Gestione di un Birrificio

Relazione del progetto di Basi di Dati

Implementazione di un piccolo database incentrato sulla gestione di un birrificio.

Corso: Dipartimento di Scienze e Tecnologie

Studente proponente: Carmine Coppola

Matricola: 0124002379

Studente proponente: Lorenzo Pergamo

Matricola: 0124002278

Professore: Antonio Maratea

Data di Consegna: 12/02/2023

Indice

Capitolo 1

Progettazione

Nelle seguente relazione si riporta la documentazione relativa alla progettazione di un sistema di basi di dati riguardante la gestione di un birrificio.
I requisiti presenti sono frutto di un attenta analisi sulla gestione di un birrificio e delle modalità di tracciamento relative a tutte le materie prime acquistate, utilizzate e vendute.

1.1 Sintesi dei requisiti

Il database nasce con lo scopo di **gestire un birrificio** dedito alla **produzione** di birra e la conseguente **vendita**. Vengono riportate di seguito le fasi principali di questo database:

- Gestione ***ordine di approvvigionamento***;
- Gestione della ***produzione***;
- Gestione ***vendita***;

Per la **Gestione ordine di approvvigionamento** si intende la possibilità di avere tutte le informazioni relative alle **materie prime** utilizzate per la successiva produzione della birra. Delle Materie prime, ovvero **Malto**, **Luppolo** e **Lievito** sappiamo la **provenienza** e il **GTIN** che non è altro che un codice identificativo.

Per la **Gestione della produzione** si intendono le varie fasi della produzione dei vari lotti di birra, questo è il fulcro del database poichè tramite **l'ammostamento** e la **fermentazione** si otterrà il prodotto finale. Tutto ciò avverrà in due tipologie di **contenitori** differenti, rispettivamente **bollitore** e **fermentatore**.

Infine per la **Gestione vendita** intendiamo ciò che avviene una volta che si ottiene il prodotto finito, ovvero il lotto di **birra prodotta**. Quest'ultimo verrà poi venduto ai **PUB** di cui dobbiamo avere la **particella catastale** e il **codice della fattura**.

1.2 Glossario

Qui verranno mostrati i termini utilizzati all'interno di diagrammi o nel codice, così che sia comprensibile ciò che rappresentano.

Termine	Descrizione	Sinonimi
P.IVA	Codice che identifica la partita IVA del fornitore.	
Numero_di_Tracking	Codice alfanumerico che viene fornito per monitorare lo stato dell'ordine.	
NumeroLotto	Codice che identifica il lotto della materia prima acquistata.	N_Lotto,cod_lotto
NumLottoBirraProd	Numero identificativo di ogni birra prodotta.	
NumLottoFermentato	Numero identificativo di ogni lotto fermentato.	
DataInizioF	Indica la data dell'inizio di ogni fermentazione.	
DataFineF	Indica la data di fine di ogni fermentazione.	
QuantitaLievUsato	Indica la quantità di lievito utilizzata per ogni fermentazione.	
GTIN	Global Trade Item Number è un identificatore di prodotto univoco riconosciuto a livello internazionale.	GS1_Fornit
GS1_Fornitore	Parte identificativa del fornitore e da una parte identificativa del prodotto. La parte identificativa del fornitore è un numero assegnato dall'ente che gestisce il sistema GTIN (ad esempio GS1 per EAN), che identifica univocamente il fornitore	
CapacitaComplessiva	Esprime la totale capienza del contenitore.	
CapacitaLavorazione	Esprime la reale quantità utilizzata della capienza del contenitore.	
Potenza	Esprime i watt del contenitore (bollitore, fermentatore).	
Quanti_most	Esprime la quantità di mosto utilizzata.	
Gradi_plato	Indicano la percentuale di zuccheri presente nel mosto, prima della fermentazione.	
SSN	Numero di previdenza sociale di un lavoratore, in questo caso il nostro mastro birraio.	
Grado_alc	Esprime la percentuale di alcol presente nella birra.	
Part_catastale	Codice identificativo delle attività commerciali.	
NumFusti	Numero di fusti è un identificatore di barili o contenitori di birra. Il numero di fusto è spesso utilizzato in combinazione con altri sistemi di identificazione, come ad esempio GTIN, per fornire una tranciabilità completa del fusto dalla produzione al consumo finale.	

Tabella 1.1: Glossario

1.3 Diagramma EE/R

Di seguito viene riportato il **diagramma EE/R**, estensione del modello E/R (*entity-relationship*), che rappresenta lo schema concettuale del database. Ogni **entità** rappresentata nel diagramma possiede **attributi** che descrivono le loro **proprietà**. Inoltre vengono espresse anche le **totalità delle relazioni tra entità**, che possono godere di diversi tipi di **molteplicità**.

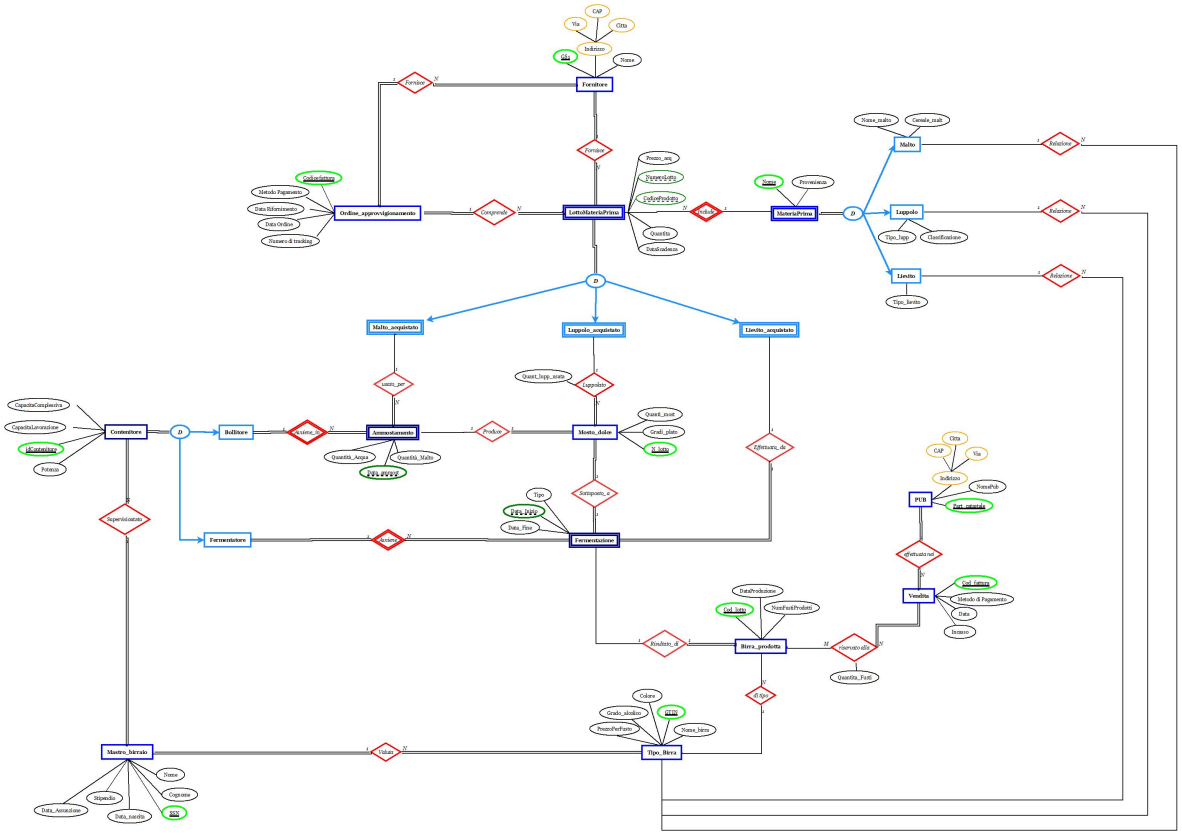

Figura 1.1: Diagramma EE/R

Vediamo di seguito, nel dettaglio, le tre sezioni di questo diagramma.

1.3.1 Diagramma EE/R - Gestione_ordine

Il questa parte possiamo vedere come avviene la **fase di acquisto**, tramite **fornitore**, delle **materie prime**. Di ogni materia prima dobbiamo sapere il fornitore di provenienza.

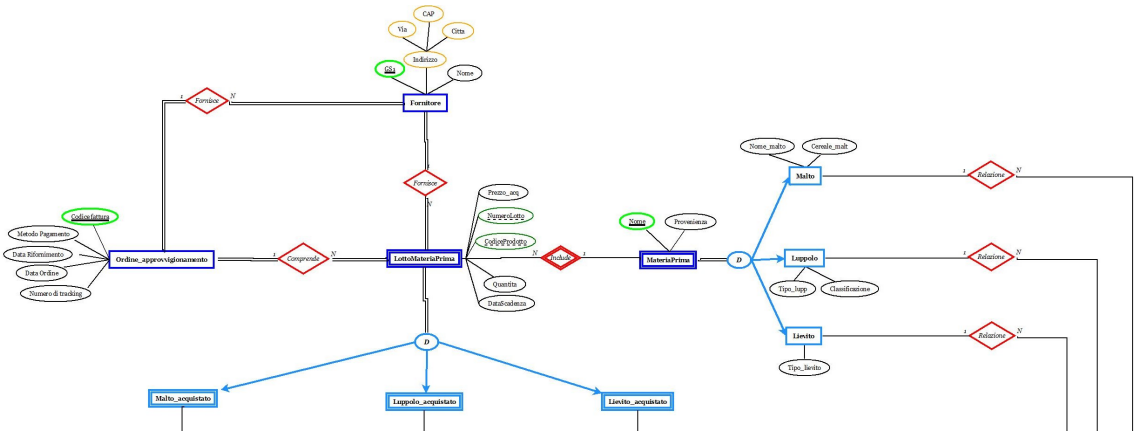

Figura 1.2: Diagramma EE/R - Gestione_ordine

1.3.2 Diagramma EE/R - Gestione_produzione

In questa sezione possiamo vedere nel dettaglio come avviene la **produzione della birra**. Si passa dall'**ammostamento** che avviene nei **bollitori** che poi produce il **mosto dolce**, quest'ultimo viene sottoposto alla **fermentazione** che avviene negli appositi **fermentatori**. Dopodiché dalla **fermentazione** si ottiene la **birra prodotta**. Il **mastro birraio** si occupa del controllo qualità.

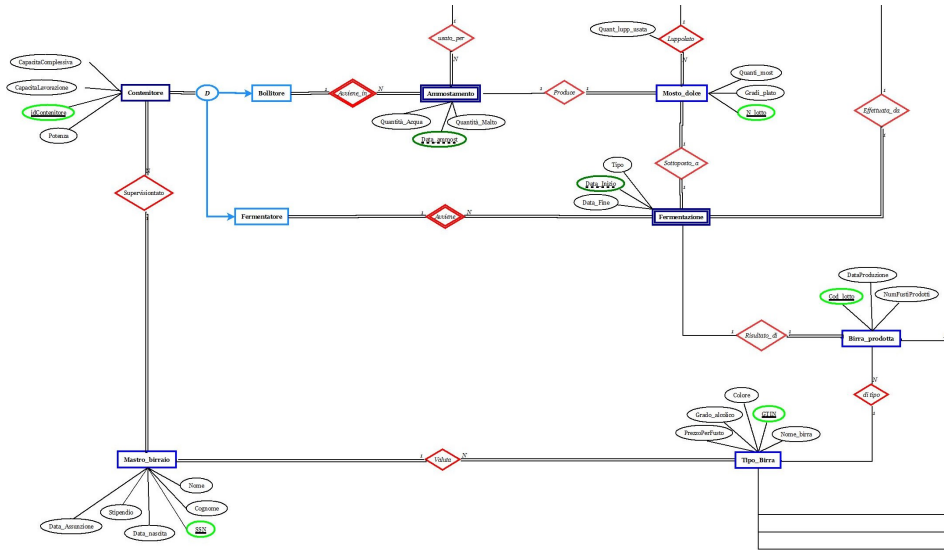

Figura 1.3: Diagramma EE/R - Gestione_produzione

1.3.3 Diagramma EE/R - Gestione_vendita

Qui vediamo la sezione del diagramma dedicata alla **vendita della birra** nei **PUB**.

