

**数据库系统实验报告**

题 目 关系数据库应用系统开发实践

学 院 计算学部

专 业 软件工程

学 号 120L022109

学 生 李世轩

任 课 教 师 战徳臣

哈尔滨工业大学计算机科学与技术学院

2023.4

##### 实验环境

操作系统 Windows 11

编程环境 JDK8

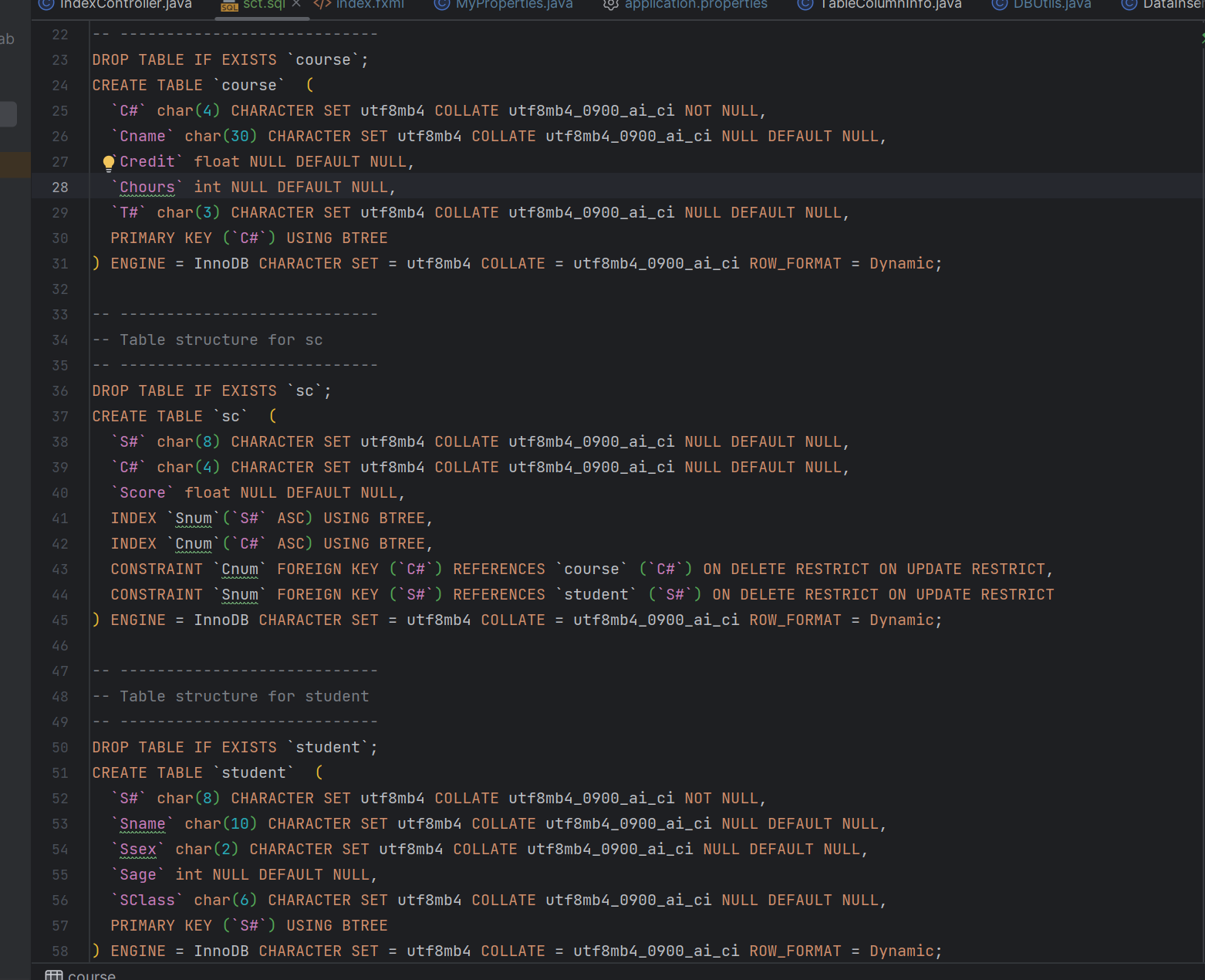
编程IDE IntelliJ IDEA

数据库 MySQL 8.0.32

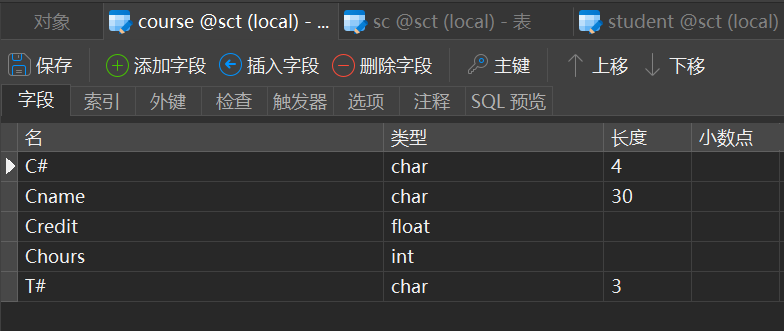
##### 各部分程序运行结果

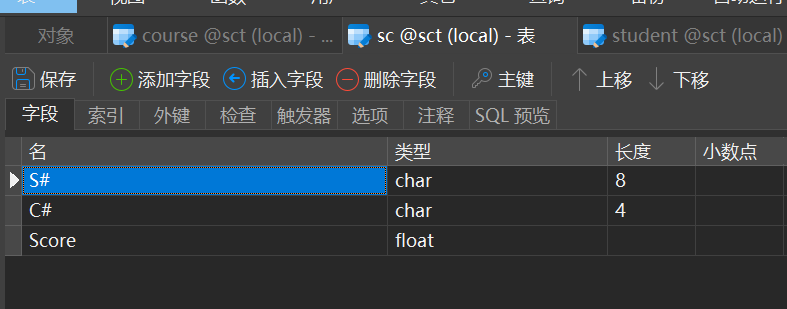
**【编程1】创建数据库与数据表:创建示例数据库SCT，创建示例数据表Student,Course,SC。**

