ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

CAMPUS DI CESENA

DIPARTIMENTO DI INGEGNERIA DELL'ENERGIA ELETTRICA E DELL'INFORMAZIONE "GUGLIELMO MARCONI"

Corso di Laurea in Ingegneria Biomedica

STUDIO E SPERIMENTAZIONE DI APACHE SUPERSET IN UNO SCENARIO DI MANUTENZIONE PREVENTIVA

Elaborato in

Calcolatori Elettronici

Relatore

Prof. LUCA ROFFIA

Presentata da

GIORGIO ALLEGRI

Correlatore

SIMONE PERSIANI

Anno Accademico 2019/2020

ABSTRACT

La gestione di *device* da remoto è esistente, ma ancora in via di sviluppo. In questo elaborato si espone il progetto che ha come obiettivo un modo innovativo per monitorare da remoto un grande insieme di trasformatori elettrici. Per analizzare i *big data* in uscita, è stata utilizzata un'applicazione ancora in via di sviluppo da parte dei programmatori del gruppo Apache: Superset. Lo scopo principale della tesi è quello di creare un'applicazione contenente un insieme di *slices* e *dashboards* per la visualizzazione degli elementi rilevati dalle centraline. Vengono gestiti anche gli accessi dei vari utenti, che hanno diversi permessi a seconda del ruolo che ricoprono. Quest'ultimo aspetto permette di adattare la visualizzazione e quindi le dashboard mostrate sulla base dell'utente.

Sommario

INTRODUZIONE

Negli ultimi anni, lo sviluppo di applicazioni di Business Intelligence è diventato sempre più fondamentale per la crescita delle grandi aziende; vengono infatti utilizzate per analizzare e visualizzare i dati che provengono da ogni settore e che sono, di conseguenza, molto eterogenei; sono quindi necessarie per i CIOs (Chief Information Officers) delle imprese per catalogare tutti i dati in ingresso ed organizzare i dati in uscita in modo che siano facilmente interpretabili. Tutto questo è essenziale per agevolare le scelte che vengono poi effettuate dai CEOs (Chief Executive Officers) delle società.

Metodi di visualizzazione e di analisi dei dati (BI): lo stato dell'arte

Con Business Intelligence (BI) si intendono quei processi che prelevano dati grezzi e li trasformano in informazioni utili per una più efficace visione strategica, operativa e decisionale, in modo da produrre reali benefici per il business. Inoltre, questa tecnica svolge un ruolo molto importante per colmare la carenza di una forte connessione tra i sistemi aziendali e l'informatica industriale. Dato che le imprese devono elaborare una quantità sempre maggiore di elementi, con questa tecnologia possono estrarre in modo efficiente delle informazioni utili da vaste moli di dati (conosciuti come Big Data), in modo da prendere importanti decisioni in meno tempo, e aumentare così il valore dell'azienda. Dal 2011 [1], l'utilizzo della BI è in continuo aumento perché è sempre più importante per le aziende per rimanere competitive e poiché fornisce molte opportunità di ricerca.

La Business Intelligence comprende le categorie di:

- data warehouse: è il contenitore dei dati storici che generalmente derivano da una vasta gamma di origini disparate. È progettato per abilitare e supportare le attività di BI;
- On-Line Analytical Processing (OLAP): insieme di tecniche software per l'analisi interattiva e veloce di grandi quantità di dati, che è possibile esaminare in modalità piuttosto complesse;
- data mining (estrazione di dati): insieme di tecniche e metodologie che hanno per oggetto l'estrazione di informazioni utili da grandi quantità di dati. [1]

Il data warehouse è uno strumento necessario per il data mining, che rende possibile la ricerca di relazioni tra i dati in grandi insiemi di elementi, rilevando relazioni precedentemente sconosciute.

Tra le più importanti applicazioni di BI ci sono l'Executive Information System (EIS), il Customer Relationship Management (CRM) ed il Corporate Performance Management (CPM):

- gli EISs sono utilizzati per trasformare notevoli volumi di dati primari, generati dai processi aziendali di base, in strutture logiche che rappresentano i processi di gestione e di decisione in azienda;
- il CRM è una modalità di comportamento dell'azienda nei confronti dei propri clienti, una strategia mirata alla soddisfazione delle esigenze del cliente;
- il CPM si basa sull'idea dell'integrazione degli strumenti di BI a supporto della pianificazione aziendale. Tutte le metodologie, i metodi e gli strumenti informativi necessari per monitorare e gestire le performance di un'azienda o di un ente sono parte del CPM. [2]

Attualmente, le applicazioni di business intelligence per la visualizzazione dei dati più utilizzate sono Microsoft Power BI [3], Tableau [4], Google Data Studio [5]; tra le open-source si possono elencare Superset [6], Metabase [7] e Redash. [8]; [9].

