# A: Inštalácia a nastavenie

1. Nainštalujte ovládač lokálne:

V termináli alebo príkazovom riadku prejdite do adresára projektu.

Spustite nasledujúci príkaz na inštaláciu ovládača pomocou nástroja Maven:

*mvn clean install*

Tento príkaz zostaví ovládač a nainštaluje ho do vášho lokálneho úložiska Maven.

1. Zahrňte závislosť ovládača:

Otvorte súbor pom.xml svojho projektu.

Do sekcie <dependencies> pridajte nasledujúcu závislosť:

<dependency>  
 <groupId>com.github.jfsql</groupId>  
 <artifactId>JFSQL-driver</artifactId>  
 <version>1.0</version>  
</dependency>

1. Reťazec pripojenia pre ovládač JFSQL JDBC sa riadi štandardným formátom JDBC. Má nasledujúcu štruktúru:

jdbc:jfsql:C:path/to/myDatabase

kde myDatabase je priečinok, ktorý bude slúžiť ako databáza.

# B: Gramatika SQL parsera

grammar JFSQL;  
  
root  
 : statement ( SCOL )? EOF  
 ;  
  
statement  
 : alterTable  
 | createTable  
 | delete  
 | dropTable  
 | insert  
 | select  
 | update  
 ;  
  
alterTable  
 : ALTER TABLE tableName ( renameTable | renameColumn | addColumn | dropColumn )  
 ;  
  
renameTable  
 : RENAME TO tableName  
 ;  
  
renameColumn  
 : RENAME COLUMN columnName TO columnName  
 ;  
  
addColumn  
 : ADD COLUMN columnDefinition  
 ;  
  
dropColumn  
 : DROP COLUMN columnName  
 ;  
  
createTable  
 : CREATE TABLE ( ifNotExists )? tableName OPEN\_PAR columnDefinition ( COL columnDefinition )\* CLOSE\_PAR  
 ;  
  
delete  
 : DELETE FROM tableName ( WHERE expr )?  
 ;  
  
dropTable  
 : DROP TABLE ( ifExists )? tableName  
 ;  
  
insert  
 : INSERT INTO tableName ( OPEN\_PAR columnName ( COL columnName )\* CLOSE\_PAR )? VALUES valuesInParentheses ( COL valuesInParentheses )\*  
 ;  
  
select  
 : SELECT columnName ( COL columnName )\* FROM tableName ( joinOperation )\* ( WHERE expr )? ( limit )?  
 ;  
  
limit  
 : LIMIT numericValue  
 | LIMIT numericValue offset  
 ;  
  
offset  
 : OFFSET numericValue  
 ;  
  
numericValue  
 : NUMERIC\_LITERAL  
 ;  
  
joinOperation  
 : innerJoin  
 | leftJoin  
 ;  
  
innerJoin  
 : ( INNER )? JOIN tableName ON tableDotColumnName EQ tableDotColumnName  
 ;  
  
leftJoin  
 : LEFT ( OUTER )? JOIN tableName ON tableDotColumnName EQ tableDotColumnName  
 ;  
  
update  
 : UPDATE tableName SET columnName EQ value ( COL columnName EQ value )\* ( WHERE expr )?  
 ;  
  
columnDefinition  
 : columnName columnType ( notNull )?  
 ;  
  
notNull  
 : NOT NULL;  
  
ifExists  
 : IF EXISTS  
 ;  
  
ifNotExists  
 : IF NOT EXISTS  
 ;  
  
columnType  
 : INTEGER  
 | REAL  
 | TEXT  
 | BLOB  
 ;  
  
expr  
 : columnName symbol value  
 | columnName symbol value ( binaryOperator columnName symbol value )\*  
 ;  
  
binaryOperator  
 : AND  
 | OR  
 ;  
  
symbol  
 : EQ  
 | NOT\_EQ  
 | LT  
 | LT\_EQ  
 | GT  
 | GT\_EQ  
 | LIKE  
 ;  
  
value  
 : NUMERIC\_LITERAL  
 | STRING\_LITERAL  
 | QUESTION\_MARK  
 | 'default'  
 ;  
  
valuesInParentheses  
 : OPEN\_PAR value ( COL value )\* CLOSE\_PAR  
 ;  
  
tableName  
 : IDENTIFIER  
 ;  
  
tableDotColumnName  
 : IDENTIFIER DOT IDENTIFIER  
 ;  
  
columnName  
 : IDENTIFIER  
 | IDENTIFIER DOT IDENTIFIER  
 ;  
  
COL : ',';  
SCOL : ';';  
DOT : '.';  
OPEN\_PAR : '(';  
CLOSE\_PAR : ')';  
LT : '<';  
LT\_EQ : '<=';  
GT : '>';  
GT\_EQ : '>=';  
EQ : '=';  
NOT\_EQ : '!=';  
QUESTION\_MARK : '?';  
  
NOT : N O T;  
NULL : N U L L;  
IF : I F;  
EXISTS : E X I S T S;  
LIKE : L I K E;  
LIMIT : L I M I T;  
OFFSET : O F F S E T;  
AND : A N D;  
OR: O R;  
ADD : A D D;  
ALTER : A L T E R;  
CREATE : C R E A T E;  
COLUMN : C O L U M N;  
DELETE : D E L E T E;  
DATABASE: D A T A B A S E;  
DROP : D R O P;  
FROM : F R O M;  
INSERT : I N S E R T;  
INTO : I N T O;  
SELECT : S E L E C T;  
SET : S E T;  
TABLE : T A B L E;  
UPDATE : U P D A T E;  
VALUES : V A L U E S;  
WHERE : W H E R E;  
RENAME : R E N A M E;  
TO : T O;  
ON : O N;  
LEFT : L E F T;  
JOIN : J O I N;  
INNER : I N N E R;  
OUTER : O U T E R;  
INTEGER : I N T E G E R;  
REAL : R E A L;  
TEXT : T E X T;  
BLOB : B L O B;  
  
