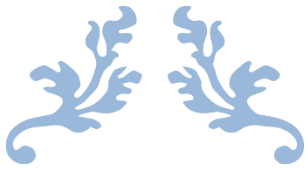


GAMESTOP

Basi di Dati

Armando Molino Mat: 0124001677
Ciro Cozzolino Mat: 0124001804
Carmine Alberto Montanino Mat: 0124001831

Sommario

Progettazione

SINTESI DEI REQUISITI
EE/R
RELAZIONALE
UTENTI
OPERAZIONI
VOLUMI
VENCOLI

Progettazione

Si vuole creare un database per la gestione di una catena, a livello regionale, di negozi a tema videoludico e ludico.
I seguenti negozi oltre a vendere vari prodotti essi permettono ai clienti registrati l'affitto di postazioni e/o tavoli per giocare.
Vengono anche organizzati degli eventi (tornei e distribuzioni) ai quali possono partecipare solo i clienti registrati.

Dall'elaborazione di quanto detto sono scaturiti i seguenti requisiti.

Sintesi dei requisiti

Il database deve gestire:

- Le vendite che vengono effettuate dal reparto vendita.
- Le ore di affitto delle postazioni del reparto videoludico[1].
- Le ore di affitto dei tavoli del reparto ludico[2].
- Gli eventi organizzati in ogni filiale.
- Le offerte.
- I pagamenti degli stipendi di ogni lavoratore.

Gli impiegati devono potersi connettere al database, registrare le vendite e gli affitti, iscrivere i clienti registrati ai tornei, registrare i clienti ed eventualmente aggiornare le tessere dei clienti registrati.

Gli amministratori[3] devono potersi connettere al database, creare eventi, creare offerte, vedere se sono stati pagati tutti gli stipendi e leggere tutte le entrate e le uscite relative ad una filiale

I direttori[4] devono potersi connettere al database e leggere tutte le entrate e le uscite relative ad una filiale.

Ogni cliente per effettuare la registrazione deve recarsi in uno dei negozi e richiede la registrazione; è l'impiegato di turno ad effettuare la registrazione ed a rilasciare la tessera.
Il rinnovo della tessera deve essere effettuato dagli impiegati in negozio.

Per partecipare ad un torneo il cliente registrato[5] deve recarsi in negozio e richiedere la registrazione al torneo desiderato ed infine pagare la relativa quota di partecipazione.

I clienti registrati oltre ad aver accesso ai tornei possono anche partecipare alle distribuzioni[6] ed accedere a degli sconti sui vari articoli in vendita.

[1] Relativo ai videogiochi.
[2] Attinente al gioco
[3] Coloro che amministrano una filiale
[4] Colui che dirige un reparto di una filiale
[5] Clienti che hanno effettuato la registrazione
[6] Evento nel quale vengono distribuiti gratuitamente dei prodotti promozionali

Diagramma EE/R

Il diagramma EE/R è visibile nella seguente immagine. Nelle associazioni non vengono indicate le cardinalità massime e minime delle associazioni.

Diagramma relazionale

Il diagramma relazionale è visibile nella seguente immagine.

Utenti e le loro categorie

Il DB ha solo quattro utenti:

1. Un amministratore del database.
2. Un proprietario_catena (account utilizzato dai proprietari del GameSTOP).
3. Un impiegato (account utilizzato dagli impiegati del GameSTOP).
4. Un direttore (account utilizzato dai direttori del GameSTOP).
5. Un amministratore (account utilizzato dai amministratore del GameSTOP).

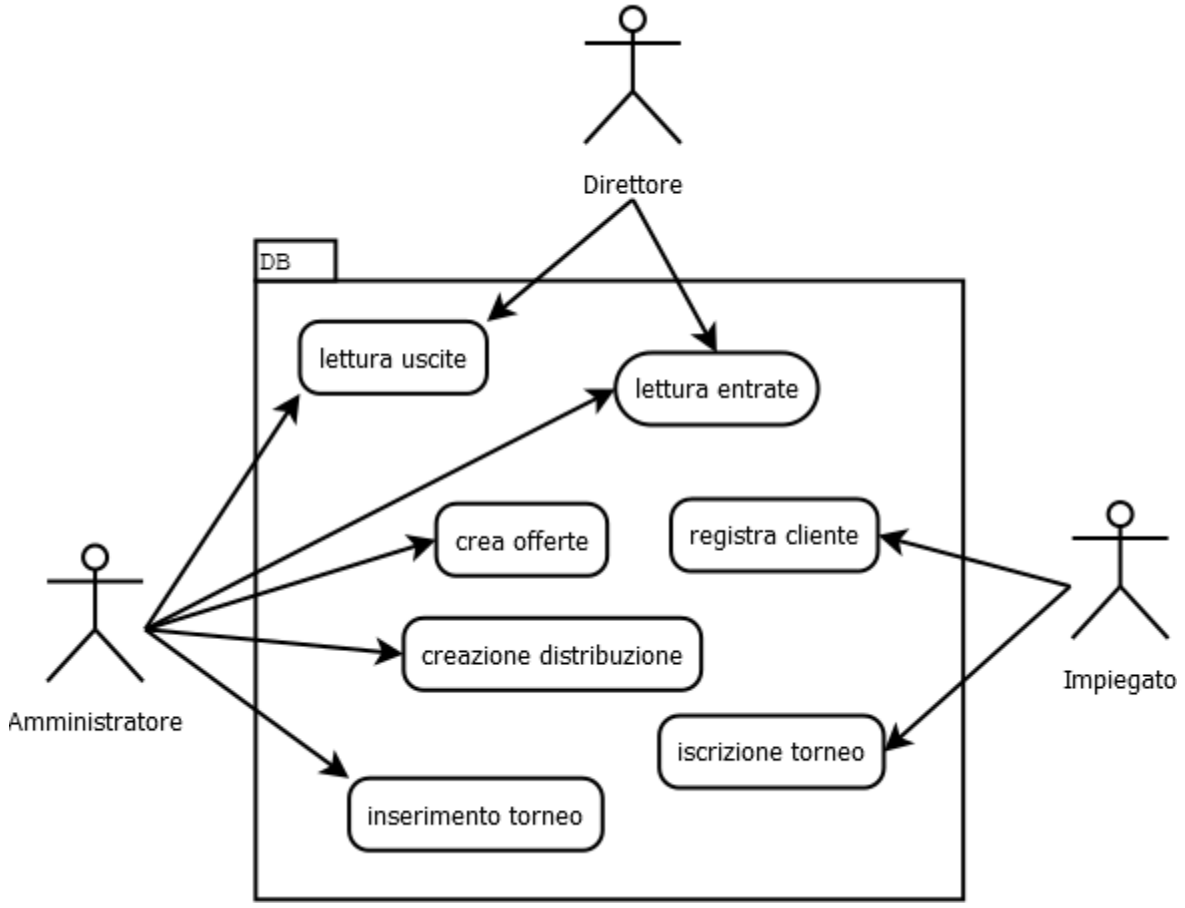
Non vengono definiti ruoli per gestire i permessi, è sufficiente un uso moderato delle viste.

Di seguito viene mostrata la tavola degli utenti. Nella tavola viene mostrato le operazioni che possono essere eseguite dai vari utenti.

Utente	**Tipo**	**Permessi**
db_owner	Amministratore	ALL
proprietari_catena	Comune	INSERT ON lavoratore INSERT ON impiegato INSERT ON direttore INSERT ON amministratore INSERT ON stipendi INSERT ON fornitura INSERT ON prodotti_contenuti
direttore	Comune	SELECT ON impiegato EXECUTE ON lettura_entrate EXECUTE ON lettura_uscite
amministratore	Comune	SELECT ON impiegato SELECT ON direttore INSERT ON offerta INSERT ON approvviggionamento INSERT ON consiste EXECUTE ON inserisci_torneo EXECUTE ON crea_distribuzione EXECUTE ON lettura_entrate EXECUTE ON lettura_uscite
impiegato	Comune	INSERT ON vende INSERT ON affitta_tavolo INSERT ON affitta_postazione UPDATE ON tessera EXECUTE ON registra_cliente EXECUTE ON iscrizione_torneo

Operazioni degli utenti

Le operazioni degli utenti sono quelle, più complesse delle operazioni di base di SQL(INSERT, UPDATE, DELETE), che vanno implementate tramite le procedure e funzioni.



Nel nostro progetto le procedure sono: crea_distribuzione, inserisci_torneo, iscrizione_torneo, lettura_entrate, lettura_uscite, registra_cliente.
Nel diagramma dei casi d'uso nella seguente figura sono visibili sia le operazioni che gli utenti.

Di seguito è presente uno schema che descrive sinteticamente le operazioni degli utenti.

OPERAZIONE	**crea_distribuzione**
SCOPO:	Creare un evento di distribuzione gadget
ARGOMENTI:	Codice fiscale dell'amministratore che lo crea, la data di inizio, la data di fine, il codice del gadget da distribuire e la quantità di gadget disponibili alla distribuzione
RISULTATO	Aggiunta di una distribuzione /errore
ERRORE:	Periodo della distribuzione non consentito Codice fiscale amministratore errato
USA:	Evento, organizza, distribuzione, gadget_dist, amministratore, filiale
MODIFICA:	Evento, organizza, distribuzione, gadget_dist
PRIMA:	Non c'è un'altra distribuzione in quel periodo
DOPO:	C'è una sola distribuzione in quel periodo

OPERAZIONE	**Inserisci_torneo**
SCOPO:	Creare un torneo
ARGOMENTI:	Codice fiscale dell'amministratore che lo crea, la data di inizio, la data di fine, la quota di partecipazione, il premio, il nome del gioco , la piattaforma e il massimo numero di partecipanti
RISULTATO	Aggiunta di un torneo /errore
ERRORE:	Periodo del torneo non consentito Codice fiscale amministratore errato
USA:	Evento, organizza, torneo, amministratore, filiale
MODIFICA:	Evento, organizza, torneo
PRIMA:	Non c'è un altro torneo in quel periodo
DOPO:	C'è un solo torneo in quel periodo

OPERAZIONE	**iscrizione_torneo**
SCOPO:	Iscrivere i clienti registrati ai tornei
ARGOMENTI:	Codice fiscale cliente registrato, località, data inizio torneo
RISULTATO	Aggiunta iscrizione al torneo /errore
ERRORE:	Tessera scaduta Troppo tardi Torneo non trovato Codice fiscale errato Torneo al completo
USA:	Tessera, torneo
MODIFICA:	Torneo
PRIMA:	C'è un torno e c'è posto in quella data
DOPO:	Il cliente viene registrato come partecipante

OPERAZIONE	**lettura_entrate**
SCOPO:	Conoscere tutte le entrate di denaro di una filiale in un dato periodo
ARGOMENTI:	Località filiale, mese, anno
RISULTATO	Valore dell'entrate totali
ERRORE:	
USA:	Vende, prodotto, torneo, partecipa, affitta_tavolo, tavolo, affitta_postazione, postazione
MODIFICA:	
PRIMA:	Sono presenti delle entrate
DOPO:	Viene mostrata l'entrata totale

OPERAZIONE	**lettura_uscite**
SCOPO:	Conoscere tutte le uscite di denaro di una filiale in un dato periodo
ARGOMENTI:	Località filiale, mese, anno
RISULTATO	Valore dell'uscite totali
ERRORE:	
USA:	Stipendi, impiegato, direttore, amministratore, forniture, torneo
MODIFICA:	
PRIMA:	Sono presenti delle uscite
DOPO:	Viene mostrata l'uscita totale

OPERAZIONE	**registra_cliente**
SCOPO:	Registrare un cliente
ARGOMENTI:	Codice fiscale cliente, e-mail del cliente
RISULTATO	Aggiunta di un cliente registrato /errore
ERRORE:	Codice fiscale errato
USA:	Cliente_reg, tessera
MODIFICA:	Cliente_reg, tessera
PRIMA:	Il cliente non è registrato
DOPO:	il cliente è registrato

Di seguito viene rappresentata la tavola delle operazioni.
In questo caso si ipotizza che si creino: 2 tornei al mese, poiché un torneo non dura più di una

settima(da notare che i tornei organizzati non sono di un'importanza rilevante quindi è giusto ipotizzare che essi possano terminare in breve termine), una distribuzione al mese, poiché essa può durare al massimo un mese, una lettura sia di entrate monetarie che di uscite al mese.

Nota: il numero di volte in cui l'operazione è lanciata non corrisponde necessariamente al numero di volte in cui ha successo.

OPERAZIONE	**TIPO[7]**	**VOLUME**	**PERIODO**
Crea distribuzione	B	1	Mese
Crea torneo	B	2	Mese
Iscrizione torneo	B	> Del numero massimo partecipanti dei tornei presenti nel mese considerato	Mese
Lettura entrate	B	1	Mese
Lettura uscite	B	1	Mese
Registrazione cliente	B	> Del numero dei clienti non registrati	Giorno

Volumi

La tavola dei volumi, riportata di seguito, oltre a riportare il numero verosimile di tuple presenti in ciascuna tabella una volta che il database sia a regime rappresenta anche il suo incremento atteso in un periodo di tempo prefissato.

Nel nostro caso le tuple di evento sono conservate per un anno, le tuple di approvvigionamento e fornitura sono conservate per 3 anni, le tuple di stipendio sono conservate per 10 anni mentre le tuple facenti riferimento alle altre tabelle del progetto non vengono mai cancellate.

Nota: vengono cancellate anche le tuple che fanno riferimento alle tuple cancellate.

