 셀 수정하기(C)
모드:
문자열
가져오기
내보내기
NULL로 만들기
현재 데이터 타입: 널
0 바이트
적용
원격
식별하기
이름
커밋
마지막 수정
크기
SQL 로그
플롯
DB 스키마
원격
UTF-8

■ DML

○ INSERT

```
conn.execute("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \
              VALUES (1, 'Paul', 32, 'California', 20000.00 )");

conn.execute("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \
              VALUES (:id, :name, :age, :address, :salary)",
             {'id':2, 'name':'Allen', 'age':25, 'address':'Texas', 'salary':15000.00});

data = [(3, 'Teddy', 23, 'Norway ', 200000.00 ),
        (4, 'Mark', 25, 'Rich-Mond ', 65000.00 )]

conn.executemany("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \
                  VALUES (?, ?, ?, ?, ?)", data);

conn.executescript("""
    INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)
    VALUES (5, 'Mark1', 25, 'Rich-Mond ', 65000.00 );

    INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)
    VALUES (6, 'Mark2', 25, 'Rich-Mond ', 65000.00 );
""");

conn.commit()
print("Records created successfully")
```

○ INSERT 확인


DB Browser for SQLite - /Users/gon0121/Downloads/create.db
새 데이터베이스(N)
데이터베이스 열기(O)
변경사항 저장하기(W)
변경사항 취소하기(R)
데이터베이스 구조
데이터 보기
Pragma 수정
SQL 실행
테이블: COMPANY
새 레코드
레코드 삭제
ID NAME AGE ADDRESS SALARY
1 Paul 32 California 20000.0
2 Allen 25 Texas 15000.0
3 Teddy 23 Norway 200000.0
4 Mark 25 Rich-Mond 65000.0
5 Mark1 25 Rich-Mond 65000.0
6 Mark2 25 Rich-Mond 65000.0
1 - 6 of 6
특정 레코드 행으로 가기:
데티어베이스 셀 수정하기(C)
모드: 문자열
가져오기
내보내기
NULL로 만들기
현재 데이터 타입: 문자열 / 숫자
1 문자
적용
원격
식별하기
이름 커밋 마지막 수정 크기
SQL 로그
플롯
DB 스키마
원격
UTF-8

○ SELECT

```
cursor = conn.execute("SELECT id, name, address, salary from COMPANY")

for row in cursor:
    print("ID = ", row[0])
    print("NAME = ", row[1])
    print("ADDRESS = ", row[2])
    print("SALARY = ", row[3], end='\n\n')

print("Operation done successfully")
```

```
ID =  1
NAME =  Paul
ADDRESS =  California
SALARY =  20000.0

ID =  2
NAME =  Allen
ADDRESS =  Texas
SALARY =  15000.0

ID =  3
NAME =  Teddy
ADDRESS =  Norway
SALARY =  200000.0

ID =  4
NAME =  Mark
ADDRESS =  Rich-Mond
SALARY =  65000.0

ID =  5
NAME =  Mark1
ADDRESS =  Rich-Mond
SALARY =  65000.0

ID =  6
NAME =  Mark2
ADDRESS =  Rich-Mond
SALARY =  65000.0

Operation done successfully
```

○ UPDATE

```
cid = 1

conn.execute("UPDATE COMPANY set SALARY = 25000.00 where ID = :id", {'id':cid})
conn.commit()

print("Total number of rows updated :", conn.total_changes)

cursor = conn.execute("SELECT id, name, address, salary from COMPANY")
for row in cursor:
    print("ID = ", row[0])
    print("NAME = ", row[1])
    print("ADDRESS = ", row[2])
    print("SALARY = ", row[3], "\n")
```

```
Total number of rows updated : 1
ID =  1
NAME =  Paul
ADDRESS =  California
SALARY =  25000.0
```

○ DELETE

```
conn.execute("DELETE from COMPANY where ID = 2;")

print("Total number of rows deleted :", conn.total_changes)

cursor = conn.execute("SELECT id, name, address, salary from COMPANY")
for row in cursor:
    print("ID = ", row[0])
    print("NAME = ", row[1])
    print("ADDRESS = ", row[2])
    print("SALARY = ", row[3], "\n")
```

```
Total number of rows deleted : 1
ID =  1
NAME =  Paul
ADDRESS =  California
SALARY =  25000.0

ID =  3
NAME =  Teddy
ADDRESS =  Norway
SALARY =  200000.0
```

