**CLAUDIO MASCIOCCHI**

**Questions**

1. Cosa si intende per database?
2. Cos’è un DBMS?
3. Indica le principali clausole di uno statement SELECT in ordine di esecuzione logica. Descrivi per ciascuna delle clausole indicate la logica di funzionamento.
4. Descrivi, immaginando uno scenario a te familiare, il concetto di group by. Utilizza l’approccio che ritieni più efficiente per trasmettere il concetto (suggerimento: disegna anche una sola tabella in Excel o in word con poche colonne e pochi record e descrivi, basandosi sulla tabella stessa, un esempio di group by).
5. Descrivi la differenza tra uno schema OLTP e uno schema OLAP.
6. Dato un medesimo scenario di analisi, qual è la differenza in termini di risultato ottenibile tra una join e una subquery?
7. Cosa si intende per DML e DDL?
8. Quali istruzioni possono utilizzare per estrarre l’anno da un campo data? Proponi degli esempi.
9. Qual è la differenza tra gli operatori logici AND e OR?
10. È possibile innestare una query nella clausola SELECT?
11. Qual è la differenza tra l’operatore logico OR e l’operatore logico IN?
12. L’operatore logico BETWEEN include anche gli estremi del range specificato?

**Risposte**

**1)**

Con il termine database si intende una collezione di dati organizzata in modo logico e coerente per la gestione delle operazioni di creazione/inserimento, lettura, modifica o aggiornamento ed eliminazione di dati, ovvero le cosiddette operazioni CRUD (Create, Read, Update, Delete).

Detto ciò, esistono diverse tipologie di database, il modello più utilizzato è quello relazionale in cui i dati sono organizzati in tabelle.

**2)**

Un DBMS (acronimo di Database Management System) è un software che permette di creare, gestire, manipolare e interrogare database in modo efficiente, sicuro e strutturato. In poche parole, un DBMS è l’interfaccia tra l’utente e il database, che consente di gestire grandi quantità di dati in modo organizzato e sicuro. MySQL è un esempio di DMBS (relazionale).

**3)**

Le principali clausole di uno statement SELECT, in ordine di esecuzione, sono:

1. FROM: consente di individuare la/le tabelle (o le view) dalle quali prelevare i dati. Eventualmente esegue JOIN tra più tabelle.
2. WHERE: consente di filtrare le righe restituite dalla clausola FROM in base a una o più condizioni di ricerca. Quindi seleziona solo le righe che soddisfano una determinata condizione.
3. GROUP BY: consente di raggruppare i record restituiti dagli step precedenti per ogni combinazione univoca dei campi indicati nella GROUP BY list.
4. HAVING: consente di filtrare i gruppi restituiti dalla GROUP BY in base a una o più condizione di ricerca. Si usa principalmente per filtrare i gruppi in base a valori aggregati (es. COUNT, SUM, AVG, ecc.), anche se non si limita solo a questi.
5. SELECT: definisce i campi che devono essere restituiti nel risultato finale, cioè nell’output della query.
6. ORDER BY: consente di ordinare i risultati secondo una o più colonne. Es. impostando l’ordine crescente ASC (che è di default) o decrescente DESC dei risultati.

**4)**

Per fare un esempio in ambito calcistico, supponiamo di voler calcolare il numero di goal segnati da ciascuna squadra di Serie A:

Immagine che contiene testo, schermata, numero, Carattere

Il contenuto generato dall'IA potrebbe non essere corretto.

Se questa fosse la tabella Partite scriveremmo:

SELECT

Squadra,

SUM(Goal\_segnati) AS TotaleGoal

FROM Partite

GROUP BY Squadra

;

In questo esempio molto semplificato il group by raggruppa tutte le righe per “Squadra” (tutte le partite giocate dal Napoli, dal Milan, dalla Roma e così via) e applica la funzione SUM su ogni gruppo. In questo modo sommerà i goal segnati da ogni squadra in tutte le partite. Es:

Squadra TotaleGoal

Napoli 5

Milan 4

Roma 1

**5)**

La differenza tra uno schema OLTP e uno schema OLAP riguarda il modo in cui i dati vengono organizzati e utilizzati in due tipi diversi di sistemi informatici, infatti:

* I sistemi OLTP (OnLine Transactional Processing) sono tipici di scenari con intensa attività di scrittura quindi transazioni e operazioni CRUD brevi ma intense, tipiche nei sistemi gestionali. Lo scopo di un sistema OLTP è garantire la consistenza, l’integrità e la sicurezza di queste transazioni.

