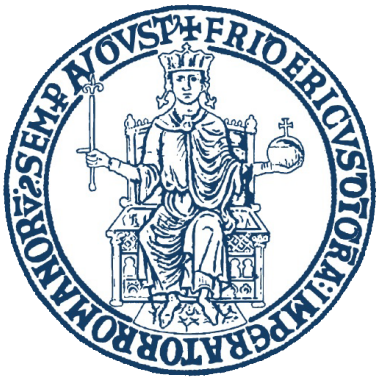


UNIVERSITÀ DEGLI STUDI
DI NAPOLI FEDERICO II

Documentazione DB: Sistema di gestione del ciclo di vita di una pagina Wiki

Febbraio 2024

Autori:
Lorenzo Trignano (N86004496)
Francesco Simone (N86004627)

Indice

1 Descrizione ed Analisi del Progetto

1.1 Introduzione

Si vuole sviluppare un sistema di database relazionale per la gestione del ciclo di vita di una pagina Wiki. In questa fase andremo ad analizzare i requisiti funzionali e non funzionali che dovrà possedere il sistema.
In primo luogo bisogna individuare le varie entità che compongono la nostra base dati e le relazioni che hanno tra loro. Andremo quindi ad esaminare la documentazione fornita per la realizzazione del sistema.

1.2 Analisi dei Requisiti

"Una pagina di una wiki ha un titolo e un testo. Ogni pagina è creata da un determinato autore. Il testo è composto di una sequenza di frasi. Il sistema mantiene traccia anche del giorno e ora nel quale la pagina è stata creata. La pagina può contenere anche dei collegamenti. Ogni collegamento è caratterizzato da una frase da cui scaturisce il collegamento e da un'altra pagina destinazione del collegamento"

Consideriamo di dover gestire diverse pagine di una Wiki. Abbiamo quindi l'entità "Pagina", la quale è composta da un titolo, un testo, la data di creazione, e un autore.
Inoltre sappiamo che un testo è composto da una sequenza di frasi; è opportuno dunque considerare l'esistenza dell'entità "Frase" piuttosto che quella dell'entità "Testo". Una frase in un testo ha diverse caratteristiche da dover memorizzare: la riga su cui si estende, l'ordine di posizionamento su una riga (se si tratta della prima/seconda/terza frase e così via), e le parole che compongono la frase, quindi il suo contenuto.
Individuiamo anche la presenza di frasi che possono scaturire un collegamento con un'altra pagina del sistema: l'entità "Collegamento". Costruiamo dunque una specializzazione, dove "Frase" è l'entità padre, e "Collegamento" è l'entità figlio. Abbiamo inoltre strutturato il nostro sistema wiki per temi. Definiamo l'entità "Tema", infatti ogni pagina verrà memorizzata per tema a cui fa riferimento. Abbiamo quindi le seguenti associazioni:

- L'entità "Pagina" *contiene* "Frase".
- L'entità "Collegamento" *fa riferimento* a "Pagina".
- L'entità "Pagina" *è archiviata* in "Tema".

"Il testo può essere modificato da un altro utente del sistema, che seleziona una o più delle frasi, scrive la sua versione alternativa (modifica) e prova a proporla. La modifica proposta verrà notificata all'autore del testo originale la prossima volta che utilizzerà il sistema. L'autore potrà vedere la sua versione originale e la modifica proposta. Egli potrà accettare la modifica (in quel caso la pagina

originale diventerà ora quella con la modifica apportata), rifiutare la modifica (la pagina originale rimarrà invariata). La modifica proposta dall'autore verrà memorizzata nel sistema e diventerà subito parte della versione corrente del testo. Il sistema mantiene memoria delle modifiche proposte e anche delle decisioni dell'autore (accettazione o rifiuto)."

Comprendiamo che è necessaria l'introduzione di un'entità "Utente", la quale verrà analizzata successivamente, e soprattutto dell'entità "Operazione".
Un operazione può essere effettuata direttamente dall'autore della pagina oppure da un utente generico, il quale può proporre una determinata operazione sul testo. Specifichiamo quindi un attributo "proposta" che ci indicherà se l'operazione in analisi è una proposta o meno. Ora analizziamo come potrebbe essere strutturata una certa operazione su un testo. Un utente potrebbe inserire una nuova frase, modificare una già esistente, oppure rimuoverla. Inoltre, va specificata la posizione nel testo (riga e ordine) della frase inserita, modificata o rimossa. Capiamo quindi che un operazione può essere di inserimento, modifica e cancellazione. Costruiamo dunque una generalizzazione in cui l'entità operazione è quella padre, e le tre entità: "Inserimento", "Modifica", "Cancellazione" sono quelle figlio. Tali tre entità mostreranno il contenuto della frase che ha subito l'operazione nel seguente modo:

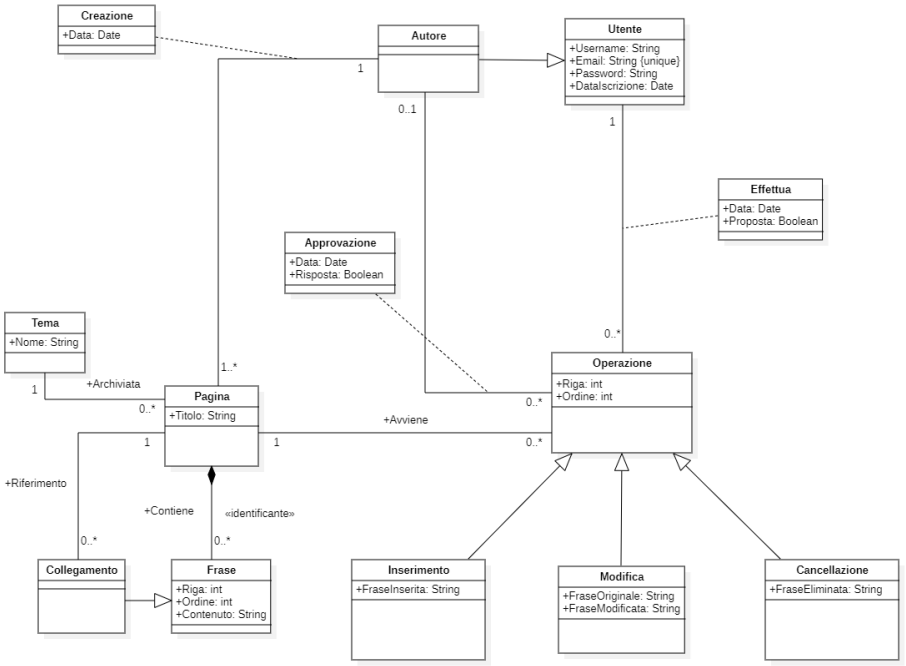
- "Inserimento": conterrà il contenuto della frase inserita.
- "Modifica": conterrà il contenuto della frase prima e dopo la modifica.
- "Cancellazione": conterrà il contenuto della frase rimossa.

Analizziamo le caratteristiche che ha un utente nel nostro sistema:
"Gli utenti generici del sistema potranno cercare una pagina e il sistema mostrerà la versione corrente del testo e i collegamenti. Gli autori dovranno prima autenticarsi fornendo la propria login e password. Tutti gli autori potranno vedere tutta la storia di tutti i testi dei quali sono autori e di tutti quelli nei quali hanno proposto una modifica."

Assumiamo che un qualsiasi utente per poter accedere al sistema dovrà prima autenticarsi, indicando username e password. Nel momento di una creazione di un utente, sarà necessario anche indicare un email personale, e verrà memorizzata la data di iscrizione. Un utente può essere anche un autore di una pagina. Dunque definiamo una specializzazione tra l'entità "Utente" (padre) e l'entità "Autore" (figlio). Nel momento in cui un autore scrive una pagina, verrà memorizzata la data in cui è avvenuta la creazione della suddetta pagina.
Ora definiamo le varie associazioni che vi sono tra queste entità appena descritte:

- L'entità "Autore" *crea* "Pagina".
- L'entità "Operazione" *avviene* su "Pagina".
- L'entità "Utente" *effettua* "Operazione".
- L'entità "Autore" *approva* "Operazione".

1.3 Schema concettuale



1.4 Dizionario delle entità e delle associazioni

1.4.1 Dizionario delle entità

Entità	Descrizione	Attributi
Utente	Un utente generico del sistema Wiki.	Username: Il soprannome che identificherà l'utente generico. Email: La posta elettronica dell'utente. Password: La password dell'utente. Data Iscrizione: La data in cui l'utente si è iscritto al sistema
Autore	Un utente che è autore di almeno una pagina.	Stessi attributi di Utente.
Pagina	Una pagina generica del sistema Wiki.	Titolo: Il titolo della pagina.
Tema	Il tema a cui appartiene una pagina del sistema.	Nome: Il nome del tema. Es: 'Storia', 'Geografia', 'Cucina'.
Frase	Una generica frase presente nel testo di una pagina.	Riga: La riga su cui si estende la frase. Ordine: L'ordine di posizionamento di una frase su una determinata riga. Contenuto: Le parole che compongono la frase.
Collegamento	Una frase che possiede un collegamento con un'altra pagina del sistema.	Stessi attributi di Frase.
Operazione	Un'operazione generica eseguita da un utente del sistema su una pagina.	Riga: La riga su cui si estende la frase coinvolta nell'operazione. Ordine: Ordine di posizionamento della frase coinvolta nell'operazione.
Inserimento	Un'operazione di inserimento di una frase eseguita da un utente del sistema su una pagina.	Stessi attributi di Operazione. Frase Inserita: Il contenuto della frase che è stata inserita nella pagina.
Modifica	Un'operazione di modifica di una frase eseguita da un utente del sistema su una pagina.	Stessi attributi di Operazione. Frase Originale: Il contenuto della frase prima della modifica. Frase Modificata: Il contenuto della frase dopo la modifica.
Cancellazione	Un'operazione di cancellazione di una frase eseguita da un utente del sistema su una pagina.	Stessi attributi di Operazione. Frase Eliminata: Il contenuto della frase che è stata rimossa.