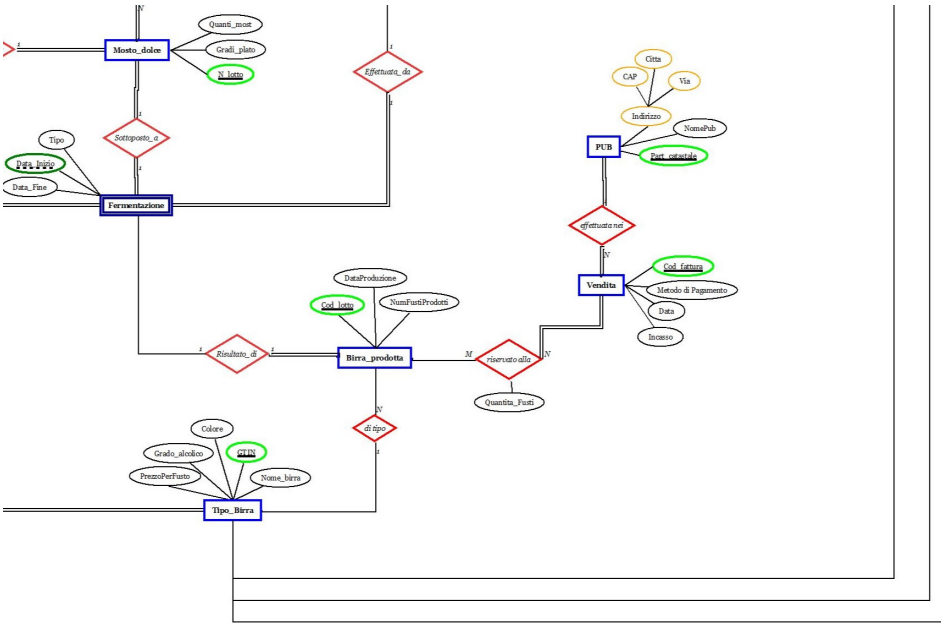

Figura 1.4: Diagramma EE/R - Gestione_vendita

1.4 Diagramma relazionale

Il diagramma relazionale è visibile in figura, ed è molto importante per orientarci sull'implementazione nel **DBMS** della nostra base di dati. Per quanto riguarda la traduzione del **diagramma EER** in **diagramma relazionale**, verranno utilizzate le regole delle molteplicità viste durante il corso: per quanto riguarda le specializzazioni presenti, è stato scelto di utilizzare la tecnica del **partizionamento verticale**, la quale consiste nel creare una tabella per ***l'entità madre*** *avente tutti gli attributi comuni per le* ***entità figlie****, ed una tabella per ogni* ***entità figlia*** *avente come chiave esterna e primaria la chiave primaria dell'****entità madre***, oltre ad avere gli attributi esclusivi di tale entità.

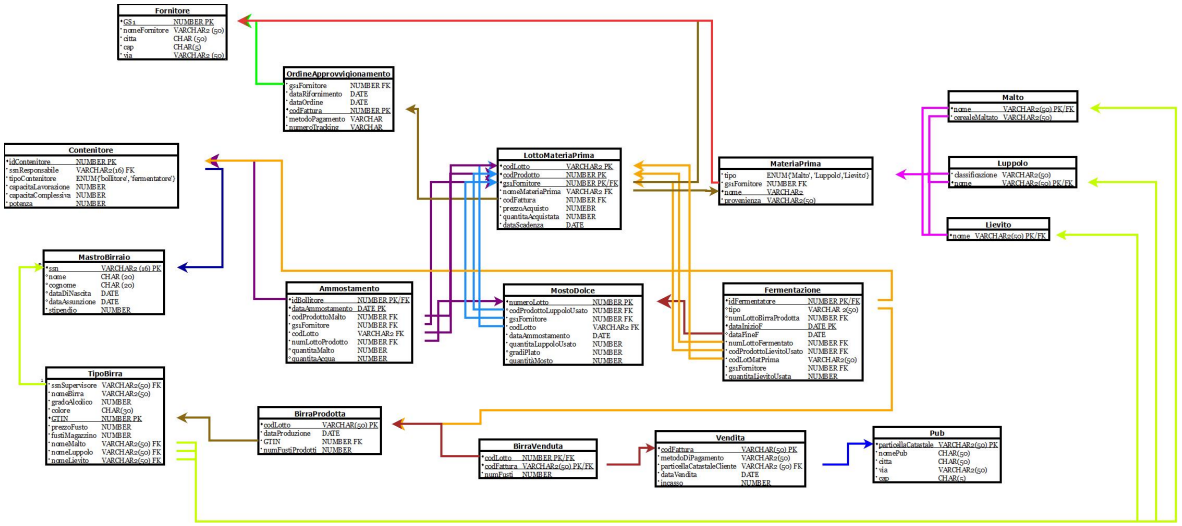

Figura 1.5: Diagramma Relazionale

1.4.1 Diagramma relazionale - Gestione_ordine

Di seguito è riportata nello specifico la traduzione della parte relativa alla gestione dell' ordine.

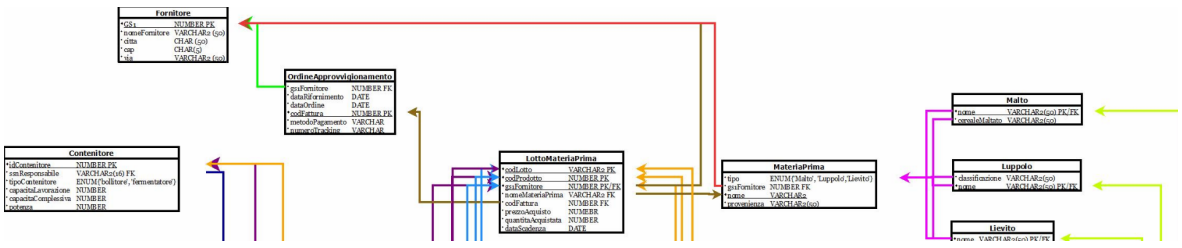

Figura 1.6: Diagramma Relazionale - Gestione_ordine

1.4.2 Diagramma relazionale - Gestione_produzione

Di seguito riportiamo, nel dettaglio, la parte relativa alla gestione della fase di produzione della birra, che avviene principalmente tra l'ammostamento e la fermentazione.

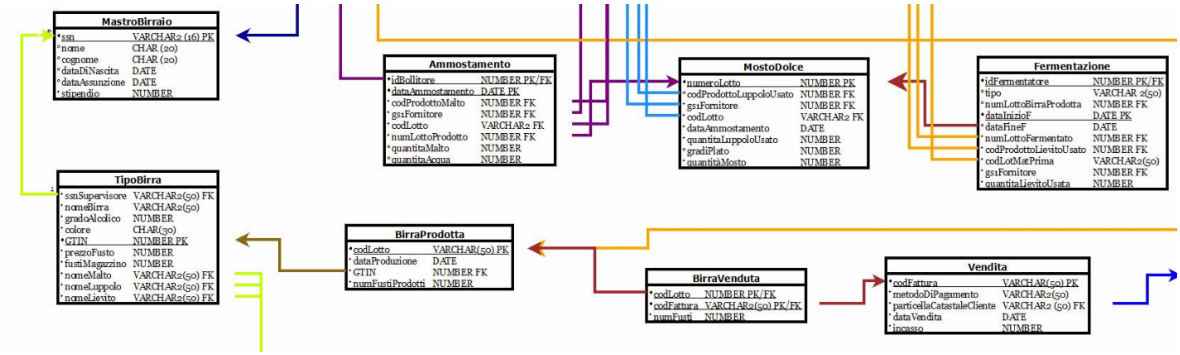

Figura 1.7: Diagramma Relazionale - Gestione_produzione

1.4.3 Diagramma relazionale - Gestione_vendita

Qui nello specifico abbiamo evidenziato la gestione delle vendite.

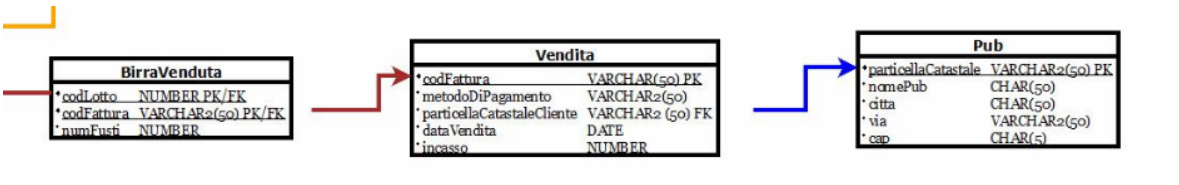

Figura 1.8: Diagramma Relazionale - Gestione_vendita

1.5 Utenti e le loro categorie

Al fine di rendere il nostro database una base dati chiara ed efficiente, si è reso necessario suddividere in quattro categorie gli utenti che possono accedervi.
Il primo utente è **l'amministratore** del database, una persona esterna all'azienda che si occupa dell'implementazione della base di dati, e che di conseguenza **ha accesso a tutti i privilegi possibili**, quali la visualizzazione dei dati e la modifica di essi.
Il secondo utente e il **mastro birraio** che ha il compito di ricercare la birra che viene venduta di meno e applicare uno sconto del 20% sul prossimo acquisto del fusto di tale birra (***ScontoFusto***). Inoltre deve controllare che il malto inglese che costa di più in media, cosi da aumentare del 20% il prezzo delle birre che lo utilizzano(***IncrementaPrezzo***). Infine deve diminuire del 30% il prezzo dei fusti delle birre fatte con malti che hanno più di 100 unità in stock (***DecrementaPrezzo***).

