Progetto Database Wikipedia

Tammaro Iavarazzo

Angelo Marcone

N86004573

N86004638

Anno Accademico
23/24

Indice

1 Progettazione Concettuale

1.1 Analisi dei Requisiti

La nostra traccia richiede di creare un sistema informativo per la gestione del ciclo di vita di una pagina di una wiki. Dalla traccia andiamo a trovare i seguenti requisiti:

1. **Gestione degli utenti:**
   (a) Gli utenti si registrano al sistema fornendo le proprie informazioni come email, nome, cognome e password.
   (b) Gli autori sono utenti speciali che hanno un nome d'arte e un anno di inizio della carriera. Ogni autore è associato a un utente.
2. **Gestione delle pagine wiki:**
   (a) Ogni pagina ha un titolo univoco e un testo associato.
   (b) Ogni pagina è creata da un autore e ha una data e un'ora di creazione.
3. **Gestione delle modifiche:**
   (a) Gli utenti possono proporre modifiche al testo di una pagina.
   (b) gni modifica è caratterizzata da un codice univoco, la frase originale, la frase modificata, lo stato della modifica (accettata, rifiutata o in attesa), data e ora della modifica, l'utente che ha proposto la modifica, l'autore della pagina e il codice della frase associata alla modifica.
   (c) Le modifiche proposte vengono notificate all'autore originale della pagina per la revisione.
   (d) L'autore può accettare o rifiutare una modifica. Se accettata, diventa parte della versione corrente del testo della pagina.
4. **Gestione delle frasi:**
   (a) Il testo di una pagina è composto da una sequenza di frasi.
   (b) Ogni frase può contenere collegamenti ad altre pagine wiki.
5. **Gestione dei collegamenti:**
   (a) I collegamenti collegano una frase della pagina a un'altra pagina wiki.
   (b) Ogni collegamento è caratterizzato da una frase di origine e dalla pagina di destinazione del collegamento.

6. **Autenticazione degli autori:**

   (a) Gli autori devono autenticarsi fornendo la loro email e password per accedere al sistema.

   (b) Gli autori possono accedere alla storia di tutti i testi di cui sono autori e di quelli in cui hanno proposto modifiche.

7. **Ricerca delle pagine:**

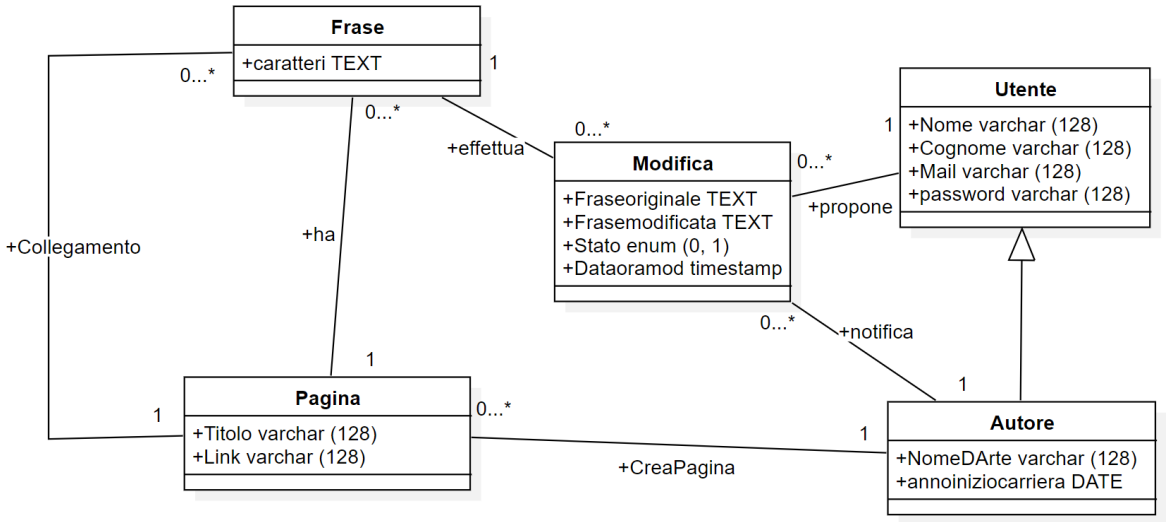
   (a) Gli utenti possono cercare una pagina e visualizzare la versione corrente del testo e i collegamenti associati.

