SUSS
SINGAPORE UNIVERSITY
OF SOCIAL SCIENCES

Aug 2024

Data Management Seminar 2

ICT233 Data Programming

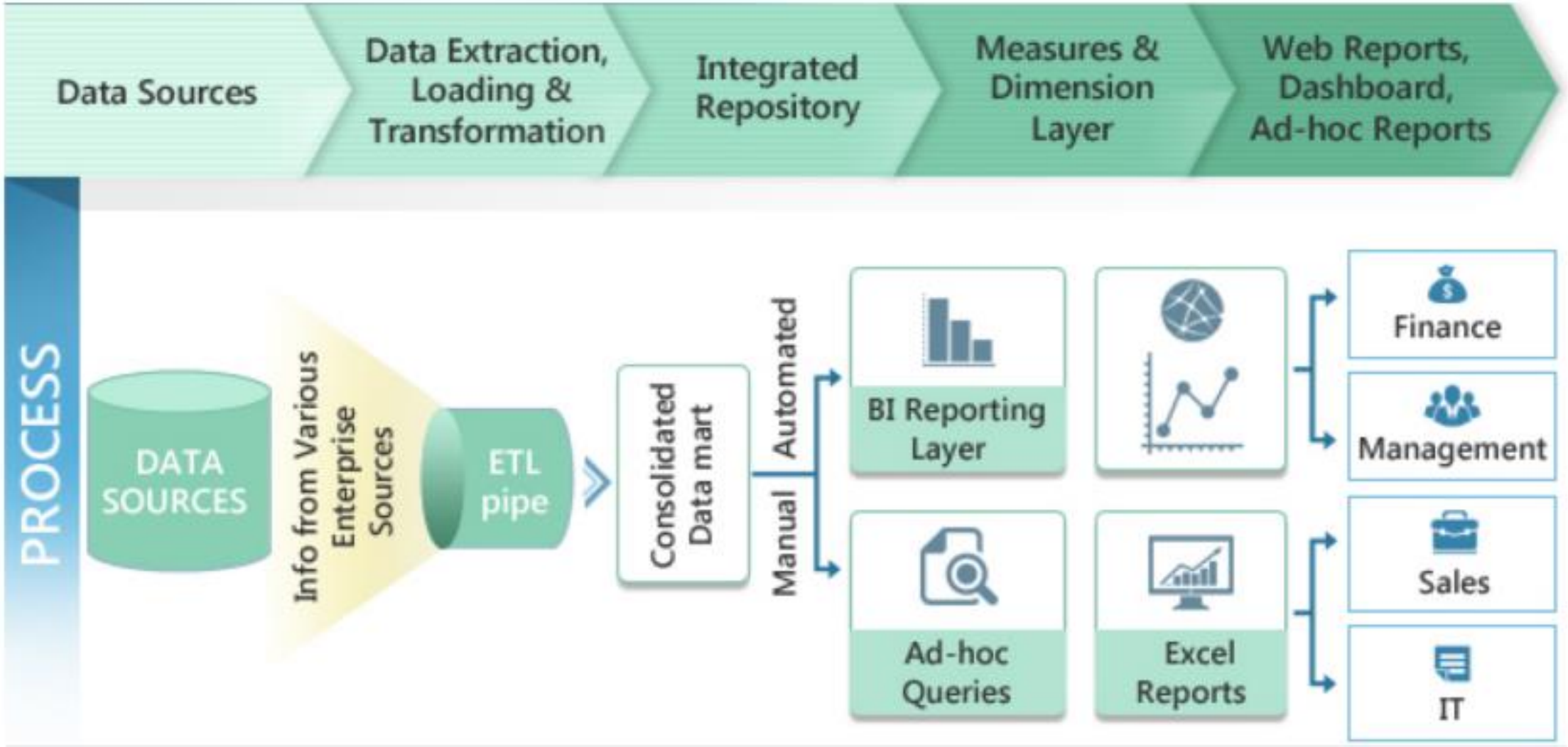
Veronica Hu **huhe001@suss.edu.sg**

RECAP

S1 Key Learning Objectives

1) Recap using Python programming language to retrieve, create, and update data via a variety of data sources and file types.
2) Reflect on the various techniques of processing and parsing information, e.g., using regular expression libraries to recognize and process text patterns.
3) Understand how information is presented, delivered, and consumed on the internet via HTTP (Hypertext Transport Protocol).
4) Understand the concept of web services as a mean to provide and exchange information between consumers and providers of information.
5) Appreciate the architectural philosophies behind web services.
6) Know the popular data formats (XML, JSON) and their relevant usage patterns in information retrieval and exchanges.