1.5.1 Operazioni degli utenti

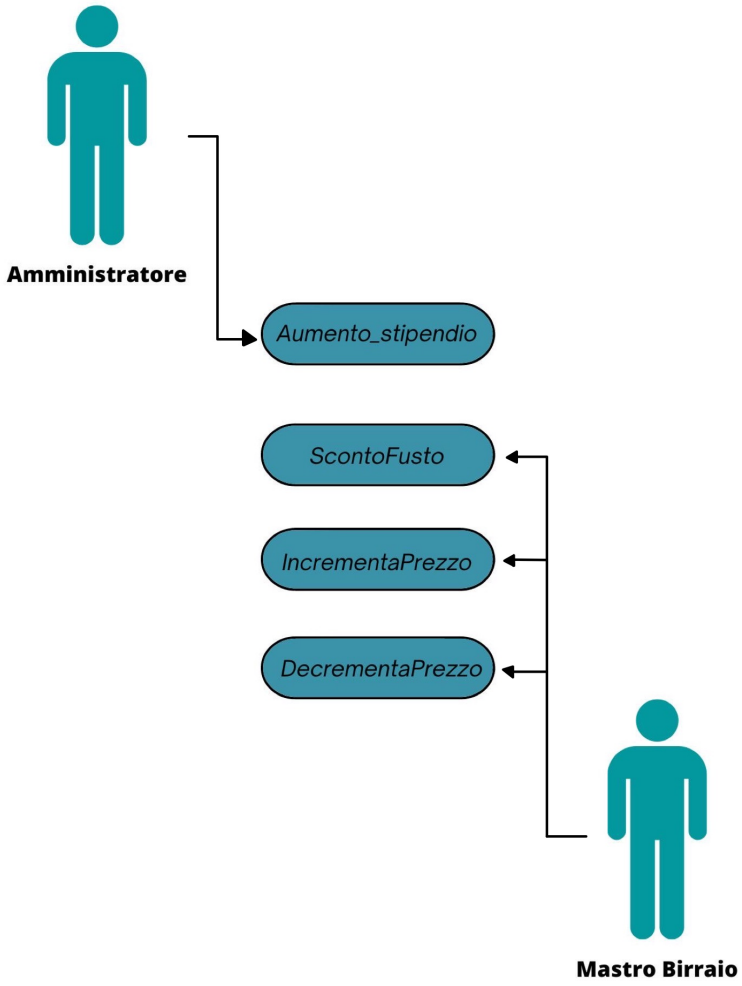

Figura 1.9: Operazioni degli utenti

Escludendo le operazioni base, quali l'inserimento, l'aggiornamento o la modifica di tuple e la visualizzazione di informazioni tramite viste, le operazioni più complesse che possono essere effettuate dagli utenti vengono implementate tramite **procedure**.
Nel diagramma seguente sono rappresentati i casi d'uso, in cui sono visibili le operazioni implementate e gli utenti coinvolti per ogni operazione.
Più nello specifico seguono le schede delle diverse operazioni:

Operazione	***AumentoStipendio***
Scopo:	Attribuisce un incremento del 20% allo stipendio del mastro birraio responsabile del tipo di birra più venduta.
Argomenti:	Non prende argomenti in input.
Risultato:	Si stampa il corretto inserimento dello stipendio aumentato.
Usa:	`MASTROBIRRAIO`, `TIPOBIRRA`, `BIRRAPRODOTTA`, `BIRRAVENDUTA`
Modifica:	Si modifica il valore di `Stipendio`, al quale viene aggiunto il valore calcolato del bonus del 20%.
Prima:	Lo stipendio dato ad un masto birraio è sempre quello base.
Poi:	Al mastro birraio che ha assunto uno stipendio e che è responsabile del tipo di birra più venduto, viene aggiunto l'incremento e mostrato a video.

Operazione	***ScontoFusto***
Scopo:	Trovare la birra che viene venduta di meno e applicare uno sconto del 25% sul prossimo acquisto del fusto di tale birra.
Argomenti:	Non prende argomenti in input.
Risultato:	Si stampa la birra che viene venduta di meno.
Usa:	`TIPOBIRRA`, `BIRRAPRODOTTA`,`BIRRAVENDUTA`
Modifica:	Si modifica il valore di `prezzo Fusto`, al quale viene applicato lo sconto del 25%.
Prima:	Il prezzo della birra meno venduta è quello predefinito.
Poi:	Alla birra venduta di meno viene decrementato il 25% dal prezzo finale per ogni fusto.

Operazione	***IncrementaPrezzo***
Scopo:	Al malto inglese che è stato pagato mediamente di più viene aumentato del 20% il prezzo delle birre che lo usano.
Argomenti:	Non prende argomenti in input.
Risultato:	Non stampa niente a video ma incrementa il prezzo delle birre che utilizzano il malto inglese.
Usa:	`TIPOBIRRA`, `LOTTOMATERIAPRIMA`,`MATERIAPRIMA`
Modifica:	Si modifica il valore di **prezzo fusto** e lo si incrementa del 20%.
Prima:	Il prezzo della birra che utilizza malto inglese mediamente pagata di più il prezzo è quello predefinito.
Poi:	Alla birra che fa uso di malto inglese pagata in media di più viene incrementato il pezzo del 20%.

La successiva procedura fa uso di tre **viste** che vengono menzionate nel campo **"usa"**, che poi vedremo nel dettaglio più avanti.

Operazione	***DecrementaPrezzo***
Scopo:	Diminuisce del 30% il prezzo dei fusti delle birre fatte con malti che hanno più di 100 unità in stock.
Argomenti:	Non prende argomenti in input.
Risultato:	Non stampa niente a video ma diminuisce il prezzo delle birre che utilizzano i malti con più di 100 unità in stock.
Usa:	`TIPOBIRRA`, `LOTTOMATERIAPRIMA`, `Acquisti`, `MaltiUsati`, `MaltiRimanenti`
Modifica:	Si modifica il valore di **prezzo fusto** e viene decrementato del 30%.
Prima:	Il prezzo della birra che utilizza malto con più di 100 unità in stock è quello predefinito.
Poi:	Alla birra che fa uso di malto con più di 100 unità in stock decrementato il pezzo del 30%.

1.6 Volumi

Abbiamo racchiuso nel seguente elenco le informazioni contenute nelle tabelle del database, il numero verosimile di tuple che conterranno una volta che il database sarà funzionante e l'incremento atteso per ognuna di loro in un tempo prefissato.
Poiché come regola aziendale non ci possono essere più di 20 mastri birrai, si stima di assumere altri 5 mastri birrai nell'anno e che poi l'incremento sia nullo.

Tabella	Tipo	Volume	Incremento	Periodo
FORNITORE	E	15	15	anno
MASTROBIRRAIO	E	15	5	anno
CONTENITORE	E	15	5	anno
PUB	E	15	10	trimestre
MATERIAPRIMA	ED	15	15	settimana
MALTO	E	5	20	mese
LUPPOLO	E	5	20	mese
LIEVITO	E	5	20	mese
TIPOBIRRA	E	15	10	anno
ORDINEAPPROVIGGIONAMENTO	E	15	1	mese
LOTTOMATERIAPRIMA	ED	15	1	mese
MOSTODOLCE	E	15	2	settimana
AMMOSTAMENTO	ED	8	2	settimana
BIRRAPRODOTTA	E	15	100	bimestre
FERMENTAZIONE	ED	5	1	settimana
VENDITA	E	15	15	settimana
BIRRAVENDUTA	A	15	30	settimana
Malto_acquistato	A	5	5	settimana
Luppolo_acquistato	A	5	5	settimana
Lievito_acquistato	A	5	5	settimana

Tabella 1.2: Tavola dei volumi

- **Tipo**: **E-ED-A** rappresentano rispettivamente le **entità**, le **entità deboli** e le **tabelle di transizione**.
- **Incremento**: è l'incremento atteso del **numero di tuple**.
- **Periodo**: è il periodo entro il quale **l'incremento è atteso**.

1.7 Vincoli d'integrità

Vengono qui riportati tutti i vincoli, **statici** e **dinamici**, del database. Da questi sono esclusi i vincoli di chiave primaria e di chiave esterna in quanto banali.

1.7.1 Vincoli d'integrità statici

I vincoli di integrità **statici** impongono dei limiti sui valori ammissibili degli attributi delle entità. Per quanto riguarda la parte relativa alla **gestione dell'ordine**:

- Per quanto riguarda il **metodoPagamento** di un ordine di approvvigionamento deve essere un "Assegno" o un "Bonifico.
- Il **Tipo** di materia prima deve essere compreso tra *"Malto", "Lievito" e "Luppolo"*.
- Il **CerealeMaltato** deve essere tra *"Orzo","Segale","Frumento","Mais"*.
- La **Classificazione** del luppolo è compresa tra *"Amaricante","Aromatizzante","Misto"*.
- **nomeLievito** sono utilizzabili solo *"American Ale","Belgian Ale","English Ale","German Wheat","Irish Ale"*.

Quelli relativi alla **gestione della produzione**:

- Per quanto riguarda lo **stipendio** di un mastro birraio, non è possibile inserire uno stipendio inferiore alla soglia minima di $1000,00£$.
- L' **idContenitore** deve essere esclusivamente un "bollitore" o "fermentatore".
- I **GradiPlato** del mosto dolce deve essere una quantità maggiore di 0.
- La **QuantitaMosto** deve essere una quantità maggiore di 0.
- La **QuantitàMalto** contenuta nell'ammostamento deve essere una quantità maggiore di 0.
- La **QuantitaAcqua** contenuta nell'ammostamento deve essere una quantità maggiore di 0.
- Il **TipoFermentazione** deve essere compreso tra "alta" o "bassa".
- La **QuantitaLievUsato** nella fermentazione deve essere una quantità maggiore di 0.
- Il **Colore** della birra deve essere "bionda","rossa","scura".

Infine quelli della **gestione vendita**:

- Il **MetodoPagamento** della vendita deve essere uno tra "assegno" o "bonifico".

1.7.2 Vincoli d'integrità dinamici

I vincoli **dinamici**, sono vincoli di integrità che possono variare nel tempo o che riguardano alcune *regole di business*. Per quanto riguarda la parte relativa alla **gestione dell'ordine**:

- Non si può **acquistare un lievito** che non sia uno di quelli prestabiliti.
- Non si può **acquistare un tipo di luppolo** che non sia tra quelli prestabiliti.
- Non si può acquistare **una materia prima** con una data di scadenza non adeguata.
- Non si può avere una data di rifornimento prima di aver inserito la data di ordine.

Quelli relativi alla **gestione della produzione**:

- Non si può iniziare una nuova **fermentazione** se prima non si è finito l'**ammostamento**.
- Non si può utilizzare una quantità maggiore di quella di **lavorazione** prevista dal **contenitore**.
- Non può essere prodotta una birra di colore diverso da quelli prestabiliti.

Infine quelli della **gestione vendita**:

- Non possono essere venduti più fusti di quelli prodotti.

1.8 Verifica di normalità

Una delle cose più importanti da dover analizzare nel momento della normalizzazione di un database è che non ci siano dipendenze funzionali anomale, limitando la ridondanza. Il processo di normalizzazione sottopone uno schema di relazione a una serie di test per "certificare" se soddisfa una data forma normale.

1.8.1 Prima forma normale

La **prima forma normale** (o **1NF**) è stata definita per non permettere l'uso di attributi multivalore, di attributi composti e delle loro combinazioni. Il nostro database rispetta la prima forma normale in quanto sono presenti tutti campi unici e atomici. Tutti i campi di tipo `DATE`, sono considerati atomici convenzionalmente in ORACLE DBMS.

1.8.2 Seconda forma normale

Questa forma normale (detta anche **2NF**)si applica per le relazioni in cui la chiave primaria è costituita da più attributi, gli attributi non-chiave non devono dipendere funzionalmente solo da una parte della chiave primaria.

1.8.3 Terza forma normale

La **terza forma normale** (o **3NF**) si basa sul concetto di dipendenza transitiva. Ovvero uno schema è in 3NF quando tutti gli attributi di una relazione dipendono funzionalmente solo dalla chiave primaria della relazione e non da attributi non-chiave. All'interno del nostro database è stato controllato che non ci fossero dipendenze funzionali anomale, e ciò ci permette di dire che la terza forma normale è rispettata. Rispettando tale forma, possiamo dire che lo schema è in **Boys-Codd Normal Form**.

Capitolo 2

Implementazione

Di seguito si riportano tutte le implementazioni necessarie a garantire il corretto funzionamento del database, sulla base dei requisiti ottenuti in fase di progettazione.
L'ordine in cui compariranno le varie entità, associazioni, e altre componenti del database non per forza rappresenta l'effettivo ordine in cui vanno eseguite, bensì sono posizionate in questo ordine per poter seguire in maniera più lineare il funzionamento e il legame tra le diverse entità.

2.1 Creazione degli Utenti

Come spiegato precedentemente, **gli utenti del database sono due**, e sono stati creati come segue.

```
01 | create user amministratore  identified by admin;
02 | create user mastrobirraio   identified by supervisore;
```

2.2 Data Definition Language - DDL

Il **DDL** rappresenta uno specchio dello schema relazionale. Tutte le entità e le relazioni tra loro sono state create mediante tante istruzioni `CREATE TABLE`. Ogni tabella contiene tutti i vincoli di integrità necessari a garantirne il corretto funzionamento.