Il sistema OLTP è ottimizzato per la gestione dei dati ma non per l’analisi dei dati (reporting).

* I sistemi OLAP (OnLine Analytical Processing) invece sono sistemi per l’analisi di grandi volumi di dati a supporto della Business Intelligence, che costituisce il cuore del data warehouse.

**6)**

La JOIN combina righe da più tabelle basandosi su una condizione comune mentre una subquery è una query annidata dentro ad un’altra. A volte possono portare allo stesso risultato ma in molti altri casi la JOIN può escludere delle righe se non ci sono corrispondenze, a meno che non si usi una LEFT o RIGHT JOIN.

**7)**

Sono due categorie di comandi SQL:

DDL sta per Data Definition Language e comprende comandi utilizzati per definire e gestire la struttura di un database come CREATE, DROP, ALTER, TRUNCATE e RENAME.

DML invece sta per Data Manipulation Language e comprende comandi per interagire con i dati all’interno del DB, es. INSERT, UPDATE, DELETE, CALL, LOCK.

**8)**

La funzione YEAR è sicuramente il metodo migliore che mi viene in mente per estrarre l’anno da un campo di tipo DATE o DATETIME.

Se ad esempio volessimo sapere in quale anno si sono registrati dei clienti (in una ipotetica tabella “Clienti”) potremmo utilizzare la funzione YEAR in questo modo:

SELECT

Nome,

Cognome,

YEAR(DataRegistrazione) AS AnnoRegistrazione

FROM Clienti

;

In questo modo andremmo ad estrarre l’anno dalla colonna DataRegistrazione, dandoci un alias “parlante” come AnnoRegistrazione.

**9)**

La differenza tra gli operatori logici AND e OR sta nel fatto che nel primo TUTTE le condizioni devono essere vere, mentre nel secondo basta che UNA condizione sia vera. Quindi in AND la riga viene selezionata solo se tutte le condizioni specificate risultano essere vere, mentre in OR la riga viene selezionata se almeno una delle condizioni specificate risulti essere vera.

**10)**

Si, è possibile. Si tratta di una Subquery ed è utile se si vuole calcolare un valore specifico per ogni riga.

Es. utilizzo di una subquery dentro la SELECT per calcolare la somma degli importi degli ordini per ogni cliente.

Tabella 1: Clienti (ClienteID, Nome, Cognome),

Tabella 2: Ordini (OrdineID, ClienteID, Importo)

SELECT

Nome,

Cognome,

(SELECT SUM(Importo)

FROM Ordini

WHERE Ordini.ClienteID = Clienti.ClienteID) AS TotaleOrdini

FROM Clienti

;

**11)**

L’operatore IN consente di ricercare la corrispondenza in una lista di valori specificati (Es.: WHERE Color IN (‘Red’, ‘Blue’, ‘Grey’) mentre l’operatore OR confronta una colonna con un elenco di valori specifici (Es.: WHERE Color = ‘Red’ OR Color = ‘Blue’ OR Color =’Grey’)

Quindi si può dire che entrambi vengono usati per confrontare un valore con più possibili alternative, OR confronta una condizione alla volta mentre IN confronta un valore con una lista.

**12)**

Sì, BETWEEN include anche gli estremi del range, es.:

WHERE Età BETWEEN 20 AND 30

È come scrivere WHERE Età >= 20 AND Età <= 30

20 e 30 sono inclusi.