Business Reporting



ETL Process

Seminar 2

Extract

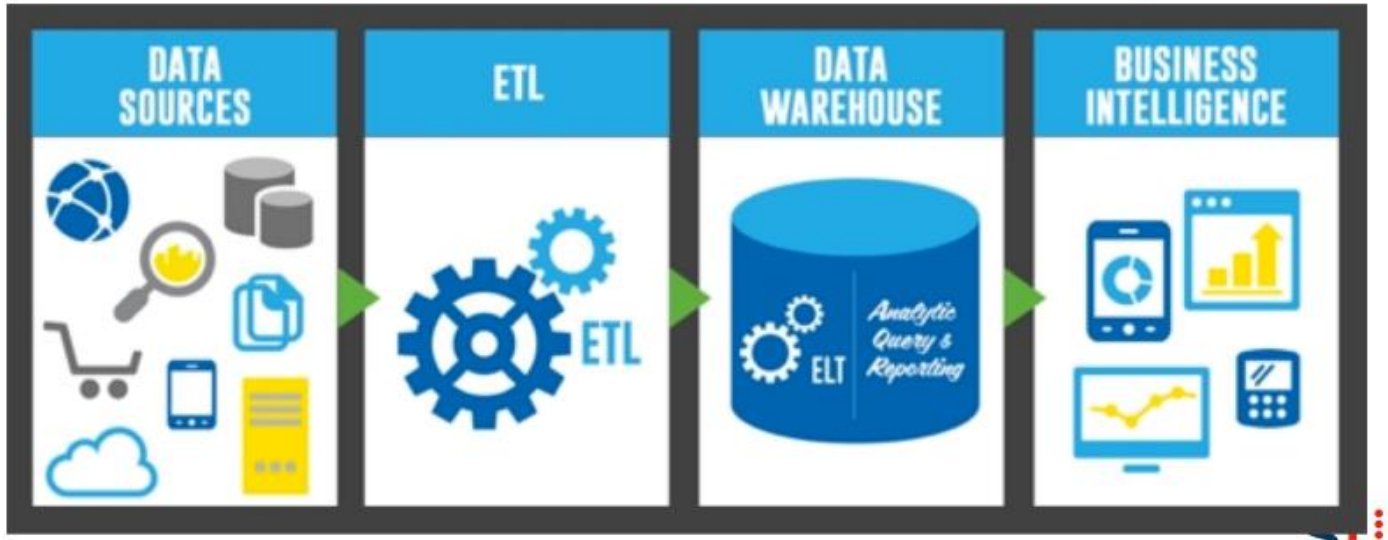
- One or more source systems containing customer, financial, or product data (CRM, Accounting system, Warehouse, MES)
- Files types - Flat files, XML, Oracle, IBM DB2, SQL Server,, IBM Websphere MQ, ODBC, JDBC, Hadoop Distributed File System (HDFS), Hive/HCatalog, JSON, Mainframe (IBM z/OS), Salesforce.com, SAP/R3

Transform

- Applying business rules, cleansing, and validating the data.
- Aggregation, Copy, Join, Sort, Merge, Partition, Filter, Reformat, Lookup
- Mathematical: +, -, x, /, Abs, IsValidNumber, Mod, Pow, Rand, Round, Sqrt, ToNumber, Truncate, Average, Min, Max
- Logical: And, Or, Not, IfThenElse, RegEx, Variables
- Text: Concatenate, CharacterLengthOf, LengthOf, Pad, Replace, ToLower, ToText, ToUpper, Translate, Trim, Hash
- Date: DateAdd, DateDiff, DateLastDay, DatePart, IsValidDate
- Format: ASCII, EBCDIC, Unicode

Load

- Load the results into one or more target systems such as a data warehouse, datamart, or business intelligence reporting system. Output: Flat files, XML, Oracle, IBM DB2, SQL Server, Teradata, Sybase, Vertica, Netezza, Greenplum, ODBC, JDBC, Hadoop Distributed File System (HDFS), Hive/HCatalog, Mainframe (IBM z/OS), Salesforce.com, Tableau, QlikView



