Atitit mysql db 根据sql得到应该使用的索引强制索引

**public** **static** **void** main(String[] **args**) **throws** Exception {

*iniDBConnPool*();

String sql = "SELECT SQL\_NO\_CACHE (money) FROM archive\_log\_user\_coin\_20210107 \r\n" + "WHERE\r\n"

+ " create\_time >= '2021-01-01 00:00:00' \r\n"

+ " AND create\_time < '2021-01-11 00:00:00' AND company\_short\_name = '11086' AND money>0";

***log***.info(DbUtil.*getIndex*(sql, *dataSource*)); ;

**public** **class** DbUtil {

**public** **static** **final** Log ***log*** = LogFactory.*getLog*(DbUtil.**class**);

**public** **static** **void** main(String[] **args**) {

}

**public** **static** String getIndex(String **sql**,DataSource **ds**) {

List<String> li\_cols=*getCols*( **sql**);

String cols=*getInParams*(li\_cols);

String sql2="SELECT INDEX\_NAME,count(\*) ikldCols FROM INFORMATION\_SCHEMA.`STATISTICS` a WHERE a.table\_name = '"+*getTable*(**sql**)+"' \r\n"

+ "and INDEX\_NAME!='PRIMARY' and COLUMN\_NAME in("+cols+")\r\n"

+ "GROUP BY index\_name HAVING ikldCols="+li\_cols.size();

***log***.info(sql2);

JdbcTemplate jdbcTemplate =**new** JdbcTemplate(**ds**);

Map<String, Object> m= jdbcTemplate.queryForMap(sql2) ;

**return** m.get("INDEX\_NAME").toString();

}

**private** **static** String getIndex(List<String> **li\_cols**, String **table**,DataSource **ds**) {

String cols=*getInParams*(**li\_cols**);

String sql="SELECT INDEX\_NAME,count(\*) ikldCols FROM INFORMATION\_SCHEMA.`STATISTICS` a WHERE a.table\_name = '"+**table**+"' \r\n"

+ "and INDEX\_NAME!='PRIMARY' and COLUMN\_NAME in("+cols+")\r\n"

+ "GROUP BY index\_name HAVING ikldCols="+**li\_cols**.size();

JdbcTemplate jdbcTemplate =**new** JdbcTemplate(**ds**);

Map<String, Object> m= jdbcTemplate.queryForMap(sql) ;

**return** m.get("INDEX\_NAME").toString();

}

**private** **static** String getInParams(List<String> **li**) {

List<String> li2=Lists.*newArrayList*();

**for** (String string : **li**) {

li2.add("'"+string+"'");

}

**return** Joiner.*on*(",").join(li2);

}

**private** **static** String getTable(String **sql**) {

List<String> li=Lists.*newArrayList*();

String dbType = JdbcConstants.***MYSQL***;

//格式化输出

String result = SQLUtils.*format*(**sql**, dbType);

// System.out.println(result); // 缺省大写格式

List<SQLStatement> stmtList = SQLUtils.*parseStatements*(**sql**, dbType);

//解析出的独立语句的个数

// System.out.println("size is:" + stmtList.size());

// for (int i = 0; i < stmtList.size(); i++) {

SQLStatement stmt = stmtList.get(0);

MySqlSchemaStatVisitor visitor = **new** MySqlSchemaStatVisitor();

stmt.accept(visitor);

//获取表名称

**return** visitor.getCurrentTable();

//获取操作方法名称,依赖于表名称

// System.out.println("Manipulation : " + visitor.getTables());

//获取字段名称

// System.out.println("fields : " + visitor.getColumns());

}

**private** **static** List<String> getCols(String **sql**) {

List<String> li=Lists.*newArrayList*();

String dbType = JdbcConstants.***MYSQL***;

//格式化输出

String result = SQLUtils.*format*(**sql**, dbType);

// System.out.println(result); // 缺省大写格式

List<SQLStatement> stmtList = SQLUtils.*parseStatements*(**sql**, dbType);

//解析出的独立语句的个数

// System.out.println("size is:" + stmtList.size());

// for (int i = 0; i < stmtList.size(); i++) {

SQLStatement stmt = stmtList.get(0);

MySqlSchemaStatVisitor visitor = **new** MySqlSchemaStatVisitor();

stmt.accept(visitor);

//获取表名称

// System.out.println("Tables : " + visitor.getCurrentTable());

//获取操作方法名称,依赖于表名称

// System.out.println("Manipulation : " + visitor.getTables());

//获取字段名称

// System.out.println("fields : " + visitor.getColumns());

List<Condition> conditions = visitor.getConditions();

**for** (Condition condition : conditions) {

String colName = condition.getColumn().getName();

// System.out.println(colName);

//mustl de mlt,,,bcz cdt only add same col,,,,esp time range..so liag time fld..

**if**(!li.contains(colName))

li.add(colName);

}

**return** li;

}

}