# Test di fine settimana – Week 4

Nome Laura

Cognome Martines

Data 16/07/2021

*Leggete attentamente ogni domanda e argomentare quanto più possibile **fornendo anche degli esempi**. ATTENZIONE: Le domande a risposta multipla possono contenere più risposte corrette.*

---

1. *Spiegare brevemente la differenza tra i comandi SELECT, INSERT, UPDATE E DELETE e fare un esempio per ognuno*

Il comando SELECT serve per estrarre dati di un database. Il suo statement è

SELECT "qualcosa"
FROM "tabella"

Esso fa una proiezione, quindi proietta gli attributi contenuti nelle una o più colonne selezionate; per avere una proiezione di tutte le colonne della tabella, si usa un asterisco. Se al posto di "SELECT" si scrive "SELECT DISTINCT", esso restituisce una sola volta le combinazioni degli attributi delle varie righe che appaiono più volte (quindi se si sceglie di proiettare un singolo attributo, esso proietta ogni valore una volta anche se nella tabelle è presente N volte, mentre se si proiettano più attributi, deve essere una la combinazione tra loro).
All'interno del FROM è possibile compiere operazioni tra tabelle (join ad esempio) per combinarne i risultati. Inoltre, è possibile porre condizioni ai risultati da proiettare tramite il comando WHERE, che si mette in coda al FROM.
Esempio: visualizzare i nomi di tutti gli autori russi:

```
SELECT Autori.Nome as [Nome Autore]
From Autori
WHERE Autori.Nazionalita = 'RUS'
```

Il comando INSERT serve per aggiungere uno o più record (righe) in una tabella. Il suo statement è

INSERT INTO "nome_tabella" (colonna1, colonna2, …, colonnaN)
VALUES (valore1, valore2, …, valoreN)

Se non vengono specificati i nomi delle colonne in cui inserire i dati, esso considera le colonne nell'ordine in cui appaiono in tabella (in tal caso, bisogna fare attenzione che i tipi di dati delle colonne e quelli dei dati inseriti siano uguali, altrimenti dà errore); se invece si specificano un ugual numero di nomi di colonne e di valori, considera ogni coppia colonna-valore in ordine (in tal modo, è possibile inserire i dati solo in colonne specifiche, sempre che le rimanenti possano contenere valori nulli). Se la colonna contiene un ID auto-generato, questo non deve essere considerato nel conto delle colonne in cui inserire il valore (il software riconosce che il codice è auto-generato e compila quel campo in autonomia).
Ripetendo più volte il comando VALUES (valore1, valore2, …, valoreN), è possibile fare un inserimento multiplo
Esempio: inserire un nuovo autore nella tabella Autori:

```
INSERT INTO Autori (Nome, DataNascita, DataMorte, Nazionalita)
VALUES ('Alessandro Manzoni', 1785, 1873, ITA
```

UPDATE si usa per modificare un record. Il suo statement è

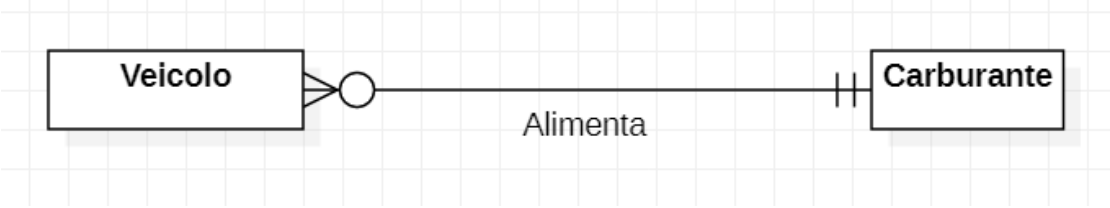
UPDATE "tabella" SET colonna 1 = valore1, colonna 2 = valore2,… WHERE questa condizione è vera;

Il WHERE non è strettamente necessario, ma senza di esso le modifiche vengono apportate a tutti i record della tabella, quindi è un processo piuttosto pericoloso
UPDATE
Esempio: Modificare l'anno di nascita dell'autore il cui ID è 5:

```
UPDATE Autore SET Autore.AnnoNascita=1968
WHERE Autore.ID =5;
```

