Query utenti esperti

(a) Visualizzare la descrizione delle foto che l'utente ha inserito e, qualora siano presenti apprezzamenti, nome e cognome degli utenti che hanno fatto l'apprezzamento.

```
SELECT foto.email_utente as pubblicato_da, foto.descrizione_foto, giudizio_foto.email_utente as
        ha_giudicato, giudizio_foto.commento

        FROM `foto` LEFT OUTER JOIN giudizio_foto on foto.id_cinguettio =
                giudizio_foto.id_cinguettio and foto.email_utente = giudizio_foto.utente_giudicato
```

(b) Dato un utente esperto A e una data, individuare i cinguettii di tipo “foto" che sono stati pubblicati da utenti seguiti da A e che abbiano ricevuto almeno un apprezzamento da A.

```
SELECT giudizio_foto.utente_giudicato, giudizio_foto.id_cinguettio, giudizio_foto.email_utente,
                giudizio_foto.commento

        FROM (seguaci_seguiti JOIN utente on seguaci_seguiti.utente_seguace = utente.email_utente )
                JOIN giudizio_foto ON utente_seguace = giudizio_foto.email_utente

                WHERE utente.utente_esperto = 1 AND utente.data_esperto IS NOT Null AND
                        utente_giudicato = seguaci_seguiti.utente_seguito
```

(c) Dato un utente A, riportare per ogni giorno il numero di cinguettii ricevuti da utenti esperti seguiti da A.

```
SELECT utente_seguace, COUNT(*) as numero_cinguettii

        FROM (seguaci_seguiti JOIN utente on seguaci_seguiti.utente_seguito = utente.email_utente ) JOIN
                cinguettio ON seguaci_seguiti.utente_seguito = cinguettio.email_utente

                WHERE utente.email_utente IN

                        ( SELECT email_utente FROM utente WHERE utente_esperto = 1) GROUP BY
                                utente_seguace, DAYOFMONTH(data_ora_cinguettio)
```

(d) Dato un utente esperto A, determinare l'utente seguito da A che ha postato il maggior numero di cinguettii nell'ultimo mese.

```
SELECT utente_seguace, utente_seguito, MAX(conteggio) as  numero_cinguettii

    FROM ( SELECT seguaci_seguiti.utente_seguace, seguaci_seguiti.utente_seguito, COUNT(*) as
            conteggio

                FROM cinguettio JOIN seguaci_seguiti ON cinguettio.email_utente =
                    seguaci_seguiti.utente_seguito

                    WHERE seguaci_seguiti.utente_seguace in
                        (SELECT email_utente FROM utente WHERE utente_esperto = 1)
                            GROUP by seguaci_seguiti.utente_seguace,
                            seguaci_seguiti.utente_seguito, extract(YEAR_MONTH
                            from cinguettio.data_ora_cinguettio) ) as t

                        GROUP by utente_seguace
```

(e) Creare la lista degli utenti top. Un utente considerato top se ha un elevato numero di utenti che lo seguono e ha un ridotto numero di messaggi che sono stati considerati inappropiati. definire una formula che permetta di mettere insieme queste due proprieta.

*definiamo un utente top, un utente che ha un numero di seguaci superiore alla media ed un numero di messaggi inappropiati, inferiori a quest'ultima.

```
SELECT DISTINCT utente_seguito as utente_top FROM seguaci_seguiti WHERE utente_Seguito in

    (SELECT testo.email_utente FROM testo GROUP BY testo.email_utente HAVING count(*) <=
    ( select avg (conteggio) FROM( SELECT count(*) as conteggio FROM seguaci_seguiti GROUP by
    utente_seguito ) as t ))

    and utente_seguito in

    (SELECT utente_seguito FROM seguaci_seguiti GROUP by utente_seguito HAVING count(*) >=
    ( select avg (conteggio) FROM( SELECT count(*) as conteggio FROM seguaci_seguiti GROUP by
    utente_seguito ) as t ) )
```

Trigger

```
/*************************************************************
*
*   Nome progetto: CINGUETTIO
*           File: db_trigger.sql
*          Scopo: creazione dei trigger
*      Creato da: Donà Ismail, Urso Francesco
*           Anno: 2017/2018
*
*   Nome database: db_cinguettio
*      Tipo DBMS: MariaDB
*   Versione DBMS: 10.1.9
*
*     Web server: Apache 2.4.17
*
*
*************************************************************/

-- seleziono il database
USE db_cinguettio;

/*
* ========= SOLUZIONE VINCOLI V12, V13, V14, V36, V37 ===========
*/
```