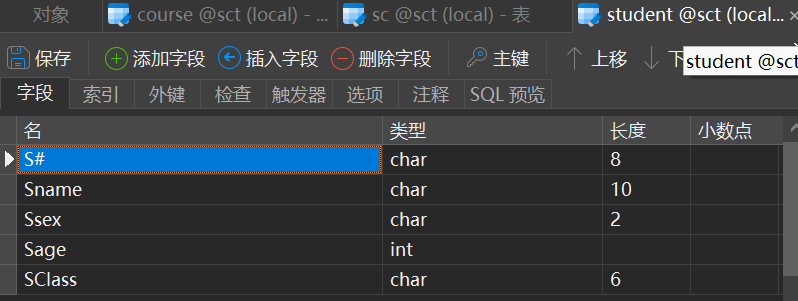
创建示例数据库SQL代码如下（包含在源代码sql路径下）



创建后结果如图

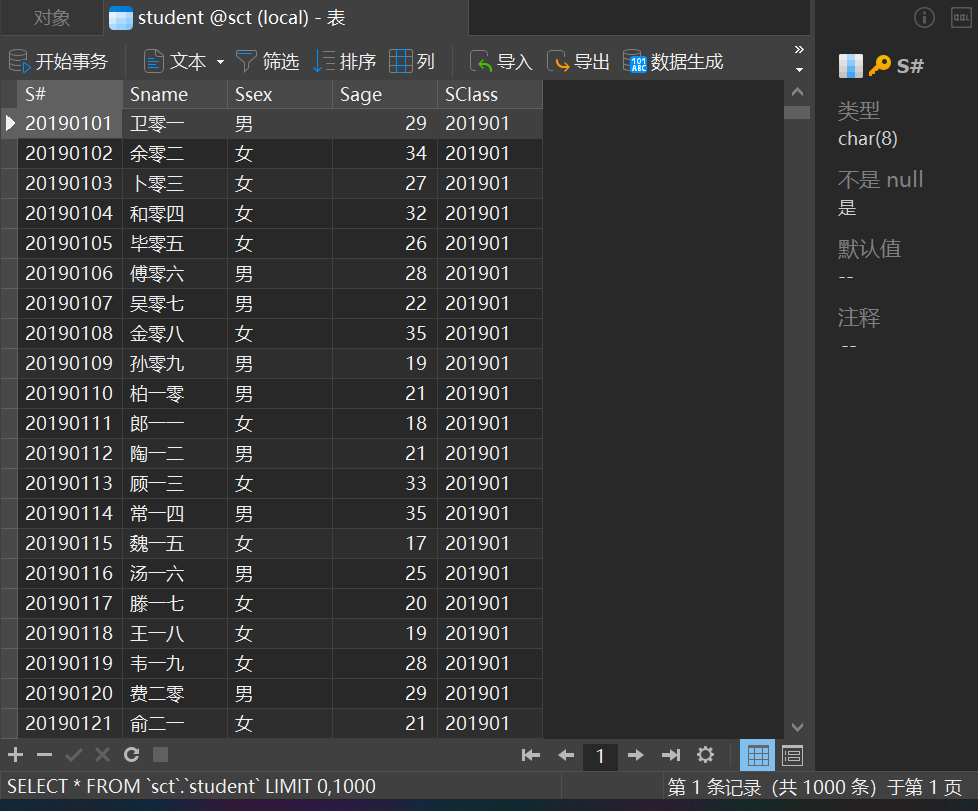


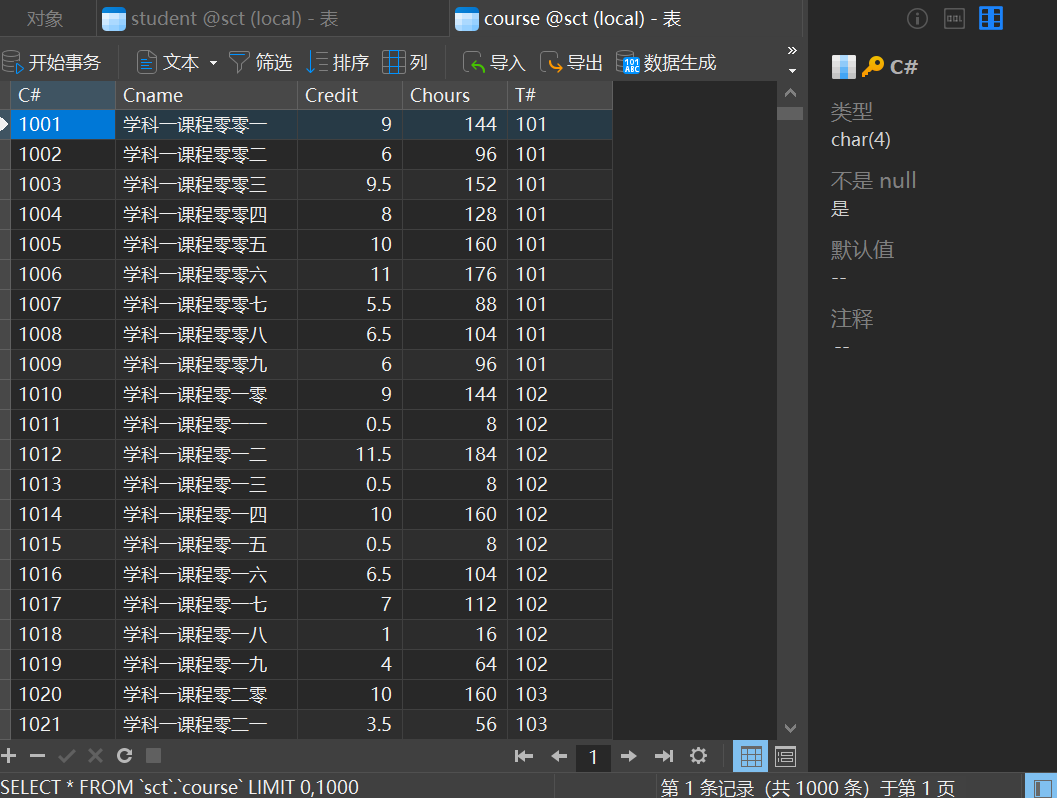


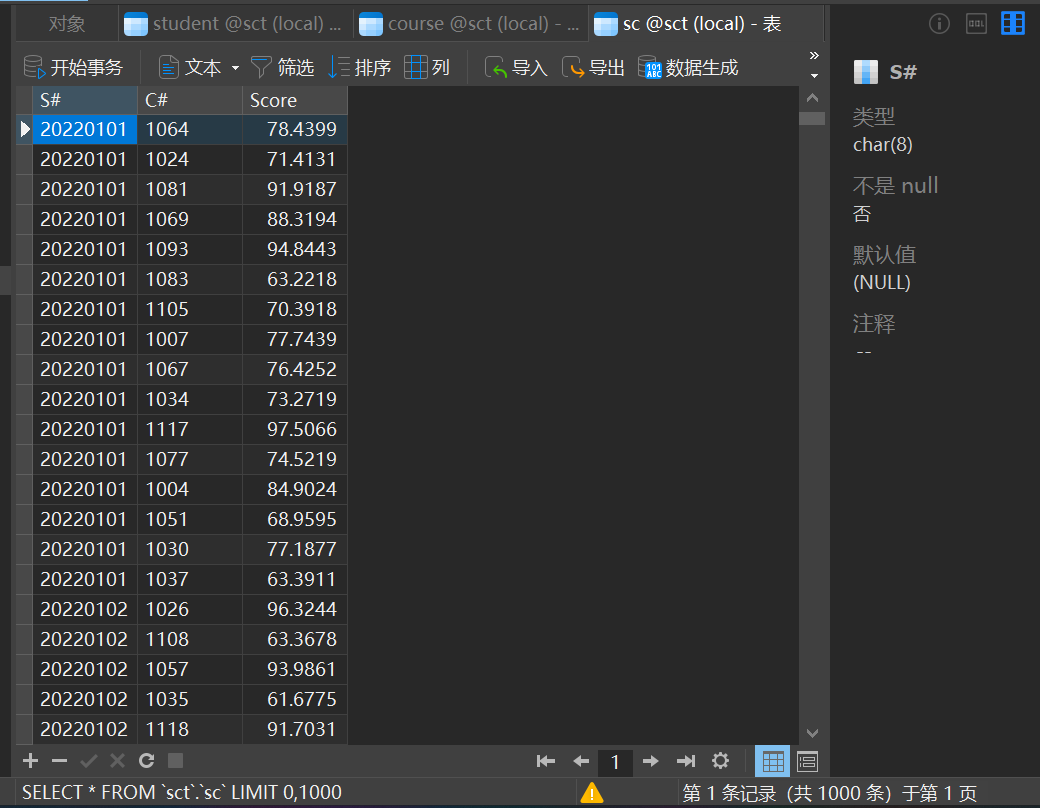


**【编程3】编制程序：批量产生示例数据库中各数据表的数据。**

插入数据后数据库各个表如图

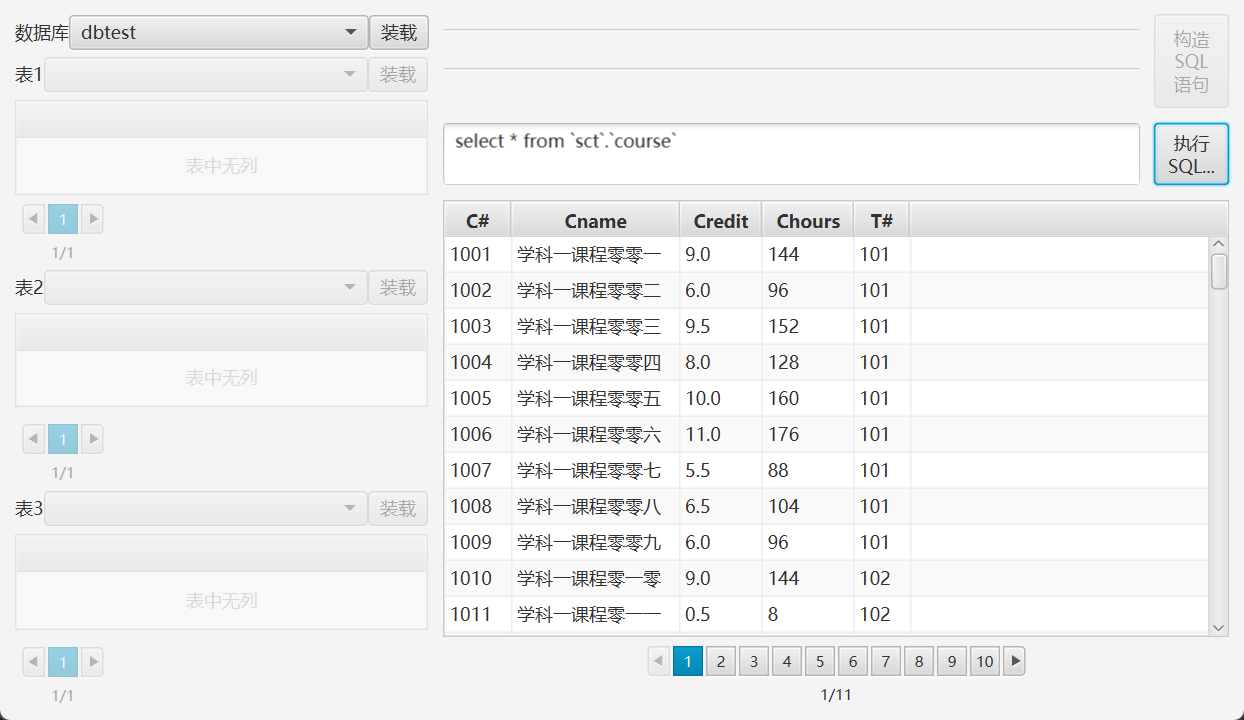


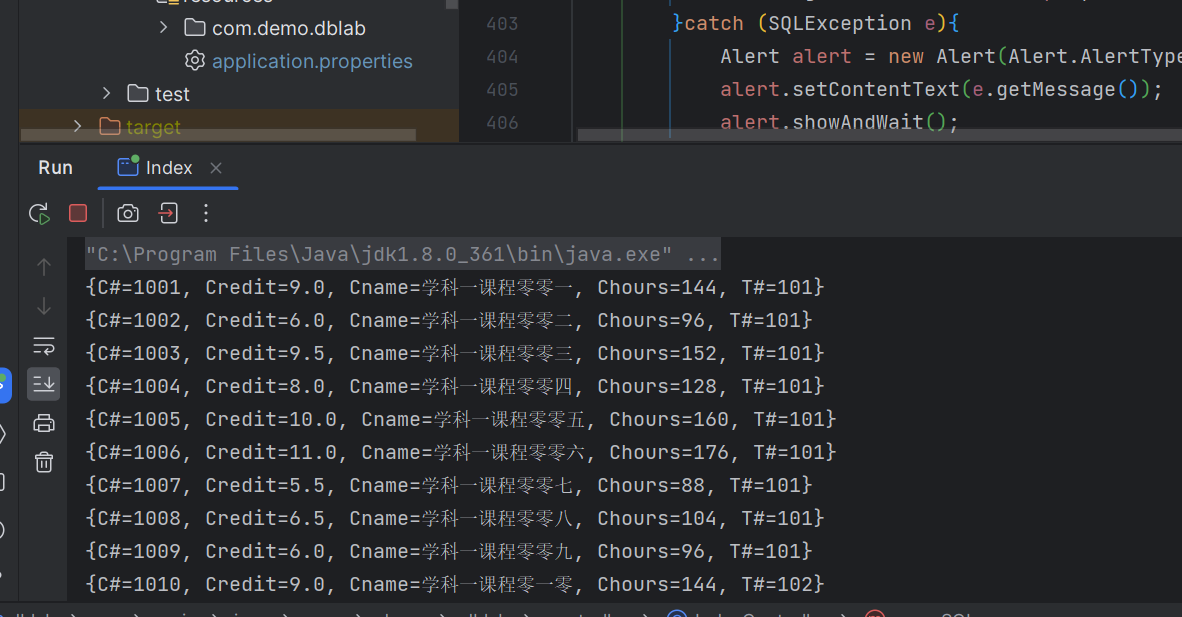




**【编程4】编制程序：书写SQL语句，并对SQL语句执行后的结果数据进行处理形成列表数据。**

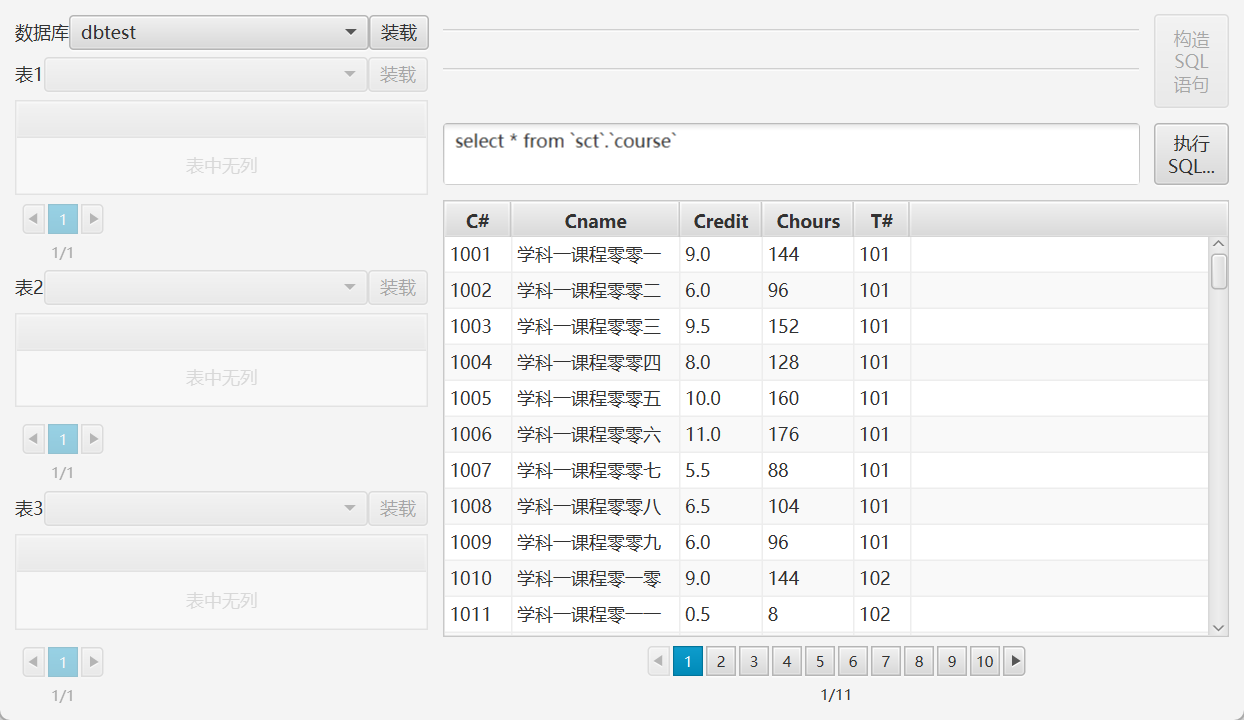
输入如图SQL语句后，得到如下输出。（该控制台输出在最终程序中被删除，只在界面中显示）