1.4.2 Dizionario delle associazioni

Associazione	Descrizione
Creazione	Associazione **uno-a-molti** tra "Autore" e "Pagina" Un Autore può creare da 1 a più pagine. Una pagina è creata da un solo Autore. Data: La data in cui è stata creata la pagina.
Contiene	Associazione **uno-a-molti** tra "Pagina" e "Frase" Una Pagina può contenere da zero a più frasi, una determinata Frase è contenuta in una sola pagina.
Riferimento	Associazione **uno-a-molti** tra "Collegamento" e "Pagina" Un collegamento si riferisce ad una sola pagina. Una pagina può essere riferita da più collegamenti.
Archiviata	Associazione **uno-a-molti** tra "Tema" e "Pagina" Una pagina è archiviata in un solo e determinato Tema. Un tema può avere archiviati da zero a più pagine.
Effettua	Associazione **uno-a-molti** tra "Utente" e "Operazione" Un utente può effettuare da zero a più operazioni. Un Operazione è effettuata da un solo Utente. Proposta: Se l'operazione che sta effettuando l'utente è una proposta o meno. Data: La data in cui è stata effettuata un'operazione.
Approvazione	Associazione **uno-a-molti** tra "Autore" e "Operazione" Un autore può approvare da zero a più operazioni. Un operazione è approvata da un solo Autore. Data: La data in cui è stata approvata la proposta di operazione. Risposta: La risposta da parte dell'autore che indica se ha accettato (true), rifiutato (false) o ancora in attesa (NULL) la proposta di operazione.
Avviene	Associazione **uno-a-molti** tra "Operazione" e "Pagina" Una determinata Operazione avviene su una sola Pagina. Una pagina può essere coinvolta in zero o più operazioni.

2 Ristrutturazione del modello concettuale

2.1 Introduzione

In questa fase, con lo scopo di rendere il Class Diagram idoneo per la traduzione in schemi relazionali e di migliorare l'efficienza dell'implementazione, si procede alla **ristrutturazione** dello stesso. Al termine di questo operazione il Class Diagram non conterrà alcun attributo multiplo, composto, eventuali specializzazioni o generalizzazioni e si procederà all'inserimento di Identificativi validi.

2.1.1 Analisi delle ridondanze

In questo Class Diagram non sono presenti significative ridondanze tali da essere eliminate.

2.1.2 Analisi degli identificativi

In questa fase andremo a scegliere uno o più attributi per identificare univocamente le varie entità presenti nello schema precedente, in particolare:

- Per un **Utente** è già presente un attributo *Username* che rappresenta una possibile chiave primaria.
- Per una **Pagina**, al fine di permettere agli autori di scegliere senza limitazioni il Titolo, si è preferito aggiungere un attributo *IDPagina* e renderlo chiave primaria.
- Per la **Frase** si è scelta di identificarla tramite un insieme di attributi già presenti: *Ordine* e *Riga*.
- Dato che l'entità **Operazione** non presenta alcuna chiave candidata, è stato aggiunto l'attributo *IDOperazione*.
- Per un **Tema**, al fine di costruire un indice funzionale si è deciso di aggiungere un attributo *IDTema* di rapida lettura rispetto all'attributo *Nome*.

2.1.3 Rimozione degli attributi multivalore

Non sono presenti attributi multivalore.

2.1.4 Rimozione degli attributi composti

Non sono presenti attributi composti.

2.1.5 Partizione/Accorpamento delle associazioni

Non sono state effettuate operazioni di partizione/accorpamento sui concetti espressi nel class diagram.

2.1.6 Rimozione delle gerarchie

In questo diagramma sono presenti 3 generalizzazioni e 5 relative specializzazioni. In particolare:

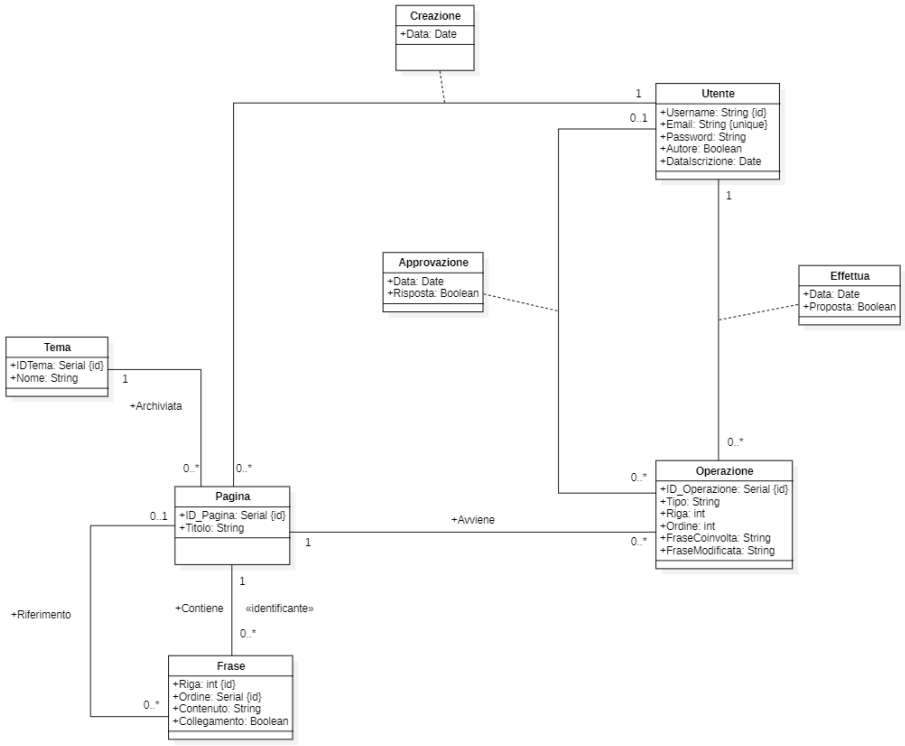
- Per quanto riguarda la generalizzazione **Operazione**, si è scelto di accorpare le entità figlie della generalizzazione nell'entità padre, ottenendo così come risultato un'entità **Operazione** avente tutti gli attributi di **Operazione**, più gli attributi delle precedenti entità **Inserimento**, **Modifica** e **Cancellazione**. Gli attributi *FraseInserita*, *FraseOriginale* e *FraseEliminata* sono stati successivamente accorpati in un unico attributo denominato *FraseCoinvolta*. È stato quindi aggiunto un nuovo attributo *Tipo* che indica il tipo di istanza di operazione nel seguente modo: 'I': inserimento, 'M': modifica, 'C': cancellazione.

- Per quanto riguarda invece la generalizzazione **Utente**, si è analogamente scelto di accorpare l'entità figlia nell'entità padre, ottenendo così come risultato un'entita **Utente** avente tutti gli attributi di **Utente**, con l'aggiunta di un nuovo attributo di tipo booleano *Autore*, con il quale è possibile verificare se un dato Utente è anche un Autore.

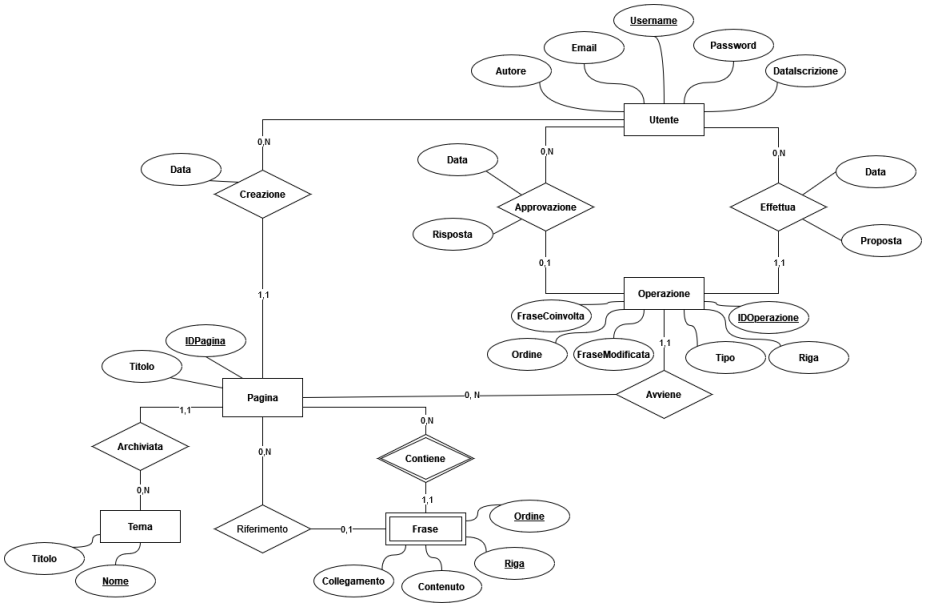
- Identico procedimento è stato applicato per la generalizzazione **Frase**, in cui abbiamo incorporato l'entità figlio **Collegamento** e abbiamo definito un attributo *Collegamento* di tipo booleano, che differenzierà le istanze di frase che possiedono un collegamento con una pagina da quelle che non lo possiedono.