2.2.1 DROP TABLE

Per **evitare conflitti** può essere utile questo drop ai fini della **cancellazione delle tabelle**:

```
01 | drop table Fornitore;
02 | drop table MastroBirraio;
03 | drop table Contenitore;
04 | drop table OrdineApproviggionamento;
05 | drop table MateriaPrima;
06 | drop table Malto;
07 | drop table Luppolo;
08 | drop table Lievito;
09 | drop table LottoMateriaPrima;
10 | drop table MostoDolce;
11 | drop table TipoBirra;
```

```
12 drop table BirraProdotta;
13 drop table Ammostamento;
14 drop table Fermentazione;
15 drop table PUB;
16 drop table Vendita;
17 drop table BirraVenduta;
```

2.3 CREATE TABLE

Vediamo più nel dettaglio le singole tabelle cosa rappresentano e quali informazioni hanno bisogno di immagazzinare ai fini di ottenere un tracciamento completo delle materie utilizzate.

2.3.1 FORNITORE

Questa tabella rappresenta i **fornitori della birreria**. Ogni riga della tabella rappresenta un fornitore univoco, identificato dalla colonna **GS1** che è la chiave primaria della tabella. La tabella include informazioni sul nome del fornitore, la città, il codice postale (CAP) e l'indirizzo.

```
01 CREATE TABLE Fornitore(
02    GS1                NUMBER(9) NOT NULL,
03    nomeFornitore      VARCHAR2(50) NOT NULL,
04    citta              CHAR(50) NOT NULL,
05    cap                CHAR(5) NOT NULL,
06    via                VARCHAR2(50),
07
08    CONSTRAINT FORN_PK
09       PRIMARY KEY(GS1)
10 );
```

2.3.2 MASTRO BIRRAIO

Questa tabella rappresenta i **mastri birrai della birreria**. Ogni riga della tabella rappresenta un mastro birraio univoco, identificato dalla colonna **ssn** che è la chiave primaria della tabella. La tabella include informazioni sul nome, cognome, data di nascita, data di assunzione, stipendio del mastro birraio sul quale è presente un ***check*** che indica lo stipendio base di ogni mastro birraio che viene inserito.

```
01 CREATE TABLE MastroBirraio(
02    ssn                VARCHAR2(16) NOT NULL,
03    nome               CHAR(20) NOT NULL,
04    cognome            CHAR(20) NOT NULL,
05    dataNascita        DATE NOT NULL,
06    dataAssunzione     DATE NOT NULL,
07    stipendio          NUMBER NOT NULL CHECK(Stipendio >= 1000),
08
09    CONSTRAINT MB_PK
10       PRIMARY KEY(ssn)
11 );
```

2.3.3 CONTENITORE

Questa tabella rappresenta i **contenitori utilizzati nella produzione di birra**. Ogni riga della tabella rappresenta un contenitore univoco, identificato dalla colonna **idContenitore** che è la chiave primaria della tabella. La tabella include informazioni sul responsabile del contenitore, il tipo di contenitore, la capacità di lavorazione, la capacità complessiva e la potenza. La tabella fa **riferimento alla tabella MastroBirraio** tramite una chiave esterna sulla colonna **ssnResponsabile**.

```
01 | CREATE TABLE Contenitore(
02 |    idContenitore       NUMBER NOT NULL,
03 |    ssnResponsabile     VARCHAR2(16),
04 |    tipoContenitore     CHAR(20) CHECK(tipoContenitore IN ('Bollitore' , ↩
   |     'Fermentatore')),
05 |    capacitaLavorazione NUMBER,
06 |    capacitaComplessiva NUMBER NOT NULL,
07 |    potenza             NUMBER NOT NULL,
08 |
09 |    CONSTRAINT CONT_PK
10 |       PRIMARY KEY(idContenitore),
11 |    CONSTRAINT CONT_FK_MB
12 |       FOREIGN KEY (ssnResponsabile) REFERENCES MastroBirraio(ssn) on ↩
   |     delete cascade
13 | );
```

2.3.4 ORDINEAPPROVIGGIONAMENTO

La tabella rappresenta **gli ordini di approvvigionamento effettuati dalla birreria**. Ogni riga della tabella rappresenta un ordine univoco, identificato dalla colonna **codFattura** che è la chiave primaria della tabella. La tabella include informazioni sul fornitore dell'ordine (identificato dalla colonna gs1Fornitore), la data di rifornimento, la data dell'ordine, il metodo di pagamento e il numero di tracking. La tabella fa riferimento alla tabella Fornitore tramite una chiave esterna sulla colonna **gs1Fornitore**.

```
01 | CREATE TABLE OrdineApproviggionamento(
02 |    gs1Fornitore       NUMBER(9) NOT NULL,
03 |    dataRifornimento   DATE NOT NULL,
04 |    dataOrdine         DATE NOT NULL,
05 |    codFattura         NUMBER NOT NULL,
06 |    metodoPagamento    VARCHAR2(50) CHECK(metodoPagamento IN('Assegno','↩
   |     Bonifico')),
07 |    numeroTracking     VARCHAR2(50),
08 |
09 |    CONSTRAINT OAPP_PK
10 |       PRIMARY KEY(codFattura),
11 |    CONSTRAINT OAPP_FK_FORN
12 |       FOREIGN KEY(gs1Fornitore) REFERENCES Fornitore(GS1) on delete ↩
   |     cascade
13 |
14 | );
```

2.3.5 MATERIAPRIMA

Questa tabella, rappresenta la **materia prima utilizzata nella produzione di birra**. Ha tre colonne:

- "**tipo**" (il tipo di materia prima, che può essere malto, luppolo o lievito);
- "**nomeMatPrim**" (il nome della materia prima);
- "**provenienza**" (la provenienza della materia prima);

C'è anche un vincolo di chiave primaria sulla colonna "nomeMatPrim".

```
01 | CREATE TABLE MateriaPrima(
02 |    tipo               VARCHAR2(50) CHECK(tipo IN('Malto','Luppolo','↩
   |     Lievito')),
03 |    nomeMatPrim        VARCHAR2(50),
04 |    provenienza        VARCHAR2(50),
05 |
06 |    CONSTRAINT MAT_P_PK
07 |       PRIMARY KEY(nomeMatPrim)
08 | );
```

2.3.6 MALTO

La tabella Malto descrive **i vari tipi di malto utilizzati nella produzione della birra**. Ha due colonne:

- **nomeMalto**: rappresenta il **nome del malto**;
- **cerealeMaltato**: rappresenta il **cereale** che è stato maltato. La colonna "cerealeMaltato" ha un vincolo di controllo "CHECK", che specifica che i valori possibili per questa colonna sono solo *"**Orzo**"*, *"**Segale**"*, *"**Frumento**"* e *"**Mais**"*;

. La tabella Malto ha anche due vincoli:

- una **chiave primaria** "MALT_PK" sulla colonna "nomeMalto";
- una **chiave esterna** "MALT_PKFK" sulla colonna "nomeMalto" che fa riferimento alla tabella "MateriaPrima" sulla colonna "nomeMatPrim";

```
01 | CREATE TABLE Malto(
02 |    nomeMalto          VARCHAR2(50) NOT NULL,
03 |    cerealeMaltato     VARCHAR2(50) NOT NULL CHECK(cerealeMaltato IN ('↩
   |     Orzo','Segale','Frumento','Mais')),
04 |
05 |    CONSTRAINT MALT_PK
06 |       PRIMARY KEY(nomeMalto),
07 |    CONSTRAINT MALT_PKFK
08 |       FOREIGN KEY (nomeMalto) REFERENCES MateriaPrima(nomeMatPrim) on ↩
   |     delete cascade
09 | );
```

2.3.7 LUPPOLO

La tabella Luppolo descrive **i vari tipi di luppolo utilizzati nella produzione della birra**. Ha due colonne:

- **nomeLuppolo**: rappresenta il nome del luppolo;
- **classificazione**: rappresenta la classificazione del luppolo, ovvero se è amaricante, aromatizzante o misto. La colonna "classificazione" ha un vincolo di controllo "CHECK", che specifica che i valori possibili per questa colonna sono solo "Amaricante", "Aromatizzante" e "Misto";

La tabella "Luppolo" ha anche due vincoli:

- una **chiave primaria** "LUPP_PK" sulla colonna "nomeLuppolo" ;
- una **chiave esterna** "LUPP_PKFK" sulla colonna "nomeLuppolo" che fa riferimento alla tabella "MateriaPrima" sulla colonna "nomeMatPrim";

```
01 | CREATE TABLE Luppolo(
02 |    nomeLuppolo        VARCHAR2(50) NOT NULL,
03 |    classificazione    VARCHAR2(50) NOT NULL CHECK(classificazione IN ('↩
   |    Amaricante','Aromatizzante','Misto')),
04 |
05 |    CONSTRAINT LUPP_PK
06 |       PRIMARY KEY(nomeLuppolo),
07 |    CONSTRAINT LUPP_PKFK
08 |       FOREIGN KEY (nomeLuppolo) REFERENCES MateriaPrima(nomeMatPrim) on ↩
   |    delete cascade
09 |
10 | );
```

2.3.8 LIEVITO

La tabella Lievito **descrive i vari tipi di lievito utilizzati nella produzione della birra**. Ha una sola colonna:

- **nomeLievito**: rappresenta il nome del lievito e ha un vincolo di controllo "CHECK", che specifica che i valori possibili per questa colonna sono solo "American Ale", "Belgian Ale", "English Ale", "German Wheat" e "Irish Ale";

La tabella "Lievito" ha anche due vincoli:

- una **chiave primaria** "LIEV_PK" sulla colonna "nomeLievito";
- una **chiave esterna** "LIEV_PKFK" sulla colonna "nomeLievito" che fa riferimento alla tabella "MateriaPrima" sulla colonna "nomeMatPrim";

```
01 | CREATE TABLE Lievito(
02 |    nomeLievito        VARCHAR2(50) NOT NULL CHECK (nomeLievito IN ('↩
   |    American Ale','Belgian Ale','English Ale','German Wheat','Irish Ale')↩
   |    ),
03 |
04 |    CONSTRAINT LIEV_PK
05 |       PRIMARY KEY(nomeLievito),
06 |    CONSTRAINT LIEV_PKFK
```

```
07 ||       FOREIGN KEY(nomeLievito) REFERENCES MateriaPrima(nomeMatPrim) on ↩
   ||    delete cascade
08 ||
09 || );
```

2.3.9 LOTTOMATERIAPRIMA

La tabella LottoMateriaPrima rappresenta **un lotto di materia prima che è stato acquistato da un fornitore**. Ha informazioni sul lotto stesso, come il "**codLotto**", il "**codProdotto**", il "**gs1Fornitore**" e il "**nomeMateriaPrima**". Ha anche informazioni sul **prezzo d'acquisto**, la **quantità acquistata** e la **data di scadenza**. C'è un vincolo di **chiave primaria** composta su tre colonne e tre vincoli di **chiave esterna** che fanno riferimento alla tabella MateriaPrima, alla tabella **OrdineApprovvigionamento** e alla tabella **Fornitore**.

```
01 || CREATE TABLE LottoMateriaPrima(
02 ||    codLotto           VARCHAR2(50) NOT NULL,
03 ||    codProdotto        NUMBER(4) NOT NULL,
04 ||    gs1Fornitore       NUMBER(9) NOT NULL,
05 ||    nomeMateriaPrima   VARCHAR2(50),
06 ||    codFattura         NUMBER NOT NULL,
07 ||    prezzoAcquisto     NUMBER NOT NULL,
08 ||    quantitaAcquistata NUMBER NOT NULL,
09 ||    dataScadenza       DATE NOT NULL,
10 ||
11 ||    CONSTRAINT LMAT_P_PK
12 ||       PRIMARY KEY(codLotto,codProdotto,gs1Fornitore),
13 ||    CONSTRAINT LMAT_P_FK_MAT_P
14 ||       FOREIGN KEY (nomeMateriaPrima) REFERENCES MateriaPrima(nomeMatPrim↩
   ||    ) on delete cascade,
15 ||    CONSTRAINT LMAT_P_FK_OAPP
16 ||       FOREIGN KEY(codFattura) REFERENCES OrdineApproviggionamento(↩
   ||    codFattura) on delete cascade,
17 ||    CONSTRAINT LMAT_P_FK_FORN
18 ||       FOREIGN KEY(gs1Fornitore) REFERENCES Fornitore(GS1) on delete ↩
   ||    cascade
19 || );
```