Tabella	**Tipo[8]**	**Volume**	**Incremento**	**Periodo**
Cliente	E	32	300	giorno
Cliente_reg	E	16	150	giorno
Tessera	ED	16	150	giorno
Lavoratore	E	181	0	anno
Impiegato	E	96	0	anno
Direttore	E	32	0	anno
Amministratore	E	53	0	anno
Stipendio	ED	362	0	mese
Filiale	E	15	1	2 anni
Reparto	ED	32	0	anno
Tavolo	ED	61	5	anno
Affitta_tavolo	A	29	1000	giorno
Giochi_tavolo_rep	AM	72	10	mese
Postazione	ED	71	5	anno
Afitta_postazione	A	24	1000	giorno
Videogiochi_rep	AM	51	10	mese
Casa produttrice	E	15	10	anno

[7] B sta per *batch*

[8] **E** sta per *entità*, **ED** sta per *entità debole*, **A** sta per *associazione*, **AM** sta per *attributo multivalore*.

Prodotto	E	75	10	mese

Tabella	**Tipo**	**Volume**	**Incremento**	**Periodo**
Genere	AM	30	5	mese
Vende	A	86	300	giorno
Gadget	E	15	3	mese
Periferica	E	15	3	mese
Console	E	15	1	anno
Gioco_tavolo	E	15	2	mese
Videogioco	E	15	2	mese
Fornitore	E	15	2	anno
Fornitura	ED	16	0	mese
Prodotti_contenuti	A	86	0	mese
Approvvigionamento	ED	15	0	mese
Consiste	A	154	0	mese
Evento	ED	30	0	mese
Torneo	ED	15	0	mese
Distribuzione	ED	15	0	mese
Partecipa	A	63	0	mese
Gadget_dist	A	15	0	mese
Offerta	E	15	1	mese
Ottiene	A	26	100	mese
Crea	A	15	1	mese
Organizza	A	30	0	mese
Soggetto	A	26	100	mese

Vincoli di integrità

Sono detti statici i vincoli di integrità che limitano i valori assumibili da alcuni attributi, cioè limitano il loro dominio di valori, indipendentemente dal tempo.

Sono detti dinamici i vincoli che riguardano i valori che cambiano nel tempo; anche molte regole di business si presentano come vincoli dinamici.

Qui di seguito riportiamo i vincoli più importanti nel nostro DB(escludendo i vincoli di chiave primaria e chiave esterna).

Statici:

- Non sono ammesse stringhe per i codici fiscali con una lunghezza minore di 16.
- L'e-mail del cliente registrato deve terminare con una delle seguenti stringhe per essere valida: @%.com, @%.net, @%.it, @%.org (il carattere '%' sostituisce un numero arbitrario di zero o più caratteri).
- Per identificare il reparto viene utilizzata la variabile *flag_reparto* che può assumere solo i seguenti valori: L[9], VE[10], VI[11].
- La data di fine degli eventi deve essere successiva alla data di inizio.

[9] ludico
[10] vendita
[11] videoludico

* Le piattaforme ammesse per i tornei sono: SNES Classic Mini, NES Classic Mini, Sega Megadrive Mini, PlayStation Classic, Xbox 360, PlayStation 3, Wii U, Xbox One, PlayStation 4, Nintendo Switch, PC.
* il numero massimo dei partecipanti deve essere un valore compreso nell'intervallo (4,31).

Dinamici:

* La data di scadenza immessa per una tessera deve essere successiva, di almeno un anno, dalla data di registrazione dell'utente.
* la data di scadenza di un'offerta deve essere successiva, di almeno una settimana, alla data inizio dell'offerta.
* Un tavolo non può essere prenotato se esso è occupato, se il negozio è chiuso a quell'ora oppure se il negozio chiude prima della fine del tempo di affitto.
* Una postazione non può essere prenotata se essa è occupata, se il negozio è chiuso a quell'ora oppure se il negozio chiude prima della fine del tempo di affitto.
* Viene aggiunto un partecipante al torneo solo se ci sono posti disponibili.
* Non è possibile inviare approvvigionamenti a filiali senza reparto vendita.
* Non posso esserci 2 tornei nello stesso periodo.
* Non posso esserci 2 distribuzioni nello stesso periodo.
* Calcolo automatico della generazione delle console.

Implementazione

CREAZIONE UTENTI
DDL
DML
TRIGGER
PROCEDURA
DCL
SCHEDULER

Implementazione

Completata la progettazione, bisogna scrivere il codice eseguibile, ed è quanto illustrato in questo capitolo.
Il codice si riferisce al DBMS Oracle 10g XE ed il linguaggio dottato è il PL/SQL. L'ordine con cui lanciare gli script non è necessariamente quello con cui sono presentati nel capitolo.
Tutti i nomi arbitrari sono in minuscolo, mentre le parole riservate in maiuscolo, opportunamente evidenziate in blu.

Creazione utenti

Il primo passo da compiere è accedere al DBMS come amministratore di sistema e creare
l'utente proprietario della base di dati. E'possibile creare contestualmente anche altri utenti, sebbene essi non possano ricevere alcun privilegio di oggetto, poiché lo schema ancora non esiste.

```
CREATE USER db_owner              IDENTIFIED BY ADMIN ;
CREATE USER proprietari_catena    IDENTIFIED BY GAMESTOP ;
CREATE USER direttore             IDENTIFIED BY MIDUSER ;
CREATE USER amministratore        IDENTIFIED BY SUPERUSER ;
CREATE USER impiegato             IDENTIFIED BY LOWUSER;
GRANT ALL PRIVILEGES TO db_owner ;
```

In seguito, occorre disconnettersi e riconnettersi con le credenziali dell'utente db_owner, che è l'amministratore della nuova base di dati e che avrà i permessi per creare tutti gli oggetti che occorrono[12].

Data Definition Language

il **Data Definition Language** (**DDL**) è un linguaggio, parte del linguaggio SQL, che permette di creare, modificare o eliminare gli oggetti in un database ovvero agire sullo schema di database. Sono i comandi DDL a definire la struttura del DB e quindi l'organizzazione logica dei dati in esso contenuti.
Le varie tabelle sono create mediante le istruzioni di **CREATE TABLE** che includono tutti i vincoli di integrità esprimibili nel modello relazionale (vincoli di integrità referenziale, vincoli di chiave, vincoli, vincoli d'integrità statici etc.).

CLIENTE

La tabella cliente è popolata dagli impiegati tramite inserimento diretto, dunque è necessario includere tutti i controlli di integrità possibili nel DDL.
Si noti il vincolo di tupla sul codice fiscale, che deve essere di lunghezza pari a 16.

```
--%%%%%          CLIENTE
CREATE TABLE cliente
(
```

[12] *db_prenotati* ha ricevuto tutti i privilegi ed è quindi amministratore dell'intero DBMS.

```
        cf          CHAR(16),
        data_di_nascita     DATE            NOT NULL,
        nome                VARCHAR(15)     NOT NULL,
        cognome             VARCHAR(15)     NOT NULL,
        secondo_nome        VARCHAR(15),
        citta_residenza     VARCHAR(25)     NOT NULL,
        CONSTRAINT pk_cli   PRIMARY KEY(cf),
        CONSTRAINT          size_cf_cliente CHECK( length(cf) = 16 )
);
```

CLIENTE REGISTRATO - TESSERA

La tabella cliente registrato e la tabella tessera è popolata tramite la procedura *registra_cliente*, lanciata dall'impiegato, la maggior parte dei vincoli di integrità è gestita all'interno della procedura stessa, rendendone superflua la definizione nel DDL.

```
--%%%%%         CLIENTE REGISTRATO
CREATE TABLE cliente_reg
(
        cf_cliente          CHAR(16),
        e-mail              VARCHAR(35)     NOT NULL ,
        data_registrazione  DATE            NOT NULL ,
        CONSTRAINT pk_cli_reg   PRIMARY KEY(cf_cliente) ,
        CONSTRAINT fk_reg_cli   FOREIGN KEY(cf_cliente)   REFERENCES cliente(cf)
            ON DELETE CASCADE,
        CONSTRAINT email_valide CHECK( email like '%@%.com' or email like '%@%.net' or
email like '%@%.it' or email like '%@%.org')
);

--%%%%%         TESSERA
CREATE TABLE tessera
(
        cod_tessera         CHAR(10),
        data_scadenza       DATE        NOT NULL,
        cf_cli_reg          CHAR(16),
        CONSTRAINT          pk_tessera  PRIMARY KEY(cod_tessera,cf_cli_reg),
        CONSTRAINT          fk_tes_reg  FOREIGN KEY(cf_cli_reg)   REFERENCES
cliente_reg(cf_cliente)
            ON DELETE CASCADE
);
```

LAVORATORE

La tabella lavoratore è popolata dai proprietari della catena tramite inserimento diretto, dunque è necessario includere tutti i controlli di integrità possibili nel DDL.
La tabella lavoratore rappresenta la superclasse di: impiegato, direttore e amministratore. Anche per lavoratore è presente il vincolo di tupla sul codice fiscale, che deve essere di lunghezza pari a 16.

```
--%%%%%         LAVORATORE
```

```
CREATE TABLE lavoratore(
        cf                  CHAR(16),
        data_di_nascita     DATE                NOT NULL,
        nome                VARCHAR(15)         NOT NULL ,
        cognome             VARCHAR(15)         NOT NULL ,
        secondo_nome        VARCHAR(15) ,
        citta_residenza     VARCHAR(20)         NOT NULL ,
        data_assunzione     DATE                NOT NULL,
        CONSTRAINT          pk_lav          PRIMARY KEY(cf),
        CONSTRAINT          size_cf_lavoratore          CHECK( length(cf) = 16 )
        );
```

IMPIEGATO

La tabella impiegato è popolata dai proprietari della catena tramite inserimento diretto, dunque è necessario includere tutti i controlli di integrità possibili nel DDL.
La tabella impiegato rappresenta la specializzazione di lavoratore.
Per l'impiegato è presente il vincolo che diminuisce i valori sul dominio dell'attributo flag_rep_imp, poiché i valori ammissibili da esso sono solo la flag che identificano i tre possibili reparti.

```
CREATE TABLE impiegato(
        cf_imp              CHAR(16),
        reparto_loc_imp     VARCHAR(20)         NOT NULL,
        flag_rep_imp        CHAR(2)             CHECK(flag_rep_imp in ( 'L', 'VE', 'VI' )),
        CONSTRAINT          pk_imp          PRIMARY KEY(cf_imp),
        CONSTRAINT          fk_imp_rep      FOREIGN KEY(flag_rep_imp,reparto_loc_imp)
        REFERENCES reparto(flag_reparto,loc_reparto)
                ON DELETE CASCADE,
        CONSTRAINT          fk_imp_lav      FOREIGN KEY(cf_imp)         REFERENCES
lavoratore(cf)
                ON DELETE CASCADE
        );
```

DIRETTORE

La tabella direttore è popolata dai proprietari della catena tramite inserimento diretto, dunque è necessario includere tutti i controlli di integrità possibili nel DDL.
La tabella direttore rappresenta la specializzazione di lavoratore.
Per il direttore è presente il vincolo che diminuisce i valori sul dominio dell'attributo flag_rep_dir, poiché i valori ammissibili da esso sono solo i flags che identificano i tre possibili reparti.
Poiché ogni reparto può avere solo un direttore viene definito un ulteriore vincolo di univocità che garantisce questa condizione.
Questo vincolo si aggiunge a quello di chiave primaria, in modo che in fase di popolamento, perché una tupla sia legale, debba soddisfare entrambe le univocità: quella del codice fiscale e quella sulla località del reparto e del flag_rep_dir.

```
--%%%%%             DIRETTORE
CREATE TABLE direttore(
        cf_dir              CHAR(16),
```

```
    reparto_loc_dir     VARCHAR(20)     NOT NULL,
    flag_rep_dir        CHAR(2)         CHECK(flag_rep_dir in ( 'L', 'VE', 'VI' )),
    UNIQUE(reparto_loc_dir,flag_rep_dir),
    CONSTRAINT          pk_dir          PRIMARY KEY(cf_dir),
    CONSTRAINT          fk_dir_rep      FOREIGN KEY(reparto_loc_dir,flag_rep_dir)
    REFERENCES reparto(loc_reparto,flag_reparto)
        ON DELETE CASCADE,
    CONSTRAINT          fk_dir_lav      FOREIGN KEY(cf_dir)  REFERENCES lavoratore(cf)
        ON DELETE CASCADE
    );
```

AMMINISTATORE

La tabella amministratore è popolata dai proprietari della catena tramite inserimento diretto, dunque è necessario includere tutti i controlli di integrità possibili nel DDL.
La tabella amministratore rappresenta la specializzazione di lavoratore.

```
--%%%%%         AMMINISTRATORE
CREATE TABLE amministratore(
    cf_amm      CHAR(16),
    loc_filiale VARCHAR(20)     NOT NULL,
    CONSTRAINT          pk_amm          PRIMARY KEY(cf_amm),
    CONSTRAINT          fk_amm_fil      FOREIGN KEY(loc_filiale)    REFERENCES
filiale(localita)
        ON DELETE CASCADE,
    CONSTRAINT          fk_amm_lav      FOREIGN KEY(cf_amm)     REFERENCES
lavoratore(cf)
        ON DELETE CASCADE
);
```

