ICT233
Data Programming

Seminar 2

Dr. Kelvin DU
zddu001@suss.edu.sg

SUSS
SINGAPORE UNIVERSITY
OF SOCIAL SCIENCES

Course Structure

13-Jan-26 to 31-Mar-26: 6 lectures (3 hrs) & 6 labs (2 hrs)

S/N	Topic	Date & Time	Venue
1	Introduction to Data Programming	7pm to 10pm, Tue, 13-Jan-26	HQ BLK C-SR.C.4.16
2	Lab	7pm to 9pm, Tue, 20-Jan-26	ICT233_JAN26_T01 -> Virtual Class -> Zoom
3	Data Management	7pm to 10pm, Tue, 27-Jan-26	HQ BLK C-SR.C.4.16
4	Lab	7pm to 9pm, Tue, 3-Feb-26	ICT233_JAN26_T01 -> Virtual Class -> Zoom
5	Data Types and Structure	7pm to 10pm, Tue, 10-Feb-26	HQ BLK C-SR.C.4.16
6	**Lab (Rescheduled due to CNY)**	**10am to 12pm, Sat, 21-Feb-26**	**ICT233_JAN26_T01 -> Virtual Class -> Zoom**
7	Data Manipulation	7pm to 10pm, Tue, 24-Feb-26	HQ BLK C-SR.C.4.16
8	Lab	7pm to 9pm, Tue, 3-Mar-26	ICT233_JAN26_T01 -> Virtual Class -> Zoom
9	Data Munging	7pm to 10pm, Tue, 10-Mar-26	HQ BLK C-SR.C.4.16
10	Lab	7pm to 9pm, Tue, 17-Mar-26	ICT233_JAN26_T01 -> Virtual Class -> Zoom
11	Data Scraping	7pm to 10pm, Tue, 24-Mar-26	HQ BLK C-SR.C.4.16
12	Lab	7pm to 9pm, Tue, 31-Mar-26	ICT233_JAN26_T01 -> Virtual Class -> Zoom

3 pre-class quizzes (PCQ), 1 quiz, 1 Tutor-Marked Assignment (TMA) and 1 ECA:
(Please always refer to CANVAS for the latest update!)

Assessment	Description	Weight Allocation	Deadline
Quiz	PCQ 1	6%	06 Jan 2026 - 13 Jan 2026, 2355hrs
	PCQ 2		27 Jan 2026 - 3 Feb 2026, 2355hrs
	PCQ 3		24 Feb 2026 - 3 Mar 2026, 2355hrs
	Quiz		28 March 2026 (0900hrs) to 30 March 2026 (2355 hrs)
Assignment	TMA	24%	Thursday, 12 March 2026, 2355 hours
Examination	ECA	70%	Start of ECA: Thursday, 02 April 2026, 12 noon End of ECA: Sunday, 05 April 2026, 12 noon End of ECA Grace Period: Monday, 06 April 2026, 12 am
TOTAL		**100%**	

SUSS
SINGAPORE UNIVERSITY OF SOCIAL SCIENCES

Learning Outcomes

By the end of this course, you should be able to:

Knowledge & Understanding (Theory Component)

- Develop analytic mindset to understand and interpret datasets
- Analyze HTTP and design parsing methods for information retrieval
- Apply Object-Relational Mapping (ORM) to manage information between objects and databases
- Compose and utilize query languages for information retrieval from databases

Key Skills (Practical Component)

- Perform ETL (Extraction, Transformation, Loading) and calculations using Python and Pandas
- Develop programs to perform CRUD operations on database information
- Formulate communication methods for exchanging information over the WWW
- Conduct effective data visualization

Data Management

SUSS
SINGAPORE UNIVERSITY
OF SOCIAL SCIENCES

Learning Outcomes

At the end of this unit, you are expected to be able to:

- Appreciate the concept of databases and how they are used for storage, filtering and extraction of data
- Know the data concept and differences between SQL and NoSQL databases
- Understand how to structure database query languages to store and retrieve information
- Apply the concept of data modelling and using Object-Relational Mapping to store data objects in a database
- Know the basic CRUD (Create, Read, Update, Delete) operations for data management and applying such operations on a database

Database Fundamentals

Question:

What is the difference between data and information?

