

HO CHI MINH CITY UNIVERSITY OF TRANSPORT**Kiến thức - Kỹ năng - Sáng tạo - Hội nhập**Sứ mệnh - Tầm nhìnTriết lý Giáo dục - Giá trị cốt lõi

Contents

0	Database and Python Resources	2
0.1	Database	2
0.2	Python Environment	2
0.3	Python is a programming interface	2
1	Student Management Database (SMDB)	2
2	Retail Invoice Database (RIDB)	4
3	Warehouse Management Database (WMDB)	5
4	Order Management Database (OMDB)	6
5	Vietnam Geographic Database (VGDB)	7

0 Database and Python Resources

0.1 Database

1. SQLite: <https://sqlite.org>
Tool: <https://sqlitebrowser.org> , <https://dbeaver.io>
2. PostgreSQL: <https://www.postgresql.org>
Tool: <https://www.pgadmin.org> , <https://dbeaver.io>

0.2 Python Environment

1. Online: <https://colab.research.google.com>
2. Offline: Anaconda → Jupyter Notebook: <https://www.anaconda.com/products/individual-d>

0.3 Python is a programming interface

1. Python tutorial: <https://pythonbasics.org/>
2. Using python to connect with database to execute queries.
3. Tkinter GUI: <https://docs.python.org/3/library/tk.html>
4. PyQt: <https://www.pythonguis.com/>

1 Student Management Database (SMDB)

1. **Subject (SubjectID, SubjectName, Units)**
Predicate: Each subject (**Subject**) has a certain code (**SubjectID**) to distinguish it from other subjects; We know the subject name (**SubjectName**) and the units (**Units**) for that subject.
2. **Class (ClassID, ClassName, ClassYear)**
Predicate: Each class (**Class**) has a unique code (**ClassID**) to distinguish it from other classes; We know the class name (**ClassName**) and the year of that class (**ClassYear**).
3. **Student (StudentID, StudentName, StudentAddress, ClassID)**
Predicate: Each student (**Student**) has a unique code to distinguish it from other students (**StudentID**); We know the student's name (**StudentName**), student address (**StudentAddress**) and that student's class (**ClassID**).
4. **StudentGrades (StudentID, SubjectID, Grades)**
Predicate: Student grades relational scheme (**StudentGrades**) records grade (**Grades**) of subject (**SubjectID**) for student (**StudentID**).

Require

1. Determine all the keys of the Relational Schemes.
2. Create database SMDB.
3. Create the Relational Schemes.
4. Insert data:
 - **Subject**. SubjectID: S01 → S05
 - **Class**. ClassID: C01 → C03
 - **Student**. StudentID: T01 → T20
 - **StudentGrades**. Distribute grades of subject to the students. There are one to three subjects for each student. Only one half students have grades.
5. **Query by Relational Algebra and SQL:**
 - 5.1. Show Students of class ID = "C02".
 - 5.2. Show Students of class name = "Computer Science".
 - 5.3. Show Students (All information) of class year = "2020-2024".
 - 5.4. Show Subject name and units of the Subject ID = "S01".
 - 5.5. Grades of Subject ID = "S02" of Student ID = "T02".

- 5.6. Find Subject (ID, Name and Grades) that Student ID = "T02" fail.
 - 5.7. Show all the Subject (*) that Student ID = "T03" never took the exam.
 - 5.8. Number of Students for each class.
 - 5.9. Find the classes with the largest number of students.
 - 5.10. GPA (grade point average) of student ID = "T02".
 - 5.11. GPA for each student.
 - 5.12. GPA of class ID = "C02".
 - 5.13. GPA for each class.
 - 5.14. Find students have the largest GPA.
 - 5.15. Find students (ID and Name) have the largest GPA.
 - 5.16. Find classes (ID and Name) have the largest GPA.
 - 5.17. GPA with weight for each student.
 - 5.18. GPA with weight for each student (ID and name).
 - 5.19. GPA with weight for each class.
6. Show all integrity constraints.
 7. Thinter GUI for this database.

2 Retail Invoice Database (RIDB)

1. **Category (CategoryID, CategoryName)**
Predicate: Each category (**Category**) has a certain code (**CategoryID**) to distinguish it from other categories; We know the category name (**CategoryName**) for that category.
2. **Product (ProductID, ProductName, UnitPrice, CategoryID)**
Predicate: Each product (**Product**) has a unique code (**ProductID**) to distinguish it from other products; we know the product name (**ProductName**), unit price (**UnitPrice**) and the category of the product (**CategoryID**).
3. **Invoice (InvoiceID, InvoiceDate, Description)**
Predicate: Each invoice (**Invoice**) has a unique code (**InvoiceID**) to distinguish it from other invoices; We know the date of create invoice (**InvoiceDate**), and description of that invoice (**Description**).
4. **InvoiceDetail (InvoiceID, ProductID, Quantity)**
Predicate: Invoice Detail relational scheme (**InvoiceDetail**) store the quantity (**Quantity**) of the invoice (**InvoiceID**) and the product (**ProductID**).