DELETE serve per cancellare uno o più record da una tabella. Il suo statement è
DELETE FROM "tabella" WHERE questa condizione è vera
Anche in questo caso, il WHERE non è strettamente necessario, ma se non si mette, il comando cancella tutti i record della tabella, di fatto svuotandola completamente (la struttura della tabella resta invece intatta): il DELETE usato in questo modo equivale al comando TRUNCATE. Nella condizione WHERE del DELETE si usa di solito la chiave primaria, per maggiore sicurezza (ad esempio, per evitare di eliminare più record in caso di campi uguali).
Es. Eliminare il personaggio "Poliziotto" dalla tabella "Personaggio":
DELETE FROM Personaggio WHERE Personaggio.Nome = 'Poliziotto';

2. *Disegnare un esempio di tabelle con relazione 1:N e un esempio di relazione N:N e spiegare quali sono le differenze*
La relazione 1:N indica che ogni istanza di una entità (nell'esempio sottostante, veicolo) può avere una relazione con una sola istanza della seconda entità coinvolta nella relazione (nell'esempio sottostante, Carburante); mentre ogni istanza dell'entità carburante può avere relazione con 0 o più istanze dell'entità veicolo. Si dice che un veicolo può essere alimentato da un solo carburante, mentre un carburante può alimentare diversi veicoli.




Nella relazione N:N invece, ogni istanza di una entità può essere in relazione con più istanze della seconda e viceversa. Nell'esempio sottostante, un Romanzo può avere più personaggi, ma anche un personaggio può trovarsi in più romanzi. Nel passaggio dal diagramma concettuale a quello logico, di solito la relazione N:N viene ricondotta a due relazioni 1:N mediante l'inserimento di una "Bridge Table" o "tabella ponte" che contiene le coppie romanzo/personaggio, ossia tiene traccia di quali personaggi figurano in ogni romanzo. Tale tabella può poi contenere altri attributi, ma la loro presenza è opzionale.


3. *Spiegare la differenza tra una PRIMARY KEY e una FOREIGN KEY*
La Primary Key o Chiave Primaria (indicata come PK), identifica all'interno di una tabella l'attributo (la colonna) il cui valore può identificare in modo univoco ogni record. La Primary Key deve essere definita per ogni tabella creata, e può essere sia un attributo già presente nella tabella (per esempio, il codice fiscale per un individuo o la targa per un veicolo) oppure può essere un ID aggiunto (ed eventualmente anche autogenerato). Ogni tabella può avere una sola chiave primaria (anche se essa può essere composta, ossia può essere creata dalla combinazione

di due o più attributi) e non è possibile inserire in una tabella due record che abbiano uguale primary key.

La Foreign Key o Chiave Esterna (indicata come FK), è un campo (o una combinazione di campi) in una tabella che viene usata come riferimento in un'altra tabella. Essa viene usata per generare le relazioni tra tabelle

4. *Quando si utilizza l'istruzione "GROUP BY"? Fare un esempio pratico comprensivo di query SQL*
L'istruzione GROUP BY serve per raggruppare tutti i record di una tabella che hanno gli stessi valori nella colonna specificata dopo l'istruzione GROUP BY, appunto. Si usa quando si devono compiere operazioni su un set di dati attraverso le funzioni aggregate per definire come dividere i dati.
Esempio: Mostrare il prezzo medio dei prodotti venduti in un negozio in un certo lasso di tempo, con il loro codice prodotto

```
SELECT codice_prodotto, AVG(prezzo_prodotto) AS [prezzo medio]
FROM elenco_prodotti
GROUP BY codice_prodotto
```

In tal caso, il codice raggruppa i prodotti secondo il loro codice (quindi raggruppa tutti i record che riguardano un singolo prodotto) e fa la media dei prezzi che tale prodotto ha avuto nel tempo. L'AS inserito dopo la media è un alias che serve solamente a dare un nome più comprensibile a quella colonna nella proiezione che verrà eseguita dal SELECT.

