

COGNOME**NOME****MATRICOLA****Basi di Dati 2022/23 – 09 giugno 2023****Closed book (non è possibile consultare materiale)****Tempo a disposizione: 1h 45' parte I e II [1h 20' se senza esercizio I.A] + 45' parte III****Esercizio I.A REVERSE ENGINEERING * gli studenti che hanno aderito a opzione 2 sono esonerati**

Si consideri il seguente schema relazionale

SPEDIZIONE(CodiceSp, Mittente^{SOGGETTO}, Destinatario^{SOGGETTO}, IndirizzoSpedizioneo, CodiceCat^{CATEGORIA}, Stato)

SOGGETTO(CodiceSogg, Nome, Indirizzo)

CATEGORIA(CodiceCat Descrizione, Prezzo)

PACCO(CodiceSp^{SPEDIZIONE}, NumPacco, Peso)

LUOGO(Coordinate, Descrizione)

POSIZIONE(CodiceSp^{SPEDIZIONE}, Coordinate^{LUOGO}, DataOra)

La spedizione viene effettuata all'indirizzo del destinatario, a meno che non sia indicato un indirizzo di spedizione. Possibili valori per Stato in SPEDIZIONE sono consegnata, rifiutata, in viaggio. Possibili valori per Descrizione in CATEGORIA sono standard, fast, express.

1. si proponga uno schema concettuale Entity Relationship la cui traduzione dia luogo a tale schema logico

2. si modifichi lo schema in 1. per gestire il fatto che un soggetto possa essere cliente della società di spedizione, nel qual caso ha associato un codice cliente, una data di sottoscrizione e una data di scadenza dell'abbonamento e la tipologia di abbonamento sottoscritto.

COGNOME**NOME****MATRICOLA****Esercizio I.B NORMALIZZAZIONE**

1. Si consideri il seguente schema di relazione, che rappresenta alcune informazioni sui prodotti di una falegnameria e i relativi componenti. Vengono indicati: il nome del prodotto (attributo Prodotto), il nome di un componente del prodotto (Componente), il tipo del componente di un prodotto (attributo Tipo), la quantità del componente necessaria per un certo prodotto (attributo Q), il prezzo unitario del componente di un certo prodotto (attributo PC), il fornitore del componente (attributo Fornitore) e il prezzo totale del singolo prodotto (attributo PT).

PRODOTTO (Prodotto, Componente, Tipo, Q, PC, Fornitore, PT)

Determinare, per ciascuna delle seguenti affermazioni, se rappresentano dipendenze funzionali per la relazione PRODOTTO e, in caso affermativo, presentare la dipendenza:

- a) Il prezzo unitario di vendita di un componente è unico per ogni fornitore e quantità del componente ordinata.
- b) Un prodotto può contenere diversi tipi di componenti.
- c) La quantità di un componente necessaria per produrre un certo prodotto è sempre la stessa, indipendentemente dal fornitore.

1. Data la relazione $R(A,B,C,D,E)$ e le dipendenze funzionali $A \rightarrow B$, $DE \rightarrow C$ e $AB \rightarrow E$, determinare le chiavi di R e specificare se R è in 3NF o in BCNF, motivando le risposte.

| COGNOME | NOME | MATRICOLA |
|---------|------|-----------|
|---------|------|-----------|

Esercizio II.A – ALGEBRA RELAZIONALE

In riferimento al seguente schema relazionale:

SPEDIZIONE(CodiceSp, Mittente^{SOGGETTO}, Destinatario^{SOGGETTO}, IndirizzoSpedizione^o, CodiceCat^{CATEGORIA}, Stato)
 SOGGETTO(CodiceSogg, Nome, Indirizzo)
 CATEGORIA(CodiceCat, Descrizione, Prezzo)
 PACCO(CodiceSp^{SPEDIZIONE}, NumPacco, Peso)
 LUOGO(Coordinate, Descrizione)
 POSIZIONE(CodiceSp^{SPEDIZIONE}, Coordinate^{LUOGO}, DataOra)

La spedizione viene effettuata all'indirizzo del destinatario, a meno che non sia indicato un indirizzo di spedizione. Possibili valori per Stato in SPEDIZIONE sono consegnata, rifiutata, in viaggio. Possibili valori per Descrizione in CATEGORIA sono standard, fast, express.

