Alessia Gualtieri 30/07/2021

**TEST WEEK 3 ACADEMY**

1. **SQL è definite come DDL, DML, DCL e TCL. Spiegare il significato.**

**DDL**: Data Definition Language, cioè permette di strutturare il database. Presenta quindi funzioni come CREATE (per la creazione del database),

ALTER (per la modifica della struttura di un database già esistente),

DROP(per l’eliminazione di oggetti dal database),

RENAME(per rinominare le entità)..

**DML**: Data Manipulation Language, cioè permette di modificare i dati, recuperarli, selezionarli. Ha quindi funzioni come

SELECT (Seleziona degli attributi da una entità),

INSERT(inserisce record in una entità),

UPDATE (aggiorna i record in una entità),

DELETE (cancella dei record in una entità)

**DCL**: Data Control Language, cioè permette di organizzare le autorizzazioni degli utenti. Ha quindi funzioni come GRANT per consentire a degli utenti di aver accesso al database e REVOKE per revocare i privilegi concessi con GRANT.

**TCL**: Transition Control Language, permette di organizzare le transazioni del database da uno stato stabile ad un altro stato stabile. Ha quindi funzioni come

COMMIT per commettere una transazione nel nuovo stato stabile

ROLLBACK per tornare all’ultimo stato stabile annullando le modifiche fatte successivamente

SET TRANSACTION per specificare le caratteristiche di una transazione

1. **Descrivere le varie tipologie di JOIN. Fornire un esempio**

Le JOIN servono per mettere in collegamento attraverso le FOREIGN KEY due tabelle.

**INNER JOIN**: Restituisce l’intersezione delle due tabelle, quindi solo i record che hanno sia gli attributi di una che dell’altra tabella. È la JOIN di default, quindi se non si specifica quale si vuole viene effettuata questa.

ES:

SELECT \*

FROM Persona p

INNER JOIN Assicurazione a

ON p.ID\_Assicurazione = a.ID\_Assicurazione

Mostra solo le persone che hanno una assicurazione e la rispettiva assicurazione.

**LEFT JOIN**: Restituisce tutta la prima tabella e associa agli elementi possibili gli attributi della seconda tabella.

ES:

SELECT \*

FROM Persona p

LEFT JOIN Assicurazione a

ON p.ID\_Assicurazione = a.ID\_Assicurazione

Mostra tutte le persone, anche quelle senza assicurazione, e per quelle con una assicurazione mostra anche l’assicurazione rispettiva.

**RIGHT JOIN**: Restituisce tutta la seconda tabella e associa agli elementi possibili gli attributi della prima tabella.

ES:

SELECT \*

FROM Persona p

RIGHT JOIN Assicurazione a

ON p.ID\_Assicurazione = a.ID\_Assicurazione

Mostra tutte le assicurazioni, e per quelle con almeno una persona assicurata anche i dettagli di tale persona

**FULL JOIN:** Restituisce entrambe le tabelle complete

ES:

SELECT \*

FROM Persona p

FULL JOIN Assicurazione a

ON p.ID\_Assicurazione = a.ID\_Assicurazione

Mostra tutte le persone e tutte le assicurazioni, associando quelle associabili.

1. **Spiega l’utilità di definire un indice di ricerca nelle tabelle**

L’indice di ricerca delle tabelle serve per velocizzare la ricerca di qualche determinato recold filtrando per un determinato attributo. Per creare un indice è necessario fare

CREATE [UNIQUE] INDEX NomeIndex

ON NomeTabella (campo1 ASC/DESC, campo2 ASC/DESC,...)

Una tabella può avere più indici e ciascun indice può essere su più campi. La keyword UNIQUE va inserita solo se si vuole che l’indice sia unico.

1. **Che cos’è un trigger? Qual è la sintassi del loro utilizzo? Fornire un esempio**

Un trigger è un particolare costrutto di SQL simile agli eventi, associato ad una tabella o ad una view. Viene attivato quando avviene /prima che avvenga una determinata cosa, e svolge dei ruoli stabiliti. Può essere per esempio usato per creare un campo automaticamente in base a certi parametri.

Sintassi:

CREATE TRIGGER [Schema\_Nome.] NomeTrigger

ON [Nome\_Tabella|Nome\_View]

[**FOR | AFTER | INSTEAD OF**]

[INSERT | UPDATE | DELETE]

AS

BEGIN

//Sql statement

END

FOR, AFTER e INSTEAD OF sono delle keyword che stabiliscono quando/in che modo deve attivarsi il trigger, in particolare se prima, dopo o al posto di un comando.

Esempio:

CREATE TRIGGER AvvenutoInserimento

ON Persona AFTER INSERT

AS

BEGIN

PRINT ‘Inserimento avvenuto con successo’

END