ROLLO – RIDE YOUR WAY

Gruppo 1: Duluta, Manfo, Marcolin, Fedorenco

Documentazione tecnica del sistema informativo per la gestione di un servizio di bike sharing basato su una piattaforma web.
Viene descritto il processo di progettazione della base di dati, dall'analisi del problema alla normalizzazione dello schema, fino all'implementazione in SQL e alla definizione delle query per l'interazione con l'applicazione.

Gruppo 1: Duluta, Manfo, Marcolin, Fedorenco

ROLLO – RIDE YOUR WAY

Documentazione tecnica per la parte informatica

1. Analisi del problema

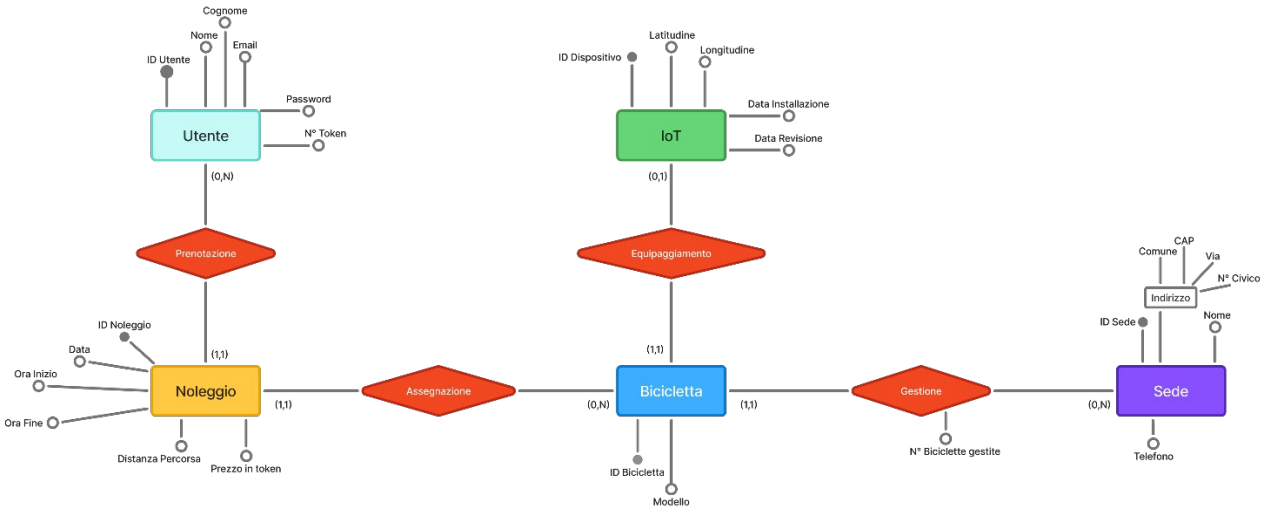
Il progetto "Rollo - Ride your way" nasce con l'obiettivo di realizzare una web app per il noleggio e la gestione di biciclette.
La base di dati che si appoggia alla web app deve permettere la registrazione e gestione degli utenti, la localizzazione delle sedi (che gestiscono un certo numero di biciclette e si occupano della loro revisione), il controllo dei dispositivi IoT installati su ogni bicicletta e la tracciabilità dei noleggi effettuati.

Ogni noleggio sarà registrato con informazioni relative a:

- data e orario di inizio/fine
- distanza percorsa
- costo (in token)
- bicicletta utilizzata
- utente che ha effettuato il noleggio

Si è scelto di utilizzare dei token per i pagamenti in quanto la connessione ad un servizio per i pagamenti bancari come "Nexi" porterebbe con sé delle complicanze realizzative rilevanti, come ad esempio l'obbligo di registrazione dell'attività all'Agenzia delle Entrate. I token saranno guadagnabili attraverso il "Bonus di Benvenuto", il bonus "Droppa una story con Rollo" (un contenuto sui social che promuova il brand) e le eventuali pubblicità facoltative, se si riuscirà ad implementarle.

