| 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$^{SOGGETTO}$, 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.

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 una persona fisica oppure un'azienda, con in un caso codice fiscale e la data di nascita e nell'altro la ragione sociale e il tipo di società.

| 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) Ogni fornitore propone un unico prezzo unitario di vendita per ogni singola componente.

   FORNITORE, COMPONENTE → PREZZO

   b) Ci possono essere più componenti con lo stesso prezzo unitario.

   NON ESISTE DIPENDENZA FUNZIONALE, RELAZIONE MOLTI A MOLTI

   c) Il tipo di ogni componente è unico.

   COMPONENTE → TIPO

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

CHIAVE CANDIDATA: (B,D). CALCOLIAMO LA SUA CHIUSURA:

$\{B,D\}^+ = \{B, D, E, A, C\}$

SICCOME SIAMO TORNATI ALLA RELAZIONE INIZIALE (BD) È CHIAVE

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

**Esercizio II.A – ALGEBRA RELAZIONALE**

In riferimento al seguente schema relazionale:

SPEDIZIONE(<u>CodiceSp</u>, Mittente[SOGGETTO], Destinatario[SOGGETTO], IndirizzoSpedizione$_0$, CodiceCat[CATEGORIA], Stato)
SOGGETTO(<u>CodiceSogg</u>, Nome, Indirizzo)
CATEGORIA(<u>CodiceCat</u> *Descrizione*, Prezzo)
PACCO(<u>CodiceSp</u>[SPEDIZIONE], <u>NumPacco</u>, Peso)
LUOGO(<u>Coordinate</u>, Descrizione)
POSIZIONE(<u>CodiceSp</u>[SPEDIZIONE], <u>Coordinate</u>[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 la posizione corrente delle spedizioni contenenti pacchi di peso superiore a 5kg con stato "in viaggio" e nome del destinatario "DIBRIS UniGE".

$\pi_{\text{Coordinate}}(\text{POSIZIONE} \bowtie \sigma_{\text{Stato}=\text{in viaggio}}(\text{SPEDIZIONE}))$

2. Determinare i nomi dei soggetti che hanno utilizzato per le loro spedizioni tutte le categorie disponibili (cioè sono stati mittenti di almeno una spedizione con quella categoria, per ogni categoria).

$\pi_{\text{Nome, CodiceCat}}(\text{SOGGETTO} \bowtie \text{SPEDIZIONE}) \div \pi_{\text{CodiceCat}}(\text{CATEGORIA})$

*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(<u>CodiceSp</u>, Mittente[SOGGETTO], Destinatario[SOGGETTO], IndirizzoSpedizione$_0$, CodiceCat[CATEGORIA], Stato)
SOGGETTO(<u>CodiceSogg</u>, Nome, Indirizzo)
CATEGORIA(<u>CodiceCat</u> *Descrizione*, Prezzo)
PACCO(<u>CodiceSp</u>[SPEDIZIONE], <u>NumPacco</u>, Peso)
LUOGO(<u>Coordinate</u>, Descrizione)
POSIZIONE(<u>CodiceSp</u>[SPEDIZIONE], <u>Coordinate</u>[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 nomi dei soggetti che non hanno mai effettuato (cioè non sono mai stati mittenti di) spedizioni in categoria "express".

```
SELECT DISTINCT
FROM SOGGETTO
WHERE (CODICESOGG) NOT IN ( SELECT MITTENTE
                            FROM SPEDIZIONE
                            NATURAL JOIN
                            CATEGORIA
                            WHERE DESCRIZIONE = EXPRESS )
```

2. Determinare lo stato della spedizione con peso complessivo massimo.

```
SELECT DISTINCT STATO
FROM SPEDIZIONE NATURAL JOIN PACCO
GROUP BY CODICESP
HAVING SUM(PESO) ≥ ALL
(SELECT SUM(PESO)
  FROM PACCO
  GROUP BY CODICESP)
```

15420765101 11454232046 27401706422 75721021601 14307131652 43135317123 134115447451

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

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

$\rho_{A \leftarrow A'}$ $R(\mathrm{U}) \rightarrow R(\mathrm{U} \backslash \mathrm{A} \cup \mathrm{A}')$ $\mathrm{A} \subseteq \mathrm{U}$

## SQL

**SELECT** *column_name(s)* **FROM** *table_name*

Select data from a table.

**SELECT * FROM** *table_name*

Select all data from a table.

**SELECT DISTINCT** *column_name(s)* **FROM** *table_name*

Select only distinct (different) data from a table.

**SELECT** *column_name(s)* **FROM** *table_name*
**WHERE** *column operator value*
**AND** *column operator value*
**OR** *column operator value*
**AND (... OR ...)**
**...**

Select only certain data from a table.

Operato

| 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.<br>A "%" sign can be used to define wildcards (missing letters in the pattern) |

**SELECT** *column_name(s)* **FROM** *table_name*
**WHERE** *column_name* **IN (***value1***,** *value2***, ...)**

The IN operator may be used if you know the exact value you want to return for at least one of the columns.

**SELECT** *column_name(s)* **FROM** *table_name*
**ORDER BY** *row_1***,** *row_2* **DESC,** *row_3* **ASC, ...**

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** *column_1***, ...,**
***AGGREGATE_FUN*(***agg_column_name*)**
**FROM** *table_name*
**GROUP BY** *group_column_name*

Without the GROUP BY all the tuples in the table are gouped together

Some aggregate

| 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** *column_1***, ...,**
***AGGREGATE_FUN*(***agg_column_name*)**
**FROM** *table_name*
**GROUP BY** *group_column_name*
**HAVING**
**SUM(***group_column_name***)** *condition value*

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** *column_name* **AS** *column_alias*
**FROM** *table_name*

Column name alias

**SELECT** *table_alias.column_name*
**FROM** *table_name* **AS** *table_alias*

Table name alias

**SELECT** *column_1_name***,** *column_2_name***, ...**
**FROM** *first_table_name*
**JOIN** *second_table_name*
**ON** *first_table_name.keyfield* **=** *second_table_name.foreign_keyfield*

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)

*SQL_Statement_1*
**UNION**
*SQL_Statement_2*

Select all different values from *SQL_Statement_1* and *SQL_Statement_2*

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

**PARTE III. DOMANDE, SOLO PER 12 CFU**

a) Descrivere gli operatori fisici per la realizzazione della selezione.

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

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