STIPENDIO

La tabella stipendio è popolata dai proprietari della catena tramite inserimento diretto, dunque è necessario includere tutti i controlli di integrità possibili nel DDL.
Oltre al vincolo della chiave primaria abbiamo bisogno di un ulteriore vincolo univocità, poiché ad ogni lavoratore deve essere pagato un solo stipendio ogni mese.
questo problema viene risolto definendo un vincolo di integrità, esterno alla tablella, che garantisce le **DATE** univoche per lavoratore.
Questo vincolo si aggiunge a quello di chiave primaria, in modo che in fase di popolamento, perché una tupla sia legale, debba soddisfare entrambe le univocità.

```
--%%%%%         STIPENDIO
CREATE TABLE stipendio(
    numero_bollo    CHAR(16),
    denaro          FLOAT       NOT NULL,
    data_erogazione DATE        NOT NULL,
    cf_lav          CHAR(16)    NOT NULL,
    CONSTRAINT          pk_stip         PRIMARY KEY(numero_bollo),
    CONSTRAINT          fk_stip_lav     FOREIGN KEY(cf_lav)         REFERENCES
lavoratore(cf)
        ON DELETE CASCADE
```

```
);

CREATE UNIQUE INDEX stipendio_univoco
ON stipendio ( TRUNC ( data_erogazione ), cf_lav
);
```

FILIALE

La tabella filiale include tutte le informazioni riguardanti una filiale.

```
--%%%%%		FILIALE
CREATE TABLE filiale(
	localita	VARCHAR(25) ,
	cap		CHAR(5)		NOT NULL,
	via		VARCHAR(30)	NOT NULL,
	CONSTRAINT		pk_fil		PRIMARY KEY(localita)
);
```

REPARTO

La tabella reparto contiene le informazioni riguardanti i reparti.
Ogni reparto viene identificato dal flag del reparto che grazie al vincolo di integrità statica può assumere solo tre valori ('L', 'VE', 'VI').

```
--%%%%%		REPARTO
CREATE TABLE reparto(
	loc_reparto		VARCHAR(25) ,
	flag_reparto		CHAR(2)	CHECK(flag_reparto in ( 'L', 'VE', 'VI' )),
	CONSTRAINT		pk_rep		PRIMARY KEY(loc_reparto,flag_reparto),
	CONSTRAINT		fk_rep_fil		FOREIGN KEY(loc_reparto)	REFERENCES
filiale(localita)
		ON DELETE CASCADE
);
```

TAVOLO

La tabella tavolo contiene le informazioni riguardanti i tavoli.
Ogni tavolo può far parte solo ad una filiale che ha un reparto ludico, questo vincolo viene espresso dal vincolo di integrità statico che controlla che il valore dato a flag_rep sia uguale ad 'L'.

```
--%%%%%		TAVOLO
CREATE TABLE tavolo(
	numero_tavolo		INT,
	conto_orario		FLOAT		NOT NULL,
	loc_rep			VARCHAR(25),
	flag_rep		CHAR(2)	DEFAULT 'L'	CHECK(flag_rep = 'L'),
	CONSTRAINT	pk_tav		PRIMARY KEY(numero_tavolo, loc_rep, flag_rep),
	CONSTRAINT	fk_tav_rep	FOREIGN KEY(loc_rep, flag_rep)	REFERENCES
reparto(loc_reparto, flag_reparto)
		ON DELETE CASCADE
);
```

AFFITTA_TAVOLO

La tabella stipendio è popolata dagli impiegati tramite inserimento diretto, dunque è necessario includere tutti i controlli di integrità possibili nel DDL.
In questa tabella rappresenta la relazione che esiste tra tavolo e cliente_registrato.
I clienti non possono affittare un tavolo per più di 6 ore, questo viene espresso dal vincolo di identità statico.
Ogni tupla di affitta_postazione può far riferimento solo ad una filiale che ha un reparto videoludico, questo vincolo viene espresso dal vincolo di integrità statico che controlla che il valore dato a flag_rep sia uguale ad 'L'.

```
CREATE TABLE affitta_tavolo(
        numero_tav          INT,
        cf_cliente_reg      CHAR(16),
        loc_rep             VARCHAR(25),
        flag_rep            CHAR(2)     DEFAULT 'L'     CHECK(flag_rep = 'L'),
        ore_affitto         INT         NOT NULL        CHECK(ore_affitto <= 6),
        data_ora_inizio     DATE        NOT NULL,
        CONSTRAINT  pk_aff_tav          PRIMARY KEY
(numero_tav,loc_rep,flag_rep,cf_cliente_reg,data_ora_inizio),
        CONSTRAINT  fk_aff_tav          FOREIGN KEY(numero_tav,loc_rep,flag_rep)
        REFERENCES tavolo(numero_tavolo,loc_rep,flag_rep)
                ON DELETE CASCADE,
        CONSTRAINT  fk_afftav_clireg            FOREIGN KEY(cf_cliente_reg)
        REFERENCES cliente_reg(cf_cliente)
                ON DELETE CASCADE
);
```

GIOCHI_TAVOLO_REP

Questa tabella viene rappresenta l'attributo multivalore dell'entità Ludico presente nel diagramma EE/R.

```
--%%%%%             GIOCHI_TAVOLO_REP
CREATE TABLE giochi_tavolo_rep(
        loc_rep_tav         VARCHAR(25),
        flag_rep_tav        CHAR(2)     DEFAULT 'L'     CHECK(flag_rep_tav = 'L'),
        nome                VARCHAR(25),
        CONSTRAINT  pk_giochi_tav       PRIMARY KEY (loc_rep_tav,flag_rep_tav,nome),
        CONSTRAINT  fk_giotav_rep       FOREIGN KEY(loc_rep_tav,flag_rep_tav)
        REFERENCES reparto(loc_reparto,flag_reparto)
                ON DELETE CASCADE
);
```

POSTAZIONE

La tabella postazione contiene le informazioni riguardanti i postazioni.
Ogni postazione può far parte solo ad una filiale che ha un reparto videoludico, questo vincolo viene espresso dal vincolo di integrità statico che controlla che il valore dato a

flag_rep sia uguale ad 'VI'.
Un ulteriore vincolo statico fa si che solo alcune console sono ammissibili.

```
--%%%%%          POSTAZIONE
CREATE TABLE postazione(
        numero_postazione  INT,
        conto_orario  FLOAT NOT NULL,
        nome_console        VARCHAR(20)         NOT NULL      CHECK(nome_console in ( 'SNES
Classic Mini', 'NES Classic Mini', 'Sega Megadrive Mini', 'PlayStation Classic', 'Xbox 360',
'PlayStation 3', 'Wii U', 'Xbox One', 'PlayStation 4', 'Nintendo Switch', 'PC' )),
        loc_rep             VARCHAR(25),
        flag_rep            CHAR(2)       DEFAULT 'VI'   CHECK(flag_rep = 'VI'),
        CONSTRAINT  pk_pos        PRIMARY KEY(numero_postazione, loc_rep, flag_rep),
        CONSTRAINT  fk_pos_rep          FOREIGN KEY(loc_rep, flag_rep)     REFERENCES
reparto(loc_reparto, flag_reparto)
        ON DELETE CASCADE
);
```

AFFITTA_POSTAZIONE

La tabella stipendio è popolata dagli impiegati tramite inserimento diretto, dunque è necessario includere tutti i controlli di integrità possibili nel DDL.
In questa tabella rappresenta la relazione che esiste tra postazione e cliente_registrato.
I clienti non possono affittare una postazione per più di 6 ore, questo viene espresso dal vincolo di identità statico.
Ogni tupla di affitta_postazione può far riferimento solo ad una filiale che ha un reparto videoludico, questo vincolo viene espresso dal vincolo di integrità statico che controlla che il valore dato a flag_rep sia uguale ad 'VI'.

```
--%%%%%          AFFITTA_POSTAZIONE
CREATE TABLE affitta_postazione(
        numero_pos  INT     ,
        cf_cliente_reg              CHAR(16),
        ore_affitto         INT             NOT NULL      CHECK(ore_affitto <= 6),
        data_ora_inizio             DATE,
        loc_rep             VARCHAR(25),
        flag_rep            CHAR(2)       DEFAULT 'VI'   CHECK(flag_rep = 'VI'),
        CONSTRAINT  pk_aff_pos          PRIMARY
KEY(numero_pos,cf_cliente_reg,loc_rep,flag_rep),
        CONSTRAINT  fk_aff_pos          FOREIGN KEY(numero_pos,loc_rep,flag_rep)
        REFERENCES postazione(numero_postazione,loc_rep,flag_rep)
        ON DELETE CASCADE,
        CONSTRAINT  fk_affpos_clireg            FOREIGN KEY(cf_cliente_reg)
        REFERENCES cliente_reg(cf_cliente)
        ON DELETE CASCADE
);
```

VIDEOGIOCHI_REP

Questa tabella viene rappresenta l'attributo multivalore dell'entità Videoludico presente nel diagramma EE/R.

```
--%%%%%          VIDEOGIOCHI_REP
CREATE TABLE videogiochi_rep(
        loc_rep_video           VARCHAR(25),
        flag_rep_pos    CHAR(2)     DEFAULT 'VI'    CHECK(flag_rep_pos = 'VI'),
        nome  VARCHAR(25),
        CONSTRAINT  pk_videogiochi          PRIMARY
KEY(loc_rep_video,flag_rep_pos,nome),
        CONSTRAINT  fk_vidgio_rep       FOREIGN KEY(loc_rep_video,flag_rep_pos)
        REFERENCES reparto(loc_reparto,flag_reparto)
        ON DELETE CASCADE
);
```

CASAPRODUTTRICE

Qui di seguito viene riportata la tabella casa produttrice.

```
--%%%%%          CASAPRODUTTRICE
CREATE TABLE casaproduttrice
(
        partita_iva     CHAR(11),
        nome  VARCHAR(20)       NOT NULL,
        nazionalita         VARCHAR(16)     NOT NULL,
        CONSTRAINT  pk_casaproduttrice  PRIMARY KEY(partita_iva)
);
```

PRODOTTO

Questa tabella rappresenta l'aggregazione dell'entità: Gadget, Videogioco, GiocoDaTavolo, Periferiche, Console.

```
--%%%%%          PRODOTTO
CREATE TABLE prodotto
(
        codice_prodotto     VARCHAR(16),
        anno_produzione      DATE   NOT NULL,
        prezzo          FLOAT  NOT NULL,
        nome  VARCHAR(80)  NOT NULL,
        partita_iva_casaprod        CHAR(11)      NOT NULL,
        CONSTRAINT pk_prod PRIMARY KEY(codice_prodotto),
        CONSTRAINT fk_imp_casaprod      FOREIGN KEY(partita_iva_casaprod)
        REFERENCES   casaproduttrice(partita_iva)
                ON DELETE CASCADE
);
```

VENDE

La tabella impiegato è popolata dai proprietari della catena tramite inserimento diretto, dunque è necessario includere tutti i controlli di integrità possibili nel DDL.
Contiene tutte le informazioni delle vendite effettuate.
Ogni tupla di vende deve fare riferimento ad una filiale che ha un reparto di vendita, questo

vincolo viene espresso dal vincolo di integrità statico che controlla che il valore dato a flag_rep sia uguale ad 'VE'.

```
--%%%%%          VENDE
CREATE TABLE vende
(
        loc_rep         VARCHAR(25) ,
        flag_rep                CHAR(2)         DEFAULT  'VE'  CHECK(flag_rep = 'VE'),
        cod_prod        VARCHAR(16),
        cf_cli   CHAR(16) ,
        data_e_ora   DATE   NOT NULL,
        quantita        INTEGER                 NOT NULL,
        CONSTRAINT pk_vende       PRIMARY
KEY(loc_rep,cod_prod,flag_rep,cf_cli,data_e_ora),
        CONSTRAINT fk_vende_cli   FOREIGN KEY(cf_cli)  REFERENCES cliente(cf)
                ON DELETE CASCADE,
        CONSTRAINT fk_vende_rep  FOREIGN KEY(loc_rep,flag_rep)      REFERENCES
reparto(loc_reparto,flag_reparto)
                ON DELETE CASCADE,
        CONSTRAINT fk_vende_prod          FOREIGN KEY(cod_prod) REFERENCES
prodotto(codice_prodotto)
                ON DELETE CASCADE
);
```

GADGET

Qui di seguito viene riportata la tabella gadget.

```
--%%%%%          GADGET
CREATE TABLE gadget
(
        codice_gadget VARCHAR(16) ,
        tipo VARCHAR(20),
        CONSTRAINT  pk_gad              PRIMARY KEY(codice_gadget),
        CONSTRAINT  fk_gad_prod         FOREIGN KEY(codice_gadget)      REFERENCES
prodotto(codice_prodotto)
                ON DELETE CASCADE
);
```

PERIFERICA

Qui di seguito viene riportata la tabella periferica.

```
--%%%%%          PERIFERICA
CREATE TABLE periferica
(
        codice_periferica    VARCHAR(16),
        CONSTRAINT           pk_per PRIMARY KEY(codice_periferica),
        CONSTRAINT           fk_per_prod   FOREIGN KEY(codice_periferica) REFERENCES
prodotto(codice_prodotto)
                ON DELETE CASCADE
```

```
);
```

CONSOLE

Qui di seguito viene riportata la tabella console.

```
--%%%%%		CONSOLE
CREATE TABLE console
(
	codice_console VARCHAR(16) NOT NULL,
	generazione INTEGER,
	CONSTRAINT pk_console  PRIMARY KEY(codice_console),
	CONSTRAINT fk_console_prod FOREIGN KEY(codice_console) REFERENCES
prodotto(codice_prodotto)
		ON DELETE CASCADE
);
```