2. Proposta iniziale di schema E-R



Gruppo 1: Duluta, Manfo, Marcolin, Fedorenco

3. Motivazione delle scelte iniziali

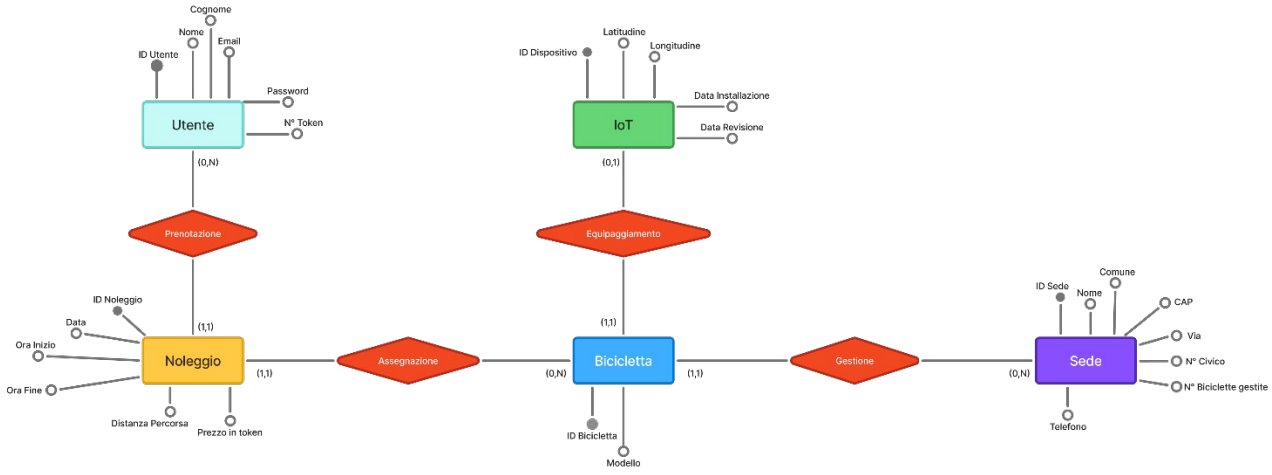
Lo schema iniziale è stato progettato tenendo conto dei seguenti criteri:

- Separazione chiara tra entità distinte: Utenti, Biciclette, Noleggi, Sedi e IoT.
- Ogni entità ha un suo identificatore singolo interno, sufficiente ad indivituare in modo univoco ogni sua istanza.

1. **Utente**
   Id_Utente, nome e cognome, email, password e numero di token collegati all'account.
   Un utente può prenotare uno o più noleggi.
2. **Noleggio**
   Id_Noleggio, data, ora inizio e ora fine, distanza percorsa e prezzo in token.
   Un noleggio è prenotato da un solo utente ed ha soltanto una bicicletta assegnata.
3. **Bicicletta**
   Id_Bicicletta, modello.
   Una bicicletta può essere assegnata ad uno o più noleggi. Equipaggia una sola scheda IoT ed è gestita da una sola Sede.
4. **Sede**
   Id_Sede, nome (può avere un nome in codice con cui è riconosciuta più facilmente tra i dipendenti), Indirizzo (**attributo composto** da comune, CAP, via e numero civico) e contatto telefonico.
   Nota: l'associazione tra Bicicletta e Sede presenta un attributo che costituisce il numero di biciclette gestite dalla singola sede.
5. **IoT**
   Id_Dispositivo, latitudine e logitudine, data dell'installazione e data della prossima revisione programmata.

Questa struttura riflette la realtà operativa di un sistema di bike sharing e consente una gestione flessibile ed espandibile del servizio.

