UNIVERSITA' DEGLI STUDI DI NAPOLI PARTHENOPE

DIPARTIMENTO DI SCIENZE E TECNOLOGIE

CORSO DI BASI DI DATI E LABORATORIO

RELAZIONE PROGETTO "CASA DISCOGRAFICA"

PROPONENTI

COGNOME	NOME	MATRICOLA
SARNO	ANTONIO	0124001914
COLUCCI	ANTONIO	0124001929
DE MICHELE	JONATHAN	0124001966

DATA CONSEGNA:

27/06/2020

ANNO ACCADEMICO:

2019/2020

CATEGORIA:

Ufficio discografico

Sommario

INTRODUZIONE

Abbiamo deciso di presentare, un caso reale di una casa discografica nella gestione delle attività dei suoi artisti. Galvanizzati dalla comune passione per la musica abbiamo immaginato di essere, noi stessi, i proprietari e i gestori della casa discografica che agisce prevalentemente sul territorio nazionale, il nome della società è la "BING DISCHI" (nome fittizio).
Se alla nascita dell'idea del progetto avevamo avuto una percezione di un lavoro semplice e rapido, strada facendo ci siamo resi conto di tutt'altro, incontrando le prime difficoltà già nella gestione dei contratti degli artisti, delle loro produzioni e degli eventi a cui partecipavano.
Il compito della nostra casa discografica, la "BING DISCHI", è quello di gestire, sia dal lato amministrativo e sia da quello economico, le attività degli artisti che abbiamo sotto contratto discografico.
Obiettivo fondamentale è quello di stampare e vendere gli album, per farlo ci siamo affidati a dei distributori (digitali, streaming e distributori fisici, cioè i negozi di dischi).
Gli artisti saranno i realizzatori dei propri album, tutti ricoperti da diritti d'autore. Ogni album avrà almeno una canzone, e per ogni canzone ci sarà almeno un autore. Gran parte degli artisti parteciperà a dei concerti live organizzati da agenzie di booking e sponsorizzati con delle promozioni.
Per produrre un disco ci preoccuperemo dei costi da sostenere per la registrazione in studio e dell'eventuale realizzazione di videoclip associati alle canzoni. Infine terremo sotto controllo eventuali premi e riconoscimenti che i nostri artisti riceveranno per le canzoni prodotte.

SINTESI DEI REQUISITI

Il database deve amministrare ed analizzare la produzione, la distribuzione e le attività degli artisti sotto contratto discografico. Quando un artista firma un contratto con la casa discografica viene inserito all'interno del database e gli viene assegnato un manager, ma è da precisare che sotto la definizione di "artista" viene inteso il cantante solista ma anche la band musicale (questa differenza è notificata dal numero di componenti attribuito all'entità "Artista").

Il contratto siglato con la casa discografica avrà una data di inizio e una data di fine, generalmente della durata di 5 anni. Alla scadenza del contratto, c'è la possibilità di registrare un nuovo contratto.

Quando un artista partecipa ad un concerto viene inserita la data, il luogo e la struttura in cui si tiene l'evento; il tutto organizzato da un'agenzia di booking e pubblicizzato con una promozione (che dura 30 giorni fino al giorno prima del concerto).

Quando viene inserita una canzone, prima di essere pubblicata, viene inserita nelle registrazioni in studio, in questo caso viene riportato il nome dello studio, la tariffa pagata e la data di registrazione. Inoltre, ogni canzone avrà uno o più autori relativi alla scrittura del brano, e verrà riportata la data in cui è stata scritta.

Tutti gli album, e quindi tutte le produzioni dell'artista, nel momento in cui vengono pubblicati sono tutelati dal diritto d'autore che ha una durata di 70 anni. Una volta prodotti, gli album (al cui interno ci sono una o più canzoni) vengono distribuiti da negozi fisici e/o piattaforme in streaming.

Le canzoni possono avere uno o massimo due videoclip associati (uno filmografico e un altro testuale, quest'ultimo è sempre segnalato dalla parola inglese "LYRIC" all'interno del titolo) e per ogni videoclip verrà riportato il costo di produzione e il regista che lo ha realizzato. Inoltre, le canzoni possono anche ricevere premi sulla base di riconoscimenti per copie vendute o per aver vinto una gara competitiva.

GLOSSARIO

*In grassetto i nomi delle tabelle presenti all'interno del Diagramma Relazionale

TERMINE	DEFINIZIONE	SINONIMI	OMONIMI
Artista	Band/solista che realizza album come interprete	Performer	-
Nome d'arte (artista)	Nome con cui è conosciuto un gruppo musicale o cantante solista	Nome commerciale	Nome d'arte (autore)
Numero componenti	Numero delle persone fisiche incluse in artista	Numero partecipanti	-
Telefono artista	Contatto telefonico associato all'artista	Recapito telefonico	-
Email artista	Contatto di posta elettronica associato all'artista	Posta elettronica	-
ID artista	Codice identificativo dell'artista	-	-
Concerti	Evento live a cui partecipano gli artisti	Live	-
Nome concerto	Nome dell'evento concerto	-	-
Data concerto	Data dell'evento concerto	-	-
ID concerto	Codice identificativo relativo al concerto		
Strutture	Luogo che ospita un concerto	-	-
ID strutture	Codice identificativo relativo alla struttura	-	-
Nome struttura	Nome della struttura che ospita un concerto	-	-
Agenzia Booking	Società che si occupa dell'organizzazione del concerto	Promoter	-
Telefono agenzia	Contatto telefonico associato all'agenzia booking	-	-
Nome agenzia	Nome della società di agenzia booking	-	-
Email agenzia	Contatto di posta elettronica associato all'agenzia booking	-	-
IBAN	IBAN associato all'agenzia booking	Codice alfanumerico identificativo della	-

		banca e del conto corrente	
Promozione	Promozione associata al concerto	Pubblicità	-
Inizio promozione	Data di inizio della promozione	-	-
Tipologia	Tipologia della promozione (radio, tv, giornali)	-	-
Durata	Data di fine della promozione	-	-
ID promozione	Codice identificativo relativo alla promozione	-	-
Contratto	Contratto che la casa discografica registra con artisti e manager	Accordo giuridico	-
ID contratto	Codice identificativo relativo al contratto	-	-
Inizio contratto	Data di inizio del contratto	-	-
Durata contratto	Data di fine del contratto	-	-
Stipendio artista	Retribuzione in denaro che viene corrisposta ad un artista mensilmente	Mensilità artista	-
Stipendio manager	Retribuzione in denaro che viene corrisposta ad un manager mensilmente	Mensilità manager	-
Manager	Rappresentante degli interessi di uno o più artisti	-	-
Nome manager	Nome del manager	-	-
Cognome manager	Cognome del manager	-	-
Data nascita manager	Data di nascita del manager	-	-
Sesso manager	Sesso del manager	-	-
CF manager	Codice fiscale del manager	-	-
Album	Supporto discografico contenente canzoni di uno stesso artista	Disco	-
ID Album	Codice identificativo dell'album	-	-
Titolo	Nome dell'album	-	-
Numero canzoni	Numero di canzoni presenti nell'album	-	-
Data di uscita	Data di pubblicazione dell'album	-	-
Genere	Categoria dell'album (Rock, Pop...)	-	-
Durata album	Durata complessiva delle canzoni presenti all'interno dell'album	-	-
Formato	Tipologia di stampa dell'album	Stampa discografica	Formato (attributo)
Formato (attributo)	Tipo di formato dell'album (vinile, LP, CD, digitale)	-	Formato (tabella)
Distributore	Che si occupa di distribuire sul mercato l'album prodotto	-	-
Tipo	Tipologia di distributore (digitale, fisico, streaming)	Tipologia distributore	-
Partita IVA	Identificativo del distributore ai fini dell'imposizione fiscale	-	-

Telefono distributore	Contatto telefonico del distributore	-	-
Nome distributore	Nome sociale del distributore	-	-
Email distributore	Contatto di posta elettronica del distributore	-	-
Diritti d'autore	Istituto giuridico che tutela le opere artistiche (album discografici)	Copyright	-
Inizio diritto	Data di inizio del diritto d'autore	-	-
Durata diritto	Data di fine del diritto d'autore	-	-
Codice	Codice identificativo del diritto d'autore sull'album	-	-
Autore	Scrittore e/o compositore dell'opera artistica	-	-
Nome d'arte (autore)	Nome con cui è conosciuto un autore	Nome commerciale	Nome d'arte (artista)
Nome autore	Nome dell'autore	-	-
Cognome autore	Cognome dell'autore	-	-
Sesso autore	Sesso dell'autore	-	-
CF autore	Codice fiscale dell'autore	-	-
Data scrittura	Data in cui è stata scritta una canzone	-	-
Canzone	Componimento musicale contenuto in un album	Singolo, disco	-
ID canzone	Codice identificativo della canzone	-	-
Nome canzone	Nome della canzone	Titolo	-
Durata canzone	Durata della canzone	-	-
Videoclip	Cortometraggio televisivo che presenta una canzone con immagini e testo	Video	-
Durata videoclip	Durata del videoclip	-	-
ID video	Codice identificativo del videoclip	-	-
Nome video	Nome del videoclip	Titolo	-
Uscita video	Data di pubblicazione di un videoclip	-	-
Costo	Spesa monetaria per la produzione del video	-	-
Regista	Autore del videoclip	Autore	-
Premi	Riconoscimento di una canzone per merito e risultati ottenuti	Riconoscimento	.
Consegna premio	Data di consegno del premio	-	-
Paese	Paese del premio	Nazione	-
Guadagno	Compenso del premio	Incasso	-
Nome premio	Nome del premio	-	-
Categoria	Tipo di categoria del premio (gara musicale, numero vendite...)	-	-
Studio	Studio di registrazione	-	-

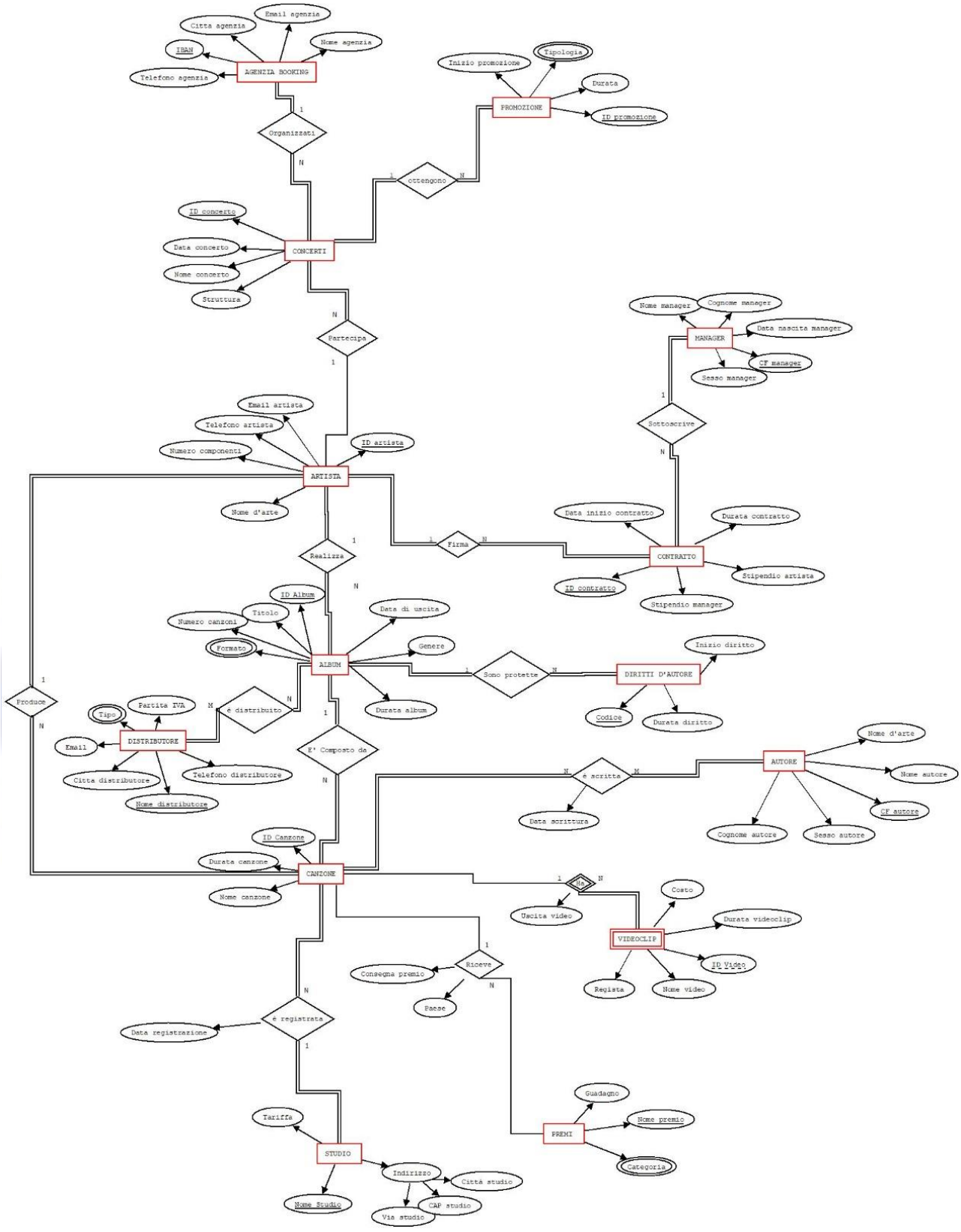
Data registrazione	Data di registrazione in studio della canzone	-	-
Tariffa	Costo dell'utilizzo dello studio di registrazione per canzone	Prezzo	-
Indirizzo	Indirizzo della sede dello studio di registrazione	-	-
ID indirizzo	Codice identificativo relativo all'indirizzo dello studio di registrazione	-	-
Codice postale	CAP in cui è inserito lo studio di registrazione	-	-
Nome studio	Nome dello studio di registrazione	-	-

DIAGRAMMA EE/R

Il modello EE/R è costituito da 14 entità:

1. AGENZIA BOOKING
2. PROMOZIONE
3. CONCERTI
4. ARTISTA
5. CONTRATTO
6. MANAGER
7. ALBUM
8. DIRITTI D'AUTORE
9. CANZONE
10. DISTRIBUTORE
11. AUTORE
12. VIDEOCLIP
13. STUDIO
14. PREMI

Nella figura è riportato il diagramma EE/R.