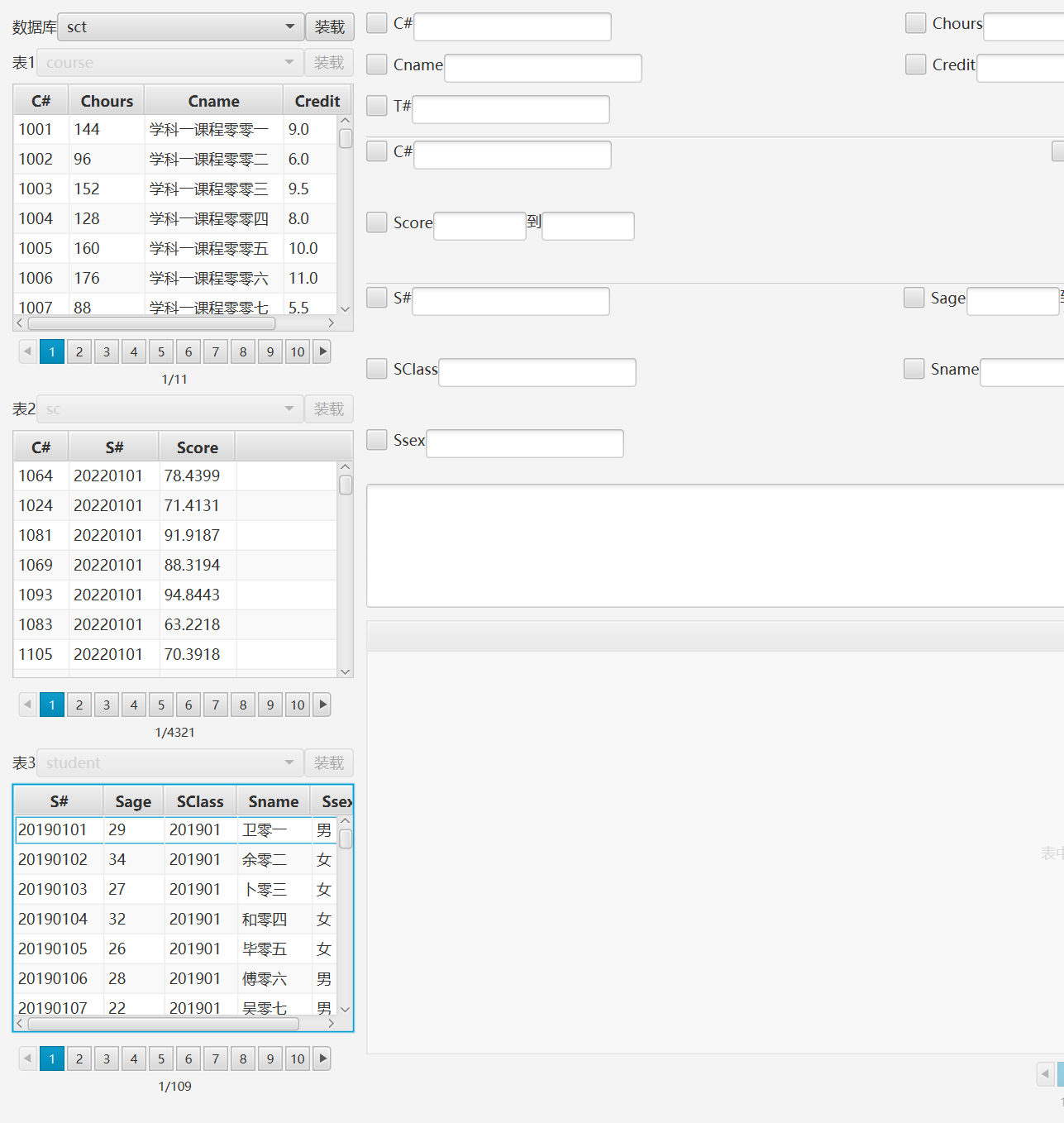
**【编程5】编制程序：显示如图5示意的界面。其中左侧分别装载数据库中的三个数据表（默认示例Student，Course和SC）。右上侧SQL语句框输出SQLString的内容。右下侧输出执行完SQLString的检索结果SQLResultList。**

如图为实现的界面



**【编程6】编制程序：以默认数据库SCT为例，分别检索3个表的数据记录，显示于图5示意界面的左侧部分。**

如图为分别显示的3个表的数据记录，这里为了检索效率的提高及前端负载的减小，进行了分页处理。

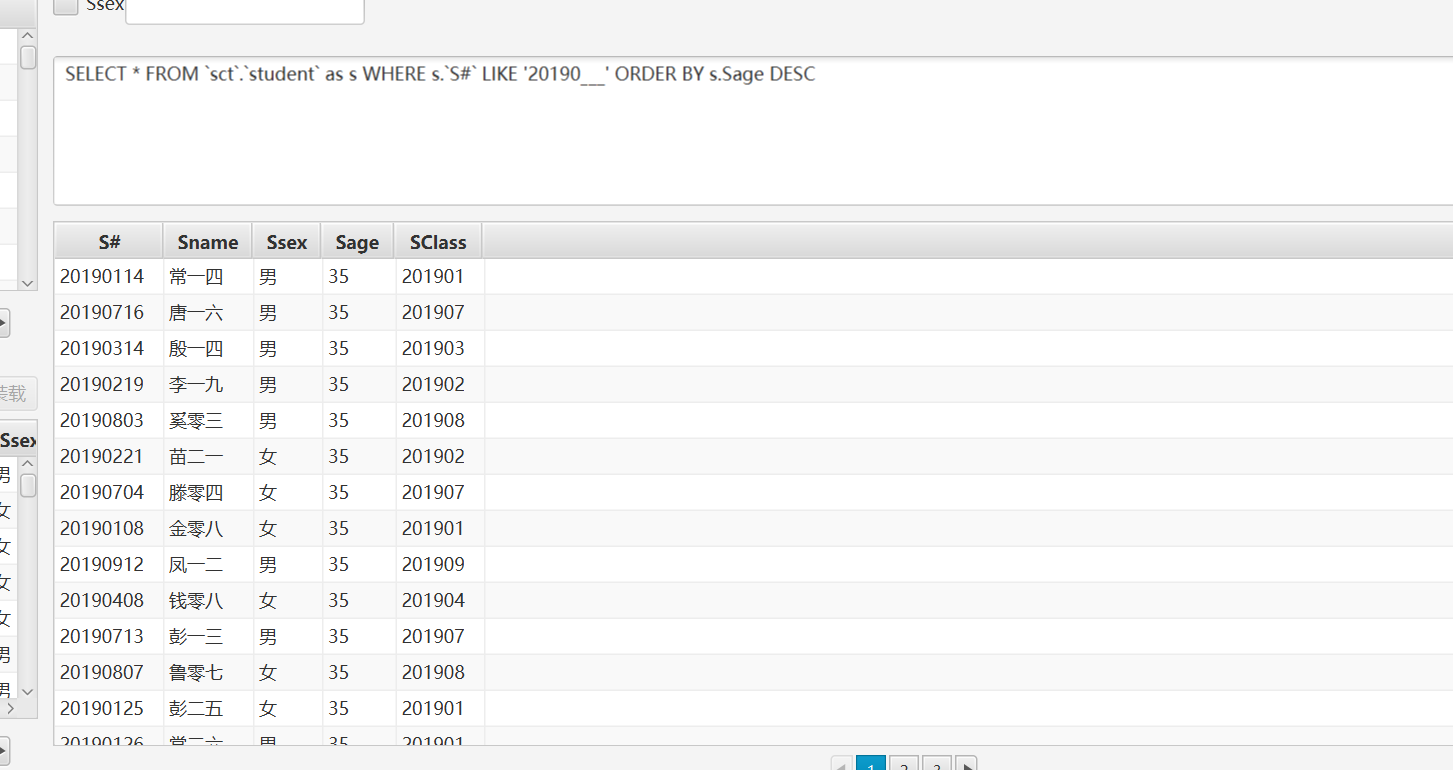


**【编程7】编制程序：针对Student、Course、SC三个表，完成各种形式的查询练习，为图5界面<执行SQL语句>编写相应的程序。**

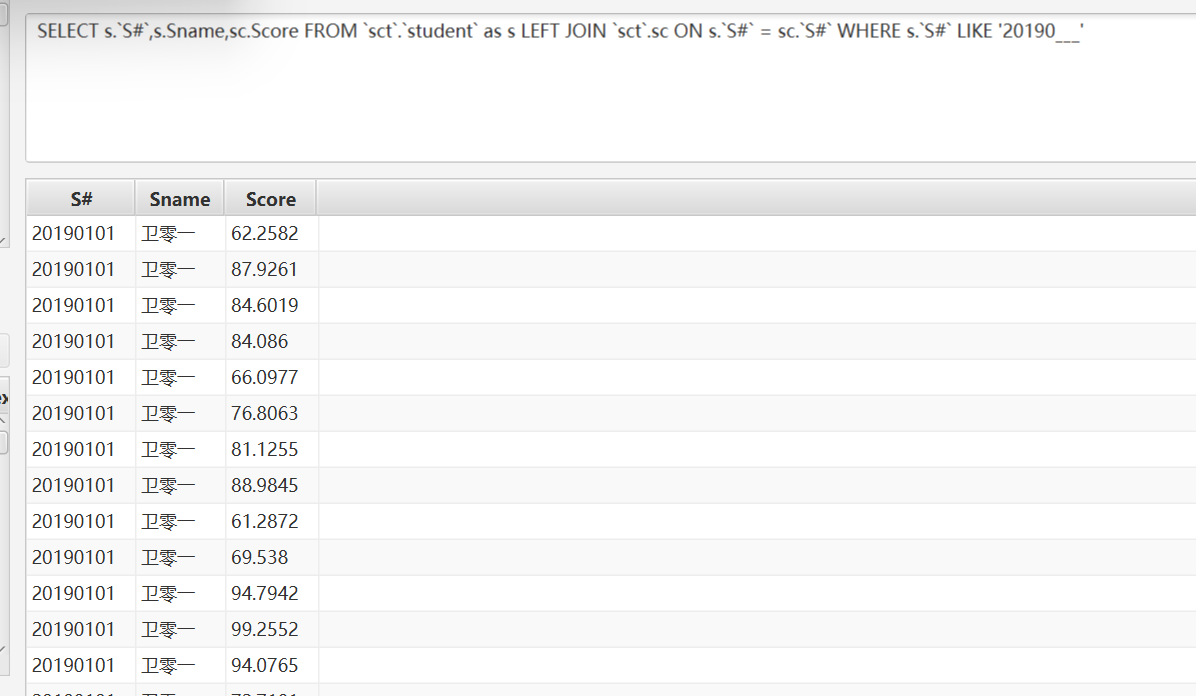
这里分别演示简单查询（重点：检索条件书写、模糊查询、结果排序）、多表连接查询（重点：连接条件书写）、分组聚集查询（重点：分组条件、求和/求平均/求最大最小/计数）、分组过滤查询（重点：分组过滤条件）等。