**Task 1: Proponi una progettazione concettuale e logica della base dati**

**Immagine che contiene testo, diagramma, Piano, Disegno tecnico

Il contenuto generato dall'IA potrebbe non essere corretto.**

**Immagine che contiene testo, schermata, diagramma, Carattere

Il contenuto generato dall'IA potrebbe non essere corretto.**

**Immagine che contiene testo, schermata, numero, Parallelo

Il contenuto generato dall'IA potrebbe non essere corretto.**

Per il task 1 ho utilizzato Diagrams.net (prima immagine), Reverse Engineer in MySQL Workbench (seconda foto) e infine ho inserito tutti i dati in delle tabelle Excel, in modo da collegarli a mano e commentare le relazioni e le cardinalità accanto.

**Task 2: Descrivi la struttura delle tabelle che reputi utili e sufficienti a modellare lo scenario proposto tramite la sintassi DDL. Implementa fisicamente le tabelle utilizzando il DBMS SQL Server(o altro).**

Questo è come ho creato il DB in MySQL Workbench:

CREATE DATABASE ToysGroup;

USE ToysGroup;

CREATE TABLE Category (

CategoryID INT PRIMARY KEY AUTO\_INCREMENT,

CategoryName VARCHAR(100) NOT NULL

);

CREATE TABLE Product (

ProductID INT PRIMARY KEY AUTO\_INCREMENT,

ProductName VARCHAR(100) NOT NULL,

Model VARCHAR(50),

LaunchYear YEAR,

CategoryID INT NOT NULL,

FOREIGN KEY (CategoryID) REFERENCES Category(CategoryID)

);

CREATE TABLE Region (

RegionID INT PRIMARY KEY AUTO\_INCREMENT,

RegionName VARCHAR(100) NOT NULL

);

CREATE TABLE Country (

CountryID INT PRIMARY KEY AUTO\_INCREMENT,

CountryName VARCHAR(100) NOT NULL,

ISOCode CHAR(3),

RegionID INT NOT NULL,

FOREIGN KEY (RegionID) REFERENCES Region(RegionID)

);

CREATE TABLE Sales (

SaleID INT PRIMARY KEY AUTO\_INCREMENT,

SaleDate DATE NOT NULL,

Quantity INT NOT NULL,

UnitPrice DECIMAL(10,2) NOT NULL,

ProductID INT NOT NULL,

CountryID INT NOT NULL,

FOREIGN KEY (ProductID) REFERENCES Product(ProductID),

FOREIGN KEY (CountryID) REFERENCES Country(CountryID)

);

**Task 3: Popola le tabelle utilizzando dati a tua discrezione (sono sufficienti pochi record per tabella; riporta le query utilizzate).**

Di seguito è come ho popolato il mio database in MySQL:

INSERT INTO Category (CategoryID, CategoryName) VALUES

(1, 'Bikes'),

(2, 'Clothing'),

(3, 'Accessories');

INSERT INTO Product (ProductID, ProductName, Model, LaunchYear, CategoryID) VALUES

(1, 'Bike-100', 'B100-X', 2021, 1),

(2, 'Bike-200', 'B200-Z', 2022, 1),

(3, 'Bike Glove M', 'GL-M', 2023, 2),

(4, 'Bike Glove L', 'GL-L', 2023, 2),

(5, 'Helmet Pro', 'HP-1', 2020, 3),

(6, 'Water Bottle', 'WB-700', 2021, 3);

INSERT INTO Region (RegionID, RegionName) VALUES

(1, 'WestEurope'),

(2, 'SouthEurope'),

(3, 'NorthAmerica');

INSERT INTO Country (CountryID, CountryName, ISOCode, RegionID) VALUES

(1, 'France', 'FRA', 1),

(2, 'Germany', 'DEU', 1),

(3, 'Italy', 'ITA', 2),

(4, 'Greece', 'GRC', 2),

(5, 'USA', 'USA', 3),

(6, 'Canada', 'CAN', 3),

(7, 'Spain', 'ESP', 2),

(8, 'Netherlands', 'NLD', 1);

