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

Contents

0 Database Management Systems (DBMS)

0.1 PostgreSQL

1. Server: https://www.postgresql.org
2. IDE: https://www.pgadmin.org

0.2 MySQL

1. Server: https://dev.mysql.com/downloads/installer/
2. IDE: https://www.mysql.com/products/workbench/

0.3 SQLServer

1. Server: https://www.microsoft.com/en-us/sql-server/sql-server-downloads
2. IDE: SSMS

0.4 SQLite

1. Server: https://sqlite.org/
2. IDE: https://sqlitestudio.pl/

1 Student Management Database (SMDB)

1. **Subject (SubjectID, SubjectName, Units)**
   Predicate: Each subject **(Subject)** is assigned a unique code **(SubjectID)** to distinguish it from other subjects. The subject name **(SubjectName)** and the number of units **(Units)** for that subject are known..
2. **Class (ClassID, ClassName, ClassYear)**
   Predicate: Each class **(Class)** is assigned a unique code **(ClassID)** to distinguish it from other classes. The class name **(ClassName)** and the class year **(ClassYear)** are known.
3. **Student (StudentID, StudentName, StudentAddress, ClassID)**
   Predicate: Every student **(Student)** has a unique code **(StudentID)** that differentiates them from other students. The student's name **(StudentName)**, address **(StudentAddress)**, and class **(ClassID)** are recorded.
4. **StudentGrades (StudentID, SubjectID, Grades)**
   Predicate: The Student Grades relational schema **(StudentGrades)** records the grades **(Grades)** of students **(StudentID)** for subjects **(SubjectID)**.

Require

1. Find all keys of the relational schemas.
2. Create database **DB01**.
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. **Performing Queries with Relational Algebra and SQL:**
   5.1. Find the students that belong to class ID "C02".
   5.2. Find the students that belong to class name = "Computer Science".
   5.3. Find the students (All information) whose class year is "2023"
   5.4. Find the Subject (name and units) of the Subject ID "S01".
   5.5. What are the grades of student "T02" for subject "S02"?

5.6. Find the subjects (ID, Name, and Grades) in which student ”T02” failed.
5.7. Which subjects (*) did student ”T03” never take the exam for?
5.8. How many students are in each class?
5.9. Find the classes with the largest number of students.
5.10. Calculate the GPA (grade point average) of student ID ”T02”.
5.11. Calculate the GPA for each student.
5.12. Calculate the GPA of class ID ”C02”.
5.13. Calculate the GPA for each class.
5.14. Find the students who have the largest GPA.
5.15. Find the students (ID and Name) who have the largest GPA.
5.16. Find the classes that have the largest average GPA.
5.17. Find the classes (ID and Name) that have the largest average GPA.
5.18. Calculate the GPA with weights for each student.
5.19. Weighted GPA calculation for each student (ID and name).
5.20. Which students have received a grade in all subjects?
5.21. Which students have received a grade in all subjects where each subject has 2 units?
5.22. Which students have passed (grade >= 5) in all subjects where each subject has 2 units?

6. All integrity constraints.

2 Retail Invoice Database (RIDB)

1. **Category (CategoryID, CategoryName)**
   Predicate: Each category **(Category)** is assigned a unique code **(CategoryID)** to distinguish it from other categories. The corresponding category name is known as **(CategoryName)**.
2. **Product (ProductID, ProductName, UnitPrice, CategoryID)**
   Predicate: Every product **(Product)** has a unique code **(ProductID)** that differentiates it from other products. The product's name **(ProductName)**, unit price **(UnitPrice)**, and category **(CategoryID)** are recorded.
3. **Invoice (InvoiceID, InvoiceDate, Description)**
   Predicate: Each invoice **(Invoice)** is assigned a unique code **(InvoiceID)** to distinguish it from other invoices. The invoice creation date **(InvoiceDate)** and a description **(Description)** are recorded.
4. **InvoiceDetail (InvoiceID, ProductID, Quantity)**
   Predicate: The Invoice Detail relational schema **(InvoiceDetail)** stores the quantity **(Quantity)** of each product **(ProductID)** included in an invoice **(InvoiceID)**.