5. *Cos'è un Constraint? Fornire 2 esempi di uso in SQL*
Un constraint è un vincolo che si usa per settare delle regole da seguire durante l'inserimento dei dati in una tabella. Essi possono essere di vario tipo. Anche le PK e le FK sono constraint, altri esempi sono il DEFAULT, che definisce che valore inserire in un determinato campo, se esso viene lasciato nullo (quindi se l'utente che inserisce il dato non specifica un altro valore);
oppure IDENTITY, che permette di avere un intero autogenerato. In questo caso si può definire il seme (seed), ossia il valore di partenza di tale intero, e l'incremento, ossia che valore verrà sommato a tale intero ad ogni nuovo inserimento. IDENTITY è il modo in cui si creano le PK auto-generate.

6. *Cos'è una Stored Procedure? Quali sono i casi in cui conviene ricorrere ad essa?*

Una Stored Procedure è un modo per salvare un codice SQL per poterlo usare più volte senza doverlo riscrivere; si usano in tutti i casi in cui uno stesso codice deve essere utilizzato più volte, come ad esempio per l'inserimento dei dati in un database; al loro interno possono essere inserite anche le transazioni.
Tale procedura può prendere in ingresso diversi parametri, e restituisce uno scalare, che è 0 se tutti gli step all'interno sono andati a buon fine, mentre è un valore diverso da 0 (e anche settabile dall'utente attraverso il ciclo TRY/CATCH) se ha riscontrato problemi. Molte Stored Procedure sono già presenti all'interno di SSMS, ma l'utente può creare le proprie: per convenzione, il loro nome inizia per spt_ o sp_
Una volta creata (tramite il comando CREATE PROCEDURE) viene salvata all'interno della cartella programmability→ storedProcedures del database. Può essere modificata usando il comando ALTER PROCEDURE e si può richiamare nel codice usando il comando EXEC.

---

***Esercitazione pratica***

*Si vuole realizzare un sistema informativo per automatizzare la gestione di un negozio di dischi.*

*Le entità coinvolte (con i relativi attributi) sono:*

*Album:*

- *Titolo*
- *Anno di uscita*
- *Casa discografica*
- *Genere*
- *Supporto di distribuzione*

*Brano:*

- *Ttitolo*
- *Durata (espressa in secondi)*

*Band:*

- *Nome*
- *NumeroComponenti*

*È possibile che uno stesso brano faccia parte di più di un album (ad es. le raccolte contengono brani appartenenti, in genere, ad album già pubblicati).*

*Individuare la soluzione più adatta a livello di tabelle e creare tutte le relazioni necessarie.*

*Implementare i seguenti vincoli:*

- *Gli id devono essere autoincrementali*
- *Un album deve essere considerato unico sulla base del titolo, anno di uscita, casa editrice e genere e supporto (se uno stesso album viene memorizzato su, ad esempio, due supporti differenti, i dati relativi a quell'album devono essere registrati separatamente).*
- *Il genere può essere di queste tipologie: Classico, Jazz, Pop, Rock, Metal*
- *Il supporto di distribuzione deve essere scelto tra: CD, Vinile, Streaming*

*Una volta realizzato il modello entità-relazionale realizzare le seguenti query SQL:*

1) *Scrivere una query che restituisca i titoli degli album di Franco Battiato;*
2) *Selezionare tutti gli album editi dalla casa editrice nell'anno specificato;*
3) *Scrivere una query che restituisca tutti i titoli delle canzoni dei U2 appartenenti ad album pubblicati prima del 1990;*
4) *Individuare tutti gli album in cui è contenuta la canzone "Imagine";*
5) *Restituire il numero totale di canzoni eseguite dai Pooh;*
6) *Contare per ogni album, la somma dei minuti dei brani contenuti*

*Una delle query (a scelta) deve essere realizzata come Stored Procedure con parametri.*

*Creare una view che mostri i dati completi dell'album, della band e dei brani contenuti in esso.*

*Scrivere una funzione utente che calcoli per ogni genere musicale quanti album sono inseriti in catalogo.*

*Caricare la prova pratica e teorica su Github. Per la parte pratica, caricare gli script SQL necessari a ricreare il modello, le query, la view e la funzione.*