INSERT INTO Sales (SaleID, ProductID, CountryID, SaleDate, Quantity, UnitPrice) VALUES

(1 , 1, 1, '2024-01-10', 5, 250.00),

(2 , 2, 2, '2024-01-15', 3, 450.00),

(3 , 3, 3, '2024-01-20', 10, 20.00),

(4 , 4, 4, '2024-02-01', 8, 25.00),

(5 , 5, 5, '2024-02-10', 2, 60.00),

(6 , 6, 6, '2024-02-18', 6, 10.00),

(7 , 3, 1, '2024-03-05', 4, 20.00),

(8 , 4, 7, '2024-03-10', 9, 25.00),

(9 , 5, 8, '2024-03-15', 1, 65.00),

(10, 6, 5, '2024-04-01', 10, 9.50),

(11, 2, 2, '2024-04-15', 2, 450.00),

(12, 1, 3, '2024-04-20', 1, 250.00);

#controllo le tabelle

SELECT \* FROM Category;

SELECT \* FROM Product;

SELECT \* FROM Region;

SELECT \* FROM Country;

SELECT \* FROM Sales;

**Task 4: Dopo aver popolate le tabelle, scrivi delle query utili a:**

1. Verificare che i campi definiti come PK siano univoci. In altre parole, scrivi una query per determinare l’univocità dei valori di ciascuna PK (una query per tabella implementata).
2. Esporre l’elenco delle transazioni indicando nel result set il codice documento, la data, il nome del prodotto, la categoria del prodotto, il nome dello stato, il nome della regione di vendita e un campo booleano valorizzato in base alla condizione che siano passati più di 180 giorni dalla data vendita o meno (>180 -> True, <= 180 -> False)
3. Esporre l’elenco dei prodotti che hanno venduto, in totale, una quantità maggiore della media delle vendite realizzate nell’ultimo anno censito. (ogni valore della condizione deve risultare da una query e non deve essere inserito a mano). Nel result set devono comparire solo il codice prodotto e il totale venduto.
4. Esporre l’elenco dei soli prodotti venduti e per ognuno di questi il fatturato totale per anno.
5. Esporre il fatturato totale per stato per anno. Ordina il risultato per data e per fatturato decrescente.
6. Rispondere alla seguente domanda: qual è la categoria di articoli maggiormente richiesta dal mercato?
7. Rispondere alla seguente domanda: quali sono i prodotti invenduti? Proponi due approcci risolutivi differenti.
8. Creare una vista sui prodotti in modo tale da esporre una “versione denormalizzata” delle informazioni utili (codice prodotto, nome prodotto, nome categoria)
9. Creare una vista per le informazioni geografiche

**1) Verificare che i campi definiti come PK siano univoci. In altre parole, scrivi una query per determinare l’univocità dei valori di ciascuna PK (una query per tabella implementata).**

SELECT CategoryID,

COUNT(\*) AS Duplicati

FROM Category

GROUP BY CategoryID

HAVING COUNT(\*) > 1;

SELECT ProductID,

COUNT(\*) AS Duplicati

FROM Product

GROUP BY ProductID

HAVING COUNT(\*) > 1;

SELECT RegionID,

COUNT(\*) AS Duplicati

FROM Region

GROUP BY RegionID

HAVING COUNT(\*) > 1;

SELECT CountryID,

COUNT(\*) AS Duplicati

FROM Country

GROUP BY CountryID

HAVING COUNT(\*) > 1;

SELECT SaleID,

COUNT(\*) AS Duplicati

FROM Sales

GROUP BY SaleID

HAVING COUNT(\*) > 1;

Queste sono le query che ho inserito in MySQL, in questo modo posso controllare se ogni chiave primaria delle 5 tabelle è effettivamente univoca, quindi senza avere più righe con lo stesso valore.

Il risultato è stato positivo in quanto tutte e cinque le query non hanno dato risultato, ciò significa che le chiavi primarie sono risultate essere tutte univoche.