■ 예제

○ Database 생성(열기), Cursor 생성

```
import sqlite3 as sql

con = None

try:
    con = sql.connect('test.db')

    cur = con.cursor()

    cur.execute('SELECT SQLITE_VERSION()')

    data = cur.fetchone()

    print("SQLite Version: ", data)

except Exception as e:
    print("Error: ", e)

finally:
    if con:
        con.close()
```

○ DDL – CREATE, DML – INSERT(execute)

```
con = sql.connect('test.db')

with con:
    cur = con.cursor()
    cur.execute("CREATE TABLE Cars(Id INT, Name TEXT, Price INT)")
    cur.execute("INSERT INTO Cars VALUES(1,'Audi',52642)")
    cur.execute("INSERT INTO Cars VALUES(2,'Mercedes',57127)")
    cur.execute("INSERT INTO Cars VALUES(3,'Skoda',9000)")
    cur.execute("INSERT INTO Cars VALUES(4,'Volvo',29000)")
    cur.execute("INSERT INTO Cars VALUES(5,'Bentley',350000)")
    cur.execute("INSERT INTO Cars VALUES(6,'Citroen',21000)")
    cur.execute("INSERT INTO Cars VALUES(7,'Hummer',41400)")
    cur.execute("INSERT INTO Cars VALUES(8,'Volkswagen',21600)")
```

○ DDL – CREATE, DML – INSERT(executemany)

```
cars = [
    (1, 'Audi', 52642),
    (2, 'Mercedes', 57127),
    (3, 'Skoda', 9000),
    (4, 'Volvo', 29000),
    (5, 'Bentley', 350000),
    (6, 'Hummer', 41400),
    (7, 'Volkswagen', 21600)
]

con = sql.connect('test.db')

with con:
    cur = con.cursor()
    cur.execute("DROP TABLE IF EXISTS Cars")
    cur.execute("CREATE TABLE Cars(Id INT PRIMARY KEY, Name TEXT, Price INT)")
    cur.executemany("INSERT INTO Cars VALUES(?, ?, ?)", cars)
```

○ DDL – DROP, CREATE, DML – INSERT(executescript)

```
try:
    con = sql.connect('test.db')

    cur = con.cursor()

    cur.executescript("""
        DROP TABLE IF EXISTS Cars;
        CREATE TABLE Cars(Id INT, Name TEXT, Price INT);
        INSERT INTO Cars VALUES(1,'Audi',52642);
        INSERT INTO Cars VALUES(2,'Mercedes',57127);
        INSERT INTO Cars VALUES(3,'Skoda',9000);
        INSERT INTO Cars VALUES(4,'Volvo',29000);
        INSERT INTO Cars VALUES(5,'Bentley',350000);
        INSERT INTO Cars VALUES(6,'Citroen',21000);
        INSERT INTO Cars VALUES(7,'Hummer',41400);
        INSERT INTO Cars VALUES(8,'Volkswagen',21600);
        """)

    con.commit()

except Exception as e:
    if con:
        con.rollback()

    print("Error: ", e)

finally:
    if con:
        con.close()
```

○ DDL – DROP, CREATE, DML – INSERT(lastrowid)

```
con = sql.connect(':memory:')

with con:
    cur = con.cursor()
    cur.execute("CREATE TABLE Friends(Id INTEGER PRIMARY KEY, Name TEXT);")
    cur.execute("INSERT INTO Friends(Name) VALUES ('Tom');")
    cur.execute("INSERT INTO Friends(Name) VALUES ('Rebecca');")
    cur.execute("INSERT INTO Friends(Name) VALUES ('Jim');")
    cur.execute("INSERT INTO Friends(Name) VALUES ('Robert');")

    lid = cur.lastrowid
    print("The last Id of the inserted row is", lid)
```

○ DML – SELECT(fetchone)

```
con = sql.connect('test.db')

with con:
    cur = con.cursor()
    cur.execute("SELECT * FROM Cars")

    while True:
        row = cur.fetchone()

        if row == None:
            break

        print(row[0], row[1], row[2])
```

