LABORATORIO DI BASE DI DATI: ***PROGETTO “COLLECTORS”***

**Gruppo di lavoro:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Matricola** | **Nome** | **Cognome** | **Contributo al progetto** |
| 288804 | Francesca | Ciccarelli |  |
| 246996 | Federico | Falcone |  |
| 278439 | Gea | Viozzi |  |

**Data di consegna del progetto:**

* **Analisi dei requisiti**

l database "Collectors" memorizza informazioni relative a collezioni di dischi. Nel database andranno registrati i dati relativi ai collezionisti (nickname, email, anagrafica) e i dati relativi alle loro collezioni (ogni collezionista può creare più collezioni, ognuna con un nome distinto). Ogni collezione è formata da dischi, per i quali abbiamo differenziato la tipologia come CD, vinile, musicassetta, EP e digitale. Abbiamo inoltre indicato titolo, anno di uscita, etichetta, stato di conservazione e barcode (se presente). Ogni disco è caratterizzato inoltre da una o più immagini che possono trovarsi su fronte, retro, facciate interne, libretti ecc. Ogni disco è composto da brani (tracce), dove andranno specificati titolo, durata, esecutore/i, compositore/i e genere. Il collezionista può inoltre possedere una o più copie di uno stesso disco, magari a seguito di scambi o perchè ne prevede la rivendita. I collezionisti possono condividere la propria collezione con specifici utenti o in maniera pubblica; ogni collezione avrà quindi un flag provato/pubblico e la lista di collezionisti con cui è condivisa.

In seguito riportiamo un glossario dei termini:

|  |  |  |
| --- | --- | --- |
| **Termine** | **Descrizione** | **Collegamenti** |
| Collezionista | Utente che può creare collezioni e che può decidere se condividerle con altri utenti. Include anche i collezionisti con cui vengono condivise le collezioni. | Collezioni |
| Collezione | Insieme di dischi posseduti dal collezionista. | Collezionista, Disco |
| Disco | Oggetto contenuto in una collezione, che può essere gestito in diverse copie. | Collezione, Etichetta, Brano |
| Brano | Unità che compone un disco. | Disco, Artista |

* **Progettazione concettuale**

Nel file "Allegato1" riportiamo il diagramma ER realizzato. Di seguito riportiamo il dizionario relativo al diagramma ER:

**Entità**

|  |  |  |  |
| --- | --- | --- | --- |
| **Entità** | **Descrizione** | **Attributi** | **Identificatore** |
| Collezionista | Utente che crea collezioni | email, nickname, anagrafica¹ | nickname |
| Collezione | Insieme di dischi posseduti dal collezionista | nome, flag² | nome, nickname |
| Disco | Oggetto contenuto in una collezione | titolo, anno\_uscita, tipo³,barcode, stato\_di\_conservazione | CONTROLLARE |
| Brano | Traccia che compone un disco | ISRC, durata, titolo, genere⁴ | ISRC |
| Artista | Colui che compone e/o esegue il brano | anagrafica⁵ | CONTROLLARE |
| Solista | Singola persona che forma l’artista | nome\_dArte |  |
| Gruppo | Insieme di persone che formano l’artista | nome\_dArte |  |
| Etichetta | Azienda che produce un disco | nome, p\_iva | nome |
| Immagine | Figura presente sul/nel disco | sorgente, collocazione | sorgente |
| ¹ nome, cognome | | ⁴ rock, pop, metal, blues, musica classica, altro | |
| ² privato, pubblico | | ⁵ nome, cognome | |
| ³ CD, vinile, EP, musicassetta, digitale | |  | |

**Relazioni**

|  |  |  |  |
| --- | --- | --- | --- |
| **Relazione** | **Descrizione** | **Entità coinvolte** | **Attributi** |
| crea | Associa un collezionista in quanto possessore e creatore ad una collezione | Collezionista(1,N), Collezione(1,1) |  |
| condivide | Associa una collezione ad un collezionista con il quale è condivisa | Collezionista(0,N), Collezione(1,1) |  |
| contiene | Associa una collezione con un disco | Collezione(0,N), Disco(1,1) | quantita |
| pubblica | Associa un disco con un artista | Disco(1,1), Artista(1,N) |  |
| compone | Associa un artista ad un brano in quanto compositore | Artista(1,N), Brano(1,N) |  |
| esegue | Associa un artista ad un brano in quanto esecutore | Artista(1,N), Brano(1,N) |  |
| forma | Associa un brano ad un disco | Brano(1,1), Disco(1,1) |  |
| produce | Associa un disco ad un’etichetta | Disco(1,1), Etichetta(1,N) |  |
| raffigura | Associa un disco ad un’immagine | Disco(0,N), Immagine(1,1) |  |

