# UML Class Diagram

**<<interface>>**
**Database**

+createDatabase()
+executeStructureQuery(): bool
+executeQuery()
+executeUpdateQuery()

---

**SyntaxChecker**

-pattern: Pattern
-matcher: Matcher

+createTable(): bool
+createDatabase(): bool
+dropTable(): bool
+dropDatabase(): bool
+select(): bool
+selectAll(): bool
+selectWithCondition(): bool
+selectAllWithCondition(): bool
+delete(): bool
+deleteAll(): bool
+insert(): bool
+insertWithoutColumnsName(): bool
+update(): bool
+updateWithoutCondition():bool

---

**SyntaxCheckerFacade**

+check(): bool
-simplificationForString()

---

**Parser**

+gotoDatabase()

---

**DatabaseManager**

-fileseparator: String

+getInstance()
+createTable(): bool
-found(): bool
-binarySearch(): bool
-checkOnUpdate(): bool

---

**ImplementUpdateQuery**

-tableName: String
-table: String[ ]
-tableColumns: String[ ]
-tableDatatype: String[ ]
-path1: String
-condition: string[ ]

+insert()
+deleteAll()
+delete()
+update()
+updateAll()
-equals(): bool
-isValidData(): bool
-getNewTable()
-getColumnNumber()
-sort()
-search()

---

**Select**

+selectAll()
+selectAllAndOr()
+selectAllNot()
+selectAllWhere()
+select()
+selectAndOr()
+selectNot()
+selectWhere()

---

**Informations**

-tableName: String
-createTablePropertiesName: String[ ]
-createTablePropertiesDatatype: String[ ]
-databaseName: String
-insertIntoPropertiesName: String[ ]
-insertIntoPropertiesValues: String[ ]
-condition: String
-updatePropertiesName: String[ ]
-updatePropertiesValue: String[ ]
-selectProperties: String[ ]

+getInstance()
+setCreateTableProperties()
+getTablename()
+getCreateTablePropertiesDatatype()
+getInsertIntoPropertiesName()
+getInsertIntoPropertiesValues()
+getUpdatePropertiesName()
+getUpdatePropertiesValue()
+getDatabaseName()
+getSelectProperties()
+getCondition()
+setCondition()
+setTableName()
+setDatabaseName()
+setSelectProperties()
+setUpdateProperties()
+setInsertIntoProperties()
-spliter()
+setALLNuLL()

---

**Print**

+removeColumns()
+print()

---

**Compare**

+twoStrings(): bool

---

**DirectoryHandler**

~fileSeparator: String

+makeFolder()
+searchForDirectory()
+delete()
+containFolder()
+containFile()
+deleteFile(): bool

---

**XML**

-Datatype()
+write()
+Read()
+getColumnsName()
+getDataType()
-readOperation()

---

**Storage**

-databaseName: String
-tableName: String
-dataTypes: String[ ]
columnNames: String[ ]
-content: Object[ ][ ]

+store()
+load()
+validate(): bool

---

**Builder**

-databaseName: String
-tableName: String
-dataTypes: String[ ]
-columnNames: String[ ]
-content: Object[ ][ ]

+tableName()
+databaseName()
+columnNames()
+dataTypes()
+content()
+build()

---

**StringQuery**

+optimization()
-simplificationForString()

---

**XSD**

---

**XsdCreator**

-NS_PREFIX: String

+writeSchema()

