sqlalchemy-ibmi

May 19, 2020

Contents

CHAPTER 1

DBAPI Connection

This dialect uses the pyODBC DBAPI and the IBM i Access ODBC Driver.

Connection string:

```
engine = create_engine("ibmi://user:password@host:port/rdbname[?key=value&key=value...
↪]")
```

CHAPTER 2

Connection Arguments

The sqlalchemy-ibmi dialect supports multiple connection arguments that are passed in the URL to the create_engine function.

- `autocommit` - If `False`, Connection.commit must be called; otherwise each statement is automatically committed. Defaults to `False`.
- `readonly` - If `True`, the connection is set to read-only. Defaults to `False`.
- `timeout` - The login timeout for the connection, in seconds.

CHAPTER 3

Transaction Isolation Level / Autocommit

Db2 for i supports 5 isolation levels:

- `SERIALIZABLE`: `*RR`
- `READ COMMITTED`: `*CS`
- `READ UNCOMMITTED`: `*CHG`
- `REPEATABLE READ`: `*ALL`
- `NO COMMIT`: `*NC`

At this time, sqlalchemy-ibmi supports all of these isolation levels except NO COMMIT.

Autocommit is supported on all available isolation levels.

To set isolation level globally:

```
engine = create_engine("ibmi://user:pass@host/", isolation_level='REPEATABLE_READ')
```

To set using per-connection execution options:

```
connection = engine.connect()
connection = connection.execution_options(
    isolation_level="SERIALIZABLE"
)
```

CHAPTER 4

Table Creation String Size

When creating a table with SQLAlchemy, Db2 for i requires that the size of a String column be provided.

Provide the length for a String column as follows:

```
class User(Base):
    __tablename__ = 'users'
    id = Column(Integer, Sequence('user_id_seq'), primary_key=True)
    name = Column(String(50))

users = Table('users', metadata,
    Column('id', Integer, Sequence('user_id_seq'), primary_key=True),
    Column('name', String(50)),
)
```

CHAPTER 5

Query Function Strings

Db2 for i doesn't support parameter markers in the SELECT clause of a statement. As a result, the following command will not work with sqlalchemy-ibmi:

```
session.query(func.count("*")).select_from(User).scalar()
```

Instead, please use the following:

```
session.query(func.count()).select_from(User).scalar()
```

Addtionally, column names cannot be passed as strings, so convert your column strings to literal columns as follows:

```
from sqlalchemy.sql import literal_column
session.query(func.count(literal_column("colname"))).select_from(User).scalar()
```

CHAPTER 6

Text search support

The ColumnOperators.match function is implemented using a basic LIKE operation by default. However, when OmniFind Text Search Server for Db2 for i is installed, match will take advantage of the CONTAINS function that it provides.

Python Module Index

S

Index

S