2.2 Class Diagram ristrutturato

2.2.1 UML Diagram



2.2.2 ER Diagram



2.3 Dizionario delle classi

Classe	Descrizione	Attributi
Utente	Un utente generico del sistema Wiki.	Username (ID) [String]: Il soprannome che identificherà l'utente generico. Email [String]: La posta elettronica dell'utente. Password [String]: La password dell'utente. Data Iscrizione [Date]: La data in cui l'utente si è iscritto al sistema Autore [Boolean]: Attributo booleano che indica se un determinato utente è autore di almeno una pagina.
Pagina	Una pagina generica del sistema Wiki.	ID Pagina (ID) [INT]: Codice numerico progressivo che identifica univocamente una determinata pagina. Titolo [String]: Il titolo della pagina.
Frase	Una generica frase presente nel testo di una pagina.	Riga (ID) [INT]: La riga su cui si estende la frase. Ordine(ID)[INT]: L'ordine di posizionamento di una frase su una determinata riga. Contenuto [String]: Le parole che compongono la frase. Collegamento[Boolean]: Attributo che indica se una frase è un collegamento o meno.
Operazione	Un'operazione generica eseguita da un utente del sistema su una pagina.	ID Operazione (ID) [INT]: codice numerico progressivo che identifica univocamente una determinata operazione. Riga [INT]: La riga su cui si estende la frase coinvolta nell'operazione. Ordine [INT]: Ordine di posizionamento della frase coinvolta nell'operazione. Tipo [String]: Stringa composta da un solo carattere che indica il tipo di istanza di operazione: 'I': inserimento, 'M': modifica, 'C': cancellazione. Frase Coinvolta [String]: il contenuto della frase coinvolta nell'operazione. Inserimento: è la frase inserita, Modifica: è la frase prima della modifica, Cancellazione: è la frase rimossa. Frase Modificata [String]: Eventualmente, il contenuto della frase dopo la modifica.

Tema	Il tema a cui appartiene una pagina del sistema.	ID Tema (ID) [INT]: codice numerico progressivo che identifica univocamente un determinato tema del sistema. Nome [String]: Il nome del tema. Es: 'Storia', 'Geografia', 'Cucina'.

3 Traduzione al Modello Logico

3.1 Introduzione

In questo capitolo andremo ad analizzare dettagliatamente i meccanismi che ci permetteranno di passare da uno schema concettuale (già precedentemente predisposto a ristrutturazione) ad uno schema logico, scendendo ad un livello di astrazione ancora più profondo.

3.2 Schema

Utente: Username, Email, Password, DataIscrizione, Autore

Tema: IdTema, Nome

Pagina: IDPagina, Titolo, Tema, DataCreazione, UserAutore

- UserAutore → Utente(Username)
- Tema → Tema(IdTema)

Spiegazione del processo:

- **Associazione *"Creazione"* (Utente-Pagina)**: Si tratta di un'associazione uno-a-molti, in cui l'entità con cardinalità massima 1 (Pagina) ha una partecipazione totale. Pertanto, gli abbiamo assegnato una chiave esterna "UserAutore" che si riferisce al username dell'utente che ha creato la pagina.
- **Associazione *"Archiviata"* (Pagina-Tema)**: Si tratta di un'associazione uno-a-molti, in cui l'entità con cardinalità massima 1 (Pagina) ha una partecipazione totale. Pertanto, gli abbiamo assegnato una chiave esterna "Tema" che si riferisce al id progressivo che identifica univocamente un tema del sistema.

Frase: IDPagina, Riga, Ordine, Contenuto, Collegamento

- IDPagina → Pagina(IDPagina)

Spiegazione del processo:

- **Associazione *"Contiene"* (Pagina-Frase)**: Si tratta di un'associazione uno-a-molti, in cui *Frase* è un entità debole con chiave parziale (Riga, Ordine). Pertanto, gli abbiamo assegnato l'attributo chiave esterna IDPagina che fa riferimento al id della pagina che contiene la suddetta frase. Essa, assieme a riga ed ordine, saranno la nuova chiave primaria.

Collegamento: IDPagina, RigaFrase, OrdineFrase, IDPaginaCollegata

- IDPagina, RigaFrase, OrdineFrase, → Frase(IDPagina, Riga, Ordine)
- IDPaginaCollegata → Pagina(IDPagina)

Spiegazione del processo:

- **Associazione *"Riferimento"* (Frase-Pagina)**: Si tratta di un'associazione uno-a-molti, in cui *Frase* ha partecipazione parziale e cardinalità massima uguale ad 1. Pertanto abbiamo creato una nuova relazione di nome "Collegamento", la quale conterrà la chiave esterna riferita a *Frase* (IDPagina, Riga, Ordine), e la chiave esterna riferita a Pagina, la quale farà riferimento al id della pagina coinvolta nel processo di collegamento. Abbiamo definito come chiave primaria solamente la chiave esterna di *Frase*, in modo da non avere più tuple di *Collegamento* che si riferiscono ad una stessa tupla di *Frase*.

Operazione: IDOperazione, Tipo, Proposta, Riga, Ordine, FraseCoinvolta, FraseModificata, Data, IDPagina, Utente

- Utente → Utente(Username)
- IDPagina → Pagina(IDPagina)

Spiegazione del processo:

- **Associazione *"Effettua"* (Utente-Operazione)**: Si tratta di un'associazione uno-a-molti, in cui *Operazione* ha partecipazione totale e cardinalità massima uguale ad 1. Pertanto gli assegniamo un attributo chiave esterna "Utente" che farà riferimento al username di *Utente*.
- **Associazione *"Avviene"* (Operazione-Pagina)**:Si tratta di un'associazione uno-a-molti, in cui *Operazione* ha partecipazione totale e cardinalità massima uguale ad 1. Pertanto gli assegniamo un attributo chiave esterna "IDPagina" che farà riferimento al id di *Pagina*.

Approvazione: IDOperazione, Autore, Data, Risposta

- IDOperazione → Operazione(IDOperazione)
- Autore → Utente(Username)

Spiegazione del processo:

- **Associazione *"Approvazione"* (Utente-Operazione)**: Si tratta di un'associazione uno-a-molti, in cui *Operazione* ha partecipazione parziale e cardinalità massima uguale ad 1. Pertanto. abbiamo creato una nuova relazione di nome "Approvazione", che avrà la chiave esterna riferita ad *Operazione* (IDOperazione), e una chiave esterna riferita ad *Utente* (Autore). Abbiamo definito come chiave primaria solamente la chiave esterna "IDOperazione", in modo da non avere più tuple di *Approvazione* che si riferiscono ad una stessa tupla di Operazione.

4 Progettazione Fisica

In questo capitolo, ultimo del progetto, analizzeremo i meccanismi di traduzione da uno schema logico ad uno schema fisico. Saranno effettuate le definizioni delle tabelle (con i vari attributi e i loro corrispettivi tipi), le definizioni di *funzioni*, *procedure* e altre *automazioni*, i *triggers*, i *vincoli*, le *sequenze* e i *domini*.

4.1 Definizione delle tabelle

Seguono le definizioni delle tabelle estratte dal documento .sql di creazione del DataBase.

4.1.1 Definizione della Tabella Utente

```
/*
   ----------------------------
      !Table-UTENTE!
   ----------------------------
*/
CREATE TABLE UTENTE
(
   Username USERNAME˙DOMINIO,
   Email EMAIL˙DOMINIO NOT NULL,
   Password PASSWORD˙DOMINIO NOT NULL,
   DataIscrizione TIMESTAMP DEFAULT CURRENT˙TIMESTAMP,
   Autore BOOLEAN DEFAULT FALSE,

   PRIMARY KEY(Username),
   UNIQUE(Email)
);
```

4.1.2 Definizione della Tabella Tema

```
/*
   ----------------------------
      !Table-TEMA!
   ----------------------------
*/

CREATE TABLE TEMA
(
   idTema SERIAL,
   Nome VARCHAR(50),

   PRIMARY KEY(idTema),
   UNIQUE(Nome)
);
```

4.1.3 Definizione della Tabella Pagina

```
/*
   ---------------------------
      !Table-PAGINA!
   ---------------------------
*/
CREATE TABLE PAGINA
(
   ID_Pagina SERIAL,
   Titolo VARCHAR(50) NOT NULL,
   Tema SERIAL,
   DataCreazione TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   UserAutore USERNAME_DOMINIO NOT NULL,

   PRIMARY KEY(ID_Pagina),
   FOREIGN KEY(UserAutore) REFERENCES UTENTE(Username)
   ON DELETE CASCADE
   ON UPDATE CASCADE,
   FOREIGN KEY(Tema) REFERENCES TEMA(idTema)
   ON DELETE SET NULL
   ON UPDATE CASCADE
);
```

4.1.4 Definizione della Tabella Frase

```
/*
   ---------------------------
      !Table-FRASE!
   ---------------------------
*/
CREATE TABLE FRASE
(
   Riga INT,
   Ordine INT,
   ID_Pagina INT,
   Contenuto LEN_FRASE NOT NULL,
   Collegamento BOOLEAN DEFAULT FALSE,

   PRIMARY KEY(Riga, Ordine, ID_Pagina),
   FOREIGN KEY(ID_Pagina) REFERENCES PAGINA(ID_Pagina)
   ON DELETE CASCADE

);
```