GIOCO_TAVOLO

Qui di seguito viene riportata la tabella gioco_tavolo.

```
--%%%%%		GIOCO_TAVOLO
CREATE TABLE gioco_tavolo
(
	codice_gioco_tavolo	VARCHAR(16),
	pegi	INTEGER	NOT NULL,
	CONSTRAINT	pk_gio_tav	PRIMARY KEY(codice_gioco_tavolo),
	CONSTRAINT	fk_gio_tav_prod	FOREIGN KEY(codice_gioco_tavolo)
REFERENCES prodotto(codice_prodotto)
		ON DELETE CASCADE
);
```

VIDEOGIOCO

Qui di seguito viene riportata la tabella videogioco.

```
--%%%%%		VIDEOGIOCO
CREATE TABLE videogioco
(
	codice_videogioco VARCHAR(16),
	pegi	INTEGER		NOT NULL,
	CONSTRAINT pk_vid PRIMARY KEY(codice_videogioco),
	CONSTRAINT fk_vid_prodotto FOREIGN KEY(codice_videogioco) REFERENCES
prodotto(codice_prodotto)
		ON DELETE CASCADE
);
```

GENERE

Questa tabella viene rappresenta l'attributo multivalore dell'entità Videogioco e GiocoDaTavolo presente nel diagramma EE/R.

```
--%%%%%		GENERE
```

```
CREATE TABLE genere
(
	codice_prodotto VARCHAR(16) NOT NULL,
	tipo VARCHAR(30),
	CONSTRAINT pk_gen PRIMARY KEY(codice_prodotto,tipo),
	CONSTRAINT fk_gen_prod FOREIGN KEY(codice_prodotto) REFERENCES
prodotto(codice_prodotto)
		ON DELETE CASCADE
);
```

FORNITORE

Qui di seguito viene riportata la tabella fornitore.

```
--%%%%%		FORNITORE
CREATE TABLE fornitore
(
	partita_iva	CHAR(11),
	nome	VARCHAR(16)		NOT NULL,
	nazionalita		VARCHAR(16)		NOT NULL,
	CONSTRAINT	pk_fornitore	PRIMARY KEY(partita_iva)
);
```

FORNITURA

Qui di seguito viene riportata la tabella fornitore.

```
--%%%%%		FORNITURA
CREATE TABLE fornitura
(
	data DATE,
	partita_iva_for		CHAR(11)	NOT NULL,
	costo	FLOAT ,
	CONSTRAINT	pk_fornitura	PRIMARY KEY(data,partita_iva_for),
	CONSTRAINT	fk_fornitura_fornitore		FOREIGN KEY(partita_iva_for) REFERENCES
fornitore(partita_iva)
		ON DELETE CASCADE
);
```

PRODOTTI_CONTENUTI

Qui di seguito viene riportata la tabella prodotti_contenuti.

```
--%%%%%		PRODOTTI_CONTENUTI
CREATE TABLE prodotti_contenuti
(
	data_for	DATE,
	partita_iva_for		CHAR(11) ,
	codice_prod	VARCHAR(16),
	quantita	INTEGER	DEFAULT(1),
	CONSTRAINT	pk_prodcont	PRIMARY KEY(data_for,partita_iva_for,codice_prod),
```

```
        CONSTRAINT  fk_prodcont_fornitura        FOREIGN KEY(data_for,partita_iva_for)
        REFERENCES fornitura(data,partita_iva_for)
                ON DELETE CASCADE,
        CONSTRAINT  fk_prodcont_prod    FOREIGN KEY(codice_prod) REFERENCES
prodotto(codice_prodotto)
                ON DELETE CASCADE
);
```

APPROVVIGGIONAMENTO

La tabella approvviggionamento è popolata dagli amministratori tramite inserimento diretto, dunque è necessario includere tutti i controlli di integrità possibili nel DDL.
Qui di seguito viene riportata la tabella approvviggionamento.

```
--%%%%%             APPROVVIGGIONAMENTO
CREATE TABLE approvviggionamento
(
        data   DATE,
        loc_filiale     VARCHAR(25),
        CONSTRAINT  pk_app      PRIMARY KEY(data,loc_filiale),
        CONSTRAINT  fk_app_cons         FOREIGN KEY(loc_filiale)     REFERENCES
filiale(localita)
                ON DELETE CASCADE
);
```

CONSISTE

La tabella consiste è popolata dagli amministratori tramite inserimento diretto, dunque è necessario includere tutti i controlli di integrità possibili nel DDL.
Qui di seguito viene riportata la tabella consiste.

```
--%%%%%             CONSISTE
CREATE TABLE consiste
(
        data_app     DATE,
        loc_fil  VARCHAR(25),
        codice_prod         VARCHAR(16),
        quantita     INTEGER,
        CONSTRAINT  pk_cons             PRIMARY KEY(data_app,loc_fil,codice_prod),
        CONSTRAINT  fk_cons_app         FOREIGN KEY(data_app,loc_fil)     REFERENCES
approvviggionamento(data,loc_filiale)
                ON DELETE CASCADE,
        CONSTRAINT  fk_cons_prod        FOREIGN KEY(codice_prod) REFERENCES
prodotto(codice_prodotto)
                ON DELETE CASCADE
);
```

EVENTO

La tabella torneo è popolata tramite la procedura inserisci_torneo e crea_distribuzione, lanciate dall'amministratore, e la maggior parte dei vincoli è gestita all'interno della

procedura stessa, rendendone superflua la definizione nel DDL.
Qui di seguito viene riportata la tabella evento.

```
--%%%%%          EVENTO
CREATE TABLE evento
(
        localita_evento     VARCHAR(25),
        data_inizio         DATE,
        data_fine           DATE   NOT   NULL,
        CONSTRAINT  pk_eve        PRIMARY KEY(data_inizio, localita_evento),
        CONSTRAINT  fk_eve_filiale  FOREIGN KEY(localita_evento)          REFERENCES
filiale(localita)
                ON DELETE CASCADE,
        CONSTRAINT  data_ok_evento          CHECK(data_fine > data_inizio)
);
```

TORNEO

La tabella torneo è popolata tramite la procedura inserisci_torneo, lanciata dall' amministratore, e la maggior parte dei vincoli è gestita all'interno della procedura stessa, rendendone superflua la definizione nel DDL.
Qui di seguito viene riportata la tabella torneo.

```
--%%%%%          TORNEO
CREATE TABLE torneo
(
        data_inizio_torneo  DATE,
        localita_torneo      VARCHAR(25),
        quota FLOAT         NOT NULL,
        premio              INTEGER             NOT NULL,
        nome_gioco          VARCHAR(30)         NOT NULL,
        piattaforma         VARCHAR(20)         NOT NULL,
        max_partecipanti    INTEGER             DEFAULT       15,
        CONSTRAINT          max_ammessi             CHECK ( max_partecipanti > 4 AND
max_partecipanti < 31 ),
        CONSTRAINT          piattaforma_ammessa     CHECK( piattaforma in ( 'SNES Classic
Mini', 'NES Classic Mini', 'Sega Megadrive Mini', 'PlayStation Classic', 'Xbox 360', 'PlayStation 3',
'Wii U', 'Xbox One', 'PlayStation 4', 'Nintendo Switch', 'PC' )),
        CONSTRAINT          pk_tor          PRIMARY KEY(data_inizio_torneo,localita_torneo),
        CONSTRAINT          fk_tor_eve              FOREIGN
KEY(data_inizio_torneo,localita_torneo)             REFERENCES
evento(data_inizio,localita_evento)
                ON DELETE CASCADE
);
```

DISTRIBUZIONE

La tabella distribuzione è popolata tramite la procedura crea_distribuzione, lanciata dall'amministratore, e la maggior parte dei vincoli è gestita all'interno della procedura stessa,

rendendone superflua la definizione nel DDL.
Qui di seguito viene riportata la tabella distribuzione.

```
--%%%%%          DISTRIBUZIONE
CREATE TABLE distribuzione
(
        data_inizio_distribuzione      DATE,
        localita_distribuzioneVARCHAR(25),
        CONSTRAINT          pk_distr              PRIMARY
KEY(data_inizio_distribuzione,localita_distribuzione),
        CONSTRAINT          fk_distr_eve          FOREIGN
KEY(data_inizio_distribuzione,localita_distribuzione)          REFERENCES
evento(data_inizio,localita_evento)
                ON DELETE CASCADE
);
```

PARTECIPA

La tabella partecipa è popolata tramite la procedura iscrizione_torneo, lanciata dall'impiegato, e la maggior parte dei vincoli è gestita all'interno della procedura stessa, rendendone superflua la definizione nel DDL.

Qui di seguito viene riportata la tabella partecipa.

```
--%%%%%          PARTECIPA
CREATE TABLE partecipa
(
        loc_eve         VARCHAR(25),
        cf_cliente_reg          CHAR(16) ,
        data_inizio             DATE,
        CONSTRAINT              pk_part         PRIMARY KEY(cf_cliente_reg,data_inizio,loc_eve),
        CONSTRAINT              fk_part_eve             FOREIGN KEY(data_inizio,loc_eve)
        REFERENCES evento(data_inizio,localita_evento)
                ON DELETE CASCADE,
        CONSTRAINT              fk_part_clireg          FOREIGN KEY(cf_cliente_reg)
        REFERENCES cliente_reg(cf_cliente)
                ON DELETE CASCADE
);
```

GADGET_DIST

La tabella gadget_dist è popolata tramite la procedura iscrizione_torneo, lanciata dall'impiegato, e la maggior parte dei vincoli è gestita all'interno della procedura stessa, rendendone superflua la definizione nel DDL.
Qui di seguito viene riportata la tabella gadget_dist.

```
--%%%%%          GADGET_DIST
CREATE TABLE gadget_dist
(
        codice_gad    VARCHAR(16) ,
        loc_dist        VARCHAR(25),
```

```
        quantita        INTEGER                 NOT NULL,
        data_distribuzione    DATE,
        CONSTRAINT  pk_gad_dist             PRIMARY
KEY(codice_gad,loc_dist,data_distribuzione),
        CONSTRAINT  fk_gad_dist_gad             FOREIGN KEY(codice_gad)  REFERENCES
gadget(codice_gadget)
                ON DELETE CASCADE,
        CONSTRAINT  fk_gad_dist_dist            FOREIGN KEY(data_distribuzione,loc_dist)
        REFERENCES distribuzione(data_inizio_distribuzione,localita_distribuzione)
                ON DELETE CASCADE
);
```

OFFERTA

La tabella consiste è popolata dagli amministratori tramite inserimento diretto, dunque è necessario includere tutti i controlli di integrità possibili nel DDL.
Qui di seguito viene riportata la tabella offerta.

```
--%%%%%             OFFERTA
CREATE TABLE offerta
(
        codice_offerta      INTEGER,
        data_inizio         DATE   NOT NULL,
        data_fine           DATE   NOT    NULL,
        CONSTRAINT  pk_off        PRIMARY KEY(codice_offerta)
);
```

OTTIENE

Qui di seguito viene riportata la tabella ottiene.

```
--%%%%%             OTTIENE
CREATE TABLE ottiene
(
        cod_off             INTEGER,
        cf_cliente_reg      CHAR(16),
        CONSTRAINT  pk_ott        PRIMARY KEY(cod_off,cf_cliente_reg),
        CONSTRAINT          fk_ott_clireg       FOREIGN KEY(cf_cliente_reg)
        REFERENCES cliente_reg(cf_cliente)
                ON DELETE CASCADE,
        CONSTRAINT          fk_ott_off          FOREIGN KEY(cod_off)
        REFERENCES offerta(codice_offerta)
                ON DELETE CASCADE
);
```

CREA

Qui di seguito viene riportata la tabella crea.

```
--%%%%%             CREA
CREATE TABLE crea
```

```
(
        cod_off                 INTEGER,
        cf_org CHAR(16),
        CONSTRAINT  pk_cre          PRIMARY KEY(cf_org,cod_off),
        CONSTRAINT              fk_cre_clireg           FOREIGN KEY(cod_off)
        REFERENCES offerta(codice_offerta)
                ON DELETE CASCADE,
        CONSTRAINT              fk_amm_cre              FOREIGN KEY(cf_org)
        REFERENCES amministratore(cf_amm)
                ON DELETE CASCADE
);
```

ORGANIZZA

La tabella organizza è popolata tramite la procedura inserisci_torneo e crea_distribuzione, lanciate dall'amministratore, e la maggior parte dei vincoli è gestita all'interno della procedura stessa, rendendone superflua la definizione nel DDL.
Qui di seguito viene riportata la tabella organizza.

```
--%%%%%             ORGANIZZA
CREATE TABLE organizza
(
        loc_eve         VARCHAR(25),
        data_inizio_eve             DATE,
        cf_org CHAR(16),
        CONSTRAINT  pk_org          PRIMARY KEY(data_inizio_eve,cf_org),
        CONSTRAINT              fk_org_eve              FOREIGN KEY(data_inizio_eve,loc_eve)
        REFERENCES evento(data_inizio,localita_evento)
                ON DELETE CASCADE,
        CONSTRAINT              fk_amm_eve              FOREIGN KEY(cf_org)
        REFERENCES amministratore(cf_amm)
                ON DELETE CASCADE
);
```

SOGGETTO

Qui di seguito viene riportata la tabella soggetto.

```
--%%%%%             SOGGETTO
CREATE TABLE soggetto
(
        cod_off                 INTEGER,
        cod_prod        VARCHAR(16),
        CONSTRAINT  pk_sog          PRIMARY KEY(cod_off,cod_prod),
        CONSTRAINT              fk_sog_off              FOREIGN KEY(cod_off)
        REFERENCES offerta(codice_offerta)
                ON DELETE CASCADE,
        CONSTRAINT              fk_sog_prod             FOREIGN KEY(cod_prod)
        REFERENCES prodotto(codice_prodotto)
                ON DELETE CASCADE
```

);

Data Manipulation Language

A titolo di mero esempio vengono mostrate anche le operazioni di INSERT su tutte le tabelle, poiché solo l'amministratore potrebbe eseguirle tutte, eludendo i controlli presenti nelle rispettive procedure.