**2) Esporre l’elenco delle transazioni indicando nel result set il codice documento, la data, il nome del prodotto, la categoria del prodotto, il nome dello stato, il nome della regione di vendita e un campo booleano valorizzato in base alla condizione che siano passati più di 180 giorni dalla data vendita o meno (>180 -> True, <= 180 -> False).**

SELECT

s.SaleID,

s.SaleDate,

p.ProductName,

c.CategoryName,

co.CountryName,

r.RegionName,

DATEDIFF(CURDATE(), s.SaleDate) > 180 AS Oltre180Giorni

FROM Sales s

JOIN Product p

ON s.ProductID = p.ProductID

JOIN Category c

ON p.CategoryID = c.CategoryID

JOIN Country co

ON s.CountryID = co.CountryID

JOIN Region r

ON co.RegionID = r.RegionID

ORDER BY s.SaleDate

;

Non avendo studiato il campo booleano ho usato questa query, dovrebbe essere molto simile in quanto DATEDIFF(...) > 180 restituisce 1 (vero) o 0 (falso), che corrispondono a TRUE/FALSE.

Immagine che contiene testo, schermata, numero, Carattere

Il contenuto generato dall'IA potrebbe non essere corretto.

**3) Esporre l’elenco dei prodotti che hanno venduto, in totale, una quantità maggiore della media delle vendite realizzate nell’ultimo anno censito. (ogni valore della condizione deve risultare da una query e non deve essere inserito a mano). Nel result set devono comparire solo il codice prodotto e il totale venduto.**

SELECT

ProductID,

SUM(Quantity) AS TotaleVenduto

FROM Sales

WHERE YEAR(SaleDate) = (

SELECT

MAX(YEAR(SaleDate)) FROM Sales

)

GROUP BY ProductID

HAVING SUM(Quantity) > (

SELECT AVG(SommaQuantita)

FROM (

SELECT SUM(Quantity) AS SommaQuantita

FROM Sales

WHERE YEAR(SaleDate) = (

SELECT MAX(YEAR(SaleDate)) FROM Sales

)

GROUP BY ProductID

) AS MediaPerProdotto

);

Questo è come ho ragionato:

Con il primo select inserisco quello che deve comparire nel result set come da consegna;

Con WHERE YEAR limito I dati all’anno più recente nel database;

Con GROUP BY e SUM(quantity) raggruppo per prodotto e calcolo quanta quantità ha venduto ciascun prodotto nel 2024 (anno più recente);

Con la subquery nell’HAVING calcolo la media delle quantità vendute per ciascun prodotto nel 2024;

Con HAVING SUM(quantity) > della media calcolata ottengo solo i prodotti che nel 2024 hanno venduto una quantità maggiore della media precedentemente calcolata.

Immagine che contiene testo, schermata, Carattere, linea

Il contenuto generato dall'IA potrebbe non essere corretto.

**4) Esporre l’elenco dei soli prodotti venduti e per ognuno di questi il fatturato totale per anno.**

SELECT

p.ProductID,

p.ProductName,

YEAR(s.SaleDate) AS Anno,

SUM(s.Quantity \* s.UnitPrice) AS FatturatoTotale

FROM Sales s

JOIN Product p

ON s.ProductID = p.ProductID

GROUP BY ProductID, YEAR(s.SaleDate)

ORDER BY ProductID, Anno

;

Qui ho semplicemente scelto di mostrare ProductID, ProductName (grazie ad una join), anno (estraendolo con YEAR) e fatturato totale (calcolato moltiplicando quantità per prezzo unitario).

Con GROUP BY ho raggruppato le righe per prodotto e per anno e con ORDER BY le ho ordinate per chiarezza (forse inutilmente perché l’anno è solamente il 2024 quindi senza order by non sarebbe cambiato nulla in questo caso).

Immagine che contiene testo, schermata, Carattere, numero

Il contenuto generato dall'IA potrebbe non essere corretto.

**5) Esporre il fatturato totale per stato per anno. Ordina il risultato per data e per fatturato decrescente.**

SELECT

c.CountryName,

YEAR(s.SaleDate) AS Anno,

SUM(s.Quantity \* s.UnitPrice) AS FatturatoTotale

FROM Sales s

JOIN Country c

ON s.CountryID = c.CountryID

GROUP BY c.CountryName, YEAR(s.SaleDate)

ORDER BY Anno, FatturatoTotale DESC

;

In questa ho semplicemente fatto una JOIN per mostrare CountryName, anno e FatturatoTotale, calcolato sempre con un SUM(quantità \* prezzo unitario). Poi ho raggruppato per paese e per anno e infine ho ordinato come da consegna.

Immagine che contiene testo, Carattere, numero, schermata

Il contenuto generato dall'IA potrebbe non essere corretto.

**6) Rispondere alla seguente domanda: qual è la categoria di articoli maggiormente richiesta dal mercato?**

SELECT

cat.CategoryName,

SUM(s.Quantity) AS TotaleVenduto

FROM Sales s

JOIN Product p

ON s.ProductID = p.ProductID

JOIN Category cat

ON p.CategoryID = cat.CategoryID

GROUP BY cat.CategoryName

ORDER BY TotaleVenduto DESC

LIMIT 1

;

Faccio un paio di JOIN per arrivare dalla vendita al prodotto e dal prodotto alla categoria;

Sommo le quantità vendute con SUM(quantity) e le ordino per categoria con GROUP BY, infine le ordino per TotaleVenduto (in due varianti):

Immagine che contiene testo, Carattere, linea, schermata

Il contenuto generato dall'IA potrebbe non essere corretto.

Senza LIMIT uscirebbe così, giusto per vedere anche le altre categorie (dato che sono poche):

Immagine che contiene testo, Carattere, linea, numero

Il contenuto generato dall'IA potrebbe non essere corretto.

**7) Rispondere alla seguente domanda: quali sono i prodotti invenduti? Proponi due approcci risolutivi differenti.**

#variante 1

SELECT

p.ProductID,

p.ProductName

FROM Product p

LEFT JOIN Sales s

ON p.ProductID = s.ProductID

WHERE s.SaleID IS NULL

;

#variante 2

SELECT

ProductID,

ProductName

FROM Product

WHERE ProductID NOT IN (

SELECT DISTINCT ProductID

FROM Sales

);

Dato che si vogliono identificare i prodotti presenti nella tabella Product ma assenti in quella Sales ho sviluppato due varianti, come da richiesta:

Nella prima variante ho fatto una LEFT JOIN (in modo da prendere tutti i prodotti, anche quelli senza vendite) più WHERE s.SaleID IS NULL in modo da filtrare solo quelli mai venduti.

Nella seconda variante, attraverso un NOT IN con subquery, ho selezionato solo i prodotti il cui ProductID non appare mai nella tabella delle vendite.

Nel mio dataset non esistono prodotti invenduti, infatti entrambe le varianti portano ad un result vuoto, quindi nel mio caso non ho nessuna riga restituita.

**8) Creare una vista sui prodotti in modo tale da esporre una “versione denormalizzata” delle informazioni utili (codice prodotto, nome prodotto, nome categoria).**

CREATE VIEW v\_ProductsInfo AS

SELECT

p.ProductID,

p.ProductName,

c.CategoryName

FROM Product p

JOIN Category c

ON p.CategoryID = c.CategoryID

;

SELECT \* FROM v\_ProductsInfo;

Immagine che contiene testo, Carattere, numero, linea

Il contenuto generato dall'IA potrebbe non essere corretto.

**9) Creare una vista per le informazioni geografiche.**

CREATE VIEW vista\_geografica AS

SELECT

c.CountryID,

c.CountryName,

c.ISOCode,

r.RegionID,

r.RegionName

FROM Country c

JOIN Region r

ON c.RegionID = r.RegionID

;

SELECT \* FROM vista\_geografica;

Immagine che contiene testo, numero, Carattere, schermata

Il contenuto generato dall'IA potrebbe non essere corretto.