Anton Rotter-Sieren Navid Nadvi

999219083 912309925

For README for programs, scroll to bottom.

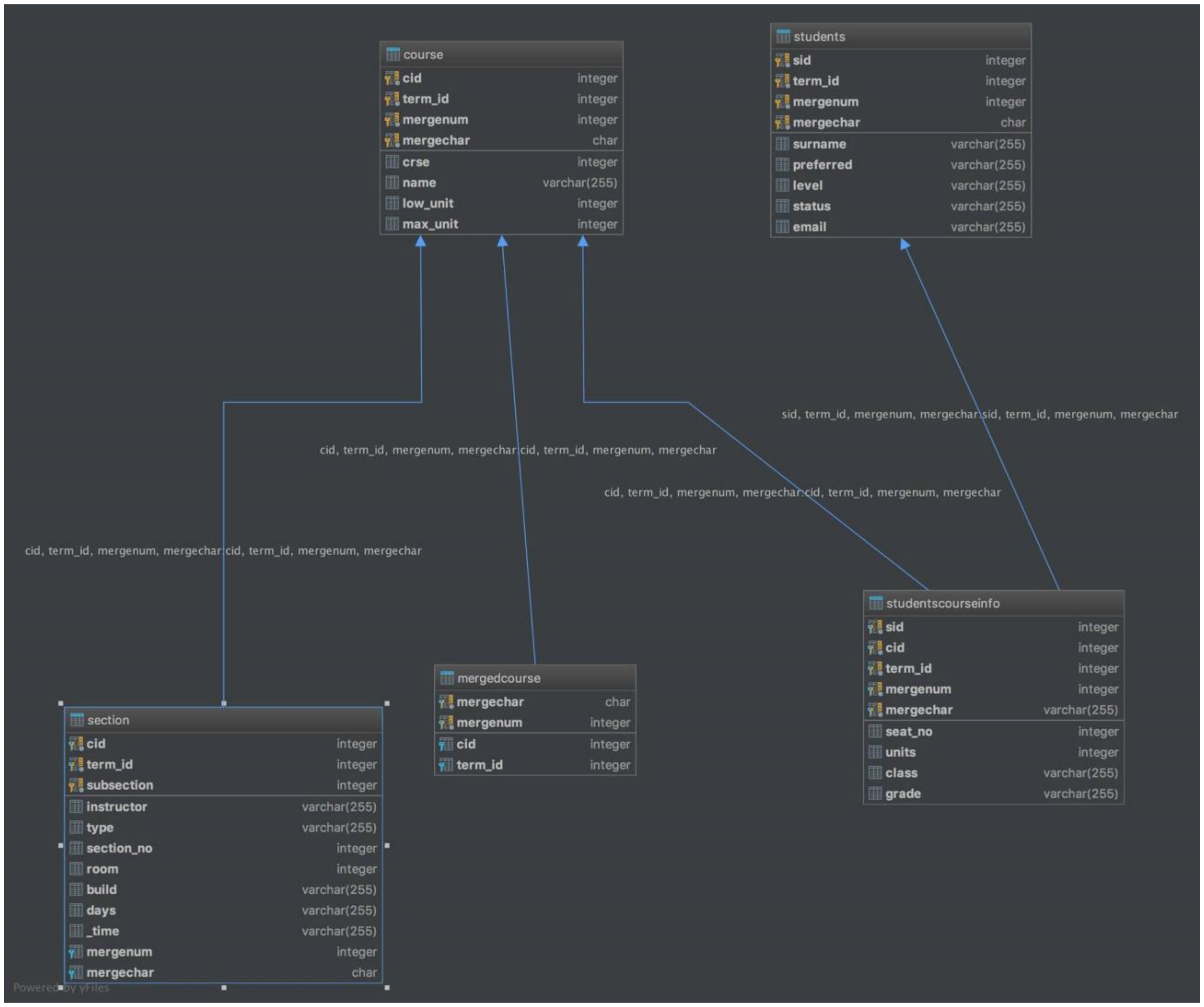
For our schema, we decided to split up the various attributes of the previous database into a few tables.