Il caso di studio: Apache Superset

Il gruppo Apache Software Foundation (ASF), creatore di programmi come Apache HTTP Server [12] e Apache OpenOffice [13], ha sviluppato una di queste app per la BI: Apache Superset. È ancora in fase di sviluppo, perciò viene continuamente aggiornata con nuove funzionalità dagli sviluppatori di Apache Incubator, che è il gateway per i progetti open source destinati a diventare di Apache Software Foundation [14] a pieno titolo.

Apache Superset è un'applicazione web creata per andare incontro ai bisogni delle grandi aziende nell'analisi e nella visualizzazione di grandi moli di dati. Grandi compagnie come Twitter e Airbnb, che devono analizzare e visualizzare enormi quantità di dati in ogni momento, utilizzano e contribuiscono alla crescita ed allo sviluppo di Superset. [10], [11]

In questa tesi verranno mostrate e spiegate molte delle funzionalità che offre Apache Superset, fornendo un esempio pratico applicato ad una situazione di realtà simulata. Sono state inserite delle temperature rilevate dalle sonde di alcune centraline. Ogni centralina è gestita da un utente che, accedendo all'applicazione, ha la possibilità di visualizzare vari grafici che offrono diversi modi per interpretare i dati.

Quindi nel primo capitolo verrà descritto Apache Superset e le sue funzionalità; nella seconda sezione verrà minuziosamente esposto lo scenario preso in esame e, infine, nella terza parte verrà raccontata la creazione delle *dashboard*.

1. CAPITOLO 1 - APACHE SUPERSET

Figura 1: logo di Apache Superset

Creato ed inizialmente sviluppato da Max Beauchemin, Superset è nato nel 2015 come un progetto open-source di Airbnb per la visualizzazione dei dati. L'applicazione è cresciuta molto rapidamente, superando in fretta Tableau come principale soluzione di Airbnb per l'analisi dei dati. Nel 2016 è entrato a far parte di Apache Incubator, un programma di sviluppo portato avanti dalla Apache Software Foundation (ASF). Ad oggi, Superset è una delle piattaforme open-source leader per lo studio e la visualizzazione dei dati, pronta per le grandi imprese come Airbnb e Twitter.

Come descritto sul sito, Apache Superset è un'applicazione web di business intelligence pronta per le aziende. Ha la particolarità di essere intuitiva e perciò può essere utilizzata anche da utenti non esperti nella programmazione, ma dà anche l'opportunità ad utenti più esperti di approfondire le funzionalità ed ampliare lo spettro di opzioni possibili [10], [11].

1.1. Installazione di Apache Superset

1.1.1. I primi problemi

Inizialmente è stato installato il container Docker [16] con Superset da GitHub, seguendo la guida fornita da Apache Superset. GitHub [11] è una piattaforma di hosting di codici, che permette di lavorare insieme ad altri programmatori da ogni luogo. Dopo aver effettuato il download dell'immagine desktop di GitHub, è stato scaricato, in locale, l'applicazione di Docker e, dopo, sono stati inseriti sul terminale i seguenti comandi:

```
git clone https://github.com/apache/incubator-superset/

cd incubator-superset

docker-compose up
```

Si installa così l'immagine Docker di Superset. Si entra poi nel browser e, digitando:

http://localhost:8088

si apre la schermata di accesso di Superset. [Figura 2]