AGENZIA BOOKING
Citta agenzia
Email agenzia
Nome agenzia
IBAN
Telefono agenzia
Organizzati
PROMOZIONE
Inizio promozione
Tipologia
Durata
ID promozione
ottengono
CONCERTI
ID concerto
Data concerto
Nome concerto
Struttura
Partecipa
ARTISTA
Email artista
Telefono artista
Numero componenti
ID artista
Nome d'arte
MANAGER
Nome manager
Cognome manager
Data nascita manager
CF manager
Sesso manager
Sottoscrive
CONTRATTO
Data inizio contratto
Durata contratto
ID contratto
Stipendio artista
Stipendio manager
Firma
Realizza
ALBUM
ID Album
Titolo
Numero canzoni
Formato
Data di uscita
Genere
Durata album
Sono protette
DIRITTI D'AUTORE
Inizio diritto
Codice
Durata diritto
Produce
è distribuito
Tipo
Partita IVA
DISTRIBUTORE
Email
Citta distributore
Telefono distributore
Nome distributore
E' Composto da
CANZONE
ID Canzone
Durata canzone
Nome canzone
è scritta
AUTORE
Nome d'arte
Nome autore
CF autore
Cognome autore
Sesso autore
Data scrittura
Ha
Uscita video
VIDEOCLIP
Costo
Durata videoclip
ID Video
Regista
Nome video
Riceve
Consegna premio
Paese
è registrata
Data registrazione
PREMI
Guadagno
Nome premio
Categoria
STUDIO
Tariffa
Nome Studio
Indirizzo
Città studio
CAP studio
Via studio

Nello schema logico della base di dati realizzata, le entità forti sono inserite in un rettangolo, mentre le entità deboli sono all'interno di un doppio rettangolo. Gli attributi sono all'interno di ovali, le chiavi primarie sono sottolineate, mentre gli attributi multivalore sono cerchiati due volte. Non sono indicate le cardinalità massime e minime delle associazioni.

Un punto fondamentale del nostro EE/R è quello relativo al collegamento che esiste tra le entità ARTISTA, MANAGER e CONTRATTO, in quanto ogni volta che un artista firma un contratto, viene sempre aggiunto all'interno dell'accordo un manager a esso associato.

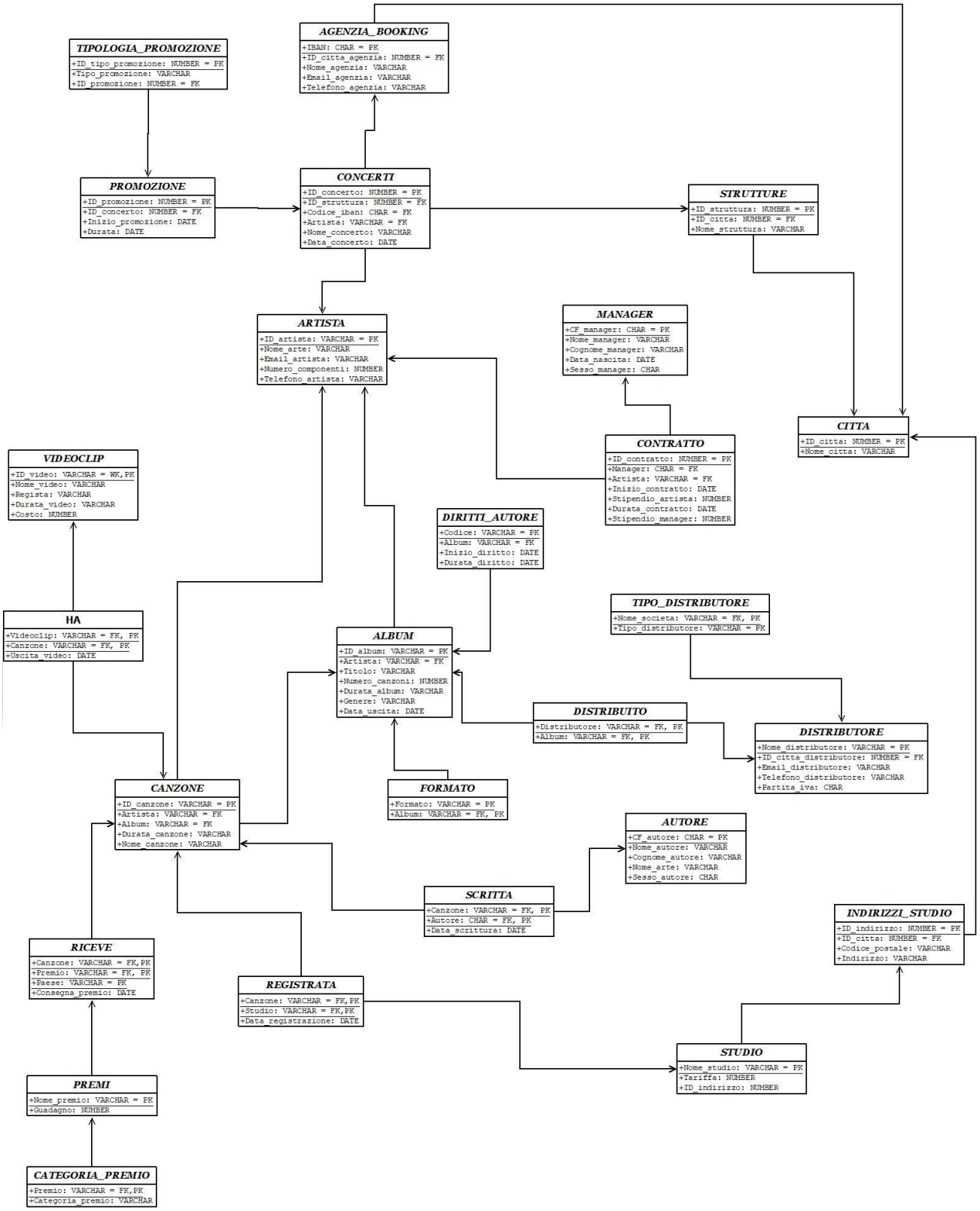
Un'altra situazione da notificare è quella che lega l'entità debole VIDEOCLIP a CANZONE, questo rapporto esiste solo se esiste una canzone da associare ad un video. Possiamo dire che non tutte le canzoni hanno videoclip associati, e che tutti i videoclip sono associati ad una canzone.

DIAGRAMMA RELAZIONALE

Nella traduzione da EE/R a relazionale, le entità e le associazioni con attributo sono mostrate come tabelle che riportano come nome quello corrispondente, mentre al loro interno sono riportate rispettivamente gli attributi dell'entità e l'attributo legato all'associazione . Inoltre sono inserite anche le chiavi delle due entità legate, che sono primarie se prese insieme, ma allo stesso tempo sono anche chiavi esterne se prese singolarmente. Esempio l'associazione `SCRITTA` riporta al suo interno l'attributo `Data_scrittura`, e le due chiavi `Canzone` e `Autore`.

Anche per gli attributi multivalore, nel passaggio da EE/R a relazionale, sono state create tabelle a parte.

Nel relazionale invece, all'attributo strutturato di un'entità gli vengono estratti i suoi sottoattributi all'interno della stessa tabella. Esempio nell'EE/R l'attributo strutturato INDIRIZZO viene tradotto nel relazionale direttamente con i suoi sottoattributi VIA, CAP, CITTA' (argomento approfondito più avanti nel paragrafo ***Prima forma normale***).


TIPOLOGIA_PROMOZIONE
+ID_tipo_promozione: NUMBER = PK
+Tipo_promozione: VARCHAR
+ID_promozione: NUMBER = FK
AGENZIA_BOOKING
+IBAN: CHAR = PK
+ID_citta_agenzia: NUMBER = FK
+Nome_agenzia: VARCHAR
+Email_agenzia: VARCHAR
+Telefono_agenzia: VARCHAR
PROMOZIONE
+ID_promozione: NUMBER = PK
+ID_concerto: NUMBER = FK
+Inizio_promozione: DATE
+Durata: DATE
CONCERTI
+ID_concerto: NUMBER = PK
+ID_struttura: NUMBER = FK
+Codice_iban: CHAR = FK
+Artista: VARCHAR = FK
+Nome_concerto: VARCHAR
+Data_concerto: DATE
STRUTTURE
+ID_struttura: NUMBER = PK
+ID_citta: NUMBER = FK
+Nome_struttura: VARCHAR
ARTISTA
+ID_artista: VARCHAR = PK
+Nome_arte: VARCHAR
+Email_artista: VARCHAR
+Numero_componenti: NUMBER
+Telefono_artista: VARCHAR
MANAGER
+CF_manager: CHAR = PK
+Nome_manager: VARCHAR
+Cognome_manager: VARCHAR
+Data_nascita: DATE
+Sesso_manager: CHAR
CITTA
+ID_citta: NUMBER = PK
+Nome_citta: VARCHAR
CONTRATTO
+ID_contratto: NUMBER = PK
+Manager: CHAR = FK
+Artista: VARCHAR = FK
+Inizio_contratto: DATE
+Stipendio_artista: NUMBER
+Durata_contratto: DATE
+Stipendio_manager: NUMBER
VIDEOCLIP
+ID_video: VARCHAR = WK,PK
+Nome_video: VARCHAR
+Regista: VARCHAR
+Durata_video: VARCHAR
+Costo: NUMBER
DIRITTI_AUTORE
+Codice: VARCHAR = PK
+Album: VARCHAR = FK
+Inizio_diritto: DATE
+Durata_diritto: DATE
TIPO_DISTRIBUTORE
+Nome_societa: VARCHAR = FK, PK
+Tipo_distributore: VARCHAR = PK
HA
+Videoclip: VARCHAR = FK, PK
+Canzone: VARCHAR = FK, PK
+Uscita_video: DATE
ALBUM
+ID_album: VARCHAR = PK
+Artista: VARCHAR = FK
+Titolo: VARCHAR
+Numero_canzoni: NUMBER
+Durata_album: VARCHAR
+Genere: VARCHAR
+Data_uscita: DATE
DISTRIBUITO
+Distributore: VARCHAR = FK, PK
+Album: VARCHAR = FK, PK
DISTRIBUTORE
+Nome_distributore: VARCHAR = PK
+ID_citta_distributore: NUMBER = FK
+Email_distributore: VARCHAR
+Telefono_distributore: VARCHAR
+Partita_iva: CHAR
CANZONE
+ID_canzone: VARCHAR = PK
+Artista: VARCHAR = FK
+Album: VARCHAR = FK
+Durata_canzone: VARCHAR
+Nome_canzone: VARCHAR
FORMATO
+Formato: VARCHAR = PK
+Album: VARCHAR = FK, PK
AUTORE
+CF_autore: CHAR = PK
+Nome_autore: VARCHAR
+Cognome_autore: VARCHAR
+Nome_arte: VARCHAR
+Sesso_autore: CHAR
SCRITTA
+Canzone: VARCHAR = FK, PK
+Autore: CHAR = FK, PK
+Data_scrittura: DATE
INDIRIZZI_STUDIO
+ID_indirizzo: NUMBER = PK
+ID_citta: NUMBER = FK
+Codice_postale: VARCHAR
+Indirizzo: VARCHAR
RICEVE
+Canzone: VARCHAR = FK,PK
+Premio: VARCHAR = FK, PK
+Paese: VARCHAR = PK
+Consegna_premio: DATE
REGISTRATA
+Canzone: VARCHAR = FK,PK
+Studio: VARCHAR = FK,PK
+Data_registrazione: DATE
STUDIO
+Nome_studio: VARCHAR = PK
+Tariffa: NUMBER
+ID_indirizzo: NUMBER
PREMI
+Nome_premio: VARCHAR = PK
+Guadagno: NUMBER
CATEGORIA_PREMIO
+Premio: VARCHAR = FK,PK
+Categoria_premio: VARCHAR

UTENTI E LE LORO CATEGORIE

UTENTE	TIPO	VOLUME	PERMESSI
casa_discografica	Admin	1	`ALL`

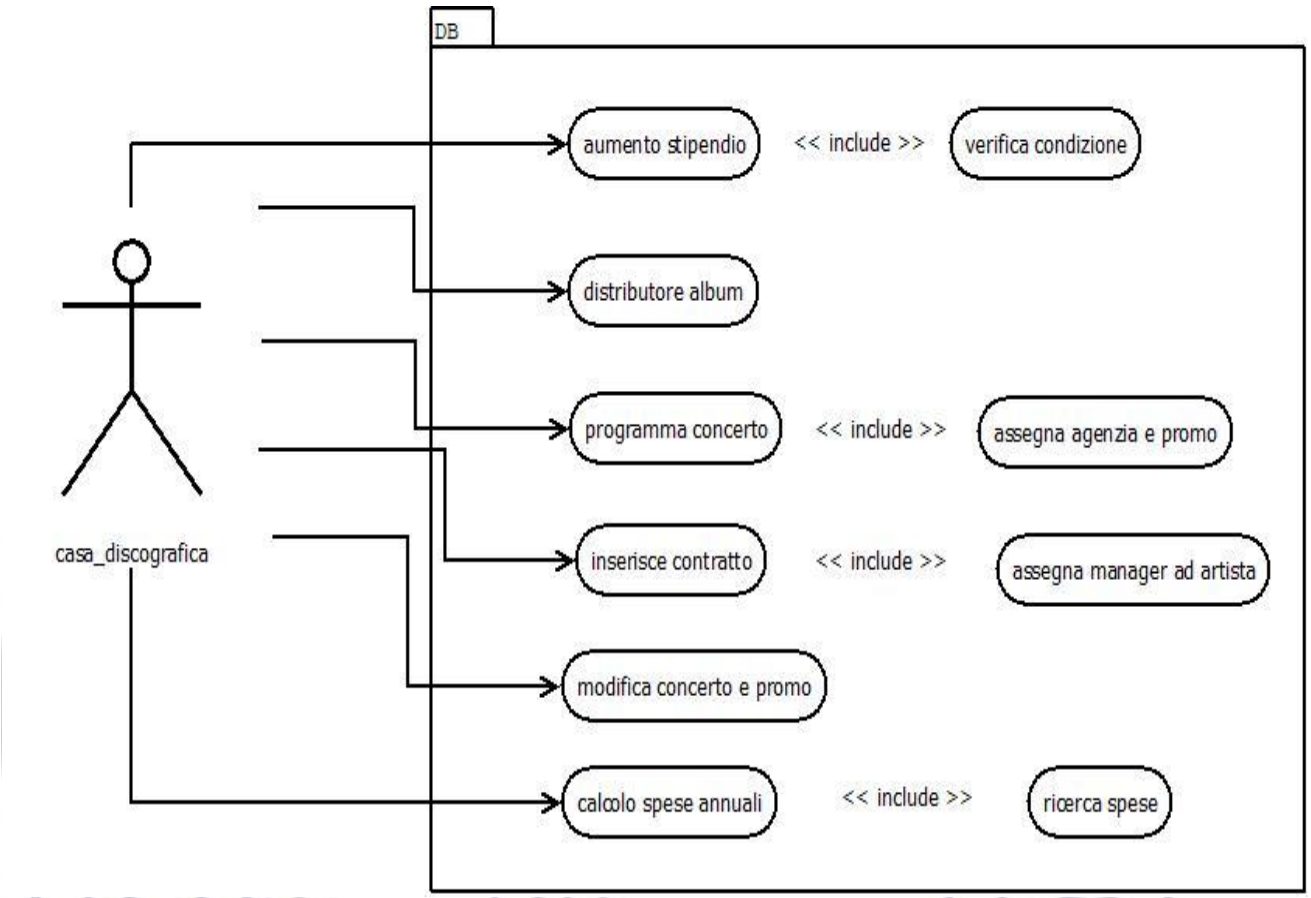
contabile	Comune	1	`SELECT ON studio;`
			`SELECT ON contratto;`
			`SELECT ON videoclip;`
			`SELECT ON tot_spese_log;`
			`EXECUTE ON tot_spese;`
			`EXECUTE ON aumento_stipendio;`

impiegato	Comune	1	`SELECT ON tipo_distributore;`
			`SELECT ON manager;`
			`SELECT ON distribuito;`
			`SELECT ON tipologia_promozione;`
			`SELECT ON artista;`
			`SELECT ON promozione;`
			`SELECT ON distributore;`
			`SELECT ON concerti;`
			`SELECT ON agenzia_booking;`
			`SELECT ON contratto;`
			`EXECUTE ON live;`
			`EXECUTE ON distributore_album;`
			`EXECUTE ON nuovo_contratto;`
			`EXECUTE ON promo_concerto;`