Nota: vengono mostrate solo alcune delle operazioni presenti nel file DML

```
--%%%%%           CLIENTE
INSERT INTO       cliente(cf, data_di_nascita, nome, cognome, secondo_nome,
citta_residenza)
      VALUES('MRTCLD94L12F839L',to_date('12/07/1994','dd/mm/yyyy'),'Claudio','Ma
rtini',NULL,'Napoli');
INSERT INTO       cliente(cf, data_di_nascita, nome, cognome, secondo_nome,
citta_residenza)
      VALUES('LMRGNM92E20A783Z',to_date('20/05/1992','dd/mm/yyyy'),'Geronimo','A
lmera','Cori','Benevento');
INSERT INTO       cliente(cf, data_di_nascita, nome, cognome, secondo_nome,
citta_residenza)
      VALUES('VREDMN90B10A783V',to_date('10/02/1990','dd/mm/yyyy'),'Damiano','Ve
ra',NULL,'Benevento');
INSERT INTO       cliente(cf, data_di_nascita, nome, cognome, secondo_nome,
citta_residenza)
      VALUES('VNNZRE91T58F839G',to_date('18/12/1991','dd/mm/yyyy'),'Ezra','Venni
',NULL,'Napoli');
INSERT INTO       cliente(cf, data_di_nascita, nome, cognome, secondo_nome,
citta_residenza)
      VALUES('CLMVLA91C22B905Y',to_date('22/03/1991','dd/mm/yyyy'),'Valio','Calm
i',NULL,'Casalnuovo');
INSERT INTO       cliente(cf, data_di_nascita, nome, cognome, secondo_nome,
citta_residenza)
      VALUES('FNLCLD98B24F839X',to_date('24/02/1998','dd/mm/yyyy'),'Claudio','Fi
nale',NULL,'Napoli');

--%%%%%           CLIENTE REGISTRATO
INSERT INTO       cliente_reg(cf_cliente, data_registrazione, email)
      VALUES('MRTCLD94L12F839L',to_date('12/07/2008','dd/mm/yyyy'),'emailfacile@
gmail.com');
INSERT INTO       cliente_reg(cf_cliente, data_registrazione, email)
      VALUES('LMRGNM92E20A783Z',to_date('20/05/2016','dd/mm/yyyy'),'cascataevent
o@gmail.com');
INSERT INTO       cliente_reg(cf_cliente, data_registrazione, email)
      VALUES('VREDMN90B10A783V',to_date('10/02/2009','dd/mm/yyyy'),'rossonto@hot
mail.it');

--%%%%%           TESSERA
INSERT INTO        tessera(cod_tessera, data_scadenza, cf_cli_reg)
      VALUES('9076754517',to_date('12/07/2020','dd/mm/yyyy'),'MRTCLD94L12F839L')
;
INSERT INTO        tessera(cod_tessera, data_scadenza, cf_cli_reg)
      VALUES('5679812465',to_date('20/05/2020','dd/mm/yyyy'),'LMRGNM92E20A783Z')
;
INSERT INTO        tessera(cod_tessera, data_scadenza, cf_cli_reg)
      VALUES('6516100638',to_date('10/02/2020','dd/mm/yyyy'),'VREDMN90B10A783V')
;

--FILIALI
```

```
INSERT INTO filiale(localita, cap, via) VALUES('Napoli',   '80121','via shishi
sonson');



--REPARTI
INSERT INTO reparto(loc_reparto, flag_reparto) VALUES('Napoli','L');
INSERT INTO reparto(loc_reparto, flag_reparto) VALUES('Napoli','VI');
INSERT INTO reparto(loc_reparto, flag_reparto) VALUES('Napoli','VE');

--TAVOLI
INSERT INTO tavolo(numero_tavolo,conto_orario,loc_rep) VALUES(1,2.5,'Napoli');
INSERT INTO tavolo(numero_tavolo,conto_orario,loc_rep) VALUES(2,2.5,'Napoli');
INSERT INTO tavolo(numero_tavolo,conto_orario,loc_rep) VALUES(3,2.5,'Napoli');
INSERT INTO tavolo(numero_tavolo,conto_orario,loc_rep) VALUES(4,2.5,'Napoli');
INSERT INTO tavolo(numero_tavolo,conto_orario,loc_rep) VALUES(5,2.5,'Napoli');
INSERT INTO tavolo(numero_tavolo,conto_orario,loc_rep) VALUES(6,2.5,'Napoli');
INSERT INTO tavolo(numero_tavolo,conto_orario,loc_rep) VALUES(7,3.5,'Napoli');
INSERT INTO tavolo(numero_tavolo,conto_orario,loc_rep) VALUES(8,3.5,'Napoli');
INSERT INTO tavolo(numero_tavolo,conto_orario,loc_rep) VALUES(9,3.5,'Napoli');
INSERT INTO tavolo(numero_tavolo,conto_orario,loc_rep) VALUES(10,2.5,'Napoli');

--AFFITTA TAVOLI
INSERT INTO
affitta_tavolo(numero_tav,cf_cliente_reg,loc_rep,flag_rep,ore_affitto,data_ora_i
nizio) VALUES (1,'VLNFTN94T55A783G','Napoli', 'L', 3, to_date('01/11/2019
15:00','dd/mm/yyyy hh24:mi'));
INSERT INTO
affitta_tavolo(numero_tav,cf_cliente_reg,loc_rep,flag_rep,ore_affitto,data_ora_i
nizio) VALUES (2,'PRTVCN89S13F839N','Napoli', 'L', 4, to_date('12/11/2019
16:00','dd/mm/yyyy hh24:mi'));
INSERT INTO
affitta_tavolo(numero_tav,cf_cliente_reg,loc_rep,flag_rep,ore_affitto,data_ora_i
nizio) VALUES (3,'CRTRND86R18F839V','Napoli', 'L', 2, to_date('18/11/2019
18:00','dd/mm/yyyy hh24:mi'));
INSERT INTO
affitta_tavolo(numero_tav,cf_cliente_reg,loc_rep,flag_rep,ore_affitto,data_ora_i
nizio) VALUES (5,'SRNNTT90P56F839F','Napoli', 'L', 1, to_date('12/10/2019
20:00','dd/mm/yyyy hh24:mi'));
INSERT INTO
affitta_tavolo(numero_tav,cf_cliente_reg,loc_rep,flag_rep,ore_affitto,data_ora_i
nizio) VALUES (6,'VLNFTN94T55A783G','Napoli', 'L', 3, to_date('06/10/2019
16:00','dd/mm/yyyy hh24:mi'));
INSERT INTO
affitta_tavolo(numero_tav,cf_cliente_reg,loc_rep,flag_rep,ore_affitto,data_ora_i
nizio) VALUES (10,'SRNNTT90P56F839F','Napoli', 'L', 4, to_date('22/10/2019
15:30','dd/mm/yyyy hh24:mi'));

--POSTAZIONE
INSERT INTO
postazione(numero_postazione,conto_orario,nome_console,loc_rep,flag_rep)
VALUES(1,3.5,'SNES Classic Mini','Napoli','VI');
INSERT INTO
postazione(numero_postazione,conto_orario,nome_console,loc_rep,flag_rep)
VALUES(2,3.5,'PlayStation Classic','Napoli','VI');
INSERT INTO
postazione(numero_postazione,conto_orario,nome_console,loc_rep,flag_rep)
VALUES(3,3.5,'Xbox 360','Napoli','VI');
INSERT INTO
postazione(numero_postazione,conto_orario,nome_console,loc_rep,flag_rep)
VALUES(4,3.5,'PlayStation 3','Napoli','VI');
```

```
INSERT INTO
postazione(numero_postazione,conto_orario,nome_console,loc_rep,flag_rep)
VALUES(5,3.5,'Wii U','Napoli','VI');
INSERT INTO
postazione(numero_postazione,conto_orario,nome_console,loc_rep,flag_rep)
VALUES(6,3.5,'Xbox One','Napoli','VI');
INSERT INTO
postazione(numero_postazione,conto_orario,nome_console,loc_rep,flag_rep)
VALUES(7,4.5,'PlayStation 4','Napoli','VI');
INSERT INTO
postazione(numero_postazione,conto_orario,nome_console,loc_rep,flag_rep)
VALUES(8,4.5,'Nintendo Switch','Napoli','VI');
INSERT INTO
postazione(numero_postazione,conto_orario,nome_console,loc_rep,flag_rep)
VALUES(9,4.5,'PC','Napoli','VI');
INSERT INTO
postazione(numero_postazione,conto_orario,nome_console,loc_rep,flag_rep)
VALUES(10,3.5,'Sega Megadrive Mini','Napoli','VI');

--AFFITTA POSTAZIONE
INSERT INTO
affitta_postazione(numero_pos,cf_cliente_reg,loc_rep,flag_rep,ore_affitto,data_o
ra_inizio) VALUES (1 ,'VLNFTN94T55A783G','Napoli', 'VI', 3, to_date('02/11/2019
14:00','dd/mm/yyyy hh24:mi'));
INSERT INTO
affitta_postazione(numero_pos,cf_cliente_reg,loc_rep,flag_rep,ore_affitto,data_o
ra_inizio) VALUES (2 ,'PRTVCN89S13F839N','Napoli', 'VI', 4, to_date('13/11/2019
15:00','dd/mm/yyyy hh24:mi'));
INSERT INTO
affitta_postazione(numero_pos,cf_cliente_reg,loc_rep,flag_rep,ore_affitto,data_o
ra_inizio) VALUES (3 ,'CRTRND86R18F839V','Napoli', 'VI', 2, to_date('19/11/2019
17:00','dd/mm/yyyy hh24:mi'));
INSERT INTO
affitta_postazione(numero_pos,cf_cliente_reg,loc_rep,flag_rep,ore_affitto,data_o
ra_inizio) VALUES (5 ,'SRNNTT90P56F839F','Napoli', 'VI', 1, to_date('13/10/2019
19:00','dd/mm/yyyy hh24:mi'));
INSERT INTO
affitta_postazione(numero_pos,cf_cliente_reg,loc_rep,flag_rep,ore_affitto,data_o
ra_inizio) VALUES (6 ,'VLNFTN94T55A783G','Napoli', 'VI', 3, to_date('07/10/2019
15:00','dd/mm/yyyy hh24:mi'));
INSERT INTO
affitta_postazione(numero_pos,cf_cliente_reg,loc_rep,flag_rep,ore_affitto,data_o
ra_inizio) VALUES (10,'SRNNTT90P56F839F','Napoli', 'VI', 4, to_date('21/10/2019
14:30','dd/mm/yyyy hh24:mi'));

--VIDEOGIOCHI_REP
INSERT INTO videogiochi_rep(loc_rep_video,nome) VALUES ('Napoli','League of
Legend');
INSERT INTO videogiochi_rep(loc_rep_video,nome) VALUES ('Napoli','DOTA');
INSERT INTO videogiochi_rep(loc_rep_video,nome) VALUES ('Napoli','League of
Runeterra');
INSERT INTO videogiochi_rep(loc_rep_video,nome) VALUES ('Napoli','Pubg');
INSERT INTO videogiochi_rep(loc_rep_video,nome) VALUES ('Napoli','Overwatch');
INSERT INTO videogiochi_rep(loc_rep_video,nome) VALUES ('Napoli','League of
Runeterra');
INSERT INTO videogiochi_rep(loc_rep_video,nome) VALUES ('Napoli','Team Fight
Tattics');

--GIOCHI_TAVOLO_REP
INSERT INTO giochi_tavolo_rep(loc_rep_tav,nome) VALUES ('Napoli','Alpha');
INSERT INTO giochi_tavolo_rep(loc_rep_tav,nome) VALUES ('Napoli','Zamit');
INSERT INTO giochi_tavolo_rep(loc_rep_tav,nome) VALUES ('Napoli','Span');
INSERT INTO giochi_tavolo_rep(loc_rep_tav,nome) VALUES ('Napoli','Overhold');
```

```
INSERT INTO giochi_tavolo_rep(loc_rep_tav,nome) VALUES ('Napoli','Zontrax');
INSERT INTO giochi_tavolo_rep(loc_rep_tav,nome) VALUES ('Napoli','Matsoft');
INSERT INTO giochi_tavolo_rep(loc_rep_tav,nome) VALUES ('Napoli','Flowdesk');
INSERT INTO giochi_tavolo_rep(loc_rep_tav,nome) VALUES ('Napoli','Konklux');

--%%%%%           LAVORATORE
INSERT INTO       Lavoratore(cf, data_di_nascita, nome, cognome, secondo_nome,
citta_residenza,data_assunzione)
      VALUES('EUILMF99Y494JIK9',to_date('12/07/1994','dd/mm/yyyy'),'Claudio','In
izio',NULL,'Napoli',to_date('1/11/2005','dd/mm/yyyy'));
INSERT INTO       Lavoratore(cf, data_di_nascita, nome, cognome, secondo_nome,
citta_residenza,data_assunzione)
      VALUES('2938DHU3D37HDUH3',to_date('26/03/1996','dd/mm/yyyy'),'Fiuso','Fina
le',NULL,'Milano',to_date('13/12/2005','dd/mm/yyyy'));
INSERT INTO       Lavoratore(cf, data_di_nascita, nome, cognome, secondo_nome,
citta_residenza,data_assunzione)
      VALUES('WOECW8W3HDEDDHUH',to_date('29/11/1992','dd/mm/yyyy'),'Salvo','Rosa
',NULL,'Napoli',to_date('02/9/2005','dd/mm/yyyy'));

--%%%%%           IMPIEGATO
INSERT INTO       Impiegato(cf_imp,reparto_loc_imp, flag_rep_imp)
VALUES('EUILMF99Y494JIK9','Salerno','VI');

--%%%%%           Direttore
INSERT INTO       Direttore(cf_dir,reparto_loc_dir, flag_rep_dir)