同样的，这里为了检索效率的提高及前端负载的减小，进行了分页处理。

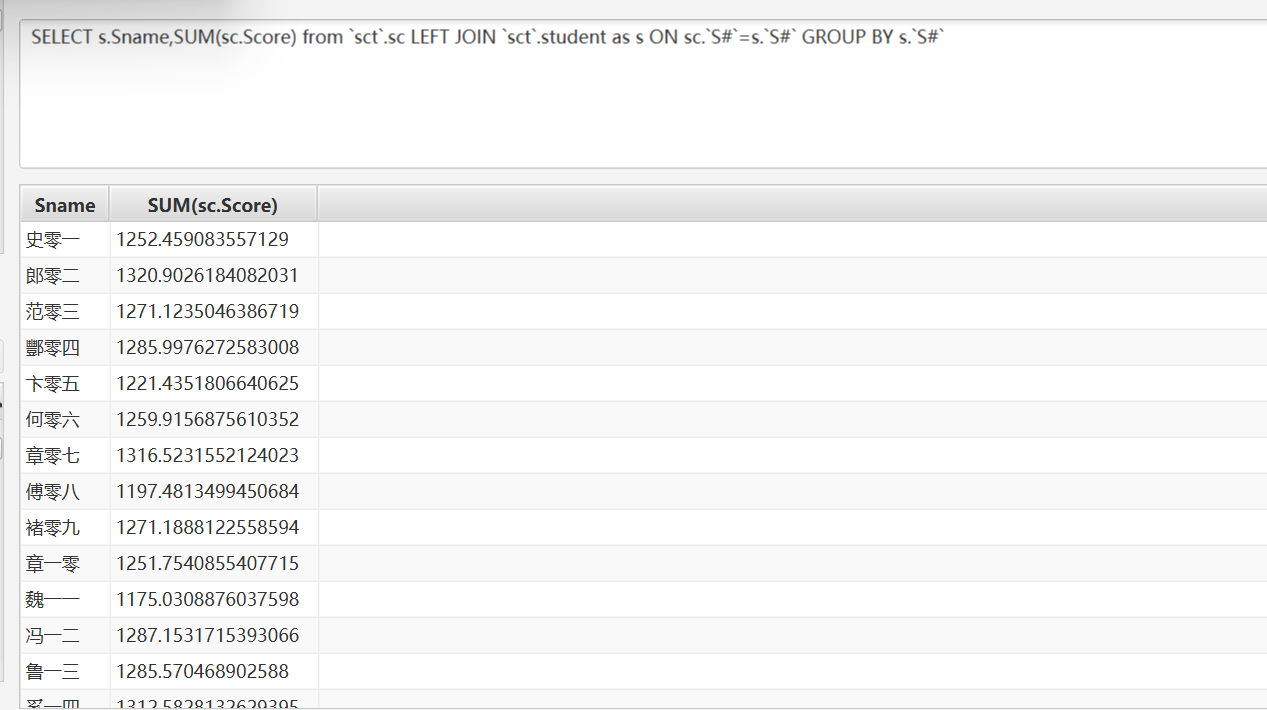
简单查询：

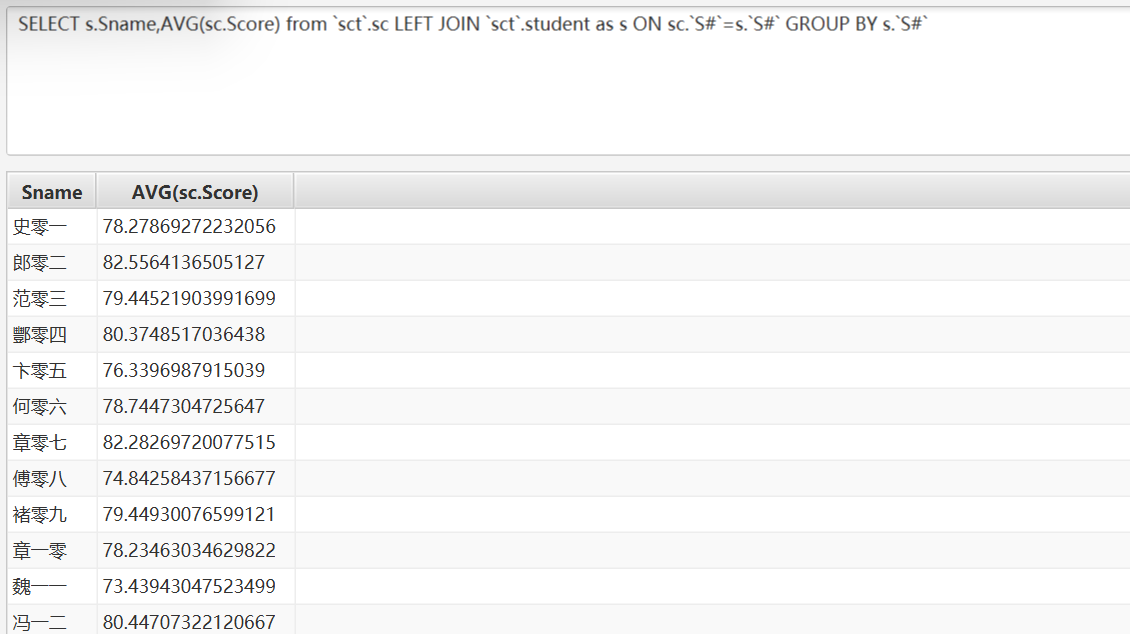


多表连接查询：

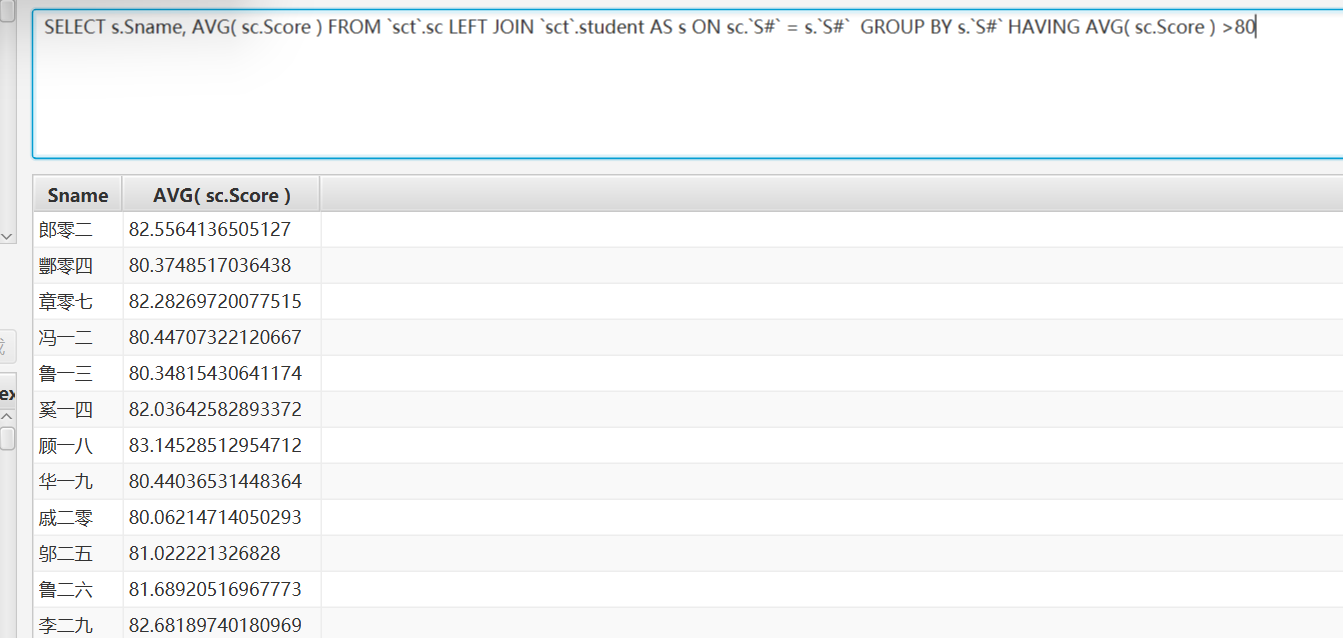


分组聚集查询：



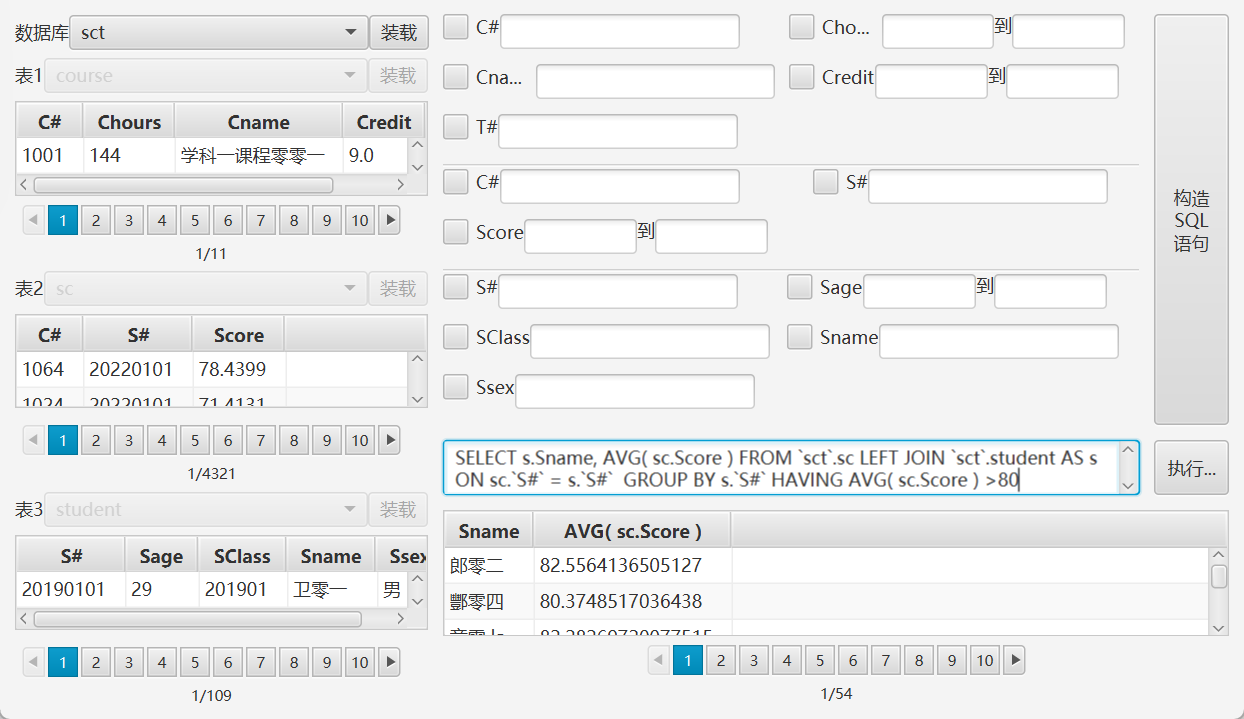


分组过滤查询：



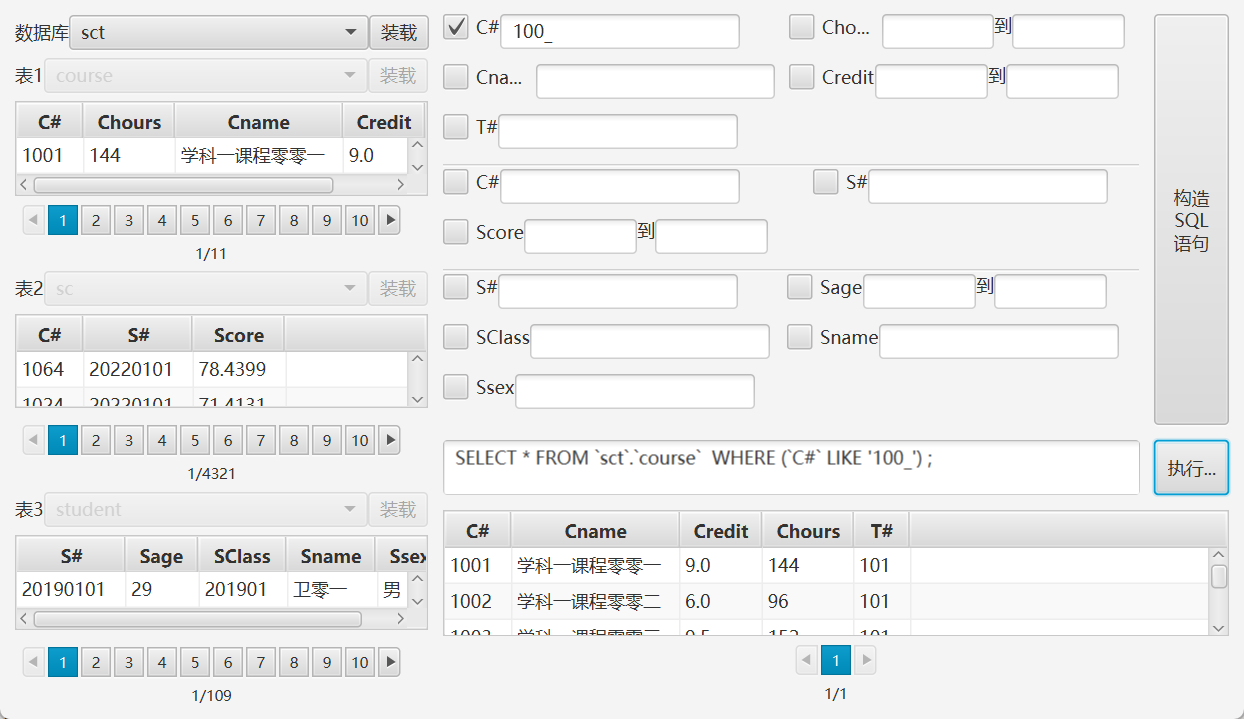
**【编程8】编制程序：在4-5节基础上，以示例数据库/数据表SCT/Student为例，增加交互式条件表达界面，如图6。**

