BASE DI DATI PER KALEN DARIO

Giulia Stazio, Davide Tonziello

Giugno 2025

Contents

1 Introduzione

Questo è lo sviluppo di una base di dati per un'applicazione chiamata 'Kalen Dario'. In questa applicazione un utente potrà gestire i propri impegni sotto forma di ToDo, organizzandoli in Bacheche apposite, con la possibilità di modificarli e condividerli con altri utenti.

2 Specifica dei requisiti

L'utente deve poter accedere all'applicazione con apposito Login, avrà inizialmente quattro bacheche, tra cui una di default non eliminabile, potrà modificarle in qualsiasi momento, aggiungendone anche altre fino a un massimo di dieci. Le bacheche sono dotate di titolo, descrizione, colore di sfondo e contengono i ToDo.

I ToDo sono entità dotate di proprietario, titolo, descrizione, colore di sfondo, data di scadenza, un URL associato all'impegno, stato di completamento e elenco degli utenti che condividono il ToDo. Infatti i ToDo sono condivisibili tra utenti e compaiono, una volta condivisi, nella tabella di default da cui possono essere successivamente spostati.

3 Progettazione concettuale

Il class diagram di partenza.

L'utente è identificato dal proprio Username che deve essere unico.

Bacheche e ToDo ammettono duplicati pertanto è stato necessario inserire un ID univoco.

L'utente può creare ToDo tramite le proprie bacheche. Una bacheca può contenere un numero arbitrario di ToDo e i ToDo possono appartenere a più bacheche contemporaneamente, ma non meno di una