      VALUES('WOECW8W3HDEDDHUH','Napoli','VE');

--%%%%% Amministratore
INSERT INTO       Amministratore(cf_amm,loc_filiale)
      VALUES('2938DHU3D37HDUH3','Casalnuovo di Napoli');

--%%%%%     Stipendi
INSERT INTO Stipendio(numero_bollo,denaro,data_erogazione,cf_lav)
      VALUES('adts873egfe78d','1306',to_date('03/11/2019','dd/mm/yyyy'),'EUILMF9
9Y494JIK9');
INSERT INTO Stipendio(numero_bollo,denaro,data_erogazione,cf_lav)
      VALUES('mcnbue2o382hdn','2609',to_date('03/12/2019','dd/mm/yyyy'),'EUILMF9
9Y494JIK9');
INSERT INTO Stipendio(numero_bollo,denaro,data_erogazione,cf_lav)
      VALUES('yw0wdm0hzlegsy','1700',to_date('03/11/2019','dd/mm/yyyy'),'WOECW8W
3HDEDDHUH');
INSERT INTO Stipendio(numero_bollo,denaro,data_erogazione,cf_lav)
      VALUES('5en9yesi2i5m86','2400',to_date('03/12/2019','dd/mm/yyyy'),'WOECW8W
3HDEDDHUH');
INSERT INTO Stipendio(numero_bollo,denaro,data_erogazione,cf_lav)
      VALUES('sa4vcvs2bz6z0m','2200',to_date('03/12/2019','dd/mm/yyyy'),'2938DHU
3D37HDUH3');
INSERT INTO Stipendio(numero_bollo,denaro,data_erogazione,cf_lav)
      VALUES('vlxs5lyv2lic89','1400',to_date('03/11/2019','dd/mm/yyyy'),'2938DHU
3D37HDUH3');

--%%%%%           CASA PRODUTTRICE
INSERT INTO       casaproduttrice(partita_iva,nome,nazionalita)
      VALUES('26095833181','Mysidia','Danimarca');
INSERT INTO       casaproduttrice(partita_iva,nome,nazionalita)
      VALUES('16295833231','Oerba','Cina');
INSERT INTO       casaproduttrice(partita_iva,nome,nazionalita)
      VALUES('15495833151','Enlight','Cina');


--%%%%% PRODOTTO
```

```
 INSERT INTO
      prodotto(codice_prodotto,anno_produzione,prezzo,nome,partita_iva_casaprod)
            VALUES('gz2w8fi4wjhx3nar',
      to_date('17/06/2019','dd/mm/yyyy'),75,'Playstation
Classic','11372837411');
 INSERT INTO
      prodotto(codice_prodotto,anno_produzione,prezzo,nome,partita_iva_casaprod)
            VALUES('6il6gl938okpjxeo',
      to_date('17/06/2019','dd/mm/yyyy'),150,'Playstation 3','11372837411');
 INSERT INTO
      prodotto(codice_prodotto,anno_produzione,prezzo,nome,partita_iva_casaprod)
            VALUES('renvwiush87dpw26',
      to_date('17/06/2019','dd/mm/yyyy'),350,'Playstation 4','11372837411');

 --%%%%% GADGET
 INSERT INTO           gadget(codice_gadget,tipo)
      VALUES('123456789123456r','Figure');
 INSERT INTO           gadget(codice_gadget,tipo)
      VALUES('123456789123456s','Figure');
 INSERT INTO           gadget(codice_gadget,tipo)
      VALUES('123456789123456t','Tazza');

 --%%%%% PERIFERICA
 INSERT INTO           periferica(codice_periferica)
      VALUES('ajhe567891234e67');
 INSERT INTO           periferica(codice_periferica)
      VALUES('ajhe567891234d67');
 INSERT INTO           periferica(codice_periferica)
      VALUES('123ak67891234c67');

  --%%%%%   CONSOLE
 INSERT INTO           console(codice_console)      VALUES('gz2w8fi4wjhx3nar');
 INSERT INTO           console(codice_console)      VALUES('6il6gl938okpjxeo');
 INSERT INTO           console(codice_console)      VALUES('renvwiush87dpw26');

 --%%%%% GIOCO DA TAVOLO
 INSERT INTO           gioco_tavolo(codice_gioco_tavolo,pegi)
      VALUES('otjk1lfwqxpqvgta',4);
 INSERT INTO           gioco_tavolo(codice_gioco_tavolo,pegi)
      VALUES('9lps4u0xljlvdkyw',4);
 INSERT INTO           gioco_tavolo(codice_gioco_tavolo,pegi)
      VALUES('jxgix5jcnefzfkdj',4);

--%%%%% VIDEOGIOCO
 INSERT INTO           videogioco(codice_videogioco,pegi)
      VALUES('utec8l2ez39z6t1z',13);
 INSERT INTO           videogioco(codice_videogioco,pegi)
      VALUES('huwznmbcwsp76jk0',13);
 INSERT INTO           videogioco(codice_videogioco,pegi)
      VALUES('h0fike4ua40o03te',13);

 --%%%%% GENERE
 INSERT INTO           genere(codice_prodotto,tipo)
      VALUES('utec8l2ez39z6t1z','Picchiaduro');
```

```
 INSERT INTO           genere(codice_prodotto,tipo)
      VALUES('huwznmbcwsp76jk0','Sparatutto');
 INSERT INTO           genere(codice_prodotto,tipo)
      VALUES('h0fike4ua40o03te','Sparatutto');
 INSERT INTO           genere(codice_prodotto,tipo)
      VALUES('rozywndt91wakcfn','Battle royale');
 INSERT INTO           genere(codice_prodotto,tipo)
      VALUES('98sbhuws45tv8t49','Sport');
 INSERT INTO           genere(codice_prodotto,tipo)
      VALUES('dmrbgosmzmaa5hqx','Simulatore');


  --%%%%% VENDE
INSERT INTO       vende(loc_rep,cod_prod,cf_cli,data_e_ora,quantita)
      VALUES('Napoli','renvwiush87dpw26','MRTCLD94L12F839L',to_date('11/10/2019
14:00','dd/mm/yyyy hh24:mi'),1);
INSERT INTO       vende(loc_rep,cod_prod,cf_cli,data_e_ora,quantita)
      VALUES('Napoli','renvwiush87dpw26','LMRGNM92E20A783Z',to_date('21/10/2019
15:00','dd/mm/yyyy hh24:mi'),1);
INSERT INTO       vende(loc_rep,cod_prod,cf_cli,data_e_ora,quantita)
      VALUES('Napoli','renvwiush87dpw26','VREDMN90B10A783V',to_date('12/10/2019
17:00','dd/mm/yyyy hh24:mi'),1);
INSERT INTO       vende(loc_rep,cod_prod,cf_cli,data_e_ora,quantita)
      VALUES('Napoli','ajhe567891234n67','DFRSTR99B52A509N',to_date('18/10/2019
19:00','dd/mm/yyyy hh24:mi'),1);
INSERT INTO       vende(loc_rep,cod_prod,cf_cli,data_e_ora,quantita)
      VALUES('Napoli','ajhe567891234m67','DMAGCM94E30A783J',to_date('01/11/2019
15:00','dd/mm/yyyy hh24:mi'),1);
INSERT INTO       vende(loc_rep,cod_prod,cf_cli,data_e_ora,quantita)
      VALUES('Napoli','ajhe567891234l67','VLNFTN94T55A783G',to_date('09/11/2019
14:30','dd/mm/yyyy hh24:mi'),1);
INSERT INTO       vende(loc_rep,cod_prod,cf_cli,data_e_ora,quantita)
      VALUES('Napoli','ajhe567891234i67','PRTVCN89S13F839N',to_date('25/11/2019
17:00','dd/mm/yyyy hh24:mi'),1);

--%%%%% EVENTI
INSERT INTO       evento(localita_evento,data_inizio,data_fine)
      VALUES('Napoli',
      to_date('12/10/2019','dd/mm/yyyy'),to_date('19/10/2019','dd/mm/yyyy'));
INSERT INTO       evento(localita_evento,data_inizio,data_fine)
      VALUES('Napoli',
      to_date('12/11/2019','dd/mm/yyyy'),to_date('19/11/2019','dd/mm/yyyy'));

--%%%%%           TORNEO
INSERT INTO
      torneo(data_inizio_torneo,localita_torneo,quota,premio,nome_gioco,piattafo
rma,max_partecipanti)
      VALUES(to_date('12/11/2019','dd/mm/yyyy'),'Napoli',
      '20','100','Tekken7','PlayStation 4',20);


--%%%%%           DISTRIBUZIONE
INSERT INTO distribuzione(localita_distribuzione,data_inizio_distribuzione)
      VALUES('Napoli',                    to_date('12/10/2019','dd/mm/yyyy'));
```

```
--%%%%%           PARTECIPA
INSERT INTO partecipa(loc_eve,cf_cliente_reg,data_inizio)  VALUES('Napoli',
           'VLNFTN94T55A783G',to_date('12/11/2019','dd/mm/yyyy'));
INSERT INTO partecipa(loc_eve,cf_cliente_reg,data_inizio)  VALUES('Napoli',
           'PRTVCN89S13F839N',to_date('12/11/2019','dd/mm/yyyy'));
INSERT INTO partecipa(loc_eve,cf_cliente_reg,data_inizio)  VALUES('Napoli',
           'CRTRND86R18F839V',to_date('12/11/2019','dd/mm/yyyy'));

--%%%%%           GADGET_DIST
INSERT INTO       gadget_dist(loc_dist,codice_gad,data_distribuzione,quantita)
          VALUES('Napoli',
     '123456789123456r',to_date('12/10/2019','dd/mm/yyyy'),'3');

--%%%%%           OFFERTA
INSERT INTO       offerta(codice_offerta,data_inizio,data_fine)        VALUES(1
,to_date('01/10/2019','dd/mm/yyyy'),to_date('01/11/2020','dd/mm/yyyy'));
INSERT INTO       offerta(codice_offerta,data_inizio,data_fine)        VALUES(2
,to_date('12/10/2019','dd/mm/yyyy'),to_date('12/11/2020','dd/mm/yyyy'));
INSERT INTO       offerta(codice_offerta,data_inizio,data_fine)        VALUES(3
,to_date('14/10/2019','dd/mm/yyyy'),to_date('14/11/2020','dd/mm/yyyy'));
INSERT INTO       offerta(codice_offerta,data_inizio,data_fine)        VALUES(4
,to_date('11/10/2019','dd/mm/yyyy'),to_date('11/11/2020','dd/mm/yyyy'));
INSERT INTO       offerta(codice_offerta,data_inizio,data_fine)        VALUES(5
,to_date('15/10/2019','dd/mm/yyyy'),to_date('15/11/2020','dd/mm/yyyy'));


--%%%%%           OTTIENE
INSERT INTO ottiene(cod_off,cf_cliente_reg)          VALUES(1
,'MRTCLD94L12F839L');
INSERT INTO ottiene(cod_off,cf_cliente_reg)          VALUES(2
,'LMRGNM92E20A783Z');
INSERT INTO ottiene(cod_off,cf_cliente_reg)          VALUES(3
,'VREDMN90B10A783V');
INSERT INTO ottiene(cod_off,cf_cliente_reg)          VALUES(4
,'DFRSTR99B52A509N');
INSERT INTO ottiene(cod_off,cf_cliente_reg)          VALUES(5
,'DMAGCM94E30A783J');
INSERT INTO ottiene(cod_off,cf_cliente_reg)          VALUES(6
,'VLNFTN94T55A783G');

--%%%%%           SOGGETTO
INSERT INTO soggetto(cod_off,cod_prod)          VALUES(1 ,'gz2w8fi4wjhx3nar');
INSERT INTO soggetto(cod_off,cod_prod)          VALUES(2 ,'6il6gl938okpjxeo');
INSERT INTO soggetto(cod_off,cod_prod)          VALUES(3 ,'renvwiush87dpw26');
INSERT INTO soggetto(cod_off,cod_prod)          VALUES(4 ,'r48ymft9ua96aagk');
INSERT INTO soggetto(cod_off,cod_prod)          VALUES(5 ,'8y2d33sjsmd4jrqw');
INSERT INTO soggetto(cod_off,cod_prod)          VALUES(6 ,'uu3g5qehpdzmlcxl');

--%%%%%           CREA
INSERT INTO       crea(cod_off,cf_org)          VALUES(1 ,'2938DHU3D37HDUH3');
INSERT INTO       crea(cod_off,cf_org)          VALUES(2 ,'DHU2I3DH2U3IDHDU');
INSERT INTO       crea(cod_off,cf_org)          VALUES(3 ,'DHU32I3DH23UDHHD');
```

```
--%%%%%           ORGANIZZA
INSERT INTO organizza(loc_eve,data_inizio_eve,cf_org)      VALUES('Napoli',
                to_date('12/10/2019','dd/mm/yyyy'),'ISERVE99S70F8396');
INSERT INTO organizza(loc_eve,data_inizio_eve,cf_org)      VALUES('Napoli',
                to_date('12/11/2019','dd/mm/yyyy'),'KI214CNJEB3FJ4F6');
```