Formulare le seguenti interrogazioni in **algebra relazionale**.

1. Determinare nome e indirizzo del destinatario delle spedizioni di categoria “express” la cui posizione corrente è “Genova Aeroporto”.

2. Determinare le posizioni da cui sono transitate tutte le spedizioni effettuate dal cliente con codice 54661.

Suggerimento per verifica/autovalutazione: Per ogni interrogazione, dopo averla formulata, effettuare i controlli richiesti e validare con V se si ritiene che il controllo sia superato, con X se si ritiene che non lo sia.

| Verifica/autovalutazione | a) | b) |
|---|----|----|
| L'interrogazione formulata è corretta dal punto di vista dei vincoli di schema | | |
| La richiesta e l'interrogazione formulata restituiscono una relazione con lo stesso schema | | |
| La richiesta e l'interrogazione formulata sono entrambe monotone/non monotone | | |
| Su una piccola istanza, la richiesta e l'interrogazione formulata restituiscono lo stesso risultato | | |

| COGNOME | NOME | MATRICOLA |
|---------|------|-----------|
|---------|------|-----------|

Esercizio II.B - SQL

In riferimento al seguente schema relazionale:

SPEDIZIONE(CodiceSp, Mittente^{SOGGETTO}, Destinatario^{SOGGETTO}, IndirizzoSpedizione^o, CodiceCat^{CATEGORIA}, Stato)

SOGGETTO(CodiceSogg, Nome, Indirizzo)

CATEGORIA(CodiceCat *Descrizione*, Prezzo)

PACCO(CodiceSp^{SPEDIZIONE}, NumPacco, Peso)

LUOGO(Coordinate, Descrizione)

POSIZIONE(CodiceSp^{SPEDIZIONE}, Coordinate^{LUOGO}, DataOra)

La spedizione viene effettuata all'indirizzo del destinatario, a meno che non sia indicato un indirizzo di spedizione. Possibili valori per Stato in SPEDIZIONE sono consegnata, rifiutata, in viaggio. Possibili valori per Descrizione in CATEGORIA sono standard, fast, express.

Formulare le seguenti interrogazioni in **SQL**.

1. Determinare i soggetti che non hanno mai rifiutato spedizioni (cioè non sono mai stati destinatari di spedizioni con stato “rifiutata”).

2. Determinare la spedizione con stato “in viaggio” che contiene il maggior numero di pacchi.

| COGNOME | NOME | MATRICOLA |
|---------|------|-----------|
|---------|------|-----------|

| Op. | Funzionalità | Cond. | Semantica |
|-------------|---|------------------------|---|
| Π_A | $\mathcal{R}(U) \rightarrow \mathcal{R}(A)$ | $A \subseteq U$ | $\Pi_A(R) = \{t[A] \mid t \in R\}$ |
| σ_F | $\mathcal{R}(U) \rightarrow \mathcal{R}(U)$ | $A(F) \subseteq U$ | $\sigma_F(R) = \{t \mid t \in R \wedge F(t)\}$ |
| \times | $\mathcal{R}(U) \times \mathcal{R}(V) \rightarrow \mathcal{R}(U \cup V)$ | $U \cap V = \emptyset$ | $R_1 \times R_2 = \{t_1 \cdot t_2 \mid t_1 \in R_1 \wedge t_2 \in R_2\}$ |
| \cup | $\mathcal{R}(U) \times \mathcal{R}(U) \rightarrow \mathcal{R}(U)$ | | $R_1 \cup R_2 = \{t \mid t \in R_1 \vee t \in R_2\}$ |
| $-$ | $\mathcal{R}(U) \times \mathcal{R}(U) \rightarrow \mathcal{R}(U)$ | | $R_1 - R_2 = \{t \mid t \in R_1 \wedge t \notin R_2\}$ |
| \cap | $\mathcal{R}(U) \times \mathcal{R}(U) \rightarrow \mathcal{R}(U)$ | | $R_1 \cap R_2 = \{t \mid t \in R_1 \wedge t \in R_2\}$ |
| \bowtie_F | $\mathcal{R}(U) \times \mathcal{R}(V) \rightarrow \mathcal{R}(U \cup V)$ | $U \cap V = \emptyset$ | $R_1 \bowtie_F R_2 = \{t_1 \cdot t_2 \mid t_1 \in R_1 \wedge t_2 \in R_2\}$ |
| \bowtie | $\mathcal{R}(U) \times \mathcal{R}(V) \rightarrow \mathcal{R}(U \cup V)$ | | $\wedge F(t_1 \cdot t_2)\}$ |
| \div | $\mathcal{R}(U) \times \mathcal{R}(V) \rightarrow \mathcal{R}(U \setminus V)$ | $V \subset U$ | $R_1 \bowtie R_2 = \{t \mid t[U] \in R_1 \wedge t[V] \in R_2\}$ |
| | | | $R_1 \div R_2 = \{t \mid \forall t_2 \in R_2 \exists t_1 \in R_1 t.c. t_1[U \setminus V] = t, t_1[V] = t_2\}$ |