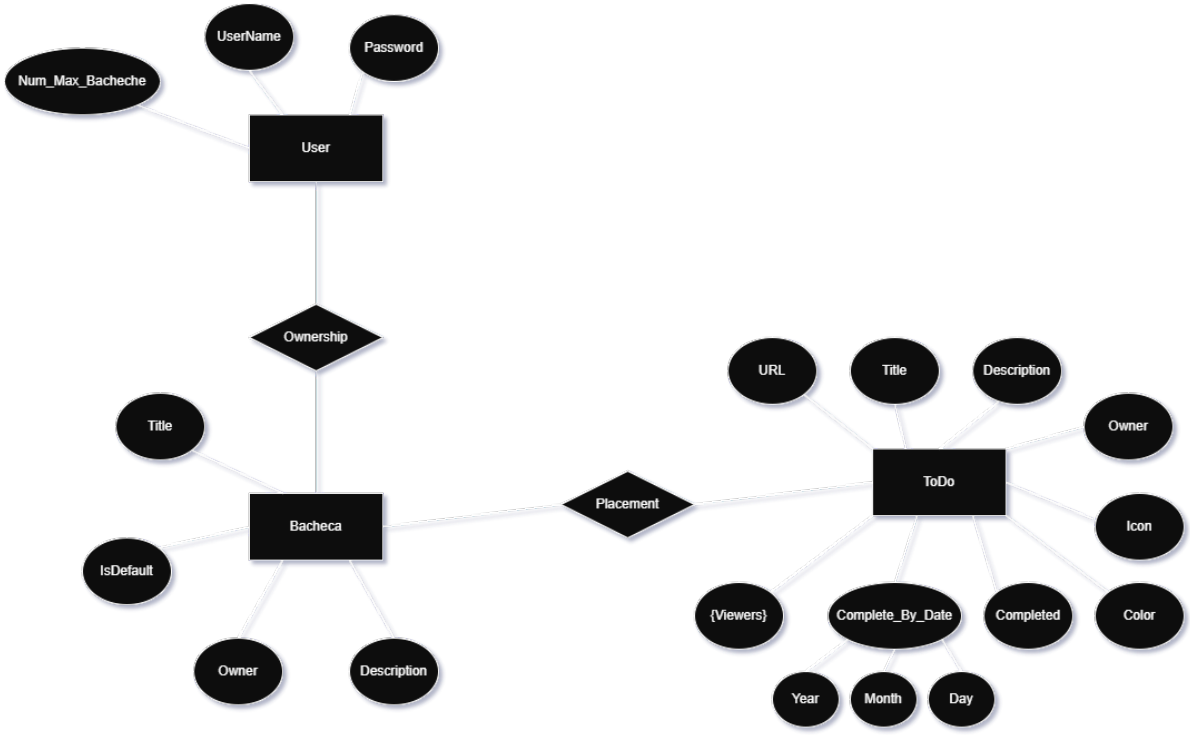


Figure 1: diagramma concettuale di partenza

4 Ristrutturazione

4.1 Analisi delle ridondanze

L'attributo Viewers dell'entità ToDo è una ridondanza, in quanto può essere ottenuto dalla relazione Placement controllando in quali bacheche si trova, inoltre è un attributo multiplo quindi sarà eliminato.

L'attributo Owner dell'entità Bacheca è un'altra ridondanza, in quanto espressa dalla relazione Ownership. Tornerà utile per collegare le tabelle User e Bacheca, per evitare di creare una tabella apposita per la relazione Ownership, pertanto non verrà eliminato

Da notare che TODO.Owner non è una ridondanza in quanto in una Bacheca sono presenti i ToDo visualizzabili dal proprietario della Bacheca, anche se non ne è il creatore. Non sono presenti altre ridondanze.

4.2 Eliminazione di: generalizzazioni, attributi composti e multivalore

Non sono presenti generalizzazioni.

L'attributo composto Complete_By_Date è rappresentabile con il tipo DATE di SQL. Non sono presenti altri attributi composti.

L'attributo multivalore Viewers verrà già rimosso in quanto ridondante. Non sono presenti altri attributi multivalore.

4.3 Partizione e accorpamento di entità e associazioni

Non è stato ritenuto necessario effettuare partizioni o accorpamenti.

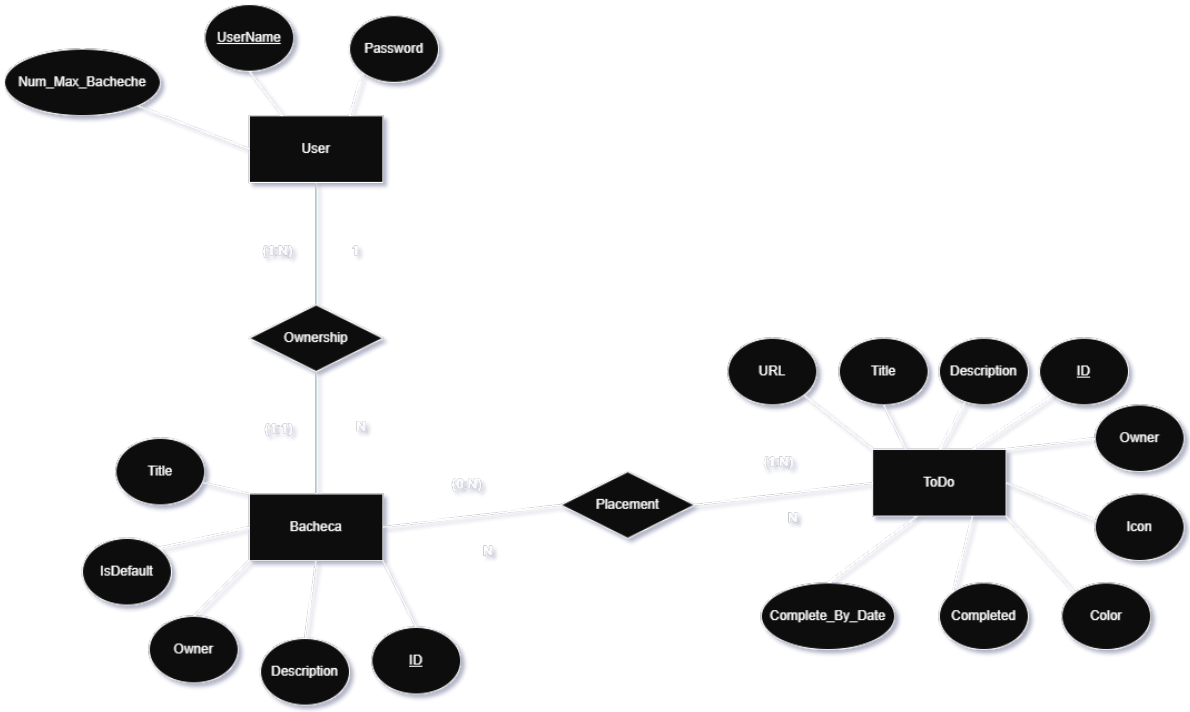
4.4 Scelta degli identificatori principali

