



# Quiz

---



## Sql - PgAdmin

---

- 1. Come si crea una nuova tabella in pgAdmin?**
  - A. Clic su "Tables" > New Table
  - B. Tasto destro su "Tables" > Create > Table...
  - C. Clic su "Database" > Create Table
  - D. Menu File > New > Table
- 2. Quale comando SQL serve per aggiornare un valore in una tabella?**
  - A. ALTER
  - B. INSERT
  - C. UPDATE
  - D. SELECT
- 3. Quale comando elimina record da una tabella?**
  - A. DROP
  - B. DELETE
  - C. REMOVE
  - D. CLEAR
- 4. Quale campo è adatto come chiave primaria?**
  - A. id
  - B. modello
  - C. costo
  - D. anno\_immatricolazione
- 5. Quale tipo dati si usa per salvare date?**
  - A. TIMESTAMP
  - B. VARCHAR
  - C. TEXT
  - D. DATE
- 6. Come si visualizzano tutti i dati di una tabella in pgAdmin?**
  - A. View Data > All
  - B. View/Edit Data > All Rows
  - C. Run SQL > SELECT
  - D. Edit Table > Show Rows
- 7. Qual è il comando per eseguire una query SQL nel Query Tool?**
  - A. Ctrl+Enter
  - B. F5

- C. Alt+Q
- D. F11

8. Quale simbolo si usa per i commenti su una riga in SQL?

- A. #
- B. //
- C. --
- D. /\*

9. Quale funzione calcola la media di un campo?

- A. SUM()
- B. AVG()
- C. MEDIAN()
- D. COUNT()


10. Cosa fa il comando GROUP BY ?

- A. Ordina i risultati
- B. Raggruppa i dati per valore di una colonna
- C. Unisce due tabelle
- D. Conta i record

11. Quale clausola si usa per filtrare i dati aggregati?

- A. FILTER
- B. HAVING
- C. ORDER BY
- D. WHERE

12. Quale comando visualizza solo le auto con costo > media?

- A. SELECT \* FROM automobile WHERE costo > AVG(costo);
- B. SELECT \* FROM automobile WHERE costo > (SELECT AVG(costo) FROM automobile); 
- C. SELECT WHERE costo > MEDIAN FROM automobile;
- D. Nessuno dei precedenti

13. Come si esegue un file .sql in pgAdmin?

- A. Apri il file in Notepad
- B. Tasto destro sul DB > Query Tool > File > Open
- C. Trascina il file sul database
- D. Non è possibile

14. Come si selezionano righe che iniziano per "Fiat"?

- A. LIKE '%Fiat'
- B. LIKE 'Fiat%'
- C. ILIKE 'Fiat\*'
- D. STARTS WITH 'Fiat'

15. Qual è la differenza tra **WHERE** e **HAVING** ?

- A. Nessuna
- B. **WHERE** filtra righe, **HAVING** filtra gruppi
- C. **HAVING** è più veloce
- D. **WHERE** lavora solo su tabelle temporanee

## Risposte SQL:

1. ☒ B. Tasto destro su "Tables" > Create > Table...
2. ☒ C. UPDATE
3. ☒ B. DELETE
4. ☒ A. id
5. ☒ D. DATE
6. ☒ B. View/Edit Data > All Rows
7. ☒ B. F5
8. ☒ C. --
9. ☒ B. AVG()
10. ☒ B. Raggruppa i dati per valore di una colonna
11. ☒ B. HAVING
12. ☒ B. `SELECT * FROM automobile WHERE costo > (SELECT AVG(costo) FROM automobile);`
13. ☒ B. Tasto destro sul DB > Query Tool > File > Open
14. ☒ B. `LIKE 'Fiat%'`
15. ☒ B. WHERE filtra righe, HAVING filtra gruppi

1. Qual è l'annotazione per indicare una classe persistente in JPA?
  - A. @Table
  - B. @Entity
  - C. @Persistence
  - D. @Model
  
