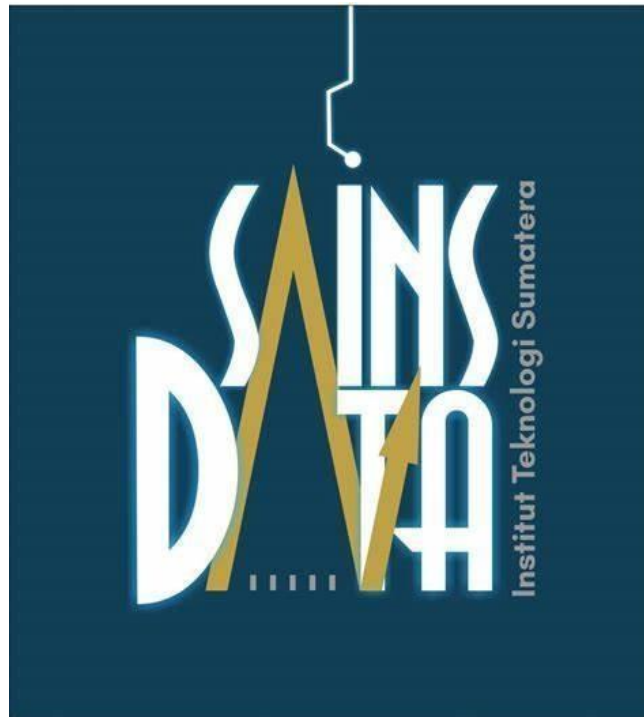


Modul Praktikum TBD

Modul I

[XML dan SQLite]



Program Studi Sains Data

Fakultas Sains

INSTITUT TEKNOLOGI SUMATERA

Tahun Ajaran Genap 2023/2024

I. Tujuan dan Indikator praktikum

- a. Memahami bagaimana pembuatan basis data XML
- b. Memahami penggunaan XPath dan XQuery
- c. Memahami penggunaan XML dengan pemrograman python
- d. Memahami bagaimana menggunakan SQLite untuk membuat basis data.

II. Teori Dasar

a. XML

Pengertian XML

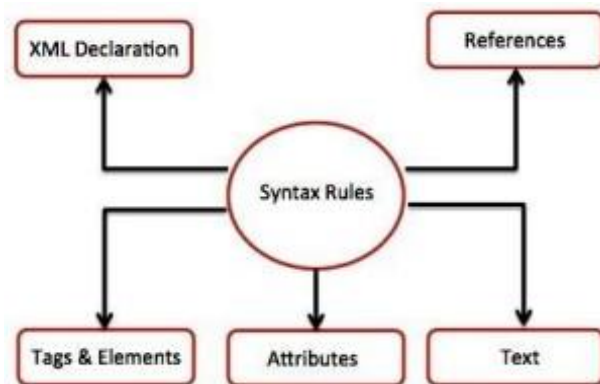
ada beberapa hal yang dapat anda ketahui terkait dengan XML, sebagai berikut:

- Extensible Markup Language (XML) adalah merupakan markup berbasis teks bahasa yang berasal dari Standard Generalized Markup Language (SGML). Meskipun XML berasal dari SGML, XML telah menyederhanakan prosesnya
- mendefinisikan dan menggunakan metadata ini. dengan cepat menjadi standar untuk data
- XML adalah pertukaran data dalam aplikasi Internet generasi berikutnya. XML memungkinkan tag yang ditentukan pengguna yang membuat penanganan dokumen XML lebih fleksibel.
- XML menyimpan data dalam format teks biasa. Hal ini menyediakan perangkat lunak dan perangkat keras
- perangkat lunak dan perangkat keras untuk menyimpan, mengangkut, dan berbagi data.
- XML adalah sintaks untuk mengekspresikan data terstruktur dalam format teks
- XML bukanlah sebuah bahasa yang berdiri sendiri. Sebaliknya, XML digunakan untuk membangun markup bahasa.
- XML dirancang untuk menyimpan dan mengangkut data, tetapi tidak untuk menampilkan data.
- XML dirancang agar dapat dibaca oleh manusia dan mesin.
- XML adalah bahasa markup seperti HTML. Namun, ia bukanlah pengganti HTML
- XML dirancang untuk menjadi deskriptif sendiri
- XML tidak bergantung pada platform dan bahasa
- XML adalah Rekomendasi W3C
- XML pada awalnya disebut Web SGML, kemudian berganti nama menjadi
- Extensible Markup Language (XML)

Struktur dokumen XML

- The XML declaration
- The Document Type Declaration(DOCTYPE)
 - Used to define Schema / DTD definition / user-defined entity reference
- Entity reference
- The element data
- The attribute data
- The character data or XML content

- PCDATA
- CDATA
- Comments



Gambar. 1. Aturan syntax XML.

b. SQLite

pengertian SQLite:

- SQLite adalah paket perangkat lunak domain publik yang menyediakan sistem manajemen basis data relasional, atau RDBMS.
- relational database management system (RDBMS) adalah digunakan untuk menyimpan catatan yang ditentukan pengguna dalam tabel besar.
- RDBMS Selain penyimpanan dan manajemen data, mesin basis data dapat memproses perintah kueri kompleks yang menggabungkan data dari beberapa tabel untuk menghasilkan laporan dan ringkasan data.
- RDBMS populer lainnya
- populer lainnya termasuk Oracle Database, IBM DB2, dan Microsoft SQL Server di sisi komersial, dengan MySQL dan PostgreSQL sebagai produk sumber terbuka yang populer.
- berikut perbandingan antara tradisional RDBMS dan SQLite:

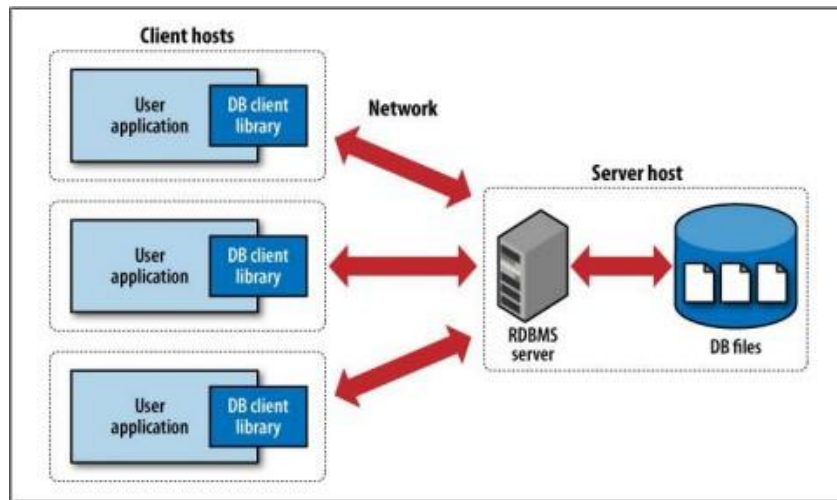


Figure 1-1. Traditional RDBMS client/server architecture that utilizes a client library.