Per l'entità User è stato scelto l'attributo Username.

Per le entità Bacheca e ToDo sono stati scelti degli ID in quanto tutti gli attributi presenti devo poter essere riutilizzati.

Ad esempio possono esistere Bacheche con lo stesso Title, Description, ecc. Lo stesso vale per un ToDo.

4.5 Diagramma ristrutturato



4.6 Dizionario classi

Nome classe	Descrizione	Attributi
User	Un utente dell'applicazione	- Username(varchar(100)): Il nome scelto dall'utente - Password(varchar(100)): La password dell'account - Num_Max_Bacheche(int): Il numero massimo di bacheche che si possono avere contemporaneamente
Bacheca	Una bacheca di un utente	- ID(int): L'identificativo univoco di una bacheca - Owner(varchar(100)): Il proprietario della bacheca - Title(varchar(100)): Il titolo della bacheca - Description(varchar(1000)): La descrizione della bacheca - IsDefault(boolean): Indica se la bacheca è quella predefinita

ToDo	I Todo creati dagli utenti	- ID(int): L'identificativo univoco di un ToDo - Owner(varchar(100)): Il creatore del ToDo - Title(varchar(100)): Il titolo del ToDo - URL(varchar(2048)): Un eventuale URL relativo al ToDo - Description(varchar(1000)): La descrizione del ToDo - Color(int): Un intero che indica quale colore è stato scelto come Background del ToDo (ai colori disponibili sono associati dei numeri da 0 a 4) - Icon(int): Un intero che indica quale icona è associata al ToDo. Le icone dei file png numerati che si trovano in una apposita cartella - Complete_By_Date(date): La data entro la quale deve (o doveva) essere stato completato il ToDo - Completed(boolean): Un flag che indica se il ToDo è stato completato

4.7 Dizionario relazioni

Nome Relazione	Descrizione	Attributi
Placement	Un utente dell'applicazione	- IDBacheca(int): L'identificativo della bacheca in cui si trova il ToDo referenziato da IDToDo - IDToDo(int): L'identificativo del ToDo contenuto nella bacheca referenziata da IDBacheca

5 Progettazione logica (modello relazionale)

ENTITÀ:

USER:(UserName, Password, Num_Max_Bacheche)
BACHECA:(ID, Description, Owner, Title) BACHECA.Owner → USER.UserName
TODO:(ID, URL, Title, Description, Owner, Icon, Color, Complete_By_Date) TODO.Owner → USER.UserName

RELAZIONI:

La relazione Ownership di Bacheca (che è una relazione 1:N) è stata implementata con la chiave esterna BACHECA.Owner. Per la relazione Placement è necessario creare una tabella in quanto è una relazione N:N.
PLACEMENT:(IDBacheca, IDToDo) PLACEMENT.IDBacheca → BACHECA.ID PLACEMENT.IDToDo → TODO.ID

6 Progettazione fisica

Di seguito è presente il codice per la creazione e gestione del DataBase, alcune considerazioni:
-Vengono utilizzate le " per termini considerati "reserved" in SQL
-Per la creazione di funzioni e trigger è utile inserire OR REPLACE dopo CREATE per facilitare l'apporto di eventuali modifiche in futuro, riutilizzando la stessa signature

6.1 Creazione delle tabelle

Tabella USER:

```
CREATE TABLE "user"
(
UserName varchar(100) PRIMARY KEY,
"Password" varchar(100) NOT NULL,
Num_Max_Bacheche INTEGER DEFAULT 10
)
```