$$\rho_{A \leftarrow A'} \quad R(U) \rightarrow R(U \setminus A \cup A')$$

$$A \subseteq U$$

| SQL | | | | | | | | | | | | | | | | | | | |
|---|--|----------|--|-------------|---------------------------------------|---------------|---|-------------|---------------------------------------|-------------|--------------------------------------|-------------|-----------------------------------|----|--------------------|---------|----------------------------|------|--|
| SELECT <i>column_name(s)</i> FROM <i>table_name</i> | Select data from a table. | | | | | | | | | | | | | | | | | | |
| SELECT * FROM <i>table_name</i> | Select all data from a table. | | | | | | | | | | | | | | | | | | |
| SELECT DISTINCT <i>column_name(s)</i> FROM <i>table_name</i> | Select only distinct (different) data from a table. | | | | | | | | | | | | | | | | | | |
| SELECT <i>column_name(s)</i> FROM <i>table_name</i> WHERE <i>column operator value</i> AND <i>column operator value</i> OR <i>column operator value</i> AND (... OR ...) ... | Select only certain data from a table. | | | | | | | | | | | | | | | | | | |
| | <table> <tr> <th>Operator</th><th></th></tr> <tr> <td>=</td><td>Equal</td></tr> <tr> <td><></td><td>Not equal</td></tr> <tr> <td>></td><td>Greater than</td></tr> <tr> <td><</td><td>Less than</td></tr> <tr> <td>>=</td><td>Greater than or equal</td></tr> <tr> <td><=</td><td>Less than or equal</td></tr> <tr> <td>BETWEEN</td><td>Between an inclusive range</td></tr> <tr> <td>LIKE</td><td>Search for a pattern. A "%" sign can be used to define wildcards (missing letters in the pattern)</td></tr> </table> | Operator | | = | Equal | <> | Not equal | > | Greater than | < | Less than | >= | Greater than or equal | <= | Less than or equal | BETWEEN | Between an inclusive range | LIKE | Search for a pattern. A "%" sign can be used to define wildcards (missing letters in the pattern) |
| Operator | | | | | | | | | | | | | | | | | | | |
| = | Equal | | | | | | | | | | | | | | | | | | |
| <> | Not equal | | | | | | | | | | | | | | | | | | |
| > | Greater than | | | | | | | | | | | | | | | | | | |
| < | Less than | | | | | | | | | | | | | | | | | | |
| >= | Greater than or equal | | | | | | | | | | | | | | | | | | |
| <= | Less than or equal | | | | | | | | | | | | | | | | | | |
| BETWEEN | Between an inclusive range | | | | | | | | | | | | | | | | | | |
| LIKE | Search for a pattern. A "%" sign can be used to define wildcards (missing letters in the pattern) | | | | | | | | | | | | | | | | | | |
| SELECT <i>column_name(s)</i> FROM <i>table_name</i> WHERE <i>column_name</i> IN (<i>value1, value2, ...</i>) | The IN operator may be used if you know the exact value you want to return for at least one of the columns. | | | | | | | | | | | | | | | | | | |
| SELECT <i>column_name(s)</i> FROM <i>table_name</i> ORDER BY <i>row_1, row_2 DESC, row_3 ASC, ...</i> | Select data from a table with sort the rows. ASC (ascend) is a alphabetical and numerical order (optional) DESC (descend) is a reverse alphabetical and numerical order | | | | | | | | | | | | | | | | | | |