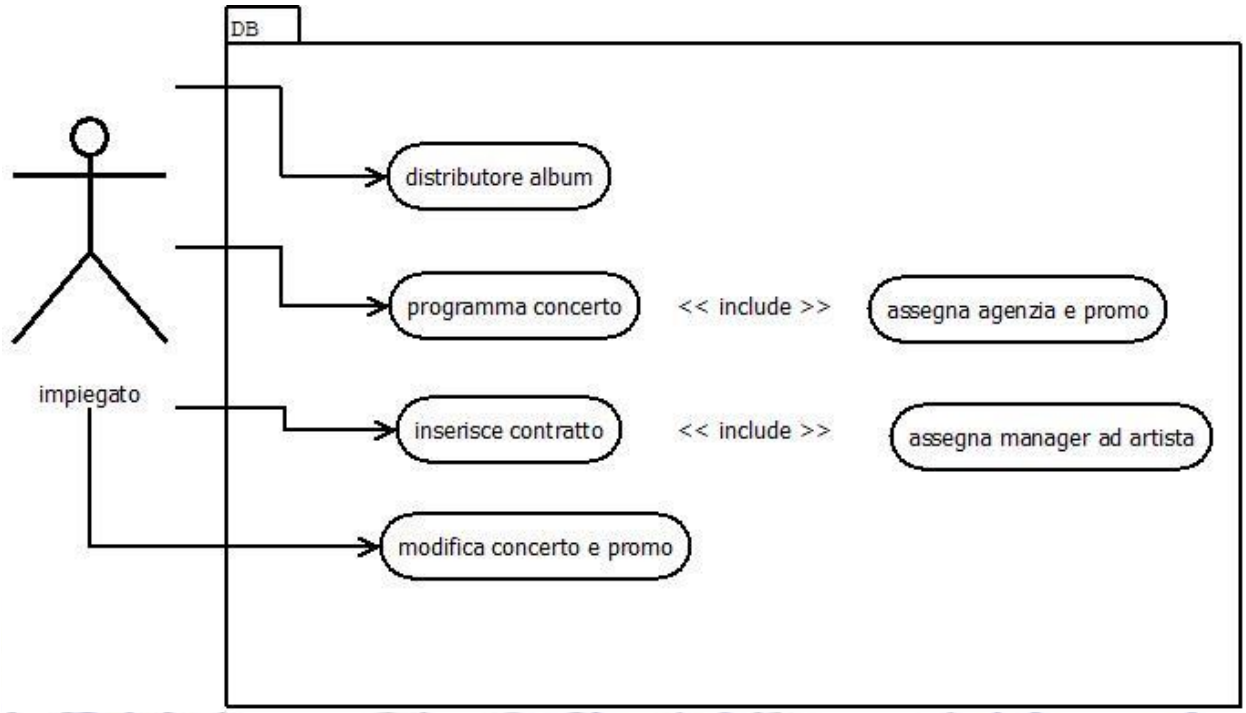
Tabella: Tavola degli utenti.

OPERAZIONI DEGLI UTENTI

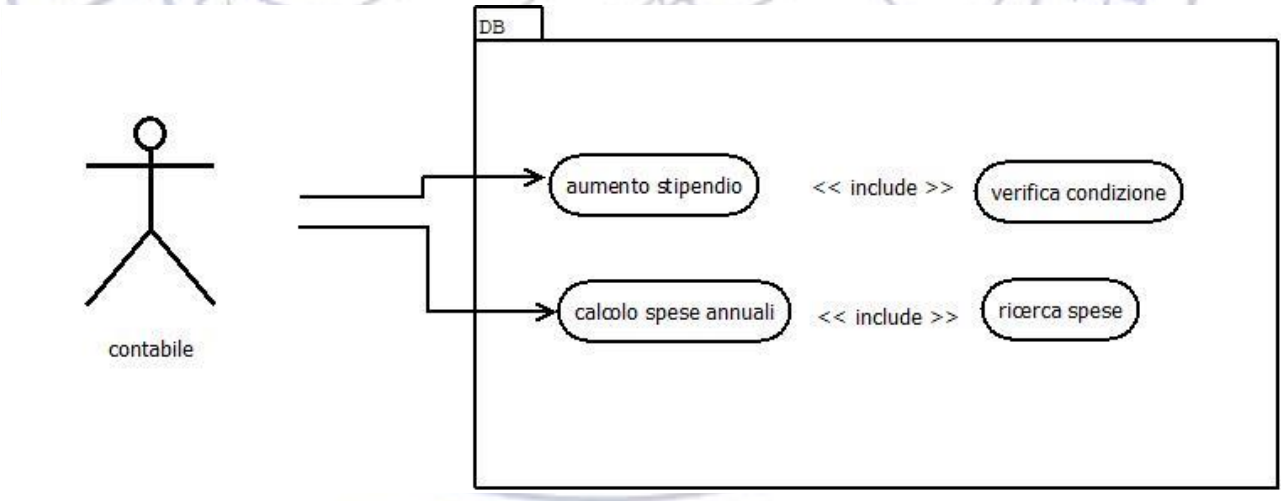
Utente : casa_discografica (fig.1)



Utente : impiegato (fig.2)



Utente : contabile (fig.3)



Descrizione delle operazioni degli utenti:

OPERAZIONE	**`aumento_stipendio`**
SCOPO:	aumento stipendio di un artista
ARGOMENTI:	id artista
RISULTATO:	aggiornamento del contratto relativo all'artista
ERRORI:	contratto scaduto condizioni non rispettate
USA:	contratto, riceve, album, artista, canzone
MODIFICA:	contratto
PRIMA:	c'è uno stipendio associato all'artista
POI:	viene aggiornato lo slot dello stipendio

OPERAZIONE	**`distributore_album`**
SCOPO:	inserisce un nuovo distributore da associare ad un album
ARGOMENTI:	id album, nome distributore, sede distributore, email distributore telefono distributore, partita iva distributore, tipo distributore
RISULTATO:	inserimento distributore
ERRORI:	errore nell'inserimento del distributore errore nell'inserimento in distribuito errore nell'inserimento nel tipo distributore errore nel conteggio id_citta errore compatibilità del formato dell'album con il tipo del distributore (trigger)
USA:	città, distributore, tipo distributore, distribuito
MODIFICA:	città, distributore, tipo distributore, distribuito
PRIMA:	manca un nuovo distributore all'album
POI:	viene aggiunto un nuovo distributore all'album

OPERAZIONE	**`live`**
SCOPO:	inserisce un nuovo concerto assegnandogli una nuova promozione e la migliore agenzia di booking
ARGOMENTI:	nome concerto, struttura concerto, data concerto, città concerto, artista
RISULTATO:	inserimento concerto e inserimento promozione
ERRORI:	errore ricerca migliore agenzia errore nell'inserimento concerto errore nell'inserimento promozione errore nell'inserimento tipologia promozione
USA:	agenzia booking, concerti, strutture, città, promozione, tipologia promozione
MODIFICA:	concerti, promozione, tipologia promozione
PRIMA:	manca un concerto
POI:	viene aggiunto un nuovo concerto

OPERAZIONE	**Nuovo_contratto**
SCOPO:	inserisce un nuovo contratto ad un artista e gli associa il manager più giovane che ha meno contratti
ARGOMENTI:	id artista, data inizio contratto, stipendio artista, durata contratto, stipendio manager
RISULTATO:	inserimento contratto
ERRORI:	errore nella selezione del manager con meno contratti e più giovane errore nell'inserimento del contratto
USA:	contratto, manager, artista (input)
MODIFICA:	contratto
PRIMA:	manca un nuovo contratto
POI:	viene aggiunto un nuovo contratto

OPERAZIONE	**Promo_concerti**
SCOPO:	quando si aggiorna un concerto modifica la struttura, e cancella la vecchia promozione con l'inserimento di una nuova
ARGOMENTI:	nome concerto, data concerto, data promozione, nuova data promozione, nuovo tipo promozione, nuova struttura, nuova durata promozione
RISULTATO:	modifica struttura concerto e tipologia promozione
ERRORI:	errore nell'aggiornamento della struttura errore nella cancellazione della tipologia promozione errore nella cancellazione della promozione errore nell'inserimento della nuova promozione errore nell'inserimento della nuova tipologia promozione
USA:	concerti, tipologia promozione, promozione, struttura
MODIFICA:	concerti, tipologia promozione, promozione
PRIMA:	c'è un concerto con struttura e promozione
POI:	viene modificata struttura e promozione del concerto

OPERAZIONE	**Tot_spese**
SCOPO:	calcolare tutte le spese* di un intero anno della casa discografica
ARGOMENTI:	anno di interesse
RISULTATO:	inserisce il totale in una tabella di log**
ERRORI:	errore nell'inserimento totale nella tabella di log errore calcolo spesa annuale costo videoclip errore calcolo spesa annuale stipendio artista e stipendio manager errore calcolo spesa tariffa studio
USA:	contratto, videoclip, ha, studio, registrata, tot_spese_log
MODIFICA:	tot_spese_log
PRIMA:	non c'è un totale spesa annuale nella tabella log
POI:	viene inserito un totale spesa annuale nella tabella log

*spese: costo videoclip, stipendio artista, stipendio manager, tariffa studio registrazione

**tabella di log: tabella creata appositamente per questa procedura

TAVOLA DELLE OPERAZIONI

OPERAZIONE	VOLUME	PERIODO
Aumento Stipendio	5	anno
Distributore album	2	mese
Programma concerto	3	settimana
Inserisci contratto	5	anno
Modifica concerto e promo	10	mese
Calcolo spese annuali	4	anno

TAVOLA DEI VOLUMI

TABELLA	TIPO	VOLUME	INCREMENTO	PERIODO
agenzia_booking	E	15	5	anno
promozione	E	26	15	mese
concerti	E	26	10	mese
artista	E	17	5	anno
contratto	E	27	10	anno
manager	E	15	5	anno
album	E	25	10	anno
formato	E	52	30	anno
diritti_autore	E	28	15	anno
distribuito	A	58	10	mese
distributore	E	15	5	anno
canzone	E	77	50	anno
ha	A	15	10	anno
videoclip	ED	15	10	anno
riceve	A	47	25	anno
premi	E	15	4	anno
registrata	A	77	50	anno
studio	E	15	4	anno
scritta	A	110	70	anno
autore	E	17	5	anno

Tabella: Tavola dei volumi. E sta per entità, ED per entità debole, mentre per le tabelle di transizione si indica una A, che sta per associazione.

ELENCO DEI VINCOLI DI INTEGRITA'

Per evitare di fare un elenco completo di tutti i vincoli, sono stati dati per ovvi quelli di chiave primaria e chiave esterna, ma riportiamo quelli più importanti per il database *discografica*. Per accertare la totalità, ove prevista, le chiavi esterne sono state rese *not null.*

STATICI

- Email, Telefono, Partita IVA e IBAN sono *unique key* nelle tabelle in cui sono presenti.
- Per gli attributi "Sesso_autore" e "Sesso_manager" c'è il constraint *check in ("m" "f" "M" "F")*
- Data uscita e Titolo *sono not null* per Album.
- Inizio diritto e durata diritto sono *not null* per la tabella "diritti d'autore".
- Nome manager e cognome manager sono *not null* per la tabella "Manager".
- Tipo distributore è *not null* per la tabella "tipo_distributore".

DINAMICI

- Le tabelle *citta, concerti, contratto, indirizzi_studio, promozione, tipologia_promozione* e *strutture* hanno come chiave gli id di tipo number che, nel momento in cui si inserisce un nuovo record all'interno di questi attributi, vengono autogenerati (in ordine numerico crescente) grazie ad un trigger.
- Un album non può contenere più di 10 canzoni.
- Un artista non può partecipare a più di un concerto in una stessa data.
- Un contratto non può avere una data di inizio diversa da quella odierna.
- E' possibile pubblicare un album solo di venerdì.
- La data di registrazione di una canzone deve essere sempre posteriore alla sua data di scrittura, e anteriore alla sua data di pubblicazione.
- Il costo di un videoclip lyric ha un costo massimo di 100 euro.
- Un album digitale non può essere distribuito da un distributore fisico e viceversa
- Quando è immesso un nuovo contratto, si aggiorna il valore dello stipendio del manager aumentandolo del 10% se è il più giovane o il più anziano.
- Non si può inserire un disco di platino se prima non ha ottenuto un disco d'oro nello stesso paese. Inoltre non si può inserire un premio Fimi se prima l'artista non ha ottenuto un disco di platino nello stesso paese.

VERIFICA DI NORMALITA' DELLO SCHEMA

Prima forma normale

Per l'entità PROMOZIONE è stato inserito un attributo multivalore TIPOLOGIA_PROMOZIONE, quest'ultimo è stato tradotto in una tabella separata avente come chiave id_tipo_promozione questo permette agli utenti di creare più tipologie di promozione di un singolo evento.

Per l'entità ALBUM è stato inserito un attributo multivalore FORMATO, quest'ultimo è tradotto in una tabella separata avente come chiavi la coppia album e formato, di cui Album è ereditata dall'entità madre, e quindi è allo stesso tempo sia PK e sia FK. Questo per far sì che un Album possa avere più di un formato (esempio, un album può essere stampato in formato CD, VINILE, e DIGITALE allo stesso tempo).

Per l'attributo multivalore TIPO_DISTRIBUTORE abbiamo creato una tabella che ha come doppia chiave nome_societa (ereditata dalla tab. DISTRIBUTORE) e tipo_distributore. Stesso discorso è stato applicato per l'attributo multivalore CATEGORIA_PREMIO, avente come chiave unica il nome del premio stesso.

E' stata infine considerato l'attributo strutturato INDIRIZZO, scomposto in Via, CAP, e città che per rispettare anche la terza forma normale saranno inseriti in altre tabelle.