4.1.5 Definizione della Tabella Collegamento

```
/*
   ---------------------------
      !Table-COLLEGAMENTO!
   ---------------------------
*/
CREATE TABLE COLLEGAMENTO
(
   RigaFrase INT,
   OrdineFrase INT,
   ID˙Pagina INT,
   ID˙PaginaCollegata INT,

   PRIMARY KEY(RigaFrase, OrdineFrase, ID˙Pagina),
   FOREIGN KEY(RigaFrase, OrdineFrase, ID˙Pagina) REFERENCES
       FRASE(Riga, Ordine, ID˙Pagina)
   ON DELETE CASCADE
   ON UPDATE CASCADE,
   FOREIGN KEY(ID˙PaginaCollegata) REFERENCES PAGINA(ID˙Pagina)
   ON DELETE CASCADE
);
```

4.1.6 Definizione della Tabella Operazione

```
/*
   ----------------------------
      !Table-OPERAZIONE!
   ----------------------------
*/

CREATE TABLE OPERAZIONE
(
   ID_Operazione SERIAL,
   Tipo TIPO_OPERAZIONE NOT NULL,
   Proposta BOOLEAN NOT NULL,
   Riga INT NOT NULL,
   Ordine INT NOT NULL,
   FraseCoinvolta LEN_FRASE NOT NULL,
   FraseModificata LEN_FRASE,
   Data TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   ID_Pagina SERIAL NOT NULL,
   Utente USERNAME_DOMINIO,

   PRIMARY KEY(ID_Operazione),
   FOREIGN KEY(ID_Pagina) REFERENCES PAGINA(ID_Pagina) ON
      DELETE CASCADE,
   FOREIGN KEY(Utente) REFERENCES UTENTE(Username) ON DELETE
      SET NULL
);
```

4.1.7 Definizione della Tabella Approvazione

```
/*
   ----------------------------
      !Table-APPROVAZIONE!
   ----------------------------
*/

CREATE TABLE APPROVAZIONE
(
   ID_Operazione SERIAL,
   Autore USERNAME_DOMINIO,
   Data TIMESTAMP,
   Risposta BOOLEAN,

   PRIMARY KEY(ID_Operazione),
   FOREIGN KEY(ID_Operazione) REFERENCES OPERAZIONE(
      ID_Operazione) ON DELETE CASCADE,
   FOREIGN KEY(Autore) REFERENCES UTENTE(Username) ON DELETE
      CASCADE
);
```

4.2 Creazione Domini

4.2.1 Dominio: USERNAME_DOMINIO

```
-- Vincolo Di Dominio: Username
CREATE DOMAIN USERNAME_DOMINIO AS VARCHAR(30)
   CHECK (VALUE ~ '^[a-z0-9_]-5,30"$');
```

4.2.2 Dominio: PASSWORD_DOMINIO

```
-- Vincolo Di Dominio : Password
CREATE DOMAIN PASSWORD_DOMINIO AS VARCHAR(40)
   CHECK (VALUE ~ '^.*(?=.*[@!#$^*%&])(?=.*[0-9])(?=.*[a-zA-Z])
       .*$'
      AND VALUE LIKE '________%');
```

4.2.3 Dominio: EMAIL_DOMINIO

```
-- Vincolo Di Dominio: Email
CREATE DOMAIN EMAIL_DOMINIO AS VARCHAR(50)
 CHECK (VALUE LIKE '%_@%_._%');
```

4.2.4 Dominio: LEN_FRASE

```
CREATE DOMAIN LEN_FRASE AS VARCHAR(500);
```

4.3 Implementazione dei Vincoli Semantici

4.3.1 Controllo Modifica

Non può esistere un operazione di modifica in cui l'attributo "fraseModificata" sia NULL.

```
ALTER TABLE OPERAZIONE
ADD CONSTRAINT controlloModifica CHECK(NOT(Tipo LIKE 'M' AND
    FraseModificata IS NULL));
```

4.3.2 Controllo Inserimento e Cancellazione

Non può esistere un operazione di cancellamento o di inserimento che abbia l'attributo "fraseModificata" NOT NULL.

```
ALTER TABLE OPERAZIONE
ADD CONSTRAINT controlloIC CHECK(NOT((Tipo LIKE 'I' OR Tipo
    LIKE 'C') AND FraseModificata IS NOT NULL));
```

4.3.3 Controllo Data e Risposta in Approvazione

Non può esiste un tupla in approvazione che abbia data NOT NULL e risposta NULL o viceversa.

```
ALTER TABLE APPROVAZIONE
ADD CONSTRAINT dataRisposta CHECK((data IS NULL AND risposta IS
     NULL) OR (data IS NOT NULL AND risposta IS NOT NULL));
```

4.3.4 Controllo Idoneità Proposta

Non può esistere un operazione con proposta=true effettuata da un utente che è lo stesso autore della pagina.

```
CREATE OR REPLACE FUNCTION check_proposta_function()
RETURNS TRIGGER AS $$
BEGIN
   IF NEW.proposta=TRUE AND NEW.utente = (SELECT UserAutore
       FROM PAGINA WHERE ID_Pagina=NEW.ID_Pagina) THEN
      RAISE EXCEPTION 'Impossibile inserire un record in "
          OPERAZIONE" con proposta=true e utente  lo stesso
          autore della pagina';
   END IF;

   RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_proposta
BEFORE INSERT
ON OPERAZIONE
FOR EACH ROW
EXECUTE FUNCTION check_proposta_function();
```

4.3.5 Controllo Idoneità Approvazione

Non può esistere un approvazione riferita ad un'operazione che non è una proposta. Inoltre, un autore di una pagina non può approvare proposte di operazioni su pagine non scritte da lui.

```
1  CREATE OR REPLACE FUNCTION check_approvazione_function()
2  RETURNS TRIGGER AS $$
3  DECLARE
4     paginaInteressata INT;
5     autorePagina USERNAME_DOMINIO;
6  BEGIN
7     IF((SELECT proposta FROM OPERAZIONE WHERE ID_Operazione=NEW.
          ID_Operazione)=FALSE) THEN
8        RAISE EXCEPTION 'Impossibile inserire un record in "
             APPROVAZIONE" il cui id_operazione faccia riferimento
             ad una tupla di OPERAZIONE con proposta=false';
9     END IF;
10
11    SELECT id_pagina INTO paginaInteressata FROM OPERAZIONE
          WHERE id_operazione = NEW.id_operazione;
12    SELECT UserAutore INTO autorePagina FROM PAGINA WHERE
          id_pagina = paginaInteressata;
13
14    IF(autorePagina ¡¿ NEW.autore) THEN
15       RAISE EXCEPTION 'Impossibile inserire un record in "
             APPROVAZIONE" che fa riferimento ad un operazione di
             una pagina, il cui autore non  l''utente indicato';
16    END IF;
17
18    RETURN NEW;
19 END;
20 $$ LANGUAGE plpgsql;
21
22
23 CREATE TRIGGER check_approvazione
24 BEFORE INSERT
25 ON APPROVAZIONE
26 FOR EACH ROW
27 EXECUTE FUNCTION check_approvazione_function();
```

4.4 Trigger e Funzioni Annesse

4.4.1 Trigger: Diventa Autore

Nel momento in cui inseriamo una nuova pagina, il campo "Autore" dell'utente indicato verrà impostato a TRUE.

```
CREATE OR REPLACE FUNCTION setAutore()
RETURNS TRIGGER AS $$
BEGIN
   UPDATE UTENTE
   SET autore = TRUE
   WHERE Username = NEW.UserAutore;

   RETURN NEW;
END;
$$ LANGUAGE plpgsql;


-- quando un utente crea una pagina il suo valore Autore
    diventa true
CREATE TRIGGER diventaAutore
AFTER INSERT ON PAGINA
FOR EACH ROW
EXECUTE FUNCTION setAutore();
```

4.4.2 Trigger: Ordina Frase Inserimento

Nel momento in cui inseriamo una frase e viene collocata in mezzo ad altre frase, verrà riordinato il campo "Ordine".

```
CREATE OR REPLACE FUNCTION ordinamentoFraseInserimento()
    RETURNS TRIGGER AS
$$
DECLARE
   maxFrase INT;
BEGIN
   SELECT MAX(ordine) INTO maxFrase FROM FRASE WHERE ID˙Pagina=
       NEW.ID˙Pagina AND Riga=NEW.riga;

   IF(maxFrase IS NULL) THEN
      maxFrase := 0;
   END IF;

   IF(maxFrase IS NOT NULL AND maxFrase¿=NEW.ordine) THEN

      FOR i IN REVERSE maxFrase..NEW.ordine LOOP

         UPDATE FRASE
         SET ordine = ordine + 1
         WHERE ordine = i AND id˙pagina = NEW.id˙pagina AND
             Riga = NEW.riga;


      END LOOP;

   ELSIF (NEW.ordine IS NULL OR NEW.ordine ¿ maxFrase) THEN
      NEW.ordine = maxFrase + 1;
   END IF;

   RETURN NEW;
END
$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER ordinaFraseInserimento
BEFORE INSERT
ON FRASE
FOR EACH ROW
EXECUTE FUNCTION ordinamentoFraseInserimento();
```

