SQLTypeproviders - Specifikace

Description

Provides jupyter notebook with dynamicly generated types (other enviroments may benefit too but only staticly generated hits will be available e.g. running it second time) to enhance database response with types. Changing simple Tuple[Any] to an object with fields of defined types if possible (see SupportedTypes).
This will have impact on performance and is primarily intended for jupyter-notebook-runtime(IPython).
The dot notation accesibility leads to both comfort of the programmer and robustness of the code as it enables static type checking.

Libraries

jupyter-notebook - jupyter-notebook UI
psycopg2 - possgress connection
(mysql connector)
sqlparse - getting table (or pseudotable) from SQL select

Supported databases

- Posgress
- (planing mysql)

Usage example

PYTHON

```
from TypeProvider import SqlTypeProvider

connection = /* server.connect(...) */
cursor = connection.cursor()

cmd = """
SELECT * FROM employees;
"""
providedType = SqlTypeProvider.Provide(connection, cmd)
cursor.execute(cmd)

# ensure .fetch_one() returns value
response_typed = providedType(*cursor.fetchone())
# or
responses_typed =
        [ providedType(*val) for val in cursor.fetchall() ]

cursor.close()
connection.close()
```

Supported types

reference

- Char / Varchar -> string
- Bool / Boolean -> bool
- SmallInt / MediumInt / Int / Inteeger -> int
- Float / Double / Decimal / Dec -> float
- Date -> DateTime.Date

If database type is not mention above, the field will still be accesible via dot notation, but type of that field will be not specified (e.g. Any) at its acctual type will be decided by the database-connection library (for example psycopg2).

Generating types

The typeprovider will generate __pychache__/_autogenerated_clases.py, where all the classes will be. The classes will contain a constructor taking *n* arguments, where *n* is the number of datafields in the new type. It will also support printing in form of

```txt
name_of_field1 = <value of field1>
name_of_field2 = <value of field2>
name_of_field3 = <value of field3>
name_of_field4 = <value of field4>
...
```

Accepting SQL

All SQL statments will be accepted if both Oracle SQL and your target database accept it. In other words.
Database specific features will not work like '\dt' for postgress.
And your statment must be complient with the database additional restrictions (postgress requires querries to end with ';')