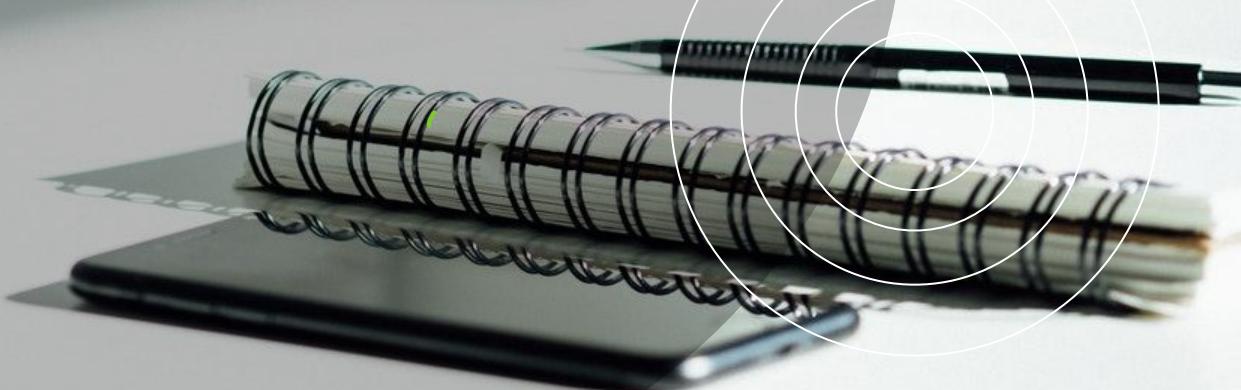


실전 프로젝트1



8

SQLite

공지

- 출석확인 방법
 - 화요일 (대면 / 줌), 금요일 (줌), LMS 인경우 HD LMS에서 자동출석체크
 - Hisnet 온라인 출석 : 수강과목 선택 > 강의정보 > 강의출석현황 메뉴에서 확인
 - 5주차
 - 10월 1일(금) - HDLMS 동영상 시청 + 실시간 온라인 실습으로 수업 진행함
 - LMS 5주차 수업 동영상을 아래 실습 시간 시작 전까지 시청 완료해야 함! (자동 출석 체크)
 - 줌 참석 여부도 출석에 포함
 - 1분반(김광) - 오전 10시부터 줌으로 실시간 온라인 실습 진행
 - 2분반(장소연) - 오전 10시부터 줌으로 실시간 온라인 실습 진행
 - 3분반(장소연) - 오후 5시부터 줌으로 실시간 온라인 실습 진행
- ~금요일 아침 10시

TodoListApp Update#2 review

- 기능 변경
- 목록 출력 시 일련 번호를 함께 출력하도록 하여, 특정 항목을 선택할 때에 이 번호를 사용할 것.
- [항목 수정], [항목 삭제]에서 제목으로 찾는 대신, 번호를 입력하도록 변경하여 기능 수행

```
public static void deleteItem(TodoList l) {  
  
    Scanner sc = new Scanner(System.in);  
  
    System.out.print("[항목 삭제]\n" + "삭제할 항목의 번호를 입력하시오 > ");  
    int index = sc.nextInt();  
    if (index > l.getCount()) {  
        System.out.println("없는 번호입니다!");  
        return;  
    }  
    System.out.println((index)+". "+l.getItem(index-1).toString());  
  
    System.out.print("위 항목을 삭제하시겠습니까? (y/n) > ");  
    String yesno = sc.next();  
    if (yesno.equals("y")) {  
        l.deleteItem(l.getItem(index-1));  
        System.out.println("삭제되었습니다.");  
    }  
}
```

```
public class TodoList {  
    private List<TodoItem> list;  
  
    public TodoList() {  
        this.list = new ArrayList<TodoItem>();  
    }  
  
    public TodoItem getItem(int index) {  
        return list.get(index);  
    }  
}
```

내장 라이브러리 함수
(내장 라이브러리 함수 최대한 활용하기 연습)

TodoListApp Update#2 review

- 검색 기능 추가
- 명령어 형식 :
find <키워드>
- 현재 목록 내
제목과 내용에
서 <키워드>를
포함하고 있는
모든 항목을 출력한다.

```
case "find":  
    String keyword = sc.next();  
    TodoUtil.findList(l, keyword);  
    break;
```

```
public static void findList(TodoList l, String keyword) {  
    int count=0;  
    for (int i=0; i<l.getCount(); i++) {  
        if (l.getItem(i).getTitle().contains(keyword)||l.getItem(i).getDesc().contains(keyword)) {  
            System.out.println((i+1)+". "+l.getItem(i).toString());  
            count++;  
        }  
    }  
    System.out.printf("총 %d개의 항목을 찾았습니다.\n", count);  
}
```

