```python
import sqlite3
from threading import Lock

class BDHandler():
    """
    Classe para a manipulação do banco de dados
    """
    def __init__(self,dbpath,tags,tablename='dataTable'):
        """
        Construtor
        """
        self._dbpath = dbpath
        self._tablename = tablename
        self._con = sqlite3.connect(self._dbpath,check_same_thread=False)
        self._cursor = self._con.cursor()
        self._col_names = tags.keys()
        self._lock = Lock()
        self.createTable()

    def __del__(self):
        self._con.close()

    def createTable(self):
        """
        Método que cria a tabela para armazenamento dos dados caso ela não exista
        """
        try:
            sql_str = f"""
            CREATE TABLE IF NOT EXISTS {self._tablename} (
                id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
                timestamp TEXT NOT NULL,
                """
            for n in self._col_names:
                sql_str += f'{n} REAL,'

            sql_str = sql_str[:-1]
            sql_str += ');'
            self._lock.acquire()
            self._cursor.execute(sql_str)
            self._con.commit()
            self._lock.release()
        except Exception as e:
            print("Erro: ",e.args)

    def insertData(self, data):
        """
        Método para inserção dos dados no BD
        """
        try:
            self._lock.acquire()
            timestamp = str(data['timestamp'])
            str_cols = 'timestamp,' + ','.join(data['values'].keys())
            str_values = f"'{timestamp}'," + ','.join([str(data['values'][k]) for k
in data['values'].keys()])
            sql_str = f'INSERT INTO {self._tablename} ({str_cols}) VALUES
({str_values});'
```

```
            self._cursor.execute(sql_str)
            self._con.commit()
        except Exception as e:
            print("Erro: ",e.args)
        finally:
            self._lock.release()



    def selectData(self, cols, init_t, final_t):
        """
        Método que realiza a busca no BD entre 2 horários especificados
        """
        try:
            self._lock.acquire()
            sql_str = f"SELECT {','.join(cols)} FROM {self._tablename} WHERE
timestamp BETWEEN '{init_t}' AND '{final_t}'"
            self._cursor.execute(sql_str)
            dados = dict((sensor,[]) for sensor in cols)
            for linha in self._cursor.fetchall():
                for d in range(0,len(linha)):
                    dados[cols[d]].append(linha[d])
            return dados
        except Exception as e:
            print("Erro: ",e.args)
        finally:
            self._lock.release()


```