Database Fundamentals

➢ Basic Database Concepts

- Basic concepts: use SQLite3 to understand the various database concepts (https://docs.python.org/3/library/sqlite3.html)
- Schemas: the logical structure of a database
- Tables: relation
- Rows: tuple
- Columns: attribute

} primary data structures in a database schema

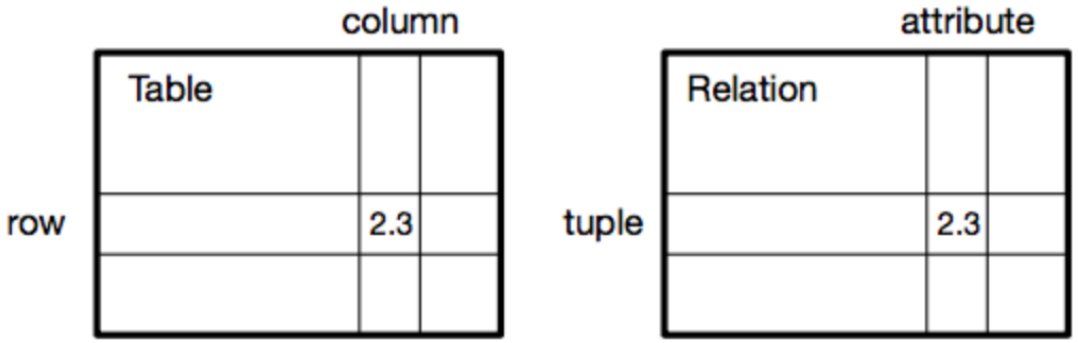


Figure 2.1 Relational Database
(Source: Python for Everybody, Charles R Severance)

Database Fundamentals

➢ Basic Database Concepts

- Schema Definitions

The following code creates a database file, music.sqlite, and a table named Tracks with two columns in the database:

```
import sqlite3
conn = sqlite3.connect('music.sqlite')

cur = conn.cursor()
cur.execute('DROP TABLE IF EXISTS Tracks')

cur.execute('CREATE TABLE Tracks (title TEXT, plays INTEGER)') conn.close()
```

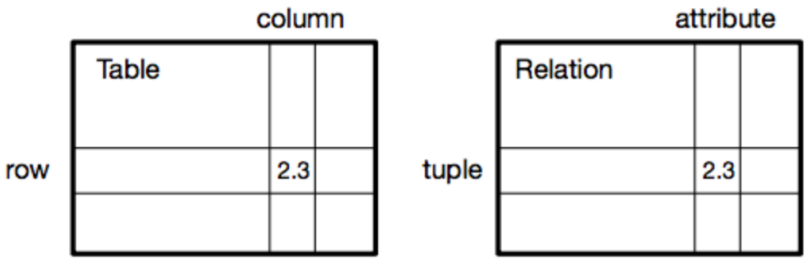


Figure 2.1 Relational Database
(Source: Python for Everybody, Charles R Severance)

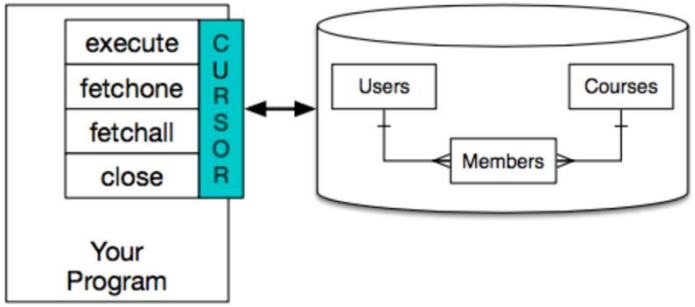


Figure 2.2 Database Cursor

Calling cursor() is conceptually similar to calling open() when dealing with text files

