

# Pemrograman Desktop 8

Yonathan F. Hendrawan

# Database

- Materi sebelumnya selalu menempatkan data di memory computer
  - Jika komputer dimatikan, data jadi hilang
- Database dapat menyimpan data secara lebih permanen
- Di materi ini akan digunakan Database SQLite
  - Chinook\_Sqlite.sqlite dari <https://github.com/lerocha/chinook-database>

# Program 1 - QSqlTableModel

```
import os
import sys

from PyQt6.QtCore import QSize, Qt
from PyQt6.QtSql import QSqlDatabase, QSqlTableModel
from PyQt6.QtWidgets import QApplication, QMainWindow, QTableView

basedir = os.path.dirname(__file__)
db = QSqlDatabase("QSQLITE")
db.setDatabaseName(os.path.join(basedir, "Chinook_Sqlite.sqlite"))
db.open()

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.table = QTableView()
        self.model = QSqlTableModel(db=db)
        self.table.setModel(self.model)
        self.model.setTable("Track")
        self.model.select()
        self.setMinimumSize(QSize(1024, 600))
        self.setCentralWidget(self.table)

app = QApplication(sys.argv)
window = MainWindow()
window.show()
app.exec()
```

# Program 1 - QSqlTableModel

- Database pada QTableView dapat diedit
  - double-click pada cell untuk mengubah nilainya
- Perubahan disimpan kembali ke database setelah editing selesai
- Sorting:

```
self.model.setTable("Track")  
self.model.setSort(2, Qt.SortOrder.DescendingOrder)  
self.model.select()
```
- 2 pilihan: Qt.SortOrder.AscendingOrder atau Qt.SortOrder.DescendingOrder
- Alternatif:

```
self.model.setTable("Track")  
idx = self.model.fieldIndex("Milliseconds")  
self.model.setSort(idx, Qt.SortOrder.DescendingOrder)  
self.model.select()
```

# Program 1 - QSqlTableModel

- Mengubah judul kolom:

```
self.model.setTable("Track")
```

```
self.model.setHeaderData(2, Qt.Orientation.Horizontal, "Album (ID)")
```

```
self.model.select()
```