8. **Revisione delle modifiche multiple:**

   (a) Se ci sono più modifiche pendenti su una pagina, l'autore deve decidere se accettarle o rifiutarle nell'ordine cronologico delle proposte.

1.2 Modello Concettuale

Schema concettuale del database costruito tramite le informazioni ottenute dall'analisi dei requisiti



1.3 Dizionario Entità

entita	descrizione	Attributi
Utente	Utente di base loggato, in grado di richiedere la modifica di una pagina	**Nome(VARCHAR128):** è il nome dell'utente **Cognome(VARCHAR128):** è il cognome dell'utente **Mail(VARCHAR128)**: è la mail dell'utente **Password(VARCHAR128):** è la password dell'utente
Autore	è un utente in grado di creare pagine	**NomeDArte(VARCHAR128):** è il nome d'arte che contraddistingue l'autore **AnnoInizioCarriera(Timestamp):** l'anno in cui l'autore ha scritto la sua prima pagina
Pagina	Le pagine che verranno mostrate agli utenti	**Titolo(VARCHAR128):** è il titolo della pagina **Link(VARCHAR128):** è il link della pagina
Modifica	La modifica richiesta, nel caso dell'utente, oppure effettuata, nel caso dell'autore	**FraseOriginal(TEXT):** è la frase originale prima della modifica **FraseModificata(TEXT):** è la frase dopo la modifica **Stato(1,0,null):** 1 sta per modifica accettata, 0 sta per modifica rifiutata,e null sta per modifica in stallo **DataOra(Timestampo):** è la data e l'ora in cui la modifica viene richiesta
Frase	Sono le svariate frasi che si ritrovano all'interno della pagina	**Caratteri(TEXT):** è il contenuto della frase

1.4 Dizionario Associazioni

Associazione	Descrizione	Attributi
Collegamento	Associazione tra Frase e Pagina uno-a-molti	
effettua	Associazione tra Frase e Modifica molti-a-molti	
ha	Associazione tra Frase e Pagina uno-a-molti	
CreaPagina	Associazione tra Pagina e Autore uno-a-molti	
notifica	Associazione tra Modifica e Autore uno-a-molti	
propone	Associazione tra Modifica e Utente uno-a-molti	

2 Ristrutturazione Modello Concettuale

2.1 Analisi delle Ridondanze

Non sono presenti Ridondanze.

2.2 Eliminazione delle Generalizzazioni

Nel nostro diagramma abbiamo una generalizzazione di tipo parziale e la definiamo tale poichè non è detto che un utente debba essere necessariamente un Autore. Nella ristrutturazione abbiamo deciso di trasformare la generalizzazione in un'associazione 1 a 1 parziale poichè un utente può o non può essere un Autore, e un Autore deve essere necessariamente un Utente.