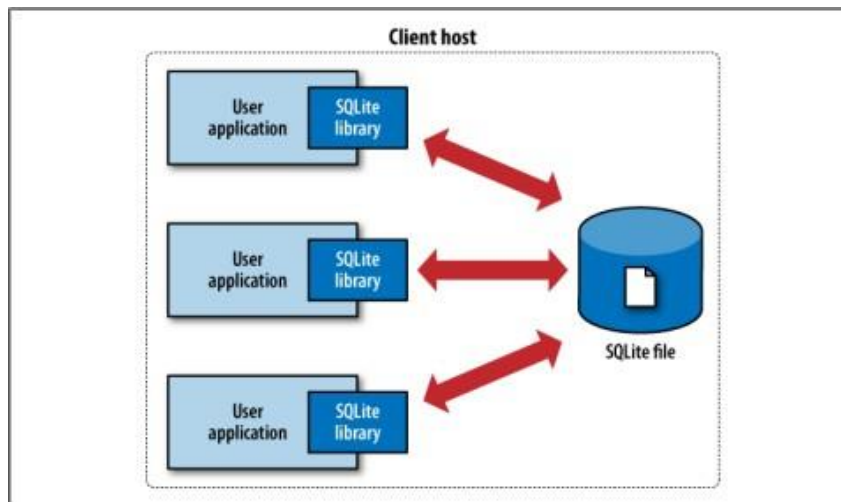


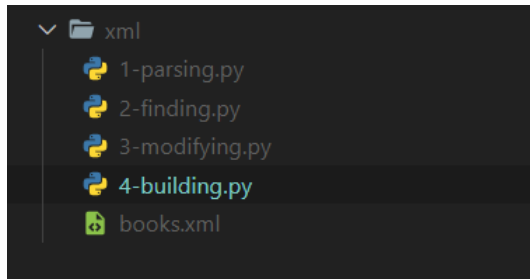
Figure 1-2. The SQLite server-less architecture.

III. Instruktur Praktikum

a. XML

1. Persiapan

- Siapkan struktur folder seperti berikut



- Siapkan data books.xml

```
xml > books.xml > ...
1  <books>
2      <book id="1">
3          <title>The Catcher in the Rye</title>
4          <author>J.D. Salinger</author>
5          <genre>Fiction</genre>
6          <price>20.00</price>
7      </book>
8      <book id="2">
9          <title>To Kill a Mockingbird</title>
10         <author>Harper Lee</author>
11         <genre>Fiction</genre>
12         <price>18.00</price>
13     </book>
14     <book id="3">
15         <title>1984</title>
16         <author>George Orwell</author>
17         <genre>Science Fiction</genre>
18         <price>15.00</price>
19     </book>
20 </books>
```

2. Parsing

- Buat file 1-parsing.xml

```

xml > 1-parsing.py > ...
1  import xml.etree.ElementTree as ET
2
3  # Membaca file XML
4  tree = ET.parse('xml/books.xml')
5  root = tree.getroot()
6
7  # Menampilkan informasi tentang setiap buku
8  for book in root.findall('book'):
9      # Mengakses informasi buku
10     title = book.find('title').text
11     author = book.find('author').text
12     genre = book.find('genre').text
13     price = book.find('price').text
14
15     # Menampilkan informasi buku
16     print(f"Title: {title}")
17     print(f"Author: {author}")
18     print(f"Genre: {genre}")
19     print(f"Price: {price}")
20     print()
21
22

```

Dalam contoh ini, kita membaca file XML "books.xml" menggunakan ET.parse(), lalu mendapatkan elemen root menggunakan tree.getroot(). Kemudian, kita melakukan iterasi melalui setiap elemen "book" dalam root menggunakan root.findall('book'). Untuk setiap elemen "book", kita menggunakan metode .find() untuk mengakses informasi seperti judul, penulis, genre, dan harga buku, dan kemudian menampilkannya.

- Output(terminal)

```
Title: The Catcher in the Rye  
Author: J.D. Salinger  
Genre: Fiction  
Price: 20.00
```

```
Title: To Kill a Mockingbird  
Author: Harper Lee  
Genre: Fiction  
Price: 18.00
```

```
Title: 1984  
Author: George Orwell  
Genre: Science Fiction  
Price: 15.00
```

```
PS D:\TBD Modul 1>
```

3. Finding

- Buat file 2-finding.xml

```

xml > 2-finding.py > ...
1  import xml.etree.ElementTree as ET
2
3  # Membaca file XML
4  tree = ET.parse('xml/books.xml')
5  root = tree.getroot()
6
7  # Mencari semua buku dengan genre "Fiction"
8  fiction_books = root.findall("./book[genre='Fiction']")
9
10 # Menampilkan informasi buku
11 for book in fiction_books:
12     title = book.find('title').text
13     author = book.find('author').text
14     print(f>Title: {title}, Author: {author}")
15
16
17 # Mencari buku dengan judul "1984"
18 book = root.find("./book[title='1984']")
19
20 # Jika buku ditemukan, tampilkan informasi buku
21 if book is not None:
22     title = book.find('title').text
23     author = book.find('author').text
24     print(f>Title: {title}, Author: {author}")
25 else:
26     print("Book not found.")
27
28 # Mencari buku dengan judul "1985" yang tidak ada di dalam books.xml
29 book = root.find("./book[title='1985']")
30
31 # Jika buku ditemukan, tampilkan informasi buku
32 if book is not None:
33     title = book.find('title').text
34     author = book.find('author').text
35     print(f>Title: {title}, Author: {author}")
36 else:
37     print("Book not found.")
38

```

Dalam melakukan pencarian elemen atau informasi tertentu dalam file XML menggunakan modul `xml.etree.ElementTree`, Anda dapat menggunakan beberapa metode seperti `find()`, `findall()`, atau `iter()`.

Dalam contoh di atas, kita menggunakan ekspresi XPath untuk mencari informasi tertentu dalam file XML "books.xml". Setelah menemukan elemen yang sesuai, kita mengekstrak informasi yang diperlukan, seperti judul dan penulis, dan menampilkannya.