2.3.10 MOSTODOLCE

La tabella MostoDolce rappresenta **il mosto dolce prodotto dalla birreria**. Ha informazioni sul lotto di produzione, come la **data di ammostamento** e la **quantità di luppolo utilizzata**, nonché i **gradi Plato** e la **quantità di mosto prodotto**. C'è un vincolo di **chiave primaria** sulla colonna "**numeroLotto**" e un vincolo di **chiave esterna** che fa riferimento alla tabella **LottoMateriaPrima**.

```
01 || CREATE TABLE MostoDolce(
02 ||    numeroLotto        NUMBER NOT NULL,
03 ||    codProdLuppUsato   NUMBER(4) NOT NULL,
04 ||    gs1Fornitore       NUMBER(9) NOT NULL,
05 ||    codLotto           VARCHAR2(50) NOT NULL,
```

```
    dataAmmostamento   DATE NOT NULL,
    quantitaLuppUsato  NUMBER,
    gradiPlato         NUMBER CHECK (gradiPlato > 0),
    quantitaMosto      NUMBER CHECK (quantitaMosto > 0),

    CONSTRAINT MD_PK
       PRIMARY KEY(numeroLotto),
    CONSTRAINT MD_LUP_FK
       FOREIGN KEY(codProdLuppUsato,gs1Fornitore,codLotto) REFERENCES
    LottoMateriaPrima(codProdotto,gs1Fornitore,codLotto) on delete
     cascade
);
```

2.3.11 TIPOBIRRA

La tabella TipoBirra contiene informazioni sul **tipo di birra che viene prodotto dalla birreria**, come il **GTIN** (numero identificativo univoco), il **ssn** del supervisore, il **nome della birra**, il **grado alcolico**, il **colore**, il **numero di fusti in magazzino**, il **prezzo per fusto**, il **nome del malto**, **luppolo** e **lievito** usato per produrre la birra. La tabella ha anche diverse restrizioni, come la chiave primaria (GTIN), e le **chiavi esterne** che fanno riferimento ad altre tabelle come "**MastroBirraio**", "**Malto**", "**Luppolo**" e "**Lievito**".

```
CREATE TABLE TipoBirra(
    GTIN               NUMBER(4) NOT NULL,
    ssnSupervisore     VARCHAR2(16) NOT NULL,
    nomeBirra          VARCHAR2(50) NOT NULL,
    gradoAlcolico      NUMBER NOT NULL CHECK(gradoAlcolico >= 0),
    colore             VARCHAR2(30) CHECK (LOWER(colore) IN ('bionda','
     rossa','scura')),
    fustiMagazzino     NUMBER NOT NULL,
    prezzoFusto        NUMBER NOT NULL,
    nomeMalto          VARCHAR2(50) NOT NULL,
    nomeLuppolo        VARCHAR2(50) NOT NULL,
    nomeLievito        VARCHAR2(50) NOT NULL,

    CONSTRAINT TIPBIRR_PK
       PRIMARY KEY(GTIN),
    CONSTRAINT TIPBIRR_FK
       FOREIGN KEY(ssnSupervisore) REFERENCES MastroBirraio(ssn) on
     delete cascade,
    CONSTRAINT TIPBIRR_MALTO_FK
       FOREIGN KEY(nomeMalto) REFERENCES Malto(nomeMalto) on delete
     cascade,
    CONSTRAINT TIPBIRR_LUPP_FK
       FOREIGN KEY(nomeLuppolo) REFERENCES Luppolo(nomeLuppolo) on delete
      cascade,
    CONSTRAINT TIPBIRR_LIEV_FK
       FOREIGN KEY(nomeLievito) REFERENCES Lievito(nomeLievito) on delete
      cascade

);
```

2.3.12 BIRRAPRODOTTA

La tabella BirraProdotta contiene informazioni sulla **produzione di birra**, come il **codice lotto**, la **data di produzione**, il **GTIN del tipo di birra prodotto** e il **numero di fusti prodotti**. Anche in questa tabella ci sono restrizioni, come la **chiave primaria** (codLotto) e la **chiave esterna** che fa riferimento alla tabella "**TipoBirra**" (GTIN).

```
01 | CREATE TABLE BirraProdotta(
02 |    codLotto           VARCHAR2(50),
03 |    dataProduzione     DATE NOT NULL,
04 |    GTIN               NUMBER(4) NOT NULL,
05 |    numFustiProdotti   NUMBER NOT NULL,
06 |
07 |    CONSTRAINT BIRRP_PK
08 |       PRIMARY KEY(codLotto),
09 |    CONSTRAINT BIRRP_FK
10 |       FOREIGN KEY(GTIN) REFERENCES TipoBirra(GTIN) on delete cascade
11 | );
```

2.3.13 AMMOSTAMENTO

La tabella Ammostamento contiene informazioni **sulla fase dell'ammostamento**, come l'**identificativo del bollitore**, la **data dell'ammostamento**, il **codice prodotto malto**, il codice **GS1 del fornitore**, il **codice lotto**, il **numero di lotto prodotto**, la **quantità di malto e acqua usati**. Questa tabella ha anche restrizioni, come la **chiave primaria** composta (idBollitore, dataAmmostamento), e le **chiavi esterne** che fanno riferimento a tabelle come "**LottoMateriaPrima**", "**Contenitore**" e "**MostoDolce**".

```
01 | CREATE TABLE Ammostamento(
02 |    idBollitore        NUMBER NOT NULL,
03 |    dataAmmostamento   DATE NOT NULL,
04 |    codProdottoMalto   NUMBER(4) NOT NULL,
05 |    gs1Fornitore       NUMBER(9) NOT NULL,
06 |    codLotto           VARCHAR2(50) NOT NULL,
07 |    numLottoProdotto   NUMBER NOT NULL,
08 |    quantitaMalto      NUMBER NOT NULL CHECK(quantitaMalto > 0),
09 |    quantitaAcqua      NUMBER NOT NULL CHECK(quantitaAcqua > 0),
10 |
11 |    CONSTRAINT AMM_PK
12 |       PRIMARY KEY (idBollitore,dataAmmostamento),
13 |    CONSTRAINT AMM_FK_LOTMP
14 |       FOREIGN KEY(codProdottoMalto,gs1Fornitore,codLotto) REFERENCES ↩
    |    LottoMateriaPrima(codProdotto,gs1Fornitore,codLotto) on delete ↩
    |    cascade,
15 |    CONSTRAINT AMM_FKPK_CONT
16 |       FOREIGN KEY(idBollitore) REFERENCES Contenitore(idContenitore) on ↩
    |    delete cascade,
17 |    CONSTRAINT AMM_FK_MSTDLC
18 |       FOREIGN KEY(numLottoProdotto) REFERENCES MostoDolce(numeroLotto) ↩
    |    on delete cascade
19 | );
```

2.3.14 FERMENTAZIONE

La tabella Fermentazione contiene informazioni **sulla fase della fermentazione**, come **l'identificativo del fermentatore**, il **tipo di fermentazione**, il **numero di lotto della birra prodotta**, la **data di inizio e fine della fermentazione**, il **numero di lotto fermentato**, il **codice lotto della materia prima**, il **codice prodotto del lievito usato**, il **codice GS1 del fornitore**, la **quantità di lievito usata**. Anche in questa tabella ci sono restrizioni, come la **chiave primaria** composta (idFermentatore, dataInizioF) e le **chiavi esterne** che fanno riferimento a tabelle come "**LottoMateriaPrima**" e "**Lievito**".

```
01 | CREATE TABLE Fermentazione(
02 |    idFermentatore     NUMBER NOT NULL,
03 |    tipoFermentazione  VARCHAR2(50) CHECK(LOWER(tipoFermentazione) IN('↩
   |    alta','bassa')),
04 |    numLottoBirraProd  VARCHAR2(50),
05 |    dataInizioF        DATE NOT NULL,
06 |    dataFineF          DATE NOT NULL,
07 |    numLotFermentato   NUMBER NOT NULL,
08 |    codLottoMatPrim    VARCHAR2(50) NOT NULL,
09 |    codProdLievUsato   NUMBER(4) NOT NULL,
10 |    gs1Fornit          NUMBER(9) NOT NULL,
11 |    quantitaLievUsato  NUMBER NOT NULL CHECK(quantitaLievUsato > 0),
12 |
13 |    CONSTRAINT FERM_PK
14 |       PRIMARY KEY (idFermentatore,dataInizioF),
15 |    CONSTRAINT FERM_PKFK_CONT
16 |       FOREIGN KEY(idFermentatore) REFERENCES Contenitore(idContenitore) ↩
   |    on delete cascade,
17 |    CONSTRAINT FERM_FK_MD
18 |       FOREIGN KEY(numLotFermentato) REFERENCES MostoDolce(numeroLotto) ↩
   |    on delete cascade,
19 |    CONSTRAINT FERM_FK_LOTMATPRIM
20 |       FOREIGN KEY(codLottoMatPrim,codProdLievUsato,gs1Fornit) REFERENCES↩
   |     LottoMateriaPrima(codLotto,codProdotto,gs1Fornitore) on delete ↩
   |    cascade,
21 |    CONSTRAINT FERM_FK_BIRRP
22 |       FOREIGN KEY(numLottoBirraProd) REFERENCES BirraProdotta(codLotto) ↩
   |    on delete cascade
23 | );
```

2.3.15 PUB

Questa tabella rappresenta i pub che vendono birra. Qui è importante avere la **particella catastale del PUB**, oltre alle informazioni di base come il **nome del pub** e il **luogo** in cui si trova.

```
01 | CREATE TABLE PUB(
02 |     particellaCatastale  VARCHAR2(50) NOT NULL,
03 |     nomePub              CHAR(50) NOT NULL,
04 |     citta                CHAR(50) NOT NULL,
05 |     via                  VARCHAR2(50) NOT NULL,
06 |     cap                  CHAR(5) NOT NULL,
```

```
07 |
08 |    CONSTRAINT PUB_PK
09 |       PRIMARY KEY(particellaCatastale)
10 | );
```

2.3.16 VENDITA

Questa tabella rappresenta le vendite effettuate dai pub. Ha 6 colonne:

- **codFattura**: identificativo univoco della fattura, è la chiave primaria della tabella
- **partCatastaleCli**: identificativo del pub che ha effettuato la vendita, è una chiave esterna che fa riferimento alla tabella PUB
- **metodoPagamento**: metodo utilizzato per il pagamento
- **dataVendita**: data in cui è stata effettuata la vendita
- **ricavo**: quantità di denaro ricavata dalla vendita

```
01 | CREATE TABLE Vendita(
02 |     codFattura          VARCHAR2(50) NOT NULL,
03 |     partCatastaleCli    VARCHAR2(50),
04 |    metodoPagamento      VARCHAR2(50) CHECK(LOWER(metodoPagamento) IN('↩
   |     assegno','bonifico')),
05 |    dataVendita          DATE NOT NULL,
06 |    ricavo               NUMBER,
07 |
08 |    CONSTRAINT V_PK
09 |       PRIMARY KEY(codFattura),
10 |    CONSTRAINT V_FK_PUB
11 |       FOREIGN KEY(partCatastaleCli) REFERENCES PUB(particellaCatastale) ↩
   |     on delete cascade
12 | );
```