Our Students table contains information about each student on a term basis. This means that anything that stays constant during a term (such as major, and level) will be in the Students table. To differentiate between summer sessions, we added 2 attributes to Students, merge Number, and Merge Char. Merge Number and Merge Char are prevalent in all of our tables, they designate courses that were summer merges. All courses marked with 'C' are not summer merges, courses marked with 'A' and 'B' and that have the same Merge Number are courses that happened in 2 summer sessions. For information on students on a course basis, we have the table StudentCourseInfo. This links to Students with foreign key SID, and term id. With these 2 tables, we maintain all information about the students in the past. For updates, term based information would go in Students, course based information would go in StudentCourseInfo.

For each of the courses, we have a Course table, which describes information about each course. With CID and term id, we can link the term info to StudentCourseInfo. Each course can have multiple sections, and to account for that we have the Section table. The section table contains information about each section in the course, like the various the instructor that teach the course and meeting times and places for the courses. For classes that were not summer merges, we would have liked to make Merge Number always 0 or make it null, to make updates to the database easier, as now to avoid a conflict, you must insert a Merge Number 1 greater than the max merge Number, as the numbers were generated as we scanned the courses. Due to the rules of Postgre, we could not make Merge Number null for classes that were not summer merges.

Updates are relatively easy, student information about each term goes in Students table, student information about each course would go in StudentCourseInfo, the information about the course will go in Course, and section/meeting information would go in Section table.

This graphic shows our schema in detail: IGNORE MERGECOURSE TABLE



```
TABLE STUDENTS (
    sid INTEGER,
    term_id INTEGER,
    surname VARCHAR(255),
    preferred VARCHAR(255),
    level VARCHAR(255),
    email VARCHAR(255),
    major VARCHAR(255),
    mergechar CHAR,
    mergenum INTEGER,
    PRIMARY KEY (sid,term_id,mergenum,mergechar)
)
```

FDs:

Sid,term_id,mergechar,mergenum → surname,preferred,level,major

Sid,email → surname,preferred

Sid → surname,preferred

Email → surname, preferred

MVDs:

All those from promotion and complementation

```
TABLE COURSE (
    crse INTEGER,
    cid INTEGER,
    term_id INTEGER,
    name VARCHAR (255),
    low_unit INTEGER,
    max_unit INTEGER,
    mergenum INTEGER,
    mergechar CHAR,
    PRIMARY KEY(cid, term_id,mergenum,mergechar)
)
```

FDs:

CID,term_id,mergenum,mergchar → crse,low_unit,max_unit,name

Name,crse → low_unit, max_unit.

cid,term_id → mergenum,mergechar

MVDs:

All those from promotion and complementation

```
SECTION (
    instructor VARCHAR(255),
    type VARCHAR(255),
    section_no INTEGER,
    cid INTEGER,
    room INTEGER,
    build VARCHAR(255),
    days VARCHAR(255),
    _time VARCHAR(255),
    term_id INTEGER,
    subsection INTEGER,
    mergenum INTEGER,
    mergechar CHAR,
    PRIMARY KEY (cid,term_id,subsection,mergenum,mergechar),
    FOREIGN KEY (cid, term_id, mergenum, mergechar) REFERENCES COURSE (cid, term_id,
mergenum, mergechar)
)
```

FDs:
cid,term_id,subsection,mergenum,mergchar → instructor, type,section_no ,room,build,days,_time

MVDs: complement and promotion of FDs.

```
TABLE STUDENTSCOURSEINFO (
    seat_no INTEGER,
    sid INTEGER,
    cid INTEGER,
    term_id INTEGER,
    units INTEGER,
    class VARCHAR(255),
    grade VARCHAR(255),
    mergenum INTEGER,
    mergechar VARCHAR(255),
    status VARCHAR(255),
    PRIMARY KEY(sid,cid,term_id,mergenum,mergechar),
    FOREIGN KEY (cid, term_id, mergenum, mergechar) REFERENCES COURSE (cid, term_id,
mergenum, mergechar),
    FOREIGN KEY (sid, term_id, mergenum, mergechar) REFERENCES STUDENTS(sid,
term_id,mergenum, mergechar)
)
```

FDs:

Sid,cid,term_id,mergenum,mergechar → seat_no,units,class,grade,status

MVDs: Promotions, and compliments.

README:

For loader: Run the python program, type in the path of where .csv files are located.

For Queries: Uncomment function calls to the respective function to run queries.

For EXI: in exipd directory, go to build/gcc. Call make clean, then make all, then make examples. The executable can be found in bin/examples, it is called exipd. Note that the EXI parser is unfinished. Use absolute path of .exi file as command line argument.