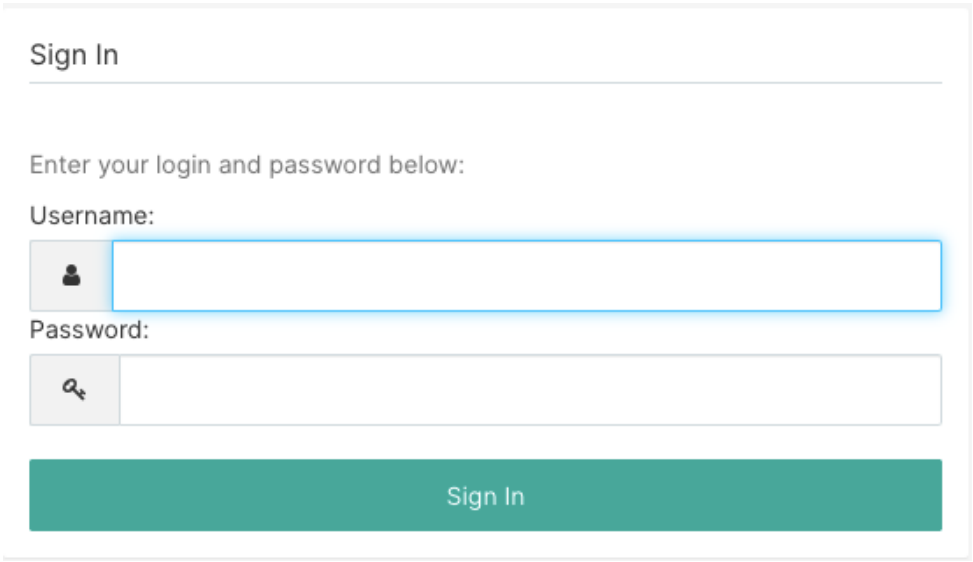


Figura 2 schermata di accesso di Superset.

Inserendo "admin" sia nel campo "Username" che in "Password", si viene indirizzati nella pagina iniziale di Superset, che mostra tutte le dashboard disponibili on-line e quelle create dall'utente in locale.

Per poter lavorare sui propri database bisogna aggiungere una sorgente dal menù "Sources"; in questo caso è stato utilizzato MySQL [15] per la creazione delle tabelle e della base di dati.

A questo punto sono iniziati i problemi di connessione al driver di MySQL ("mysqlclient"), che doveva essere installato. Dopo aver scaricato il driver giusto, il programma incorreva in un altro tipo di errore quando provavo la connessione al database – non riusciva a collegarsi al server attraverso un "socket" – e così si è deciso di cambiare il metodo di installazione.

1.1.2. La soluzione

L'ostacolo era dato da un problema legato all'installazione con Docker, nel senso che, oltre all'immagine di Superset, ne venivano installate altre con diverse funzionalità che entravano in conflitto con la connessione al server locale. Perciò, dopo aver disinstallato l'immagine di Superset e pulito le Directory da tutti i file Docker in eccesso, si è deciso di reinstallarlo in locale seguendo la guida per Mac OS X, evitando così di utilizzarlo nella sua immagine Docker.

Alla fine del processo di installazione, per entrare in Superset è sempre necessario attivare un ambiente virtuale Python:

```
. venv/bin/activate
```

Ora è possibile far partire l'applicazione, specificando la porta di accesso e le modalità di esecuzione.

superset run -p 8088 --with-threads --reload --debugger

Per entrare nell'applicazione dal browser bisogna digitare di nuovo:

http://localhost:8088 :

- "localhost" è l'indirizzo IP del computer locale;
- "8088" è la porta di accesso al server. MySQL, ad esempio, accede alla porta 3306, altre applicazioni web che lavorano con i server, entrano con altre porte.

Così facendo, ora è possibile lavorare con Superset in locale, senza alcun intoppo. È stata, quindi, connessa l'applicazione al database di MySQL che era stato creato in precedenza ed è iniziato il viaggio alla scoperta delle (quasi) infinite potenzialità di Superset. [10]

1.2. Le funzionalità di Apache Superset

Dal menù principale di Superset si può navigare tra le diverse pagine:

- **Security** - In questa sezione è possibile assegnare e modificare i ruoli degli utenti che accedono. Di default sono presenti quattro ruoli nel sistema: *Admin, Alpha, Gamma, Public, granter, sql_lab.*
  Il ruolo di ***Admin*** ha tutti i diritti possibili: può quindi modificare i ruoli degli altri utenti, ed inoltre ha la possibilità di cambiare gli *slices* e le *dashboards* di chiunque effettui l'accesso.
  Gli utenti ***Alpha*** hanno l'accesso a tutte le sorgenti di dati, ma non possono garantire o revocare i permessi degli altri utenti. Inoltre, possono modificare solo le *dashboard* e gli *slices* che possiedono. A ***Gamma*** è permesso visualizzare solo i dati, gli *slices* e le *dashboard* a cui hanno accesso, e non hanno la possibilità di modificare le sorgenti degli elementi; però possono creare e modificare le *dashboard*.
  Il ruolo ***SQL Lab*** può accedere solo alla sezione "SQL Lab", a differenza degli utenti *Alpha* e *Gamma.* Gli utenti ***Public*** sono coloro che hanno il permesso di accedere da "esterni", perciò hanno funzionalità limitate.