- Output

Buku dengan genre Fiction(Terminal)

```
Title: The Catcher in the Rye, Author: J.D. Salinger  
Title: To Kill a Mockingbird, Author: Harper Lee  
Title: 1984, Author: George Orwell  
Book not found.  
PS D:\TBD Modul 1>
```

Buku dengan tittle 1984(Terminal)

```
Title: The Catcher in the Rye, Author: J.D. Salinger  
Title: To Kill a Mockingbird, Author: Harper Lee  
Title: 1984, Author: George Orwell  
Book not found.  
PS D:\TBD Modul 1>
```

Buku dengan tittle 1985 tidak ditemukan (Terminal)

```
Title: The Catcher in the Rye, Author: J.D. Salinger  
Title: To Kill a Mockingbird, Author: Harper Lee  
Title: 1984, Author: George Orwell  
Book not found.  
PS D:\TBD Modul 1>
```

4. Modifying

- Buat file 3-modifying.xml

```

xml > 3-modifying.py > ...
1  import xml.etree.ElementTree as ET
2
3  # Membaca file XML
4  tree = ET.parse('xml/books.xml')
5  root = tree.getroot()
6
7  # Membuat elemen untuk buku baru
8  new_book = ET.Element('book')
9  new_book.set('id', '4')
10
11 # Menambahkan sub-elemen untuk buku baru
12 title = ET.SubElement(new_book, 'title')
13 title.text = 'The Great Gatsby'
14
15 author = ET.SubElement(new_book, 'author')
16 author.text = 'F. Scott Fitzgerald'
17
18 genre = ET.SubElement(new_book, 'genre')
19 genre.text = 'Classic'
20
21 price = ET.SubElement(new_book, 'price')
22 price.text = '25.00'
23
24 # Menambahkan buku baru ke dalam root
25 root.append(new_book)
26 ET.indent(tree, space="\t", level=0)
27
28 # Menyimpan perubahan ke dalam file XML
29 tree.write('xml/books.xml')
30
31

```

Dalam contoh ini, kita membuat sebuah elemen baru untuk buku yang akan ditambahkan ke file XML. Kemudian, kita menambahkan sub-elemen untuk judul, penulis, genre, dan harga buku baru ini. Setelah itu, kita menambahkan elemen buku baru ke dalam root elemen dari dokumen XML. Terakhir, kita menyimpan perubahan tersebut ke dalam file XML yang sama dengan nama "books.xml"

- Output

Sebelum modify

```

xml > books.xml > ...
1  <books>
2      <book id="1">
3          <title>The Catcher in the Rye</title>
4          <author>J.D. Salinger</author>
5          <genre>Fiction</genre>
6          <price>20.00</price>
7      </book>
8      <book id="2">
9          <title>To Kill a Mockingbird</title>
10         <author>Harper Lee</author>
11         <genre>Fiction</genre>
12         <price>18.00</price>
13     </book>
14     <book id="3">
15         <title>1984</title>
16         <author>George Orwell</author>
17         <genre>Science Fiction</genre>
18         <price>15.00</price>
19     </book>
20 </books>

```

Sesudah di modify

```

xml > books.xml > ...
1  <books>
2      <book id="1">
3          <title>The Catcher in the Rye</title>
4          <author>J.D. Salinger</author>
5          <genre>Fiction</genre>
6          <price>20.00</price>
7      </book>
8      <book id="2">
9          <title>To Kill a Mockingbird</title>
10         <author>Harper Lee</author>
11         <genre>Fiction</genre>
12         <price>18.00</price>
13     </book>
14     <book id="3">
15         <title>1984</title>
16         <author>George Orwell</author>
17         <genre>Science Fiction</genre>
18         <price>15.00</price>
19     </book>
20     <book id="4">
21         <title>The Great Gatsby</title>
22         <author>F. Scott Fitzgerald</author>
23         <genre>Classic</genre>
24         <price>25.00</price>
25     </book>
26 </books>
27

```

Setelah modify data buku dengan id 4 berhasil ditambahkan

5. Building

- Buat file 4-building.py

Dalam contoh ini, kita membuat sebuah dokumen XML yang berisi daftar buku dengan judul, penulis, genre, dan harga. Setiap buku direpresentasikan sebagai elemen "book" dengan atribut "id". Setelah kita selesai membuat struktur XML dengan menggunakan modul `xml.etree.ElementTree`, kita menyimpannya ke dalam file dengan nama "new-books.xml".

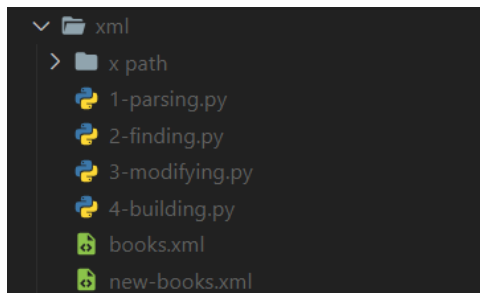
```
xml > 4-building.py > ...
1  import xml.etree.ElementTree as ET
2
3  # Membuat root element
4  root = ET.Element("books")
5
6  # Membuat child elements untuk setiap buku
7  book1 = ET.SubElement(root, "book")
8  book1.set("id", "1")
9
10 title1 = ET.SubElement(book1, "title")
11 title1.text = "The Catcher in the Rye"
12
13 author1 = ET.SubElement(book1, "author")
14 author1.text = "J.D. Salinger"
15
16 genre1 = ET.SubElement(book1, "genre")
17 genre1.text = "Fiction"
18
19 price1 = ET.SubElement(book1, "price")
20 price1.text = "20.00"
21
```

```

21
22     book2 = ET.SubElement(root, "book")
23     book2.set("id", "2")
24
25     title2 = ET.SubElement(book2, "title")
26     title2.text = "To Kill a Mockingbird"
27
28     author2 = ET.SubElement(book2, "author")
29     author2.text = "Harper Lee"
30
31     genre2 = ET.SubElement(book2, "genre")
32     genre2.text = "Fiction"
33
34     price2 = ET.SubElement(book2, "price")
35     price2.text = "18.00"
36
37     # Membuat XML tree dari root element
38     tree = ET.ElementTree(root)
39     ET.indent(tree, space="\t", level=0)
40
41     # Menyimpan XML ke file
42     tree.write("xml/new-books.xml")

```

- Output



File new-books.xml berhasil dibuat, dengan isi data seperti berikut

```

xml > new-books.xml > ...
1  <books>
2      <book id="1">
3          <title>The Catcher in the Rye</title>
4          <author>J.D. Salinger</author>
5          <genre>Fiction</genre>
6          <price>20.00</price>
7      </book>
8      <book id="2">
9          <title>To Kill a Mockingbird</title>
10         <author>Harper Lee</author>
11         <genre>Fiction</genre>
12         <price>18.00</price>
13     </book>
14 </books>

```

6. XPath

XPath (XML Path Language) adalah bahasa untuk menentukan lokasi tertentu di dalam sebuah dokumen XML. Ini mirip dengan konsep dari alamat file dalam sistem berkas komputer. Dengan XPath, Anda dapat menentukan elemen atau himpunan elemen yang ingin Anda temukan atau manipulasi dalam dokumen XML.