2.3.17 BIRRAVENDUTA

Questa tabella rappresenta la quantità di birra venduta in una singola transazione. Ha 4 colonne:

- **codLotto**: identificativo univoco del lotto di birra venduto, è una chiave esterna che fa riferimento alla tabella BirraProdotta
- **codFattura**: identificativo della fattura associata alla vendita, è una chiave esterna che fa riferimento alla tabella Vendita
- **numFusti**: quantità di fusti di birra venduti

```
01 | CREATE TABLE BirraVenduta(
02 |    codLotto           VARCHAR2(50) NOT NULL,
03 |    codFattura         VARCHAR2(50),
04 |    numFusti           NUMBER,
05 |
06 |    CONSTRAINT BV_PK
07 |       PRIMARY KEY(codLotto,codFattura),
```

```
08 |    CONSTRAINT BV_PKFK1
09 |       FOREIGN KEY(codLotto) REFERENCES BirraProdotta(codLotto) on delete↩
       cascade,
10 |    CONSTRAINT BV_PKFK2
11 |       FOREIGN KEY(codFattura) REFERENCES Vendita(codFattura) on delete ↩
       cascade
12 | );
```

2.4 Data Manipulation Language - DML

Il popolamento delle tabelle sopra create avviene attraverso operazioni messe a disposizione dal linguaggio SQL, cioè i comandi `INSERT` e `UPDATE`. In questo caso sono stati utilizzati solo **INSERT** in quanto si fa riferimento al primo inserimento nelle tabelle sopra create.
Verrà mostrato solo un' esempio di inserimento per ogni tipo di tabella creata:

```
01 | insert into Fornitore values (000123011, 'AURIAN SAS', 'Condom', '32100'↩
       , '5 Avenue de la Gare');
02 |
03 | insert into MastroBirraio values ('IT0011220', 'Maurizio', 'Scarponi', ↩
       TO_DATE('15/01/1992','DD/MM/YYYY'), TO_DATE('25/01/2006','DD/MM/YYYY'↩
       ), 1600.00);
04 |
05 | insert into Contenitore values (1, 'IT0011220', 'Bollitore', 400, 500, ↩
       2000);
06 |
07 | insert into PUB values ('PT00-01', 'Casa del Popolo', 'Verbano', 'Via F.↩
       lli Borghini 34', '28877');
08 |
09 | insert into MateriaPrima values ('Malto', 'Pale Ale Malt', 'Regno Unito'↩
       );
10 |
11 | insert into Malto values ('Pale Ale Malt', 'Orzo');
12 |
13 | insert into Luppolo values ('Cascade', 'Amaricante');
14 |
15 | insert into Lievito values ('American Ale');
16 |
17 | insert into TipoBirra values (1001, 'IT0011220', 'Stout', 8.5, 'Scura', ↩
       0, 100.00 , 'Pale Ale Malt', 'Cascade', 'American Ale');
18 |
19 | insert into OrdineApprovviggionamento values (000123011, TO_DATE('↩
       09/01/2021','DD/MM/YYYY'), TO_DATE('01/01/2021','DD/MM/YYYY'), ↩
       1000550, 'Bonifico', 'SP-RT-0051-01');
20 |
21 | insert into LottoMateriaPrima values ('LT-0220001', 1001, 000123011,'↩
       Pale Ale Malt', 1000550, 200.99, 100, TO_DATE('01/03/2023','DD/MM/↩
       YYYY'));
22 |
23 | insert into MostoDolce values (0102001, 1003, 000123013, 'LT-0220003', ↩
       TO_DATE('01/02/2021','DD/MM/YYYY'), 2, 50, 1);
```

```
24 |
25 | insert into Ammostamento values (1, TO_DATE('02/02/2021','DD/MM/YYYY'), ↩
       1001, 000123011,'LT-0220001', 0102001, 9, 3);
26 |
27 | insert into BirraProdotta values ('BP-0010201', TO_DATE('15/02/2021','DD↩
       /MM/YYYY'), 1001, 400);
28 |
29 | insert into Fermentazione values (4, 'alta', 'BP-0010202', TO_DATE('↩
       01/02/2023','DD/MM/YYYY'), TO_DATE('15/02/2023','DD/MM/YYYY'), ↩
       0102002, 'LT-0220002', 1002, 000123012, 2);
30 |
31 | insert into Vendita values ('VD--0220001', 'PT00-01', 'Assegno', TO_DATE↩
       ('20/02/2021','DD/MM/YYYY'), 500.99);
32 |
33 | insert into BirraVenduta values ('BP-0010201', 'VD--0220001', 9);
```

2.5 Trigger

I diversi vincoli di produzione e i **controlli** in fase di inserimento, aggiornamento, eliminazione dei dati sono stati implementati tramite **trigger DML**. Nel dettaglio il database presenta i seguenti trigger:

2.5.1 Check_scorte_malto

Questo trigger verifica che la **quantità di malto utilizzata per l'ammostamento** sia disponibile tra le **scorte in magazzino**. Il trigger viene attivato prima di inserire una nuova riga nella tabella "**Ammostamento**" e controlla se la **quantità di malto** specificata nella nuova riga è **maggiore** della **quantità disponibile in magazzino**. In caso negativo, viene sollevata un'eccezione e visualizzato un messaggio di errore.

```
01 | CREATE OR REPLACE TRIGGER Check_scorte_malto
02 |         BEFORE INSERT ON Ammostamento
03 |         FOR EACH ROW
04 |         DECLARE
05 |             notEnoughMalt EXCEPTION;
06 |             BUYED NUMBER;
07 |             USED NUMBER;
08 |         BEGIN
09 |             SELECT totacq INTO BUYED
10 |             FROM (
11 |                     SELECT L.codProdotto, L.gs1Fornitore,SUM(↩
   |     quantitaAcquistata) totacq
12 |                     FROM LottoMateriaPrima L
13 |                     WHERE L.codProdotto = :new.codProdottomalto AND L.↩
   |     gs1Fornitore = :new.gs1Fornitore
14 |                     GROUP BY L.codProdotto, L.gs1Fornitore
15 |             );
16 |
17 |             SELECT totused INTO USED
18 |             FROM (
19 |                     SELECT A.codProdottoMalto, A.gs1Fornitore,SUM(↩
   |     quantitaMalto) totused
20 |                     FROM Ammostamento A
21 |                     WHERE A.codProdottomalto = :new.codProdottomalto AND↩
   |      A.gs1Fornitore = :new.gs1Fornitore
22 |                     GROUP BY A.codProdottoMalto, A.gs1Fornitore
23 |             );
24 |
25 |
26 |             IF USED IS NOT NULL AND BUYED-USED < :new.quantitaMalto
27 |                 THEN RAISE notEnoughMalt;
28 |             END IF;
29 |         EXCEPTION
30 |             WHEN NO_DATA_FOUND THEN
31 |             IF :new.quantitaMalto > BUYED THEN RAISE_APPLICATION_ERROR ↩
   |     (-20108,'Malto in scorta insufficiente');
32 |             END IF;
```

```
33 |             WHEN notEnoughMalt THEN  RAISE_APPLICATION_ERROR (-20108,'↩
   |    Malto in scorta insufficiente');
34 |     END;
```

2.5.2 Check_scorte_luppolo

Il secondo trigger funziona in modo simile al primo trigger, ma verifica la **disponibilità del luppolo** utilizzato per la produzione del **mosto dolce**.

```
01 | CREATE OR REPLACE TRIGGER Check_scorte_luppolo
02 |         BEFORE INSERT ON MostoDolce
03 |         FOR EACH ROW
04 |         DECLARE
05 |             notEnoughLupp EXCEPTION;
06 |             BUYED NUMBER;
07 |             USED NUMBER;
08 |         BEGIN
09 |             SELECT totacq INTO BUYED
10 |             FROM (
11 |                     SELECT L.codProdotto, L.gs1Fornitore,SUM(↩
   |    quantitaAcquistata) totacq
12 |                     FROM LottoMateriaPrima L
13 |                     WHERE L.codProdotto = :new.codProdLuppUsato AND L.↩
   |    gs1Fornitore = :new.gs1Fornitore
14 |                     GROUP BY L.codProdotto, L.gs1Fornitore
15 |             );
16 |
17 |
18 |             SELECT totused INTO USED
19 |             FROM (
20 |                     SELECT MD.codProdLuppUsato, MD.gs1Fornitore,SUM(↩
   |    quantitaLuppUsato) totused
21 |                     FROM MostoDolce MD
22 |                     WHERE MD.codProdLuppUsato = :new.codProdLuppUsato ↩
   |    AND MD.gs1Fornitore = :new.gs1Fornitore
23 |                     GROUP BY MD.codProdLuppUsato, MD.gs1Fornitore
24 |             );
25 |
26 |
27 |             IF BUYED-USED < :new.quantitaLuppUsato
28 |                 THEN RAISE notEnoughLupp;
29 |             END IF;
30 |         EXCEPTION
31 |             WHEN NO_DATA_FOUND THEN
32 |             IF :new.quantitaLuppUsato > BUYED THEN ↩
   |    RAISE_APPLICATION_ERROR (-20107,'Luppolo in scorta insufficiente');
33 |             END IF;
34 |             WHEN notEnoughLupp THEN  RAISE_APPLICATION_ERROR (-20107,'↩
   |    Luppolo in scorta insufficiente');
35 |     END;
```

2.5.3 Check_scorte_lievito

Il terzo trigger verifica che la **quantità di lievito** utilizzato per la **fermentazione** sia disponibile tra le **scorte in magazzino**. Funziona in modo simile ai primi due trigger.

```
01 | CREATE OR REPLACE TRIGGER Check_scorte_lievito
02 |         BEFORE INSERT ON Fermentazione
03 |         FOR EACH ROW
04 |         DECLARE
05 |             notEnoughLiev EXCEPTION;
06 |              BUYED NUMBER;
07 |             USED NUMBER;
08 |         BEGIN
09 |             SELECT totacq INTO BUYED
10 |             FROM (
11 |                     SELECT L.codProdotto, L.gs1Fornitore,SUM(↩
    quantitaAcquistata) totacq
12 |                     FROM LottoMateriaPrima L
13 |                     WHERE L.codProdotto = :new.codProdLievUsato AND L.↩
    gs1Fornitore = :new.gs1Fornit
14 |                     GROUP BY L.codProdotto, L.gs1Fornitore
15 |             );
16 |
17 |
18 |             SELECT totused INTO USED
19 |             FROM (
20 |                     SELECT F.codProdLievUsato, F.gs1Fornit,SUM(↩
    quantitaLievUsato) totused
21 |                     FROM Fermentazione F
22 |                     WHERE F.codProdLievUsato = :new.codProdLievUsato AND↩
     F.gs1Fornit = :new.gs1Fornit
23 |                     GROUP BY F.codProdLievUsato, F.gs1Fornit
24 |             );
25 |
26 |             IF USED IS NOT NULL AND :new.codProdLievUsato > BUYED-USED
27 |                 THEN RAISE notEnoughLiev;
28 |             END IF;
29 |             EXCEPTION
30 |             WHEN NO_DATA_FOUND THEN
31 |             IF :new.quantitaLievUsato > BUYED THEN ↩
    RAISE_APPLICATION_ERROR (-20004,'Lievito in scorta insufficiente');
32 |             END IF;
33 |                 WHEN notEnoughLiev THEN  RAISE_APPLICATION_ERROR ↩
    (-20004,'Lievito in scorta insufficiente');
34 |     END;
```