- **Manage** - Dal *Manage* è possibile importare *dashboard* da altri file e visualizzare tutte le *query* che sono state eseguite precedentemente. Inoltre, modificando il file di configurazione di Superset, installato insieme all'applicazione, è possibile impostare l'invio di report via e-mail, con la possibilità di visualizzare le *dashboard* e i charts.

- **Sources** - Nel menù *Sources* si possono selezionare le sorgenti dei dati da lavorare, che possono essere tabelle o dashboards; inoltre, il sistema offre l'opportunità di caricare direttamente dei file CSV. Nella pagina *Dashboard* è presente un bottone che permette di aggiungere una nuova sorgente per creare una nuova dashboard; nel caso in esame era necessario connettersi al server di MySQL.

- **Charts** – In questa sezione sono elencati tutti i *Charts* – i grafici – che sono stati creati.

- **Dashboards** - Nella pagina *Dashboards* viene visualizzato la lista delle dashboard, comprese quelle caricate on-line e quelle salvate in locale. Sono tutte visibili se si ha il permesso di *admin*, altrimenti si possono vedere solo quelle di cui si è il proprietario, o solo i dati a cui si può avere accesso.

- **SQL Lab** - L'applicazione comunica con i database attraverso il linguaggio SQL, è quindi presente una sezione del menù che permette di lavorare con le "query" proprie di questo codice: "SQL Lab". In questa parte di Superset è necessario dapprima selezionare il database al quale ci si vuole connettere, per poi scegliere le tabelle che si vuole prendere in esame. Una volta effettuati questi primi passi si possono scrivere le query per selezionare le colonne che interessano. Da questo momento si possono esplorare i grafici che meglio descrivono ciò che si vuole comunicare: i "charts". Per ogni informazione esistono diversi grafici che più si adattano. Per esempio, se si vuole visualizzare l'andamento nel tempo di una certa grandezza bisognerà usare un "Line Chart"; se invece si vuole porre l'attenzione sulla composizione di un certo insieme si sceglierà un "Pie Chart" o una "Table"; esiste anche la possibilità di visualizzare la posizione sulla mappa di certe concentrazioni di elementi, utilizzando una "World Map" o una "Country Map" [Figura 3].