如图为实现的界面



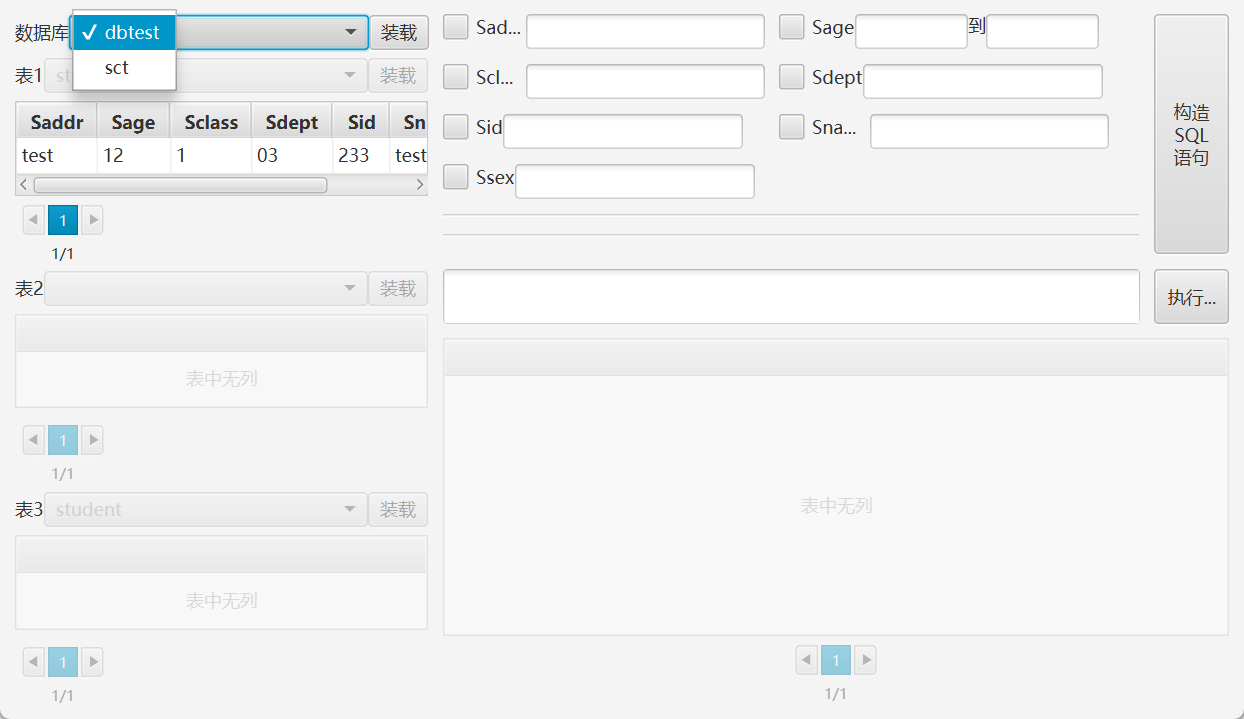
**【编程9】编制程序：为编程8的界面编写相关业务逻辑程序。**

如图这里选择条件为`C#` like ‘100\_’



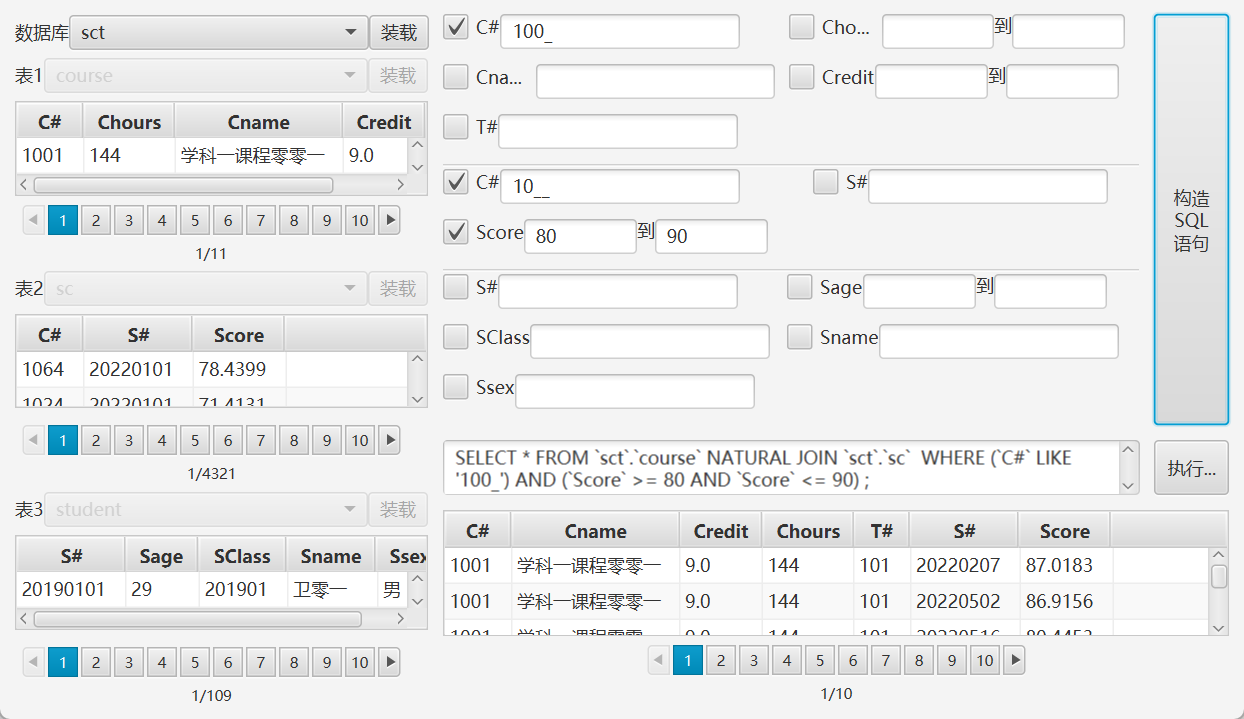
**【编程10】编制程序：能装载任何一个数据库及其中的不多于3个的表。**

如图，这里通过选择界面左上角的数据库，即可装载任意数据库中的不超过三个表。



**【编程11】以示例数据库为例编制程序：编写条件表达更丰富更复杂的用户交互应用程序。**

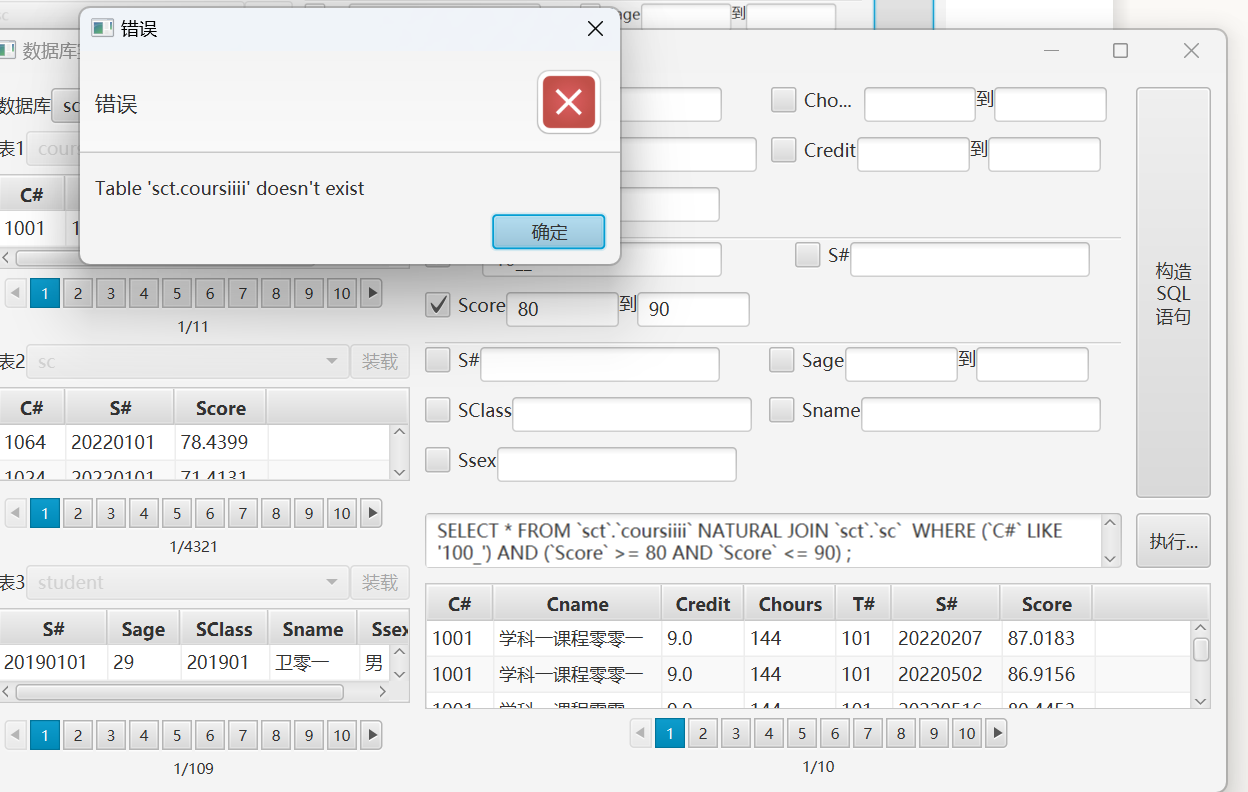
如图，这里可以通过选择不同的条件进行多表连接查询（因为目前界面连接条件表达能力有限，这里默认选择自然连接。）另外，当多张表的相同属性被同时选择时，只会接受上面表的属性。如当前界面中国Course表和SC表都含有C#属性，那么在构建SQL语句时，只会选择上面的C#并进行相应构造。



**【编程12】编制程序：数据库访问状态监测与报警。**

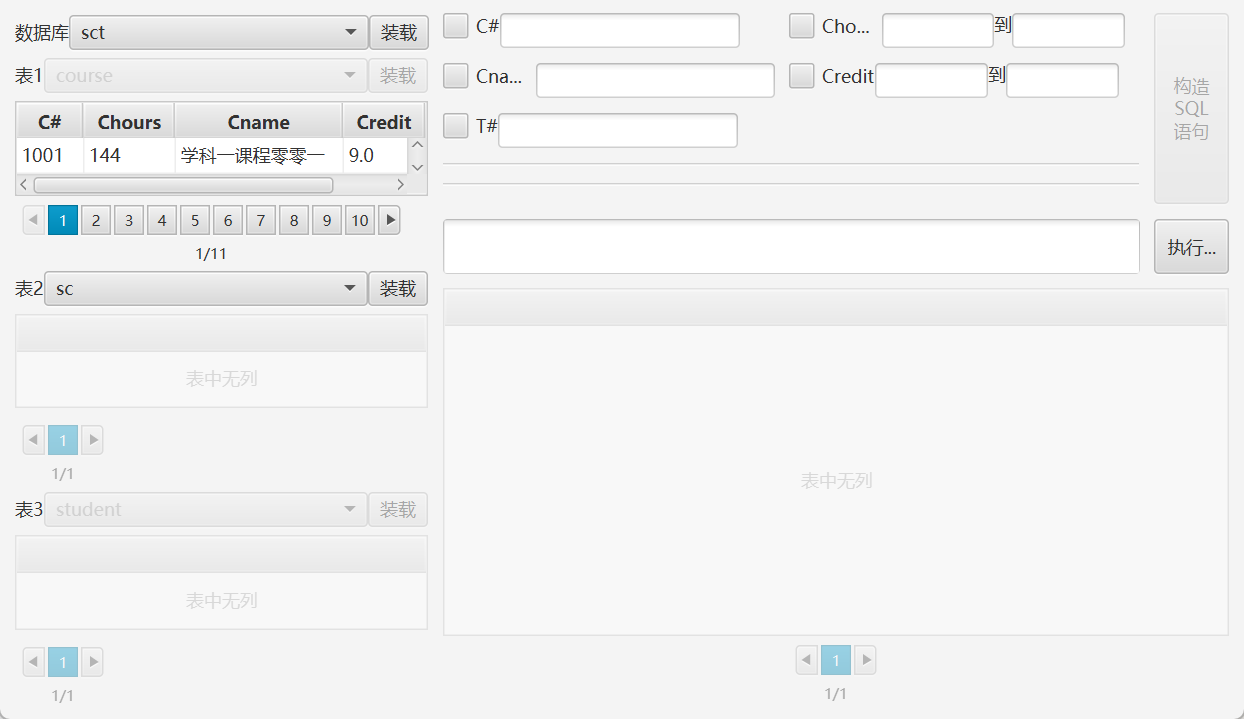
当输入语句有错误时，或者执行过程中有任何异常状态，会通过弹窗进行提醒。

如图，这里在SQL语句中将正常的‘course’改成了‘coursiiii’。



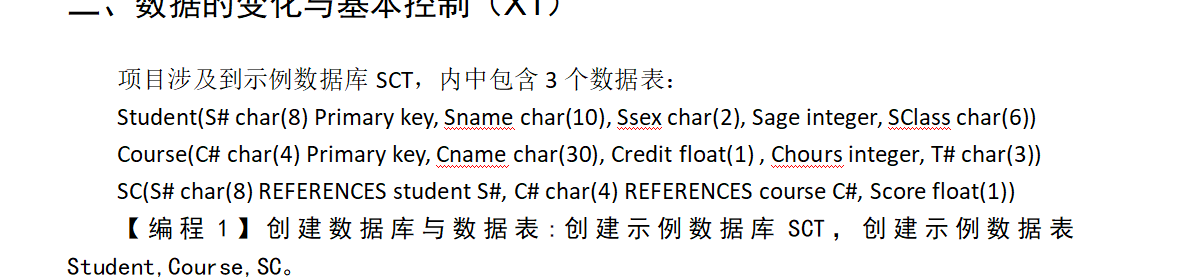
**【\*编程13】编制程序：可对任何表/任何字段进行检索条件表达的用户交互查询应用程序。**