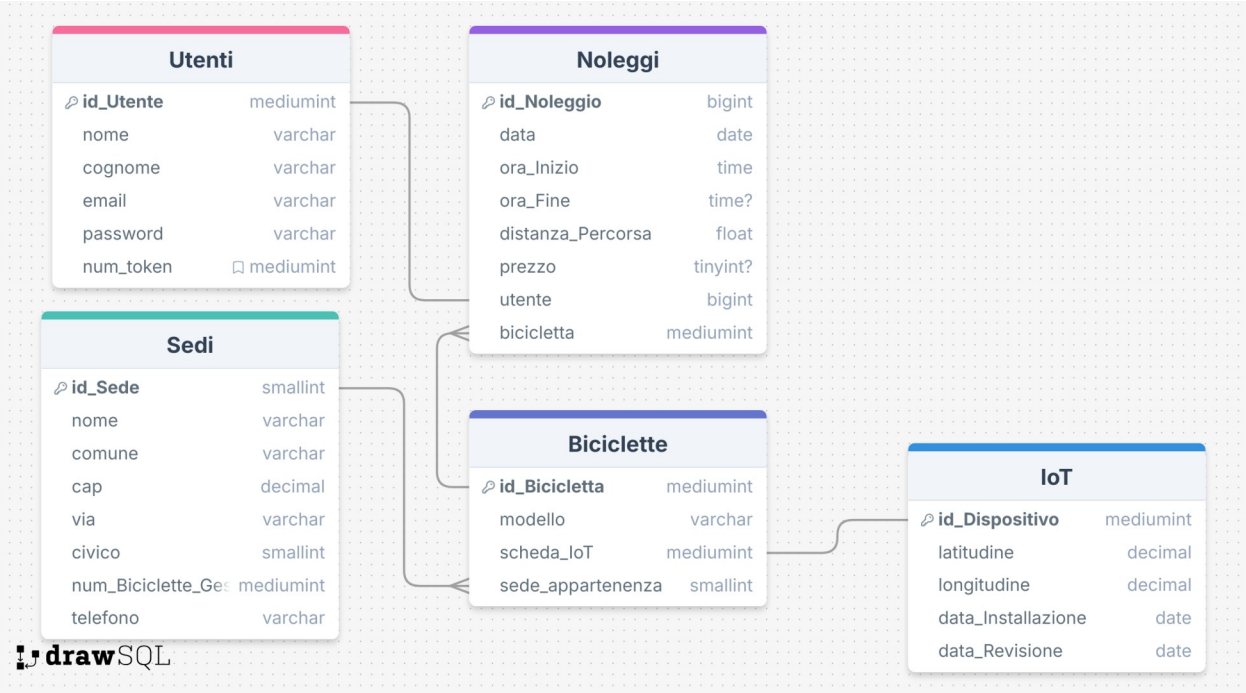
4. Schema E-R ristrutturato



Nella ristrutturazione dello schema E-R vengono eliminate le generalizzazioni (in questo caso non sono state previste) e gli attributi delle associazioni. Inoltre, gli attributi composti vengono separati (in attributi singoli) nel caso in cui non è efficace l'archiviazione di tutti i dati che ne fanno parte in una sola tupla.
Vengono applicate le seguenti modifiche:

- L'attributo "N° Biciclette gestite" dell'associazione "Gestione" diventa attributo di "Sede".
- L'attributo composto "Indirizzo" di "Sede" viene scomposto negli attributi "comune", "cap", "via" e "n° civico".

5. Traduzione a schema logico e normalizzazione



Per controllare se lo schema prodotto rispetta le regole della normalizzazione, ogni entità è stata convertita in una tabella con attributi atomici e chiavi primarie ben definite. I nomi delle entità diventano al plurale.

Utenti(id_Utente, nome, cognome, email, password)

Noleggi (id_Noleggio, data, ora_Inizio, ora_Fine, distanza_Percorsa, prezzo, *utente*, *bicicletta*)

Biciclette (id_Bicicletta, modello, *scheda_IoT*, *sede_appartenenza*)

IoT (id_Dispositivo, latitudine, longitudine, data_Installazione, data_Revisione)

Sedi (id_Sede, nome, comune, cap, via, civico, num_Biciclette_Gestite, telefono)

Viene di seguito descritto il confronto con le principali forme normali studiate.