Per le associazioni con attributi, nella stesura del diagramma relazionale, sono state create tabelle separate.

Per le motivazioni sopracitate il DB risulta in prima forma normale.

Seconda forma normale

Per come era stato strutturato inizialmente il DB, avevamo varie tabelle con doppie chiavi con dipendenze parziali, ma abbiamo risolto questo problema correggendole con una singola chiave.
Ad esempio, la tabella dell'entità CONTRATTO era composta dalla doppia chiave Nome_artista e Data_inizio_contratto, ma ci siamo accorti che alcuni attributi dipendevano solo da una di esse, e quindi abbiamo ovviato per la soluzione della chiave unica.

Le associazioni che invece conservano al loro interno l'utilizzo di chiavi multiple (HA, REGISTRATA, RICEVE, e SCRITTA) sono state lasciate invariate perché dipendono totalmente dalle loro chiavi primarie.

Per queste motivazioni è rispettata la seconda forma normale.

Terza forma normale e BCNF

Inizialmente avevamo all'interno della tabella CONCERTI l'attributo Struttura che dipendeva dall'attributo Citta e non dalla sua chiave; questo portava a non rispettare la terza forma normale. Per ovviare a questo problema abbiamo creato due tabelle, CITTA e STRUTTURA, dove all'interno della prima c'è l'id della città (chiave) e il nome della città, nella seconda invece troviamo l'ID_struttura (chiave), Nome_struttura, e ID_citta che viene ereditato dalla tabella città.

Quindi, grazie a questa modifica, il nome della città non è più presente come attributo nella tabella CONCERTI, ma è presente l'ID della struttura dove si può ricavare quello della città.

Stesso discorso si può fare per le tabelle AGENZIA_BOOKING e DISTRIBUTORE, che vengono localizzate geograficamente grazie alla chiave esterna id_citta.

Inoltre, un'altra correzione è stata effettuata sulla tabella STUDIO, che in un primo momento aveva l'attributo strutturato INDIRIZZO (via, cap, città) ma non rispettando la dipendenza della terza forma normale, è stato trasformato portando INDIRIZZO in una nuova tabella composta da id_indirizzo (chiave primaria), via , cap e id_citta (chiave esterna).

Tutto questo porta a definire il seguente DB in prima, seconda, terza forma normale e in BCNF.

POSSIBILI ESTENSIONI

Le estensioni possibili potrebbero essere l'aggiunta delle seguenti entità:

- UFFICIO STAMPA che gestisce le PROMOZIONI
- APPUNTAMENTI che vengono fissati dal MANAGER
- FIRMACOPIE gestiti dall'UFFICIO STAMPA

I trigger che potrebbero essere aggiunti sono diversi, e dando spazio alla fantasia, abbiamo supposto che:

- potrebbe esistere un vincolo sulla possibilità di un Artista di partecipare ad un concerto per un massimo di 10 giorni consecutivi.
- un manager non può assistere contemporaneamente a più di 5 artisti
- modificare il vincolo del trigger *trig_alb* sul numero massimo di canzoni incluse in un album da 10 a 20.

IMPLEMENTAZIONE

CREAZIONE UTENTI

```
CREATE USER casa_discografica IDENTIFIED BY musica;
CREATE USER contabile IDENTIFIED BY password1;
CREATE USER impiegato IDENTIFIED BY password2;
```

Data Definition Language

Il DDL segue lo schema relazionale, quindi le 27 tabelle sono create mediante altrettante istruzioni di CREATE TABLE con al loro interno tutti i vincoli di integrità.

```
CREATE TABLE ARTISTA
(
  ID_ARTISTA          VARCHAR2(3 BYTE)    PRIMARY KEY,
  NOME_ARTE           VARCHAR2(30 BYTE)   UNIQUE,
  EMAIL_ARTISTA       VARCHAR2(30 BYTE)   UNIQUE NOT NULL,
  NUMERO_COMPONENTI   NUMBER(1),
  TELEFONO_ARTISTA    VARCHAR2(10 BYTE)   UNIQUE
);
```

```
CREATE TABLE AUTORE
(
  CF_AUTORE         CHAR(16 BYTE)         PRIMARY KEY,
  NOME_AUTORE       VARCHAR2(15 BYTE)     NOT NULL,
  COGNOME_AUTORE    VARCHAR2(15 BYTE)     NOT NULL,
  NOME_ARTE         VARCHAR2(30 BYTE),
  SESSO_AUTORE      CHAR(1 BYTE)          CHEK IN ('M', 'm', 'F', 'f')
);
```

```
CREATE TABLE CITTA
(
  ID_CITTA     NUMBER    PRIMARY KEY,
  NOME_CITTA   VARCHAR2(50 BYTE)
);
```

```
CREATE TABLE DIRITTI_AUTORE
(
  CODICE           VARCHAR2(5 BYTE)       PRIMARY KEY,
  INIZIO_DIRITTO   DATE                   NOT NULL,
  DURATA_DIRITTO   DATE                   NOT NULL,
  ALBUM            VARCHAR2(3 BYTE)       NOT NULL


  CONSTRAINT DISCO FOREIGN KEY (ALBUM) REFERENCES ALBUM(ID_ALBUM)
);
```

```
CREATE TABLE DISTRIBUTORE
(
  NOME_DISTRIBUTORE       VARCHAR2(20 BYTE)      PRIMARY KEY,
  EMAIL_DISTRIBUTORE      VARCHAR2(30 BYTE)      UNIQUE,
  TELEFONO_DISTRIBUTORE   VARCHAR2(12 BYTE)      UNIQUE,
  PARTITA_IVA             CHAR(11 BYTE)          UNIQUE,
  ID_CITTA_DISTRIBUTORE   NUMBER
```

```
  CONSTRAINT FK_CITTA_DISTRIBUTORE FOREIGN KEY (ID_CITTA_DISTRIBUTORE) REFERENCES
CITTA(ID_CITTA)
);
```

```
CREATE TABLE INDIRIZZI_STUDIO
(
  ID_INDIRIZZO    NUMBER              PRIMARY KEY,
  CODICE_POSTALE  VARCHAR2(50 BYTE),
  INDIRIZZO       VARCHAR2(50 BYTE),
  ID_CITTA        NUMBER,

  CONSTRAINT FK_CITTA_INDIRIZZO FOREIGN KEY (ID_CITTA) REFERENCES CITTA(ID_CITTA)
);
```

```
CREATE TABLE MANAGER
(
  CF_MANAGER        CHAR(16 BYTE)      PRIMARY KEY,
  NOME_MANAGER      VARCHAR2(15 BYTE) NOT NULL,
  COGNOME_MANAGER   VARCHAR2(15 BYTE) NOT NULL,
  DATA_NASCITA      DATE,
  SESSO_MANAGER     CHAR(1 BYTE)       CHEK IN ('M', 'm', 'F', 'f')

);
```

```
CREATE TABLE PREMI
(
  NOME_PREMIO  VARCHAR2(50 BYTE)    PRIMARY KEY,
  GUADAGNO     NUMBER(4)
);
```

```
CREATE TABLE FORMATO
(
  ALBUM    VARCHAR2(3 BYTE),
  FORMATO  VARCHAR2(50 BYTE),

  CONSTRAINT PKFORMATO PRIMARY KEY (ALBUM,FORMATO),
  CONSTRAINT FKFORMATO FOREIGN KEY (ALBUM) REFERENCES ALBUM(ID_ALBUM)
);
```

```
CREATE TABLE RICEVE
(
  CANZONE           VARCHAR2(3 BYTE),
  PREMIO            VARCHAR2(15 BYTE),
  CONSEGNA_PREMIO   DATE                          NOT NULL,
  PAESE             VARCHAR2(30 BYTE),

  CONSTRAINT PKRICEVE PRIMARY KEY (CANZONE,PREMIO,PAESE),
  CONSTRAINT FKSINGOLO FOREIGN KEY (CANZONE) REFERENCES CANZONE(ID_CANZONE),
  CONSTRAINT FKPREMIO FOREIGN KEY (PREMIO) REFERENCES PREMI(NOME_PREMIO)
);
```

```
CREATE TABLE SCRITTA
(
  CANZONE          VARCHAR2(3 BYTE),
  AUTORE           CHAR(16 BYTE),
  DATA_SCRITTURA   DATE,

  CONSTRAINT PKSCRITTA PRIMARY KEY (CANZONE,AUTORE),
  CONSTRAINT FKSCRITTA FOREIGN KEY (CANZONE) REFERENCES CANZONE(ID_CANZONE),
  CONSTRAINT FKCFAUTORE FOREIGN KEY (AUTORE) REFERENCES AUTORE(CF_AUTORE)
);
```

```
CREATE TABLE STUDIO
(
  NOME_STUDIO    VARCHAR2(50 BYTE)    PRIMARY KEY,
  TARIFFA        NUMBER(4),
  ID_INDIRIZZO   NUMBER
```

```
);
```

```
CREATE TABLE TOT_SPESE_LOG
(
  DATA    DATE,
  UTENTE  VARCHAR2(30 BYTE),
  TOTALE  NUMBER,
  ANNO    VARCHAR2(15 BYTE)
);
```

```
CREATE TABLE TIPO_DISTRIBUTORE
(
  NOME_SOCIETA       VARCHAR2(20 BYTE),
  TIPO_DISTRIBUTORE  VARCHAR2(50 BYTE)          NOT NULL,

  CONSTRAINT PKDIS PRIMARY KEY (NOME_SOCIETA,TIPO_DISTRIBUTORE),
  CONSTRAINT SOCIETA FOREIGN KEY (NOME_SOCIETA) REFERENCES
DISTRIBUTORE(NOME_DISTRIBUTORE)
);
```

```
CREATE TABLE VIDEOCLIP
(
  ID_VIDEO      VARCHAR2(3 BYTE)     PRIMARY KEY,
  NOME_VIDEO    VARCHAR2(50 BYTE)    NOT NULL,
  REGISTA       VARCHAR2(30 BYTE),
  DURATA_VIDEO  VARCHAR2(30 BYTE),
  COSTO         NUMBER(3)
);
```

```
CREATE TABLE AGENZIA_BOOKING
(
  IBAN              CHAR(27 BYTE)           PRIMARY KEY,
  NOME_AGENZIA      VARCHAR2(15 BYTE),
  EMAIL_AGENZIA     VARCHAR2(30 BYTE)       UNIQUE,
  TELEFONO_AGENZIA  VARCHAR2(12 BYTE)       UNIQUE,
  ID_CITTA_AGENZIA  NUMBER,

  CONSTRAINT FK_CITTA_AGENZIA FOREIGN KEY (ID_CITTA_AGENZIA) REFERENCES
CITTA(ID_CITTA)
);
```

```
CREATE TABLE ALBUM
(
  ID_ALBUM        VARCHAR2(3 BYTE)              PRIMARY KEY,
  TITOLO          VARCHAR2(40 BYTE)             NOT NULL,
  NUMERO_CANZONI  NUMBER(2)                     NOT NULL,
  DURATA_ALBUM    VARCHAR2(15 BYTE),
  GENERE          VARCHAR2(30 BYTE),
  DATA_USCITA     DATE                          NOT NULL,
  ARTISTA         VARCHAR2(3 BYTE)              NOT NULL,

  CONSTRAINT IDARTISTA FOREIGN KEY (ARTISTA) REFERENCES ARTISTA(ID_ARTISTA)
);
```

```
CREATE TABLE CANZONE
(
  ID_CANZONE      VARCHAR2(3 BYTE)              PRIMARY KEY,
  NOME_CANZONE    VARCHAR2(50 BYTE)             NOT NULL,
  DURATA_CANZONE  VARCHAR2(50 BYTE),
  ARTISTA         VARCHAR2(3 BYTE)              NOT NULL,
  ALBUM           VARCHAR2(3 BYTE)              NOT NULL,

  CONSTRAINT FKDISCO FOREIGN KEY (ALBUM) REFERENCES ALBUM(ID_ALBUM),
  CONSTRAINT FKCANTANTE FOREIGN KEY (ARTISTA) REFERENCES ARTISTA(ID_ARTISTA)
);
```

```
CREATE TABLE CATEGORIA_PREMIO
(
```

```
  PREMIO            VARCHAR2(15 BYTE)           PRIMARY KEY,
  CATEGORIA_PREMIO  VARCHAR2(50 BYTE)           NOT NULL,

  CONSTRAINT PREMI FOREIGN KEY (PREMIO) REFERENCES PREMI(NOME_PREMIO)
);
```

```
CREATE TABLE CONCERTI
(
  ID_CONCERTO    NUMBER                   PRIMARY KEY,
  NOME_CONCERTO  VARCHAR2(50 BYTE),
  DATA_CONCERTO  DATE,
  ARTISTA        VARCHAR2(3 BYTE)         NOT NULL,
  CODICE_IBAN    CHAR(27 BYTE)            NOT NULL,
  ID_STRUTTURA   NUMBER,

  CONSTRAINT FK_STRUTTURA FOREIGN KEY (ID_STRUTTURA) REFERENCES
STRUTTURE(ID_STRUTTURA),
  CONSTRAINT FK_IBAN FOREIGN KEY (CODICE_IBAN) REFERENCES AGENZIA_BOOKING(IBAN),
  CONSTRAINT FKARTISTA FOREIGN KEY (ARTISTA) REFERENCES ARTISTA(ID_ARTISTA)
);
```

```
CREATE TABLE CONTRATTO
(
  ID_CONTRATTO       NUMBER               PRIMARY KEY,
  INIZIO_CONTRATTO   DATE,
  STIPENDIO_ARTISTA  NUMBER(4),
  DURATA_CONTRATTO   DATE                 NOT NULL,
  ARTISTA            VARCHAR2(3 BYTE)     NOT NULL,
  MANAGER            CHAR(16 BYTE)        NOT NULL,
  STIPENDIO_MANAGER  NUMBER(4),

  CONSTRAINT CONTRATTO_MAN_FK FOREIGN KEY (MANAGER) REFERENCES
MANAGER(CF_MANAGER),
  CONSTRAINT CONTRATTO_ART_FK FOREIGN KEY (ARTISTA) REFERENCES
ARTISTA(ID_ARTISTA)
);
```

```
CREATE TABLE HA
(
  VIDEOCLIP     VARCHAR2(3 BYTE),
  CANZONE       VARCHAR2(3 BYTE),
  USCITA_VIDEO  DATE                  NOT NULL,

  CONSTRAINT PKHA PRIMARY KEY (VIDEOCLIP, CANZONE),
  CONSTRAINT FKVIDEO FOREIGN KEY (VIDEOCLIP) REFERENCES VIDEOCLIP(ID_VIDEO),
  CONSTRAINT FKCANZONE FOREIGN KEY (CANZONE) REFERENCES CANZONE(ID_CANZONE)

);
```

```
CREATE TABLE PROMOZIONE
(
  ID_PROMOZIONE      NUMBER          PRIMARY KEY,
  ID_CONCERTO        NUMBER          NOT NULL,
  INIZIO_PROMOZIONE  DATE,
  DURATA             DATE,

  CONSTRAINT PROM_CONCERTO_FK FOREIGN KEY (ID_CONCERTO) REFERENCES
CONCERTI(ID_CONCERTO)
);
```

```
CREATE TABLE REGISTRATA
(
  CANZONE             VARCHAR2(3 BYTE),
  STUDIO              VARCHAR2(50 BYTE),
  DATA_REGISTRAZIONE  DATE                    NOT NULL,

  CONSTRAINT PKREGISTRATA PRIMARY KEY (CANZONE, STUDIO),
  CONSTRAINT FKSTUDIO FOREIGN KEY (STUDIO) REFERENCES STUDIO(NOME_STUDIO),
```

```
  CONSTRAINT FKREC FOREIGN KEY (CANZONE) REFERENCES CANZONE(ID_CANZONE)

);
```

```
CREATE TABLE TIPOLOGIA_PROMOZIONE
(
  ID_TIPO_PROMOZIONE  NUMBER              PRIMARY KEY,
  ID_PROMOZIONE       NUMBER,
  TIPO_PROMOZIONE     VARCHAR2(50 BYTE),

  CONSTRAINT TIPOLOGIA_PROM_FK FOREIGN KEY (ID_PROMOZIONE) REFERENCES
PROMOZIONE(ID_PROMOZIONE)
);
```