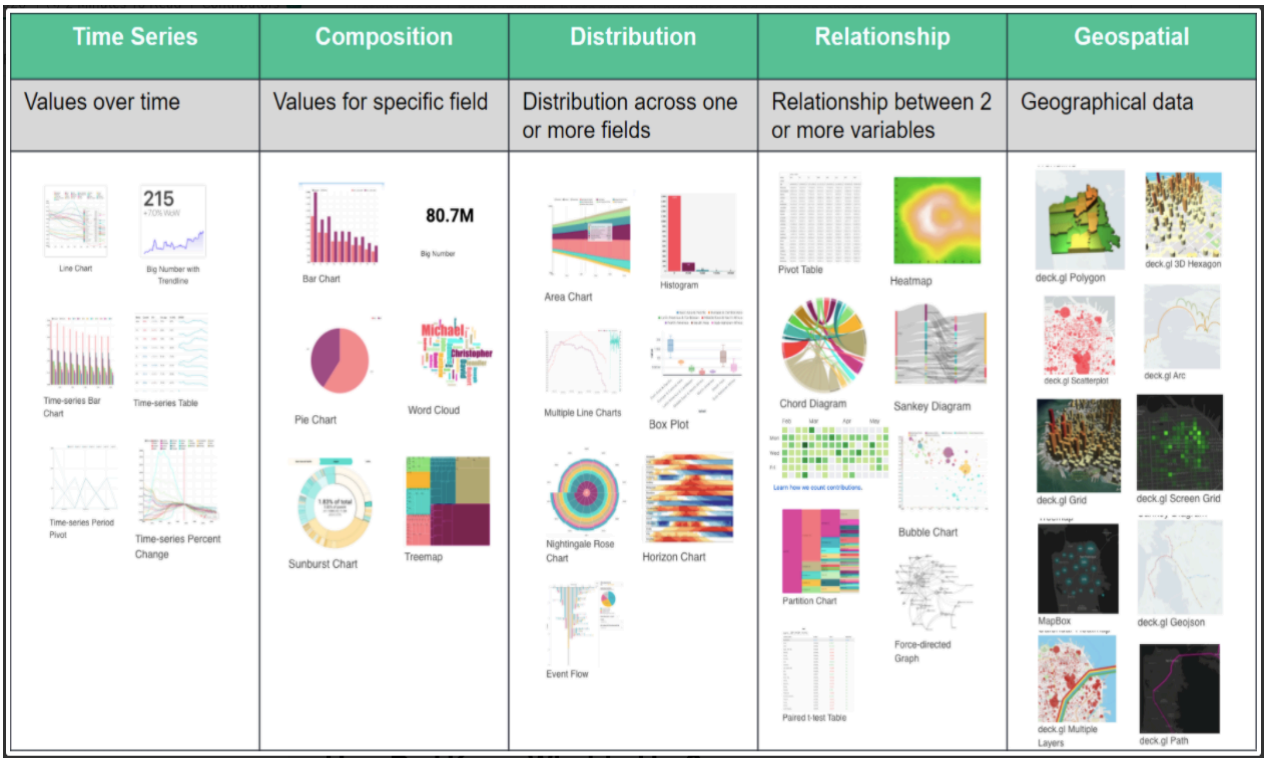


Figura 3 tipi di visualizzazione consigliati per ogni necessità (fonte: https://docs.preset.io/v1/docs/en/the-right-chart-for-your-data).

Una volta scelto ed impostato il *chart*, lo si può inserire in una *dashboard*; per crearla si entra dal menù principale attraverso "Dashboards" e, cliccando sul bottone "New", si può iniziare a personalizzare la propria *dashboard*, inserendo nuove righe, colonne, nuovi *tab*, titoli, didascalie. Tutto ciò deve essere volto a rendere la schermata la più semplice ed immediata possibile per poter comunicare con facilità i tanti dati che si stanno analizzando.

1.3. Connessione ad un Database

Per poter lavorare con Superset è basilare avere a disposizione i dati da gestire; perciò bisogna connettersi al server di un database. Andando nella sezione "Sources" e cliccando sul bottone "New" si apre la finestra in cui configurare il collegamento al server di interesse [Figura 4].

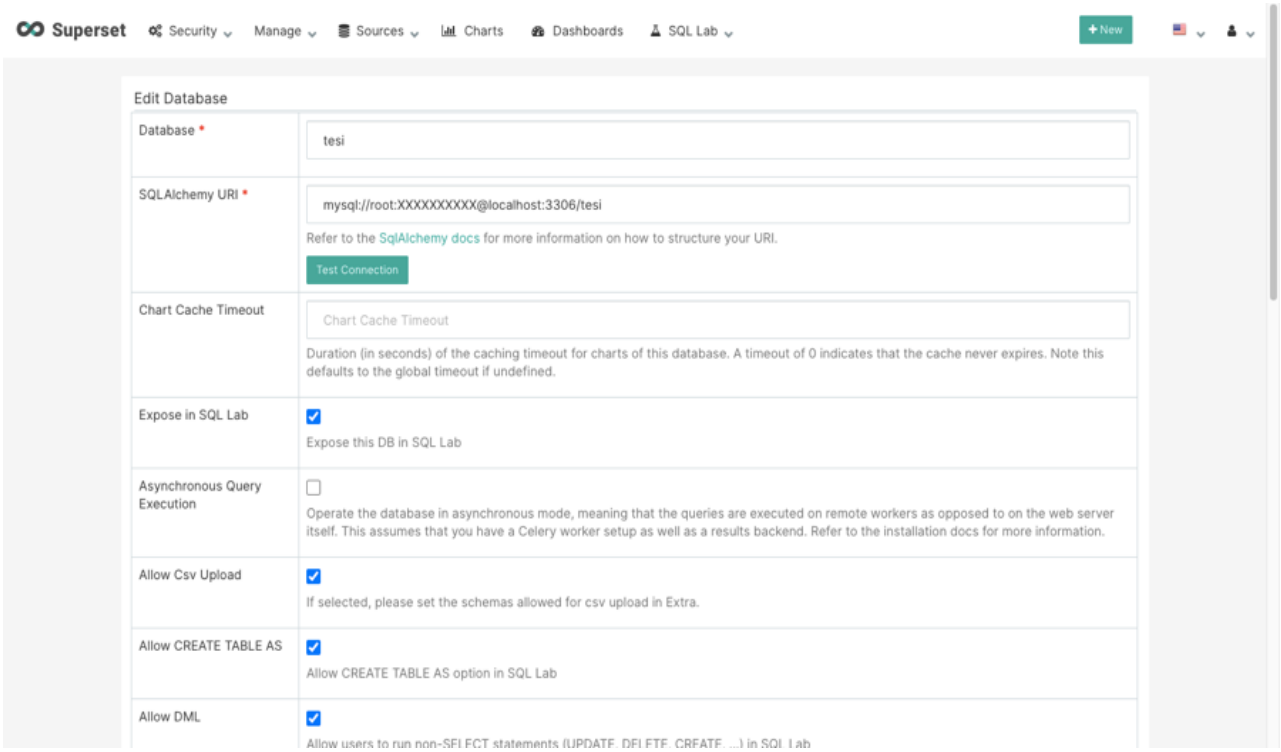


Figura 4: schermata per la connessione a database.