2.3 Eliminazione degli Attributi Multivalore

Non sono presenti Attributi Multivalore

2.4 Eliminazione degli Attributi Strutturati

Non sono presenti degli Attributi Strutturati

2.5 Partizionamento/Accorpamento di Entità e Associazioni

Nessun Partizionamento/Accorpamento di Entità e Associazioni effettuato

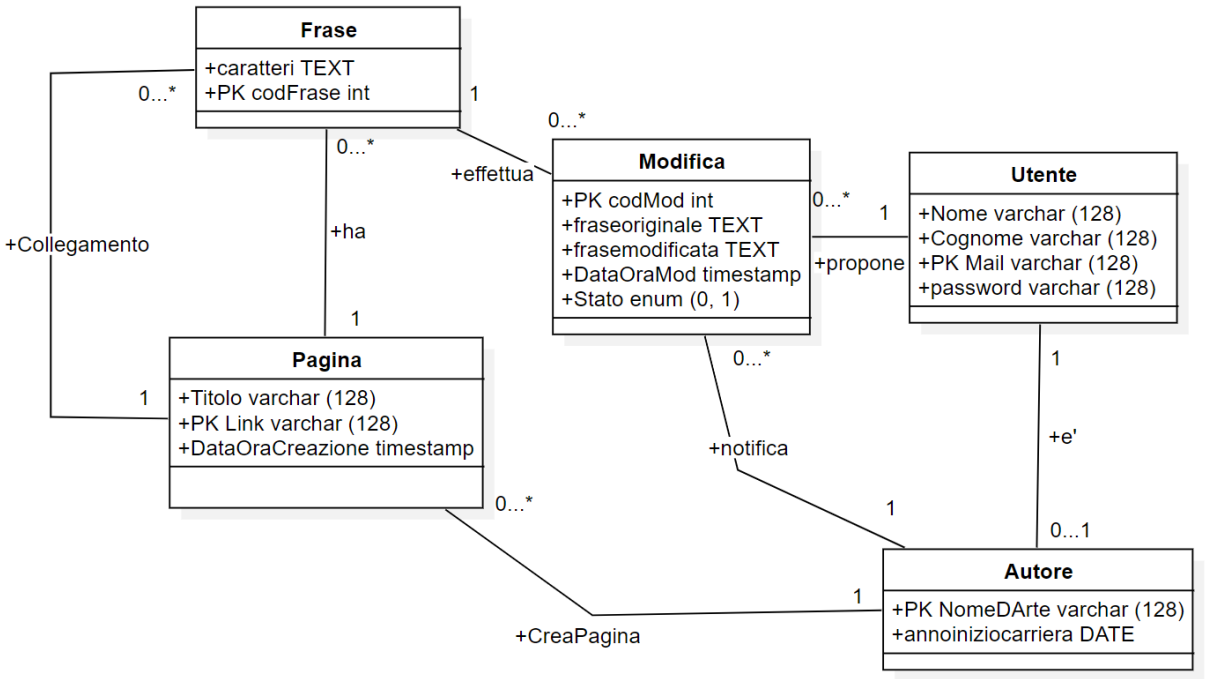
2.6 Identificazioni chiavi primarie

In questa fase sceglieremo i nostri identificativi univoci delle nostre entità,in particolare:

1. Nell'entità **Frase** la chiave primaria è una chiave composta formata da codFrase(che abbiamo aggiunto poichè l'entità Frase non aveva attributi univoci) e codPagina.
2. Nell'entità **Pagina** la chiave primaria è il Link.
3. L'entità **modifica** non aveva attributi univoci, motivo per cui abbiamo deciso di aggiungere codMod.
4. L'entità **Utente** ha come chiave primaria la mail.
5. L'entità **Autore** ha come chiave primaria il nome d'arte.

2.7 Modello UML Ristrutturato

Schema concettuale ottenuto costruito dopo aver effettuato una Ristrutturazione del modello



2.8 Dizionario delle Entità Ristrutturato

Entità	Descrizione	Attributi
Utente	Utente loggato sulla wiki	**mail(varchar128)**: è la mail univoca dell'utente **nome(varchar128)**: è il nome dell'utente **cognome(varchar128):** è il cognome dell'utente **password(varchar128):** è la password dell'utente
Autore	è un utente in grado di creare pagine	**nomedarte(varchar128)**: è il nome d'arte univoco dell'autore **annoiniziocarriera(DATE):** è l'anno in cui l'autore si è iscritto alla wiki
Modifica	é la richiesta di modifica di una frase all'interno di una pagina	**codmod(int)**: è il codice univoco della modifica. **fraseoriginale(TEXT)**: è la frase originale che verrà modificata. **frasemodificata(TEXT)**: è la frase modificata. **stato(1, 0)**: indica se l'autore della pagina ha approvato, oppure no la modifica. **dataoramod(timestamp)**: indica la data e l'orario in cui la richiesta della modifica viene effettuata
Pagina	è la pagina creata da un autore, presente all'interno della wiki	**link(varchar128)**: è il link univoco che porta alla pagina. **titolo(varchar128)**: è il titolo della pagina. **dataoracreazione(timestamp)**: indica la data e l'orario in cui la pagina è stata creata.
Frase	è una frase all'interno di una pagina	**codfrase(int):** è il codice univoco della frase **caratteri(TEXT):** è il contenuto della frase