Data Manipulation Language

Il popolamento avviene attraverso operazioni dirette di INSERT e l'unico utente abilitato ad eseguirle è *discografica*, che è l'amministratore. A titolo di esempio sono qui mostrate alcune operazioni di INSERT che l'amministratore esegue.

```
-- %%%%%%%% AGGIUNTA DI UN ARTISTA
insert into artista(ID_artista, nome_arte, email_artista, numero_componenti,
telefono_artista) values
('012', 'Elisa', 'elisa77@libero.com', 1, '0810000012');

insert into artista(ID_artista, nome_arte, email_artista, numero_componenti,
telefono_artista) values
('014', 'Giorgia', 'giorgia.71@gmail.com', 1, '0810000014');

insert into artista (ID_artista, nome_arte, email_artista, numero_componenti,
telefono_artista) values
('019', 'Articolo 31', 'articolo.31@gmail.com', 2, '0810000019');
```

```
-- %%%%%%%% AGGIUNTA DI UNA CANZONE
Insert into canzone (id_canzone, nome_canzone, durata_canzone, artista, album)
values (`00Z',`Ho imparato a sognare`, `00.04.09`, `021`, `A39`);

Insert into canzone (id_canzone, nome_canzone, durata_canzone, artista, album)
values (`01Z',`Sipario`, `00.04.00`, `020`, `A36`);

Insert into canzone (id_canzone, nome_canzone, durata_canzone, artista, album)
values (`02Z',`Lezione di vita`, `00.04.50`, `020`, `A36`);
```

```
-- %%%%%%%% AGGIUNTA DI DIRITTI AUTORE
insert into diritti_autore (codice, inizio_diritto, durata_diritto, album)
values('CP001','25-set-2019','25-set-2089','A03');

insert into diritti_autore (codice, inizio_diritto, durata_diritto, album) values
('CP002','10-GEN-2020','10-GEN-2090','A04');

insert into diritti_autore (codice, inizio_diritto, durata_diritto, album) values
('CP003','11-MAG-1995','11-MAG-2065','A06');
```

```
-- %%%%%%%% AGGIUNTA DI RICEVE PREMIO
insert into riceve (canzone, premio, consegna_premio, paese) values
('17Z', 'Mia Martini', '25-mag-1993', 'Argentina');

insert into riceve (canzone, premio, consegna_premio, paese) values
('17Z', 'Mia Martini', '25-mag-1993', 'Italia');
```

```
insert into riceve (canzone, premio, consegna_premio, paese) values
('04Z', 'Sanremo', '27-feb-1991', 'Italia');
```

```
-- %%%%%%%% AGGIUNTA DI CANZONE SCRITTA
Insert into SCRITTA (CANZONE, AUTORE, DATA_SCRITTURA) Values
   ('05F', 'CODFISAUT0000002', TO_DATE('24/05/1989', 'DD/MM/YYYY'));

Insert into SCRITTA (CANZONE, AUTORE, DATA_SCRITTURA) Values
   ('10F', 'CODFISAUT0000002', TO_DATE('13/03/1982', 'DD/MM/YYYY'));

Insert into SCRITTA (CANZONE, AUTORE, DATA_SCRITTURA) Values
   ('03F', 'CODFISAUT0000002', TO_DATE('24/05/1989', 'DD/MM/YYYY'));
```

```
-- %%%%%%%% AGGIUNTA DI AGENZIA BOOKING
Insert into AGENZIA_BOOKING (IBAN, NOME_AGENZIA, EMAIL_AGENZIA, TELEFONO_AGENZIA,
ID_CITTA_AGENZIA) Values
   ('IT02L1234512345123456789012', 'Bpm concerti', 'bpmconcerti@gmail.com',
'0200123456', 2);

Insert into AGENZIA_BOOKING (IBAN, NOME_AGENZIA, EMAIL_AGENZIA, TELEFONO_AGENZIA,
ID_CITTA_AGENZIA) Values
   ('IT02L1234512345123456789013', 'Vivo concerti', 'vivoconcerti@gmail.com',
'0200123457', 5);

Insert into AGENZIA_BOOKING (IBAN, NOME_AGENZIA, EMAIL_AGENZIA, TELEFONO_AGENZIA,
ID_CITTA_AGENZIA) Values
   ('IT02L1234512345123456789014', 'Dna concerti', 'dnaconcerti@hotmail.it',
'0200123458', 1);
```

```
-- %%%%%%%% AGGIUNTA DI ALBUM
Insert into ALBUM
   (ID_ALBUM, TITOLO, NUMERO_CANZONI, DURATA_ALBUM, GENERE,
    DATA_USCITA, ARTISTA) Values
   ('A03', 'Non avere paura', 1, '00:04:07', 'Pop',
    TO_DATE('25/09/2019', 'DD/MM/YYYY'), '002');

Insert into ALBUM
   (ID_ALBUM, TITOLO, NUMERO_CANZONI, DURATA_ALBUM, GENERE,
    DATA_USCITA, ARTISTA) Values
   ('A04', 'I nostri anni', 1, '00:03:38', 'Pop',
    TO_DATE('10/01/2020', 'DD/MM/YYYY'), '002');

Insert into ALBUM
   (ID_ALBUM, TITOLO, NUMERO_CANZONI, DURATA_ALBUM, GENERE,
    DATA_USCITA, ARTISTA) Values
   ('A06', 'Ligabue', 4, '00:16:11', 'Rock and Roll',
    TO_DATE('11/05/1995', 'DD/MM/YYYY'), '003');
```

```
-- %%%%%%%% AGGIUNTA DI AUTORE
Insert into AUTORE
   (CF_AUTORE, NOME_AUTORE, COGNOME_AUTORE, NOME_ARTE, SESSO_AUTORE)
 Values
   ('CODFISAUT0000001', 'Claudio', 'Baglioni', 'Claudio Baglioni', 'M');

Insert into AUTORE
   (CF_AUTORE, NOME_AUTORE, COGNOME_AUTORE, NOME_ARTE, SESSO_AUTORE)
 Values
   ('CODFISAUT0000002', 'Vasco', 'Rossi', 'Vasco Rossi', 'M');

Insert into AUTORE
   (CF_AUTORE, NOME_AUTORE, COGNOME_AUTORE, NOME_ARTE, SESSO_AUTORE)
```

```
 Values
   ('CODFISAUT0000003', 'Tommaso', 'Paradiso', 'Tommaso Paradiso', 'M');
```

```
-- %%%%%%%% AGGIUNTA DI CATEGORIA PREMIO
Insert into CATEGORIA_PREMIO
   (PREMIO, CATEGORIA_PREMIO) Values
   ('Disco oro', 'Numero vendite');

Insert into CATEGORIA_PREMIO
   (PREMIO, CATEGORIA_PREMIO) Values
   ('Disco platino', 'Numero vendite');

Insert into CATEGORIA_PREMIO
   (PREMIO, CATEGORIA_PREMIO) Values
   ('Sanremo', 'Gara musicale');
```

```
-- %%%%%%%% AGGIUNTA DI CITTA'
Insert into CITTA
   (ID_CITTA, NOME_CITTA) Values
   (1, 'Napoli');

Insert into CITTA
   (ID_CITTA, NOME_CITTA) Values
   (2, 'Milano');

Insert into CITTA
   (ID_CITTA, NOME_CITTA) Values
   (3, 'Firenze');
```

```
-- %%%%%%%% AGGIUNTA DI CONCERTI
Insert into CONCERTI
   (ID_CONCERTO, NOME_CONCERTO, DATA_CONCERTO, ARTISTA, CODICE_IBAN,
    ID_STRUTTURA) Values
   (1, 'Ligabue tour', TO_DATE('15/05/2004', 'DD/MM/YYYY'), '003',
'IT02L1234512345123456789011',
    1);

Insert into CONCERTI
   (ID_CONCERTO, NOME_CONCERTO, DATA_CONCERTO, ARTISTA, CODICE_IBAN,
    ID_STRUTTURA) Values
   (2, 'Ligabue tour', TO_DATE('16/05/2004', 'DD/MM/YYYY'), '003',
'IT02L1234512345123456789015',
    2);

Insert into CONCERTI
   (ID_CONCERTO, NOME_CONCERTO, DATA_CONCERTO, ARTISTA, CODICE_IBAN,
    ID_STRUTTURA) Values
   (3, 'Ligabue tour', TO_DATE('17/05/2004', 'DD/MM/YYYY'), '003',
'IT02L1234512345123456789016',
    1);
```

```
-- %%%%%%%% AGGIUNTA DI CONTRATTO
Insert into CONTRATTO
   (ID_CONTRATTO, INIZIO_CONTRATTO, STIPENDIO_ARTISTA, DURATA_CONTRATTO, ARTISTA,
    MANAGER, STIPENDIO_MANAGER)
 Values
   (1, TO_DATE('20/06/2020', 'DD/MM/YYYY'), 1000, TO_DATE('01/01/2017',
'DD/MM/YYYY'), '017',
```

```
    'CODFIS0000000002', 1200);

Insert into CONTRATTO
   (ID_CONTRATTO, INIZIO_CONTRATTO, STIPENDIO_ARTISTA, DURATA_CONTRATTO, ARTISTA,
    MANAGER, STIPENDIO_MANAGER)
 Values
   (2, TO_DATE('01/01/2017', 'DD/MM/YYYY'), 1200, TO_DATE('01/01/2022',
'DD/MM/YYYY'), '002',
    'CODFIS0000000001', 1200);

Insert into CONTRATTO
   (ID_CONTRATTO, INIZIO_CONTRATTO, STIPENDIO_ARTISTA, DURATA_CONTRATTO, ARTISTA,
    MANAGER, STIPENDIO_MANAGER)
 Values
   (3, TO_DATE('05/02/2010', 'DD/MM/YYYY'), 1500, TO_DATE('05/02/2015',
'DD/MM/YYYY'), '003',
    'CODFIS0000000002', 1200);
```

```
-- %%%%%%%% AGGIUNTA DI ALBUM DISTRIBUITO
Insert into DISTRIBUITO
   (ALBUM, DISTRIBUTORE)
 Values
   ('A03', 'Awal');

Insert into DISTRIBUITO
   (ALBUM, DISTRIBUTORE)
 Values
   ('A03', 'Spotify');

Insert into DISTRIBUITO
   (ALBUM, DISTRIBUTORE)
 Values
   ('A04', 'Igtv music');
```

```
-- %%%%%%%% AGGIUNTA DI DISTRIBUTORE
Insert into DISTRIBUTORE
   (NOME_DISTRIBUTORE, EMAIL_DISTRIBUTORE, TELEFONO_DISTRIBUTORE, PARTITA_IVA,
ID_CITTA_DISTRIBUTORE)
 Values
   ('Youtube', 'youtube@gmail.com', '3380000001', 'partiva001 ', 2);

Insert into DISTRIBUTORE
   (NOME_DISTRIBUTORE, EMAIL_DISTRIBUTORE, TELEFONO_DISTRIBUTORE, PARTITA_IVA,
ID_CITTA_DISTRIBUTORE)
 Values
   ('Spotify', 'spotify@gmail.com', '3380000002', 'partiva002 ', 5);

Insert into DISTRIBUTORE
   (NOME_DISTRIBUTORE, EMAIL_DISTRIBUTORE, TELEFONO_DISTRIBUTORE, PARTITA_IVA,
ID_CITTA_DISTRIBUTORE)
 Values
   ('Apple music', 'applemusic@gmail.com', '3380000003', 'partiva003 ', 1);
```

```
-- %%%%%%%% AGGIUNTA DI FORMATO ALBUM
Insert into FORMATO
   (ALBUM, FORMATO)
 Values
   ('A03', 'Digitale');

Insert into FORMATO
   (ALBUM, FORMATO)
```

```
 Values
   ('A04', 'Digitale');

Insert into FORMATO
   (ALBUM, FORMATO)
 Values
   ('A06', 'CD');
```

```
-- %%%%%%%% AGGIUNTA DI ALBUM HA CANZONE
Insert into HA
   (VIDEOCLIP, CANZONE, USCITA_VIDEO)
 Values
   ('V01', '01A', TO_DATE('17/10/2019', 'DD/MM/YYYY'));

Insert into HA
   (VIDEOCLIP, CANZONE, USCITA_VIDEO)
 Values
   ('V02', '02A', TO_DATE('22/01/2020', 'DD/MM/YYYY'));

Insert into HA
   (VIDEOCLIP, CANZONE, USCITA_VIDEO)
 Values
   ('V03', '02A', TO_DATE('20/01/2020', 'DD/MM/YYYY'));
```

```
-- %%%%%%%% AGGIUNTA DI INDIRIZZI STUDIO REGISTRAZIONE
Insert into INDIRIZZI_STUDIO
   (ID_INDIRIZZO, CODICE_POSTALE, INDIRIZZO, ID_CITTA)
 Values
   (1, '20100', 'Via Montenapoleone', 2);

Insert into INDIRIZZI_STUDIO
   (ID_INDIRIZZO, CODICE_POSTALE, INDIRIZZO, ID_CITTA)
 Values
   (2, '80100', 'Via Toledo', 1);

Insert into INDIRIZZI_STUDIO
   (ID_INDIRIZZO, CODICE_POSTALE, INDIRIZZO, ID_CITTA)
 Values
   (3, '80078', 'Via Napoli', 1);
```

```
-- %%%%%%%% AGGIUNTA DI MANAGER
Insert into MANAGER
   (CF_MANAGER, NOME_MANAGER, COGNOME_MANAGER, DATA_NASCITA, SESSO_MANAGER)
 Values
   ('CODFIS0000000001', 'Giuseppe', 'Musso', TO_DATE('10/01/1980', 'DD/MM/YYYY'),
'M');

Insert into MANAGER
   (CF_MANAGER, NOME_MANAGER, COGNOME_MANAGER, DATA_NASCITA, SESSO_MANAGER)
 Values
   ('CODFIS0000000002', 'Maria', 'Cerrato', TO_DATE('15/02/1985', 'DD/MM/YYYY'),
'f');

Insert into MANAGER
   (CF_MANAGER, NOME_MANAGER, COGNOME_MANAGER, DATA_NASCITA, SESSO_MANAGER)
 Values
   ('CODFIS0000000003', 'Andrea', 'Ferrero', TO_DATE('24/03/1990', 'DD/MM/YYYY'),
'm');
```

```
-- %%%%%%%% AGGIUNTA DI PREMI
Insert into PREMI
   (NOME_PREMIO, GUADAGNO)
 Values
   ('Disco oro', 5000);

Insert into PREMI
   (NOME_PREMIO, GUADAGNO)
 Values
   ('Disco platino', 7000);

Insert into PREMI
   (NOME_PREMIO, GUADAGNO)
 Values
   ('Sanremo', 6000);
```

```
-- %%%%%%%% AGGIUNTA DI PROMOZIONE
Insert into PROMOZIONE
   (ID_PROMOZIONE, ID_CONCERTO, INIZIO_PROMOZIONE, DURATA)
 Values
   (1, 1, TO_DATE('14/04/2004', 'DD/MM/YYYY'), TO_DATE('14/05/2004',
'DD/MM/YYYY'));

Insert into PROMOZIONE
   (ID_PROMOZIONE, ID_CONCERTO, INIZIO_PROMOZIONE, DURATA)
 Values
   (2, 2, TO_DATE('01/05/2004', 'DD/MM/YYYY'), TO_DATE('01/06/2004',
'DD/MM/YYYY'));

Insert into PROMOZIONE
   (ID_PROMOZIONE, ID_CONCERTO, INIZIO_PROMOZIONE, DURATA)
 Values
   (3, 9, TO_DATE('05/01/2016', 'DD/MM/YYYY'), TO_DATE('04/02/2016',
'DD/MM/YYYY'));
```

```
-- %%%%%%%% AGGIUNTA DI CANZONE REGISTRATA
Insert into REGISTRATA
   (CANZONE, STUDIO, DATA_REGISTRAZIONE)
 Values
   ('22Z', 'Colucci recording', TO_DATE('01/09/1993', 'DD/MM/YYYY'));

Insert into REGISTRATA
   (CANZONE, STUDIO, DATA_REGISTRAZIONE)
 Values
   ('02A', 'Air studio', TO_DATE('01/01/2020', 'DD/MM/YYYY'));

Insert into REGISTRATA
   (CANZONE, STUDIO, DATA_REGISTRAZIONE)
 Values
   ('01J', 'Radio jonni recording', TO_DATE('13/04/1999', 'DD/MM/YYYY'));
```

```
-- %%%%%%%% AGGIUNTA DI STRUTTURE
Insert into STRUTTURE
   (ID_STRUTTURA, NOME_STRUTTURA, ID_CITTA)
 Values
   (1, 'Stadio San Paolo', 1);

Insert into STRUTTURE
   (ID_STRUTTURA, NOME_STRUTTURA, ID_CITTA)
 Values
   (2, 'Stadio San Siro', 2);

Insert into STRUTTURE
   (ID_STRUTTURA, NOME_STRUTTURA, ID_CITTA)
 Values
   (3, 'Stadio Olimpico', 5);
```

```
-- %%%%%%%% AGGIUNTA DI STUDIO REGISTRAZIONE
Insert into STUDIO
   (NOME_STUDIO, TARIFFA, ID_INDIRIZZO)
 Values
   ('Air studio', 800, 1);

Insert into STUDIO
   (NOME_STUDIO, TARIFFA, ID_INDIRIZZO)
 Values
   ('Colucci recording', 500, 2);

Insert into STUDIO
   (NOME_STUDIO, TARIFFA, ID_INDIRIZZO)
 Values
   ('Shuttle studio', 900, 3);
```

```
-- %%%%%%%% AGGIUNTA DI TIPOLOGIA PROMOZIONE
Insert into TIPOLOGIA_PROMOZIONE
   (ID_TIPO_PROMOZIONE, ID_PROMOZIONE, TIPO_PROMOZIONE)
 Values
   (1, 1, 'Televisiva');

Insert into TIPOLOGIA_PROMOZIONE
   (ID_TIPO_PROMOZIONE, ID_PROMOZIONE, TIPO_PROMOZIONE)
 Values
   (2, 2, 'Riviste');

Insert into TIPOLOGIA_PROMOZIONE
   (ID_TIPO_PROMOZIONE, ID_PROMOZIONE, TIPO_PROMOZIONE)
 Values
   (3, 3, 'Radiofonica');
```

```
-- %%%%%%%% AGGIUNTA DI TIPO DISTRIBUTORE
Insert into TIPO_DISTRIBUTORE
   (NOME_SOCIETA, TIPO_DISTRIBUTORE)
 Values
   ('Amazon music', 'Digitale');

Insert into TIPO_DISTRIBUTORE
   (NOME_SOCIETA, TIPO_DISTRIBUTORE)
 Values
   ('Amazon music', 'Fisico');

Insert into TIPO_DISTRIBUTORE
   (NOME_SOCIETA, TIPO_DISTRIBUTORE)
 Values
   ('Apple music', 'Digitale');
```

```
-- %%%%%%%% AGGIUNTA DI VIDEOCLIP
Insert into VIDEOCLIP
   (ID_VIDEO, NOME_VIDEO, REGISTA, DURATA_VIDEO, COSTO)
 Values
   ('V01', 'Non avere paura', 'Carolina Sansoni', '00:04:10', 500);

Insert into VIDEOCLIP
   (ID_VIDEO, NOME_VIDEO, REGISTA, DURATA_VIDEO, COSTO)
 Values
   ('V02', 'I nostri anni', 'Carolina Sansoni', '00:03:39', 400);

Insert into VIDEOCLIP
   (ID_VIDEO, NOME_VIDEO, REGISTA, DURATA_VIDEO, COSTO)
 Values
   ('V03', 'I nostri anni - (lyric)', 'Carolina Sansoni', '00:03:38', 50);
```