# Program 1 – QSqlTableModel: Filtering

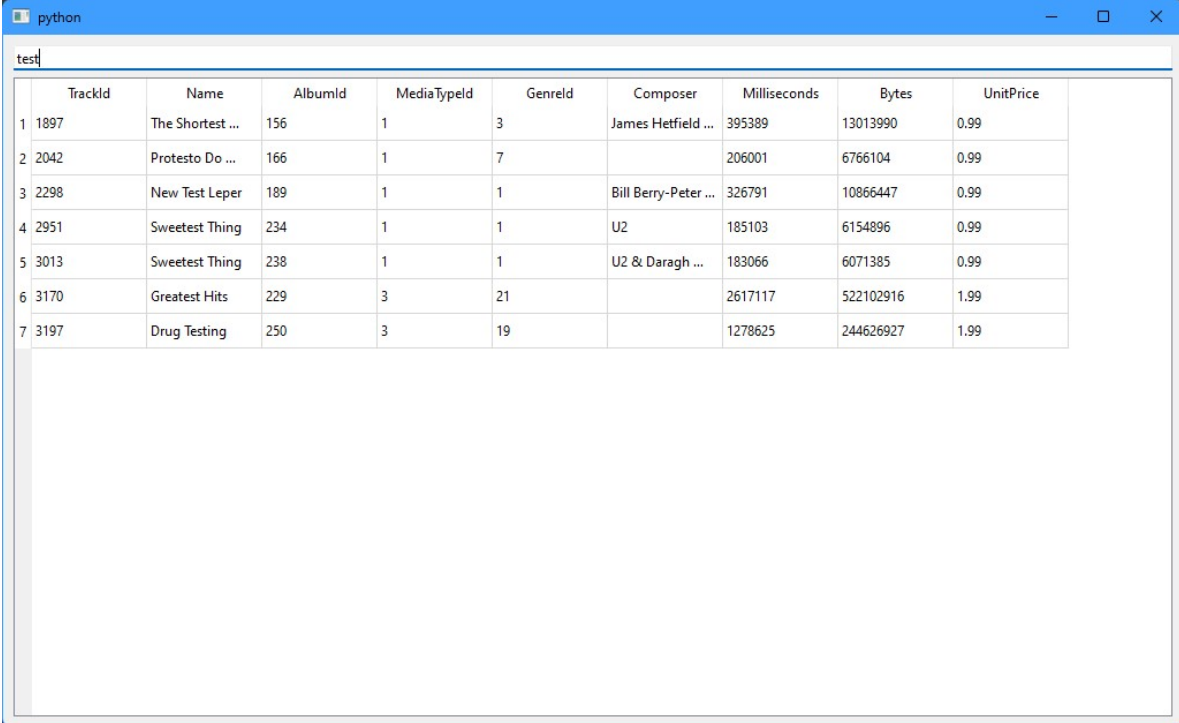
```
class MainWindow(QMainWindow):  
    def __init__(self):  
        super().__init__()   
        container = QWidget()  
        layout = QVBoxLayout()  
        self.search = QLineEdit()  
        self.search.textChanged.connect(self.update_filter)  
        self.table = QTableView()  
        layout.addWidget(self.search)  
        layout.addWidget(self.table)  
        container.setLayout(layout)  
  
        self.model = QSqlTableModel(db=db)  
        self.table.setModel(self.model)  
        self.model.setTable("Track")  
        self.model.select()  
        self.setMinimumSize(QSize(1024, 600))  
        self.setCentralWidget(container)  
  
    def update_filter(self, s):  
        filter_str = 'Name LIKE "%{}%"'.format(s)  
        self.model.setFilter(filter_str)
```

# Program 1 – QSqlTableModel: Filtering

```
class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        container = QWidget()
        layout = QVBoxLayout()
        self.search = QLineEdit()
        self.search.textChanged.connect(self.update_filter)
        self.table = QTableView()
        layout.addWidget(self.search)
        layout.addWidget(self.table)
        container.setLayout(layout)

        self.model = QSqlTableModel(db=db)
        self.table.setModel(self.model)
        self.model.setTable("Track")
        self.model.select()
        self.setMinimumSize(QSize(1024, 600))
        self.setCentralWidget(container)

    def update_filter(self, s):
        filter_str = 'Name LIKE "%{}%".format(s)
        self.model.setFilter(filter_str)
```



	TrackId	Name	AlbumId	MediaTypeId	GenreId	Composer	Milliseconds	Bytes	UnitPrice
1	1897	The Shortest ...	156	1	3	James Hetfield ...	395389	13013990	0.99
2	2042	Protesto Do ...	166	1	7		206001	6766104	0.99
3	2298	New Test Leper	189	1	1	Bill Berry-Peter ...	326791	10866447	0.99
4	2951	Sweetest Thing	234	1	1	U2	185103	6154896	0.99
5	3013	Sweetest Thing	238	1	1	U2 & Daragh ...	183066	6071385	0.99
6	3170	Greatest Hits	229	3	21		2617117	522102916	1.99
7	3197	Drug Testing	250	3	19		1278625	244626927	1.99

# Program 1 – QSqlTableModel: Filtering

- Program sebelumnya rentan SQL Injection
  - Masukkan “, program jadi hang.
  - Karena filter statement jadi: 'name LIKE "%"%"'
- Gunakan regex untuk menghilangkan input non-alphanumeric atau space:

```
def update_filter(self, s):  
    s = re.sub("[\W_]+", "", s)  
    filter_str = 'Name LIKE "%{}%"'.format(s)  
    self.model.setFilter(filter_str)
```