**1. Introduzione**

Questo documento descrive nel dettaglio in DBMS che si è scelti di utilizzare e tutta la logica riguardante la gestione dei dati persistenti.

**1.1 Scelta del DBMS**

Il problema della persistenza dei dati nasce dalla necessità di rendere permanenti alcune informazioni all’interno di un sistema anche quando questo è spento. Molte volte capita di avere dei programmi con un gran numero di dati da gestire, dati che non possono essere reinseriti dopo un blackout o dopo lo spegnimento della macchina; si pensi ad un archivio di mille utenti contenente tutti i dati anagrafici. Sarebbe impensabile il reinserimento degli stessi ogni volta che la macchina sul quale risiedono i dati si spegne. Bisogna, quindi, trovare il modo per poter tener traccia dei dati utili anche quando l’applicazione che li ha utilizzati e/o creati ha smesso di funzionare.

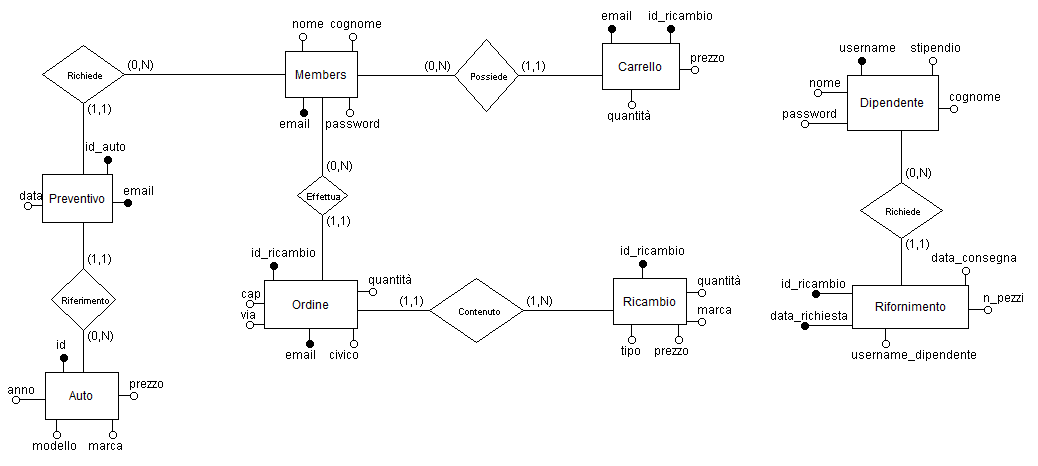
I dati persistenti sono sempre lì e vengono cancellati solo se lo si richiede espressamente. Nel caso specifico, in previsione del gran numero di accessi al database e alla necessità di gestire anche più queries contemporaneamente, si è optati per un DBMS relazionale, MySql.

MySql è il DBMS Open Source più diffuso ed utilizzato al mondo. La sigla SQL presente all'interno del nome sta a indicare che si tratta di un DBMS in grado di interpretare le istruzioni del linguaggio SQL (utile per le query). MySql permette la creazione di “database relazionali” ossia consente la conservazione dei dati in tabelle separate anziché in un unica grande entità. Questa sua particolare caratteristica consente di raggiungere una buona flessibilità e velocità di accesso ai dati ed una maggior modellazione delle basi dati. Semplicità d'uso, robustezza e velocità di esecuzione sono le caratteristiche principali di questo DBMS. La sua struttura multi-thread consta di un thread fisso che controlla le connessioni in ingresso e un thread attivo per ogni connessione. I vari client non devono aspettare che le queries di altri siano soddisfatte, possono lavorare simultaneamente. Il thread che controlla le connessioni impedisce che due thread scrivano sulla stessa tabella nello stesso momento. La velocità ottenuta utilizzando un motore multi-thread ripaga la perdita dovuta alla maggiore complessità di gestione e progettazione del sistema. In ogni tabella all’interno del nostro database sarà possibile accedere ai dati per compiere operazioni quali inserimento, modifica, cancellazione o semplicemente consultazione.