TRIGGER

Trigger 1. TRIG_CONT

Il trigger seguente impedisce durante l'inserimento di un nuovo contratto che la data di inizio sia antecedente a quella odierna.

All'interno troviamo un IF che controlla se la data di inizio del contratto che stiamo inserendo è minore alla data odierna (sysdate), se questo è vero allora imposta la data dell'inizio contratto uguale alla data odierna. Viene utilizzata la funzione "Trunc" per eliminare l'orario dalle date, in modo tale da non avere problemi con il confronto.

TRIGGER 1. TRIG_CONT

```
CREATE OR REPLACE trigger TRIG_CONT
before insert or update on CONTRATTO
for each row

/*
  TRIGGER NUMERO 1 :

  Un contratto non può avere una data di inizio diversa da quella odierna.
```

```
  Il trigger scatterà quando si inserirà una data diversa da quella odierna
  e automaticamente l'aggiornerà ad essa.
*/

begin

   if trunc(:new.inizio_contratto) < trunc(sysdate) then
      :new.inizio_contratto := trunc(sysdate);

      dbms_output.put_line('Il contratto ha una data inferiore a quella odierna.
      Pertanto al campo inizio contratto è stata sostituita la data odierna');

   end if;

end;
```

Trigger 2. TRIG_ALB

Il seguente trigger impedisce di inserire più di 10 canzoni all'interno dello stesso album.

Il trigger è stato implementato costruendo un contatore che ha il compito di contare il numero di canzoni all'interno di un album e laddove il numero di canzoni è uguale a 10 impedirà l'inserimento.

Se il contatore supera le 10 canzoni allora viene rilasciata un'eccezione ("troppe_canzoni"), la quale impedisce l'inserimento e mostra un messaggio di errore.

Trigger 2. TRIG_ALB

```
CREATE OR REPLACE TRIGGER TRIG_ALB
before insert or update on canzone
for each row
declare

cont_1            number(2,0);
troppe_canzoni    exception;

/*
  TRIGGER NUMERO 2 :

  Non è possibile inserire più di 10 canzoni in uno stesso album.
*/

begin
   --Conteggio canzoni nell'album.
   select count(*)into cont_1
   from canzone
   where album =:new.album;

   if (cont_1 >= 10) then

        raise troppe_canzoni;

    end if;

exception
   when troppe_canzoni then
   raise_application_error (-20008, 'Non puoi inserire più di 10 canzoni
                            in un album.');

end;
```

Trigger 3. TRIG_MANAGER

Il seguente trigger nel momento in cui si inserisce in un nuovo contratto lo stipendio del manager, se egli è il più giovane o il più anziano, gli viene modificato il valore dello stipendio aumentato del 10%.

Viene individuato il manager più anziano e il manager più giovane, tramite l'utilizzo delle funzioni "max" e "min" sulla data di nascita. Successivamente viene selezionata la data di nascita del manager che stiamo sottoscrivendo ad un contratto, se dovessero esserci errori durante questa selezione viene mostrato un messaggio di errore.

Infine viene controllata se la data di nascita del manager è riferita a quella del manager più giovane o più vecchio e in caso affermativo lo stipendio del manager che stiamo inserendo verrà aumentato del 10%.

Trigger 3. TRIG_MANAGER

```
CREATE OR REPLACE TRIGGER trig_manager
before insert or update on contratto
for each row
declare

man_anz          manager.data_nascita%type;
man_gio          manager.data_nascita%type;
man_ins          manager.data_nascita%type;

/*
  TRIGGER NUMERO 3 :

  Quando è immesso un nuovo contratto, si aggiorni il valore dello stipendio del
  manager aggiornandolo del 10% se egli è il manager più giovane o quello
  più anziano.
*/

begin

      select max(data_nascita), min(data_nascita) into man_gio, man_anz
      from manager;

      dbms_output.put_line('Data manager piu'' giovane : '||man_gio);
      dbms_output.put_line('Data manager piu'' vecchio : '||man_anz);

      Begin

          select data_nascita into man_ins
          from manager
          where cf_manager = :new.manager;

          exception when others then
            dbms_output.put_line('Errore nella selezione della data
            di nascita.'||sqlerrm);

      End;


      if (man_ins = man_anz or man_ins = man_gio) then
         --Aumento dello stipendio del 10%.

        :new.stipendio_manager := :new.stipendio_manager * 1.1;
      end if;

end;
```

Trigger 4. TRIG_SHOW

Il seguente trigger impedisce ad un artista di partecipare a più di un concerto in una stessa data.

Il trigger si avvale di un contatore che conta quante date del concerto di un artista ci sono con la stessa data che stiamo inserendo. Se il valore del contatore è minore di 0, vuol dire che non esistono date in quel giorno e l'inserimento sarà possibile, in caso contrario invece l'artista è già impegnato in un concerto, e allora partirà un'eccezione, che manderà un messaggio di errore.

Trigger 4. TRIG_SHOW

```
create or replace trigger trig_show
before insert on concerti
for each row
declare

data_conc          number;
date_uguali        exception;
progressivo        number:=0;
/*
  TRIGGER NUMERO 4 :

  Un artista non può partecipare a più di un concerto in una stessa data.
*/
Begin
    Begin
        select nvl(max(id_concerto),0)+1 into :new.id_concerto
        from concerti;
    end;

    begin
        --Conteggio date del concerto.
        select count(data_concerto) into data_conc
        from concerti
        where data_concerto = :new.data_concerto and artista = :new.artista;

        if (data_conc > 0) then
            raise date_uguali;
        end if;

exception
    when date_uguali then
    raise_application_error (-20011, 'Un artista non può svolgere
                             due concerti in uno stesso giorno.');


    end;
end;
```

Trigger 5. TRIG_VEN

Il seguente trigger è aggiornato alle nuove dinamiche di mercato discografico, e permette ad un artista di pubblicare un album solo durante il quinto giorno della settimana (venerdì, giorno di uscita delle nuove canzoni).

Il primo IF controlla se il giorno della data che stiamo inserendo è diverso da venerdì. Ciò è reso possibile convertendo la data di uscita della canzone in un char.

Nel secondo IF invece viene controllata se la data che stiamo inserendo sia minore della data odierna.

In entrambi i casi viene stampato un messaggio di errore.

Trigger 5. TRIG_VEN

```
CREATE OR REPLACE TRIGGER TRIG_VEN
before insert on album
for each row
declare


ex_trig      exception;
ex_trig2     exception;


/*
  TRIGGER NUMERO 5 :

  Un album puo' uscire solo di venerdi' e non prima di oggi.

*/


begin


    --Controlla se la data di uscita del nuovo album è di venerdì.

    if ( to_char(:new.data_uscita, 'D') !=  5) then
       raise ex_trig;

    end if;

    if (:new.data_uscita <  sysdate) then

       raise ex_trig2;

    end if;


exception

    when ex_trig then
    raise_application_error (-20015, 'L''album può uscire solo di venerdì.');

    when ex_trig2 then
    raise_application_error (-20016, 'L''album non può uscire prima di oggi.');

end;
```

Trigger 6. TRIG_PREM

Il seguente trigger impone una propedeuticità obbligatoria ad una canzone nel ricevere come premio un disco di platino se prima non ha già ottenuto un disco d'oro nello stesso paese, stesso discorso con il premio Fimi che ha un valore maggiore rispetto al disco di platino.

Il trigger controlla se il premio che stiamo inserendo è un disco di platino, se questo è vero, allora conta quanti dischi d'oro ci sono nel paese del premio preso in considerazione, se il numero è diverso da 1, vuol dire che la canzone non ha il disco d'oro in quel paese e allora

viene rilasciata l'eccezione e il messaggio di errore. Stessa identica cosa viene fatta con il premio Fimi, con la differenza che viene controllato se la canzone ha il disco di platino nello stesso paese.

Trigger 6. TRIG_PREM

```
CREATE OR REPLACE TRIGGER TRIG_PREM
before insert on riceve
for each row
declare

disco         number;
platino       number;

ex_prem       exception;
ex_plat       exception;

/*
  TRIGGER NUMERO 6 :

  Non si può inserire un disco di platino se prima l'artista non ha ottenuto un
  disco d'oro nello stesso paese.
  Inoltre non si può inserire un premio Fimi se prima l'artista non ha ottenuto
  un disco di platino nello stesso paese.

*/

begin

    if (:new.premio = 'Disco platino') then

        select count(premio) into disco
        from riceve
        where canzone = :new.canzone and premio = 'Disco oro'
              and paese = :new.paese;

        if (disco != 1) then

           raise ex_prem;

        end if;
     end if;

     if (:new.premio = 'Fimi') then

         select count(premio) into platino
         from riceve
         where canzone = :new.canzone and premio = 'Disco platino'
               and paese = :new.paese;


         If (platino != 1) then

               raise ex_plat;

         end if;
      end if;

exception

     when ex_prem then
     raise_application_error (-20004, 'Non puoi inserire un disco di platino
     se prima l''artista non ha ottenuto un disco d''oro nello stesso paese.');

     when ex_plat then
     raise_application_error (-20005, 'Non puoi inserire un premio Fimi se prima
     l''artista non ha ottenuto un disco di platino nello stesso paese.');
```

```
end;
```

Trigger 7. TRIG_REC

Il seguente trigger impedisce che la data di registrazione di una canzone sia antecedente alla data di scrittura e sia successiva alla data di uscita dell'album.