Sebelum mulai buat folder baru x path didalam folder xml untuk Latihan x path berikut



Berikut adalah beberapa contoh ekspresi XPath umum:

- Elemen Tunggal:
/ **book**: Mengambil semua elemen "book" yang langsung berada di bawah elemen "books".

Buat file 1-show-all-books.xml

```

xml > x path > 1-show-all-books.py > ...
1  import xml.etree.ElementTree as ET
2
3  # Membaca file XML
4  tree = ET.parse('xml/books.xml')
5  root = tree.getroot()
6
7  # XPath: /books/book
8  books = root.findall('book')
9
10 # Menampilkan judul buku
11 for book in books:
12     # Mengakses informasi buku
13     title = book.find('title').text
14     author = book.find('author').text
15
16     # Menampilkan informasi buku
17     print(f>Title: {title}")
18     print(f"Author: {author}")
19     print()
20

```

Output

```

o Title: The Catcher in the Rye
  Author: J.D. Salinger

  Title: To Kill a Mockingbird
  Author: Harper Lee

  Title: 1984
  Author: George Orwell

  Title: The Great Gatsby
  Author: F. Scott Fitzgerald

PS D:\TBD Modul 1>

```

- Pencarian Mendalam:
/ **book/titles**: Mengambil semua elemen "title" yang berada di dalam elemen "book" yang langsung berada di bawah elemen "books".

Buat file 2-show-all-tittle.py

```

xml > x path > 2-show-all-tittle.py > ...
1  import xml.etree.ElementTree as ET
2
3  # Membaca file XML
4  tree = ET.parse('xml/books.xml')
5  root = tree.getroot()
6
7  # Ekspresi XPath
8  titles = root.findall('book/title')
9
10 # Menampilkan judul buku
11 for title in titles:
12     print(title.text)
13
14

```

Output

```

The Catcher in the Rye
To Kill a Mockingbird
1984
The Great Gatsby
PS D:\TBD Modul 1>

```

- Wildcard (Pencarian Semua Elemen):

//*: Mengambil semua elemen dalam dokumen XML

Buat file 3-show-all-element.py

```

xml > x path > 3-show-all-element.py > ...
1  import xml.etree.ElementTree as ET
2
3  # Membaca file XML
4  tree = ET.parse('xml/books.xml')
5  root = tree.getroot()
6
7  # Mencocokkan semua elemen menggunakan ekspresi XPath //*
8  all_elements = root.findall('.*/*')
9
10 # Menampilkan semua elemen yang cocok
11 for element in all_elements:
12     print(element.tag)
13

```

Output


```

○ book
  title
  author
  genre
  price
book
  title
  author
  genre
  price
book
  title
  author
  genre
  price
book
  title
  author
  genre
  price
PS D:\TBD Modul 1>

```

- Akses Atribut:

/book[@category='fiction']: Mengambil semua elemen "book" yang memiliki atribut "category" dengan nilai "fiction".

Buat file 4-filter-by-genre.py

```

xml > x path > 4-filter-by-genre.py > ...
1  import xml.etree.ElementTree as ET
2
3  # Membaca file XML
4  tree = ET.parse('xml/books.xml')
5  root = tree.getroot()
6
7  # Mencari semua buku dengan genre "Fiction"
8  fiction_books = root.findall("book[genre='Fiction']")
9
10 # Menampilkan informasi buku
11 for book in fiction_books:
12     title = book.find('title').text
13     author = book.find('author').text
14     print(f"Title: {title}, Author: {author}")
15
16

```

Output

```

Title: The Catcher in the Rye, Author: J.D. Salinger
Title: To Kill a Mockingbird, Author: Harper Lee
PS D:\TBD Modul 1>

```

- Pencarian Relatif:

./title: Mengambil elemen "title" yang berada dalam konteks saat ini.

Buat file 5-access-tittle.py

```
xml > x path > 5-access-tittle.py > ...
1  import xml.etree.ElementTree as ET
2
3  # Membaca file XML
4  tree = ET.parse('xml/books.xml')
5  root = tree.getroot()
6
7  # XPath: /books/book
8  books = root.findall('book')
9
10 # Menampilkan judul buku
11 for book in books:
12     # Mengakses tittle buku yang berada dalam konteks saat ini
13     title = book.find('./title').text
14
15     # Menampilkan informasi buku
16     print(f"Title: {title}")
17     print()
18
```

Output

```
Title: The Catcher in the Rye
Title: To Kill a Mockingbird
Title: 1984
Title: The Great Gatsby
```

PS D:\TBD Modul 1>

- Mencocokkan semua element
./title: Mencocokkan semua elemen "title" di dalam dokumen menggunakan tanda.

Buat file 6-show-all-tittle-2.py

```

xml > x path > 6-show-all-tittle-2.py > ...
1  import xml.etree.ElementTree as ET
2
3  # Membaca file XML
4  tree = ET.parse('xml/books.xml')
5  root = tree.getroot()
6
7  # Ekspresi XPath
8  titles = root.findall('.//title')
9
10 # Menampilkan judul buku
11 for title in titles:
12     print(title.text)

```

Output

```

The Catcher in the Rye
To Kill a Mockingbird
1984
The Great Gatsby
PS D:\TBD Modul 1>

```

- Dapatkan 1 buku

Dalam contoh ini, ekspresi XPath `./book[@id='{book_id}']` digunakan untuk mencari elemen "book" dengan atribut id yang sesuai dengan nilai yang Anda tentukan

Buat file 7-find-by-id.py

```

xml > x path > 7-find-by-id.py > ...
1  import xml.etree.ElementTree as ET
2
3  # Membaca file XML
4  tree = ET.parse('xml/books.xml')
5  root = tree.getroot()
6
7  # ID buku yang ingin dicari
8  book_id = '2'
9
10 # Mencari buku berdasarkan id
11 found_book = root.find(f".//book[@id='{book_id}']")
12
13 # Jika buku ditemukan, tampilkan informasinya
14 if found_book is not None:
15     title = found_book.find('title').text
16     author = found_book.find('author').text
17     print(f"Book found - Title: {title}, Author: {author}")
18 else:
19     print(f"Book with id '{book_id}' not found.")