○ DML – SELECT(dictionary, fetchall)

```
con = sql.connect('test.db')

with con:
    con.row_factory = sql.Row

    cur = con.cursor()
    cur.execute("SELECT * FROM Cars")

    rows = cur.fetchall()

    for row in rows:
        print(row["Id"], row["Name"], row["Price"])
```

○ DML – UPDATE(qmark)

```
uId = 1
uPrice = 62300

con = sql.connect('test.db')

with con:
    cur = con.cursor()

    cur.execute("UPDATE Cars SET Price=? WHERE Id=?", (uPrice, uId))

    print("Number of rows updated:", cur.rowcount)
```

○ DML – UPDATE(named)

```
uId = 4

con = sql.connect('test.db')

with con:

    cur = con.cursor()

    cur.execute("SELECT Name, Price FROM Cars WHERE Id=:Id", {"Id": uId})

    row = cur.fetchone()
    print(row[0], row[1])
```

○ Dump(backup)

iterdump()

Returns an iterator to dump the database in an SQL text format. Useful when saving an in-memory database for later restoration. This function provides the same capabilities as the `.dump` command in the **sqlite3** shell.

```
# Convert file existing_db.db to SQL dump file dump.sql
import sqlite3

con = sqlite3.connect('existing_db.db')
with open('dump.sql', 'w') as f:
    for line in con.iterdump():
        f.write('%s\n' % line)
```

○ Export / Import

```
cars = (
    (1, 'Audi', 52643),
    (2, 'Mercedes', 57642),
    (3, 'Skoda', 9000),
    (4, 'Volvo', 29000),
    (5, 'Bentley', 350000),
    (6, 'Hummer', 41400),
    (7, 'Volkswagen', 21600)
)

def writeData(data):
    f = open('cars.sql', 'w')

    with f:
        f.write(data)


con = sql.connect(':memory:')

with con:
    cur = con.cursor()

    cur.execute("DROP TABLE IF EXISTS Cars")
    cur.execute("CREATE TABLE Cars(Id INT, Name TEXT, Price INT)")
    cur.executemany("INSERT INTO Cars VALUES(?, ?, ?)", cars)
    cur.execute("DELETE FROM Cars WHERE Price < 30000")

    data = '\n'.join(con.iterdump())

    writeData(data)
```

○ Export / Import

```
def readData():
    f = open('cars.sql', 'r')

    with f:
        data = f.read()
        return data


con = sql.connect(':memory:')

with con:
    cur = con.cursor()

    rowData = readData()
    cur.executescript(rowData)

    cur.execute("SELECT * FROM Cars")

    rows = cur.fetchall()

    for row in rows:
        print(row)
```