La prima select seleziona la data di scrittura dove la chiave esterna "canzone" è uguale alla chiave della canzone che stiamo inserendo. La seconda select seleziona invece la data di uscita dell'album, di cui fa parte la canzone.
Dopo troviamo un confronto in cui si verifica se la data di registrazione della canzone è antecedente alla data di scrittura e in caso affermativo allora verrà inviato un messaggio di errore. Stessa cosa succede se la data di registrazione è dopo la data di uscita dell'album.

Trigger 7. TRIG_REC

```
CREATE OR REPLACE TRIGGER TRIG_REC
before insert or update on registrata
for each row
declare

data_sc        scritta.data_scrittura%type;
data_cd        album.data_uscita%type;


data_errata    exception;
wrong_rec      exception;

/*
  TRIGGER NUMERO 7 :

  Una canzone non può essere registrata se prima non è stata scritta.
  Inoltre la canzone dell'artista non può essere registrata dopo l'uscita
  dell'album.

*/

begin

    select data_scrittura into data_sc
    from scritta
    where canzone = :new.canzone;


    select data_uscita into data_cd
    from album join canzone on id_album = album
    where id_canzone = :new.canzone;


    if (:new.data_registrazione < data_sc) then

        raise data_errata;

    end if;


    if (:new.data_registrazione > data_cd) then

        raise wrong_rec;

    end if;
```

```
exception

    when data_errata then
    raise_application_error (-20009, 'L''artista non può registrare
    una canzone prima di essere scritta.');

    when wrong_rec then
    raise_application_error (-20010, 'La canzone dell''artista
    non può essere registrata dopo l''uscita dell''album.');

end;
```

Trigger 8. TRIG_DIS

Il seguente trigger impedisce ad un distributore fisico di distribuire un album in formato digitale, allo stesso tempo impedisce ad un distributore digitale di distribuire un album in formato fisico.

Il trigger conta quanti formati ha l'album, e quanti formati ha il distributore.
Se il distributore ha più di un tipo, vuol dire che è sia fisico sia digitale, allora l'album può essere distribuito in entrambi i formati. Altrimenti se il contatore del tipo del distributore è 1, significa che l'album è solo fisico oppure solo digitale. Successivamente si entra nel primo IF dove troviamo una select che controlla di che tipo è il distributore, e, se i formati dell'album sono più di 1, controlla se ci sono formati digitali e quanti formati fisici ci sono. Dopodiché troviamo due if che controllano la compatibilità dei formati con quello del distributore e in caso di non compatibilità viene mandato un messaggio di errore.

Se il formato è solo 1 allora si seleziona il tipo del formato e si confronta con il tipo del distributore, se non combaciano, viene inviato l'errore.

Trigger 8. TRIG_DIS

```
CREATE OR REPLACE TRIGGER TRIG_DIS
before insert or update on distribuito
for each row
declare

tipo_cd             formato.formato%type;
tipo_dis            tipo_distributore.tipo_distributore%type;
cont_tp             number;
cont_fis            number;
cont_dig            number;
cont2               number;

incompatibili_1     exception;
incompatibili_2     exception;

/*
  TRIGGER NUMERO 8 :

  Un album digitale non può essere distribuito da un tipo distributore fisico.
  Inoltre un album fisico non può essere distribuito da un tipo distributore
  digitale.
*/

begin
```

```
    select count(formato) into cont2
    from formato
    where album = :new.album;

    select count(tipo_distributore) into cont_tp
    from tipo_distributore
    where nome_societa = :new.distributore;


    if (cont_tp < 2) then

       select tipo_distributore into tipo_dis
       from tipo_distributore
       where nome_societa = :new.distributore;

        if (cont2 > 1) then

           select count(formato) into cont_dig
           from formato
           where album = :new.album and formato = 'Digitale';

           select count(formato) into cont_fis
           from formato
           where album = :new.album and formato != 'Digitale';

           if (cont_dig > 0 and cont_fis < 1 and tipo_dis = 'Fisico') then

              raise incompatibili_1;

           end if;

           if (cont_fis > 0 and cont_dig < 1 and tipo_dis = 'Digitale') then

              raise incompatibili_2;

           end if;

        end if;

        if (cont2 = 1) then

           select formato into tipo_cd
           from formato
           where album = :new.album;

           if (tipo_cd = 'Digitale' and tipo_dis = 'Fisico') then

              raise incompatibili_1;

           end if;


           if (tipo_cd != 'Digitale' and tipo_dis = 'Digitale') then

              raise incompatibili_2;

           end if;

        end if;

    end if;


exception

    when incompatibili_1 then
```

```
    raise_application_error (-20011, 'Un album digitale non può
    essere distribuito da un tipo distributore fisico.');

    when incompatibili_2 then
    raise_application_error (-20012, 'Un album fisico non può
    essere distribuito da un tipo distributore digitale.');

end;
```

Trigger 9. TRIG_VID

Il seguente trigger limita il costo di un "Lyric video" ad un prezzo non superiore di 100 euro.

Il trigger è costituito da un unico IF che controlla se all'interno del nome del video che stiamo inserendo c'è la presenza della parola "Lyric" e in quel caso se il costo del video che stiamo inserendo è maggiore di 100, viene mostrato un messaggio di errore.

Trigger 9. TRIG_VID

```
CREATE OR REPLACE TRIGGER TRIG_VID
before insert or update on videoclip
for each row
declare

video      number;
lyric      exception;

/*
  TRIGGER NUMERO 9 :

  Il costo di un video lyric non deve superare 100 euro.
*/

begin

   if (:new.nome_video like '%(lyric)' and :new.costo > 100) then

      raise lyric;

   end if;


Exception
    when lyric then
    raise_application_error (-20012, 'Il video di un lyric non può
                             costare più di 100 euro.');

end;
```

Trigger CITTA'

Questo trigger è stato implementato anche per le seguenti tabelle : contratto, indirizzi_studio, promozione, concerti, strutture e tipologia_promozione.

Ha il compito di incrementare l'ID di 1 ogni volta che ci sarà un inserimento.

TRIGGER CITTA'

```
CREATE OR REPLACE TRIGGER trig_citta
before insert on citta
for each row
declare

progressivo       number:=0;
/*
   TRIGGER NUMERO 13 :

   CALCOLO PROGRESSIVO CITTA.
   Questo trigger scatta quando si inserisce un nuovo record all'interno della
   tabella città e automaticamente incrementerà l'ID di 1.
*/


begin
    begin
        select nvl(max(id_citta),0)+1
          into :new.id_citta
          from citta;
    end;

end;
```

PROCEDURE E FUNZIONI

PROCEDURA 1 - Nuovo contratto.

```
CREATE OR REPLACE procedure CASA_DISCOGRAFICA.nuovo_contratto(
                                p_id_artista          in varchar
                              , p_data_inizio         in date
                              , p_stipendio_artista   in number
                              , p_durata              in date
                              , p_stipendio_manager   in number
                              )
is
v_manager     contratto.manager%type;
ex            exception;

/*
  PROCEDURA NUMERO 1 :

  Quando l'admin inserisce un nuovo contratto (1° tabella coinvolta) gli
  associa ad un'artista (2° tabella coinvolta il manager (3° tabella coinvolta)
  più giovane che attualmente ha meno contratti.
*/

begin

  --Calcolo manager piu' giovane
  Begin

     select cf_manager into v_manager
     from manager
     where data_nascita=(select  max(data_nascita) data
                         from (select manager,(select data_nascita
                                               from manager m1
                                               where m1.CF_MANAGER=c.manager)
                                               data_nascita
                                      ,count(manager)
                               from contratto c, manager m1
                               where m1.cf_manager=c.manager
                               group by c.manager
                               having count(manager) = (select min(cont)
                                                        from(selectmanager,
                                                        count(manager)cont
                                                             from contratto
                                                        group by manager)
                                                        )
                                   )
                            );

      exception when others then dbms_output.put_line('Errore nella selezione
                                 del manager piu'' giovane');
                raise ex;
  end;

  --Inserimento del nuovo contratto.
  Begin

    Insert into contratto
    (inizio_contratto,stipendio_artista,durata_contratto,artista,manager,
    stipendio_manager)
    Values
   (p_data_inizio,p_stipendio_artista,p_durata,p_id_artista,v_manager,
    p_stipendio_manager);

     exception when others then dbms_output.put_line('Errore nell''
                                inserimento del contratto.'||sqlerrm);
                           raise ex;
  end;
```

```
  commit;
    dbms_output.put_line('Ok. Procedura terminata con successo.');

exception when ex then rollback;
          dbms_output.put_line('Procedura terminata con errori.');

end;
```

EXEC PROCEDURA 1.

```
Begin

/*
  EXEC PROCEDURA NUMERO 1 :

  Quando l'admin inserisce un nuovo contratto gli associa ad un'artista il
  manager più giovane che attualmente ha meno contratti.

  Parametri da inserire per testare la procedura :

  - p_id_artista         = 020
  - p_data_inizio        = 28/07/2020 ***Se si proverà ad inserire un data
  antecedente a quella odierna, il trigger automaticamente la aggiornerà ad essa.
  - p_stipendio_artista  = 2300
  - p_durata             = 28/07/2025
  - p_stipendio_manager  = 1775
*/


  casa_discografica.nuovo_contratto(
                                    :p_id_artista,
                                    :p_data_inizio,
                                    :p_stipendio_artista,
                                    :p_durata,
                                    :p_stipendio_manager
                                    );
end;
```

PROCEDURA 2 - TOT_SPESE.

```
CREATE OR REPLACE procedure CASA_DISCOGRAFICA.tot_spese (p_anno in varchar)

is

stip_art     number;
stip_man     number;
costo_video  number;
tar_studio   number;
totale       number;
ex_2         exception;

/*
  PROCEDURA NUMERO 2 :

  Calcolo totale delle spese d'uscita dell'etichetta :
  stipendio manager e stipendio artista  (1° tabella coinvolta = CONTRATTO)
  tariffa studio registrazione           (2° tabella coinvolta = STUDIO)
  costo videoclip                        (3° tabella coinvolta = VIDEOCLIP).

  Operazione effettuata dall'admin (casa_discografica) e dall'utente (contabile).
  Seguono le credenziali di accesso.

  1)
    ID       : casa_discografica
    Password : musica
```

```
  2)
    ID       : contabile
    Password : password1
*/

begin

   --Selezione dello stipendio dell'artista e del manager.
    begin

        select nvl(sum(stipendio_artista),0), nvl(sum(stipendio_manager),0)
        into stip_art, stip_man
        from contratto
        where to_char(inizio_contratto, 'YYYY') <= p_anno and
              to_char(durata_contratto, 'YYYY') >= p_anno;

        exception when others then stip_art := 0; stip_man := 0;

     end;

      --Selezione del costo del video.
     begin

       select nvl(sum(costo),0) into costo_video
       from videoclip join ha
       on id_video = videoclip
       where to_char(uscita_video, 'YYYY') = p_anno;

       exception when others then costo_video := 0;

      end;

      --Selezione della tariffa dello studio di registrazione.
      begin

       select nvl(sum(tariffa),0) into tar_studio
       from studio join registrata
       on nome_studio = studio
       where to_char (data_registrazione, 'YYYY') = p_anno;

       exception when others then tar_studio := 0;

      end;


      totale := (stip_art*12) + (stip_man*12) + costo_video + tar_studio;

      dbms_output.put_line('Ok. Procedura terminata con successo.'
      || chr(13) || 'Il totale e'' ' || totale);

      /*Inserimento all'interno della tabella "tot_spese_log" che indica
       la data in cui è stata effettuata, l'utente che ha effettuato la
      procedura, il totale e l'anno inserito.*/
      begin

       insert into tot_spese_log (Data, Utente, Totale, Anno)
       values (sysdate, user, totale, p_anno);

       exception when others then dbms_output.put_line('Errore nell'
                                  'inserimento del totale.'||sqlerrm);
                 raise ex_2;

      end;

     commit;

exception when ex_2 then rollback;
```

```
end;
```

EXEC PROCEDURA 2

```
Begin

/*
  EXEC PROCEDURA NUMERO 2 :

  La procedura prende in input un anno e scrive nella tabella "tot_spese_log"
  le informazioni riguardo le spese d'uscita relative all'anno inserito.

  Parametri da inserire per testare la procedura :

  - p_anno = 2016 (o un anno qualsiasi)
*/

tot_spese (:p_anno);

End;
```

PROCEDURA 3 - LIVE.

```
CREATE OR REPLACE procedure CASA_DISCOGRAFICA.live(
                        p_nome            in varchar,
                        p_struttura       in varchar,
                        p_citta           in varchar,
                        p_data            in date,
                        p_artista in varchar)
is

v_iban         agenzia_booking.iban%type;
v_data         date;
v_durata       date;
v_id_concerto number;
v_id_promo     number;
v_struttura    number;
eccezione     exception;

/*
  PROCEDURA NUMERO 3 :

  Quando si inserisce un concerto (1° tabella coinvolta), organizzato
  dall'agenzia booking (2° tabella coinvolta)
  che gestisce più concerti, automaticamente gli viene inserita una promozione
 (3°tabella coinvolta) e la tipologia promozione
 (4°tabella coinvolta)"Televisiva".
*/

begin

     --Calcolo dell'agenzia booking che gestisce più concerti.
     begin

        select iban into v_iban
        from agenzia_booking join concerti
        on iban = codice_iban
        group by iban
        having count(iban) = (select max(num_conc)
                              from(select count(codice_iban) num_conc
                                   from concerti
                                   group by codice_iban));

        exception when others then dbms_output.put_line('Errore nella
                                   query.'||sqlerrm);
                  raise eccezione;
```

```
 end;

 Begin

    select id_struttura into v_struttura
    from strutture s join citta c
    on s.id_citta = c.id_citta
    where nome_struttura = p_struttura and nome_citta = p_citta;

    exception when others then dbms_output.put_line('Errore nella
                               selezione dell''id struttura.'||sqlerrm);
              raise eccezione;

 end;

--Inserimento del concerto.
begin

    insert into concerti (nome_concerto, id_struttura, data_concerto,
                          artista, codice_iban)
    values (p_nome, v_struttura, p_data, p_artista, v_iban);

    exception when others then dbms_output.put_line('Errore nell''
                               inserimento del concerto.'||sqlerrm);
              raise eccezione;

 end;

--Inserimento della promozione.
begin

   v_data := p_data - 30;
   v_durata := p_data - 1;
   Begin
       select max(id_concerto) into v_id_concerto from concerti;
       exception when others then dbms_output.put_line('Errore recupero
                                  id concerto.'||sqlerrm);
                 raise eccezione;
   end;

   insert into promozione (id_concerto,inizio_promozione, durata)
   values (v_id_concerto,v_data, v_durata);

   exception when others then dbms_output.put_line('Errore nell''
                              inserimento della promozione.'||sqlerrm);
             raise eccezione;

 end;

 --Inserimento della tipologia promozione "Televisiva".
 Begin

    select max(id_promozione) into v_id_promo from promozione;
    exception when others then dbms_output.put_line('Errore recupero
                               id promozione.'||sqlerrm);
              raise eccezione;
 end;

 begin

    insert into tipologia_promozione (id_promozione, tipo_promozione)
    values (v_id_promo, 'Televisiva');

    exception when others then dbms_output.put_line('Errore nell''
                               inserimento della tipologia
                               promozione.'||sqlerrm);
              raise eccezione;
```

```
     end;

     commit;
     dbms_output.put_line('Procedura terminata con successo');
exception when eccezione then rollback;
          dbms_output.put_line('Procedura Terminata con errori');

End;
```