```

```
PS D:\TBD Modul 1> & C:/Users/Backenddev/AppData/Local/Programs/Python/Python38-32/Scripts/python.exe C:/Users/Backenddev/AppData/Local/Programs/Python/Python38-32/Scripts/pip.exe install nltk
Book found - Title: To Kill a Mockingbird, Author: Harper Lee
PS D:\TBD Modul 1>
```

```
Book with id '22' not found.
PS D:\TBD Modul 1>
```

Berikut adalah contoh sederhana implementasi CRUD (Create, Read, Update, Delete) menggunakan SQLite dan Python:

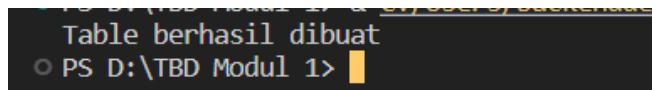
```
pip install pysqlite3
```

Buat file 1-create-table.py didalam folder sqlite

```
sqlite > 1-create-table.py > ...  
1 import sqlite3  
2  
3 def create_table():  
4     conn = sqlite3.connect('sqlite/latihan.sqlite')  
5     c = conn.cursor()  
6     c.execute('''CREATE TABLE IF NOT EXISTS users  
7         (id INTEGER PRIMARY KEY, name TEXT, age INTEGER)''')  
8     conn.commit()  
9     conn.close()  
10  
11     print('Table berhasil dibuat')  
12  
13 create_table()
```

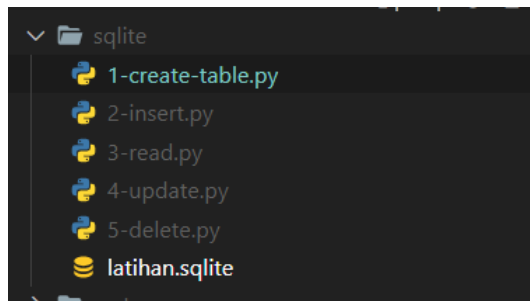
- Fungsi `create_table()` bertujuan untuk membuat tabel dalam database SQLite.
- Koneksi ke database dibuat menggunakan `sqlite3.connect()`.
- Perintah `CREATE TABLE IF NOT EXISTS` digunakan untuk membuat tabel `users` hanya jika belum ada.
- Tabel memiliki tiga kolom: `id` (sebagai primary key), `name`, dan `age`.
- Setelah membuat tabel, perubahan harus di-commit dengan `conn.commit()` dan koneksi ditutup dengan `conn.close()`.

Output terminal



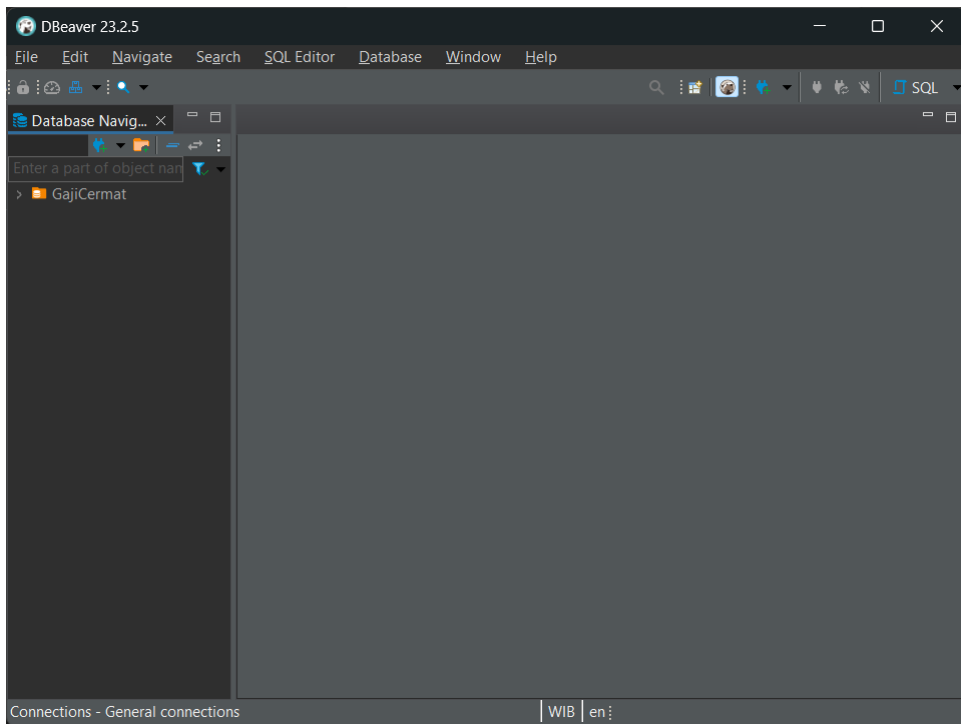
```
Table berhasil dibuat
PS D:\TBD Modul 1>
```

File SQLite dengan nama `latihan.sqlite` akan otomatis dibuat jika belum ada



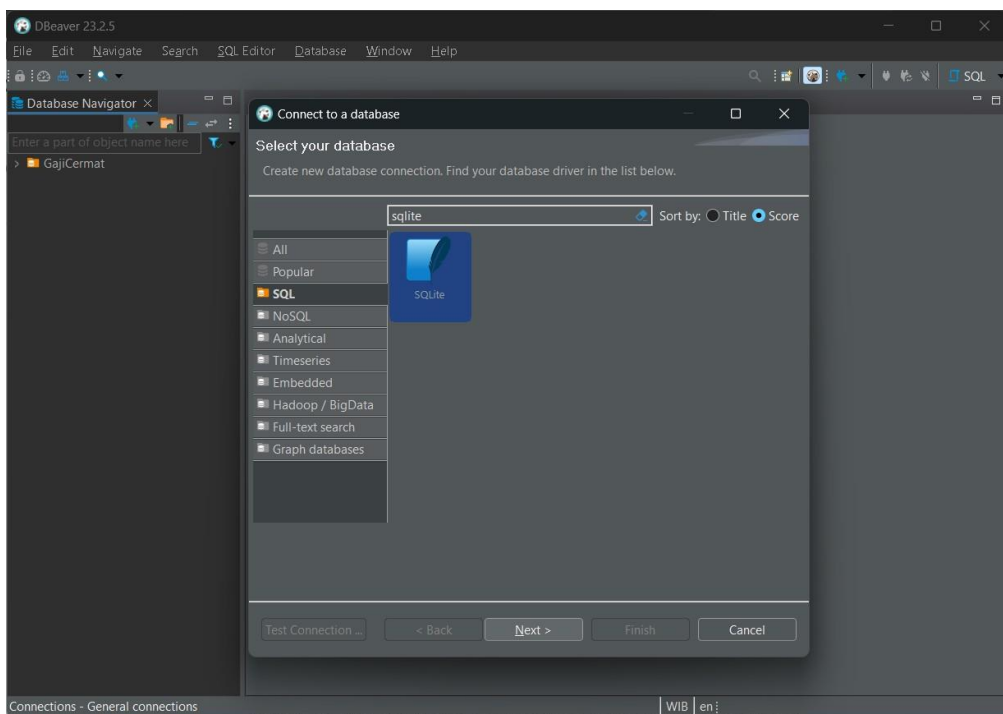
2. Membuka database SQLite `latihan.sqlite` menggunakan dbeaver