Require

1. Find all keys of the relational schemas.
2. Create database **DB02**.
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. **Performing Queries with Relational Algebra and SQL:**
   5.1. Find the products that belong to category ID "CO1".
   5.2. Find the products (ID, name and price) that belong to category ID "CO2".
   5.3. Find the products (*) with a unit price between 10 and 50.
   5.4. Which invoices were created on date d?
   5.5. Show invoices that were created on year 2025.
   5.6. Find the products (ID, name, unit price and quantity), it belong to the invoice on date d.
   5.7. List the total quantity of each invoice.
   5.8. List the total quantity of each invoice on date d.
   5.9. Calculate the total Cost (quantity times unit price) for each invoice.
   5.10. Calculate the total Cost for each year.
   5.11. What is the total Cost for each quarter in 2023?
   5.12. List the invoices which have the largest total quantity.
   5.13. Show the invoices that have the largest total quantity
   5.14. What invoices on date d have the largest total quantity?.
   5.15. What invoices on date d have the largest total Cost?
   5.16. Find the years with the largest total Cost.
   5.17. Which invoice contains all products?
   5.18. Which invoice in 2023 contains all products?
6. Show all the integrity constraints.

3 Warehouse Management Database (WMDB)

1. **Category (CategoryID, CategoryName)**
   Predicate: Each category **(Category)** is assigned a unique code **(CategoryID)** to distinguish it from other categories. The category name **(CategoryName)** is known.
2. **Product (ProductID, ProductName, UnitPrice, CategoryID)**
   Predicate: Every product **(Product)** has a unique code **(ProductID)** that differentiates it from other products. The product name **(ProductName)**, unit price **(UnitPrice)**, and category **(CategoryID)** are recorded.
3. **Warehouse (WarehouseID, WarehouseAddress, CategoryID)**
   Predicate: Each warehouse **(Warehouse)** is assigned a unique code **(WarehouseID)** to distinguish it from other warehouses. The warehouse address **(WarehouseAddress)** is known. Each warehouse stores only one category **(CategoryID)**.
4. **Instock (WarehouseID, ProductID, Quantity)**
   Predicate: The Instock relational schema **(Instock)** stores the quantity **(Quantity)** of each product **(ProductID)** in a warehouse **(WarehouseID)**.

Require

1. Find all keys of the relational schemas.
2. Create database **DB03**.
3. Create the Relational Schemes.
4. Insert all the required data for queries and integrity constraints.
5. **Performing Queries with Relational Algebra and SQL:**
   5.1. Find the the products that belong to category ID ”C02”.
   5.2. Find the warehouses (*) that store category ID ”C01”.
   5.3. List the warehouses (*) that are currently storing products with the name ’beverage’.
   5.4. List all the products which can be stored in warehouse ID ’W01’.
   5.5. Calculate the total quantity for each warehouse.
   5.6. Find the warehouse which has the largest total quantity.
   5.7. Computing the number of products in each warehouse.
   5.8. Find the warehouses that have the largest number of products.
   5.9. Calculating the total quantity for each product.
   5.10. Show the products with the largest total quantity.
6. Show all the integrity constraints.

4 Order Management Database (OMDB)

1. **Category (CategoryID, CategoryName)**
   Predicate: Each category **(Category)** is assigned a unique code **(CategoryID)** to distinguish it from other categories. The category name **(CategoryName)** is known.
2. **Product (ProductID, ProductName, UnitPrice, CategoryID)**
   Predicate: Every product **(Product)** has a unique code **(ProductID)** that differentiates it from other products. The product name **(ProductName)**, unit price **(UnitPrice)**, and category **(CategoryID)** are recorded.
3. **Customer (CustomerID, CustomerName, CustomerAddress)**
   Predicate: Each customer **(Customer)** is assigned a unique code **(CustomerID)** to distinguish them from other customers. The customer's name **(CustomerName)** and address **(CustomerAddress)** are known.
4. **Order (OrderID, OrderDate, RequiredDate, CustomerID)**
   Predicate:Every order **(Order)** has a unique code **(OrderID)** that differentiates it from other orders. The order date **(OrderDate)**, required date **(RequiredDate)**, and the customer **(CustomerID)** who placed the order are recorded.
5. **OrderDetail (OrderID, ProductID, OrderQuantity)**
   Predicate:The Order Detail relational schema **(OrderDetail)** stores the quantity **(OrderQuantity)** of each product **(ProductID)** in an order **(OrderID)**.
6. **Delivery (DeliveryID, DeliveryDate, OrderID)**
   Predicate:Each delivery **(Delivery)** is assigned a unique code **(DeliveryID)** to distinguish it from other deliveries. The delivery date **(DeliveryDate)** and the order **(OrderID)** it fulfills are known.
7. **DeliveryDetail (DeliveryID, ProductID, DeliveryQuantity)**
   Predicate: The Delivery Detail relational schema **(DeliveryDetail)** stores the quantity **(DeliveryQuantity)** of each product **(ProductID)** in a delivery **(DeliveryID)**.