**2. Progettazione del database**

**2.1 Gestione dei dati persistenti**

**2.1.1 Modello ER**



**2.1.3 Dizionario dei dati**

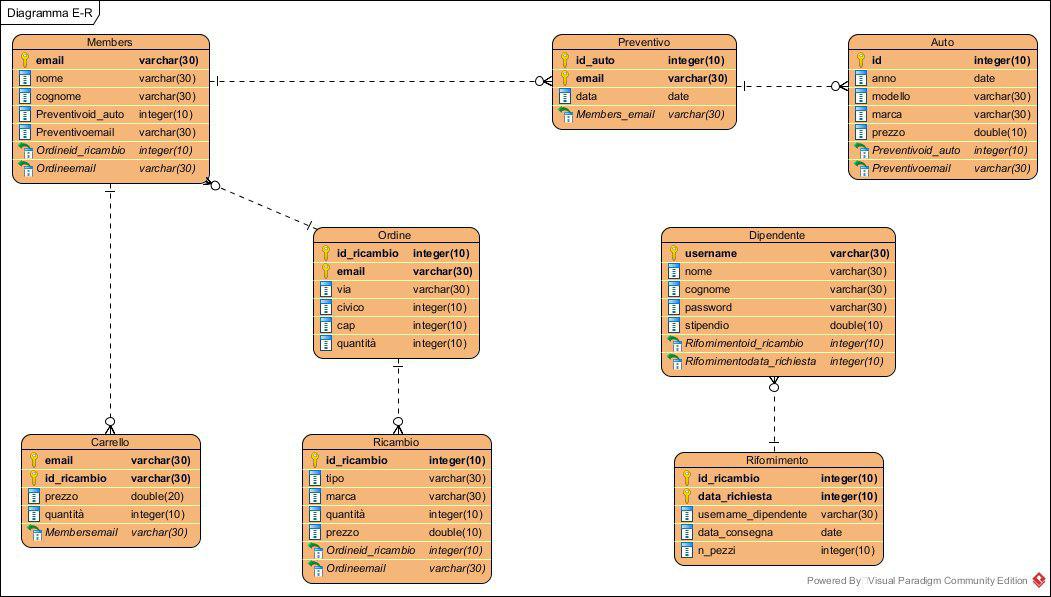
**2.1.3.1 Entità**

|  |  |  |  |
| --- | --- | --- | --- |
| **Entita’** | **Descrizione** | **Attributi** | **Identificatore** |
| Members | |  | | --- | | Contiene le informazioni riguardanti i clienti. | | Email  Nome  Cognome  Password | Email |
| Carrello | Contiene le informazioni riguardanti il carrello. | Email  Id\_ricambio  Prezzo  Quantità | (Email,ID\_ricambio) |
| Ordine | Contiene le informazioni riguardanti l’ordine. | ID\_ricambio  Email  Cap  Via  Civico  Quantità | (Email,ID\_ricambio) |
| Ricambio | Contiene le informazioni riguardanti i ricambi. | ID\_ricambio  Quantità  Marca  Tipo  Prezzo | ID\_ricambio |
| Preventivo | Contiene le informazioni riguardanti i preventivi. | ID\_auto  Email  Data | (ID\_auto, Email) |
| Auto | Contiene le informazioni riguardanti le auto. | ID\_auto  Anno  Prezzo  Modello  Marca | ID\_auto |
| Dipendente | Contiene le informazioni riguardanti i dipendenti | Username  Password  Nome  Cognome  Stipendio | Username |
| Rifornimento | Contiene le informazioni riguardanti i rifornimenti | ID\_ricambio  Data\_richiesta  Data\_consegna  Username\_dipendente  N\_pezzi | Data\_richiesta  (ID\_ricambio) |

**2.1.3.2 Relazioni**

|  |  |  |
| --- | --- | --- |
| **Relazione** | **Descrizione** | **Entità coinvolte** |
| Possiede | Relazione tra Members e Carrello | Members (0,N)  Carrello (1,1) |
| Effettua | Relazione tra Members e Ordine | Members (0,N)  Ordine (1,1) |
| Contenuto | Relazione tra Ordine e Ricambio | Ordine (1,1)  Ricambio (1,N) |
| Richiede | Relazione tra Members e Preventivo | Members (0,N)  Preventivo (1,1) |
| Riferimento | Relazione tra Preventivo e Auto | Preventivo (1,1)  Auto (0,N) |

**2.1.4 Modello logico**



**3. Descrizione del Database**

**Tabella: Members**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attributo** | **Tipo** | **Descrizione** | **Opzionale** | **Chiave** |
| Email |  |  |  |  |
| Nome |  |  |  |  |
| Cognome |  |  |  |  |
| Preventivo |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |