**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 si 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?

**Case Study**

1. Un **database** è una collezione di dati salvati in modo persistente e organizzati in modo logico per gestire operazioni CRUD.
2. Un **DBMS** (**D**ata**b**ase **M**anagement **S**ystem) è un software per gestire le operazioni fatte su un database, cioè un sistema che permette di fare operazioni CRUD su un database.
3. Il primo ad essere eseguito è il **FROM** (obbligatorio) che prende la/e tabella/e con i dati su cui si vuole fare il SELECT dal database, poi il **WHERE** che filtra le condizioni della tabella presa, seguito da **GROUP BY** che raggruppa i record di uno o più campi in modo univoco, **HAVING** (da usare solo se si usa GROUP BY teoricamente) che consente di filtrare sui raggruppamenti di GROUP BY (WHERE non lo può fare senza fare subquery), **SELECT** (obbligatorio e nella sintassi viene prima del FROM) seleziona i campi in output, **ORDER BY** ordina l’output in base ad un orientamento su uno o più campi, **LIMIT** imposta un limite hai record visualizzati dell’output.
4. Ho la tabella catalogo con prodottoID che è la PK e un campo che è categoria, voglio sapere quanti prodotti diversi ho per ogni categoria, **SELECT categoria,COUNT(prodottoID) FROM catalogo GROUP BY categoria; -- prendo la tabella catalogo raggruppo le righe per catalogo e conto quante volte viene ripetuto un record con la stessa categoria**
5. L’**OLTP** (**O**n**L**ine **T**ransactional **P**rocessing) gestisce le transazioni (una o più operazioni CRUD definite in modo atomico) con il database in modo integro e sicuro, mentre l’**OLAP** (**O**n**L**ine **A**nalytical **P**rocessing) utilizza grandi volumi di dati organizzati per facilitare le analisi.
6. Una **JOIN** viene usata per **collegare dati tra diverse tabelle** con una chiave esterna, mentre l’output ottenibile della **subquery** è **indipendente da collegamenti** tramite chiavi esterne.
7. I **DDL** (**D**ata **D**efinition **L**anguage) sono i comandi (CRUD) che vanno a **definire** (cioè creare, modificare e cancellare) **il contenitore del DB**, mentre i **DML** (**D**ata **M**anipulation **L**anguage) sono i comandi (CRUD) che vanno a **definire** (cioè creare, modificare e cancellare) **il contenuto del DB.**
8. La funzione o istruzione per ottenere la data è **YEAR**, i principali esempi del suo utilizzo sono **YEAR(‘2025-06-15’)** e anche su una colonna **SELECT ColonnaData, YEAR(ColonnaData) FROM NomeTabella; -- con ColonnaData che è una colonna di tipo data della tabella NomeTabella**
9. L’operatore logico **AND restituisce TRUE solo se entrambe sono TRUE** altrimenti restituisce FALSE, l’operatore logico **OR restituisce FALSE solo se entrambe sono FALSE** altrimenti restituisce TRUE.
10. **Si, con una query scalare**, che ha nel WHERE della subquery una condizione che cerca solo una riga nella query principale (cioè esterna) e ha la subquery che restituisce un valore singolo (cioè una riga e una colonna).
11. L’operatore logico **IN controlla se un valore è contenuto in un insieme di valori** (cioè **se un valore è uguale a più valori**), mentre l’operatore logico **OR controlla due condizioni** (quindi **non è limitato** solo **all’uguale**, **può avere** in condizioni diverse **controlli e valori diversi** e **ogni controllo** viene **fatto singolarmente e non con un insieme**).
12. L’operatore logico **BETWEEN include anche gli estremi**, cioè Valore BETWEEN ValoreMinimo AND ValoreMassimo significa Valore compreso tra ValoreMinimo e ValoreMassimo inclusi gli estremi.

ToysGroup è un’azienda che distribuisce articoli (giocatoli) in diverse aree geografiche del mondo.

I prodotti sono classificati in categorie e i mercati di riferimento dell’azienda sono classificati in regioni di vendita.

In particolare:

1. Le entità individuabili in questo scenario sono le seguenti:

* Product
* Region
* Sales

1. Le relazioni tra le entità possono essere descritte nel modo seguente:

* Product e Sales
* Un prodotto puo’ essere venduto tante volte (o nessuna) per cui è contenuto in una o più transazioni di vendita.
* Ciascuna transazione di vendita è riferita ad uno solo prodotto
* Region e Sales
* Possono esserci molte o nessuna transazione per ciascuna regione
* Ciascuna transazione di vendita è riferita ad una sola regione

1. Le entità Product e Region presentano delle gerarchie:

* L’entità prodotto contiene, oltre alle informazioni del singolo prodotto, anche la descrizione della categoria di appartenenza. L’entità prodotto contiene quindi una gerarchia: un prodotto puo’ appartenere ad una sola categoria mentre la stessa categoria puo’ essere associata a molti prodotti diversi.

*Esempio: gli articoli ‘Bikes-100’ e ‘Bikes-200’ appartengono alla categoria Bikes; gli articoli ‘Bike Glove M’ e ‘Bike Gloves L’ sono classificati come Clothing.*

* L’entità regione contiene una gerarchia: più stati sono classificati in una stessa regione di vendita e una stessa regione di vendita include molti stati.

*Esempio: gli stati ‘France’ e ‘Germany’ sono classificati nella region WestEurope; gli stati ‘Italy’ e ‘Greece’ sono classificati nel mercato SouthEurope*.

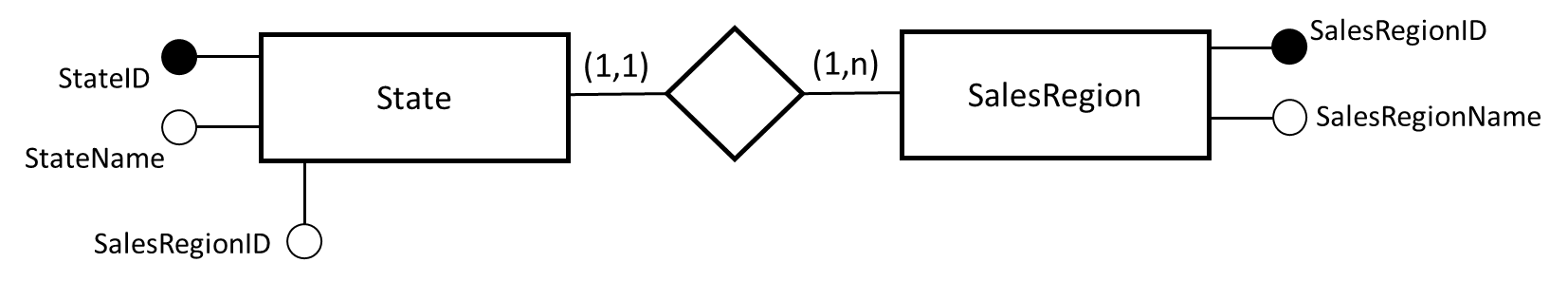
È necessario progettare e implementare fisicamente un database che modelli lo scenario garantendo l’**integrità referenziale** e la **minimizzazione della ridondanza dei dati.**

In altre parole, progetta opportunamente un numero di tabelle e di relazioni tra queste sufficiente a garantire la **consistenza del dato.**

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

La progettazione concettuale deve includere tutte le entità coinvolte e le relazioni tra queste. Per ciascuna entità indica l’attributo chiave e i principali attributi descrittivi (non è necessario indicare tutti gli attributi).

*Esempio di schema E/R*



*Lo schema proposto è puramente esemplificativo e non esaustivo o completo per la soluzione!*

La progettazione logica deve includere, per ciascuna tabella, tutte le colonne che poi verranno implementate fisicamente e deve esplicitare la cardinalità dei campi utilizzati per definire la relazione.

*Esempio di schema grafico delle tabelle e delle relazioni tra le stesse.*



n

1



*Il diagramma è puramente esemplificativo e non esaustivo o completo per la soluzione!*

Immagine che contiene testo, diagramma, linea, numero

Descrizione generata automaticamente**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).**

CREATE TABLE table\_name (

column1 datatype option,

column2 datatype option,

column3 datatype,

....

);

CREATE TABLE Region(

RegionID INT AUTO\_INCREMENT PRIMARY KEY,

RegionName VARCHAR(25)

);

CREATE TABLE Category(

CategoryID INT AUTO\_INCREMENT PRIMARY KEY,

CategoryName VARCHAR(25)

);

CREATE TABLE State(

StateID INT AUTO\_INCREMENT PRIMARY KEY,

StateName VARCHAR(25),

RegionID INT,

FOREIGN KEY (RegionID) REFERENCES Region(RegionID)

);

CREATE TABLE Product(

ProductID INT AUTO\_INCREMENT PRIMARY KEY,

ProductName VARCHAR(25),

Price DECIMAL(10,2),

CategoryID INT,

FOREIGN KEY (CategoryID) REFERENCES Category(CategoryID)

);

CREATE TABLE Sales(

OrderID INT AUTO\_INCREMENT PRIMARY KEY,

Quantity INT,

OrderDate DATE,

ProductID INT,

StateID INT,

FOREIGN KEY (ProductID) REFERENCES Product(ProductID),

FOREIGN KEY (StateID) REFERENCES State(StateID)

);

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

INSERT INTO Region(RegionName) VALUES

(WestEurope), (SouthEurope), (NorthEurope);

INSERT INTO Category(CategoryName) VALUES

(Bikes), (Clothing) (Bike-Parts);

INSERT INTO State(StateName, RegionID) VALUES

(France, 1), (Germany, 1), (Italy, 2), (Greece, 2), (Sweden, 3);

INSERT INTO Product(ProductName, Price, CategoryID) VALUES

(*Bikes-100, 250.00, 1*), (*Bikes-200, 450.00, 1*), (*Bike Glove S,12.00,2*),

(*Bike Glove M,15.00,2*), (*Bike Glove L,16.00,2*),(Bike Bell, 7.00,3);

INSERT INTO Sales (Quantity, OrderDate, ProductID, StateID) VALUES

(2, 2025-06-14, 1, 2), (1, 2025-06-14,3,2), (1, 2025-06-07, 2, 3), (3, 2025-06-21, 4, 1), (1, 2024-06-22, 6, 4);

**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 e 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

Risposte:

CREATE VIEW SalesYear AS

SELECT \*, YEAR(OrderDate) AS OrderYear FROM Sales;

CREATE VIEW ProductSoldStateYearQuantity AS

SELECT ProductID, StateID, OrderYear, SUM(Quantity) AS QuantitySold

FROM SalesYear GROUP BY ProductID, StateID, OrderYear;

CREATE VIEW ProductSoldStateYearPrice AS

SELECT PQ.ProductID AS ProductID, P.CategoryID AS CategoryID,

PQ.StateID AS StateID, PQ.OrderYear AS OrderYear,

PQ.QuantitySold AS QuantitySold, P.Price AS UnitPrice,

PQ.QuantitySold \* P.Price AS TotalSold

FROM ProductSoldStateYearQuantity PQ JOIN Product P ON PQ.ProductID = P.ProductID;

**9)** CREATE VIEW NameStateRegion AS

SELECT S.StateID AS StateID, S.StateName AS StateName, R.RegionName AS RegionName

FROM State S JOIN Region R ON S.RegionID = R.RegionID;

**8)** CREATE VIEW NameProductCategory AS

SELECT P.ProductID AS ProductID, P.ProductName AS ProductName, C.CategoryName AS CategoryName

FROM Product P JOIN Category C ON P.CategoryID = C.CategoryID;

**1)** SELECT IF((SELECT COUNT(RegionID) FROM Region) = (SELECT COUNT(DISTINCT RegionID) FROM Region),

'Campo univoco', 'Non è un campo univoco') AS Messaggio;

SELECT IF((SELECT COUNT(CategoryID) FROM Category) = (SELECT COUNT(DISTINCT CategoryID) FROM Category),

'Campo univoco', 'Non è un campo univoco') AS Messaggio;

SELECT IF((SELECT COUNT(StateID) FROM State) = (SELECT COUNT(DISTINCT StateID) FROM State),

'Campo univoco', 'Non è un campo univoco') AS Messaggio;

SELECT IF((SELECT COUNT(ProductID) FROM Product) = (SELECT COUNT(DISTINCT ProductID) FROM Product),

'Campo univoco', 'Non è un campo univoco') AS Messaggio;

SELECT IF((SELECT COUNT(OrderID) FROM Sales) = (SELECT COUNT(DISTINCT OrderID) FROM Sales),

'Campo univoco', 'Non è un campo univoco') AS Messaggio;

**2)** SELECT S.OrderID AS OrderID, S.OrderDate AS OrderDate,

NPC.ProductName AS ProductName, NPC.CategoryName AS CategoryName,

NSR.StateName AS StateName, NSR.RegionName AS RegionName,

IF(DATEDIFF(CURDATE(),S.OrderDate) > 180, 'True', 'False') AS PassedHalfYear

FROM Sales S

JOIN NameProductCategory NPC ON S.ProductID = NPC.ProductID

JOIN NameStateRegion NSR ON S.StateID = NSR.StateID;

**3)** CREATE VIEW ProductSoldStateLastYearPrice AS

SELECT \* FROM ProductSoldStateYearPrice WHERE OrderYear = (

SELECT MAX(OrderYear) FROM ProductSoldStateYearPrice

);

SELECT ProductID, TotalSold FROM ProductSoldStateLastYearPrice WHERE TotalSold > (

SELECT AVG(TotalSold) FROM ProductSoldStateLastYearPrice

);

**4)** SELECT ProductID, OrderYear, SUM(TotalSold) AS TotalTurnover

FROM ProductSoldStateYearPrice GROUP BY ProductID, OrderYear;

**5)** SELECT StateID, OrderYear, SUM(TotalSold) AS TotalTurnover

FROM ProductSoldStateYearPrice GROUP BY StateID, OrderYear

ORDER BY OrderYear, TotalTurnover;

**6)** SELECT CategoryID FROM (

SELECT CategoryID, SUM(TotalSold) AS TurnoverCategory

FROM ProductSoldStateYearPrice GROUP BY CategoryID

) AS TotalCategory

ORDER BY TurnoverCategory DESC LIMIT 1;

**7)** SELECT ProductID FROM Product WHERE ProductID NOT IN (

SELECT DISTINCT ProductID FROM Sales

);

SELECT P.ProductID FROM Product P

LEFT JOIN (

SELECT ProductID, COUNT(Quantity) AS QuantitySold FROM Sales GROUP BY ProductID

) AS S

ON P.ProductID = S.ProductID

WHERE QuantitySold IS NULL;