- Buka aplikasi dbeaver



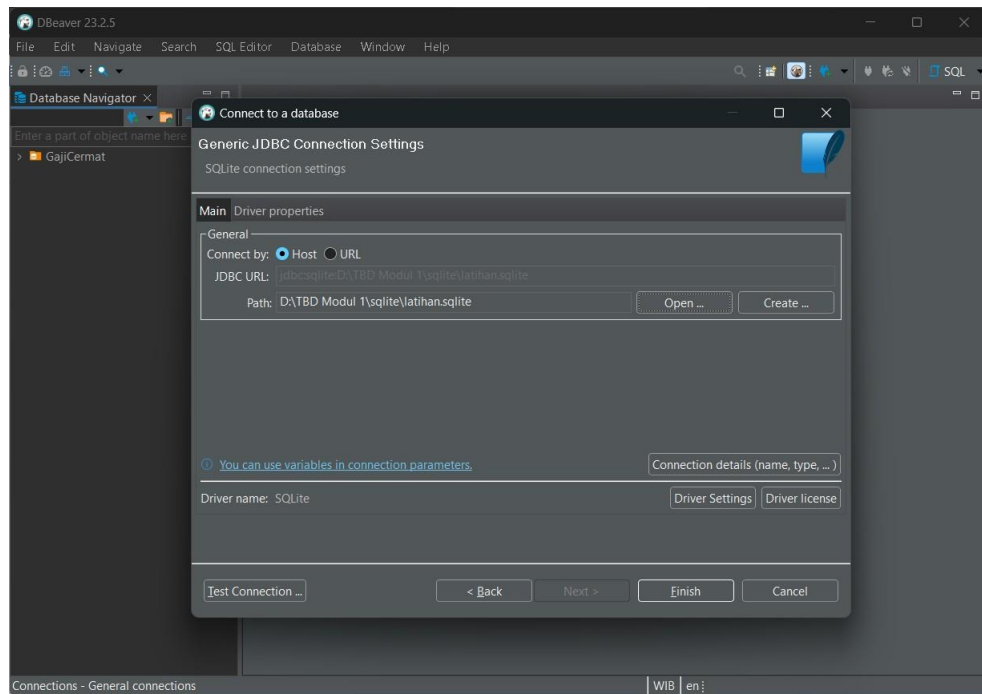
- Buat koneksi database baru

Menu toolbar database > new database connection > cari dan pilih driver sqlite
> next



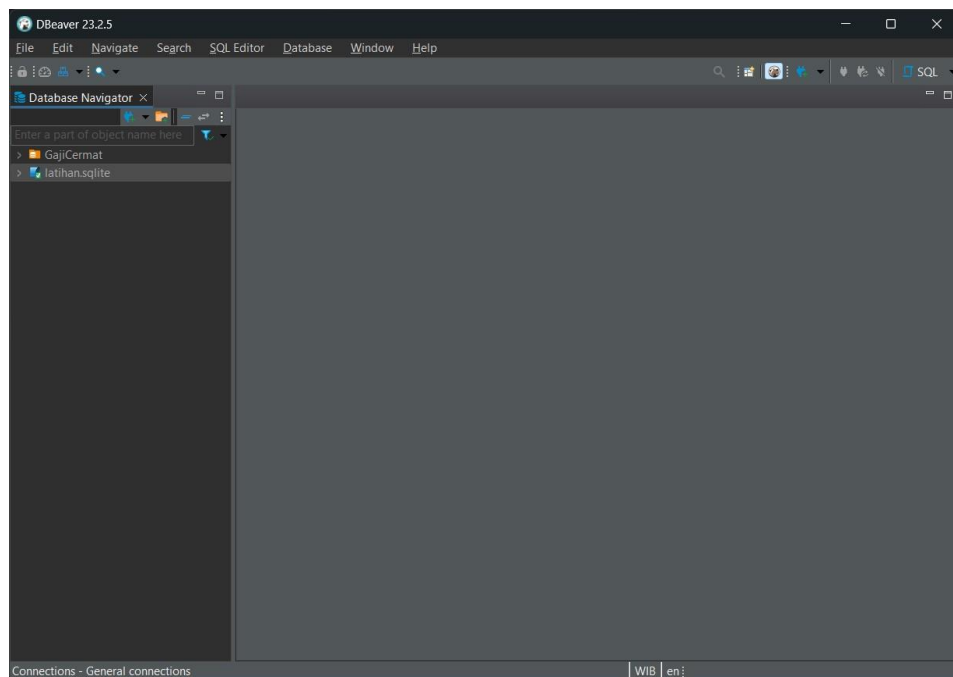
- Pilih file latihan.sqlite

Pada kolom input path klik open dan cari file latihan.sqlite di directory Anda, lalu klik finish

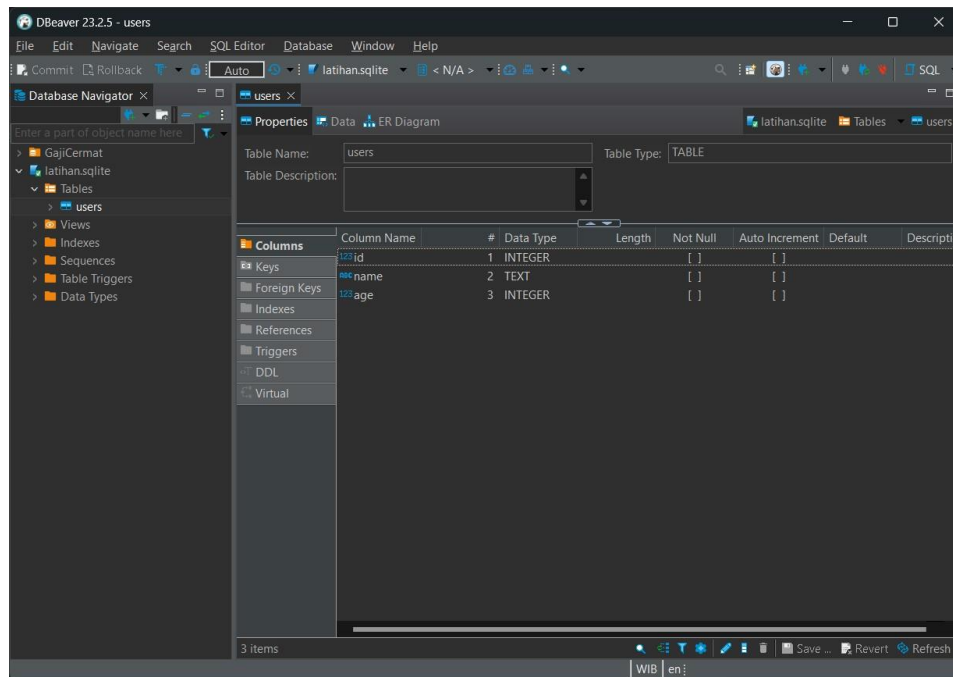


- Buka database

Setelah koneksi database berhasil ditambahkan maka bisa dilihat di list koneksi sebelah kanan