当前界面的检索条件就是由装载的表动态产生的（目前只对字符串类型，整数类型，浮点数类型做了适配，其他类型未做处理）。比如，当前界面只装载了sct中的course表，所以条件构造区域只有其相对的条件。

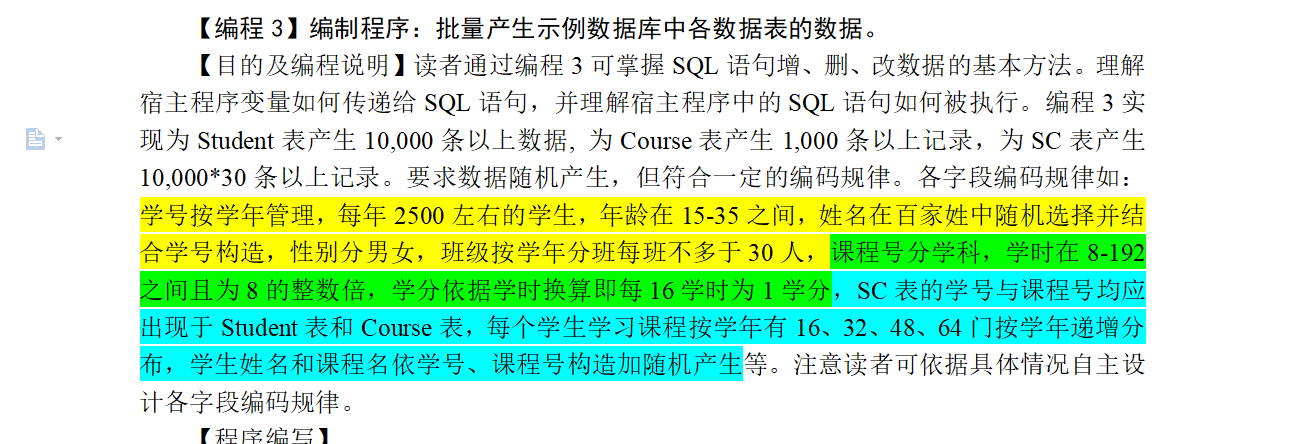


##### 程序的主要数据结构

库表设计完全参考指导书中给出的示例表。具体实现前面已经给出。

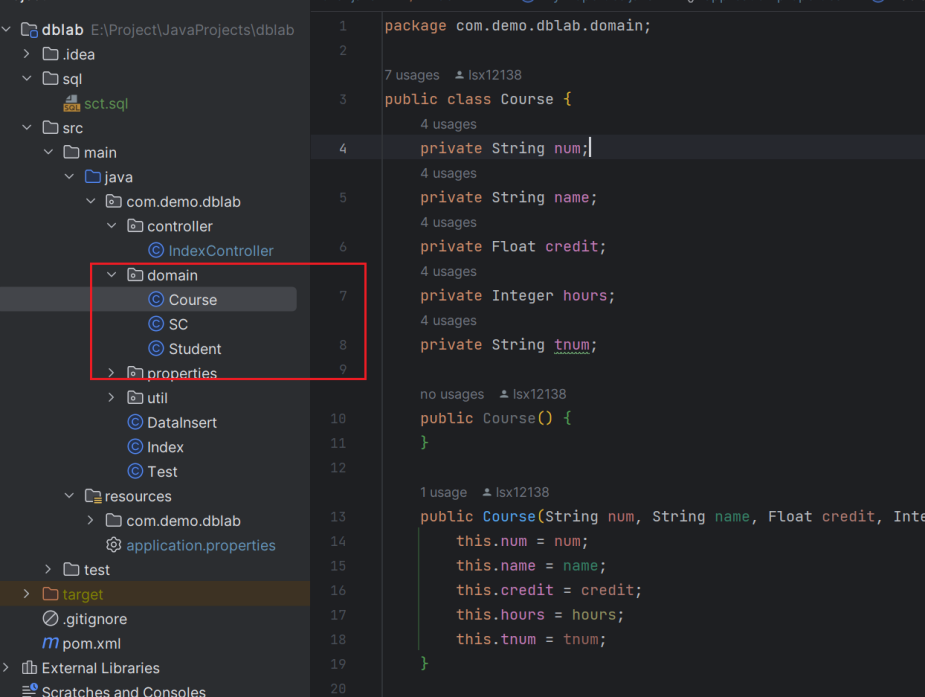


记录生成方式主要参考编程三中的要求。



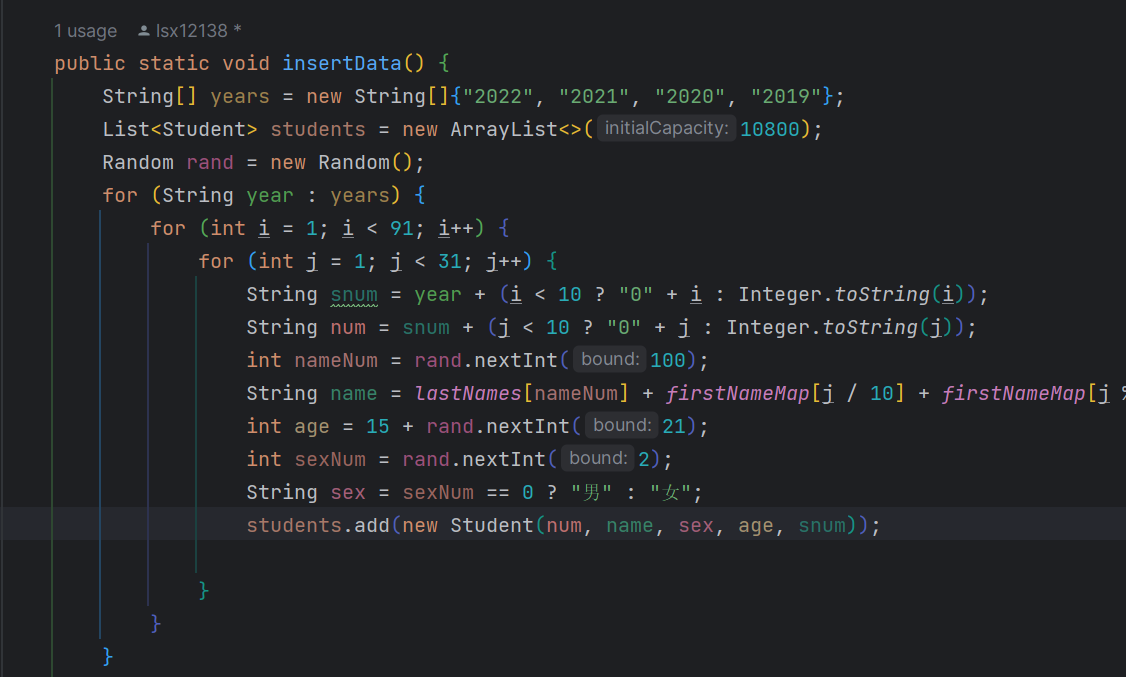
数据生成方法介绍：

首先对应表结构构造三个实体类。

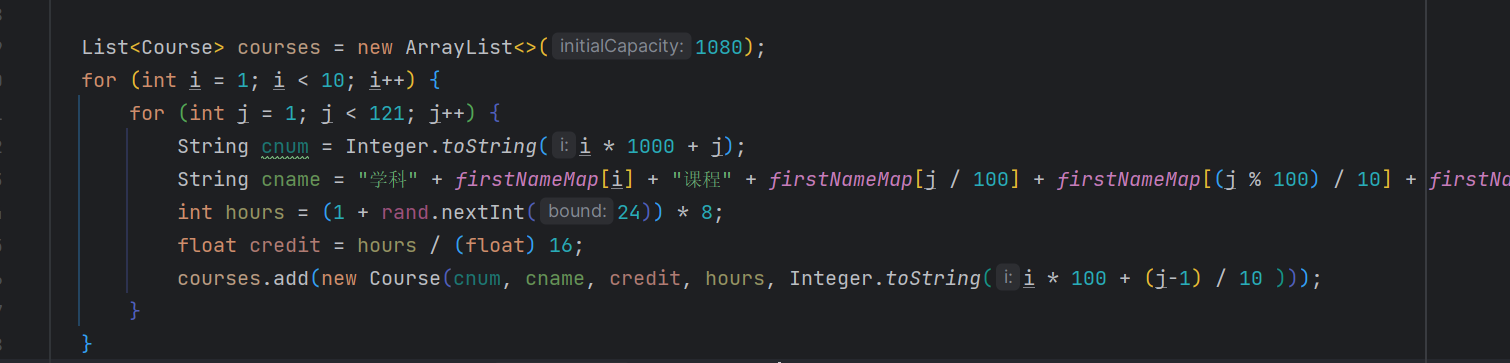


然后根据一定规律生成三种实体，并插入到数据库中。

首先关于Student的生成，这里按照4个学年，每个学年90个班，每班30人的方法生成，对应的学号前4位表示学年，接下来两位表示班（也可以将前一位表示专业，后一位表示班），最后两位表示学生在班里面的序号。学生姓名中，姓在百家姓中随机生成，名由学号最后两位对应生成，如后两位为10则其名为一零。学生的年龄，性别都随机生成。



然后关于课程的生成，按照9个学科，每个学科开设120门课生成，课程号第一位表示学科，后三位表示课程号。课程名由课程号生成，如课程号为1001，则其课程名为学科一课程零零一。课时随机生成，学分按照课时生成，教师号T#按照每个教师开设10门课生成，即教师号第一位表示学科号，后两位表示教师号。



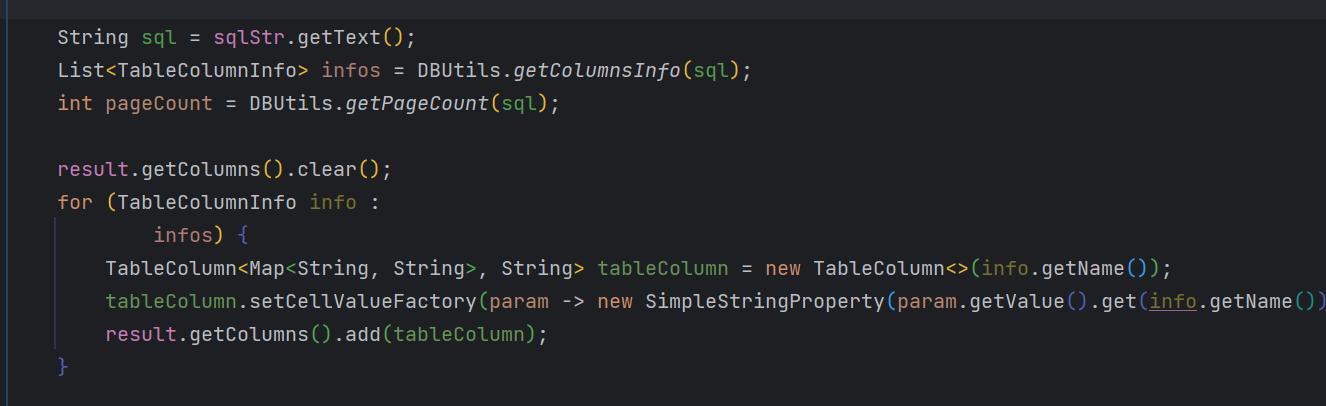
最后是SC记录的生成，其中S#的生成与Student中相同，C#从上面生成的课程中按照学科随机选择，另外按照同一S#对应的SC记录按照学年递增。