2.9 Dizionario delle Associazioni Ristrutturato

Associazione	Descrizione	Attributi
Collegamento	Associazione tra Frase e Pagina uno-a-molti	
effettua	Associazione tra Frase e Modifica molti-a-molti	
ha	Associazione tra Frase e Pagina uno-a-molti	
CreaPagina	Associazione tra Pagina e Autore uno-a-molti	
notifica	Associazione tra Modifica e Autore uno-a-molti	
propone	Associazione tra Modifica e Utente uno-a-molti	
e'	Associazione tra Utente e Autore uno-a-molti	

3 Modello Logico

Utente (Mail,Nome, Cognome, Password)
Autore (NomeDAarte, AnnoInizioCarriera, Utente)

Modifica (CodMod, FraseOriginale, FraseModificata, Stato, DataOra, Utente, Autore, codFrase, Pagina)
Pagina (Link, Titolo, DataOraCreazione, Autore)

Frase (CodFrase, Pagina, Caratteri)

Collegamento (codfrase, pagina-frase, pagina-riferimento)

4 Modello Fisico

4.1 Domini

StatoDominio:

```
CREATE DOMAIN StatoDominio AS INT
CHECK (VALUE IN (0, 1));
```

ReputazioneDomain:

```
CREATE DOMAIN reputazionedomain AS DOUBLE PRECISION
CHECK(VALUE >= 0 AND VALUE <= 100);
```

4.2 Creazione Tabelle

Utente:

```
CREATE TABLE Utente(
mail varchar(128) PRIMARY KEY,
nome varchar(128) NOT NULL,
cognome varchar(128) NOT NULL,
password varchar(128) NOT NULL
);
```

Autore:

```
CREATE TABLE Autore(
nomedarte varchar(128) PRIMARY KEY,
annoiniziocarriera DATE NOT NULL,
utente varchar(128) NOT NULL,
FOREIGN KEY(utente) REFERENCES Utente(mail)
);
```

Feature:

```
CREATE TABLE Feature (
    NomeF varchar(20) NOT NULL,
    Descrizione varchar(200),
    PRIMARY KEY(NomeF)
);
```

Modifica:

```
CREATE TABLE Modifica(
codmod int PRIMARY KEY,
fraseoriginale TEXT,
codfrase int NOT NULL,
pagina varchar(128) NOT NULL,
frasemodificata TEXT,
stato int default NULL,
dataoramod timestamp NOT NULL,
utente varchar(128) NOT NULL,
autore varchar(128) NOT NULL,
FOREIGN KEY(utente) REFERENCES Utente(mail),
FOREIGN KEY(autore) REFERENCES Autore(nomedarte),
FOREIGN KEY(stato) REFERENCES Stato(id),
FOREIGN KEY(codfrase, pagina) REFERENCES Frase(codfrase, pagina)
);
```

Pagina:

```
CREATE TABLE Pagina(
link varchar(128) PRIMARY KEY,
titolo varchar(128) NOT NULL,
dataoracreazione timestamp NOT NULL,
autore varchar(128) NOT NULL,
FOREIGN KEY(autore) REFERENCES Autore(nomedarte)
)
```

Frase:

```
CREATE TABLE Frase(
codfrase int NOT NULL,
caratteri TEXT,
pagina varchar(128) NOT NULL,
foreign key(pagina) references Pagina(link),
primary key(codfrase, pagina)
);
```

Collegamento:

```
CREATE TABLE Collegamento(
codfrase int,
pagina_frase varchar(128),
pagina_riferimento varchar(128),
FOREIGN KEY(codfrase, pagina_frase) REFERENCES Frase(codfrase, pagina),
FOREIGN KEY(pagina_riferimento) REFERENCES pagina(link)
)
```

4.3 Creazione Vincoli

1.
```
ALTER TABLE utente
ADD CONSTRAINT check_password
CHECK (
    password ~ '[A-Z]' AND        -- Almeno una lettera maiuscola
    password ~ '[a-z]' AND        -- Almeno una lettera minuscola
    password ~ '[0-9]' AND        -- Almeno un numero
    LENGTH(password) >= 8        -- Lunghezza minima della password
);

ALTER TABLE utente
ADD CONSTRAINT checkemail_valida
CHECK (mail LIKE '%@%.%');
```