4.4.3 Trigger: Ordina Frase Cancellazione

Nel momento in cui cancelliamo una frase, verrà riordinato il campo "Ordine".

```
CREATE OR REPLACE FUNCTION ordinamentoFraseCancellazione()
    RETURNS TRIGGER AS
$$
DECLARE
   maxFrase INT;
BEGIN
   SELECT MAX(ordine) INTO maxFrase FROM FRASE WHERE ID˙Pagina=
       OLD.ID˙Pagina AND Riga=OLD.riga;

   IF(maxFrase IS NOT NULL AND maxFrase¿OLD.ordine) THEN

      FOR i IN OLD.ordine+1..maxFrase LOOP

         UPDATE FRASE
         SET ordine = ordine - 1
         WHERE ordine = i AND id˙pagina = OLD.id˙pagina AND
             Riga = OLD.riga;

      END LOOP;
   END IF;

   RETURN NEW;
END
$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER ordinaFraseCancellazione
AFTER DELETE
ON FRASE
FOR EACH ROW
EXECUTE FUNCTION ordinamentoFraseCancellazione();
```

4.4.4 Trigger: Creazione Approvazione

Quando viene inserita una operazione con proposta = TRUE, verrà inserita in Approvazione una tupla a cui farà riferimento, con Rispota e data uguale a NULL.

```
CREATE OR REPLACE FUNCTION creazioneApprovazione() RETURNS
    TRIGGER AS
$$
DECLARE
   autore USERNAME˙DOMINIO;
BEGIN
   SELECT UserAutore INTO Autore FROM PAGINA WHERE ID˙Pagina =
       NEW.id˙pagina;

   INSERT INTO APPROVAZIONE VALUES (NEW.ID˙Operazione, autore,
       null, null);

   RETURN NEW;
END
$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER creazioneApprovazioneTrigger
AFTER INSERT
ON OPERAZIONE
FOR EACH ROW
WHEN (NEW.proposta = TRUE)
EXECUTE FUNCTION creazioneApprovazione();
```

4.4.5 Trigger: Sovrascrizione Proposta

Quando viene inserita un operazione con proposta = TRUE, e tale proposta riguarda una frase già precedentemente coinvolta in un'altra proposta (di qualsiasi operazione) dello stesso utente, verrà effettuato un processo di sovrascrizione.

```
CREATE OR REPLACE FUNCTION sovrascizioneProposta˙function()
    RETURNS TRIGGER AS
$$
DECLARE
   tupleRimosse INT;
BEGIN
   DELETE FROM OPERAZIONE O
   USING APPROVAZIONE A
   WHERE O.id˙operazione = A.id˙operazione
   AND O.id˙operazione ¡¿ NEW.id˙operazione
   AND O.proposta = true
   AND O.id˙pagina = NEW.id˙pagina
   AND O.riga = NEW.riga
   AND O.ordine = NEW.ordine
   AND O.utente = NEW.utente
   AND A.risposta IS NULL;

   GET DIAGNOSTICS tupleRimosse = ROW˙COUNT;

   IF(tupleRimosse¡¿0) THEN
      RAISE NOTICE 'proposta di operazione sovrascritta';
   END IF;

   RETURN NEW;
END
$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER sovrascizioneProposta
AFTER INSERT
ON OPERAZIONE
FOR EACH ROW
WHEN (NEW.proposta = TRUE)
EXECUTE FUNCTION sovrascizioneProposta˙function();
```

4.4.6 Trigger: Elimina Proposte Antiche

Questa sezione è composta da due trigger differenti, scaturiti in contesti diversi che eseguiranno la stessa operazione:

- Il **primo trigger** si attiva nel momento in cui inseriamo operazioni effettuate dall'autore della pagina (proposta = FALSE). Se quell'operazione va a modificare una frase riferita da una o più proposte ancora in attesa di risposta, quelle proposte verranno automaticamente rifiutate.
- Il **secondo trigger** si attiva nel momento in cui viene approvata una proposta di operazione. Le altre proposte (ancora in attesa di risposta) che fanno riferimento alla stessa frase coinvolta, verranno automaticamente rifiutate.

```
1  CREATE OR REPLACE FUNCTION eliminaProposteAntiche_function1()
       RETURNS TRIGGER AS
2  $$
3  BEGIN
4
5     UPDATE APPROVAZIONE AS A
6     SET Risposta = false, data = CURRENT_TIMESTAMP
7     FROM OPERAZIONE O
8     WHERE A.id_operazione = O.id_operazione
9     AND O.proposta = true
10    AND O.id_pagina = NEW.id_pagina
11    AND O.riga = NEW.riga
12    AND O.ordine = NEW.ordine
13    AND A.risposta IS NULL;
14
15
16    RETURN NEW;
17 END
18 $$ LANGUAGE plpgsql;
19
20
21 CREATE OR REPLACE FUNCTION eliminaProposteAntiche_function2()
       RETURNS TRIGGER AS
22 $$
23 DECLARE
24    operazioneInteressata OPERAZIONE%ROWTYPE;
25 BEGIN
26    SELECT * INTO operazioneInteressata FROM OPERAZIONE WHERE
          ID_Operazione = NEW.ID_Operazione;
27
28    UPDATE APPROVAZIONE AS A
29    SET Risposta = false, data = CURRENT_TIMESTAMP
30    FROM OPERAZIONE O
31    WHERE A.id_operazione = O.id_operazione
32    AND O.proposta = true
```

```
   AND O.id_pagina = operazioneInteressata.id_pagina
   AND O.riga = operazioneInteressata.riga
   AND O.ordine = operazioneInteressata.ordine
   AND A.risposta IS NULL;


   RETURN NEW;
END
$$ LANGUAGE plpgsql;


CREATE OR REPLACE TRIGGER eliminaProposteAntiche1
AFTER INSERT
ON OPERAZIONE
FOR EACH ROW
WHEN (NEW.proposta=FALSE)
EXECUTE FUNCTION eliminaProposteAntiche_function1();

CREATE OR REPLACE TRIGGER eliminaProposteAntiche2
AFTER UPDATE OF risposta ON APPROVAZIONE
FOR EACH ROW
WHEN(NEW.risposta = TRUE)
EXECUTE FUNCTION eliminaProposteAntiche_function2();
```

4.4.7 Trigger: Controllo Esito Approvazione

Quando un operazione di proposta viene approvata da un autore, verrà chiamata una funzione che renderà effettiva l'operazione sulla tabella *Frase*.

```
CREATE OR REPLACE FUNCTION effettuaProposta() RETURNS TRIGGER
    AS
$$
DECLARE
   operazioneProposta OPERAZIONE%ROWTYPE;
   stringaId VARCHAR(100);
   id_collegamento INT;
   fraseConsiderata FRASE%ROWTYPE;
BEGIN
   SELECT * INTO operazioneProposta FROM OPERAZIONE WHERE
       ID_Operazione = NEW.ID_Operazione;

   --controllare se la frase da modificare o rimuovere esiste
       ancora
   SELECT * INTO fraseConsiderata FROM FRASE WHERE id_pagina=
       operazioneProposta.id_pagina AND riga =
       operazioneProposta.riga AND ordine = operazioneProposta.
       ordine;

   IF(operazioneProposta.tipo LIKE 'I') THEN

      INSERT INTO FRASE VALUES (operazioneProposta.Riga,
          operazioneProposta.ordine, operazioneProposta.
          ID_Pagina, operazioneProposta.FraseCoinvolta, false);

   ELSIF (operazioneProposta.tipo LIKE 'M') THEN

      IF(fraseConsiderata IS NULL) THEN
         RAISE EXCEPTION 'la frase coinvolta nel operazione non
             esiste pi';
      END IF;

      IF(operazioneProposta.fraseModificata LIKE '#+%') THEN --
          se  stato aggiunto un collegamento

         SELECT SUBSTRING(operazioneProposta.fraseModificata
             FROM 3) INTO stringaId;
         SELECT SUBSTRING(stringaId, 1, (SELECT strpos(
             stringaId, '#')-1)) INTO stringaId;
         SELECT stringaId::INT INTO id_collegamento; --
             conversione da VARCHAR a INT

         -- controllo se esiste gi il collegamento
         IF(EXISTS(SELECT * FROM COLLEGAMENTO WHERE id_pagina =
              operazioneProposta.ID_Pagina AND rigaFrase=
```

```
            operazioneProposta.riga AND ordineFrase=
            operazioneProposta.ordine)=FALSE) THEN
32          INSERT INTO COLLEGAMENTO VALUES(operazioneProposta.
                riga, operazioneProposta.ordine,
                operazioneProposta.id_pagina, id_collegamento);
33
34          UPDATE FRASE
35          SET Collegamento = true
36          WHERE riga= operazioneProposta.riga AND ordine =
                operazioneProposta.ordine AND ID_Pagina =
                operazioneProposta.id_pagina;
37       ELSE
38          UPDATE COLLEGAMENTO
39          SET ID_PaginaCollegata = id_collegamento
40          WHERE id_pagina = operazioneProposta.ID_Pagina AND
                rigaFrase=operazioneProposta.riga AND
                ordineFrase=operazioneProposta.ordine;
41       END IF;
42    ELSIF(operazioneProposta.fraseModificata LIKE '#-%') THEN
         --se  stato rimosso un collegamento
43
44       DELETE FROM COLLEGAMENTO
45       WHERE id_pagina = operazioneProposta.ID_Pagina AND
             rigaFrase=operazioneProposta.riga AND ordineFrase=
             operazioneProposta.ordine;
46
47       UPDATE FRASE
48       SET Collegamento = false
49       WHERE riga= operazioneProposta.riga AND ordine =
             operazioneProposta.ordine AND ID_Pagina =
             operazioneProposta.id_pagina;
50
51    ELSE -- stato modificato il contenuto della frase
52       UPDATE FRASE
53       SET Contenuto = operazioneProposta.FraseModificata
54       WHERE riga= operazioneProposta.riga AND ordine =
             operazioneProposta.ordine AND ID_Pagina =
             operazioneProposta.id_pagina;
55    END IF;
56
57 ELSIF (operazioneProposta.tipo LIKE 'C') THEN
58
59    IF(fraseConsiderata IS NULL) THEN
60       RAISE EXCEPTION 'la frase coinvolta nel operazione non
             esiste pi';
61    END IF;
62
63    DELETE FROM FRASE
64    WHERE riga= operazioneProposta.riga AND ordine =
          operazioneProposta.ordine AND ID_Pagina =
```

```
          operazioneProposta.id_pagina;

   END IF;

   RETURN NEW;
END
$$ LANGUAGE plpgsql;


CREATE OR REPLACE TRIGGER controlloEsitoApprovazioneTrigger
AFTER UPDATE
OF Risposta ON APPROVAZIONE
FOR EACH ROW
WHEN (NEW.Risposta = TRUE)
EXECUTE FUNCTION effettuaProposta();
```