Trigger

I trigger DML sono utili principalmente per il controllo di vincoli dinamici in fase di immissione
o aggiornamento dei dati.
E'possibile usarli anche per implementare *regole di business* o per calcolare, generare o sovrascrivere il valore di alcuni campi.

SCADENZA TESSERA

Il trigger scadenza_tessera controlla che la data di scadenza immessa per la creazione di una tessera sia successiva di almeno un anno dalla data di registrazione dell'utente, se così non fosse il trigger genererebbe un'application error.

Implementazione

```
create or replace trigger scadenza_tessera
after insert or update of data_scadenza on tessera
for each row
declare
        dataReg                 cliente_reg.data_registrazione%TYPE;
begin
        select data_registrazione into dataReg
                from cliente_reg
                where cf_cliente = :new.cf_cli_reg;
        if(:new.data_scadenza < ADD_MONTHS(dataReg,12)) then
                raise_application_error(-20200,'data scadenza non successiva di almeno un
anno dalla registrazione.');
        end if;
end;
```

SCADENZA OFFERTA

Il trigger scadenza_offerta controlla che alla creazione di un'offerta la data di scadenza sia successiva alla data di inizio, se così non fosse il trigger genererebbe un'application error.

Implementazione

```
create or replace trigger scadenza_offerta
after insert or update of data_fine on offerta
for each row
declare

begin
        if( :new.data_fine < :new.data_inizio ) then
                raise_application_error(-20210,'data scadenza non successiva alla data inizio.');
        elsif( :new.data_fine < :new.data_inizio + 7) then
                raise_application_error(-20211,'data scadenza non successiva di almeno una
settimana dalla data inizio.');
        end if;
end;
```

TAVOLO OCCUPATO

Il trigger tavolo_occupato controlla che all'affitto di un tavolo non sia occupato già da qualcun altro o che l'orario di affitto non vada oltre la chiusura, se così non fosse il trigger genererebbe un'application error.

Implementazione

```
create or replace trigger tavolo_occupato
before insert on affitta_tavolo
for each row
declare
        cf              cliente.cf%TYPE;
        ora_inizio decimal(4,2) :=
to_number(replace(to_char(:new.data_ora_inizio,'hh24:mi'),':','.'),'99.99');
        cursor occupato IS
                select cf_cliente_reg
                from affitta_tavolo
                where loc_rep = :new.loc_rep and numero_tav = :new.numero_tav
                                and
                                ((data_ora_inizio between :new.data_ora_inizio and
:new.data_ora_inizio + :new.ore_affitto)
                                or
                                (:new.data_ora_inizio between data_ora_inizio and data_ora_inizio
+ ore_affitto));
begin
        if ( ora_inizio < 9.00 or ora_inizio > 21.00 or (ora_inizio + :new.ore_affitto) > 21.00) then
                raise_application_error(-20220,'A quell'orario siamo chiusi');
        end if;
        open occupato;
        fetch occupato into cf;
        if occupato%FOUND then
                raise_application_error(-20221,'tavolo numero:'|| :new.numero_tav ||' non
disponibile per la prenotazione');
        end if;
        close occupato;
end;
```

POSTAZIONE OCCUPATA

Il trigger postazione_occupata controlla che all'affitto di una postazione non sia occupata già da qualcun altro o che l'orario di affitto non vada oltre la chiusura, se così non fosse il trigger genererebbe un'application error.

Implementazione

```
create or replace trigger postazione_occupata
before insert on affitta_postazione
for each row
declare
        cf              cliente.cf%TYPE;
        ora_inizio decimal(4,2) :=
to_number(replace(to_char(:new.data_ora_inizio,'hh24:mi'),':','.'),'99.99');
```

```
        cursor occupato IS
                select cf_cliente_reg
                from affitta_postazione
                where loc_rep = :new.loc_rep and numero_pos = :new.numero_pos
                                and
                                ((data_ora_inizio between :new.data_ora_inizio and
:new.data_ora_inizio + :new.ore_affitto)
                                or
                                (:new.data_ora_inizio between data_ora_inizio and data_ora_inizio
+ ore_affitto));
begin
        if ( ora_inizio < 9.00 or ora_inizio > 21.00 or (ora_inizio + :new.ore_affitto) > 21.00) then
                raise_application_error(-20230,'A quell'orario siamo chiusi');
        end if;
        open occupato;
        fetch occupato into cf;
        if occupato%FOUND then
                raise_application_error(-20231,'tavolo numero:'|| :new.numero_pos ||' non
disponibile per la prenotazione');
        end if;
        close occupato;
end;
```

CONTROLLO PARTECIPANTI

Il trigger controllo_partecipanti controlla che alla registrazione di un cliente registrato ad un torneo ci sia almeno un posto libero, se così non fosse il trigger genererebbe un'application error.

Implementazione

```
create or replace trigger controllo_partecipanti
before insert on partecipa
for each row
declare
        counter         int;
        max_part int;
begin
        select count(*) into counter
                from partecipa
                where data_inizio = :new.data_inizio;

        select max_partecipanti into max_part
                from torneo
                where data_inizio_torneo = :new.data_inizio;
        if ( counter = max_part ) then
                raise_application_error(-20240,'raggiunto numero massimo iscrizioni al torneo.');
        end if;
end;
```

DIREZIONE APPROVVIGIONAMENTO

Il trigger direzione_approv controlla che all'inserimento di un approvvigionamento, quest'ultimo non sia assegnato ad una filiale che non è provvista di reparto vendita, se così non fosse il trigger genererebbe un'application error.

Implementazione

```
create or replace trigger direzione_approv
after insert or update of loc_filiale on approvviggionamento
for each row
declare
        cursor reparto_vendita is
                select *
                from reparto
                where flag_reparto = 'VE' and loc_reparto = :new.loc_filiale;
        rep_vendita             reparto%ROWTYPE;
begin
        open reparto_vendita;
        fetch reparto_vendita into rep_vendita;
        if( reparto_vendita%NOTFOUND ) then
                raise_application_error(-20250,'Reparto vendita non presente in questa filiale.
Impossibile mandare approvvigionamento.');
        end if;
        close reparto_vendita;
end;
```

ORGANIZZAZIONE TORNEI

Il trigger organizzazione_tornei controlla che alla creazione di un torneo, la data di svolgimento di quest'ultimo non sovrapponga nessun altro torneo, se così non fosse il trigger genererebbe un'application error.

Implementazione

```
create or replace trigger organizzazione_tornei
before insert or update of data_inizio_torneo on torneo
for each row
declare
        loc_tor VARCHAR(25);
        data_inizio             DATE;
        data_fine       DATE;
        data_fine_torneo        evento.data_fine%TYPE;
        cursor eventi is
                select data_inizio_torneo,data_fine, localita_torneo
                from torneo join evento on data_inizio=data_inizio_torneo
                where nome_gioco = :new.nome_gioco and piattaforma = :new.piattaforma;

begin
        select data_fine into data_fine_torneo
                from evento
                where data_inizio = :new.data_inizio_torneo and localita_evento =
:new.localita_torneo;
        open eventi;
        loop
```

```
            fetch eventi into data_inizio,data_fine,loc_tor;
            exit when eventi%NOTFOUND;
            if ( loc_tor = :new.localita_torneo and ((data_inizio > :new.data_inizio_torneo and
data_inizio < data_fine_torneo) or (:new.data_inizio_torneo > data_inizio and
:new.data_inizio_torneo < data_fine))) then
                  raise_application_error(-20260,'è già in corso un torneo riguardante ' ||
:new.nome_gioco || ' su ' || :new.piattaforma );
                  exit;
            end if;
      end loop;
      close eventi;
end;
```

ORGANIZZAZIONE GADGET

Il trigger organizzazione_gadget controlla che alla creazione di una nuova distribuzione, la data di quest'ultimo non si sovrapponga ad altre distribuzioni, se così non fosse il trigger genererebbe un'application error

Implementazione

```
create or replace trigger organizzazione_gadget
before insert or update of data_inizio_distribuzione on distribuzione
for each row
declare

      loc_dist VARCHAR(25);
      data_inizio          DATE;
      data_fine      DATE;
      data_fine_distribuzione       evento.data_fine%TYPE;
      codice_gadget VARCHAR(16);
      cursor eventi is
            select data_inizio_distribuzione,data_fine,codice_gad, localita_distribuzione
            from (distribuzione join evento on data_inizio=data_inizio_distribuzione) join
gadget_dist on data_distribuzione=data_inizio_distribuzione
            where codice_gad = (select codice_gad from gadget_dist where
data_distribuzione = :new.data_inizio_distribuzione);

begin
      select data_fine into data_fine_distribuzione
            from evento
            where data_inizio = :new.data_inizio_distribuzione and localita_evento =
:new.localita_distribuzione;
      open eventi;
      loop
            fetch eventi into data_inizio, data_fine, codice_gadget,loc_dist;
            exit when eventi%NOTFOUND;
            if ( loc_dist = :new.localita_distribuzione and (data_inizio >
:new.data_inizio_distribuzione and data_inizio < data_fine_distribuzione) or
(:new.data_inizio_distribuzione > data_inizio and :new.data_inizio_distribuzione < data_fine))
then
                  raise_application_error(-20270,'è già in corso una distribuzione
riguardante il gadget con cod: ' || codice_gadget );
                  exit;
            end if;
```

```
        end loop;
        close eventi;
end;
```

SET GENERAZIONE

Il trigger set_generazione genera automaticamente il numero di generazione di una console al momento dell'inserimento.

Implementazione

```
CREATE OR REPLACE TRIGGER set_generazione
before INSERT ON console
FOR EACH ROW
DECLARE
        nome_console prodotto.nome%TYPE;
BEGIN
        select lower(nome) into nome_console
                from prodotto
                where codice_prodotto = :new.codice_console;
        if nome_console in ('nintendo ds','nintendo ds lite','nintendo dsi','nintendo dsi
xl','playstation portable','xbox 360','nintendo wii','playstation 3') then
                :new.generazione := 7;
        elsif nome_console in ('nintendo 3ds','nintendo 2ds','playstation vita','wiiu','nintendo
switch','nintendo switch lite','playstation 4', 'xbox one') then
                :new.generazione := 8;
        else
                :new.generazione := 6;
        end if;
END;
```

Procedure e funzioni

Le procedure sono la parte più strettamente legata alla *logica di business* e all'automazione della Base di Dati.
Se nel DDL e nel DML il problema principale è decidere la rappresentazione ottima dei dati e garantire rispetto dei vincoli di integrità su di essi definiti, in questa fase il problema è come sfruttare al meglio tale rappresentazione per automatizzarne la gestione e trarne il massimo profitto.

CREA DISTRIBUZIONE

Le tabelle distribuzione, evento, organizza e gadget_dist sono popolate tramite la procedura crea_distribuzione, e per eseguire le suddette azioni, tramite un cursore e il codice fiscale amministratore ricaviamo la località, essenziale per il popolamento. La suddetta procedura crea una distribuzione evento.

Implementazione

```
CREATE OR REPLACE PROCEDURE crea_distribuzione(cf_amm VARCHAR, sz_data_inizio
VARCHAR, sz_data_fine VARCHAR, sz_cod_gadget VARCHAR, iQuantita INTEGER)
AS
```

```
dt_inizio_distr DATE := to_date(sz_data_inizio,'DD/MM/YYYY');
dt_fine_distr DATE := to_date(sz_data_fine,'DD/MM/YYYY');
cursor curs_localita_eve is SELECT localita FROM AMMINISTRATORE A join FILIALE on loc_filiale =
localita WHERE A.cf_amm = cf_amm;
localita_eve VARCHAR(25);

BEGIN
        open curs_localita_eve;
        FETCH curs_localita_eve into localita_eve;
        insert into evento VALUES(localita_eve,dt_inizio_distr,dt_fine_distr);
        insert into organizza VALUES(localita_eve,dt_inizio_distr, cf_amm);
        insert into distribuzione VALUES(dt_inizio_distr,localita_eve);
        insert into gadget_dist VALUES(sz_cod_gadget,localita_eve, iQuantita, dt_inizio_distr);
        close curs_localita_eve;
        COMMIT;


EXCEPTION
WHEN  NO_DATA_FOUND    THEN
                DBMS_OUTPUT.PUT_LINE('NO_DATA_FOUND.');
END;
```

INSERISCI TORNEO

Le tabelle torneo, evento e organizza sono popolate dalla procedura inserisci_torneo, e per eseguire le suddette azioni, tramite un cursore e il codice fiscale amministratore ricaviamo la località, essenziale per il popolamento. La suddetta procedura crea un torneo evento.