2.
```
ALTER TABLE Autore
ADD CONSTRAINT check_annoiniziocarriera
CHECK (annoiniziocarriera >= '1900-01-01' AND annoiniziocarriera <= CURREN
```

3.
```
ALTER TABLE Modifica
ADD CONSTRAINT check_dataoramod CHECK (dataoramod <= CURRENT_TIMESTAMP);
```

4.

```
ALTER TABLE Pagina
ADD CONSTRAINT check_lunghezza_titolo CHECK (LENGTH(titolo) <= 255);

ALTER TABLE Pagina
ADD CONSTRAINT check_dataoracreazione CHECK (dataoracreazione <= CURRENT_T
```

4.4 Creazione Trigger

1.

```
CREATE OR REPLACE FUNCTION rep_aut()
RETURNS TRIGGER AS $$
DECLARE
    nomedarteDB VARCHAR(128);
    numero_di_modifiche_proposte INT;
    numero_di_pagine_realizzate INT;
    frazione_di_modifiche_accettate DOUBLE PRECISION;
    autore_rep DOUBLE PRECISION;
BEGIN
    SELECT Autore.nomedarte INTO nomedarteDB
    FROM Autore
    INNER JOIN Modifica ON Autore.utente = Modifica.utente
    WHERE Autore.utente = NEW.utente;

    SELECT COUNT(*) INTO numero_di_modifiche_proposte
    FROM Modifica m
    WHERE m.utente = NEW.utente;

    IF numero_di_modifiche_proposte > 0 THEN
 --calcolo frazione_di_modifiche_accettate che rappresenta la percentuale
 SELECT COUNT(CASE WHEN m.stato = 1 THEN 1 END) * 100.0 / COUNT(*)
 INTO frazione_di_modifiche_accettate
 FROM Modifica m
 WHERE m.utente = NEW.utente;
```

```
 IF nomedarteDB is NULL THEN
 autore_rep = frazione_di_modifiche_accettate / numero_di_modifiche_propos
 ELSE
 --calcolo in numero di pagine realizzate da un utente
 SELECT COUNT(DISTINCT p.link) INTO numero_di_pagine_realizzate
 FROM Pagina p JOIN Autore a
 ON p.autore = a.nomedarte
 JOIN Utente u
 ON a.utente = u.mail
 WHERE u.mail = NEW.utente;

 --se il numero di pagine realizzate > 0 significa che e' un autore che cr
 --se e' < 0 calcoliamo la reputazione come se fosse un nomrale utente.
 IF numero_di_pagine_realizzate > 0 THEN
 autore_rep = (frazione_di_modifiche_accettate * numero_di_pagine_realizza
 ELSE
 autore_rep = frazione_di_modifiche_accettate / numero_di_modifiche_propos
 END IF;

 END IF;

 UPDATE Utente
 SET reputazione = autore_rep
 WHERE mail = NEW.utente;

    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;


CREATE TRIGGER tr_rep_aut
AFTER INSERT OR UPDATE ON Modifica
FOR EACH ROW
EXECUTE FUNCTION rep_aut();
```

2.

```
CREATE OR REPLACE FUNCTION aggiorna_caratteri_dopo_inserimento_modifica()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.utente = (SELECT utente FROM autore WHERE nomedarte = NEW.autore
        UPDATE Frase
        SET caratteri = NEW.frasemodificata
        WHERE codfrase = NEW.codfrase AND pagina = NEW.pagina;

 UPDATE Modifica
        SET stato = 1
        WHERE codmod = NEW.codmod;

    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER aggiorna_caratteri_trigger
AFTER INSERT ON Modifica
FOR EACH ROW
EXECUTE FUNCTION aggiorna_caratteri_dopo_inserimento_modifica();
```

3.
```
CREATE OR REPLACE FUNCTION aggiorna_pagina_dopo_modifica_stato()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.stato = 1 THEN
        UPDATE Frase
        SET caratteri = NEW.frasemodificata
        WHERE codfrase = NEW.codfrase AND pagina = NEW.pagina;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER aggiorna_pagina_trigger
AFTER UPDATE ON Modifica
FOR EACH ROW
EXECUTE FUNCTION aggiorna_pagina_dopo_modifica_stato();
```