SEMINAR OVERVIEW

DATA MANAGEMENT – LEARNING OBJECTIVES

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

Chapter 1: Database Fundamentals

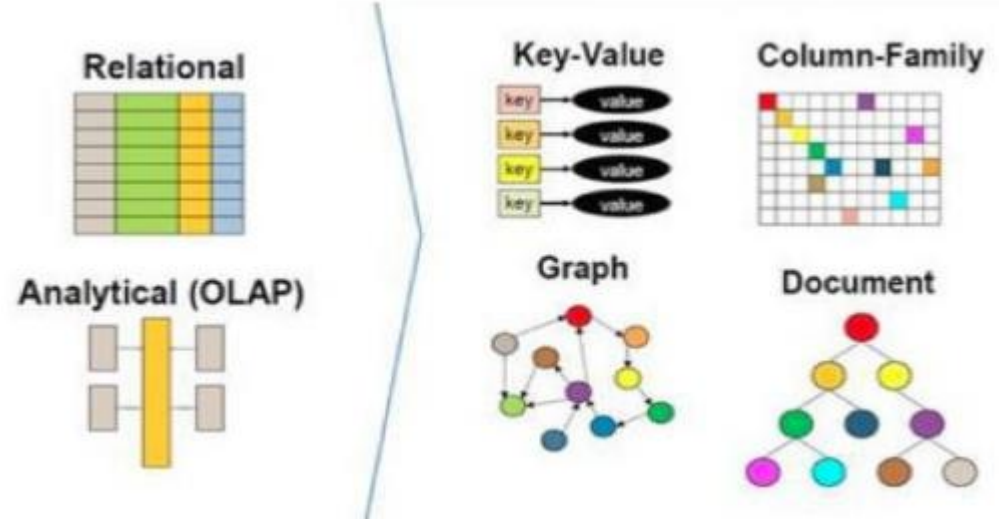
1.1 Basic Database Concepts

- An organized collection of data stored as a file on a server
- Data stores on permanent storage like disks
- Stores more data than a computer's memory
- Indexes/Keys
  - find specific data
  - enable quick data retrieval
- Optimizes data insertion and access

Chapter 1: Database Fundamentals

1.2 Type of Database

SQL	noSQL
Relational	Not
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



Chapter 1: Database Fundamentals

1.3 Structured Query Language (SQL)

- Basic CRUD Operations
  - 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
```

Chapter 2: Database Concepts

2. 1 SQL Database - SQLite3

- Basic concepts
- Schemas and Tables
- Rows & Columns

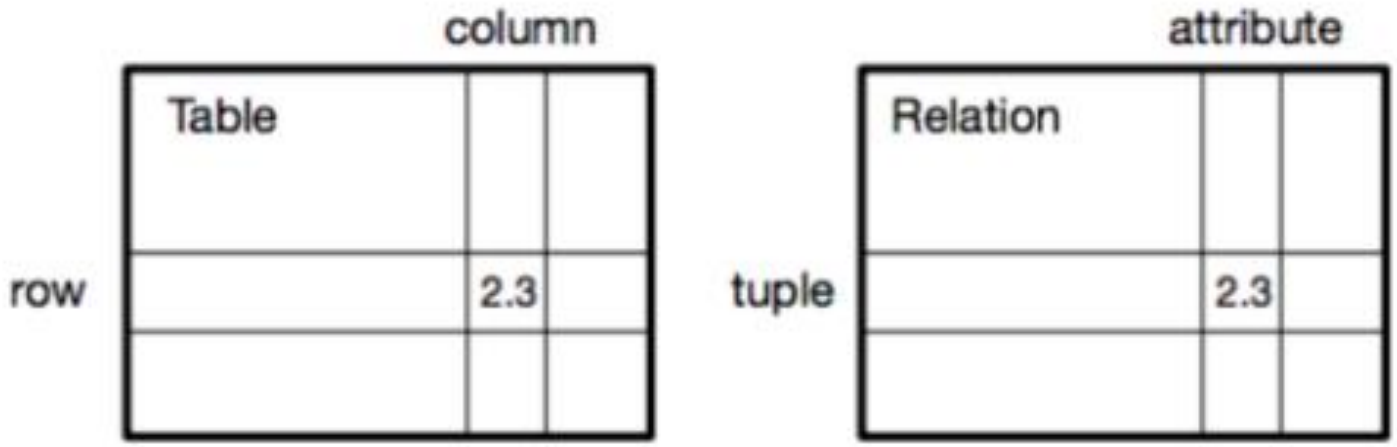


Figure 2.1 Relational Database
(Source: Python for Everybody, Charles R Severance)

Chapter 2: Database Concepts

2. 1 SQL Database - SQLite3

- 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()
```