TodoListApp Update#2 review

- 검색 기능 추가
- 명령어 형식 :
find <키워드>
- 제목과 내용에
<키워드>를 포함하고 있는 모든 항목을 출력
- 최신순 정렬 기능
 - 명령어 형식,
ls_date_desc
 - ls_date 의 역순으로 정렬

```
case "find":  
    String keyword = sc.nextLine().trim();  
    TodoUtil.findList(l, keyword);  
    break;
```

```
public static void findList(TodoList l, String keyword) {  
    int count=0;  
    for (int i=0; i<l.getCount(); i++) {  
        if (l.getItem(i).getTitle().contains(keyword)||l.getItem(i).getDesc().contains(keyword)) {  
            System.out.println((i+1)+" "+l.getItem(i).toString());  
            count++;  
        }  
    }  
    System.out.printf("총 %d개의 항목을 찾았습니다.\n", count);  
}
```

```
case "ls_date_desc":  
    l.sortByDate();  
    l.reverseList();  
    System.out.println("날짜역순으로 정렬하였습니다.");  
    isList = true;  
    break;
```

TodoListApp Update#2 review

- 카테고리 목록 출력 기능
- 명령어 형식,
ls_cate
- 현재 등록되어 있는 카테고리를 중복되지 않도록 출력한다.
- help 명령의 메뉴에 추가, 명령어에 대한 기능 추가

```
case "ls_cate":  
    TodoUtil.listcateAll(l);  
    break;
```

```
public static void listcateAll(TodoList l) {  
    Set<String> clist = new HashSet<String>();  
  
    for (TodoItem c : l.getList()) {  
        clist.add(c.getCategory());  
    }  
      
    Iterator it = clist.iterator();  
    while (it.hasNext()) {  
        String s = (String)it.next();  
        System.out.print(s);  
        if(it.hasNext()) System.out.print(" / ");  
    }  
    System.out.printf("\n총 %d개의 카테고리가 등록되어 있습니다.\n", clist.size());  
}
```

File vs Database

- 파일 시스템의 문제점
 - 응용 프로그램과 데이터(파일)간에 상호 의존성이 존재되고, 프로그램과 파일은 보통 1:1로 대응됨
 - 한 시스템 내에서 데이터가 중복 저장되어 관리됨
 - 일관성 어려움 / 보안성 / 중복 저장으로 인한 경제성 저하 / 다중 사용자 사용 어려움([효율성](#))
- 데이터베이스 특징
 - 실시간 접근성(Real-time accessibility) - 다수 사용자의 요청에 대해 몇 초 내 응답
 - 지속적인 변화(Continuous evolution) - 최신의 데이터가 정확하게 저장됨
 - 동시 공유(Concurrent sharing) - 동일한 데이터를 서로 다른 목적으로 사용
 - 내용에 의한 참조(Content Reference) - 값에 의한 참조

DBMS

DBMS가 DB를 control

- DBMS(Database Management System)
- 수많은 데이터를 편리하게 저장하고, 효율적으로 관리, 검색할 수 있는 데이터베이스 관리 시스템
- RDBMS(Relational DBMS) - 제약 조건을 DB
 - 관계형 데이터베이스 관리 시스템
 - 데이터베이스 내의 테이블은 서로 연관되어 데이터를 저장, 구성, 관리함
 - 데이터를 관리하기 위해 SQL(Structured Query Language) 사용
 - Oracle, Informix(IBM), MySQL, SQL Server(MS), MariaDB, SQLite 등
- SQL(Structured Query Language) - 구조화된 쿼리 | SQL을 능숙하게 쓸 줄 알아야 함
 - RDBMS에서 CRUD, 스키마 생성, 수정, 조회 등 관리를 위해 설계된 프로그래밍 언어
 - DDL(Data Definition Language) : 테이블, 인덱스, 제약조건 등 정의
 - DML(Data Manipulation Language) : 데이터 추가, 수정, 삭제, 조회
 - DCL(Data Control Language) : 사용자 권한, 트랜잭션 등 처리

데이터베이스 사용순서

- 데이터베이스 설치
 - 관계형 데이터베이스 종류 선택 => SQLite
 - 데이터베이스 설치
 - 샘플 데이터베이스 다운로드
 - 데이터베이스 연결 및 명령어 실행
- 데이터베이스 관리 어플리케이션(GUI) 사용
 - DB Browser for SQLite 설치
 - 데이터베이스 열기
 - 조회

↑ 기본의 DBMS (터미널 사용은 아님)

↳ 이 file을 DB처럼 쓸수 있다

↳ 굉장히 simple!