Implementazione

```
CREATE OR REPLACE PROCEDURE inserisci_torneo(cf_amm VARCHAR, sz_data_inizio VARCHAR,
sz_data_fine VARCHAR, fQuota FLOAT, iPremio INTEGER, szNomeGioco VARCHAR,szPiattaforma
VARCHAR, iMaxPartecipanti INTEGER)
AS

dt_inizio_torneo DATE := to_date(sz_data_inizio,'DD/MM/YYYY');
dt_fine_torneo DATE := to_date(sz_data_fine,'DD/MM/YYYY');
cursor curs_localita_eve is SELECT localita FROM AMMINISTRATORE A join FILIALE on loc_filiale =
localita WHERE A.cf_amm = cf_amm;
localita_eve VARCHAR(25);


BEGIN
        open curs_localita_eve;
        FETCH curs_localita_eve into localita_eve;
        insert into evento VALUES(localita_eve,dt_inizio_torneo,dt_fine_torneo);
        insert into organizza VALUES(localita_eve,dt_inizio_torneo, cf_amm);
        insert into torneo
VALUES(dt_inizio_torneo,localita_eve,fQuota,iPremio,szNomeGioco,szPiattaforma,iMaxPartecip
anti);
        close curs_localita_eve;
        COMMIT;
```

```
EXCEPTION
WHEN  NO_DATA_FOUND    THEN
                DBMS_OUTPUT.PUT_LINE('NO_DATA_FOUND.');
END;
```

ISCRIZIONE TORNEO

Le tabella partecipa è popolata dalla procedura iscrizione_torneo, la quale ha anche bisogno di controllare che la tessera non sia scaduta o il periodo d'iscrizione al torneo non sia terminato. La suddetta procedura serve ad iscrivere un cliente registrato ad un torneo.

Implementazione

```
CREATE OR REPLACE PROCEDURE iscrizione_torneo(cf VARCHAR,localita VARCHAR,str_data
VARCHAR)
IS
        scadenza date;
        dt_inizio date := to_date(str_data,'DD/MM/YYYY');
        is_torneo  date;
        tessera_scaduta                 EXCEPTION;
        troppo_tardi                    EXCEPTION;
BEGIN
        select data_scadenza into scadenza
        from tessera
        where cf_cli_reg=cf;

        if scadenza <= SYSDATE then
                raise tessera_scaduta;
        end if;

        if dt_inizio <= SYSDATE then
                raise troppo_tardi;
        end if;

        select data_inizio_torneo into is_torneo
        from torneo
        where data_inizio_torneo=dt_inizio and localita_torneo = localita;
        insert into partecipa values(localita,cf,dt_inizio);
        COMMIT;



EXCEPTION
WHEN  NO_DATA_FOUND    THEN
                DBMS_OUTPUT.PUT_LINE('NO_DATA_FOUND.');
WHEN  tessera_scaduta                   THEN
                DBMS_OUTPUT.PUT_LINE('Tessera scaduta il '|| scadenza ||'. Non e''possibile
effetuare l''iscrizione al torneo');
WHEN  troppo_tardi              THEN
                DBMS_OUTPUT.PUT_LINE('Periodo d''iscrizione terminato.');
END;
```

LETTURA ENTRATE

La procedura lettura_entrate serve a stampare a video, tutte le entrate di una determinata filiale in un determinato mese di un determinato anno.
Esse sono composte da ricavi dai prodotti venduti, dalle quote di partecipazione ai tornei, e dal ricavato dell'affitto dei tavoli e delle postazioni gaming.

Implementazione

```
CREATE OR REPLACE PROCEDURE lettura_entrate(szFiliale VARCHAR, szMese VARCHAR, szAnno VARCHAR)
AS

dtUserDate VARCHAR(10);
fTotSomme FLOAT;
fSommaProdottiVenduti prodotto.prezzo%TYPE;
fSommaQuoteTorneo torneo.quota%TYPE;
fSommaRicaviAffittoTavolo tavolo.conto_orario%type;
fSommaRicaviAffittoPostazione postazione.conto_orario%type;


BEGIN
	dtUserDate := szMese||'-'||szAnno;

	SELECT sum(prezzo * quantita) into fSommaProdottiVenduti from vende join prodotto on cod_prod = codice_prodotto where to_char(data_e_ora,'MM-YYYY') = dtUserDate and loc_rep = szFiliale;
	if (fSommaProdottiVenduti is null) then fSommaProdottiVenduti := 0; end if;

	SELECT count(*) * quota into fSommaQuoteTorneo from torneo join partecipa on data_inizio = data_inizio_torneo where to_char(data_inizio_torneo,'MM-YYYY') = dtUserDate and localita_torneo = szFiliale  group by quota,localita_torneo;
	if (fSommaQuoteTorneo is null) then fSommaQuoteTorneo := 0; end if;

	SELECT sum(conto_orario * ore_affitto) into fSommaRicaviAffittoTavolo from affitta_tavolo aft join tavolo ta on aft.loc_rep = ta.loc_rep where aft.loc_rep = szFiliale and aft.numero_tav = ta.numero_tavolo and to_char(data_ora_inizio, 'MM-YYYY') = dtUserDate;
	if (fSommaRicaviAffittoTavolo is null) then fSommaRicaviAffittoTavolo := 0; end if;


	SELECT sum(conto_orario * ore_affitto) into fSommaRicaviAffittoPostazione from affitta_postazione afp join postazione pos on afp.loc_rep = pos.loc_rep where afp.loc_rep = szFiliale and afp.numero_pos = pos.numero_postazione and to_char(data_ora_inizio, 'MM-YYYY') = dtUserDate;
	if (fSommaRicaviAffittoPostazione is null) then fSommaRicaviAffittoPostazione := 0; end if;

	fTotSomme := (fSommaProdottiVenduti+fSommaQuoteTorneo+fSommaRicaviAffittoTavolo+fSommaRicaviAffittoPostazione);

	DBMS_OUTPUT.PUT_LINE('La somma delle entrate del '|| dtUserDate ||' e di '|| TO_CHAR(fTotSomme)||' euro.');

END;
```

LETTURA USCITE

La procedura lettura_uscite serve a stampare a video, tutte le uscite di una determinata filiale in un determinato mese di un determinato anno.
Esse sono composte dagli stipendi ai dipendenti, dal costo della fornitura diviso il numero delle filiali e dai premi in uscita di ogni torneo.

Implementazione

```
CREATE OR REPLACE PROCEDURE lettura_uscite(szFiliale VARCHAR, szMese VARCHAR, szAnno
VARCHAR)
AS

dtUserDate DATE;
fTotSomme FLOAT;
fSommaStipendi stipendio.denaro%TYPE;
fSommaCostoForniture fornitura.costo%TYPE;
fSommaPremiTorneo torneo.premio%TYPE;
iNumFiliali INTEGER;

BEGIN
        dtUserDate := to_date(szMese||'-'||szAnno,'MM/YYYY');
        SELECT sum(denaro) into fSommaStipendi from (select reparto_loc_imp, cf_imp from
impiegato union select reparto_loc_dir, cf_dir from direttore union select loc_filiale,cf_amm
from amministratore) join stipendio on cf_imp = cf_lav where
to_char(data_erogazione,'MM/YYYY') = to_char(dtUserDate,'MM/YYYY') and reparto_loc_imp =
szFiliale;
        if (fSommaStipendi is null) then fSommaStipendi := 0; end if;

        SELECT count (*) into iNumFiliali from filiale;
        SELECT sum(costo) into fSommaCostoForniture from fornitura where
(to_char(data,'MM/YYYY') = to_char(dtUserDate,'MM/YYYY'));
        if (fSommaCostoForniture is null) then fSommaCostoForniture := 0;else
fSommaCostoForniture := fSommaCostoForniture/iNumFiliali; end if;

        SELECT sum(premio) into fSommaPremiTorneo from torneo where
(to_char(data_inizio_torneo,'MM/YYYY') = to_char(dtUserDate,'MM/YYYY')) and localita_torneo
= szFiliale;
        if (fSommaPremiTorneo is null) then fSommaPremiTorneo := 0; end if;

        fTotSomme := (fSommaStipendi+fSommaCostoForniture+fSommaPremiTorneo);

        DBMS_OUTPUT.PUT_LINE('La somma delle uscite del
'||to_char(dtUserDate,'MM/YYYY')||' e di '|| TO_CHAR(fTotSomme)||' euro.');

END;
```

REGISTRA CLIENTE

La procedura registra_cliente popola le tabelle cliente_reg e tessera.
Il codice tessera viene generato con un random in base al codice fiscale del cliente. Questa procedura serve per registrare un cliente come cliente registrato.

Implementazione

```
CREATE OR REPLACE PROCEDURE registra_cliente(szCF in VARCHAR, szEmail in VARCHAR) AS

codTesseraSeed INTEGER;
szRandomString VARCHAR(10);

BEGIN

INSERT INTO    cliente_reg     VALUES(szCF,szEmail,sysdate);
SELECT ORA_HASH(LOWER(szCF),999999) into codTesseraSeed from DUAL;
DBMS_RANDOM.seed(codTesseraSeed);
SELECT dbms_random.string('x',10) into szRandomString from DUAL;
INSERT INTO tessera   VALUES(szRandomString,sysdate+366,szCF);
COMMIT;


END;
```

Data Control Language

L'utente proprietari_catena può inserire tuple nelle tabelle lavoratore, impiegato, direttore, amministratore, stipendi, fornitura, prodotti_contenuti.

```
GRANT CONNECT, CREATE SESSION                TO proprietari_catena;
GRANT INSERT        ON lavoratore            TO proprietari_catena;
GRANT INSERT        ON impiegato             TO proprietari_catena;
GRANT INSERT        ON direttore             TO proprietari_catena;
GRANT INSERT        ON amministratore        TO proprietari_catena;
GRANT INSERT        ON stipendi              TO proprietari_catena;
GRANT INSERT        ON fornitura             TO proprietari_catena;
GRANT INSERT        ON prodotti_contenuti    TO proprietari_catena;
```

L'utente impiegato può inserire tuple nelle tabelle vende, cliente , affitta tavolo, affitta postazione, fare l'update delle tessere ed eseguire le procedure registra_cliente e iscrizione_torneo.

```
GRANT CONNECT, CREATE SESSION                TO impiegato;
GRANT INSERT        ON vende                 TO impiegato;
GRANT INSERT        ON cliente               TO impiegato;
GRANT INSERT        ON affitta_tavolo        TO impiegato;
GRANT INSERT        ON affitta_postazione    TO impiegato;
GRANT UPDATE        ON tessera               TO impiegato;

GRANT EXECUTE       ON registra_cliente      TO impiegato;
GRANT EXECUTE       ON iscrizione_torneo     TO impiegato;
```

L'utente direttore può selezionare tuple nella tabella impiegato ed eseguire le procedure lettura_entrate e lettura_uscite.

```
GRANT CONNECT, CREATE SESSION        TO direttore ;
GRANT SELECT        ON impiegato     TO direttore;
GRANT EXECUTE       ON lettura_entrate  TO direttore;
GRANT EXECUTE       ON lettura_uscite   TO direttore;
```

L'utente amministratore può inserire tuple nelle tabelle offerta, approvviggionamento, consiste ed eseguire le procedure lettura_uscite, lettura_entrate, inserisci_torneo, crea_distribuzione.

```
GRANT CONNECT, CREATE SESSION                TO amministratore;
GRANT SELECT      ON impiegato               TO amministratore;
GRANT SELECT      ON direttore               TO amministratore;
GRANT INSERT      ON offerta                 TO amministratore;
GRANT INSERT      ON approvviggionamento     TO amministratore;
GRANT INSERT      ON consiste                TO amministratore;

GRANT EXECUTE     ON inserisci_torneo        TO amministratore;
GRANT EXECUTE     ON crea_distribuzione      TO amministratore;
GRANT EXECUTE     ON lettura_entrate         TO amministratore;
GRANT EXECUTE     ON lettura_uscite          TO amministratore;
```

Scheduler

Lo *scheduler* serve a programmare azioni, normalmente periodiche, che avvengono in istanti
predeterminati.
Similmente ad un trigger, vale il paradigma Evento-Condizione-Azione (ECA), ma l'evento è legato allo scorrere del tempo — quindi all'orologio di sistema — piuttosto che ad un'operazione DML.
E'possibile programmare operazioni periodiche di manutenzione, *auditing*, *backup*, pulitura o aggiornamento dei dati; sia di tipo amministrativo che da utente comune.
In questo caso si crea un *job* per implementare una politica di *roll-out* dei dati.

```
--%%%%%%% JOB CHE ELIMINA GLI EVENTI DOPO UN ANNO DAL LORO INIZIO, GLI
APPROVVIGIONAMENTI E LE FORNITURE DOPO 3 ANNI ED ELIMINA GLI STIPENDI DOPO 10 ANNI
DALLA LORO EROGAZIONE.
BEGIN
DBMS_SCHEDULER.CREATE_JOB (
  job_name          =>     'Rollout',
  job_type          =>     'PLSQL_BLOCK',
  job_action        =>     'BEGIN
                                DELETE FROM evento
                                WHERE SYSDATE > data_inizio + 365;

                                DELETE FROM fornitura
                                WHERE SYSDATE > data + (365 * 3);

                                DELETE FROM approvviggionamento
                                WHERE SYSDATE > data + (365 * 3);

                                DELETE FROM stipendio
                                WHERE SYSDATE > data_erogazione + 3650;
                           END;
                           /',
  start_date        => TO_DATE('01-GEN-2020','DD-MON-YYYY'),
  repeat_interval   => 'FREQ=DAILY',
  enabled           =>     TRUE,
  comments          =>     'Cancellazione dei dati vecchi');
END;
/
```