---

**<<interface>>**
**XsdValidation**

+createValidationFile()
+isValidTable(): bool

---

**NameTypeElementMaker**

-nsPrefix: String
-doc: Document

+createElement()
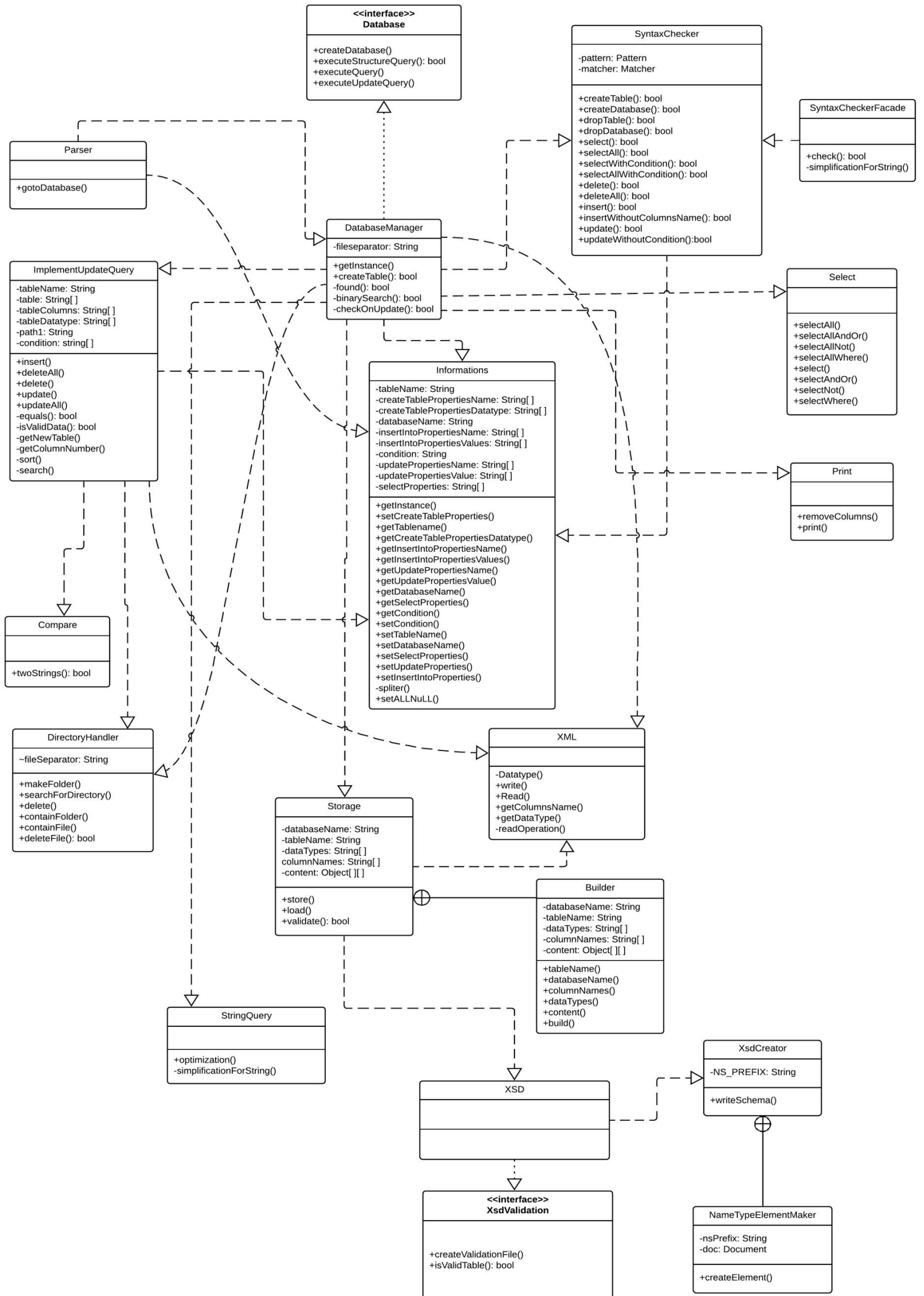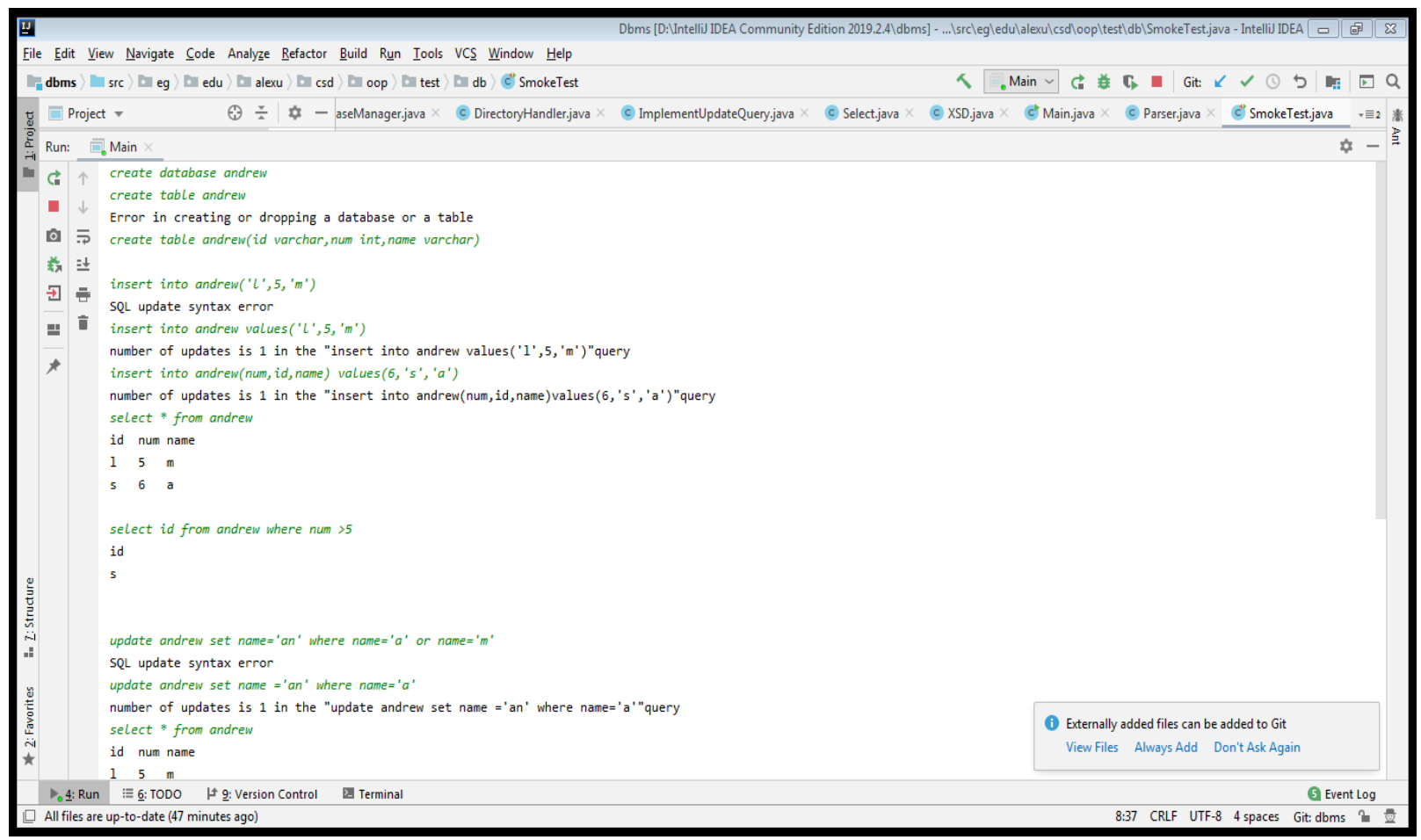
# Design Description