V12: L'attributo **id_cinguettio** della relazione **cinguettio** è un valore intero che deve seguire una numerazione progressiva per ogni utente. **[trigger]**

V13: L'attributo **data_ora_cinguettio** della relazione **cinguettio** corrisponde alla data e all'ora corrente nel momento in cui viene creato il cinguettio. **[trigger]**

V14: L'attributo **tipo_cinguettio** della relazione **cinguettio** può assumere uno tra i seguenti valori {'T', 'F', 'L'} dove 'T' indica che si tratta di un cinguettio di tipo testo mentre 'F' che si tratta di un cinguettio di tipo foto 'L' che si tratta di un cinguettio di tipo luogo. **[trigger]**

V36: La relazione **cinguettio** deve essere alternativamente associata con la relazione **foto, luogo** o **test**. **[trigger]**

V37: Quando la relazione **cinguettio** è associata con la relazione **luogo** il valore dell'attributo **tipo_cinguettio** deve essere impostato a 'L'. Quando la relazione **cinguettio** è associata con la relazione **testo** il valore dell'attributo **tipo_cinguettio** deve essere impostato a 'T'. Quando invece la relazione **cinguettio** è associata con la relazione **luogo** il valore dell'attributo **tipo_cinguettio** deve essere impostato a 'L'. **[trigger]**

```
-- inserimento cinguettio di tipo testo
DELIMITER //
CREATE TRIGGER vincolo_cinguettio_testo
BEFORE INSERT ON testo
FOR EACH ROW
        BEGIN

                DECLARE max_id_cinguettio INTEGER(11);

                -- massimo id_cinguettio da cinguettio
                SET max_id_cinguettio = (
                        SELECT MAX(id_cinguettio)
                        FROM cinguettio
                        WHERE email_utente = NEW.email_utente
                );
```

```
		-- caso in cui nessun cinguettio sia stato pubblicato
		-- dall'utente
		IF (max_id_cinguettio IS NULL) THEN
			SET NEW.id_cinguettio = 1;
			INSERT INTO cinguettio
			SET
				email_utente = NEW.email_utente,
				id_cinguettio = NEW.id_cinguettio,
				data_ora_cinguettio = DEFAULT, -- data corrente
				tipo_cinguettio = 'T';


		-- caso in cui almeno un cinguettio è stato pubblicato
		-- dall'utente
		ELSEIF (max_id_cinguettio IS NOT NULL) THEN
			SET NEW.id_cinguettio = max_id_cinguettio + 1;
			INSERT INTO cinguettio
			SET
				email_utente = NEW.email_utente,
				id_cinguettio = NEW.id_cinguettio,
				data_ora_cinguettio = DEFAULT,
				tipo_cinguettio = 'T';

		END IF;

	END //
DELIMITER ;

-- inserimento cinguettio di tipo luogo
DELIMITER //
CREATE TRIGGER vincolo_cinguettio_luogo
BEFORE INSERT ON luogo
FOR EACH ROW
	BEGIN

		DECLARE max_id_cinguettio INTEGER(11);

		-- massimo id_cinguettio da cinguettio
		SET max_id_cinguettio = (
			SELECT MAX(id_cinguettio)
			FROM cinguettio
			WHERE email_utente = NEW.email_utente
		);

		-- caso in cui nessun cinguettio sia stato pubblicato
		-- dall'utente
		IF (max_id_cinguettio IS NULL) THEN
			SET NEW.id_cinguettio = 1;
			INSERT INTO cinguettio
			SET
				email_utente = NEW.email_utente,
				id_cinguettio =  NEW.id_cinguettio,
				data_ora_cinguettio = DEFAULT, -- data corrente
				tipo_cinguettio = 'L';


		-- caso in cui almeno un cinguettio è stato pubblicato
		-- dall'utente
		ELSEIF (max_id_cinguettio IS NOT NULL) THEN
			SET NEW.id_cinguettio = max_id_cinguettio + 1;
```

```
                INSERT INTO cinguettio
                SET
                        email_utente = NEW.email_utente,
                        id_cinguettio = NEW.id_cinguettio,
                        data_ora_cinguettio = DEFAULT,
                        tipo_cinguettio = 'L';

        END IF;

    END //
DELIMITER ;

-- inserimento cinguettio di tipo foto
DELIMITER //
CREATE TRIGGER vincolo_cinguettio_foto
BEFORE INSERT ON foto
FOR EACH ROW
    BEGIN

        DECLARE max_id_cinguettio INTEGER(11);

        -- massimo id_cinguettio da cinguettio
        SET max_id_cinguettio = (
                SELECT MAX(id_cinguettio)
                FROM cinguettio
                WHERE email_utente = NEW.email_utente
        );

        -- caso in cui nessun cinguettio sia stato pubblicato
        -- dall'utente
        IF (max_id_cinguettio IS NULL) THEN
                SET NEW.id_cinguettio = 1;
                INSERT INTO cinguettio
                SET
                        email_utente = NEW.email_utente,
                        id_cinguettio = NEW.id_cinguettio,
                        data_ora_cinguettio = DEFAULT, -- data corrente
                        tipo_cinguettio = 'F';


        -- caso in cui almeno un cinguettio è stato pubblicato
        -- dall'utente
        ELSEIF (max_id_cinguettio IS NOT NULL) THEN
                SET NEW.id_cinguettio = max_id_cinguettio + 1;
                INSERT INTO cinguettio
                SET
                        email_utente = NEW.email_utente,
                        id_cinguettio = NEW.id_cinguettio,
                        data_ora_cinguettio = DEFAULT,
                        tipo_cinguettio = 'F';

        END IF;

    END //
DELIMITER ;



/*
```