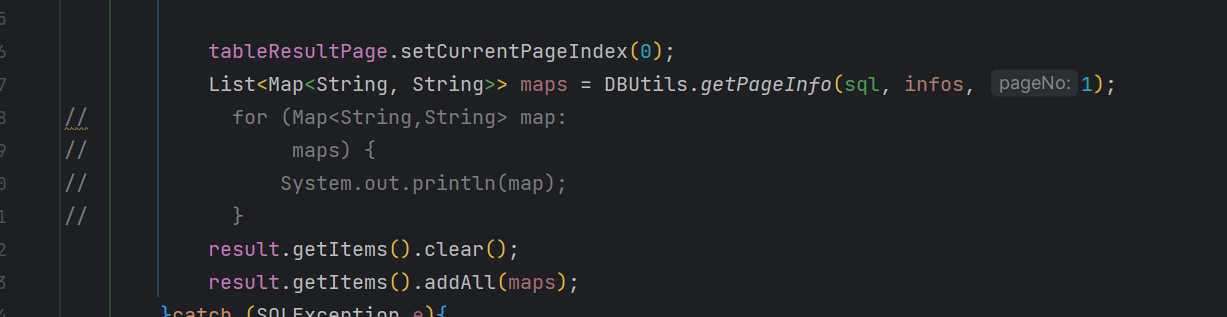


##### 系统设计思路

SQL语句执行模块

首先是SQL语句的执行方面，主要由方法querySQL实现。

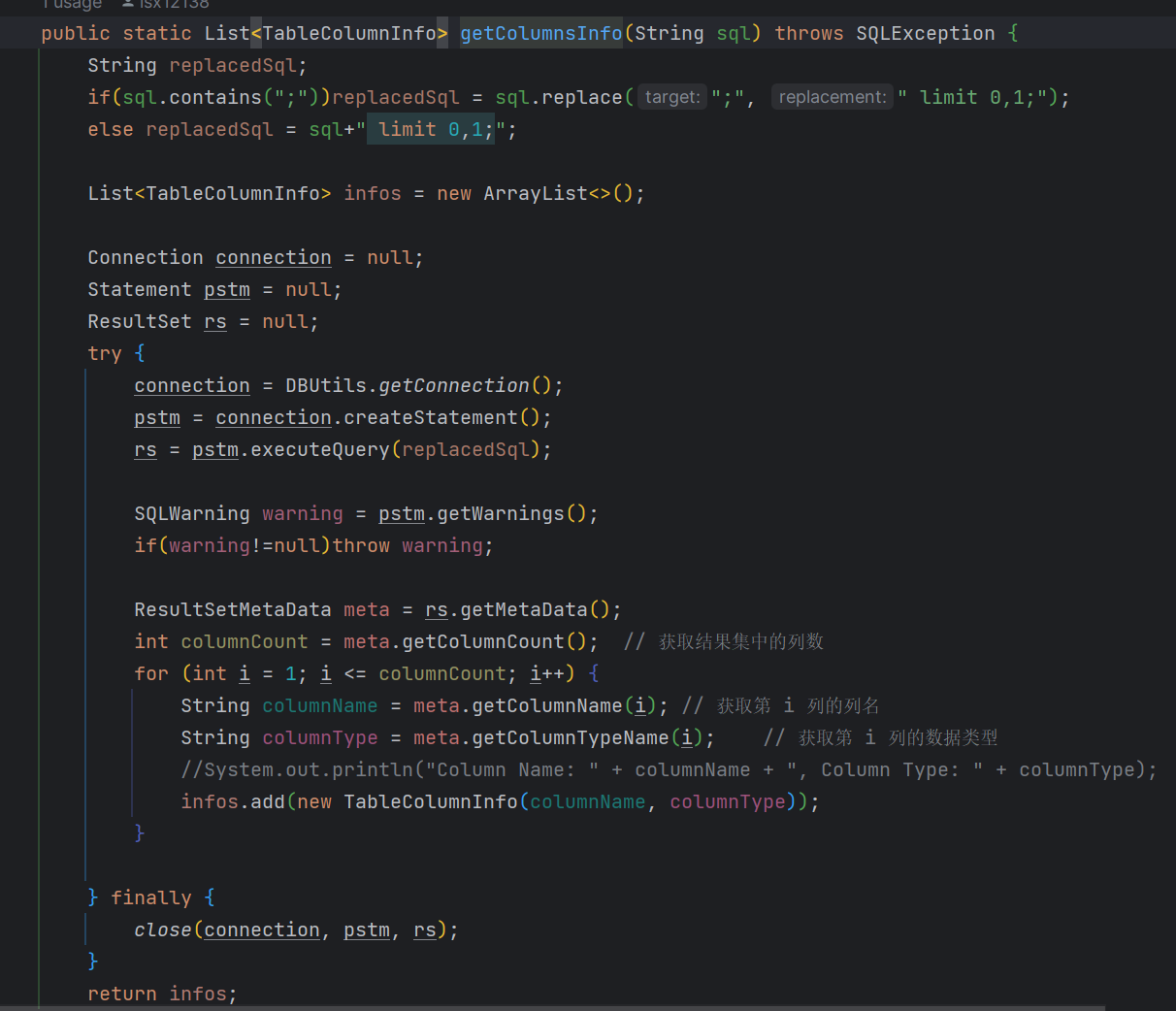




首先获取输入框中的SQL语句，然后获取该语句的查询结果对应的列信息，页数（默认每页100条），然后根据查询第一页数据并根据列信息将其显示到界面上。

获取列信息方法如下：

首先在语句后面拼接limit 0,1（这里主要是考虑到负载问题，使用该方法避免同一条语句的多次执行），然后执行语句，从结果中获取其数据字典ResultSetMetaData，解读其中信息并将其放到自定义的数据结构中（该方法在装载数据表时同样会用到）。



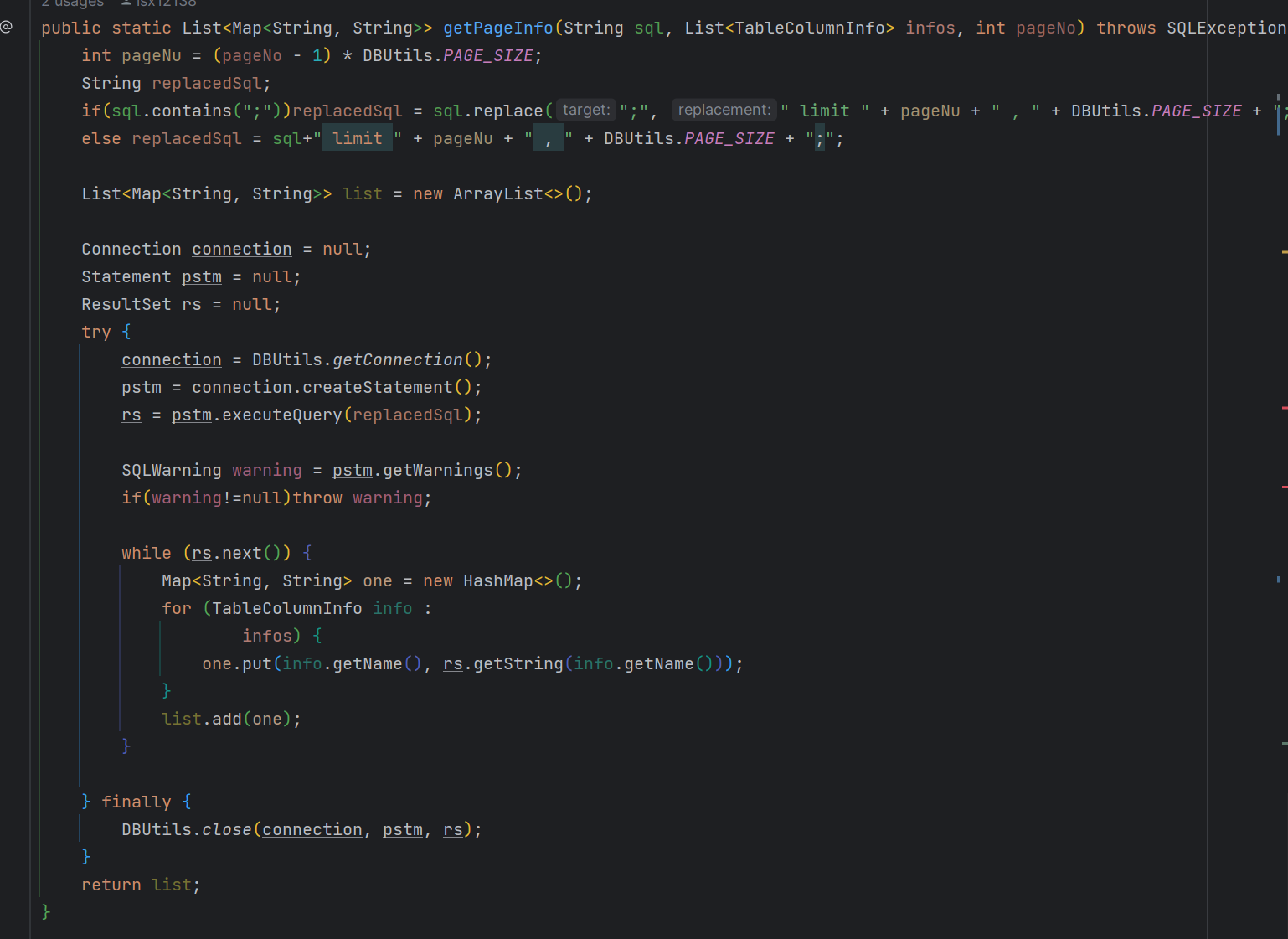
获取页数方法如下：

该方法思路为，首先去除输入sql的分号，然后将其当作子查询，将其嵌套到获取count的语句中，并根据得到的结果计算页数。



获取页面信息的方法如下：

首先对输入的SQL进行处理，将limit子句进行拼接，然后执行语句，并根据输入的列信息将查询结果放到List中。（这里使用List<Map<String,String>>存放查询结果，方便动态生成。每一个map表示查询结果中的一行）。



动态加载表并构造条件查询界面模块

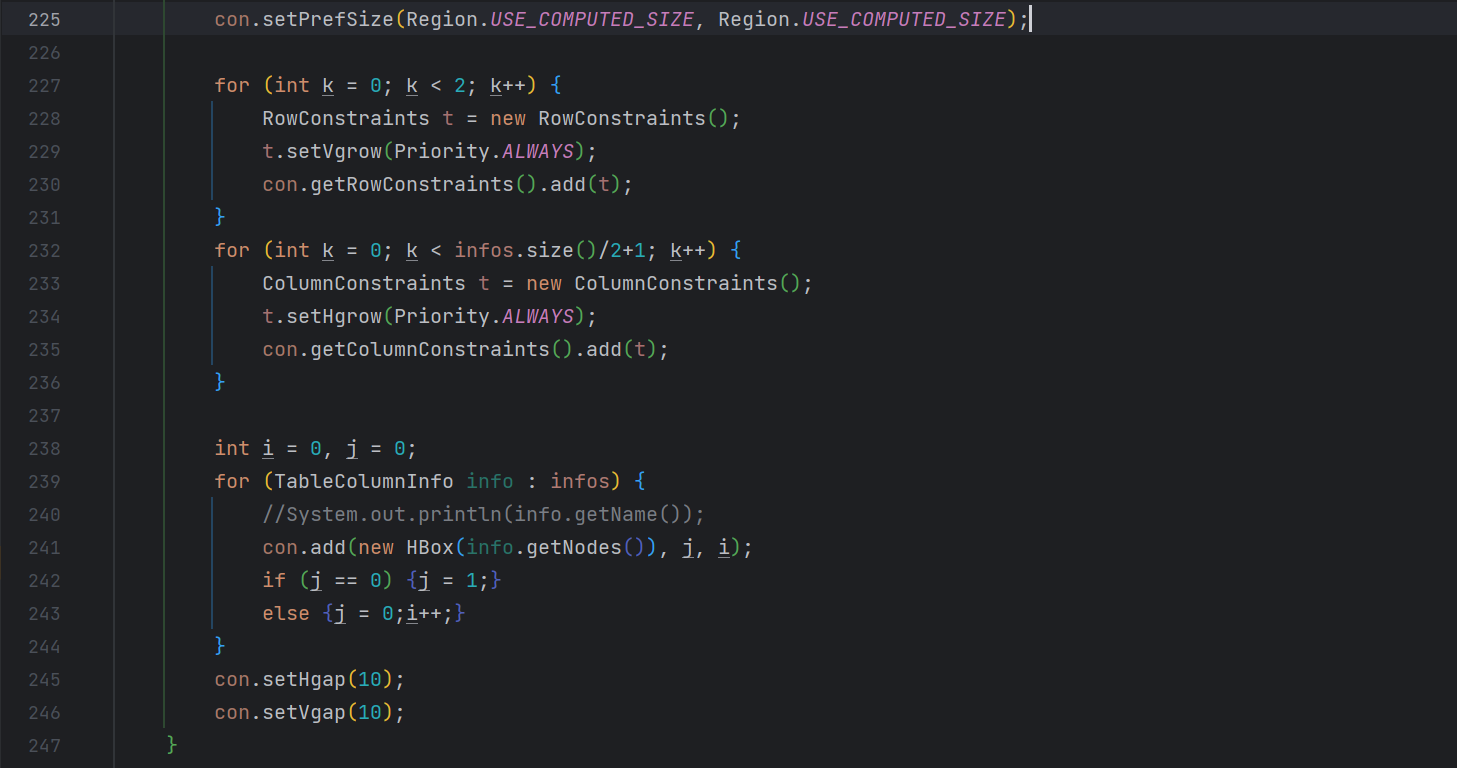
该模块功能主要由sameStepForLoadTable方法构成。这个方法主要因为加载三个表时，会有大量的重复动作，而单独抽离出来的。