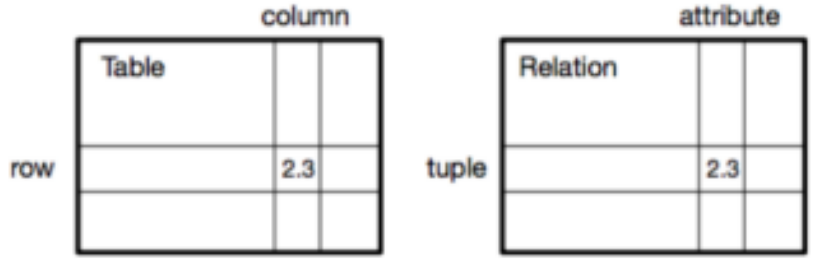
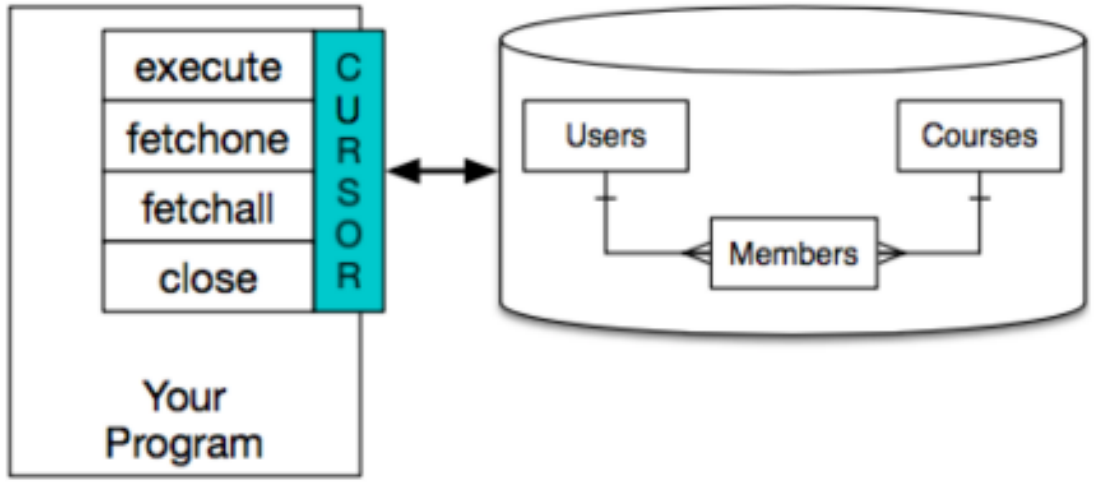


Figure 2.1 Relational Database
(Source: Python for Everybody, Charles R Severance)



Chapter 2: Database Concepts

2. 1 SQL Database - SQLite3

- Program

```
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))

# send SQL statments to the database
conn.commit()
print('Tracks:')
cur.execute('SELECT title, plays FROM Tracks')


# print all rows from the table
for row in cur:
    print(row)
```

Chapter 2: Database Concepts

2. 2 SQL Database - Data Modelling and Application

- Data Modelling
  - Relationship
    - One-to One
    - One to Many
    - Many to Many