2. Cosa fa `@Table(name = "libro")` ?
  - A. Crea una colonna
  - B. Assegna un nome alla classe
  - C. Specifica il nome della tabella nel DB
  - D. Imposta l'ID primario
  
3. Cosa indica `@Id` in una classe?
  - A. Identifica il costruttore
  - B. Imposta un campo come modificabile
  - C. Imposta la chiave primaria
  - D. Crea una relazione
  
4. Quale strategia usa `@GeneratedValue(strategy = GenerationType.IDENTITY)` ?
  - A. Enum automatico
  - B. ID generato a mano
  - C. ID auto-incrementale
  - D. ID UUID
  
5. Cosa fa l'annotazione `@Column(length = 100, nullable = false, unique = true)` ?
  - A. Imposta vincoli su una colonna
  - B. Rende il campo invisibile
  - C. Specifica relazioni
  - D. Imposta un default
  
6. Quale annotazione converte un `enum` in stringa nel DB?
  - A. `@Convert(EnumType.STRING)`
  - B. `@EnumString`
  - C. `@Enumerated(EnumType.STRING)`
  - D. `@Column(enum)`
  
7. Dove si trova il file `persistence.xml` in un progetto Maven?
  - A. `src/config/`
  - B. `src/resources/`

- C. `src/main/resources/META-INF/`
- D. `src/main/java/`

8. **A cosa serve `persistence.xml` ?**
- A. Salva le entità
  - B. Configura la connessione al DB e JPA
  - C. Mostra le query
  - D. Contiene le entity
9. **Cosa rappresenta la `persistence-unit` nel file XML?**
- A. Il nome del database
  - B. Il tipo di entità
  - C. Il nome logico per `EntityManagerFactory`
  - D. Il driver JDBC
10. **Qual è la funzione di `EntityManagerFactory` in JPA?**
- A. Gestire le DAO
  - B. Creare entità
  - C. Creare `EntityManager`
  - D. Leggere il database
11. **Come si salva un'entità `Libro` con JPA?**
- A. `em.add()`
  - B. `em.persist()`
  - C. `em.save()`
  - D. `em.push()`
12. **Quale metodo recupera un'entità tramite ID?**
- A. `em.search()`
  - B. `em.find()` C. `em.lookup()`
  - D. `em.get()`
13. **Cosa fa `em.remove()` ?**
- A. Rimuove il database
  - B. Elimina un'entità
  - C. Cancella una tabella
  - D. Elimina il file XML
14. **Quando usare `em.getTransaction().begin()` e `commit()` ?**
- A. Per inizializzare il database
  - B. Per eseguire operazioni di scrittura
  - C. Per leggere dati
  - D. Solo in test
15. **Cosa succede se dimentichi di chiudere `EntityManagerFactory` ?**
- A. Errore di compilazione

- B. Il DB si blocca
- C. Si consumano risorse di sistema inutilmente
- D. Si cancella il DB

Ecco le **risposte JPA**:

1. ☒ B. `@Entity`
2. ☒ C. Specifica il nome della tabella nel DB
3. ☒ C. Imposta la chiave primaria
4. ☒ C. ID auto-incrementale
5. ☒ A. Imposta vincoli su una colonna
6. ☒ C. `@Enumerated(EnumType.STRING)`
7. ☒ C. `src/main/resources/META-INF/`
8. ☒ B. Configura la connessione al DB e JPA
9. ☒ C. Il nome logico per `EntityManagerFactory`
10. ☒ C. Creare `EntityManager`
11. ☒ B. `em.persist()`
12. ☒ B. `em.find()`
13. ☒ B. Elimina un'entità
14. ☒ B. Per eseguire operazioni di scrittura
15. ☒ C. Si consumano risorse di sistema inutilmente



1. Quale annotazione rappresenta una relazione multi-a-uno?
  - A. @OneToMany
  - B. @ManyToMany
  - C. @ManyToOne
  - D. @OneToOne
  
2. Qual è il lato inverso in una relazione uno-a-molti?
  - A. @JoinColumn
  - B. mappedBy
  - C. inverseJoinColumn
  - D. backReference
  
3. Dove si specifica il nome della colonna per la chiave esterna?
  - A. @JoinColumn(name = "...")
  - B. @ForeignKey(name = "...")
  - C. @Column(name = "...")
  - D. @RelationColumn(name = "...")
  