In *Database* va inserito il nome del database a cui ci si vuole connettere; nella *SQLAlchemy URI* va digitata la stringa che imposta il percorso per connettersi al server.

1.3.1 SQLAlchemy

SQLAlchemy [17][18] è una libreria Python che fornisce un'interfaccia per database MySQL, Oracle e PostgreSQL; include un linguaggio SQL per database server-indipendenti e un *object-relational mapper* (ORM) che permette di mappare gli elementi immagazzinati in formato relazionale nei loro corrispettivi oggetti Python, in modo da poter lavorare direttamente su astrazioni dei dati più semplici da utilizzare per gli sviluppatori. L'obiettivo principale di SQLAlchemy è, quindi, quello di esporre i processi, automatizzando le attività ridondanti, lasciando però al programmatore la libertà e il controllo sull'organizzazione del database.

SQLAlchemy prevede anche una *Engine class* e una *MetaData class*:

- ***Engine class*** gestisce l'insieme di connessioni ed i *dialetti* SQL;
- ***MetaData class*** riguarda le informazioni sulle tabelle.

Anche se SQL è un linguaggio standardizzato, tanti produttori di database non lo implementano completamente, o semplicemente ne creano delle estensioni. Quindi, ogni dialetto (piccole differenze

nel linguaggio SQL) viene gestito da SQLAlchemy così come i moduli DB-API che implementano le connessioni. La DBAPI [19] è una specifica per un'interfaccia comune ai database relazionali.

L'*engine* è, quindi, il punto di partenza per una specifica combinazione DBAPI/database e fornisce il DBAPI alle applicazioni SQLAlchemy attraverso un *pool* di connessioni e un dialetto. [Figura 5]

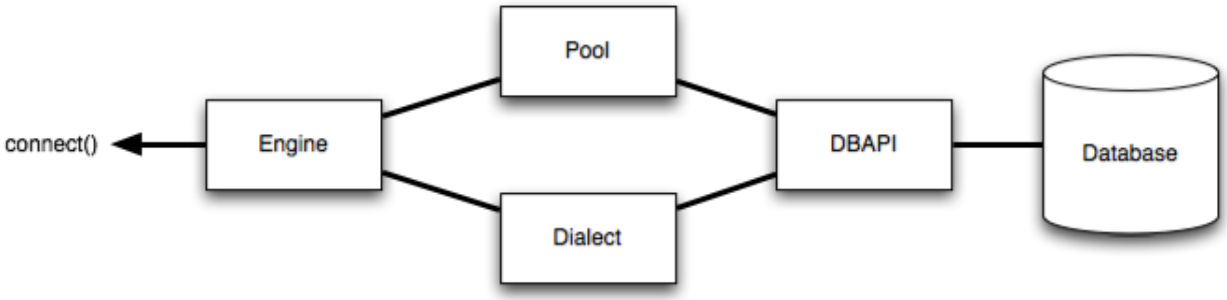


Figura 5: struttura generale della connesione engine-database (fonte: https://docs.sqlalchemy.org/en/13/core/engines.html)

L'*Engine* viene creato sulla base di un URL che include il nome utente, la password, il nome dell'host e il nome del database [20]:

```
dialect+driver://username:password@host:port/database
```

per MySQL, la stringa è:

```
mysql://root:XXXXXXXXXX@localhost:3306/tesi
```

Per costruire la stringa SQLAchemy, quindi, si deve esplicitare il *driver*, o il *dialetto* – per poter usare il driver di MySQL è stato necessario installare *mysqlclient*; bisogna poi inserire il nome utente che gestisce il server (nel caso di studio è presente un server MySQL, il cui proprietario è l'utente "root") e la password per accedervi. Inoltre, è necessario scrivere *localhost*, o 127.0.0.1, che è l'indirizzo IP del computer locale, e la porta del server di MySQL: 3306 in questo caso. Infine, si inserisce il nome del database in cui si vuole lavorare. Tutto ciò è necessario per garantire la connessione al server. Alla fine di questo processo si testa la connessione e, se risulta positiva, si può accedere al server e iniziare a porre delle *query* al database [21][22].