SUSS
SINGAPORE UNIVERSITY OF SOCIAL SCIENCES

Database Fundamentals

➢ Basic Database Concepts

- **Structured Query Language or SQL**

- remove the Tracks table from the database if it exists

DROP TABLE IF EXISTS Tracks

- create a table named Tracks with a text column named title and an integer column named plays

CREATE TABLE Tracks (title TEXT, plays INTEGER)

- **Basic CRUD (Create, Read, Update, Delete) Operations using SQL**

Create - Insert a row of data into the Tracks table which has the columns "title" and "plays"

INSERT INTO Tracks (title, plays) values ('Rainbow connection', 20)

Read - Get all rows from Tracks table where column / attribute "plays" = 20

SELECT * FROM Tracks WHERE plays=20

Update - In the Tracks table, in rows where column / attribute "title" = 'Rainbow connection", set the attribute "plays" to 5

UPDATE Tracks SET plays=5 WHERE title='Rainbow connection'

Delete - In the Tracks table, delete all rows where column / attribute "plays" = 5

DELETE FROM Tracks WHERE plays=5

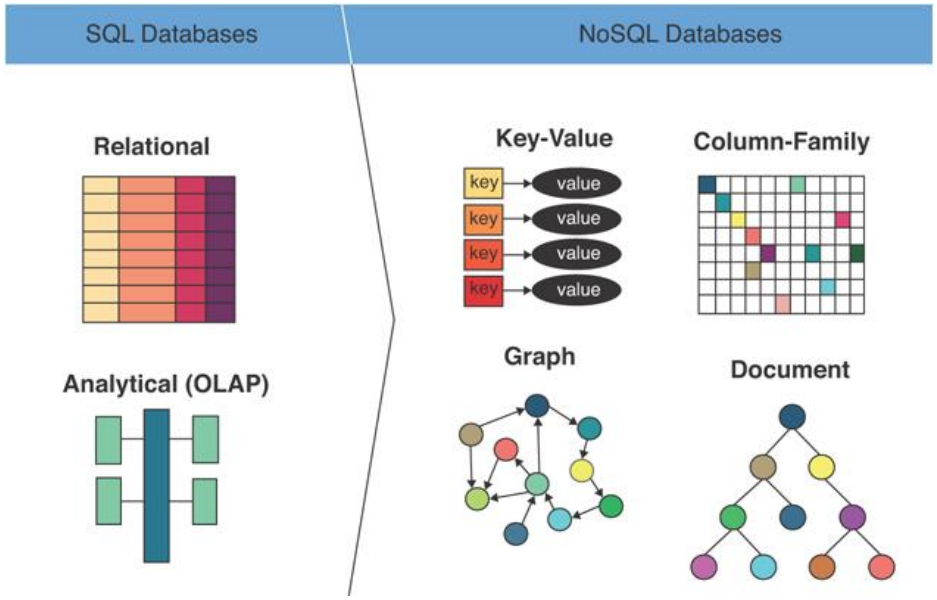
Using these SQL commands, we can write the following program to insert some data into our `Track` table:

```
import sqlite3
conn = sqlite3.connect('music.sqlite')
cur = conn.cursor()
cur.execute('INSERT INTO Tracks (title, plays) VALUES (?, ?)',
    ('Thunderstruck', 20))
cur.execute('INSERT INTO Tracks (title, plays) VALUES (?, ?)',
    ('My Way', 15))
conn.commit()
print('Tracks:')
cur.execute('SELECT title, plays FROM Tracks')
for row in cur:
    print(row)
```

SUSS SINGAPORE UNIVERSITY OF SOCIAL SCIENCES

Database Fundamentals

SQL	NoSQL
Relational	Not Relational
Tables	Documents, Key values, graphs
Scale Up	Scale Across
SQL Query	Non Standard
Complex Query syntax	Less complex
Transaction workload – atomicity, integrity	Loosely coupled workload



Database Fundamentals

- **NoSQL Databases**
- MongoDB - a popular document-centric NoSQL database
- a noSQL Document " = " JSON " = " Dictionary
- Program

```
from pymongo import MongoClient

URI = 'mongodb://user:password1@ds016118.mlab.com:16118/demo'
conn = MongoClient(URI)
db = conn['demo']
collection = db['user']
data = collection.find()
data.count()
```

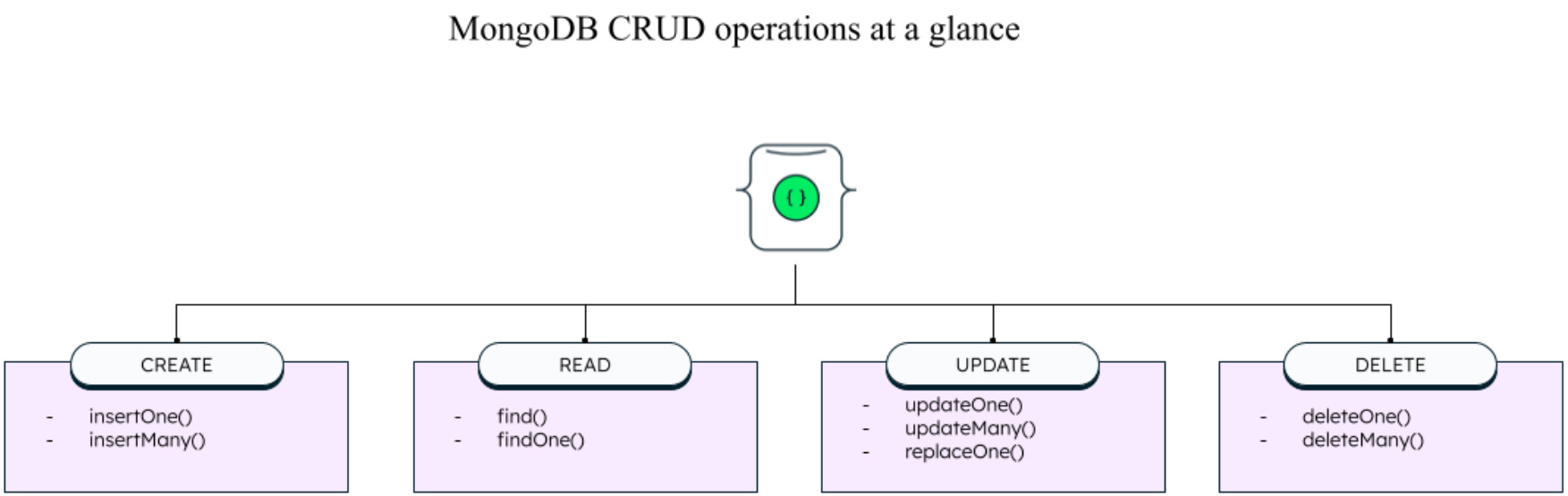
```
{
    "_id": {
        "$oid": "5b1f3531f9f31504cd8fbf0c"
    },
    "name": "john",
    "email": "john@gmail.com"
}
{
    "_id": {
        "$oid": "5b1f35aaf9f31504cd8fbf0f"
    },
    "name": "mary",
    "email": "mary@yahoo.com"
}
{
    "_id": {
        "$oid": "5b1f3a96f9f315051b0be5f3"
    },
    "name": "peter",
    "email": "peter@hotmail.com"
}
```

SUSS
SINGAPORE UNIVERSITY OF SOCIAL SCIENCES

Database Fundamentals

➢ NoSQL Databases

- Basic CRUD (Create, Read, Update, Delete) Operations (https://www.mongodb.com/resources/products/fundamentals/crud)



Database Fundamentals

➢ NoSQL Databases

- Basic CRUD (Create, Read, Update, Delete) Operations for MongoDB

CURD	Description	Example
insertOne()	Insert one document into the collection	db.RecordsDB.insertOne({ name: "Marsh", age: "6 years", species: "Dog", ownerAddress: "380 W. Fir Ave", chipped: true })
insertMany()	Insert multiple items at one time	db.RecordsDB.insertMany([{ name: "Marsh", age: "6 years", species: "Dog", ownerAddress: "380 W. Fir Ave", chipped: true}, {name: "Kitana", age: "4 years", species: "Cat", ownerAddress: "521 E. Cortland", chipped: true}])
find()	Get all the documents from a collection or find a desired subsection of the records	db.RecordsDB.find() db.RecordsDB.find({"species":"Cat"})
findOne()	Get one document that satisfies the search criteria	db.{collection}.findOne({query}, {projection})
updateOne()	Update a currently existing record and change a single document	db.RecordsDB.updateOne({name: "Marsh"}, {$set:{ownerAddress: "451 W. Coffee St. A204"}})
updateMany()	Update multiple items by passing in a list of items	db.RecordsDB.updateMany({species:"Dog"}, {$set: {age: "5"}})
replaceOne()	Replace a single document in the specified collection	db.RecordsDB.replaceOne({name: "Kevin"}, {name: "Maki"})
deleteOne()	Remove a document from a specified collection on the MongoDB server	db.RecordsDB.deleteOne({name:"Maki"})
deleteMany()	Delete multiple documents from a desired collection with a single delete operation.	db.RecordsDB.deleteMany({species:"Dog"})

Database Fundamentals

- **Handling Complex Data**
- **Conversions (between Python and database, JS and Python)**
- **SQL Data types**
  - Numeric Type (Fix, Float, Range)
  - Date and Time Type (Formats)
  - String Type (Size)
  - Spatial Data Types
  - JSON
- **noSQL Data types**
  - Document-based Store (XML, JSON, BSON)
  - Key-value store (String, JSON, BLOB)
  - Graph-based
  - Column-based

Database Fundamentals

Activity 1

Let's take a date and time, in Python datetime format, convert it into a string, and see how JavaScript can process it. First, we use Python datetime library to produce the string

Database Concepts

➢ Data Modelling

- **Data modelling in database design and object-oriented programming:** designers create a conceptual model of how the various data items in the software application relate to each other and produce a design document that shows the tables and their relationships. The document is called a data model.

- **Relationship**
  - One-to One
  - One to Many
  - Many to Many

CUSTOMERS		
customer_id	customer_name	address_id
101	John Doe	301
102	Bruce Wayne	302
ADDRESSES		
address_id	address	
301	12 Main St., Houston TX 77001	
302	1007 Mountain Dr., Gotham NY 10286	

Figure 2.4 One to One Relationship Tables

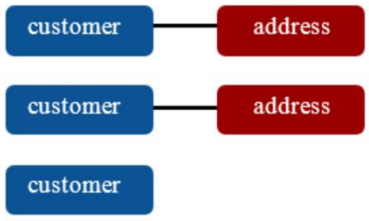


Figure 2.5 One to One Relationship

CUSTOMERS	
customer_id	customer_name
101	John Doe
102	Bruce Wayne

ORDERS			
order_id	customer_id	order_date	amount
555	101	12/24/09	$156.78
556	102	12/25/09	$99.99
557	101	12/26/09	$75.00

Figure 2.6 One to Many Relationship

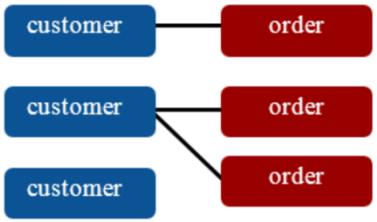


Figure 2.7 One to Many Relationship

ORDERS			
order_id	customer_id	order_date	amount
555	101	12/24/09	$156.78
556	102	12/25/09	$99.99
ITEMS			
item_id	item_name	item_description	
201	Tickle Me Elmo	It wants to be tickled	
202	District 9 DVD	Awesome sci-fi movie	
203	Batarang	It is very sharp	
ITEMS_ORDERS			
order_id	item_id		
555	201		
555	202		
556	202		
556	203		

Figure 2.8 Many to Many Relationship

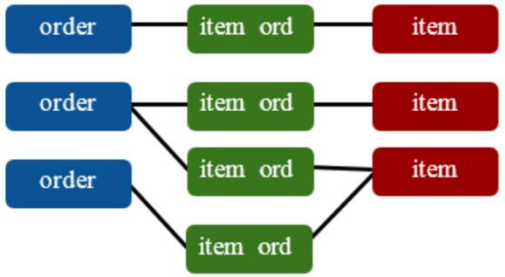
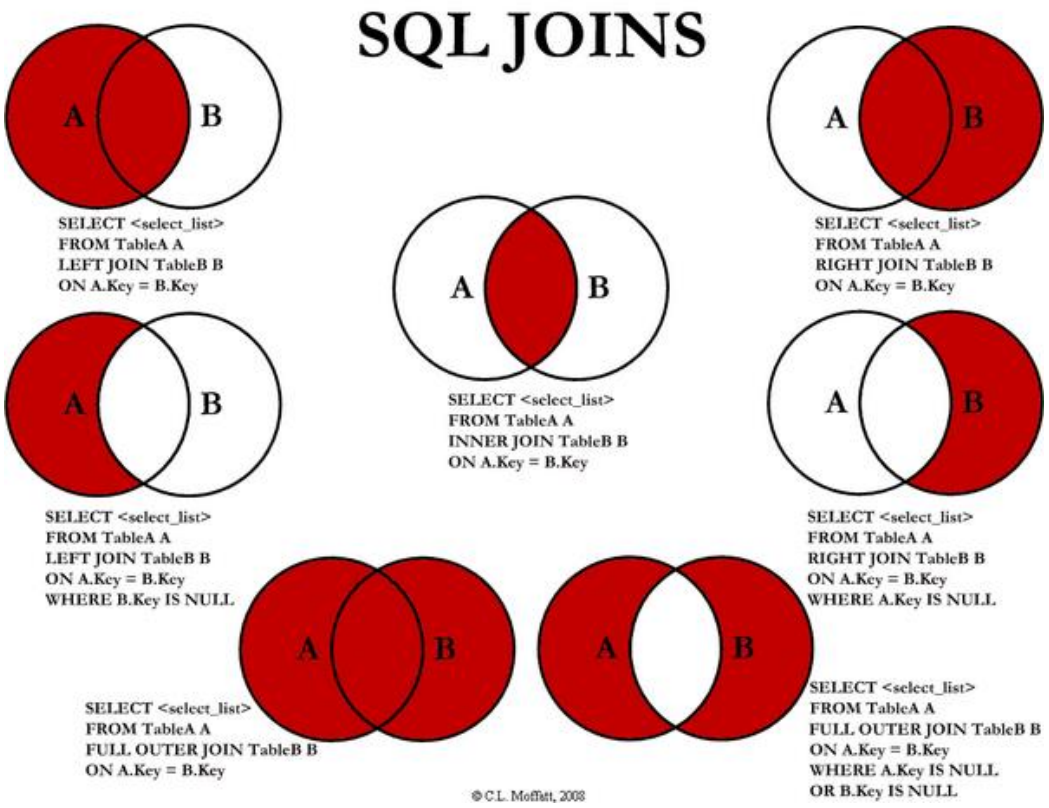


Figure 2.9 Many to Many Relationship

SUSS
SINGAPORE UNIVERSITY OF SOCIAL SCIENCES

Database Concepts

- **Data Modelling**
- **Relational Databases operations**
  - Select Joins (inner, outer, left, right)
  - Insert (child, parent)
  - Delete (child, parent)
  - Update



https://www.codeproject.com/Articles/33052/Visual-Representation-of-SQL-Joins

Database Concepts

- **Data Modelling**

- **Non Relational Databases**

  - Redis (key–value database) https://redis.io/resources/8-data-modeling-patterns-in-redis/

  - Neo4j (Graph database) https://neo4j.com/docs/getting-started/data-modeling/

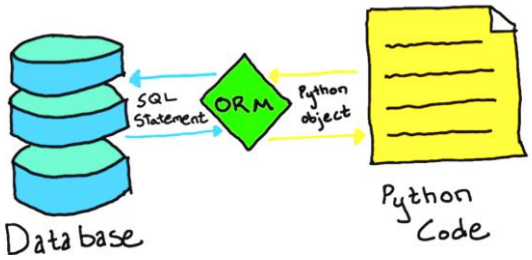
Object Relational Mapping (ORM)

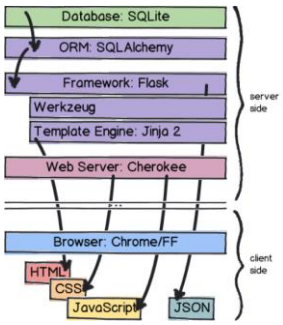
- **Object Relational Mapping (ORM)**
- **Decoupling Code from SQL**
- **Abstracts database specific implementations**

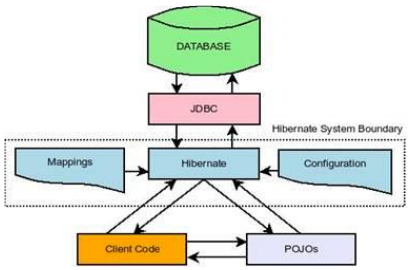
Object-Relational Mapping (ORM) is a programming technique for converting data between incompatible type systems using object-oriented programming languages.

An object is an instance of a class, as defined in object-oriented programming paradigm.

```
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy import Column, Integer, String, Enum, Float,
ForeignKey

from sqlalchemy.orm import relationship

Base = declarative_base()

class Customer(Base):
    __tablename__ = 'customers'

    customer_name = Column(String)
    customer_id = Column(Integer, primary_key=True)
    orders = relationship("Order", back_populates="customer")

    def __repr__(self):
        return "%s %s" %(self.customer_id, self.customer_name)


class Order(Base):
    __tablename__ = 'orders'
    order_id = Column(Integer, primary_key=True)
    amount = Column(Float)
    customer_id = \
       Column(Integer,  ForeignKey('customers.customer_id'))
    customer = relationship("Customer", back_populates="orders")


    def __repr__(self):
        return "%s %s" %(self.customer_id, self.amount)
```







SUSS
SINGAPORE UNIVERSITY OF SOCIAL SCIENCES

Object Relational Mapping (ORM)

- **Object Relational Mapping (ORM)**
- **CRUD Operations**

```
from sqlalchemy.orm import sessionmaker
Session = sessionmaker(bind=engine)
session = Session()

# Read
result = session.query(Customer)
for r in result:
  print (r)

# To apply filtering (e.g. search for ‘Andy’ in customer_name), we use the following syntax:
result = session.query(Customer).filter(Customer.customer_name == 'Andy’)
for r in result:
  print (r)

# Create
john = Customer(customer_name="John")
session.add(john)
session.commit()

# Update
person = session.query(Customer).filter(Customer.customer_name == 'Joe')[0]
person.customer_name = 'newJoe’
session.commit()

# Delete
person = session.query(Customer).filter(Customer.customer_name == 'newJoe').delete()
```

Object Relational Mapping (ORM)

Activity 2

Activity 2.1

- The following data.gov.sg resource contains information about mobile subscribers in Singapore, using different types of technologies, at different periods of time: total-mobile-phone-subscriptionstotal-number-of-mobilesubscriptions-by-type
- Download the relevant information to set up a database. Write a Python program that connects to SQLite3 database, so that we can use it to perform queries, e.g. "What is the number of 2G subscribers on Jan 2015?"

Activity 2.2

- Assuming you are working as a market analyst for a Telco and you are tasked to analyse the information that you have put into the database. You are hence thinking of writing a simple Python application that helps to answer questions like "What is the number of 2G pre-paid subscribers on Jan 2015?" or "What is the number of 3G post-paid subscribers on Jan 2016?".
- Using the information in the table stored in the database above, design a Python class that can help to calculate the required data for the questions. What kind of class method would you need to create ?

Thank you!

Q&A