方法思路如下：

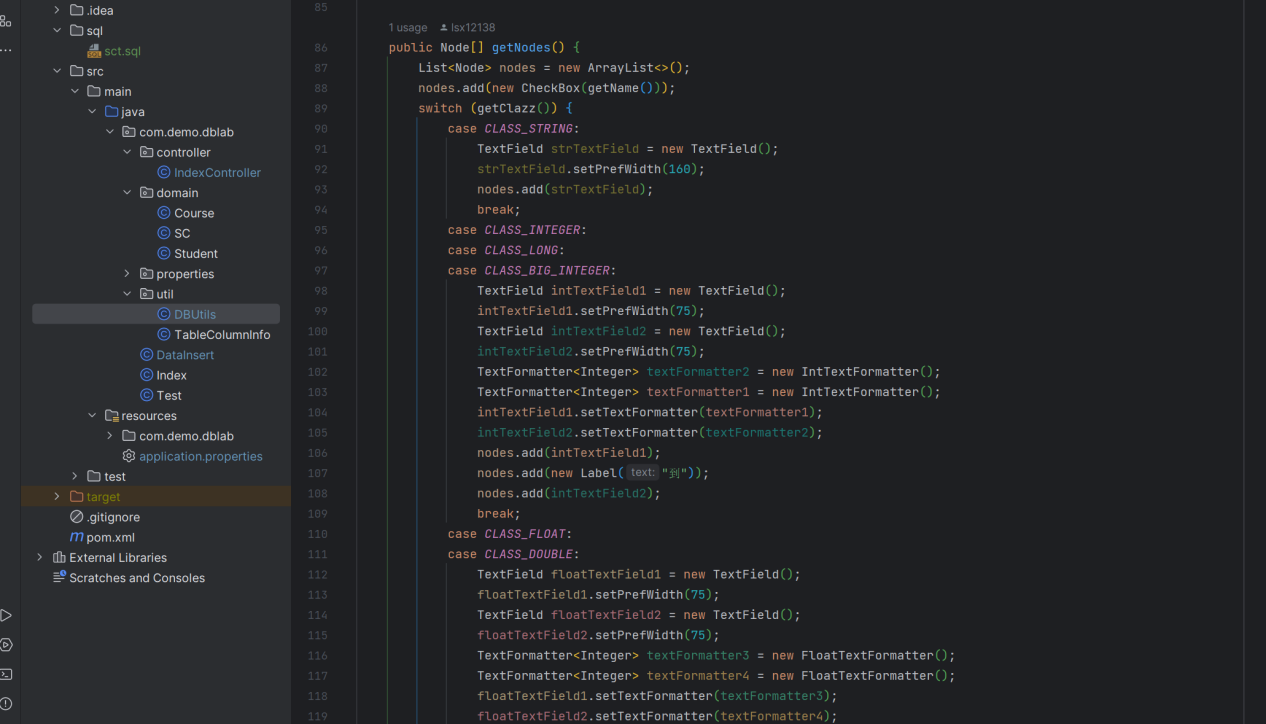
首先，根据表名获取列信息，页数，页面信息，并将其显示到界面上，该过程与上述显示sql语句执行结果的过程相似，这里不再赘述。



然后是，根据获取的列信息构造条件输入界面，这里使用栅格布局，根据列信息构造一个栅格中的内容。



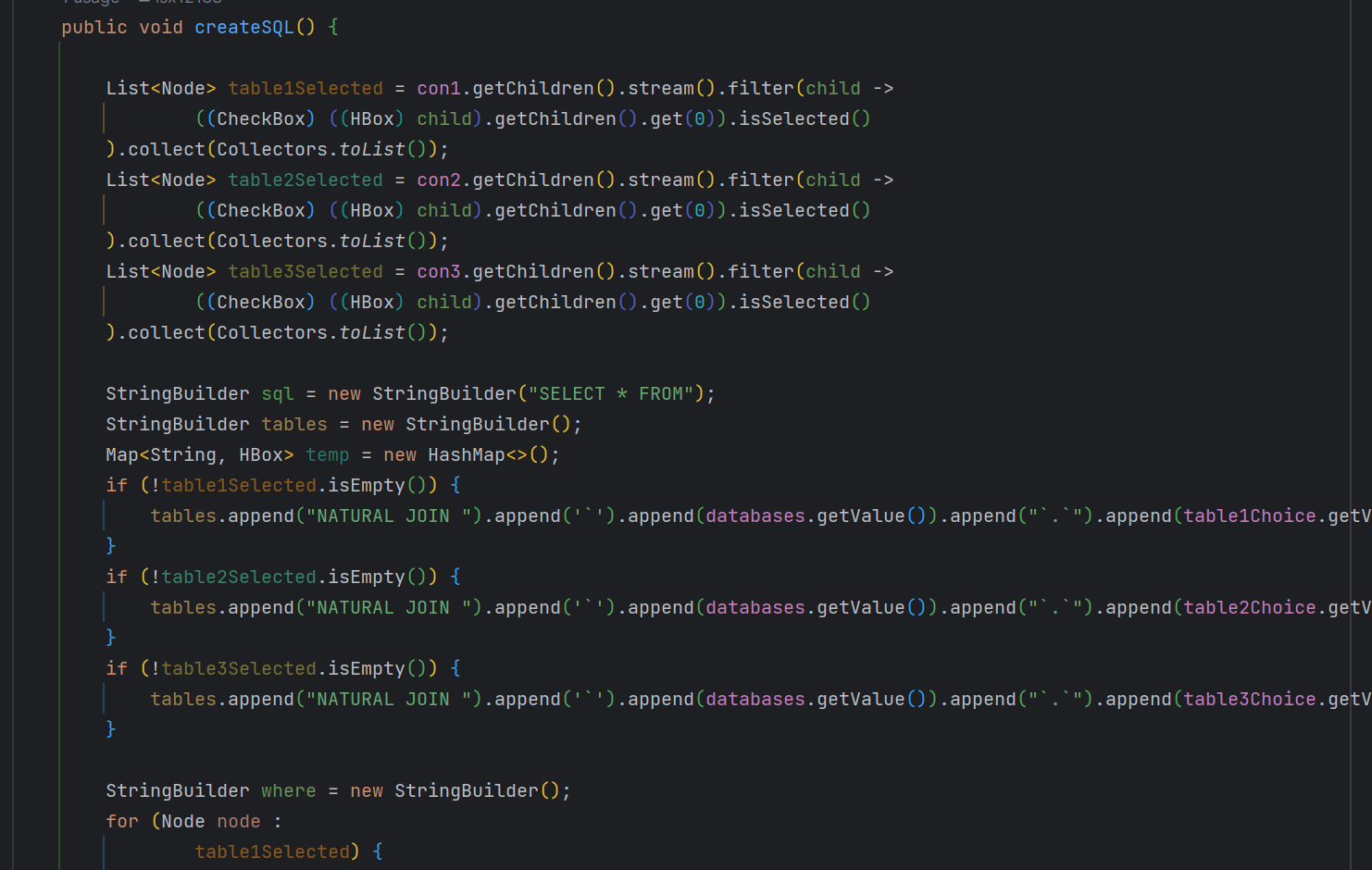
这里涉及到的根据列信息构造栅格的方法如下，其思路就是根据不同的类型，构造不同的界面。比如，如果类型是字符串，则简单的添加CheckBox和TextField，如果是整数或者浮点数，则除了CheckBox外再添加两个TextField，用来表示上下限。



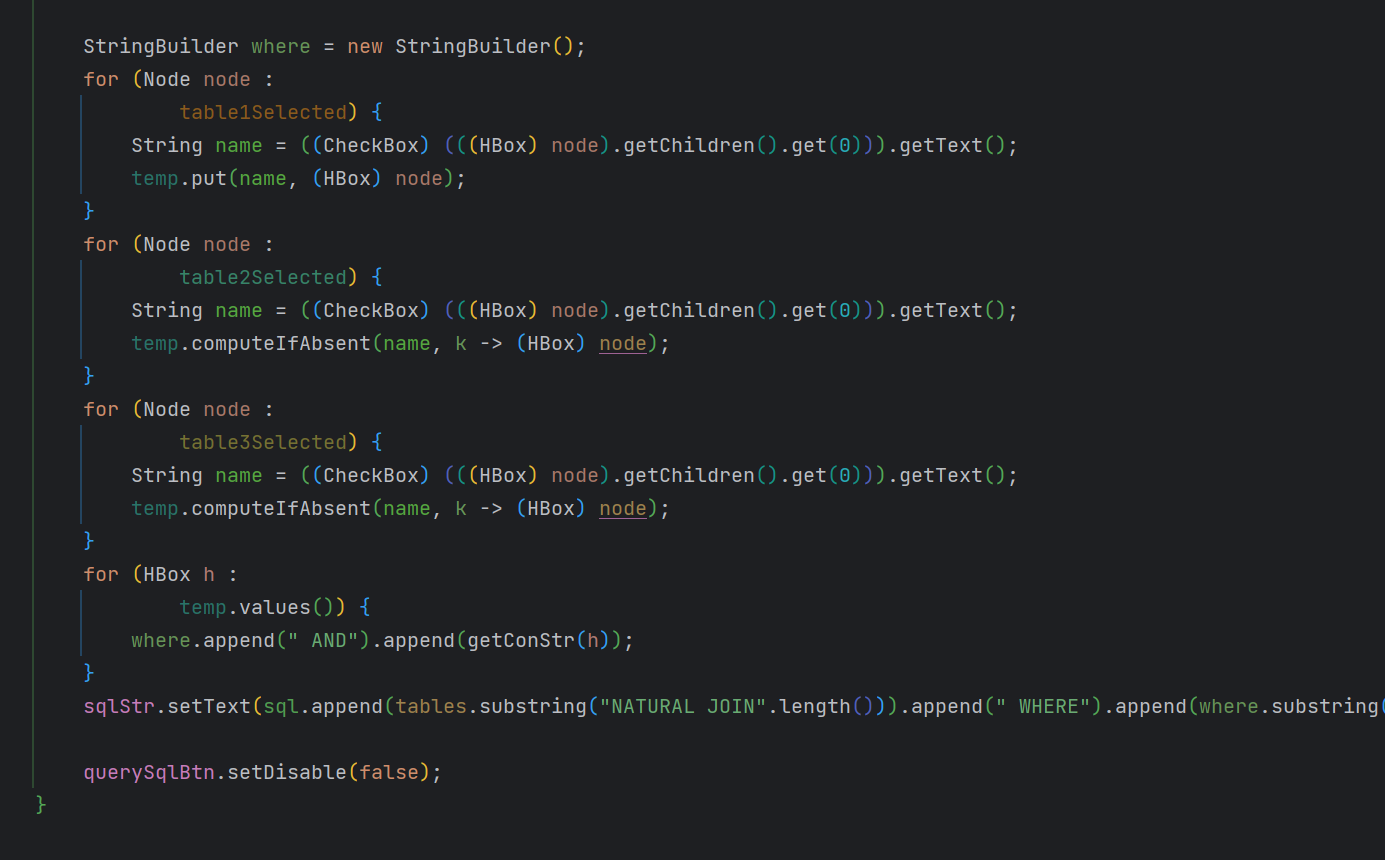
动态构造SQL语句模块

该模块由createSQL方法构成。

该方法有两个部分，第一部分获取需要连接的表名（默认进行自然连接）。



第二部分根据选择的条件构造where子句。



同样的，这里根据不同的类型进行不同的处理，若是字符串类型则判断其中是否有‘%’或者‘\_’，若有则拼接like进行模糊查询，若无则进行精确查询。若是数字类型，则拼接分别上下限。