Klik koneksi database, kita bisa melihat table user yang berhasil dibuat



3. Menambahkan data

Buat file 2-insert.py

```
sqlite > 2-insert.py > ...
1  import sqlite3
2
3  def add_user(name, age):
4      conn = sqlite3.connect('sqlite/latihan.sqlite')
5      c = conn.cursor()
6      c.execute("INSERT INTO users (name, age) VALUES (?, ?)", (name, age))
7      conn.commit()
8      conn.close()
9
10     print("User berhasil ditambahkan")
11
12     add_user('John', 30)
13     add_user('Alice', 30)
```

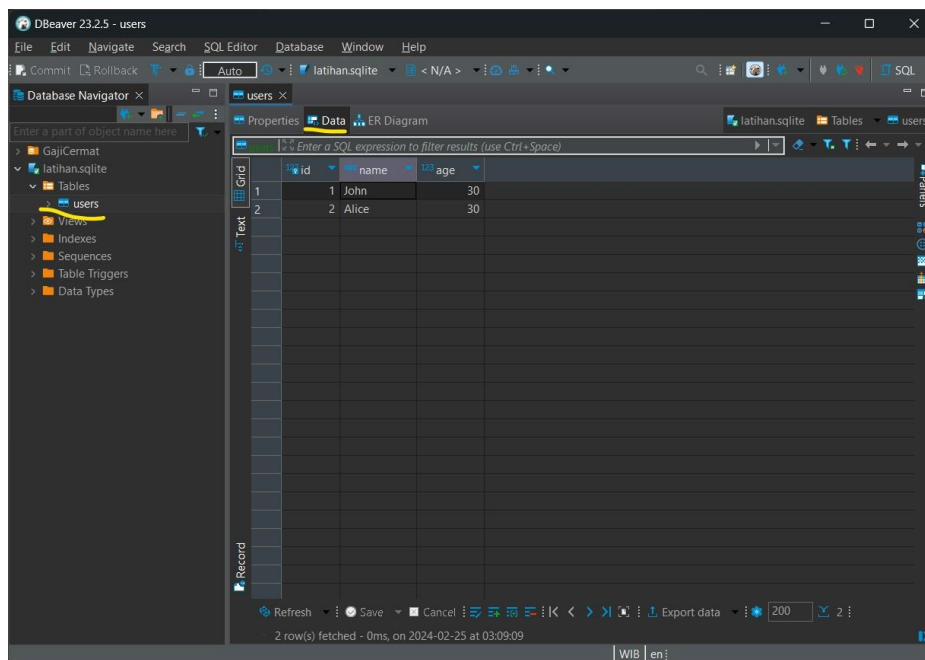
- Fungsi `add_user()` digunakan untuk menambahkan data baru ke dalam tabel.
- Koneksi ke database dibuat menggunakan `sqlite3.connect()`.
- Perintah `INSERT INTO` digunakan untuk menambahkan data ke dalam tabel `users`.
- Data yang akan dimasukkan disediakan sebagai parameter ke fungsi.

- Setelah menambahkan data, perubahan harus di-commit dengan `conn.commit()` dan koneksi ditutup dengan `conn.close()`.

Output terminal

```
User berhasil ditambahkan
User berhasil ditambahkan
PS D:\TBD Modul 1>
```

Cek di dbeaver. Double klik table users > klik tab Data



4. Mendapatkan semua data

Buat file 3-read.py

```
sqlite > 3-read.py > ...
1 import sqlite3
2
3 def get_all_users():
4     conn = sqlite3.connect('sqlite/latihan.sqlite')
5     c = conn.cursor()
6     c.execute("SELECT * FROM users")
7     rows = c.fetchall()
8     conn.close()
9     return rows
10
11 print(get_all_users())
12
```

- Fungsi `get_all_users()` digunakan untuk mendapatkan semua data dari tabel.
- Koneksi ke database dibuat menggunakan `sqlite3.connect()`.
- Perintah `SELECT * FROM users` digunakan untuk mendapatkan semua data dari tabel `users`.
- Data yang diambil dengan `c.fetchall()` dan disimpan dalam variabel `rows`.
- Koneksi ditutup dengan `conn.close()` dan data hasil query dikembalikan.