| SELECT <i>column_1, ..., AGGREGATE_FUN(agg_column_name)</i> FROM <i>table_name</i> GROUP BY <i>group_column_name</i> | Without the GROUP BY all the tuples in the table are grouped together | | | | | | | | | | | | | | | | | | |
| | <table> <tr> <th>Function</th><th></th></tr> <tr> <td>AVG(column)</td><td>Returns the average value of a column</td></tr> <tr> <td>COUNT(column)</td><td>Returns the number of rows (without a NULL value) of a column</td></tr> <tr> <td>MAX(column)</td><td>Returns the highest value of a column</td></tr> <tr> <td>MIN(column)</td><td>Returns the lowest value of a column</td></tr> <tr> <td>SUM(column)</td><td>Returns the total sum of a column</td></tr> </table> | Function | | AVG(column) | Returns the average value of a column | COUNT(column) | Returns the number of rows (without a NULL value) of a column | MAX(column) | Returns the highest value of a column | MIN(column) | Returns the lowest value of a column | SUM(column) | Returns the total sum of a column | | | | | | |
| Function | | | | | | | | | | | | | | | | | | | |
| AVG(column) | Returns the average value of a column | | | | | | | | | | | | | | | | | | |
| COUNT(column) | Returns the number of rows (without a NULL value) of a column | | | | | | | | | | | | | | | | | | |
| MAX(column) | Returns the highest value of a column | | | | | | | | | | | | | | | | | | |
| MIN(column) | Returns the lowest value of a column | | | | | | | | | | | | | | | | | | |
| SUM(column) | Returns the total sum of a column | | | | | | | | | | | | | | | | | | |
| SELECT <i>column_1, ..., AGGREGATE_FUN(agg_column_name)</i> FROM <i>table_name</i> GROUP BY <i>group_column_name</i> HAVING SUM (<i>group_column_name</i>) <i>condition value</i> | HAVING... was added to SQL because the WHERE keyword could not be used against aggregate functions (like SUM), and without HAVING... it would be impossible to test for result conditions. | | | | | | | | | | | | | | | | | | |
| SELECT <i>column_name</i> AS <i>column_alias</i> FROM <i>table_name</i> | Column name alias | | | | | | | | | | | | | | | | | | |
| SELECT <i>table_alias.column_name</i> FROM <i>table_name</i> AS <i>table_alias</i> | Table name alias | | | | | | | | | | | | | | | | | | |
| SELECT <i>column_1_name, column_2_name, ...</i> FROM <i>first_table_name</i> JOIN <i>second_table_name</i> ON <i>first_table_name.keyfield = second_table_name.foreign_keyfield</i> | The (INNER) JOIN returns all rows from both tables where there is a match. If there are rows in first table that do not have matches in second table, those rows will not be listed. If this is not the intended behavior, use OUTER JOIN instead (RIGHT/LEFT/FULL) | | | | | | | | | | | | | | | | | | |
| <i>SQL_Statement_1</i> UNION <i>SQL_Statement_2</i> | Select all different values from <i>SQL_Statement_1</i> and <i>SQL_Statement_2</i> | | | | | | | | | | | | | | | | | | |

COGNOME**NOME****MATRICOLA****PARTE III. DOMANDE, SOLO PER 12 CFU**

a) Descrivere gli operatori fisici per la realizzazione del join.

b) Definire e confrontare gli indici ordinati clusterizzati e non clusterizzati rispetto alla struttura e al costo per eseguire operazioni di ricerca.

c) Descrivere come avviene la revoca dei privilegi su relazioni in SQL.