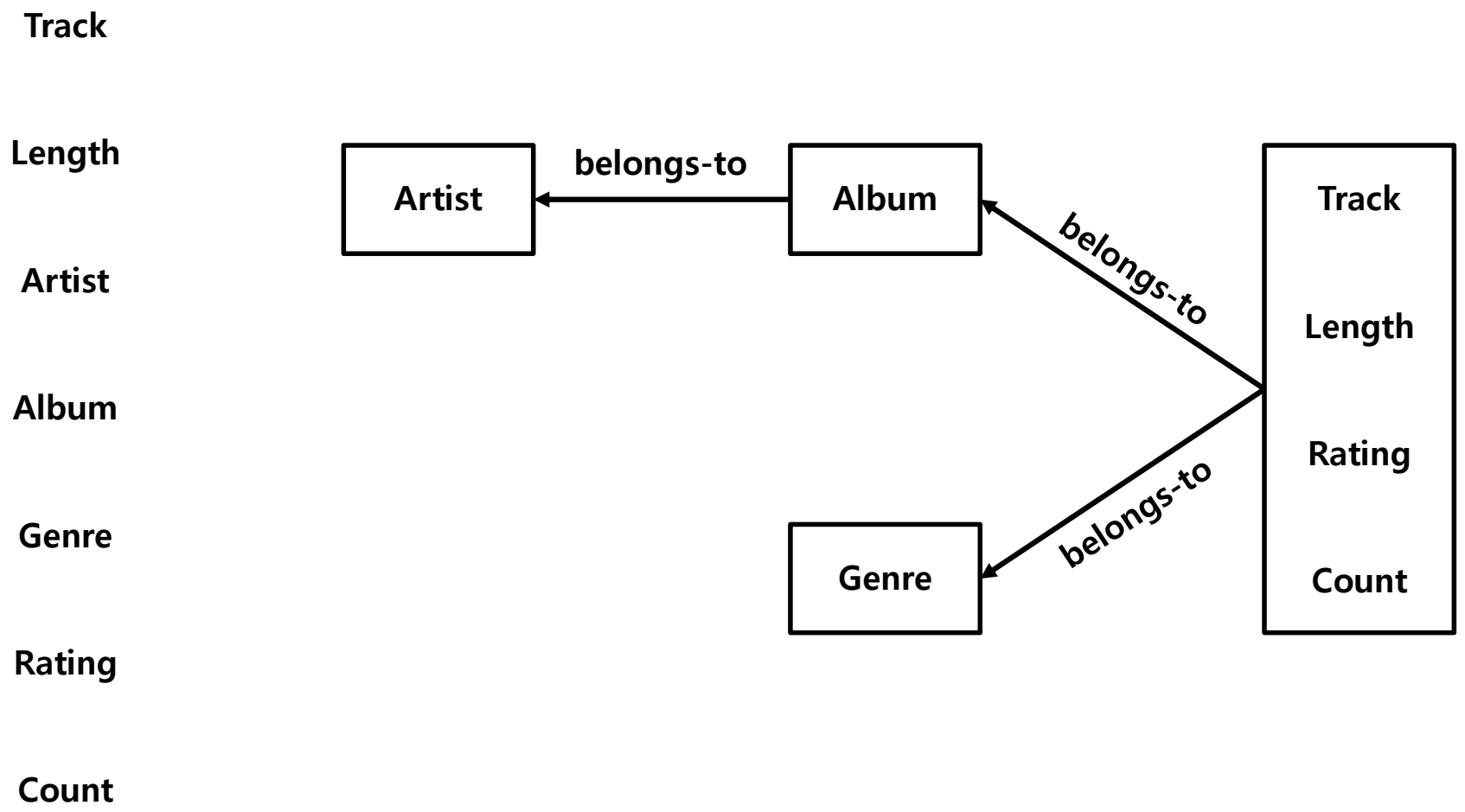
○ Transaction(Commit)

```
try:
    con = sql.connect('test.db')
    cur = con.cursor()
    cur.execute("DROP TABLE IF EXISTS Friends")
    cur.execute("CREATE TABLE Friends(Id INTEGER PRIMARY KEY, Name TEXT)")
    cur.execute("INSERT INTO Friends(Name) VALUES ('Tom')")
    cur.execute("INSERT INTO Friends(Name) VALUES ('Rebecca')")
    cur.execute("INSERT INTO Friends(Name) VALUES ('Jim')")
    cur.execute("INSERT INTO Friends(Name) VALUES ('Robert')")

    #con.commit()

except Exception as e:

    if con:
        con.rollback()

    print("Error:", e)

finally:
    if con:
        con.close()
```