Require

1. Determine all the keys of the Relational Schemes.
2. Create database **RIDB**.
3. Create the Relational Schemes.
4. Insert data:
 - **Category**. CategoryID: C01 → C05
 - **Product**. ProductID: P01 → P30
 - **Invoice**. InvoiceID: I01 → I10
 - **InvoiceDetail**. Distribute product to the invoice. There are two to five products for each invoice.
5. **Query by Relational Algebra and SQL:**
 - 5.1. Find products of the category ID = "C01".
 - 5.2. Find products (ID, name and price) of the category ID = "C02".
 - 5.3. Find products (*) with unit price from 10 to 50.
 - 5.4. Show invoices, it created at date = d.
 - 5.5. Show invoices, it created on year = 2021.
 - 5.6. Find products (ID, name, unit price and quantity), it belong to the invoice at date = d.
 - 5.7. Total quantity of each invoice.
 - 5.8. Total quantity of each invoice in date = d.
 - 5.9. Total cost (= quantity times to unit price) of each invoice.
 - 5.10. With invoices have the largest total quantity.
 - 5.11. In date = d, with invoices have the largest total quantity.
 - 5.12. With invoices have the largest total cost.
 - 5.13. In date = d, with invoices have the largest total cost.
 - 5.14. Find years have the largest total cost.
6. Show all integrity constraints.
7. TKinter GUI for this database.

3 Warehouse Management Database (WMDB)

1. **Category** (**CategoryID**, **CategoryName**)
Predicate: Each category (**Category**) has a certain code (**CategoryID**) to distinguish it from other categories; We know the category name (**CategoryName**) for that category.
2. **Product** (**ProductID**, **ProductName**, **UnitPrice**, **CategoryID**)
Predicate: Each product (**Product**) has a unique code (**ProductID**) to distinguish it from other products; we know the product name (**ProductName**), unit price (**UnitPrice**) and the category of the product (**CategoryID**).
3. **Warehouse** (**WarehouseID**, **WarehouseAddress**, **CategoryID**)
Predicate: Each warehouse (**Warehouse**) has a unique code (**WarehouseID**) to distinguish it from other warehouses; We know the address of warehouse (**WarehouseAddress**). **Each warehouse is only store one category** (**CategoryID**).
4. **Instock** (**WarehouseID**, **ProductID**, **Quantity**)
Predicate: Instock relational scheme (**Instock**) store the quantity (**Quantity**) of the product (**ProductID**) in the warehouse (**WarehouseID**).

Require

1. Determine all the keys of the Relational Schemes.
2. Create database **WMDB**.
3. Create the Relational Schemes.
4. Insert all the required data for queries and integrity constraints.
5. **Query by Relational Algebra and SQL:**
 - 5.1. All the products of category ID = "C02".
 - 5.2. All the warehouses (*) that store category ID = "C01".
 - 5.3. All the warehouses (*) in now store product name = "beverage".
 - 5.4. All the products, it can be store in warehouse ID = "W01".
 - 5.5. Calculating sum of quantity for each warehouses.
 - 5.6. Find warehouse have the largest total quantities.
 - 5.7. Calculating count of product for each warehouse.
 - 5.8. Find warehouses have the largest number of product.
 - 5.9. Calculating sum of quantity for each product.
 - 5.10. Show products have the largest total of quantities.
6. Show all integrity constraints.
7. Tkinter GUI for this database.