EXEC PROCEDURA 3

```
Begin

/*
   EXEC PROCEDURA NUMERO 3 :

   Quando si inserisce un concerto, organizzato dall'agenzia booking
   che gestisce più concerti, automaticamente gli viene assegnato una
   promozione e la tipologia promozione "Televisiva".

   Parametri da inserire per testare la procedura :

   - p_nome      = Eros tour
   - p_struttura = Stadio San Paolo
   - p_data      = 22/06/2020
   - p_citta     = Napoli
   - p_artista   = 017

*/

live(
     :p_nome,
     :p_struttura,
     :p_citta,
     :p_data,
     :p_artista
    );

End;
```

PROCEDURA 4 - PROMO_CONCERTO.

```
CREATE OR REPLACE procedure CASA_DISCOGRAFICA.promo_concerto(
                       p_nome        in concerti.nome_concerto%type,
                       p_data        in concerti.data_concerto%type,
                       p_datapromo   in promozione.inizio_promozione%type,
                       p_datapromo_new  in promozione.inizio_promozione%type,
                       p_tipo_new    in tipologia_promozione.tipo_promozione%type,
                       p_struttura_new  in strutture.nome_struttura%type,
                       p_durata          in promozione.durata%type,
                       p_id_promozione   in promozione.id_promozione%type
                       )

is

ex            exception;
v_id_concerto number;
v_id_promo    number;
v_struttura   number;

/*
  PROCEDURA NUMERO 4 :
```

```
  Dato un concerto, si vuole cancellare la vecchia promozione (1° tabella
  coinvolta) e la sua tipologia (2° tabella coinvolta) e inserirne delle
  nuove aggiornando anche la struttura del concerto (3° tabella coinvolta).

*/

Begin


    Begin

      select id_struttura into v_struttura
      from strutture
      where nome_struttura = p_struttura_new;

      exception when others then dbms_output.put_line('Errore nella
                                 selezione dell''id struttura.'||sqlerrm);
                raise ex;

    end;

    --Aggiornamento della struttura del concerto.
    Begin

      update concerti
      set id_struttura = v_struttura
      where nome_concerto = p_nome and data_concerto = p_data;

      exception when others then dbms_output.put_line('Errore nell''
                                 aggiornamento della struttura.'||sqlerrm);
                raise ex;

    end;

    --Cancellazione della vecchia tipologia promozione.
    Begin

      delete tipologia_promozione
      where id_promozione=p_id_promozione;

      exception when others then dbms_output.put_line('Errore nella
                                 cancellazione della tipologia
                                 promozione.'||sqlerrm);
                raise ex;

    end;

    --Cancellazione della vecchia promozione.
    Begin

       delete promozione
       where id_promozione = p_id_promozione;

       exception when others then dbms_output.put_line('Errore nella
                                  cancellazione della promozione.'||sqlerrm);
                 raise ex;

    end;

    -- recupero id del concerto selezionandolo per nome e data
    Begin
        select id_concerto into v_id_concerto
        from concerti
        where nome_concerto = p_nome and data_concerto = p_data;
        exception when others then dbms_output.put_line('Errore recupero
                                   id concerto.'||sqlerrm);
                  raise ex;
```

```
    End;

    --Inserimento della nuova promozione.
    Begin

      insert into promozione (id_concerto, inizio_promozione, durata)
      values (v_id_concerto,p_datapromo_new, p_durata);

      exception when others then dbms_output.put_line('Errore nell''
                                 inserimento della nuova promozione.'||sqlerrm);
                raise ex;

    end;

    Begin
           select max(id_promozione) into v_id_promo from promozione;
           exception when others then dbms_output.put_line('Errore recupero
                                      id  promozione.'||sqlerrm);
                     raise ex;
    end;

    --Inserimento della nuova tipologia promozione.
    Begin

      insert into tipologia_promozione (id_promozione, tipo_promozione )
      values (v_id_promo, p_tipo_new);

      exception when others then dbms_output.put_line('Errore nell''
                                 inserimento della nuova tipologia
                                 promozione.'||sqlerrm);
                raise ex;
    end;


    commit;
    dbms_output.put_line('Elaborazione terminata.');

exception when ex then rollback;
  dbms_output.put_line('Elaborazione terminata con errori.');

end;
```

EXEC PROCEDURA 4

```
Begin

/*
   EXEC PROCEDURA NUMERO 4 :

   Dato un concerto, si vuole cancellare la vecchia promozione
   e la sua tipologia e inserirne delle nuove aggiornando anche la struttura
   del concerto.

   Parametri da inserire per testare la procedura :

   - p_nome              = Ligabue tour
   - p_data              = 29/05/2004
   - p_datapromo         = 02/05/2004
   - p_datapromo_new     = 03/05/2004
   - p_tipo_new          = Radiofonica
   - p_struttura_new     = Alcatraz
   - p_durata            = 28/05/2004
   - p_id_promozione     = 34

*/
promo_concerto (:p_nome,
                :p_data,
```

```
                :p_datapromo,
                :p_datapromo_new,
                :p_tipo_new,
                :p_struttura_new,
                :p_durata,
                :p_id_promozione
                );

End;
```

PROCEDURA 5 - DISTRIBUTORE_ALBUM

```
CREATE OR REPLACE procedure CASA_DISCOGRAFICA.distributore_album (
                                                p_album    in varchar,
                                                p_nome     in varchar,
                                                p_sede     in varchar,
                                                p_email    in varchar,
                                                p_telefono in varchar,
                                                p_iva      in varchar,
                                                p_tipo     in varchar
                                                )
is
errore       exception;
v_id_dis     number;
contatore    number;

/*
  PROCEDURA NUMERO 5 :

  Aggiungere un nuovo distributore (1° tabella coinvolta) e la sua tipologia
  (2° tabella coinvolta) ad un album (3° tabella coinvolta).
*/


Begin

    Begin

      select count(id_citta) into contatore
      from citta
      where nome_citta = p_sede;

      exception when others then dbms_output.put_line('Errore nel
                                 conteggio id citta'||sqlerrm);
                raise errore;

     end;

     Begin

       if (contatore = 0) then

         insert into citta (nome_citta)
         values (p_sede);

       end if;

       exception when others then dbms_output.put_line('If andato a
                                  buon fine.'||sqlerrm);
                 raise errore;

     end;



     Begin

       select id_citta into v_id_dis
```

```
        from citta
        where nome_citta = p_sede;

        exception when others then dbms_output.put_line('Errore nella
                                   selezione del distributore.'||sqlerrm);
                  raise errore;

     End;

     --Inserimento nuovo distributore.
     Begin

       insert into distributore
      (nome_distributore,id_citta_distributore,email_distributore,
       telefono_distributore, partita_iva)
       values (p_nome, v_id_dis, p_email, p_telefono, p_iva);

       exception when others then dbms_output.put_line('Errore nell''
                                   inserimento del distributore.'||sqlerrm);
                  raise errore;

     end;

     --Inserimento nuova tipologia distributore.

     Begin

       insert into tipo_distributore (nome_societa, tipo_distributore)
       values (p_nome, p_tipo);

       exception when others then dbms_output.put_line('Errore nell''
                                   inserimento del tipo distributore.'||sqlerrm);
                  raise errore;

     end;

     --Associazione dell'album al distributore.
     Begin

       insert into distribuito (album, distributore)
       values (p_album, p_nome);

       exception when others then dbms_output.put_line('Errore nell''
                                   inserimento in distribuito.'||sqlerrm);
                  raise errore;

     end;

     commit;
     dbms_output.put_line('Elaborazione terminata con successo.');
exception when errore then rollback;
  dbms_output.put_line('Elaborazione terminata con errori.');

End;
```

EXEC PROCEDURA 5

```
Begin

/*
   EXEC PROCEDURA NUMERO 5 :

   Aggiungere un nuovo distributore e la sua tipologia ad un album.

   Parametri da inserire per testare la procedura :

   - p_album     = A21
   - p_nome      = Itunes
```

```
   - p_sede      = Taormina
   - p_email     = itunes.05@gmail.com
   - p_telefono  = 3380005016
   - p_iva       = partiva017
   - p_tipo      = Fisico

*/

distributore_album (
                     :p_album,
                     :p_nome,
                     :p_sede,
                     :p_email,
                     :p_telefono,
                     :p_iva,
                     :p_tipo
                    );

End;
```

PROCEDURA 6 - AUMENTO_STIPENDIO

```
CREATE OR REPLACE procedure CASA_DISCOGRAFICA.aumento_stipendio (
                                          p_id  in varchar)

is

cont_premi      number;
cont_alb        number;
cont_contratto  number;
durata          date;
contratto_scaduto exception;

/*
  PROCEDURA NUMERO 6 :

  Se un artista ha ottenuto più di 3 premi (1° tabella coinvolta), ha prodotto
  più di 1 album (2° tabella coinvolta), e ha firmato più di 1 un contratto
  (3° tabella coinvolta), allora lo stipendio dell'artista verrà aumentato di
  1000 euro.
*/

Begin
      --Conteggio dei premi.
      Begin

        select count(*) into cont_premi
        from ((riceve join canzone
              on canzone = id_canzone)join artista on artista = id_artista)
        where id_artista = p_id;

        exception when others then dbms_output.put_line('Errore nel
                                   conteggio dei premi.'||sqlerrm);
                  raise contratto_scaduto;

       End;

      --Conteggio degli album.
      Begin

        select count(*) into cont_alb
        from album join artista
        on artista = id_artista
        where id_artista = p_id;

        exception when others then dbms_output.put_line('Errore nel
                                   conteggio degli album.'||sqlerrm);
                  raise contratto_scaduto;
```

```
      End;

      --Conteggio dei contratti.
      Begin

        select count(*) into cont_contratto
        from album join artista
        on artista = id_artista
        where id_artista = p_id;

        exception when others then dbms_output.put_line('Errore nel conteggio
                                  dei contratti.'||sqlerrm);
                  raise contratto_scaduto;

      End;

      --Selezione del contratto ancora in atto.
      Begin
        select max(durata_contratto) into durata
        from contratto
        where artista = p_id;

        exception when others then dbms_output.put_line('Errore selezione
                                  dell''ultimo contratto.'||sqlerrm);
                  raise contratto_scaduto;

      End;


      if (durata > sysdate and cont_premi > 3 and cont_alb > 1) then

      Begin
          update contratto
          set stipendio_artista = stipendio_artista + 1000
          where artista = p_id and durata_contratto = durata;

         exception when others then dbms_output.put_line('Errore update'
                                    ||sqlerrm);
                   raise contratto_scaduto;

      end;

      else
       dbms_output.put_line('Errore ! Il contratto è scaduto.'||sqlerrm);
                  raise contratto_scaduto;

      end if;

      commit;
      dbms_output.put_line('Elaborazione terminata con successo.');

exception when contratto_scaduto then rollback;
  dbms_output.put_line('Elaborazione terminata con errori.');

end;
```

EXEC PROCEDURA 6

```
Begin

/*
  EXEC PROCEDURA NUMERO 6 :

  La procedura prende in input un id di un'artista e aumenta lo stipendio di 1000
  euro se l'artista ha ottenuto più di 3 premi, ha realizzato più di un album e
  se ha firmato più di 1 contratto.

  Parametri da inserire per testare la procedura :

  - p_id = 017
*/

aumento_stipendio (:p_id);

End;
```

Data Control Language

L'utente *impiegato* può visualizzare i tipi distributore, i manager, i distributori, le tipologie promozione, gli artisti, le promozioni, i concerti, le agenzie booking e i contratti. Inoltre può anche eseguire le procedure a lui assegnate, cioè le procedure live, distributore_album, nuovo_contratto e promo_concerto. Infine con il *grant resource* ha la possibilità di creare tabelle, trigger procedure ecc..

```
GRANT CONNECT,   CREATE SESSION        TO impiegato;
GRANT RESOURCE                         TO impiegato;
GRANT SELECT ON tipo_distributore      TO impiegato;
GRANT SELECT ON manager                TO impiegato;
GRANT SELECT ON distribuito            TO impiegato;
GRANT SELECT ON tipologia_promozione   TO impiegato;
GRANT SELECT ON artista                TO impiegato;
GRANT SELECT ON promozione             TO impiegato;
GRANT SELECT ON distributore           TO impiegato;
GRANT SELECT ON concerti               TO impiegato;
GRANT SELECT ON agenzia_booking        TO impiegato;
GRANT SELECT ON contratto              TO impiegato;

GRANT EXECUTE ON live                  TO impiegato;

GRANT EXECUTE ON distributore_album    TO impiegato;

GRANT EXECUTE ON nuovo_contratto       TO impiegato;

GRANT EXECUTE ON promo_concerto        TO impiegato;
```

L'utente *contabile* può visualizzare gli studi di registrazione, i contratti, i videoclip, la tabella tot_spese_log che viene usata dalla procedura "tot_spese".
Inoltre può anche eseguire le sole due procedure assegnatogli, cioè tot_spese e aumento_stipendio. Infine con il *grant resource* ha la possibilità di creare tabelle, trigger procedure ecc..

```
GRANT CONNECT,   CREATE SESSION    TO contabile;

GRANT RESOURCE                     TO contabile;

GRANT SELECT ON studio             TO contabile;

GRANT SELECT ON contratto          TO contabile;

GRANT SELECT ON videoclip          TO contabile;

GRANT SELECT ON tot_spese_log      TO contabile;

GRANT EXECUTE ON tot_spese         TO contabile;

GRANT EXECUTE ON aumento_stipendio TO contabile;
```