Tabella BACHECA:

```
CREATE TABLE bacheca
(
"ID" int GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
Title varchar(100),
Description varchar(1000),
"Owner" varchar(100) REFERENCES "user"(UserName) ON DELETE CASCADE,
IsDefault boolean DEFAULT false
)
```

Il cascade su "Owner" fa in modo che all'eliminazione di una tupla dalla tabella User, vengano eliminate anche tutte le tabelle associate a quell'utente

Tabella TODO:

```
CREATE TABLE todo
(
"ID" int GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
Title varchar(100),
URL varchar(256),
Description varchar(1000),
"Owner" varchar(100) REFERENCES "user"(UserName) ON DELETE CASCADE,
Icon int DEFAULT 0,
Color int,
Complete_By_Date date,
Completed boolean
)
```

Il cascade su "Owner" qui serve per fare in modo che alla rimozione di un Utente dalla tabella User vengano eliminati anche tutti i ToDo creati da esso

Tabella PLACEMENT:

```
CREATE TABLE placement
(
IDBacheca int REFERENCES bacheca("ID") ON DELETE CASCADE,
IDToDo int REFERENCES todo("ID") ON DELETE CASCADE,
PRIMARY KEY (IDBacheca, IDToDo)
)
```

Il cascade su IDBacheca fa in modo che all'eliminazione di una bacheca, vengano rimosse anche le tuple della tabella placement che la referenziano. La stessa cosa vale per il cascade su IDToDO e l'eliminazione di un ToDo. Da notare che dopo l'eliminazione di una bacheca, non vengono rimossi automaticamente i ToDo associati dalla tabella ToDo. Di questa operazione, si occuperà un trigger apposito.

6.2 Constraints

I constraint sono stati implementati in fase di creazione delle tabelle con le operazioni CASCADE, vincoli di integrità (predefiniti) e trigger.

6.3 Trigger

Il seguente trigger viene chiamato ogni volta che si effettua una rimozione dalla tabella Placement

```
CREATE OR REPLACE TRIGGER removeToDo AFTER DELETE ON placement
FOR EACH ROW
EXECUTE FUNCTION removeToDoFunction();
```

Questo trigger viene attivato all'inserimento di un nuovo utente nella tabella User

```
CREATE OR REPLACE TRIGGER createDefaultBacheche AFTER INSERT ON
    "user"
FOR EACH ROW
EXECUTE FUNCTION createDefaultBachecheFunction();
```

6.4 Functions

Questa funzione verifica, in seguito ad una rimozione di un elemento dalla tabella Placement, se l'utente ad aver rimosso il ToDo ne è il creatore. In caso affermativo, lo elimina anche dalla tabella dei ToDo.

```
CREATE OR REPLACE FUNCTION removeToDoFunction ()
RETURNS trigger
as $$

DECLARE
temporaneo1 varchar(100);
temporaneo2 varchar(100);

BEGIN
Select "Owner" into temporaneo1
From bacheca
where bacheca."ID" = OLD.IDBacheca;

Select "Owner" into temporaneo2
From todo
Where todo."ID" = OLD.IDToDo;

IF temporaneo1 LIKE temporaneo2 THEN
Delete from todo where todo."ID" = OLD.IDToDo;
END IF;
RETURN OLD;
END;
$$ language plpgsql;
```

Questa funzione inserisce le quattro bacheche di partenza di un utente

```
CREATE OR REPLACE FUNCTION createDefaultBachecheFunction()
    RETURNS trigger
as $$
BEGIN
INSERT INTO bacheca (title, description, "Owner", isDefault)
values
('Default', 'your default bacheca', new.username, true),
('Università', 'your university bacheca', new.username, false),
```

```
8           ('Lavoro', 'your job bacheca', new.username, false),
9           ('Tempo libero', 'your free time bacheca', new.username, false);
10          RETURN NULL;
11          END;
12          $$ language plpgsql;
```