4.5 Procedure

4.5.1 Procedura: Inserisci Frase

Questa procedura ci permette di inserire una frase in un testo di una specifica pagina del sistema. I parametri della procedura sono i seguenti:

- *IDPaginaF*: l'identificativo della pagina in cui desideriamo inserire la frase.
- *RigaF*: il numero di riga su cui desideriamo inserire la frase.
- *OrdineF*: L'ordine di posizionamento sulla riga, indicata precedentemente, della frase che verrà inserita. Quindi attraverso questo parametro possiamo indicare se la frase che stiamo inserendo sarà la prima della riga, oppure la seconda, la terza e cosi via. Se tale parametro non sarà specificato (e quindi NULL), oppure viene indicato un valore maggiore rispetto al numero delle frasi su quella riga (Es: Ci sono tre frasi sulla riga 1 e indico RigaF = 1 e ordineF = 15) ordine verrà automaticamente gestito e gli verrà assegnato il valore MAX(ordine)+1 su quella riga.
- *ContenutoF*: la stringa che indica il contenuto della frase che stiamo inserendo.
- *nomeUtente*: Username dell'utente che sta effettuando l'operazione di inserimento.

Se l'utente che indicheremo nella procedura è l'autore della stessa pagina indicata, allora oltre all'inserimento effettivo nella tabella *Frase*, verrà inserita nella tabella *Operazione*, una tupla che mostrerà le informazioni relative all'inserimento appena compiuto, con l'attributo *proposta* = *false* e tipo = 'I'.
Se l'utente indicato non è l'autore della pagina, allora non verrà effettuato alcun inserimento nella tabella *Frase*, e in *Operazione* verrà inserita una tupla (con le relative informazioni) con *proposta* = *true* e tipo = 'I'.

```
1  CREATE OR REPLACE PROCEDURE inserisciFrase(ID_PaginaF INT,
       rigaF INT, ordineF INT DEFAULT NULL, ContenutoF LEN_FRASE,
       nomeUtente USERNAME_DOMINIO)
2  LANGUAGE plpgsql
3  AS $$
4  DECLARE
5     maxOrdine INT;
6     ordineFrase INT;
7     autorePagina USERNAME_DOMINIO;
8     proposta BOOLEAN;
9  BEGIN
10    -- controllo se esiste l'utente
11    IF(EXISTS(SELECT * FROM UTENTE WHERE Username=nomeUtente)=
          FALSE) THEN
12       RAISE EXCEPTION 'utente indicato non esistente';
13    END IF;
```

```
14
15    -- controllo se esiste la pagina
16    IF(EXISTS(SELECT * FROM PAGINA WHERE ID˙Pagina=ID˙PaginaF)=
          FALSE) THEN
17       RAISE EXCEPTION 'pagina indicata non esistente';
18    END IF;
19
20    SELECT UserAutore INTO autorePagina FROM PAGINA WHERE
          id˙pagina = ID˙PaginaF;
21
22
23
24    -- controlla se l'utente  l'autore stesso
25    IF(autorePagina = nomeUtente) THEN
26
27       proposta:=false; -- inserimento diretto (da parte dell'
             autore stesso)
28
29       -- prendo l'ordine massimo sulla riga coinvolta
30       SELECT MAX(ordine) INTO maxOrdine FROM FRASE WHERE
             ID˙Pagina = ID˙PaginaF AND riga = rigaF;
31
32       -- se si tratta della prima frase sulla riga
33       IF(maxOrdine IS NULL) THEN
34          maxOrdine:=0;
35       END IF;
36
37       /* se l'utente specifica l'ordine della frase che sta
             inserendo e quest'ultimo si trova nel mezzo della
38       riga (ordineF ¡ maxOrdine) allora l'ordine della frase
             quello indicato, altrimenti l'ordine sar il massimo +1
              */
39       IF(ordineF IS NOT NULL AND maxOrdine¿ordineF) THEN
40          ordineFrase := OrdineF;
41       ELSE
42          ordineFrase := maxOrdine+1;
43       END IF;
44
45       INSERT INTO FRASE VALUES(RigaF, ordineFrase, ID˙PaginaF,
             ContenutoF, false);
46
47    ELSE
48       proposta:=true; -- altrimenti l'operazione  solo di
             proposta
49
50       SELECT MAX(O.ordine) INTO maxOrdine FROM OPERAZIONE O,
             APPROVAZIONE A WHERE O.id˙operazione=A.id˙operazione
             AND O.id˙pagina= ID˙PaginaF AND O.riga = rigaF AND O.
             utente = nomeUtente AND A.Risposta IS NULL;
51
```

```
52      IF(maxOrdine IS NULL) THEN
53         maxOrdine:=0;
54      END IF;
55
56      IF(ordineF IS NOT NULL AND maxOrdine¿ordineF) THEN
57         ordineFrase := OrdineF;
58      ELSE
59         ordineFrase := maxOrdine+1;
60      END IF;
61
62
63   END IF;
64
65   INSERT INTO OPERAZIONE VALUES(DEFAULT, 'I', proposta, rigaF,
         ordineFrase, ContenutoF, null, DEFAULT, ID_PaginaF,
        nomeUtente);
66
67 END;
68 $$;
```

4.5.2 Procedura: Modifica Frase

Questa procedura ci permette di modificare una frase in un testo di una specifica pagina del sistema. I parametri della procedura sono i seguenti:

- *ID PaginaF*: l'identificativo della pagina in cui desideriamo modificare una frase.
- *RigaF*: il numero di riga su cui si trova la frase che desideriamo modificare.
- *OrdineF*: L'ordine di posizionamento sulla riga, indicata precedentemente, della frase che desideriamo modificare.
- *ContenutoF*: la stringa che indica il nuovo contenuto che avrà la frase che modificheremo.
- *nomeUtente*: Username dell'utente che sta effettuando l'operazione di modifica.

Se l'utente che indicheremo nella procedura è l'autore della stessa pagina indicata, allora oltre alla modifica effettiva nella tabella *Frase*, verrà inserita nella tabella *Operazione*, una tupla che mostrerà le informazioni relative alla modifica appena compiuta, con l'attributo *proposta* = *false* e *tipo* = *'M'*.
Se l'utente indicato non è l'autore della pagina, allora non verrà effettuata alcuna modifica nella tabella *Frase*, e in *Operazione* verrà inserita una tupla (con le relative informazioni) con *proposta* = *true* e *tipo* = *'M'*.

```
1  CREATE OR REPLACE PROCEDURE modificaFrase(ID_PaginaF INT, rigaF
       INT, ordineF INT, ContenutoF LEN_FRASE, nomeUtente
      USERNAME_DOMINIO)
2  LANGUAGE plpgsql
3  AS $$
4  DECLARE
5     oldFrase LEN_FRASE;
6     autorePagina USERNAME_DOMINIO;
7     proposta BOOLEAN;
8  BEGIN
9     -- controllo se esiste l'utente
10    IF(EXISTS(SELECT * FROM UTENTE WHERE Username=nomeUtente)=
          FALSE) THEN
11       RAISE EXCEPTION 'utente indicato non esistente';
12    END IF;
13
14    -- controllo se esiste la pagina
15    IF(EXISTS(SELECT * FROM PAGINA WHERE ID_Pagina=ID_PaginaF)=
          FALSE) THEN
16       RAISE EXCEPTION 'pagina indicata non esistente';
17    END IF;
18
19    -- controllo se esiste la frase (e di conseguenza la pagina)
```

```
20    IF(EXISTS(SELECT * FROM FRASE WHERE riga = rigaF AND ordine=
          ordineF AND ID˙Pagina = ID˙PaginaF)=FALSE) THEN
21       RAISE EXCEPTION 'la frase indicata non esiste';
22    END IF;
23
24    SELECT UserAutore INTO autorePagina FROM PAGINA WHERE
          id˙pagina = ID˙PaginaF;
25
26    SELECT Contenuto INTO oldFrase FROM FRASE WHERE riga=rigaF
          AND ordine = ordineF AND ID˙Pagina = ID˙PaginaF;
27
28    IF(autorePagina = nomeUtente) THEN
29
30       proposta := false;
31
32       UPDATE FRASE
33       SET Contenuto = ContenutoF
34       WHERE riga=rigaF AND ordine = ordineF AND ID˙Pagina =
             ID˙PaginaF;
35    ELSE
36       proposta := true;
37    END IF;
38
39    INSERT INTO OPERAZIONE VALUES(DEFAULT, 'M', proposta, rigaF,
          ordineF, oldFrase, ContenutoF, DEFAULT, ID˙PaginaF,
          nomeUtente);
40
41 END;
42 $$;
```

4.5.3 Procedura: Rimuovi Frase

Questa procedura ci permette di rimuovere una frase in un testo di una specifica pagina del sistema. I parametri della procedura sono i seguenti:

- *ID PaginaF*: l'identificativo della pagina in cui desideriamo eliminare una frase.
- *RigaF*: il numero di riga su cui si trova la frase che desideriamo eliminare.
- *OrdineF*: L'ordine di posizionamento sulla riga, indicata precedentemente, della frase che desideriamo eliminare.
- *nomeUtente*: Username dell'utente che sta effettuando l'operazione di cancellazione.