- **1NF** ✅**:** Tutti gli attributi sono atomici. Non ci sono valori multipli o campi ripetuti.
- **2NF** ✅**:** Ogni attributo non chiave è completamente dipendente dalla chiave primaria. Nessuna tabella ha chiavi composite.
- **3NF** ✅**:** Tutte le dipendenze funzionali sono dirette: non ci sono attributi che dipendono da altri attributi non-chiave.
- **Forma Normale di Boyce-Codd (FNBC)** ✅**:**
  La FNBC è una forma normale più rigorosa della 3NF. Una tabella è in FNBC se per ogni dipendenza funzionale X→Y, X è una superchiave.

Nello schema utilizzato:
Tutte le dipendenze funzionali hanno come determinante una chiave o superchiave.
~ Lo schema è dunque **normalizzato in FNBC**.

Gruppo 1: Duluta, Manfo, Marcolin, Fedorenco

6. Vincoli di integrità

- **Chiavi primarie**
  - Utenti(id_Utente)
  - Sedi(id_Sede)
  - Biciclette(id_Bicicletta)
  - IoT(id_Dispositivo)
  - Noleggi(id_Noleggio)
- **Chiavi esterne**
  - Noleggi.utente → Utenti.id_Utente
  - Noleggi.bicicletta → Biciclette.id_Bicicletta
  - Biciclette.scheda_IoT → IoT.id_Dispositivo
  - Biciclette.sede_appartenenza → Sedi.id_Sede
- **Vincoli sui dati**
  - prezzo ≥ 0
  - distanza_Percorsa ≥ 0
  - ora_Fine ≥ ora_Inizio (se ora_Fine è presente)
  - data_Revisione ≥ data_Installazione

7. Implementazione SQL

✅ CREATE DATABASE

```
CREATE DATABASE Rollo;
USE Rollo;
```

✅ CREATE TABLE

- Utenti

```
CREATE TABLE Utenti (
    id_Utente MEDIUMINT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    cognome VARCHAR(100) NOT NULL,
    email VARCHAR(150) UNIQUE NOT NULL,
    password VARCHAR(100) NOT NULL,
    num_token MEDIUMINT DEFAULT 100  NOT NULL
);
```

Gruppo 1: Duluta, Manfo, Marcolin, Fedorenco

- Sedi

```
CREATE TABLE Sedi (
    id_Sede SMALLINT PRIMARY KEY,
    nome VARCHAR(100),
    comune VARCHAR(100) NOT NULL,
    cap DECIMAL(5,0) NOT NULL,
    via VARCHAR(100) NOT NULL,
    civico SMALLINT,
    num_Biciclette_Gestite MEDIUMINT DEFAULT 0,
    telefono VARCHAR(20) NOT NULL
);
```

- IoT

```
CREATE TABLE IoT (
    id_Dispositivo MEDIUMINT PRIMARY KEY,
    latitudine DECIMAL(9,6) NOT NULL,
    longitudine DECIMAL(9,6) NOT NULL,
    data_Installazione DATE NOT NULL,
    data_Revisione DATE NOT NULL,
    CHECK (data_Revisione >= data_Installazione)
);
```

- Biciclette

```
CREATE TABLE Biciclette (
    id_Bicicletta MEDIUMINT PRIMARY KEY,
    modello VARCHAR(100) NOT NULL,
    scheda_IoT MEDIUMINT,
    sede_appartenenza SMALLINT,
    FOREIGN KEY (scheda_IoT) REFERENCES IoT(id_Dispositivo),
    FOREIGN KEY (sede_appartenenza) REFERENCES Sedi(id_Sede)
);
```