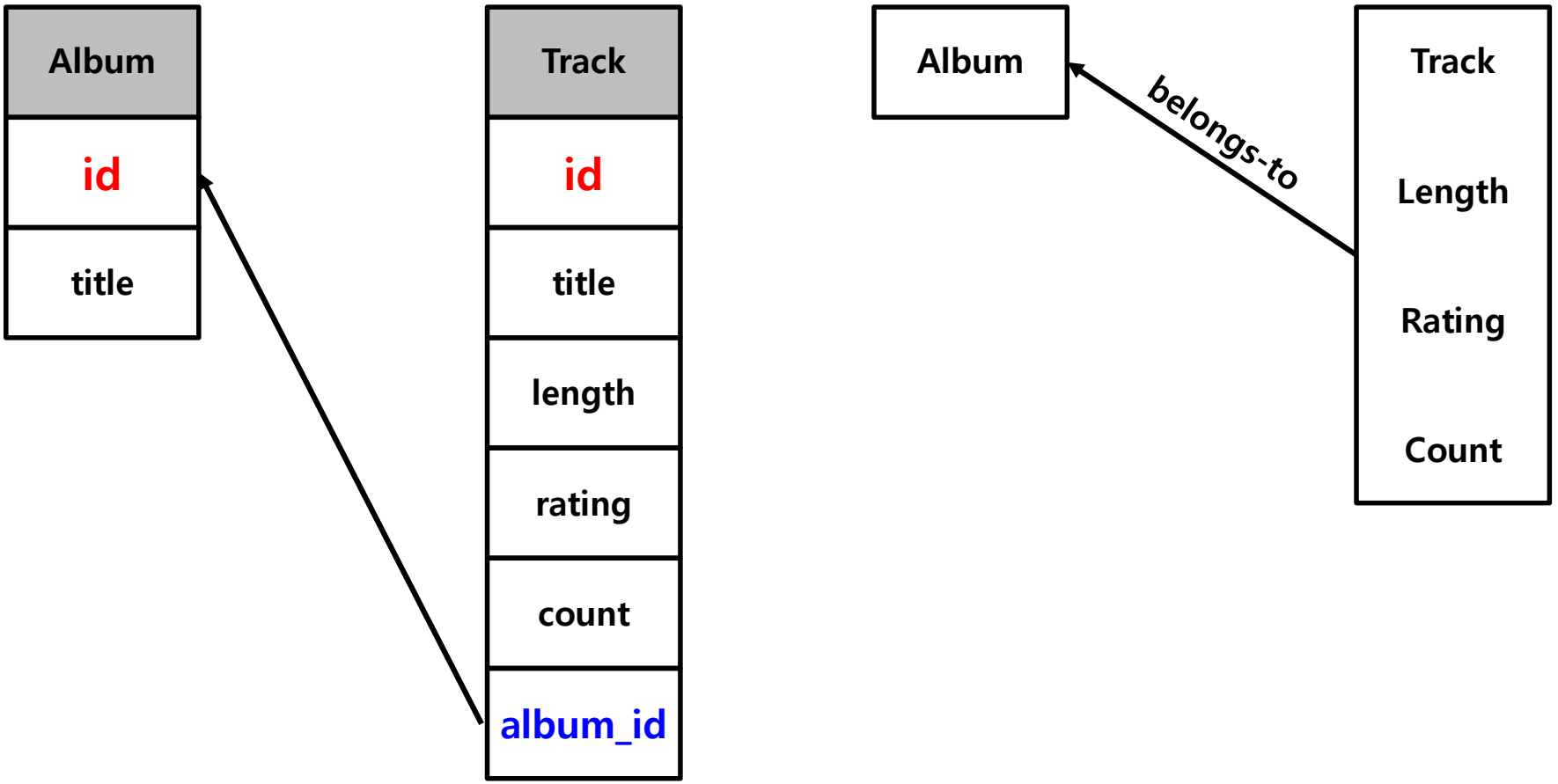
■ 예제 (ER Model – SQL)

Track	Length	Artist	Album	Genre	Rating	Count
Hells Bells	5:13	AC/DC	Who Made Who	Rock	★★★★★	61
Shake Your Foundations	3:54	AC/DC	Who Made Who	Rock	★★★★★	70
Chase the Ace	3:01	AC/DC	Who Made Who	Rock		56
For Those About To Rock (We ...	5:54	AC/DC	Who Made Who	Rock	★★★★★	61
Dúlamán	3:43	Altan	Natural Wonders M...	New Age		31
Rode Across the Desert	4:10	America	Greatest Hits	Easy Listen...	★★★★★	23
Now You Are Gone	3:08	America	Greatest Hits	Easy Listen...	★★★★★	18
Tin Man	3:30	America	Greatest Hits	Easy Listen...	★★★★★	23
Sister Golden Hair	3:22	America	Greatest Hits	Easy Listen...	★★★★★	24
Track 01	4:22	Billy Price	Danger Zone	Blues/R&B	★★★★★	26
Track 02	2:45	Billy Price	Danger Zone	Blues/R&B	★★★★★	18
Track 03	3:26	Billy Price	Danger Zone	Blues/R&B	★★★★★	22
Track 04	4:17	Billy Price	Danger Zone	Blues/R&B	★★★★★	18
Track 05	3:50	Billy Price	Danger Zone	Blues/R&B	★★★★★	21
War Pigs/Luke's Wall	7:58	Black Sabbath	Paranoid	Metal	★★★★★	25
Paranoid	2:53	Black Sabbath	Paranoid	Metal	★★★★★	22
Planet Caravan	4:35	Black Sabbath	Paranoid	Metal	★★★★★	25
Iron Man	5:59	Black Sabbath	Paranoid	Metal	★★★★★	26
Electric Funeral	4:53	Black Sabbath	Paranoid	Metal	★★★★★	22
Hand of Doom	7:10	Black Sabbath	Paranoid	Metal	★★★★★	23
Rat Salad	2:30	Black Sabbath	Paranoid	Metal	★★★★★	31
Jack the Stripper/Fairies Wear ...	6:14	Black Sabbath	Paranoid	Metal	★★★★★	24
Bomb Squad (TECH)	3:28	Brent	Brent's Album			1
clay techno	4:36	Brent	Brent's Album			2
Heavy	3:08	Brent	Brent's Album			1
Hi metal man	4:20	Brent	Brent's Album			1
Mistro	2:58	Brent	Brent's Album			1