SQLite

이 번역 다음에 SQLite 연습하고 이와 관련한 충분한 풀이

- RDBMS(관계형 데이터베이스 관리시스템)
 - MySQL(or MariaDB)나 Oracle에 비해 가벼운 데이터베이스(Light-Weight)
 - 이식성, 안정성, 강력한 성능의 독립형 파일 기반의 오픈소스 RDBMS
 - 안드로이드 운영체제에서 기본 데이터베이스로 지원
 - 장점
 - Serverless SQLite : Server process가 없음
 - Zero-configuration : 시작/중지 등 관리 기능이 필요 없음
 - SQLite는 단일 파일에 전체 데이터가 저장
 - 단점
 - 동시성(Concurrency)에 제한이 있음 - 조직에서 혼자쓰기용
 - 사용자 관리가 없음
 - 네트워크 액세스 기능 없음, 여러 사용자 사용에 적합하지 않음



SQLite 설치

- SQLite 다운로드 <https://www.sqlite.org/download.html>
- 압축해제

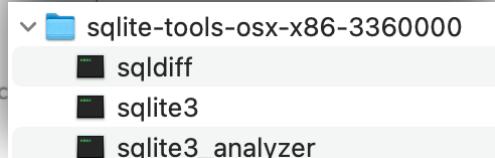
Precompiled Binaries for Mac OS X (x86)

[sqlite-tools-osx-x86-3360000.zip](#) (1.47 MiB) A bundle of command-line tools for managing SQLite database files, including the [command-line shell](#) program, the [sqlcipher](#) program, and the [sqlite3_analyzer](#) program.
(sha3:
edd862b3ad642bdf7802d5db14c778c0642059ea035462aa526ea062deac9961)

Precompiled Binaries for Windows

[sqlite-dll-win32-x86-3360000.zip](#) (542.87 KiB) 32-bit DLL (x86) for SQLite version 3.36.0.
(sha3:
dbcc568711f3f3e12a32e5abfcfa652f1c38eb71ccedd81874e9669708f9c71c
[sqlite-dll-win64-x64-3360000.zip](#) (880.05 KiB) 64-bit DLL (x64) for SQLite version 3.36.0.
(sha3:
af88804c191758431458611b8214b466348df17415b2671641793cef53ae762a)

[sqlite-tools-win32-x86-3360000.zip](#) (1.82 MiB) A bundle of command-line tools for managing SQLite database files, including the [command-line shell](#) program, the [sqlcipher.exe](#) program, and the [sqlite3_analyzer.exe](#) program.



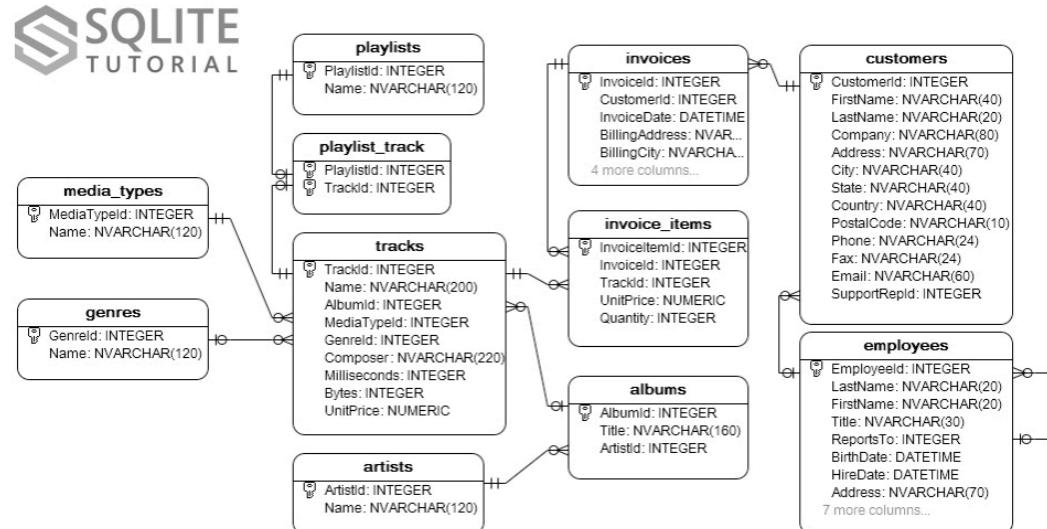
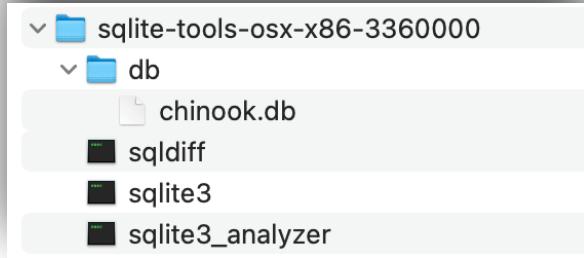
SQLite 설치