IDENTIFIER  
 : '"' ~'"'\* '"'  
 | '[' ~']'\* ']'  
 | [a-zA-Z\_] [a-zA-Z\_0-9]\*  
 | '\*'  
 ;  
  
NUMERIC\_LITERAL  
 : DIGIT+ ( '.' DIGIT\* )? ( E [-+]? DIGIT+ )?  
 | '.' DIGIT+ ( E [-+]? DIGIT+ )?  
 ;  
  
STRING\_LITERAL  
 : '\'' ( ~'\'' | '\'\'' )\* '\''  
 ;  
  
SPACES  
 : [ \u000B\t\r\n] -> channel(HIDDEN)  
 ;  
  
fragment DIGIT : [0-9];  
fragment A : [aA];  
fragment B : [bB];  
fragment C : [cC];  
fragment D : [dD];  
fragment E : [eE];  
fragment F : [fF];  
fragment G : [gG];  
fragment H : [hH];  
fragment I : [iI];  
fragment J : [jJ];  
fragment K : [kK];  
fragment L : [lL];  
fragment M : [mM];  
fragment N : [nN];  
fragment O : [oO];  
fragment P : [pP];  
fragment Q : [qQ];  
fragment R : [rR];  
fragment S : [sS];  
fragment T : [tT];  
fragment U : [uU];  
fragment V : [vV];  
fragment W : [wW];  
fragment X : [xX];  
fragment Y : [yY];  
fragment Z : [zZ];

# C: Podporované príkazy a ich formáty

* ALTER TABLE podporuje premenovanie tabuľky a stĺpca, pridanie nového stĺpca do tabuľky a odstránenie existujúceho stĺpca. Podporované formáty sú:

ALTER TABLE myTable RENAME TO myTableEdited;

ALTER TABLE myTable RENAME COLUMN id TO id\_edited;

ALTER TABLE myTable ADD COLUMN id INTEGER;

ALTER TABLE myTable DROP COLUMN id;

* CREATE TABLE podporuje dátové typy INTEGER, REAL, TEXT a BLOB. Podporovaný je klúčové slovo IF NOT EXISTS Tiež je možné pridávať klúčové slovo NOT NULL po jednotlivých dátových typoch:

CREATE TABLE myTable (id INTEGER NOT NULL, name TEXT, age INTEGER);  
CREATE TABLE IF NOT EXISTS myTable (id INTEGER, name TEXT, age INTEGER);

* DELETE podporuje štandardný formát, v kombinácií s binárnymi operátormi.

DELETE FROM myTable;

DELETE FROM myTable WHERE id > 1;

* DROP TABLE podporuje štandardný formát.

DROP TABLE myTable;  
DROP TABLE IF EXISTS myTable;

* INSERT podporuje nasledujúce formáty:

INSERT INTO myTable (c1, c2, c3) VALUES (1, 2.5, 'abc');  
INSERT INTO myTable (c1, c2, c3) VALUES (1, 2.5, 'abc'), (2, 2.3, 'def');  
INSERT INTO myTable VALUES (1, 2.5, 'def');  
INSERT INTO myTable VALUES (1, 2.5, 'abc'), (2, 2.3, 'def');

* SELECT podporuje LEFT OUTER JOIN a INNER JOIN, v kombinácií s binárnymi operátormi ako aj LIMIT a OFFSET. V príkazoch obsahujúci JOIN, prefixy pre stĺpce, ktoré sa vyskytnú vo viacerých tabuľkách je povinné.

SELECT *\** FROM myTable;

SELECT id FROM myTable;

SELECT c1, c2, c3 FROM myTable;

SELECT c1, c2, c3 FROM myTable WHERE value1 > 1;

SELECT *\** FROM myTable JOIN myTable2 ON myTable.id = myTable2.id;

SELECT *\** FROM myTable INNER JOIN myTable2 ON myTable.id = myTable2.id;

SELECT *\** FROM myTable LEFT JOIN myTable2 ON myTable.id = myTable2.id;

SELECT *\** FROM myTable LEFT OUTER JOIN myTable2 ON myTable.id = myTable2.id;

* UPDATE podporuje nasledujúce formáty, v kombinácií s binárnymi operátormi.

UPDATE myTable SET value1 = 26;

UPDATE myTable SET value1 = 26 WHERE value1 = 1;

UPDATE myTable SET value1 = 26, value2 = 'abc' WHERE value2 = 'def';

* Podporované binárne operátory sú AND a OR, ktoré možno reťaziť, podporované symboly porovnania sú LIKE, <, <=,=,>=,>.

# D: Podporované konfigurácie

Podporované sú nasledovné konfigurácie:

persistence: json alebo xml

transaction.versioning: default alebo jgit

schema.validation: true alebo false

statement.caching: true alebo false

Ak používateľ zadá niektorú hodnotu v nesprávnom tvare, bude sa používať predvolená hodnota daného atribútu.

# E: Program na porovnávanie serializácie a deserializácie

# F: Program na testovanie maximálneho počtu súčasne otvorených súborov