* ========= SOLUZIONE VINCOLO V16, V17 ========================
*/

V16: L'attributo **inappropriato** della relazione **testo** è un valore booleano che assume valore 'TRUE' quando il cinguettio di tipo testo di un utente seguito viene segnalato come "inappropriato" da un altro utente altrimenti assume valore 'FALSE'. **[trigger]**

V17: L'attributo **oscurato** della relazione **testo** è un valore booleano che assume valore 'TRUE' nel caso in cui un cinguettio di tipo testo venga segnalato come inappropriato più di 3 volte altrimenti assume valore 'FALSE'. **[trigger]**

```
-- vincolo per un cinguettio di testo segnalato come
-- inappropriato
DELIMITER //
CREATE TRIGGER vincolo_cinguettio_inappropriato_oscurato
AFTER INSERT ON segnala_testo
FOR EACH ROW

	label_uscita:
	BEGIN

		DECLARE msg_gia_oscurato BOOLEAN;
		DECLARE msg_gia_inappropriato BOOLEAN;
		DECLARE numero_di_segnalazioni INTEGER(11);

		-- setto la variabile msg_gia_oscurato al valore
		-- di oscurato
		SET msg_gia_oscurato = (
			SELECT oscurato
			FROM testo
			WHERE
				email_utente =  NEW.utente_segnalato AND
				id_cinguettio =  NEW.id_cinguettio
		);

		-- se il messaggio di testo è già oscurato
		-- non continuo
		IF(msg_gia_oscurato = TRUE) THEN
			LEAVE label_uscita;
		END IF;

		-- setto la variabile msg_gia_segnalato al valore di inappropriato
		-- di testo
		SET msg_gia_inappropriato = (
			SELECT inappropriato
			FROM testo
			WHERE
				email_utente =  NEW.utente_segnalato AND
				id_cinguettio =  NEW.id_cinguettio
		);

		-- caso in cui il messaggio di testo sia stato già segnalato
		IF(msg_gia_inappropriato = TRUE) THEN

			-- calcolo il numero di segnalazioni
			SET numero_di_segnalazioni = (
				SELECT COUNT(*)
				FROM segnala_testo
				WHERE
					utente_segnalato = NEW.utente_segnalato AND
```

```
                                id_cinguettio = NEW.id_cinguettio

                        );

                        -- aggiorno oscurato di utente se la condizione
                        -- è verificata
                        IF (numero_di_segnalazioni > 3) THEN
                                UPDATE testo
                                SET
                                        oscurato = TRUE
                                WHERE
                                        email_utente = NEW.utente_segnalato AND
                                        id_cinguettio = NEW.id_cinguettio
                                ;
                        END IF;

                        -- termino l'esecuzione
                        LEAVE label_uscita;

                END IF;

                -- aggiorno il valore di inappopriato
                UPDATE testo
                SET
                        inappropriato = TRUE
                WHERE
                        email_utente = NEW.utente_segnalato AND
                        id_cinguettio = NEW.id_cinguettio;

        END //
DELIMITER ;

/*
* ========= SOLUZIONE VINCOLI V23, V24, V25, V38 ================
*/
```