**Generalizzazioni**

|  |  |  |
| --- | --- | --- |
| **Classe Padre** | **Classi Figlie** | **Tipologia della generalizzazione** |
| Artista | Solista, Gruppo | Totale |

**Formalizzazione dei vincoli non esprimibili nel modello ER**

* Il nickname associato ad ogni *Collezionista* deve essere unico e non nullo;
* Il nome della *Collezione* che viene creata dal *Collezionista* deve essere unico e non nullo;
* Il nome dell’*Etichetta* deve essere nullo;
* Il codice ISRC, chiave dell’entità *Brano*, deve essere lungo 12 caratteri;
* Il numero di *Collezioni* condivise non può superare il numero di *Collezioni* totali possedute dal *Collezionista*;
* Una *Collezionista* non può condividere la stessa *Collezione* con un utente con cui è già condivisa;
* Una *Collezione* non può essere formata da un numero di dischi superiore a quelli posseduti;
* Un *Disco* deve necessariamente far parte di una *Collezione* che è stata creata dal *Collezionista*.
* **Progettazione logica**

**Ristrutturazione ed ottimizzazione del modello ER**

Nel file "Allegato2" riportiamo il diagramma ER ristrutturato. Le ristrutturazioni che sono state effettuate sono le seguenti:

**Eliminazione degli attributi multivalore**

* L'attributo *anagrafica* dell'entità *Collezionista* è stato esploso in due nuovi attributi: *nome, cognome;*
* L'attributo *anagrafica* dell'entità *Artista* è stato esploso in due nuovi attributi: *nome*, *cognome*;
* L'attributo *par\_tecnici* dell'entità *Immagine* è stato esploso nei seguenti nuovi attributi: *nome\_file, dimensione\_file, formato\_file.*

**Eliminazione delle gerarchie**

* La generalizzazione totale con classe padre *Artista* e classi figlie *Solista* e *Gruppo* è stata eliminata; le classi figlie sono state fuse con la classe padre ed è stato aggiunto all’entità *Artista* un attributo booleano "gruppo" che sarà True se l'artista considerato è un gruppo, False se è un solista.

**Fusione/Decomposizione di entità e relazioni**

* È possibile possedere più copie di uno stesso disco, ed ogni copia può avere uno stato di conservazione diverso; per questo motivo è stata creata una nuova entità chiamata *Copia*, legata all'entità *Disco* tramite la relazione *quantizza*, che ha gli attributi *stato\_di\_conservazione* e *barcode*;
* L'attributo *tipo* legato all'entità *Disco* è specifico per ogni copia dei dischi che possiede il collezionista, poichè è possibile possedere lo stesso disco in formati/tipi diversi (ad esempio un CD ed un vinile); per questo motivo è stata creata una nuova entità *Tipo* con l'attributo *nome*, legata a *Copia* tramite la relazione *descrive*;
* Le relazioni *esegue* e *compone* che legano gli attributi *Artista* e *Brano* rappresentano le stesse informazioni, distinguendo solo se l'artista considerato è il compositore e/o esecutore del brano. È stato ristrutturato considerando una singola relazione *appartiene* che ha come attributo un flag enum(esecutore/compositore/entrambi)
* L'attributo *genere* relativo all'entità *Brano* può contenere un'ampia lista di generi musicali in continuo aggiornamento. Per gestire la nascita di nuovi generi musicali, è stata creata un'entità *Genere*, legata a *Disco* tramite la relazione *caratterizza*, che ha come attributo *nome*.

**Altre ristrutturazioni**

* L'entità debole *Disco* fa riferimento all'entità forte *Etichetta* ma, poichè un disco è solitamente associato all'artista e non alla casa discografica che lo produce, si preferisce rendere *Disco* debole con l'entità *Artista* tramite la relazione *pubblica* (che ha cardinalità sempre (1,1));
* Per facilitare la modifica dell'entità *Disco* da parte dei *Collezionisti* abbiamo creato la relazione *possiede* che associa ad un disco uno ed un solo collezionista.

**Traduzione del modello ER nel modello relazionale**

Riportiamo qui di seguito il modello relazionale relativo allo schema ER ristrutturato.