- Sample database 다운로드
- <https://www.sqlitetutorial.net/>) START HERE

sql 명령어는 여기서 다 있음

- 3. Introduction to the SQLite Sample Database

- Download SQLite sample database : chinook.db (11 tables)



<https://www.sqlitetutorial.net/sqlite-sample-database/>

SQLite 설치

★ 기존의 풀체를 그대로 복제한 뒤
SQLite 버전으로 바꿔가면서 연습

- 데이터베이스 연결 또는 생성
 - sqlite3 [데이터베이스 파일 명] 기본적으로 [이름.db]
 - 명령어
 - .help
 - .tables - table의 이름들
 - .header on
 - .schema [테이블 명] - 테이블 속성과 토착 (영어)
 - .exit or .quit

기본적으로
SQLite 명령어로
으로 시작

```
% sqlite3 ./db/chinook.db
SQLite version 3.32.3 2020-06-18 14:16:19
Enter ".help" for usage hints.
sqlite> .tables
albums          employees        invoices        playlists
artists         genres           media_types     tracks
customers       invoice_items   playlist_track
sqlite> .exit
```

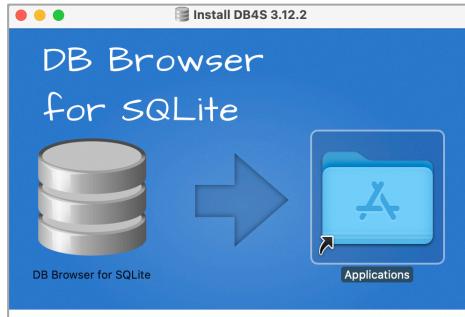
고급의 주선 예제 db

★ .insert, .delete, .update, .delete

DB Browser for SQLite 설치

- 처음부터 이렇게 쓰지 말자!
terminal로 연습할 것!

- DB관리를 위한 GUI tool
- <https://sqlitebrowser.org/>
- 버전 선택 후 다운로드 및 설치
 - Version 3.12.2 released
- chinook.db 열기



<https://sqlitebrowser.org/>

| AlbumId | Title | ArtistId |
|---------|--|----------|
| 1 | For Those About To Rock We Salute You | 1 |
| 2 | Balls to the Wall | 2 |
| 3 | Restless and Wild | 2 |
| 4 | Let There Be Rock | 1 |
| 5 | Big Ones | 3 |
| 6 | Jagged Little Pill | 4 |
| 7 | Facelift | 5 |
| 8 | Warner 25 Anos | 6 |
| 9 | Plays Metallica By Four Cellos | 7 |
| 10 | Audioslave | 8 |
| 11 | Out Of Exile | 8 |
| 12 | BackBeat Soundtrack | 9 |
| 13 | The Best Of Billy Cobham | 10 |
| 14 | Alcohol Fueled Brewtality Live! [Disc 1] | 11 |
| 15 | Alcohol Fueled Brewtality Live! [Disc 2] | 11 |

특정 레코드 행으로 가기: 1

SQL 로그(L) 플롯(P) DB 스키마(M)

UTF-8

데이터베이스 용어

Database

↳ table들이 들어있는
파일 하나

A screenshot of a database management system interface. On the left, there is a tree view with a node expanded to show 13 tables: albums, artists, customers, employees, genres, invoice_items, invoices, media_types, playlist_track, playlists, sqlite_sequence, sqlite_stat1, and tracks. A large blue arrow points from this list to the detailed table view on the right.

Table

A screenshot of a table definition screen. It shows a table named 'albums' with three columns: 'AlbumId' (INTEGER), 'Title' (NVARCHAR(160)), and 'ArtistId' (INTEGER). A pink box highlights the 'Field' column, with handwritten text above it reading '다른 db의 column' (Different db's column).