- Noleggi

```
CREATE TABLE Noleggi (
    id_Noleggio BIGINT PRIMARY KEY,
    data DATE DEFAULT NOW(),
    ora_Inizio TIME NOT NULL,
    ora_Fine TIME,
    distanza_Percorsa FLOAT CHECK (distanza_Percorsa >= 0),
    prezzo DECIMAL(5,0) CHECK (prezzo >= 0),
    utente BIGINT,
    bicicletta MEDIUMINT,
    FOREIGN KEY (utente) REFERENCES Utenti(id_Utente),
    FOREIGN KEY (bicicletta) REFERENCES Biciclette(id_Bicicletta),
    CHECK (ora_Fine IS NULL OR ora_Fine >= ora_Inizio)
);
```

Gruppo 1: Duluta, Manfo, Marcolin, Fedorenco

✅ INSERT (esempi con dati fittizi per il controllo del corretto funzionamento)

```sql
INSERT INTO Utenti VALUES (1, 'Marco', 'Rossi', 'marco.rossi@mail.com', 'password123', 3);

INSERT INTO Sedi VALUES (1, 'Sede Centrale', 'Milano', 20100, 'Via Roma', 1, 10, '0245789632');

INSERT INTO IoT VALUES (101, 45.4654, 9.1859, '2024-01-01', '2025-01-01');

INSERT INTO Biciclette VALUES (2001, 'Model X', 101, 1);

INSERT INTO Noleggi VALUES (30001, '2025-04-01', '08:00:00', '09:30:00', 12.5, 3, 1, 2001);
```

8. Viste utili per la web app

- Noleggi con dati utente e bicicletta

```sql
-- VISTA: Noleggi con dati utente e bicicletta
CREATE VIEW Vista_Noleggi_Completa AS
SELECT N.id_Noleggio, N.data, N.ora_Inizio, N.ora_Fine, N.prezzo,
       U.nome, U.cognome, B.modello
FROM Noleggi N
JOIN Utenti U ON N.utente = U.id_Utente
JOIN Biciclette B ON N.bicicletta = B.id_Bicicletta;
```

- Stato biciclette e ultima posizione

```sql
-- VISTA: Stato biciclette e ultima posizione
CREATE VIEW Vista_Stato_Biciclette AS
SELECT B.id_Bicicletta, B.modello, I.latitudine, I.longitudine, S.nome AS sede
FROM Biciclette B
JOIN IoT I ON B.scheda_IoT = I.id_Dispositivo
JOIN Sedi S ON B.sede_appartenenza = S.id_Sede;
```

9. Query SELECT previste per la web app

```sql
-- Tutti i noleggi effettuati da un utente specifico
SELECT * FROM Vista_Noleggi_Completa WHERE nome = 'Marco' AND cognome = 'Rossi';

-- Biciclette disponibili in una sede
SELECT id_Bicicletta, modello
FROM Biciclette
WHERE sede_appartenenza = 1;

-- Biciclette che devono essere revisionate entro una certa data
SELECT B.id_Bicicletta, I.data_Revisione
FROM Biciclette B
JOIN IoT I ON B.scheda_IoT = I.id_Dispositivo
WHERE I.data_Revisione <= '2025-06-01';
```

10. Conclusioni

Il progetto **Rollo - Ride your way** ha permesso di progettare e documentare in modo completo la base di dati per un sistema di bike sharing. Attraverso l'analisi del problema, la progettazione concettuale e logica, la normalizzazione fino alla forma normale di Boyce-Codd (FNBC), e l'implementazione in SQL, è stato costruito uno schema solido e coerente, pronto per l'integrazione in una web app funzionale.

In futuro, il sistema potrà essere esteso per includere:

- statistiche sull'utilizzo delle biciclette,
- sistema di prenotazione anticipata,
- gestione di account premium e abbonamenti,
- interfacciamento in tempo reale con i dispositivi IoT per il tracciamento live.

Il lavoro rappresenta una base solida sia a livello teorico che pratico per un'applicazione reale e scalabile.