* Collezionista(ID, nickname, email, nome, cognome)
* condivisa(ID\_collezionista, ID\_collezione)
* Collezione(ID, ID\_collezionista, nome, stato)
* Artista(ID, nome\_dArte, nome, cognome, gruppo)
* Disco(ID, titolo, ID\_artista, ID\_etichetta, ID\_collezionista, ID\_genere, anno\_uscita)
* Tipo(ID, nome)
* Copia(ID, stato\_di\_conservazione, ID\_disco, ID\_tipo, ID\_collezione, barcode)
* Genere(ID, nome)
* Brano(ID, ISRC, durata, titolo, ID\_disco)
* appartiene(ID\_artista, ID\_brano, flag)
* Etichetta(ID, p\_iva, nome)
* Immagine(ID, nome, dimensione, formato, url\_sorgente, collocazione, ID\_disco)
* **Progettazione fisica**

**Implementazione del modello relazionale**

***Struttura del DB***

DROP DATABASE IF EXISTS Collectors;

CREATE DATABASE IF NOT EXISTS Collectors;

USE Collectors;

DROP TABLE IF EXISTS collezionista;

DROP TABLE IF EXISTS collezione;

DROP TABLE IF EXISTS artista;

DROP TABLE IF EXISTS disco;

DROP TABLE IF EXISTS brano;

DROP TABLE IF EXISTS copia;

DROP TABLE IF EXISTS etichetta;

DROP TABLE IF EXISTS immagine;

DROP TABLE IF EXISTS condivisa;

DROP TABLE IF EXISTS quantizza;

DROP TABLE IF EXISTS appartiene;

DROP TABLE IF EXISTS genere;

DROP TABLE IF EXISTS tipo;

-- Creiamo l'utente che accederà ai dati

DROP USER IF EXISTS 'collectorsUser'@'localhost';

CREATE USER 'collectorsUser'@'localhost' IDENTIFIED BY 'collectorsPwd';

GRANT select,insert,update,delete,execute ON Collectors.\* TO 'collectorsUser'@'localhost';

CREATE TABLE collezionista(

 ID INTEGER UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

 nickname VARCHAR(100) NOT NULL UNIQUE,

 email VARCHAR(100) NOT NULL UNIQUE,

 nome VARCHAR(100) NOT NULL,

 cognome VARCHAR(100) NOT NULL

        );

CREATE TABLE collezione(

ID INTEGER UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

nome VARCHAR(100) NOT NULL UNIQUE,

stato enum('pubblico','privato') default 'privato',

ID\_collezionista INTEGER UNSIGNED NOT NULL,

CONSTRAINT collezione\_collezionista FOREIGN KEY (ID\_collezionista)

        REFERENCES collezionista (ID)

        ON DELETE RESTRICT ON UPDATE CASCADE

        );

CREATE TABLE artista(

 ID INTEGER UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

 nome\_dArte VARCHAR(100) NOT NULL,

 nome VARCHAR(100),

 cognome VARCHAR(100),

 gruppo BOOLEAN

    );

CREATE TABLE etichetta(

 ID INTEGER UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

 p\_iva VARCHAR(100) UNIQUE,

 nome VARCHAR(100) UNIQUE NOT NULL

        );

CREATE TABLE genere(

        ID INTEGER UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

    nome VARCHAR(50)

        );

CREATE TABLE disco(

 ID INTEGER UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

 titolo VARCHAR(100) NOT NULL,

 ID\_artista INTEGER UNSIGNED NOT NULL,

 ID\_etichetta INTEGER UNSIGNED NOT NULL,

 ID\_collezionista INTEGER UNSIGNED NOT NULL,

 ID\_genere INTEGER UNSIGNED NOT NULL,

 anno\_uscita INTEGER UNSIGNED,

 CONSTRAINT disco\_artista FOREIGN KEY (ID\_artista)

        REFERENCES artista (ID)

        ON DELETE RESTRICT ON UPDATE CASCADE,

  CONSTRAINT disco\_etichetta FOREIGN KEY (ID\_etichetta)

        REFERENCES etichetta (ID)

        ON DELETE RESTRICT ON UPDATE CASCADE,

CONSTRAINT disco\_collezionista FOREIGN KEY (ID\_collezionista)

        REFERENCES collezionista (ID)

        ON DELETE RESTRICT ON UPDATE CASCADE,

CONSTRAINT disco\_genere FOREIGN KEY (ID\_genere)

        REFERENCES genere (ID)

        ON DELETE RESTRICT ON UPDATE CASCADE

        );

CREATE TABLE brano(

 ID INTEGER UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

 ISRC VARCHAR(100) NOT NULL UNIQUE,

 durata TIME,

 titolo VARCHAR(100),

 ID\_disco INTEGER UNSIGNED NOT NULL,

CONSTRAINT brano\_disco FOREIGN KEY (ID\_disco)

        REFERENCES disco (ID)

        ON DELETE RESTRICT ON UPDATE CASCADE

        );

CREATE TABLE tipo(

        ID INTEGER UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

    nome VARCHAR(50)

        );

CREATE TABLE copia(

        ID INTEGER UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

    stato\_di\_conservazione ENUM ('OTTIMO','BUONO','USURATO'),

    ID\_disco INTEGER UNSIGNED NOT NULL,

    ID\_tipo INTEGER UNSIGNED NOT NULL,

    ID\_collezione INTEGER UNSIGNED NOT NULL,

    barcode VARCHAR(100),

    CONSTRAINT copia\_disco FOREIGN KEY (ID\_disco)

                REFERENCES disco (ID)

        ON DELETE RESTRICT ON UPDATE CASCADE,

    CONSTRAINT copia\_tipo FOREIGN KEY (ID\_tipo)

        REFERENCES tipo (ID)

        ON DELETE RESTRICT ON UPDATE CASCADE,

        CONSTRAINT copia\_collezione FOREIGN KEY (ID\_collezione)

        REFERENCES collezione (ID)

        ON DELETE RESTRICT ON UPDATE CASCADE

        );

CREATE TABLE immagine(

 ID INTEGER UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

 nome VARCHAR(100),

 dimensione VARCHAR(100),

 formato VARCHAR(100),

 url\_sorgente VARCHAR(255),

 collocazione VARCHAR(100),

 ID\_disco INTEGER UNSIGNED NOT NULL,

 CONSTRAINT immagine\_disco FOREIGN KEY (ID\_disco)

                REFERENCES disco (ID)

        ON DELETE RESTRICT ON UPDATE CASCADE

        );

CREATE TABLE condivisa(

    ID\_collezionista INTEGER UNSIGNED NOT NULL,

    ID\_collezione INTEGER UNSIGNED NOT NULL,

    PRIMARY KEY (ID\_collezionista , ID\_collezione),

        CONSTRAINT condivisa\_collezione FOREIGN KEY (ID\_collezione)

                REFERENCES collezione (ID)

        ON DELETE RESTRICT ON UPDATE CASCADE,

        CONSTRAINT condivisa\_collezionista FOREIGN KEY (ID\_collezionista)

                REFERENCES collezionista (ID)

        ON DELETE RESTRICT ON UPDATE CASCADE

        );

 CREATE TABLE appartiene(

  ID\_artista INTEGER UNSIGNED NOT NULL,

  ID\_brano INTEGER UNSIGNED NOT NULL,

  flag ENUM("ESECUTORE","COMPOSITORE","ENTRAMBI"),

  PRIMARY KEY (ID\_artista , ID\_brano),

        CONSTRAINT appartiene\_artista FOREIGN KEY (ID\_artista)

                REFERENCES artista (ID)

        ON DELETE RESTRICT ON UPDATE CASCADE,

        CONSTRAINT appartiene\_brano FOREIGN KEY (ID\_brano)

                REFERENCES brano (ID)

        ON DELETE RESTRICT ON UPDATE CASCADE

        );

***Popolamento del DB***

***Query***

/\* QUERY 1: Inserimento di una nuova collezione \*/

DROP PROCEDURE IF EXISTS inserimento\_collezione;

DELIMITER $$

    CREATE PROCEDURE inserimento\_collezione(ID\_collezionista INTEGER, nome VARCHAR(100), stato ENUM("PUBBLICO","PRIVATO"))

    BEGIN

    INSERT INTO collezione(ID, ID\_collezionista, nome, stato)

    VALUES(ID, ID\_collezionista, nome, stato);

    END $$

DELIMITER ;

/\* QUERY 2a: Aggiunta di dischi ad una collezione \*/

DROP PROCEDURE IF EXISTS inserimento\_disco;

DELIMITER $$

    CREATE PROCEDURE inserimento\_disco(nome VARCHAR(100), ID\_artista INTEGER, ID\_etichetta INTEGER, ID\_collezionista INTEGER, ID\_collezione INTEGER, anno\_uscita INTEGER)

    BEGIN

    INSERT INTO disco

    VALUES(ID, nome, ID\_artista, ID\_etichetta, ID\_collezionista, ID\_collezione, anno\_uscita);

    END $$

DELIMITER ;

/\* QUERY 2b: Aggiunta di tracce ad un disco \*/

DROP PROCEDURE IF EXISTS inserimento\_traccia;

DELIMITER $$

    CREATE PROCEDURE inserimento\_traccia(ID\_disco INTEGER, ISRC VARCHAR(100), durata VARCHAR(100), titolo VARCHAR(100), ID\_genere INTEGER)

    BEGIN

    INSERT INTO brano(ID\_disco, ISRC, durata, titolo, ID\_genere)

    VALUES(ID\_disco, ISRC, durata, titolo, ID\_genere);

    END $$

DELIMITER ;

/\* QUERY 3a: Modifica dello stato di pubblicazione di una collezione (da privata a pubblica e viceversa)\*/

DROP PROCEDURE IF EXISTS modifica\_pubblicazione;

DELIMITER $$

    CREATE PROCEDURE modifica\_pubblicazione(ID1 INTEGER, flag ENUM("pubblico","privato"))

    BEGIN

    SELECT ID, stato FROM collezione WHERE collezione.ID = ID1;

    IF flag = "pubblico" THEN UPDATE collezione SET stato = "pubblico" WHERE collezione.ID = ID1;

    ELSEIF flag = "privato" THEN UPDATE collezione SET stato = "privato" WHERE collezione.ID = ID1;

    END IF;

    END $$

DELIMITER ;

/\* QUERY 3b: Aggiunta di nuove condivisioni a una collezione \*/

DROP PROCEDURE IF EXISTS inserimento\_condivisioni;

DELIMITER $$

    CREATE PROCEDURE inserimento\_condivisioni(ID\_collezione INTEGER, ID\_collezionista INTEGER)

    BEGIN

    INSERT INTO condivisa(ID\_collezione, ID\_collezionista)

    VALUES(ID\_collezione, ID\_collezionista);

    END $$

DELIMITER ;

/\* QUERY 4: Rimozione di un disco da una collezione. \*/

DROP PROCEDURE IF EXISTS rimozione\_disco;

DELIMITER $$

    CREATE PROCEDURE rimozione\_disco(ID1 INTEGER)

    BEGIN

    DELETE FROM copia WHERE ID = ID1;

    END $$

DELIMITER ;

/\* QUERY 5: Rimozione di una collezione. \*/

DROP PROCEDURE IF EXISTS rimozione\_collezione;

DELIMITER $$

    CREATE PROCEDURE rimozione\_collezione(ID1 INTEGER)

    BEGIN

    DELETE FROM collezione WHERE collezione.ID = ID1;

    END $$

DELIMITER ;

/\* QUERY 6: Lista di tutti i dischi in una collezione. \*/

DROP PROCEDURE IF EXISTS lista\_dischi;

DELIMITER $$

    CREATE PROCEDURE lista\_dischi(ID1 INTEGER)

    BEGIN

    SELECT d.ID, d.titolo, d.ID\_artista, d.ID\_etichetta, d.ID\_collezionista, d.ID\_genere, d.anno\_uscita, c.stato\_di\_conservazione, c.ID\_tipo, c.barcode FROM disco d

     JOIN copia c ON(c.ID\_disco = d.ID)

    WHERE c.ID\_collezione = ID1;

    END $$

DELIMITER ;

/\* QUERY 7: Track list di un disco. \*/

DROP PROCEDURE IF EXISTS tracklist;

DELIMITER $$

    CREATE PROCEDURE tracklist(ID1 INTEGER)

    BEGIN

    SELECT b.titolo FROM brano b

        JOIN disco d ON (b.ID\_disco = d.ID)

    WHERE d.ID = ID1;

    END $$

DELIMITER ;

/\* QUERY 8: Ricerca di dischi in base a nomi di autori/compositori/interpreti e/o titoli.