Se l'utente che indicheremo nella procedura è l'autore della stessa pagina indicata, allora oltre alla cancellazione effettiva nella tabella *Frase*, verrà inserita nella tabella *Operazione*, una tupla che mostrerà le informazioni relative alla cancellazione appena compiuta, con l'attributo *proposta* = *false* e *tipo* = *'C'*.
Se l'utente indicato non è l'autore della pagina, allora non verrà effettuata alcuna cancellazione nella tabella *Frase*, e in *Operazione* verrà inserita una tupla (con le relative informazioni) con *proposta* = *true* e *tipo* = *'C'*.

```
1  CREATE OR REPLACE PROCEDURE rimuoviFrase(ID_PaginaF INT, rigaF
       INT, ordineF INT, nomeUtente USERNAME_DOMINIO)
2  LANGUAGE plpgsql
3  AS $$
4  DECLARE
5     oldFrase LEN_FRASE;
6     autorePagina USERNAME_DOMINIO;
7     proposta BOOLEAN;
8  BEGIN
9     -- controllo se esiste l'utente
10    IF(EXISTS(SELECT * FROM UTENTE WHERE Username=nomeUtente)=
          FALSE) THEN
11       RAISE EXCEPTION 'utente indicato non esistente';
12    END IF;
13
14    -- controllo se esiste la pagina
15    IF(EXISTS(SELECT * FROM PAGINA WHERE ID_Pagina=ID_PaginaF)=
          FALSE) THEN
16       RAISE EXCEPTION 'pagina indicata non esistente';
17    END IF;
18
19    -- controllo se esiste la frase (e di conseguenza la pagina)
20    IF(EXISTS(SELECT * FROM FRASE WHERE riga = rigaF AND ordine=
          ordineF AND ID_Pagina = ID_PaginaF)=FALSE) THEN
21       RAISE EXCEPTION 'la frase indicata non esiste';
22    END IF;
```

```
23
24    SELECT Contenuto INTO oldFrase FROM FRASE WHERE riga=rigaF
          AND ordine = ordineF AND ID_Pagina = ID_PaginaF;
25
26    SELECT UserAutore INTO autorePagina FROM PAGINA WHERE
          id_pagina = ID_PaginaF;
27
28    IF(autorePagina = nomeUtente) THEN
29       DELETE FROM FRASE
30       WHERE riga=rigaF AND ordine = ordineF AND ID_Pagina =
             ID_PaginaF;
31       proposta := false;
32    ELSE
33       proposta := true;
34    END IF;
35
36    INSERT INTO OPERAZIONE VALUES(DEFAULT, 'C', proposta, rigaF,
          ordineF, oldFrase, null, DEFAULT, ID_PaginaF, nomeUtente
          );
37
38 END;
39 $$;
```

4.5.4 Procedura: Inserisci Collegamento

Questa procedura ci permette di inserire (ed eventualmente sovrascrivere) un collegamento ad una frase esistente nel testo di una pagina del sistema. I parametri della procedura sono i seguenti:

- *ID PaginaF*: l'identificativo della pagina in cui desideriamo aggiungere un collegamento ad una frase.
- *RigaF*: il numero di riga su cui si trova la frase che desideriamo aggiungergli un collegamento.
- *OrdineF*: L'ordine di posizionamento sulla riga, indicata precedentemente, della frase coinvolta.
- *ID Collegamento*: l'identificativo della pagina a cui farà riferimento la frase collegamento.
- *nomeUtente*: Username dell'utente che sta effettuando l'operazione.

Se l'utente che indicheremo nella procedura è l'autore della stessa pagina indicata, allora all'inserimento di una tupla nella tabella Collegamento, verrà inserita nella tabella *Operazione*, una tupla che mostrerà le informazioni relative all'operazione di collegamento appena compiuta, con l'attributo *proposta = false* e *tipo = 'M'*.
Se l'utente indicato non è l'autore della pagina, allora non verrà effettuata alcun inserimento nella tabella Collegamento, e in *Operazione* verrà inserita una tupla (con le relative informazioni) con *proposta = true* e *tipo = 'M'*.
Le operazioni di inserimento di un collegamento, sono considerate dal sistema come operazioni di modifica, e ogni tupla verrà strutturata nel seguente modo (oltre al posizionamento della frase nel testo):

- *Frase Coinvolta*: Contiene il contenuto della frase coinvolta nell'inserimento del collegamento.
- *Frase Modificata*: Avrà una stringa composta nel modo seguente "#+idPaginaCollegamento# Titolo della pagina".

```
1 CREATE OR REPLACE PROCEDURE inserisciCollegamento(ID_PaginaF
    INT, rigaF INT, ordineF INT, id_collegamento INT, nomeUtente
    USERNAME_DOMINIO)
2 LANGUAGE plpgsql
3 AS $$
4 DECLARE
5   stringaFrase LEN_FRASE;
6   titoloCollegamento VARCHAR(100);
7   stringaCollegamento LEN_FRASE;
8   autorePagina USERNAME_DOMINIO;
9   proposta BOOLEAN;
```

```
BEGIN
   -- controllo se esiste l'utente
   IF(EXISTS(SELECT * FROM UTENTE WHERE Username=nomeUtente)=
       FALSE) THEN
      RAISE EXCEPTION 'utente indicato non esistente';
   END IF;

   -- controllo se esiste la pagina
   IF(EXISTS(SELECT * FROM PAGINA WHERE ID_Pagina=ID_PaginaF)=
       FALSE) THEN
      RAISE EXCEPTION 'pagina indicata non esistente';
   END IF;

   -- controllo se esiste la frase (e di conseguenza la pagina)
   SELECT Contenuto INTO stringaFrase FROM FRASE WHERE riga =
       rigaF AND ordine=ordineF AND ID_Pagina = ID_PaginaF;

   IF(stringaFrase IS NULL) THEN
      RAISE EXCEPTION 'la frase indicata non esiste';
   END IF;

   SELECT titolo INTO titoloCollegamento FROM PAGINA WHERE
       id_pagina = id_collegamento;

   IF(titoloCollegamento IS NULL) THEN
      RAISE EXCEPTION 'La pagina indicata per il collegamento
          non esiste';
   END IF;

   SELECT UserAutore INTO autorePagina FROM PAGINA WHERE
       id_pagina = ID_PaginaF;

   IF(autorePagina = nomeUtente) THEN
      proposta := false;
      --controlliamo se esiste gi una tupla in collegamento,
          altrimenti la creiamo
      IF(EXISTS(SELECT * FROM COLLEGAMENTO WHERE rigaFrase =
          rigaF AND ordineFrase=ordineF AND ID_Pagina =
          ID_PaginaF)=FALSE) THEN --se non esiste
         INSERT INTO COLLEGAMENTO VALUES (rigaF, ordineF,
             ID_PaginaF, id_collegamento);

         UPDATE FRASE
         SET collegamento = true
         WHERE riga = rigaF AND ordine=ordineF AND ID_Pagina =
             ID_PaginaF;
      ELSE
         UPDATE COLLEGAMENTO
         SET ID_PaginaCollegata = id_collegamento
         WHERE rigaFrase = rigaF AND ordineFrase=ordineF AND
```

```
            ID_Pagina = ID_PaginaF;
49      END IF;
50   ELSE
51      proposta := true;
52   END IF;
53
54   stringaCollegamento := '#+' || id_collegamento || '# ' ||
        titoloCollegamento;
55   INSERT INTO OPERAZIONE VALUES(DEFAULT, 'M', proposta, rigaF,
        ordineF, stringaFrase, stringaCollegamento, DEFAULT,
        ID_PaginaF, nomeUtente);
56
57 END;
58 $$;
```

4.5.5 Procedura: Rimuovi Collegamento

Questa procedura ci permette di eliminare un collegamento ad una frase esistente nel testo di una pagina del sistema. I parametri della procedura sono i seguenti:

- *ID PaginaF*: l'identificativo della pagina in cui desideriamo rimuovere un collegamento ad una frase.
- *RigaF*: il numero di riga su cui si trova la frase che desideriamo rimuovergli un collegamento.
- *OrdineF*: L'ordine di posizionamento sulla riga, indicata precedentemente, della frase coinvolta.
- *nomeUtente*: Username dell'utente che sta effettuando l'operazione.