2.5.4 Check_lavorazione

Questo trigger sta controllando la **quantità di malto** e **acqua** utilizzati per l'**ammostamento** durante l'inserimento nella tabella chiamata "Ammostamento". Prima di inserire i dati, il trigger verifica che la **quantità di acqua non superi la capacità di lavorazione del bol-**

litore specificato dall'**idBollitore**. Se la quantità di acqua supera la capacità di lavorazione, viene sollevata una eccezione "***notEnoughCapacity***" e viene visualizzato un messaggio di errore "***Capacità Insufficiente***".

```
CREATE OR REPLACE TRIGGER Check_lavorazione
        BEFORE INSERT ON Ammostamento
        FOR EACH ROW
        DECLARE
            notEnoughCapacity EXCEPTION;
            exc EXCEPTION;
            ContainerCap NUMBER;
        BEGIN
            SELECT capacitaLavorazione INTO ContainerCap
            FROM Contenitore C
            WHERE C.idContenitore = :new.idBollitore;
            IF (:new.quantitaAcqua > ContainerCap )
                THEN RAISE notEnoughCapacity;
            END IF;
        EXCEPTION
                WHEN notEnoughCapacity THEN  RAISE_APPLICATION_ERROR ↩
    (-20006,'Capacit insufficiente');
     END;
```

2.5.5 Check_Bollitore

Questo trigger controlla che, durante un inserimento di dati nella tabella ammostamento, il contenitore utilizzato sia di tipo "**Bollitore**". Se il tipo del contenitore è "Fermentatore", viene sollevata un'eccezione "***wrongContainer***". Questa eccezione viene gestita con il comando "***RAISE_APPLICATION_ERROR***", che stampa un messaggio di errore con codice 20007 e il messaggio "***Wrong type container***".

```
CREATE OR REPLACE TRIGGER Check_Bollitore
        BEFORE INSERT ON Ammostamento
        FOR EACH ROW
        DECLARE
            tipo CHAR(20);
            wrongContainer EXCEPTION;
   BEGIN
        Select tipoContenitore INTO tipo
        FROM Contenitore
        WHERE idContenitore = :new.idBollitore;
        IF (tipo = 'Fermentatore')
            THEN RAISE wrongContainer;
        END IF;
        EXCEPTION
            WHEN wrongContainer THEN RAISE_APPLICATION_ERROR(-20007,'↩
    Wrong type container');
   END;
```

2.5.6 Check_Fermentatore

Il trigger in questione per la tabella "Fermentazione" (Check_Fermentatore) funziona in modo analogo, ma in questo caso controlla che il contenitore utilizzato sia di tipo "**Fermentatore**". Anche in questo caso, se il tipo non è corretto, viene sollevata un'eccezione "***wrongContainer***" che viene gestita con il comando "***RAISE_APPLICATION_ERROR***", stampa un messaggio di errore con codice 20009 e il messaggio "***Wrong type container***".

```
01 | CREATE OR REPLACE TRIGGER Check_Fermentatore
02 |         BEFORE INSERT ON Fermentazione
03 |         FOR EACH ROW
04 |         DECLARE
05 |             tipo CHAR(20);
06 |             wrongContainer EXCEPTION;
07 |    BEGIN
08 |         Select tipoContenitore INTO tipo
09 |         FROM Contenitore
10 |         WHERE idContenitore = :new.idFermentatore;
11 |         IF (tipo = 'Bollitore')
12 |             THEN RAISE wrongContainer;
13 |         END IF;
14 |         EXCEPTION
15 |             WHEN wrongContainer THEN RAISE_APPLICATION_ERROR(-20009,'↩
    |     Wrong type container');
16 |    END;
```

2.5.7 CheckVendita

Il trigger, viene attivato prima di un' inserimento sulla tabella "**BirraVenduta**". Verifica che il **numero di fusti venduti sia disponibile in magazzino**. Viene selezionato il **totale di fusti prodotti e venduti** per un **determinato lotto di birra** e viene **confrontato con il numero di fusti venduti**. Se il totale di fusti prodotti è inferiore al totale di fusti venduti, viene sollevata un'eccezione "***notEnoughtFusti***". In caso di eccezione, viene sollevato un messaggio di errore "***Troppi pochi fusti***".

```
01 | CREATE OR REPLACE TRIGGER CheckVendita
02 |         BEFORE INSERT ON BirraVenduta
03 |         FOR EACH ROW
04 |         DECLARE
05 |         notEnoughtFusti EXCEPTION;
06 |         totpr NUMBER;
07 |         totsell NUMBER;
08 |         BEGIN
09 |         SELECT totprod INTO totpr
10 |             FROM (
11 |                     SELECT CODLOTTO,SUM(numFustiProdotti) totprod
12 |                     FROM BirraProdotta
13 |                     WHERE CODLOTTO = :new.CODLOTTO
14 |                     GROUP BY CODLOTTO
15 |             );
16 |         SELECT totprod INTO totsell
17 |             FROM (
```

```
18 ||                    SELECT CODLOTTO,CODFATTURA,SUM(NUMFUSTI) totprod
19 ||                    FROM BIRRAVENDUTA
20 ||                    WHERE  CODLOTTO = :new.codLotto AND CODFATTURA = :↩
   ||    new.CodFattura
21 ||                    GROUP BY CodLotto,CODFATTURA
22 ||            );
23 ||
24 ||        IF totpr-totsell < :new.numFusti THEN RAISE notEnoughtFusti;
25 ||        END IF;
26 ||        EXCEPTION
27 ||        WHEN NO_DATA_FOUND THEN
28 ||        IF :new.numFusti > 100 THEN
29 ||            RAISE_APPLICATION_ERROR (-20108,'Fusti in scorta ↩
   ||    insufficienti');
30 ||        END IF;
31 ||        WHEN notEnoughtFusti THEN RAISE_APPLICATION_ERROR (-20843,'↩
   ||    Troppi pochi fusti');
32 ||        END;
```

2.5.8 CheckDisponibilitaFermentatore

Il trigger, viene attivato prima di un' inserimento sulla tabella "**Fermentazione**". Verifica che il **fermentatore specificato non sia già occupato da un'altra fermentazione**. Viene eseguita una query per verificare se ci sono fermentazioni attive con lo stesso identificatore di fermentatore. Se il fermentatore è già occupato, viene sollevata un'eccezione "***FermentatoreOccupato***". In caso di eccezione, viene sollevato un messaggio di errore "***Fermentatore Occupato***".

```
01 || CREATE OR REPLACE TRIGGER CheckDisponibilitaFermentatore
02 ||        BEFORE INSERT ON Fermentazione
03 ||        FOR EACH ROW
04 ||            DECLARE
05 ||            FermentatoreOccupato EXCEPTION;
06 ||            occupato NUMBER;
07 ||        BEGIN
08 ||        SELECT COUNT(*)
09 ||        INTO occupato
10 ||        FROM FERMENTAZIONE
11 ||        WHERE idFermentatore=:new.idFermentatore AND dataFineF IS NULL;
12 ||
13 ||        IF (occupato > 0) THEN RAISE FermentatoreOccupato;
14 ||        END IF;
15 ||        EXCEPTION
16 ||        WHEN FermentatoreOccupato THEN RAISE_APPLICATION_ERROR(-20029,'↩
   ||    Fermentatore Occupato');
17 ||     END;
```

2.5.9 Check_IsMalt

Questo trigger viene eseguito prima di un'inserzione nella tabella **AMMOSTAMENTO**. Il suo compito è controllare che la materia prima inserita sia un **malto**. Per fare questo,

effettua una query sulla tabella **LottoMateriaPrima** e **MateriaPrima** per recuperare il tipo di materia prima, e poi confronta il tipo con "**Malto**". Se il tipo non è "Malto", viene sollevata un'eccezione e mostrato un messaggio di errore personalizzato.

```
01 | CREATE OR REPLACE TRIGGER Check_IsMalt
02 |         BEFORE INSERT ON AMMOSTAMENTO
03 |         FOR EACH ROW
04 |         DECLARE
05 |             tipoMateriaPrima CHAR(20);
06 |             wrongMateriaPrima EXCEPTION;
07 |        BEGIN
08 |             Select MP.tipo INTO tipoMateriaPrima
09 |             FROM LottoMateriaPrima LMP JOIN MateriaPrima MP ON LMP.↩
    nomeMateriaPrima=MP.nomeMatPrim
10 |             WHERE LMP.codProdotto = :new.codProdottoMalto AND LMP.↩
    gs1Fornitore = :new.gs1Fornitore;
11 |             IF (tipoMateriaPrima <> 'Malto')
12 |                 THEN RAISE wrongMateriaPrima;
13 |             END IF;
14 |         EXCEPTION
15 |             WHEN wrongMateriaPrima THEN RAISE_APPLICATION_ERROR(-20100,'↩
    La materia prima non  malto');
16 |    END;
```

2.5.10 Check_IsHop

Questo trigger funziona allo stesso modo del trigger precedente, ma **controlla che la materia prima inserita nella tabella MostoDolce** sia un **luppolo**.

```
01 | CREATE OR REPLACE TRIGGER Check_IsHop
02 |         BEFORE INSERT ON MostoDolce
03 |         FOR EACH ROW
04 |         DECLARE
05 |             tipoMateriaPrima CHAR(20);
06 |             wrongMateriaPrima2 EXCEPTION;
07 |        BEGIN
08 |             Select MP.tipo INTO tipoMateriaPrima
09 |             FROM LottoMateriaPrima LMP JOIN MateriaPrima MP ON LMP.↩
    nomeMateriaPrima=MP.nomeMatPrim
10 |             WHERE LMP.codProdotto = :new.codProdLuppUsato AND LMP.↩
    gs1Fornitore = :new.gs1Fornitore;
11 |             IF (tipoMateriaPrima <> 'Luppolo')
12 |                 THEN RAISE wrongMateriaPrima2;
13 |             END IF;
14 |         EXCEPTION
15 |             WHEN wrongMateriaPrima2 THEN RAISE_APPLICATION_ERROR(-20101,↩
    'La materia prima non  luppolo');
16 |    END;
```

2.5.11 Check_IsYeast

Anche questo trigger è simile agli altri due, ma controlla che **la materia prima inserita** nella tabella **Fermentazione** sia un **lievito**.

```
01 | CREATE OR REPLACE TRIGGER Check_IsYeast
02 |         BEFORE INSERT ON Fermentazione
03 |         FOR EACH ROW
04 |         DECLARE
05 |             tipoMateriaPrima CHAR(20);
06 |             wrongMateriaPrima3 EXCEPTION;
07 |        BEGIN
08 |             Select MP.tipo INTO tipoMateriaPrima
09 |             FROM LottoMateriaPrima LMP JOIN MateriaPrima MP ON LMP.↩
    nomeMateriaPrima=MP.nomeMatPrim
10 |             WHERE LMP.codProdotto = :new.codProdLievUsato AND LMP.↩
    gs1Fornitore = :new.gs1Fornit;
11 |             IF (tipoMateriaPrima <> 'Lievito')
12 |                 THEN RAISE wrongMateriaPrima3;
13 |             END IF;
14 |          EXCEPTION
15 |             WHEN wrongMateriaPrima3 THEN RAISE_APPLICATION_ERROR(-20102,↩
    'La materia prima non  lievito');
16 |    END;
```

2.6 Procedure

Le operazioni che possono essere effettuate dagli utenti sono state implementate tramite **procedure** per cercare di gestire la base dati nella maniera più efficiente possibile.

2.6.1 Aumento_stipendio

Questa procedura "***Aumento_stipendio***" incrementa lo stipendio del 20% al **mastro birraio responsabile del tipo di birra più venduto**. Questo viene fatto utilizzando una query all'interno della procedura. La procedura è divisa in **due parti principali**:

- La prima parte seleziona l'SSN del mastro birraio responsabile del tipo di birra più venduto utilizzando le tabelle "`TIPOBIRRA`", "`BIRRAPRODOTTA`" e "`BIRRAVENDUTA`".
- La seconda parte effettua **l'aggiornamento dello stipendio del mastro birraio selezionato**, moltiplicando il suo stipendio corrente per $1,20$.