4 Order Management Database (OMDB)

1. **Category (CategoryID, CategoryName)**
Predicate: Each category (**Category**) has a certain code (**CategoryID**) to distinguish it from other categories; We know the category name (**CategoryName**) of the category.
2. **Product (ProductID, ProductName, UnitPrice, CategoryID)**
Predicate: Each product (**Product**) has a unique code (**ProductID**) to distinguish it from other products; we know the product name (**ProductName**), unit price (**UnitPrice**) and the category of the product (**CategoryID**).
3. **Customer (CustomerID, CustomerName, CustomerAddress)**
Predicate: Each customer (**Customer**) has a unique code (**CustomerID**) to distinguish it from other customers; We know name (**CustomerName**) and address (**CustomerAddress**) of the customer.
4. **Order (OrderID, OrderDate, RequiredDate, CustomerID)**
Predicate: Each order (**Order**) has a unique code (**OrderID**) to distinguish it from other orders; We know order date (**OrderDate**), required date (**RequiredDate**) and customer (**CustomerID**) who took the order.
5. **OrderDetail (OrderID, ProductID, OrderQuantity)**
Predicate: Order detail relational scheme (**OrderDetail**) store the quantity (**OrderQuantity**) of the products (**ProductID**) in the order (**OrderID**).
6. **Delivery (DeliveryID, DeliveryDate, OrderID)**
Predicate: Each delivery (**Delivery**) has a unique code (**DeliveryID**) to distinguish it from other deliveries; We know delivery date (**DeliveryDate**) and order (**OrderID**) it is delivered.
7. **DeliveryDetail (DeliveryID, ProductID, DeliveryQuantity)**
Predicate: Delivery detail relational scheme (**DeliveryDetail**) store the quantity (**DeliveryQuantity**) of the products (**ProductID**) in the delivery (**DeliveryID**).

Require

1. Determine all the keys of the Relational Schemes.
2. Create database **OMDB**.
3. Create the Relational Schemes.
4. Insert all the required data for queries and integrity constraints.
5. **Query by Relational Algebra and SQL:**
 - 5.1. All the products of category ID = "C02".
 - 5.2. List of customers who took order with date from d1 to d2.
 - 5.3. List of customers (ID, name, address) who took order in year = 2021.
 - 5.4. List of products (ID) ordered in order ID = "O01".
 - 5.5. List of products (*) ordered in order ID = "O01".
 - 5.6. List of products (*) ordered in order date = d.
 - 5.7. Calculating total of quantities for each order (ID).
 - 5.8. Calculating total of quantities for each order (ID), it took in year = 2021.
 - 5.9. With orders (ID) have the largest total cost.
 - 5.10. In year = 2021, with orders (ID) have the most total cost.
 - 5.11. Calculating total cost of orders for each customer.
 - 5.12. With customers (ID) have the largest total cost.
 - 5.13. Calculating total cost of orders for each customer (ID, name).
 - 5.14. In year = 2021, Calculating total cost of orders for each customer (ID, name).
 - 5.15. In year = 2021, customers (ID, name, address) with the largest total cost.
6. Show all integrity constraints.
7. Tkinter GUI for this database.

5 Vietnam Geographic Database (VGDB)

1. **Country (CountryID, CountryName)**

Predicate: Each country (**Country**) has a certain code (**CountryID**) to distinguish it from other countries; We know the country name (**CountryName**) for the country.

2. **Province (ProvinceID, ProvinceName, Population, Area, CountryID)**

Predicate: Each province (**Province**) has a unique code (**ProvinceID**) to distinguish it from other provinces; we know the province name (**ProvinceName**), population (**Population**), area (**Area**) and the country (**CountryID**) of the province.

3. **Border (ProvinceID, NationID)**

Predicate: Border relational scheme (**Border**) store the border of province (**ProvinceID**) and nations (**NationID**).

4. **Neighbor (ProvinceID, NeighborID)**

Predicate: Neighbor relational scheme (**Neighbor**) store neighbor of province (**ProvinceID**) with other province (**NeighborID**).

Require

1. Determine all the keys of the Relational Schemes.
2. Create database **VGDB**.
3. Create the Relational Schemes.
4. Insert all the required data for queries and integrity constraints.
5. **Query by Relational Algebra and SQL:**
 - 5.1. Provinces with an area larger than 15000 square kilometers.
 - 5.2. Provinces(*) it neighbored with province have area larger than 15000 square kilometers.
 - 5.3. Provinces (*) in the country name = "North".
 - 5.4. Which Nation bordering the northern provinces.
 - 5.5. Average area of the southern provinces.
 - 5.6. Population density of the central country.
 - 5.7. Provinces with the largest population density.
 - 5.8. Provinces with the largest area.
 - 5.9. In southern country, provinces with the largest area.
 - 5.10. Provinces that have borders with two or more nations.
 - 5.11. List of Countries with the number of its provinces.
 - 5.12. Provinces with the largest total neighbor.
 - 5.13. Provinces that are area larger than area of their neighboring provinces.
 - 5.14. For each country, list the provinces with largest area.
 - 5.15. For each country, list the provinces with population larger than the average population of that country.
 - 5.16. Countries with the largest total area.
 - 5.17. Countries with the largest total population.
6. Show all integrity constraints.
7. Tkinter GUI for this database.