Se l'utente che indicheremo nella procedura è l'autore della stessa pagina indicata, allora oltre alla cancellazione di una tupla nella tabella Collegamento, verrà inserita nella tabella *Operazione*, una tupla che mostrerà le informazioni relative all'operazione di collegamento appena compiuta, con l'attributo *proposta* = *false* e *tipo* = *'M'*.
Se l'utente indicato non è l'autore della pagina, allora non verrà effettuata alcuna cancellazione nella tabella Collegamento, e in *Operazione* verrà inserita una tupla (con le relative informazioni) con *proposta* = *true* e *tipo* = *'M'*.
Le operazioni di cancellazione di un collegamento, sono considerate dal sistema come operazioni di modifica, e ogni tupla verrà strutturata nel seguente modo (oltre al posizionamento della frase nel testo):

- *Frase Coinvolta*: Contiene il contenuto della frase coinvolta nell'inserimento del collegamento.
- *Frase Modificata*: Avrà una stringa composta nel modo seguente "#-# COLLEGAMENTO RIMOSSO".

```
1  CREATE OR REPLACE PROCEDURE rimuoviCollegamento(ID_PaginaF INT,
       rigaF INT, ordineF INT, nomeUtente USERNAME_DOMINIO)
2  LANGUAGE plpgsql
3  AS $$
4  DECLARE
5     stringaFrase LEN_FRASE;
6     titoloCollegamento VARCHAR(100);
7     stringaCollegamento LEN_FRASE;
8     autorePagina USERNAME_DOMINIO;
9     proposta BOOLEAN;
10 BEGIN
11    -- controllo se esiste l'utente
12    IF(EXISTS(SELECT * FROM UTENTE WHERE Username=nomeUtente)=
          FALSE) THEN
13       RAISE EXCEPTION 'utente indicato non esistente';
```

```
14    END IF;
15
16    -- controllo se esiste la pagina
17    IF(EXISTS(SELECT * FROM PAGINA WHERE ID˙Pagina=ID˙PaginaF)=
          FALSE) THEN
18       RAISE EXCEPTION 'pagina indicata non esistente';
19    END IF;
20
21    -- controllo se esiste la frase (e di conseguenza la pagina)
22    SELECT Contenuto INTO stringaFrase FROM FRASE WHERE riga =
          rigaF AND ordine=ordineF AND ID˙Pagina = ID˙PaginaF;
23
24    IF(stringaFrase IS NULL) THEN
25       RAISE EXCEPTION 'la frase indicata non esiste';
26    END IF;
27
28    IF(EXISTS(SELECT * FROM COLLEGAMENTO WHERE rigaFrase = rigaF
           AND ordineFrase=ordineF AND ID˙Pagina = ID˙PaginaF)=
          FALSE) THEN
29       RAISE EXCEPTION 'la frase non possiede collegamento';
30    END IF;
31
32
33    SELECT UserAutore INTO autorePagina FROM PAGINA WHERE
          id˙pagina = ID˙PaginaF;
34
35    IF(autorePagina = nomeUtente) THEN
36       proposta := false;
37
38       DELETE FROM COLLEGAMENTO
39       WHERE rigaFrase = rigaF AND ordineFrase=ordineF AND
             ID˙Pagina = ID˙PaginaF;
40
41       UPDATE FRASE
42       SET collegamento = false
43       WHERE riga = rigaF AND ordine=ordineF AND ID˙Pagina =
             ID˙PaginaF;
44
45    ELSE
46       proposta := true;
47    END IF;
48
49    stringaCollegamento := '#-# COLLEGAMENTO RIMOSSO';
50    INSERT INTO OPERAZIONE VALUES(DEFAULT, 'M', proposta, rigaF,
           ordineF, stringaFrase, stringaCollegamento, DEFAULT,
          ID˙PaginaF, nomeUtente);
51
52 END;
53 $$;
```

4.5.6 Procedura: Approva Proposta

Questa procedura ci permette di approvare/rifiutare una proposta fatta da un utente non autore della pagina coinvolta. Ricordiamo che una proposta è una tupla in Operazione che ha l'attributo proposta = true. I parametri della procedura sono i seguenti:

- *ID*: L'identificativo dell'operazione di proposta che desideriamo approvare/rifiutare.
- *Risposta*: Parametro di tipo booleano, con il quale indicheremo se approvare (true) o rifiutare (false) l'operazione di proposta.
- *nomeUtente*: Username dell'autore che approva la proposta. Se l'utente indicato non è l'autore della pagina coinvolta nella proposta di operazione, la procedura manderà un'eccezione.

```
1  CREATE OR REPLACE PROCEDURE approvaProposta(id INT, risp
       BOOLEAN, nomeUtente USERNAME˙DOMINIO)
2  LANGUAGE plpgsql
3  AS $$
4  DECLARE
5     autorePagina USERNAME˙DOMINIO;
6  BEGIN
7     SELECT autore INTO autorePagina FROM APPROVAZIONE WHERE
          id˙operazione = id;
8
9     IF(autorePagina ¡¿ nomeUtente) THEN
10       RAISE EXCEPTION 'Errore, l''utente indicato non  l''
             autore della pagina';
11    END IF;
12
13    IF(EXISTS(SELECT * FROM OPERAZIONE WHERE id˙operazione = id)
          = false) THEN
14       RAISE EXCEPTION 'Errore, l''operazione indicata non
             esiste';
15    END IF;
16
17    IF((SELECT proposta FROM OPERAZIONE WHERE id˙operazione = id
       )=false) THEN
18       RAISE EXCEPTION 'Errore, id indicato non  una proposta di
              un operazione';
19    END IF;
20
21    IF((SELECT Risposta FROM APPROVAZIONE WHERE id˙operazione =
       id) IS NOT NULL) THEN
22       RAISE EXCEPTION 'Errore, La proposta indicata ha gi avuto
              risposta';
23    END IF;
24
```

```
25    UPDATE APPROVAZIONE
26    SET Risposta = risp, data = CURRENT_TIMESTAMP
27    WHERE id_operazione = id;
28 END;
29 $$;
```

4.5.7 Procedura: Ritira Proposta

Questa procedura ci permette di ritirare una proposta (ancora in attesa di risposta da parte dell'autore) fatta da un utente non autore della pagina coinvolta. Ricordiamo che una proposta è una tupla in Operazione che ha l'attributo proposta = true. I parametri della procedura sono i seguenti:

- *ID*: L'identificativo dell'operazione di proposta che desideriamo ritirare (deve avere nella tabella Approvazione l'attributo con risposta IS NULL).
- *nomeUtente*: Username dell'utente che ha proposto l'operazione. Se l'utente indicato non è l'utente effettivo della proposta, la procedura manderà un'eccezione.

```
1  CREATE OR REPLACE PROCEDURE ritiraProposta(id INT, nomeUtente
       USERNAME˙DOMINIO)
2  LANGUAGE plpgsql
3  AS $$
4  DECLARE
5     autorePagina USERNAME˙DOMINIO;
6     utenteProposta USERNAME˙DOMINIO;
7  BEGIN
8     SELECT autore INTO autorePagina FROM APPROVAZIONE WHERE
          id˙operazione = id;
9     SELECT utente INTO utenteProposta FROM OPERAZIONE WHERE
          id˙operazione = id;
10
11    IF(autorePagina = nomeUtente) THEN
12       RAISE EXCEPTION 'Errore, l''utente indicato  l''autore
             della pagina';
13    END IF;
14
15    IF(utenteProposta ¡¿ nomeUtente) THEN
16       RAISE EXCEPTION 'Errore, l''utente indicato non  colui
             che ha proposta l''operazione';
17    END IF;
18
19    IF(EXISTS(SELECT * FROM OPERAZIONE WHERE id˙operazione = id)
          = false) THEN
20       RAISE EXCEPTION 'Errore, l''operazione indicata non
             esiste';
21    END IF;
22
23    IF((SELECT proposta FROM OPERAZIONE WHERE id˙operazione = id
          )=false) THEN
24       RAISE EXCEPTION 'Errore, id indicato non  una proposta di
              un operazione';
25    END IF;
26
```

```
27    IF((SELECT Risposta FROM APPROVAZIONE WHERE id˙operazione =
          id) IS NOT NULL) THEN
28       RAISE EXCEPTION 'Errore, La proposta indicata ha gi avuto
             risposta, non puoi ritirarla';
29    END IF;
30
31    DELETE FROM OPERAZIONE
32    WHERE id˙operazione = id AND utente = nomeUtente;
33 END;
34 $$;
```

4.6 Viste

4.6.1 Lista Proposte

Vista che visualizzerà tutte le operazioni di proposta e il loro stato di approvazione.

```
CREATE OR REPLACE VIEW listaproposte AS
SELECT O.*, A.data AS dataRisposta, A.risposta
FROM OPERAZIONE O, APPROVAZIONE A
WHERE O.id_operazione = A.id_operazione;
```

4.6.2 Storico Pagine

Vista che visualizzerà tutte le operazioni eseguite effettivamente su ogni pagina. Quindi le operazioni effettuate dall'autore (proposta = FALSE) e le proposte di operazione approvate.

```
CREATE OR REPLACE VIEW storicoPagine AS
(SELECT *
FROM OPERAZIONE
WHERE proposta=false)
UNION
(SELECT O.*
FROM OPERAZIONE O, APPROVAZIONE A
WHERE O.id_operazione = A.id_operazione
AND A.Risposta=true)
ORDER BY id_pagina ASC, data ASC;
```