La procedura utilizza la clausola INNER JOIN per unire le informazioni delle tabelle "`BIRRAPRODOTTA`" e "`BIRRAVENDUTA`" in base alla colonna "codLotto" e utilizza la funzione **SUM** per **calcolare il totale delle vendite di ciascun tipo di birra**. Quindi utilizza la clausola SELECT con la funzione MAX per selezionare il **tipo di birra con le vendite totali più alte**. Infine, utilizza l'aggiornamento per **modificare lo stipendio del mastro birraio** responsabile del tipo di birra più venduto.

```
01 |     CREATE OR REPLACE PROCEDURE Aumento_stipendio IS
02 |     BEGIN
03 |
04 |     UPDATE MASTROBIRRAIO SET STIPENDIO = STIPENDIO * 1.20
05 |     WHERE SSN IN (SELECT ssnSupervisore
06 |                   FROM  TIPOBIRRA TB
07 |                   WHERE GTIN =(SELECT GTIN
08 |                                FROM (SELECT GTIN, SUM(NUMFUSTI) AS ↩
    TOTVENDITA
09 |                                      FROM BIRRAPRODOTTA BP JOIN ↩
    BIRRAVENDUTA BV ON BP.codLotto = BV.codLotto
10 |                                      GROUP BY BP.GTIN )
11 |                                WHERE TOTVENDITA = (SELECT MAX(↩
    TOTVENDITA)
12 |                                                    FROM (SELECT GTIN, ↩
    SUM(NUMFUSTI) AS TOTVENDITA
13 |                                                          FROM ↩
    BIRRAPRODOTTA BP JOIN BIRRAVENDUTA BV ON BP.codLotto = BV.codLotto
14 |                                                          GROUP BY BP.↩
    GTIN))));
15 |     END;
```

2.6.2 ScontoFusto

Questa procedura "ScontoFusto" trova **la birra che viene venduta meno**, ovvero la birra con il numero di fusti venduti più basso, e applica uno **sconto** del 25% sul prezzo del fusto della birra. La procedura effettua questa operazione attraverso le seguenti fasi:

- Utilizza una query interna che unisce le tabelle `BIRRAPRODOTTA` e `BIRRAVENDUTA` sul codice lotto e calcola la somma dei fusti venduti per ogni tipo di birra (GTIN).
- Utilizza una query esterna che seleziona il GTIN della birra con il numero di fusti venduti più basso.
- Esegue un'operazione di aggiornamento sulla tabella `TIPOBIRRA`, moltiplicando il prezzoFusto per 0.80 (80% del prezzo originale) per ogni birra con il GTIN trovato nella fase 2.

In questo modo, viene applicato uno sconto del 25% sul prossimo acquisto del fusto della birra venduta meno.

```
01 |     CREATE OR REPLACE PROCEDURE ScontoFusto IS
02 |     BEGIN
03 |
04 |     UPDATE TIPOBIRRA SET prezzoFusto = prezzoFusto * 0.80 WHERE GTIN IN ↩
   |     (SELECT GTIN
05 |     FROM    TIPOBIRRA TB
06 |     WHERE GTIN =    (SELECT GTIN
07 |                         FROM ( SELECT GTIN, SUM(NUMFUSTI) AS TOTVENDITA
08 |                         FROM BIRRAPRODOTTA BP JOIN BIRRAVENDUTA BV ON BP↩
   |     .codLotto = BV.codLotto
09 |                         GROUP BY BP.GTIN )
10 |                 WHERE   TOTVENDITA = (  SELECT MIN(TOTVENDITA)
11 |                                         FROM (  SELECT GTIN, SUM(NUMFUSTI) ↩
   |     AS TOTVENDITA
12 |                                                 FROM BIRRAPRODOTTA BP JOIN ↩
   |     BIRRAVENDUTA BV ON BP.codLotto = BV.codLotto
13 |                                                 GROUP BY BP.GTIN))));
14 |
15 |      END;
```

2.6.3 IncrementaPrezzo

Questa procedura **aumenta il prezzo delle birre che utilizzano malto inglese** che è stato **pagato di più** rispetto ad altri tipi di malto. La procedura esegue i seguenti passaggi:

- Calcola la media del prezzo di acquisto del malto inglese che proviene dal Regno Unito.
- Seleziona il malto inglese che ha il prezzo medio di acquisto più alto.
- Utilizzando il codice GTIN (Global Trade Item Number) come identificatore univoco, seleziona le birre che utilizzano il malto inglese selezionato.
- Aumenta del 20% il prezzo delle birre selezionate.

```
01 |     CREATE OR REPLACE PROCEDURE IncrementaPrezzo IS
02 |      BEGIN
03 |
04 |      UPDATE TIPOBIRRA SET prezzoFusto = prezzoFusto * 1.20 WHERE GTIN IN↩
   |      (
05 |           SELECT GTIN
06 |           FROM TIPOBIRRA
07 |           WHERE NOMEMALTO IN(SELECT nomeMateriaPrima
```

```
08 |          FROM (SELECT L.nomeMateriaPrima, AVG(prezzoAcquisto/↩
   |    quantitaAcquistata) AS Media
09 |                FROM LottoMateriaPrima L JOIN MateriaPrima M ON L.↩
   |    nomeMateriaPrima = M.NOMEMATPRIM
10 |                WHERE M.provenienza = 'Regno Unito' AND M.Tipo = 'Malto'
11 |                GROUP BY L.nomeMateriaPrima)
12 |          WHERE Media = (SELECT MAX(Media)
13 |          FROM (SELECT L.nomeMateriaPrima, AVG(prezzoAcquisto/↩
   |    quantitaAcquistata) AS Media
14 |                FROM LottoMateriaPrima L JOIN MateriaPrima M ON L.↩
   |    nomeMateriaPrima = M.NOMEMATPRIM
15 |                WHERE M.provenienza = 'Regno Unito' AND M.Tipo = 'Malto'
16 |                GROUP BY L.nomeMateriaPrima))));
17 |
18 |      END;
```

2.6.4 DecrementaPrezzo

Questa procedura, denominata "DecrementaPrezzo", ha l'obiettivo di **diminuire del** 30% **il prezzo** dei fusti di birre fatte con **malti che hanno più di 100 unità in stock**. La procedura utilizza la tabella `TIPOBIRRA` e tre viste create in precedenza. Ecco una descrizione dettagliata della procedura:

- La prima riga stabilisce che la procedura è una sostituzione o creazione (se non esiste) di una procedura con lo stesso nome.
- La riga successiva stabilisce l'inizio della procedura.
- La riga successiva esegue un' aggiornamento alla tabella TIPOBIRRA, in cui il prezzoFusto viene moltiplicato per $0,70$ (ridotto del 30%) per tutti i record in cui il GTIN corrisponde ai valori presenti nella subquery.
- La subquery seleziona il GTIN dalla tabella TIPOBIRRA in cui il nomeMateriaPrima corrisponde ai valori restituiti dalla subquery ancor più interna.
- La subquery interna più a sinistra restituisce il nomeMateriaPrima e il suo stock rimanente per tutti i malti provenienti dal Regno Unito e di tipo "Malto". Questo viene calcolato sottraendo la quantità utilizzata (`TOTUSED`) dalla quantità acquistata (`TOTACQ`) per ogni lotto di malto.

Le **tre viste** utilizzate dalla procedura (Acquisti, MaltiUsati e MaltiRimanenti) forniscono **informazioni sull'acquisto e l'utilizzo dei malti**. La vista **Acquisti** fornisce la somma delle quantità acquistate per ogni lotto di malto. La vista **MaltiUsati** fornisce la somma della quantità di malto utilizzata per ogni lotto di malto. Infine, la vista **MaltiRimanenti** utilizza le informazioni fornite dalle viste Acquisti e MaltiUsati per calcolare le rimanenze di ogni lotto di malto.

```
01 |     CREATE OR REPLACE PROCEDURE DecrementaPrezzo IS
02 |      BEGIN
03 |
04 |      UPDATE TIPOBIRRA SET prezzoFusto = prezzoFusto * 0.70 WHERE GTIN IN↩
   |      (
05 |           SELECT GTIN
06 |           FROM TIPOBIRRA
```

```
07 ||           WHERE NOMEMALTO IN(
08 ||                 SELECT nomeMateriaPrima
09 ||                 FROM ( SELECT L.nomeMateriaPrima,RIMANENZE
10 ||                       FROM LottoMateriaPrima L JOIN MaltiRimanenti A on ↩
   ||    L.CODLOTTO = A.CODLOTTO AND L.CODPRODOTTO = A.CODPRODOTTO AND L.↩
   ||    gs1Fornitore = A.gs1Fornitore)
11 ||                 WHERE RIMANENZE > 100));
12 ||     END;
13 ||
14 || /
```

2.7 Viste

Vista la mole di prodotti e, in generale, di informazioni presenti all'interno di questo database, per rendere più immediata la visualizzazione di alcuni dati sono state pensate alcune viste utilizzabili dai diversi utenti.

2.7.1 Acquisti

Questa vista crea una vista che raggruppa i dati della tabella LottoMateriaPrima (L) in base a codLotto, codProdotto e gs1Fornitore, calcolando **la somma della quantità acquistata** (totacq) per ogni gruppo. Questa vista fornirà una panoramica aggregata degli acquisti effettuati per ogni prodotto, identificato dalla combinazione di codLotto, codProdotto e gs1Fornitore.

```
01 |     CREATE VIEW Acquisti AS (SELECT *
02 |                     FROM (
03 |                          SELECT L.CODLOTTO,L.codProdotto, L.gs1Fornitore↩
   |     ,SUM(quantitaAcquistata) totacq
04 |                          FROM LottoMateriaPrima L
05 |                          GROUP BY L.CODLOTTO,L.codProdotto, L.↩
   |     gs1Fornitore
06 |                     ));
```

2.7.2 MaltiUsati

Questa vista crea una vista che raggruppa i dati della tabella Ammostamento (A) in base a codLotto, codProdottoMalto e gs1Fornitore, calcolando **la somma della quantità di malto usata** (totused) per ogni gruppo. Questa vista fornirà una panoramica aggregata dell'utilizzo di malto per ogni prodotto, identificato dalla combinazione di codLotto, codProdottoMalto e gs1Fornitore.

```
01 |     CREATE VIEW MaltiUsati AS (SELECT *
02 |                     FROM (
03 |                          SELECT A.CODLOTTO,A.codProdottoMalto, A.↩
   |     gs1Fornitore,SUM(quantitaMalto) totused
04 |                          FROM Ammostamento A
05 |                          GROUP BY A.CODLOTTO,A.codProdottoMalto, A.↩
   |     gs1Fornitore
06 |                     ));
```

2.7.3 MaltiRimanenti

Questa vista crea una vista che unisce le viste Acquisti e MaltiUsati sulla base della combinazione di codLotto, codProdotto e gs1Fornitore. La vista **calcola la differenza tra la quantità totale acquistata** (totacq) e **la quantità totale usata** (totused) per ogni prodotto, identificato dalla combinazione di codLotto, codProdotto e gs1Fornitore, come "rimanenze". Questa vista fornirà una panoramica aggregata delle rimanenze di malto per ogni prodotto.

```
01 |     CREATE VIEW MaltiRimanenti AS (SELECT A.CODLOTTO,A.CODPRODOTTO,A.↩
   |     GS1FORNITORE,TOTACQ-TOTUSED AS RIMANENZE
02 |                                 FROM Acquisti A
```

```
03 |                                  JOIN MaltiUsati MU ON A.CODLOTTO = MU.↩
   |     CODLOTTO AND
04 |                                       A.CODPRODOTTO = MU.CODPRODOTTOMALTO ↩
   |     AND
05 |                                       A.GS1FORNITORE = MU.GS1FORNITORE
06 |                                  );
```