1.4. Creare le dashboard

Per creare i grafici da inserire nelle dashboard si devono interrogare le tabelle del database. Per fare ciò è necessario entrare in *SQL Lab*. Da qui, come su scritto, si scelgono il database, lo schema e le tabelle. Per selezionare le colonne e le righe di interesse si comunica in linguaggio SQL. Dopodiché è possibile "esplorare" le colonne, andando così a creare il grafico. I *charts* disponibili sono numerosi e vengono mostrati nella schermata di SQL Lab. Scegliendo una tra quelle mostrate, è possibile creare il grafico [Figura 6].

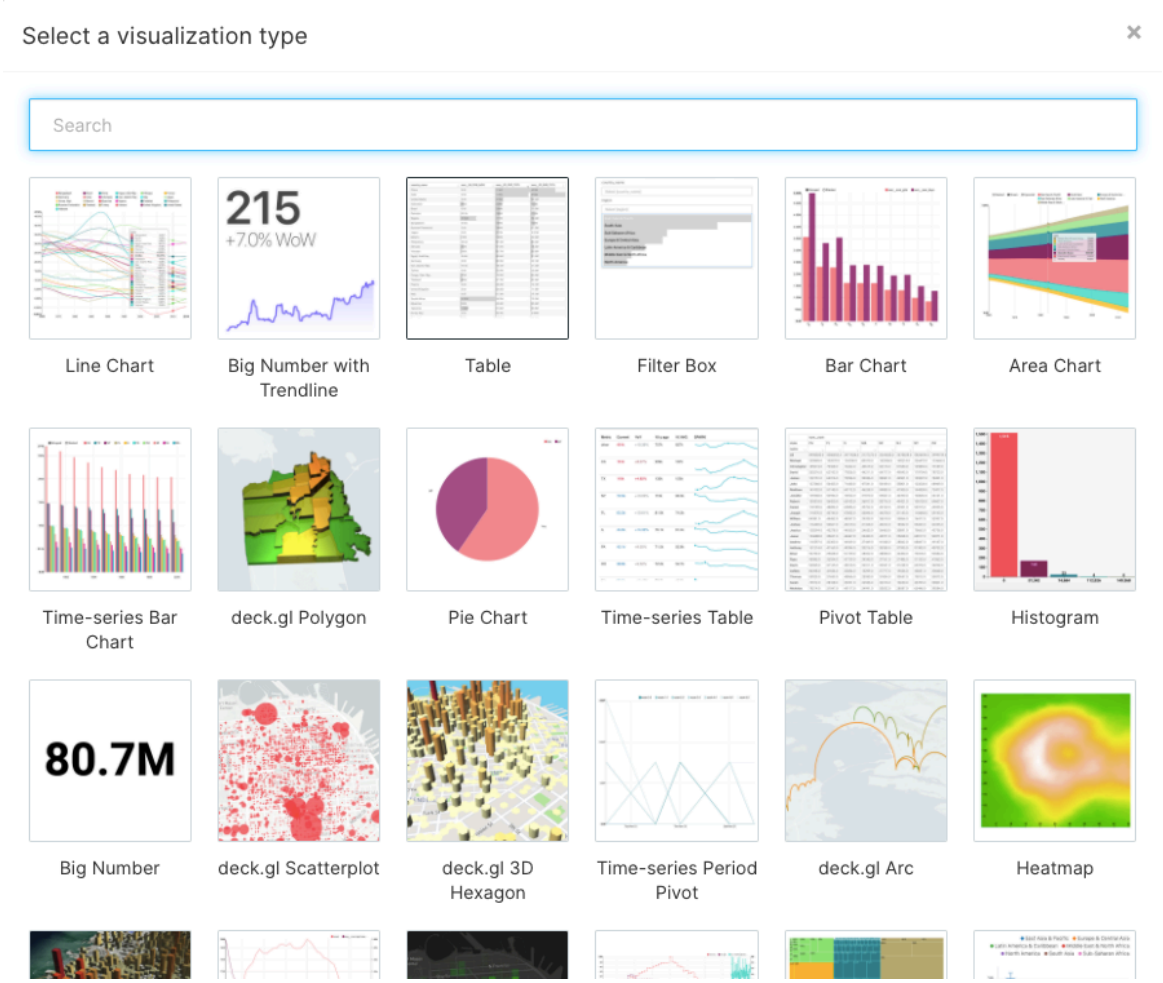


Figura 6 schermata dei tipi di visualizzazione dei dati da Superset.