4. Quale annotazione crea una tabella intermedia per relazioni multi-a-molti?
  - A. @JoinColumn
  - B. @JoinTable
  - C. @ManyToManyTable
  - D. @RelationJoin
  
5. Qual è l'effetto di `cascade = CascadeType.ALL` ?
  - A. Elimina tutte le entità
  - B. Ignora la persistenza
  - C. Propaga tutte le operazioni (persist, remove, etc.)
  - D. Crea solo la relazione
  
6. In quale entità viene normalmente messo `mappedBy` ?
  - A. Nella principale
  - B. In quella figlia
  - C. Nel lato non proprietario della relazione
  - D. Sempre in entrambe
  
7. Che tipo di relazione è tra `Libro` e `Categoria` se un libro può avere più categorie e una categoria può appartenere a più libri?
  - A. Uno a uno
  - B. Uno a molti

- C. Molti a uno
- D. Molti a molti

8. Quale classe rappresenta l'oggetto per eseguire operazioni CRUD su un'entità?

- A. Controller
- B. DAO
- C. Service
- D. Factory

9. Cosa fa il metodo `em.find()` in un DAO?

- A. Crea una nuova entità
- B. Salva l'entità
- C. Rimuove l'entità
- D. Recupera l'entità tramite ID

10. Dove vanno salvate le classi DAO nel progetto?

- A. `src/main/java/dao/`
- B. `src/main/java/model/`
- C. `src/main/resources/`
- D. `src/test/java/dao/`

11. Quale annotazione si usa per gestire l'auto-generazione del campo ID?

- A. `@IdGenerator`
- B. `@AutoId`
- C. `@GeneratedValue`
- D. `@PrimaryKey`

12. Qual è il ruolo del file `persistence.xml` ?

- A. Contiene i dati delle entità
- B. Specifica configurazioni del DB e delle entità
- C. Gestisce le relazioni tra classi
- D. Crea la struttura del progetto

13. Quando si usa `@ManyToMany` , quale campo si usa per specificare le due colonne di join?

- A. `@JoinColumn`
- B. `@JoinTable`
- C. `@JoinKey`
- D. `@JoinMapping`

14. Cosa succede se una relazione non è ben configurata (es. `mappedBy` errato)?

- A. Nessun effetto
- B. Si ignora la relazione
- C. Errore a runtime o mancata creazione del DB corretto
- D. Viene creato un indice

15. In una relazione @OneToMany , il lato "molti" contiene:

- A. Una lista di oggetti
- B. Una sola istanza
- C. Un campo booleano
- D. Un riferimento statico

## risposte Relazioni

1. ☒ C. @ManyToOne
2. ☒ B. mappedBy
3. ☒ A. @JoinColumn(name = "...")
4. ☒ B. @JoinTable
5. ☒ C. Propaga tutte le operazioni (persist, remove, etc.)
6. ☒ C. Nel lato non proprietario della relazione
7. ☒ D. Molti a molti
8. ☒ B. DAO
9. ☒ D. Recupera l'entità tramite ID
10. ☒ A. src/main/java/dao/
11. ☒ C. @GeneratedValue
12. ☒ B. Specifica configurazioni del DB e delle entità
13. ☒ B. @JoinTable
14. ☒ C. Errore a runtime o mancata creazione del DB corretto
15. ☒ A. Una lista di oggetti