V23: L'utente che ha almeno altri 3 utenti che lo seguono è definito “esperto". **[trigger]**

V24: Se un utente esperto ha un numero di seguaci inferiore a 3, smette di essere un utente esperto. **[trigger]**

V25: L’attributo **data_esperto** della relazione **utente** corrisponde alla data corrente nel momento in cui un utente diventa un utente esperto altrimenti assume valore NULL. **[trigger]**

V38: Il valore dell’attributo **utente_esperto** della relazione **utente** assume valore ‘TRUE’ quando un utente è seguito da almeno altri 3 utenti altrimenti vale ‘FALSE’. **[trigger]**

```
-- vincolo per l'utente che diventa esperto
DELIMITER //
CREATE TRIGGER vincolo_utente_diventa_esperto
AFTER UPDATE ON seguaci_seguiti
FOR EACH ROW

        label_uscita:
        BEGIN

                DECLARE numero_seguaci INTEGER(11);
                DECLARE esperto BOOLEAN;
```

```
		-- considero solo il caso in cu una richiesta appesa
		-- sia stata accettata
		IF(NEW.richiesta_accettata = TRUE AND OLD.richiesta_accettata = FALSE) THEN

			-- verifico se l'utente è già un utente esperto
			SET esperto = (
				SELECT utente_esperto
				FROM utente
				WHERE email_utente = NEW.utente_seguito
			);

			-- nel caso l'utente sia già esperto non continuo
			IF (esperto = TRUE) THEN
				LEAVE label_uscita;
			END IF;

			-- conto il numero di utenti seguaci
			SET numero_seguaci = (
				SELECT COUNT(*)
				FROM seguaci_seguiti
				WHERE
					utente_seguito = NEW.utente_seguito AND
					richiesta_accettata = TRUE
			);

			-- aggiorno i valori di utente_esperto e data_esperto
			-- se la condizione è verificata
			IF (numero_seguaci >= 3) THEN
				UPDATE utente
				SET
					utente_esperto = TRUE,
					data_esperto = (CURDATE())
				WHERE email_utente = NEW.utente_seguito;
			END IF;

		END IF;

	END //
DELIMITER ;

-- vincolo per l'utente che non è più esperto
CREATE TRIGGER vincolo_utente_non_piu_esperto
AFTER DELETE ON seguaci_seguiti
FOR EACH ROW

	-- aggiorno utente_esperto e data_esperto nel caso l'utente
	-- non sia più esperto
	UPDATE utente
	SET
		utente_esperto = FALSE,
		data_esperto = NULL
	WHERE email_utente = OLD.utente_seguito AND
		(
			SELECT COUNT(*) -- se l'utente non esiste COUNT(*) restituisce 0
			FROM seguaci_seguiti
			WHERE
				utente_seguito = OLD.utente_seguito AND
				richiesta_accettata = TRUE -- conto solo le richieste accettate
		) < 3
;
```

/*
* ========= SOLUZIONE VINCOLO V39 ==============================
*/

V39: Una volta che un utente si registra al sistema l'attributo **email_utente** della relazione **utente** dev'essere inserito all'interno della relazione **info_utente**. **[trigger]**

```
-- vincolo che garantisce ad ogni utente di avere una pagina
-- info_utente all'atto dell'iscrizione (*vedere schema er ristrutturato)
CREATE TRIGGER vincolo_inserimento_utente_in_info_utente
AFTER INSERT ON utente
FOR EACH ROW

        -- inserimento di email_utente in info_utente
        -- tutti gli altri valori sono impostati al valore di
        -- default ovvero NULL
        INSERT INTO info_utente(email_utente)
        VALUES (NEW.email_utente)

;
```