Field

Name/type/length/null

A screenshot of a table data grid. The title bar says '테이블: albums'. The table has columns: AlbumId, Title, and ArtistId. There are 6 rows of data:

| AlbumId | Title | ArtistId |
|---------|---------------------------------------|----------|
| 1 | For Those About To Rock We Salute You | 1 |
| 2 | Balls to the Wall | 2 |
| 3 | Restless and Wild | 2 |
| 4 | Let There Be Rock | 1 |
| 5 | Big Ones | 3 |
| 6 | Jagged Little Pill | 4 |

Data or Record

다른 db의 row

데이터베이스 생성 및 연결

- 기존 데이터베이스 연결
 - sqlite3 [기존 데이터베이스 파일명]
- 새 데이터베이스 생성
 - sqlite3 [새 데이터베이스 파일명]

```
% ls  
chinook.db  
% sqlite3 chinook.db  
SQLite version 3.28.0 2019-04-15 14:49:49  
Enter ".help" for usage hints.  
sqlite> .tables  
albums          employees        invoices       playlists  
artists         genres           media_types    tracks  
customers       invoice_items   playlist_track  
sqlite> .quit  
% sqlite3 myfirst.db  
SQLite version 3.28.0 2019-04-15 14:49:49  
Enter ".help" for usage hints.  
sqlite>
```

테이블 생성

- 필드정의
- 필드명/필드타입/Null 여부 등
- SQLite 필드타입
 - NULL
 - INTEGER(int) signed 1,2,3,4,5,6,8 byte number.
 - REAL: floating point value. 8 byte.
 - TEXT
 - BLOB: blob of data
- 테이블 생성 SQL

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    ...
);
```

| | ArtistId | Name |
|---|----------|-------------------|
| 1 | 1 | AC/DC |
| 2 | 2 | Accept |
| 3 | 3 | Aerosmith |
| 4 | 4 | Alanis Morissette |
| 5 | 5 | Alice In Chains |

```
% sqlite3 myfirst.db
SQLite version 3.32.3 2020-06-18 14:16:19
Enter ".help" for usage hints.
sqlite> .tables
sqlite> create table artists (
...>     ArtistId integer primary key autoincrement not null,
...>     Name varchar(30)
...> );
sqlite> .tables
artists
```

ArtistId 자동으로 매겨지기 때문에
다른 번호로

기본적으로 sqlite의 문장은 ';' 으로 끝남

CRUD SQL

. 명령어 - sqlite의 command
문장 ; - SQL 문장

○ 테이블에 데이터 추가 SQL

```
INSERT INTO table_name (column1, column2, column3, ...)  
VALUES (value1, value2, value3, ...);
```

```
sqlite> .tables artists  
artists  
sqlite> .schema artists  
CREATE TABLE artists (  
    ArtistId integer primary key autoincrement not null,  
    Name varchar(30) );  
sqlite> insert into artists (name) values ('아이유');  
sqlite> insert into artists (name) values ('태연');  
sqlite> insert into artists (name) values ('거미');  
sqlite> select * from artists;
```

autoincrement not null,
null이 들어갈수 있는
(비어있으면 안됨)

CRUD SQL

데이터 조회 SQL

```
SELECT * FROM tablename;
SELECT * FROM tablename order by column1;
SELECT * FROM tablename order by column1 desc;
SELECT * FROM tablename where column1 = 'keyword';
SELECT column1, column2 FROM tablename where column1 like 'keyword%';
```

```
sqlite> select * from artists;
1|아이유
2|태연
3|거미
sqlite> .header on
sqlite> select * from artists order by name;
ArtistId|Name
3|거미
1|아이유
2|태연
```

```
sqlite> select * from artists order by name desc;
ArtistId|Name
2|태연
1|아이유
3|거미
sqlite> select name from artists where name ='거미';
Name
거미
sqlite> select * from artists where name like '%이%';
ArtistId|Name
1|아이유
```

CRUD SQL

- 데이터 수정 SQL

```
UPDATE tablename  
SET column1 = value1, column2 = value2, ...  
WHERE condition;
```

```
sqlite> select * from artists where name ='아이유';  
ArtistId|Name  
1|아이유  
sqlite> update artists set name='아이유(이지은)' where name='아이유';  
sqlite> select * from artists where name ='아이유';  
sqlite> select * from artists ;  
ArtistId|Name  
1|아이유(이지은)  
2|태연  
3|거미
```

CRUD SQL

- 데이터 삭제 SQL

```
DELETE FROM tablename WHERE condition;
```

```
sqlite> insert into artists (name) values ('홍길동');
sqlite> select * from artists;
ArtistId|Name
1|아이유(이지은)
2|태연
3|거미
4|홍길동
sqlite> delete from artists where artistid = 4;
sqlite> select * from artists;
ArtistId|Name
1|아이유(이지은)
2|태연
3|거미
```