- Parser class receives the SQL query and selects the appropriate operation from the DatabaseManager class .

- StringQuery class is used to optimize the queries and after that we send them to SyntaxChecker class to check for their syntax.

- We use Information class to store all the information of the database and its tables.

- DirectoryHandler Class helps in managing the folders and files of the databases.

- For updating the tables after implementing the queries we use ImplementUpdateQuery class .

- if the query is to select from a table we goes to the Select class which helps to implement the select query either it has a condition or not.

- Finally we print the required part of the table using the print class.

# User Guide

- First you create the database by statement "create database <database name>".
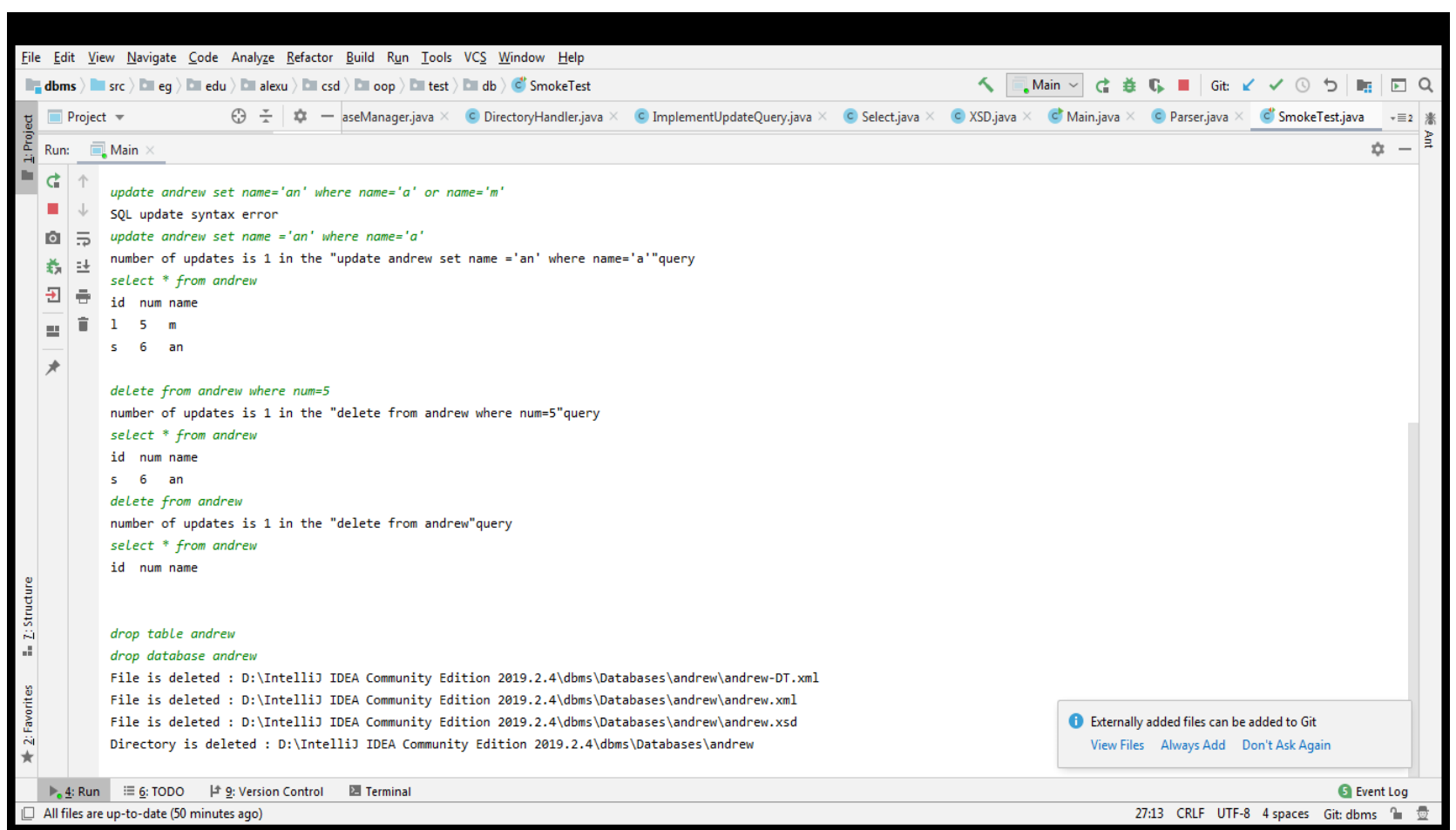
- Then you create a table by statement "create table <table name>".

- You can insert data to the table using the keyword "insert" .

- After that you can print the data by "select" keyword and you can use "where"

  keyword if there is one or more conditions.

- you can also delete from the table by "delete" keyword .

- you may change anything in the table using "update" keyword so that you can

  update your table.

- Whenever you want to drop a table or a database you will use "drop" keyword.

- Your entries must obey the syntax of SQL, otherwise you will be given an error.

# Sample Runs



```
create database andrew
create table andrew
Error in creating or dropping a database or a table
create table andrew(id varchar,num int,name varchar)

insert into andrew('l',5,'m')
SQL update syntax error
insert into andrew values('l',5,'m')
number of updates is 1 in the "insert into andrew values('l',5,'m')"query
insert into andrew(num,id,name) values(6,'s','a')
number of updates is 1 in the "insert into andrew(num,id,name)values(6,'s','a')"query
select * from andrew
id   num name
l   5   m
s   6   a


select id from andrew where num >5
id
s



update andrew set name='an' where name='a' or name='m'
SQL update syntax error
update andrew set name ='an' where name='a'
number of updates is 1 in the "update andrew set name ='an' where name='a'"query
select * from andrew
id   num name
l   5   m
```



```
update andrew set name='an' where name='a' or name='m'
SQL update syntax error
update andrew set name ='an' where name='a'
number of updates is 1 in the "update andrew set name ='an' where name='a'"query
select * from andrew
id   num name
l   5   m
s   6   an

delete from andrew where num=5
number of updates is 1 in the "delete from andrew where num=5"query
select * from andrew
id   num name
s   6   an
delete from andrew
number of updates is 1 in the "delete from andrew"query
select * from andrew
id   num name



drop table andrew
drop database andrew
File is deleted : D:\IntelliJ IDEA Community Edition 2019.2.4\dbms\Databases\andrew\andrew-DT.xml
File is deleted : D:\IntelliJ IDEA Community Edition 2019.2.4\dbms\Databases\andrew\andrew.xml
File is deleted : D:\IntelliJ IDEA Community Edition 2019.2.4\dbms\Databases\andrew\andrew.xsd
Directory is deleted : D:\IntelliJ IDEA Community Edition 2019.2.4\dbms\Databases\andrew
```

## Notes:

- We used the following design patterns:

  - Singleton design pattern in Information and DatabaseManager classes.

  - Façade design pattern in SyntaxCheckerFacade class.

  - Factory design pattern in Parser class.

  - Builder design pattern in Storage class.