# Inheritance

---

1. Quale annotazione abilita l'ereditarietà tra entità in JPA?
  - A. `@Extendable`
  - B. `@Inheritance`
  - C. `@Parent`
  - D. `@EntitySuperclass`
2. Qual è la strategia di default usata da `@Inheritance` se non specificata?
  - A. `JOINED`
  - B. `TABLE_PER_CLASS`
  - C. `SINGLE_TABLE`
  - D. Nessuna
3. Cosa fa la strategia `JOINED` ?
  - A. Crea una tabella unica con tutti i campi
  - B. Crea una tabella per ogni sottoclasse con join
  - C. Crea solo tabelle figlie
  - D. Elimina le relazioni
4. Qual è il vantaggio della strategia `JOINED` ?
  - A. Nessun campo null
  - B. Query più veloci
  - C. Maggiore duplicazione
  - D. Non usa chiavi esterne
5. Qual è lo svantaggio di `JOINED` ?
  - A. Troppi record
  - B. Nessuna relazione
  - C. Query più lente a causa dei join
  - D. Non si può usare con PostgreSQL
6. La strategia `SINGLE_TABLE` crea:
  - A. Una tabella per ogni entità
  - B. Una tabella sola con tutti i campi
  - C. Solo la tabella padre
  - D. Nessuna tabella
7. Nella `SINGLE_TABLE` , cosa rappresenta `DTYPE` ?
  - A. Un tipo booleano
  - B. Il nome della colonna con i dati duplicati

- C. Una colonna automatica che distingue le sottoclassi
- D. Una colonna ENUM

8. Qual è il problema della strategia `SINGLE_TABLE` ?
- A. Non funziona su Oracle
  - B. Genera molti record
  - C. Campi null per le sottoclassi non usate
  - D. Richiede stored procedure
9. Cosa fa `TABLE_PER_CLASS` ?
- A. Crea solo la tabella padre
  - B. Crea una tabella per ogni sottoclasse senza join
  - C. Unisce tutte le entità in una tabella
  - D. Usa ENUM per discriminare
10. In `TABLE_PER_CLASS` , qual è lo svantaggio?
- A. Mancanza di chiavi primarie
  - B. Difficoltà a inserire dati
  - C. Non si possono fare query su superclassi
  - D. Nessun DAO
11. Quale annotazione indica una classe astratta come base per le entità?
- A. `@MappedSuperclass`
  - B. `@Entity`
  - C. `abstract class`
  - D. `@InheritanceEntity`
12. Cosa succede se si cambia la strategia da `JOINED` a `SINGLE_TABLE` ?
- A. Vengono eliminate le tabelle
  - B. Le relazioni vengono perse
  - C. Viene creata una tabella unica con tutti i campi
  - D. Il codice non compila
13. In `JOINED` , come sono collegate le sottoclassi alla superclasse?
- A. Mediante ENUM
  - B. Con chiavi esterne
  - C. Non sono collegate
  - D. Solo tramite JOIN automatici
14. Dove si scrive l'annotazione `@Inheritance(...)` ?
- A. Nella sottoclasse
  - B. Nel file `persistence.xml`
  - C. Nella superclasse
  - D. In ogni entità figlia

15. Come viene gestita la tabella della superclasse in SINGLE\_TABLE ?

- A. Non esiste
- B. Contiene solo i suoi campi
- C. Contiene tutti i campi di tutte le classi
- D. Solo metadati

## risposte Inheritance

1. ☒ B. @Inheritance
2. ☒ C. SINGLE\_TABLE
3. ☒ B. Crea una tabella per ogni sottoclasse con join
4. ☒ A. Nessun campo null
5. ☒ C. Query più lente a causa dei join
6. ☒ B. Una tabella sola con tutti i campi
7. ☒ C. Una colonna automatica che distingue le sottoclassi
8. ☒ C. Campi null per le sottoclassi non usate
9. ☒ B. Crea una tabella per ogni sottoclasse senza join
10. ☒ C. Non si possono fare query su superclassi
11. ☒ C. abstract class
12. ☒ C. Viene creata una tabella unica con tutti i campi
13. ☒ B. Con chiavi esterne
14. ☒ C. Nella superclasse
15. ☒ C. Contiene tutti i campi di tutte le classi