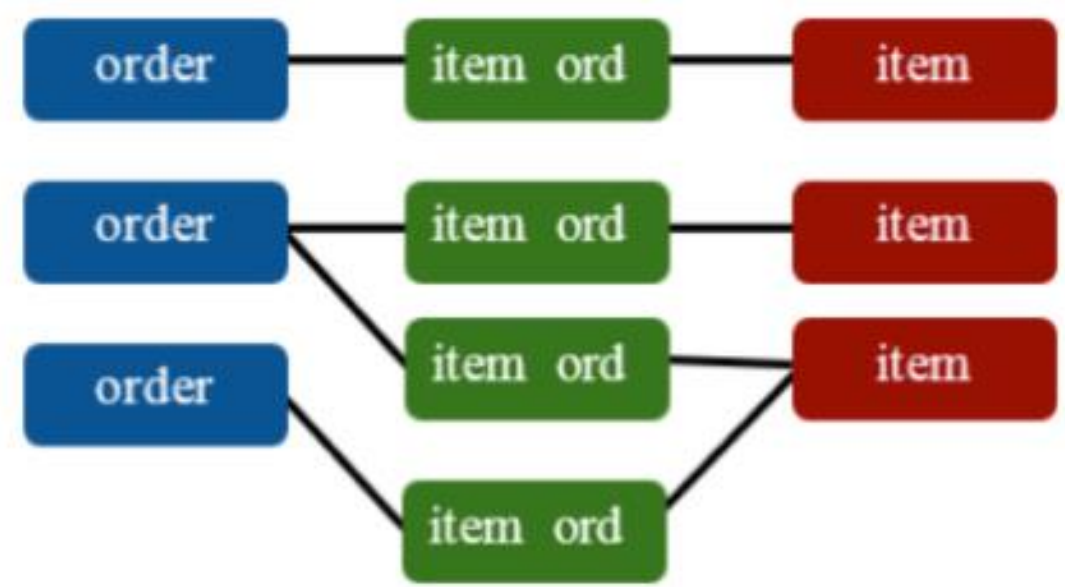
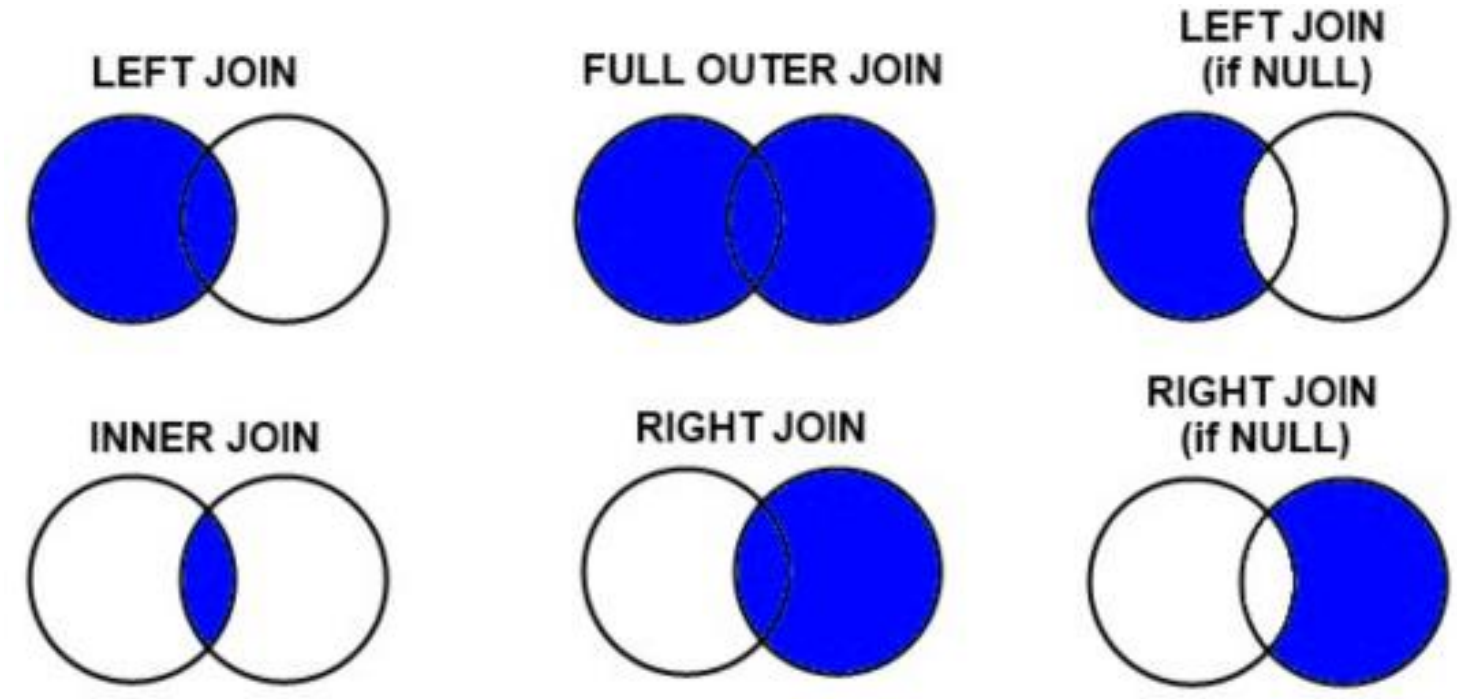


Figure 2.9 Many to Many Relationship
(Source: Created by Developer)

Chapter 2: Database Concepts

2. 2 SQL Database - Data Modelling and Application

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



Chapter 2: Database Concepts

2. 3 NoSQL Database - MongoDB

- a noSQL Document “ = “ JSON “ = “ Dictionary”
- Program

```
from pymongo import MongoClient

#URI = 'mongodb://user:password1@ds016118.mlab.com:16118/demo?retryWrites=false'
conn = MongoClient()
db = conn['demo']#conn.demo
print("Collections:", db.list_collection_names())
collection = db.posts #db["posts"]
print("Item count:", collection.count_documents({}))

for d in collection.find({}):
    print (d)
```

```
Collections: []
Item count: 0
```


```
{
    "_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"
}
```

Chapter 2: Database Concepts

2. 3 NoSQL Database - MongoDB

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

```
#Create: To add new data to a collection, the syntax is:
collection.insert_one({'author': 'veronica',
  'date': "2020-08-04",
  'tag': ['mongodb', 'python', 'pymongo'],
  'text': 'yeahyeah'})

result = collection.find({'text': 'yeahyeah'})
list(result)
```

Output:
[{'_id': ObjectId('63e24b8e9a961bd14e99436e'), 'author': 'veronica', 'date': '2020-08-04', 'tag': ['mongodb', 'python', 'pymongo'], 'text': 'Create Record'},
…
]

```
# Read: To find a particular item in a collection,
#       we use the find method taking a
#       dictionary query:
result = collection.find({'text': 'hihi'})
print(list(result))
```

Output:
[{'_id': ObjectId('63e24bcf9a961bd14e994372'), 'author': 'veronica', 'date': '2020-08-04', 'tag': ['mongodb', 'python', 'pymongo'], 'text': 'hihi'}]

Chapter 2: Database Concepts

2. 3 NoSQL Database - MongoDB

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

```
# Update: added new attribute artist to one of the existing records
collection.find_one_and_update(\
                                {'plays': '7'}, {'$set': {'plays': '39', 'artist': 'Jay'}})
```

```
# Delete
collection.delete_many(\
                        {'song':'Good Guy'})
```

Chapter 2: Database Concepts

2. 4 Handling Complex Data

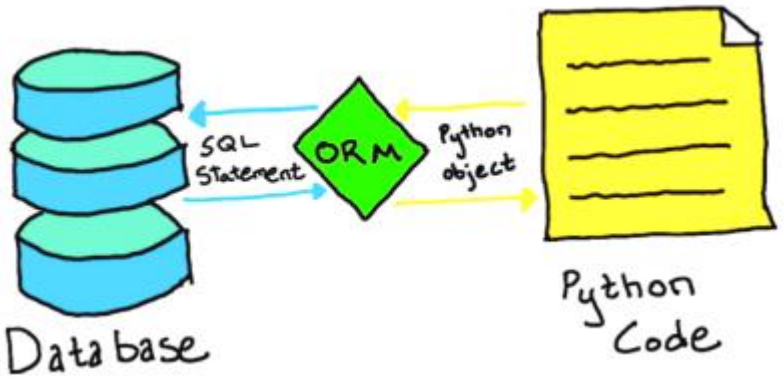
- Conversions (between Python and database, JS and Python)
- SQL Data types
  - Numeric Type (Fix, Float, Range)
  - Date and Time Type (Formats)
  - String Type (Size)
  - Spatial Data Types
  - JSON
- noSQL Data types
  - Document-based Store (XML, JSON, BSON)
  - Key-value store (String, JSON, BLOB)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

Demo on Activity 1

Chapter 3: Object Relational Mapping (ORM)

- Decoupling Code from SQL
- Abstracts database specific implementations

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

# 1) create a Base object using declarative_base.
Base = declarative_base()

# 2) Base object is used to create the databases's table schemas

# define the two tables, Customers and Orders
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)
```



Chapter 3: Object Relational Mapping (ORM)

- CRUD Operations

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

```
result = \
    session.query(Customer).filter(Customer.customer_name == 'Andy')
for r in result:
    print r
```

```
person = \
  session.query(Customer).filter(Customer.customer_name == 'Joe')[0]

person.customer_name = 'newJoe'
session.commit()
```

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

Activity

Activity 2

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-mobile-subscriptions-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?”

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 ?

SEMINAR LEARNING OUTCOME

Data MANAGEMENT – LEARNING OBJECTIVES

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

THANK YOU