Si potrà decidere di includere nella ricerca le collezioni di un certo collezionista e/o

quelle condivise con lo stesso collezionista e/o quelle pubbliche. \*/

DROP PROCEDURE IF EXISTS ricerca\_dischi;

DELIMITER $$

CREATE PROCEDURE ricerca\_dischi(

    stringa VARCHAR(100), ID\_coll INTEGER, posseduta BOOLEAN, condivisa BOOLEAN, pubblica BOOLEAN

)

BEGIN

    IF (posseduta AND condivisa AND pubblica) THEN

    CALL ricerca1(stringa, ID\_coll) ;

    CALL ricerca2(stringa, ID\_coll) ;

    CALL ricerca3(stringa, ID\_coll);

    (SELECT \* FROM ric1) UNION DISTINCT (SELECT \* FROM ric2) UNION DISTINCT (SELECT \* FROM ric3);

    ELSEIF (posseduta AND NOT pubblica AND NOT condivisa) THEN

    CALL ricerca1(stringa, ID\_coll);

    SELECT \* FROM ric1;

    ELSEIF (condivisa AND NOT pubblica AND NOT posseduta) THEN

    CALL ricerca2(stringa, ID\_coll);

    SELECT \* FROM ric2;

    ELSEIF (pubblica AND NOT posseduta AND NOT condivisa) THEN

    CALL ricerca3(stringa, ID\_coll);

    SELECT \* FROM ric3;

    ELSEIF (posseduta AND condivisa AND NOT pubblica) THEN

    CALL ricerca1(stringa, ID\_coll);

    CALL ricerca2(stringa, ID\_coll);

    (SELECT \* FROM ric1) UNION DISTINCT (SELECT \* FROM ric2);

    ELSEIF (posseduta AND pubblica AND NOT condivisa) THEN

    CALL ricerca1(stringa, ID\_coll);

    CALL ricerca3(stringa, ID\_coll);

    (SELECT \* FROM ric1) UNION DISTINCT (SELECT \* FROM ric3);

    ELSEIF (condivisa AND pubblica AND NOT posseduta ) THEN

    CALL ricerca2(stringa, ID\_coll);

    CALL ricerca3(stringa, ID\_coll);

    SELECT \* FROM ric2 UNION DISTINCT SELECT \* FROM ric3;

    END IF;

    END $$

DELIMITER ;

DROP PROCEDURE IF EXISTS ricerca1;

DELIMITER $$

CREATE PROCEDURE ricerca1(stringa VARCHAR(100), ID\_coll INTEGER)

BEGIN

DROP TEMPORARY TABLE IF EXISTS ric1;

CREATE TEMPORARY TABLE ric1 AS

    SELECT DISTINCT d.ID, d.titolo AS disco\_titolo, a.nome\_dArte AS artista\_nome

        FROM disco d

        JOIN artista a ON d.ID\_artista = a.ID

        WHERE (d.titolo = stringa OR a.nome\_dArte = stringa)

        AND d.ID\_collezionista = ID\_coll;

END $$

DELIMITER ;

DROP PROCEDURE IF EXISTS ricerca2;

DELIMITER $$

CREATE PROCEDURE ricerca2(stringa VARCHAR(100), ID\_coll INTEGER)

BEGIN

DROP TEMPORARY TABLE IF EXISTS ric2;

CREATE TEMPORARY TABLE ric2 AS

        SELECT DISTINCT d.ID, d.titolo AS disco\_titolo, a.nome\_dArte AS artista\_nome

        FROM disco d

        JOIN artista a ON d.ID\_artista = a.ID

        JOIN copia c ON c.ID\_disco = d.ID

        WHERE (d.titolo = stringa OR a.nome\_dArte = stringa) AND (c.ID\_collezione IN(

            SELECT ID\_collezione FROM condivisa con WHERE con.ID\_collezionista = ID\_coll));

END $$

DELIMITER ;

DROP PROCEDURE IF EXISTS ricerca3;

DELIMITER $$

CREATE PROCEDURE ricerca3(stringa VARCHAR(100), ID\_coll INTEGER)

BEGIN

DROP TEMPORARY TABLE IF EXISTS ric3;

CREATE TEMPORARY TABLE ric3 AS

        SELECT DISTINCT d.ID, d.titolo AS disco\_titolo, a.nome\_dArte AS artista\_nome

            FROM disco d

            JOIN artista a ON d.ID\_artista = a.ID

            JOIN copia c ON c.ID\_disco = d.ID

            WHERE (d.titolo = stringa OR a.nome\_dArte = stringa) AND (c.ID\_collezione IN(

                    SELECT ID

                    FROM collezione

                    WHERE stato = 'pubblico'