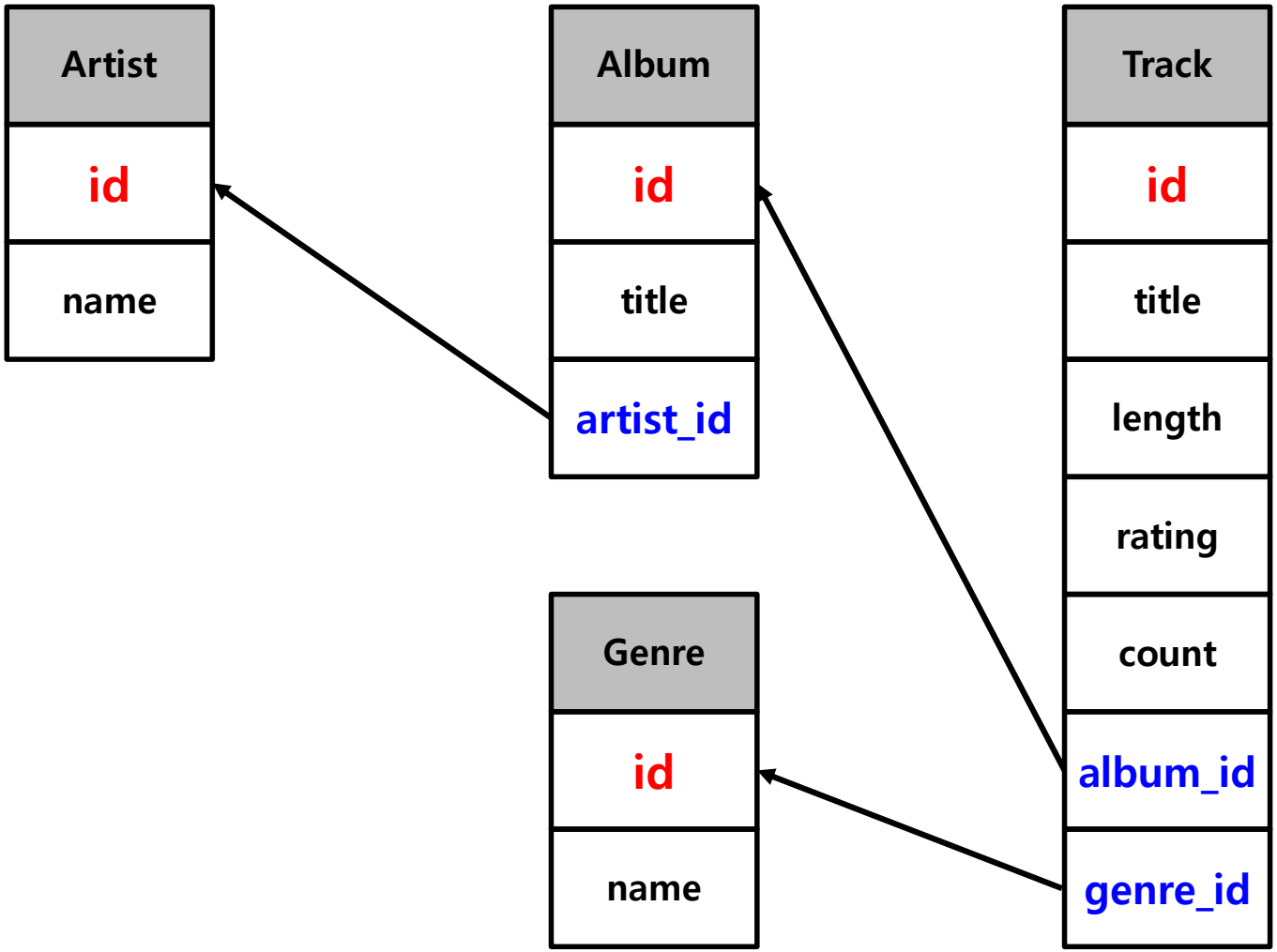
○ 개체(Entity) – 관계(Relationship)