Output terminal

```
[(1, 'John', 30), (2, 'Alice', 30)]
PS D:\TBD Modul 1>
```

5. Memperbarui data

Buat file `4-update.py`

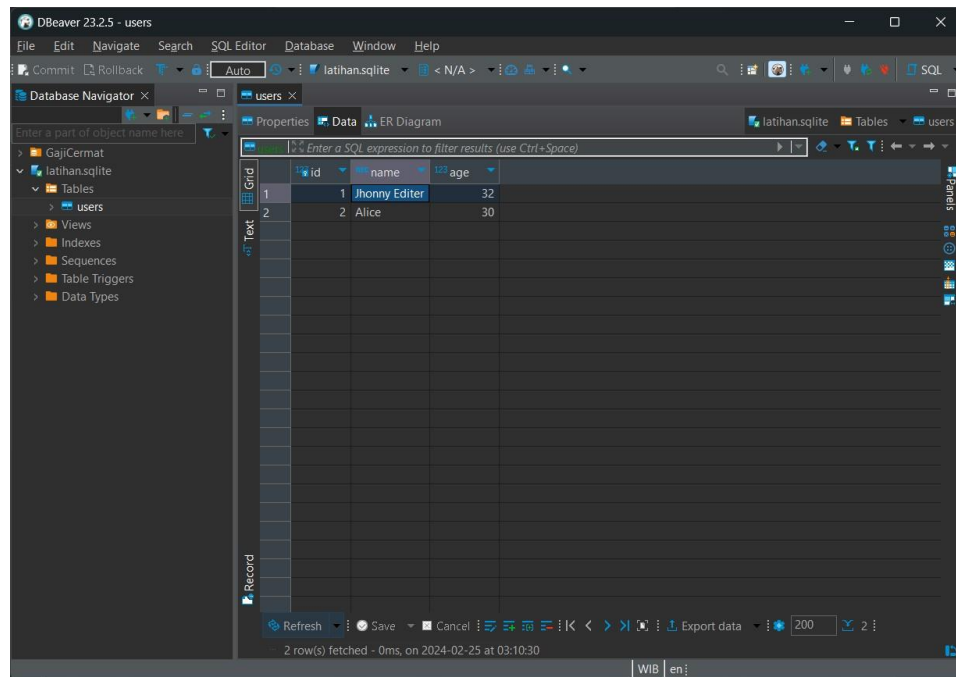
```
sqlite > 4-update.py > ...
1  import sqlite3
2
3  def get_all_users():
4      conn = sqlite3.connect('sqlite/latihan.sqlite')
5      c = conn.cursor()
6      c.execute("SELECT * FROM users")
7      rows = c.fetchall()
8      conn.close()
9      return rows
10
11 def update_user(id, name, age):
12     conn = sqlite3.connect('sqlite/latihan.sqlite')
13     c = conn.cursor()
14     c.execute("UPDATE users SET name = ?, age = ? WHERE id = ?", (name, age, id))
15     conn.commit()
16     conn.close()
17
18     print("User berhasil diperbarui")
19
20 update_user(1, "Jhonny Editer", 32)
21 print(get_all_users())
```

- Fungsi `update_user()` digunakan untuk memperbarui data yang sudah ada dalam tabel.
- Koneksi ke database dibuat menggunakan `sqlite3.connect()`.
- Perintah `UPDATE` digunakan untuk memperbarui data dalam tabel `users` berdasarkan `id`.
- Data yang akan diperbarui disediakan sebagai parameter ke fungsi.
- Setelah memperbarui data, perubahan harus di-commit dengan `conn.commit()` dan koneksi ditutup dengan `conn.close()`.

Output terminal

```
User berhasil diperbarui  
[(1, 'Jhonny Editor', 32), (2, 'Alice', 30)]  
PS D:\TBD Modul 1>
```

Cek di dbeaver



The screenshot shows the DBeaver 23.2.5 interface with the 'users' table selected. The table has three columns: 'id', 'name', and 'age'. The data is displayed in a grid view with two rows.

	id	name	age
1	1	Jhonny Editor	32
2	2	Alice	30

The interface also shows a sidebar with a tree view of the database structure, including 'GajiCermat', 'latihan.sqlite', 'Tables', 'users', 'Views', 'Indexes', 'Sequences', 'Table Triggers', and 'Data Types'. The bottom status bar indicates '2 row(s) fetched - 0ms, on 2024-02-25 at 03:10:30'.

6. Menghapus data

Buat file 5-delete.py

```

sqlite > 5-delete.py > ...
1  import sqlite3
2
3  def get_all_users():
4      conn = sqlite3.connect('sqlite/latihan.sqlite')
5      c = conn.cursor()
6      c.execute("SELECT * FROM users")
7      rows = c.fetchall()
8      conn.close()
9      return rows
10
11 def delete_user(id):
12     conn = sqlite3.connect('sqlite/latihan.sqlite')
13     c = conn.cursor()
14     c.execute("DELETE FROM users WHERE id = ?", (id,))
15     conn.commit()
16     conn.close()
17
18 delete_user(2)
19 print(get_all_users())
20

```

- Fungsi delete_user() digunakan untuk menghapus data dari tabel berdasarkan id.
- Koneksi ke database dibuat menggunakan sqlite3.connect().
- Perintah DELETE FROM digunakan untuk menghapus data dari tabel users.
- Data yang akan dihapus disediakan sebagai parameter ke fungsi.
- Setelah menghapus data, perubahan harus di-commit dengan conn.commit() dan koneksi ditutup dengan conn.close().

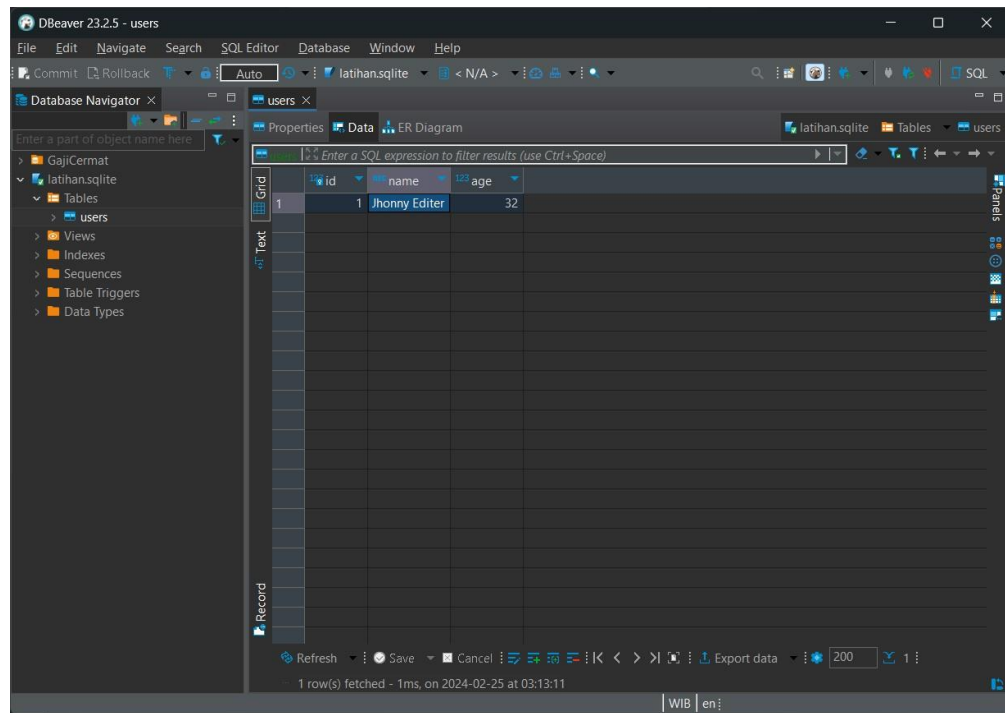
Output terminal

```

[(1, 'Jhonny Editer', 32)]
PS D:\TBD Modul 1>

```

Cek dbeaver



Daftar Pustaka

- [1] H. Book, “Hand book,” pp. 1–66, 2015, doi: 10.1109/icspcc.2015.7338978.
- [2] J. A. Kreibich, using SQLite, 1 St. United States of America: O’Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472. O’Reilly, 2010. doi: 1281104401.