                )

            );

END $$

DELIMITER ;

/\* QUERY 9: Verifica della visibilità di una collezione da parte di un collezionista. \*/

DROP PROCEDURE IF EXISTS check\_visibilita;

DELIMITER $$

    CREATE PROCEDURE check\_visibilita(ID1 INTEGER, ID2 INTEGER)

    BEGIN

    DECLARE p BOOL;

    SET p = false;

    SELECT 1 INTO p

    FROM collezione c

    WHERE c.ID = ID1 AND (c.stato = 'pubblico' OR c.ID\_collezionista = ID2);

    IF p = false then

    SELECT 1 INTO p

    FROM condivisa c

    WHERE c.ID\_collezione = ID1 AND c.ID\_collezionista = ID2;

    END IF;

    SELECT p;

    END $$

DELIMITER ;

/\* QUERY 10: Numero dei brani (tracce di dischi) distinti di un certo autore (compositore, musicista) presenti nelle collezioni pubbliche. \*/

DROP PROCEDURE IF EXISTS conta\_brani\_autore;

DELIMITER $$

CREATE PROCEDURE conta\_brani\_autore(autore\_id INTEGER UNSIGNED)

BEGIN

    DECLARE brani\_count INT;

    SELECT COUNT(b.ID) INTO brani\_count FROM brano b

        JOIN appartiene a ON (a.ID\_brano = b.ID)

        JOIN disco d ON (d.ID = b.ID\_disco)

        JOIN copia cop ON (cop.ID\_disco = d.ID)

        JOIN collezione c ON (c.ID = cop.ID\_collezione)

    WHERE a.ID\_artista = autore\_id AND c.stato = "pubblico";

    SELECT brani\_count;

END $$

DELIMITER ;

/\* QUERY 11: Minuti totali di musica riferibili a un certo autore (compositore, musicista) memorizzati nelle collezioni pubbliche. \*/

DROP PROCEDURE IF EXISTS conta\_minuti;

DELIMITER $$

CREATE PROCEDURE conta\_minuti(autore\_id INTEGER UNSIGNED)

BEGIN

    DECLARE minuti\_count INT;

    SELECT SUM(TIME\_TO\_SEC(b.durata)) INTO minuti\_count FROM brano b

        JOIN appartiene a ON (a.ID\_brano = b.ID)

        JOIN disco d ON (d.ID = b.ID\_disco)

        JOIN copia cop ON (cop.ID\_disco = d.ID)

        JOIN collezione c ON (c.ID = cop.ID\_collezione)

    WHERE a.ID\_artista = autore\_id AND c.stato = 'pubblico';

    SELECT SEC\_TO\_TIME(minuti\_count) AS minuti\_totali;

END $$

DELIMITER ;

/\* QUERY 12a: Statistica per numero di collezioni di ciascun collezionista. \*/

DROP PROCEDURE IF EXISTS conta\_collezioni\_per\_collezionista;

DELIMITER $$

CREATE PROCEDURE conta\_collezioni\_per\_collezionista()

BEGIN

    SELECT c.ID\_collezionista, COUNT(\*) AS numero\_collezioni

    FROM collezione c

    GROUP BY ID\_collezionista;

END $$

DELIMITER ;

/\* QUERY 12b: Statistica per numero di dischi per genere nel sistema. \*/

DROP PROCEDURE IF EXISTS conta\_dischi\_per\_genere;

DELIMITER $$

CREATE PROCEDURE conta\_dischi\_per\_genere()

BEGIN

    SELECT g.nome, COUNT(\*) AS numero\_dischi

    FROM disco d

    JOIN copia c ON (c.ID\_disco = d.ID)

    JOIN genere g ON (g.ID = d.ID\_genere)

    GROUP BY ID\_genere;

END $$

DELIMITER ;

/\* QUERY 12\_Alternativa: Statistica per numero di collezioni di uno specifico collezionista. \*/

DROP PROCEDURE IF EXISTS conta\_collezioniXcollezionista;

DELIMITER $$

CREATE PROCEDURE conta\_collezioniXcollezionista(ID\_coll INTEGER)

BEGIN

    DECLARE collezioni\_count INT;

    SELECT COUNT(c.ID) INTO collezioni\_count FROM collezione c

    WHERE c.ID\_collezionista = ID\_coll;

    SELECT collezioni\_count;

END $$

DELIMITER ;