Require

1. Find all keys of the relational schemas.
2. Create database **DB04**.
3. Create the Relational Schemes.
4. Insert all the required data for queries and integrity constraints.
5. **Performing Queries with Relational Algebra and SQL:**
   - 5.1. All products in category ID 'C02'
   - 5.2. List of customers who placed orders between d1 and d2.
   - 5.3. List of customers (ID, name, address) who placed orders in year 2025.
   - 5.4. List of products (ID) ordered in order ID 'O01'.
   - 5.5. List all product for order 'O01'.
   - 5.6. List all product (*) for orders on date 'd'.
   - 5.7. Calculating total quantities for each order (ID).
   - 5.8. Calculating the total quantity for each order (ID) in year 2025.
   - 5.9. Identify orders (ID) with the largest total Cost.
   - 5.10. In 2025, orders (ID) that had the highest total Cost.
   - 5.11. Computing the total Cost of orders for each customer.
   - 5.12. Identify customers (ID) with the largest total Cost.
   - 5.13. Calculating the total quantity for each customer (ID, name).
   - 5.14. In 2025, the total Cost of orders was calculated for each customer (ID, name).
   - 5.15. In 2025, customers (ID, name, address) with the highest total Cost of orders.
6. Show all the integrity constraints.

5 Vietnam Geographic Database (VGDB)

1. **Country (CountryID, CountryName)**
   Predicate: Each country **(Country)** is assigned a unique code **(CountryID)** to distinguish it from other countries. The country name **(CountryName)** is known.

2. **Province (ProvinceID, ProvinceName, Population, Area, CountryID)**
   Predicate: Each province **(Province)** is assigned a unique code **(ProvinceID)** to distinguish it from other provinces. The province's name **(ProvinceName)**, population **(Population)**, area **(Area)**, and country **(CountryID)** are known.

3. **Border (ProvinceID, NationID)**
   Predicate: The Border relational schema **(Border)** stores the border relationships between provinces **(ProvinceID)** and nations **(NationID)**.

4. **Neighbor (ProvinceID, NeighborID)**
   Predicate: The Neighbor relational schema **(Neighbor)** stores the neighbor relationships between provinces **(ProvinceID)** and neighboring provinces **(NeighborID)**.

Require

1. Find all keys of the relational schemas.
2. Create database **DB05**.
3. Create the Relational Schemes.
4. Insert all the required data for queries and integrity constraints.
5. **Performing Queries with Relational Algebra and SQL:**
   5.1. Provinces having an area greater than 15,000 square kilometers.
   5.2. Provinces (*) that neighbor provinces with an area larger than 15,000 square kilometers.
   5.3. Provinces (*) within the country 'North'?
   5.4. Which nation borders the northern provinces?
   5.5. Calculate the average area of the southern provinces.
   5.6. Calculate the population density of the central country.
   5.7. Identify the provinces with the highest population density.
   5.8. Which provinces have the greatest area?
   5.9. In the southern country, provinces with the largest area.
   5.10. Provinces that share borders with two or more nations.
   5.11. List of countries, showing the number of provinces each has.
   5.12. Provinces with the greatest number of neighboring provinces.
   5.13. Provinces whose area is larger than the area of their neighboring provinces.
   5.14. For each country, list the provinces with the largest areas.
   5.15. For each country, list the provinces with a population larger than the country's average population.
   5.16. Countries with the greatest total area.
   5.17. Countries with the largest total population.
6. Show all the integrity constraints.