○ 기본키(Primary Key) / 참조키(Foreign Key)



○ 기본키(Primary Key) / 참조키(Foreign Key)



○ TABLE 생성

➢ Artist

– CREATE TABLE Artist (
id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
name TEXT
)

➢ Genre

– CREATE TABLE Genre (
id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
name TEXT
)

➢ Album

– CREATE TABLE Album (
id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
title TEXT,
artist_id INTEGER
)

- Track
  - CREATE TABLE Track (
    id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
    title TEXT,
    length INTEGER,
    rating INTEGER,
    count INTEGER,
    album_id INTEGER,
    genre_id INTEGER
)

○ INSERT

- Artist
  - insert into Artist (name) values ('Led Zepplin')
  - insert into Artist (name) values ('AC/DC')

- Genre
  - insert into Genre (name) values ('Rock')
  - insert into Genre (name) values ('Metal')

- Album
  - insert into Album (title, artist_id) values ('Who Made Who', 2)
  - insert into Album (title, artist_id) values ('IV', 1)

- Track
  - insert into Track (title, rating, length, count, album_id, genre_id) values ('Black Dog', 5, 297, 0, 2, 1)
  - insert into Track (title, rating, length, count, album_id, genre_id) values ('Stairway', 5, 482, 0, 2, 1)
  - insert into Track (title, rating, length, count, album_id, genre_id) values ('About to Rock', 5, 313, 0, 1, 2)
  - insert into Track (title, rating, length, count, album_id, genre_id) values ('Who Made Who', 5, 207, 0, 1, 2)

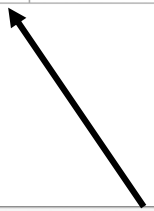
	id	name
	필터	필터
1	1	Led Zepplin
2	2	AC/DC

	id	name
	필터	필터
1	1	Rock
2	2	Metal

	id	title	artist_id
	필터	필터	필터
1	1	Who Made Who	2
2	2	IV	1

	id	title	length	rating	count	album_id	genre_id
	필터	필터	필터	필터	필터	필터	필터
1	1	Black Dog	297	5	0	2	1
2	2	Stairway	482	5	0	2	1
3	3	About to Rock	313	5	0	1	2
4	4	Who Made Who	207	5	0	1	2

	id	name
	필터	필터
1	1	Led Zepplin
2	2	AC/DC



	id	title	artist_id
	필터	필터	필터
1	1	Who Made Who	2
2	2	IV	1

○ Join

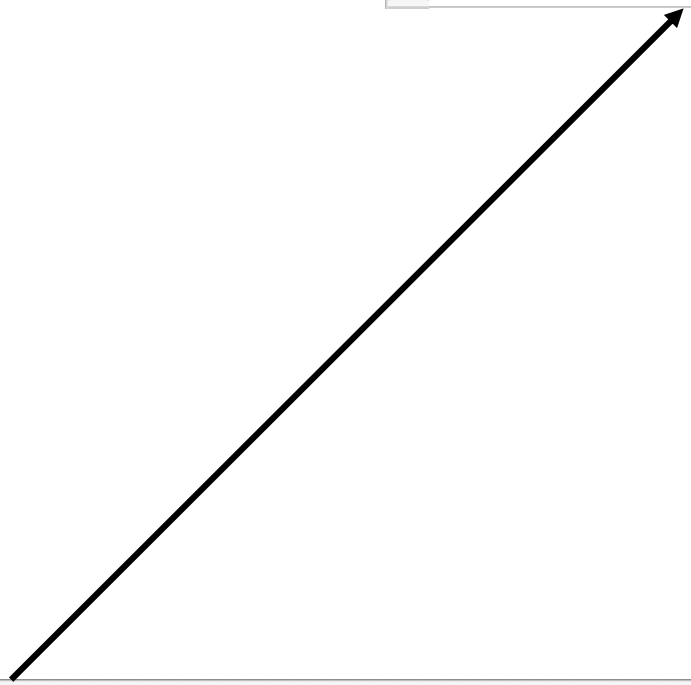
➢ select Album.title, Artist.name
from Album
join Artist on
Album.artist_id = Artist.id

	title	name
1	Who Made Who	AC/DC
2	IV	Led Zepplin

➢ select Track.title, Genre.name
from Track
join Genre on
Track.genre_id = Genre.id

	title	name
1	Black Dog	Rock
2	Stairway	Rock
3	About to Rock	Metal
4	Who Made Who	Metal

	id	name
	필터	필터
1	1	Rock
2	2	Metal

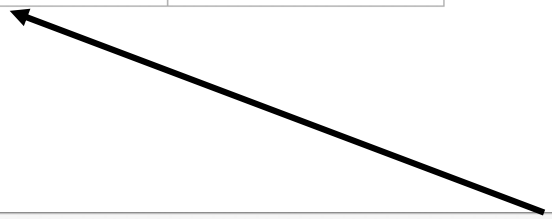


	id	title	length	rating	count	album_id	genre_id
	필터	필터	필터	필터	필터	필터	필터
1	1	Black Dog	297	5	0	2	1
2	2	Stairway	482	5	0	2	1
3	3	About to Rock	313	5	0	1	2
4	4	Who Made Who	207	5	0	1	2

➢ select Album.title, Track.title
from Track
join Album on
Track.album_id = Album.id

	title	title
1	IV	Black Dog
2	IV	Stairway
3	Who Made Who	About to Rock
4	Who Made Who	Who Made Who

	id	title	artist_id
	필터	필터	필터
1	1	Who Made Who	2
2	2	IV	1



	id	title	length	rating	count	album_id	genre_id
	필터	필터	필터	필터	필터	필터	필터
1	1	Black Dog	297	5	0	2	1
2	2	Stairway	482	5	0	2	1
3	3	About to Rock	313	5	0	1	2
4	4	Who Made Who	207	5	0	1	2

➢ select Track.title, Artist.name, Album.title, Genre.name
from Track
join Artist join Album join Genre on
Track.album_id = Album.id
and
Track.genre_id = Genre.id
and
Album.artist_id = Artist.id

	title	name	title	name
1	Black Dog	Led Zepplin	IV	Rock
2	Stairway	Led Zepplin	IV	Rock
3	About to Rock	AC/DC	Who Made Who	Metal
4	Who Made Who	AC/DC	Who Made Who	Metal

	id	title	length	rating	count	album_id	genre_id
	필터	필터	필터	필터	필터	필터	필터
1	1	Black Dog	297	5	0	2	1
2	2	Stairway	482	5	0	2	1
3	3	About to Rock	313	5	0	1	2
4	4	Who Made Who	207	5	0	1	2

	title	name	title	name
1	Black Dog	Led Zepplin	IV	Rock
2	Stairway	Led Zepplin	IV	Rock
3	About to Rock	AC/DC	Who Made Who	Metal
4	Who Made Who	AC/DC	Who Made Who	Metal

☑	Hells Bells	5:13	AC/DC	Who Made Who	Rock	★★★★★	61
☑	Shake Your Foundations	3:54	AC/DC	Who Made Who	Rock	★★★★★	70
☑	Chase the Ace	3:01	AC/DC	Who Made Who	Rock		56
☑	For Those About To Rock (We ...	5:54	AC/DC	Who Made Who	Rock	★★★★★	61
☑	Dúlamán	3:43	Altan	Natural Wonders M...	New Age		31
☑	Rode Across the Desert	4:10	America	Greatest Hits	Easy Listen...	★★★★★	23
☑	Now You Are Gone	3:08	America	Greatest Hits	Easy Listen...	★★★★★	18
☑	Tin Man	3:30	America	Greatest Hits	Easy Listen...	★★★★★	23