Una volta scelto un tipo di visualizzazione si devono identificare le colonne di interesse per formare il grafico: ad esempio, se l'obiettivo è mostrare l'andamento temporale di una certa grandezza, allora sarà necessario selezionare una colonna con i dati temporali; se, invece, si deve mostrare la quantità di un elemento in un insieme bisognerà avere una colonna che contiene il conteggio (COUNT, in linguaggio SQL) dei dati.

Ora che il grafico è stato creato, va inserito nella dashboard. Nel menù dedicato è possibile modificare l'organizzazione e la posizione delle immagini, cambiare il titolo, aggiungere sottotitoli e didascalie, per rendere la pagina chiara ed immediata.

È possibile, infine, pubblicare la propria dashboard; una volta resa pubblica sarà visibile da tutti gli utenti che accedono al server di Superset, se hanno i permessi necessari.

1.5. MySQL server

Fonte: [15]

Per creare le tabelle ed i database si utilizza il linguaggio di interrogazione SQL, il cui concetto fondamentale è la tabella. Ogni tabella è formata da più colonne, e i dati sono rappresentati come le righe di queste tabelle. Gli elementi possono essere di diversi tipi: INTEGER ("int", sono i numeri interi), CHARACTER ("varchar", sono le stringhe alfanumeriche) o TIMESTAMP (identificano l'unità di misura del tempo). Se alcuni dati sono di tipo sconosciuto, vendono rappresentati con NULL [18].

Per poter far andare un programma SQL è necessario un *Database Management System* (DBMS). Si può accedere e lavorare con un DBMS online e in locale, e il computer in cui gira il DMBS viene detto *host*. In commercio sono presenti tanti DBMS, i più utilizzati sono: Oracle, PostgreSQL, Microsoft Access, Microsoft SQL Server e MySQL. Per lo sviluppo del progetto si è optato alla fine per MySQL, applicazione open-source che è stata introdotta durante il corso di Laboratorio di Ingegneria Biomedica.

Per creare un nuovo database, o una nuova tabella vanno digitati i comandi:

"CREATE DATABASE IF NOT EXISTS …;" seguito dal nome del database da creare;

"CREATE TABLE IF NOT EXISTS …;" seguito dal nome della tabella da creare.

Dopodiché si inseriscono i nomi delle colonne e le variabili di ciascun elemento, e si identificano le chiavi primarie delle tabelle. Infine, occorre immettere i dati nelle tabelle, completando tutti i campi delle colonne, o scrivendo "NULL" laddove un certo campo non possa essere riempito.

Per fare ciò è necessario il comando:

"INSERT INTO nome_tabella (nome_colonna, ...) VALUES (valore1, valore2, …);"

Una volta create le tabelle, è possibile interrogare il database attraverso le *query* [23][26][27][15][Figura 7].

Figura 7: righe di codice SQL per la creazione di un database e di una tabella.

```
DROP DATABASE IF EXISTS tesi;

CREATE DATABASE IF NOT EXISTS tesi;

USE tesi;

create table if not exists utenti (
    id_u int(5) not null primary key,
    nome varchar(20),
    cognome varchar(20)
    );
```

2. CAPITOLO 2 – LO SCENARIO IN ESAME

Il caso di studio proviene dal progetto MONAS (Monitoring and Analysis System), che ha come obiettivo quello di monitorare le temperature rilevate dai trasformatori in resina. Il target è quello di ricevere ed elaborare i dati in tempo reale in modo da:

- pianificare gli interventi di manutenzione preventiva;
- inviare le segnalazioni agli utenti proprietari delle centraline;
- memorizzare e leggere i dati a lungo, medio e breve termine;
- confrontare i trasformatori delle cabine;
- creare un report per fornire agli utenti – cioè i clienti – un'analisi dei costi.

Ogni trasformatore contiene 4 sonde per la rilevazione della temperatura; tre di queste, A, B e C, attivano la ventilazione alla stessa temperatura (100°C), mentre vanno poi in "pre-allarme" e "sgancio" a temperature superiori. La sonda D ha una soglia diversa, poiché si trova nel nucleo del trasformatore.

Superset è necessario per poter comparare le temperature delle diverse cabine, anche in relazione agli eventi di manutenzione. [Figura 8].

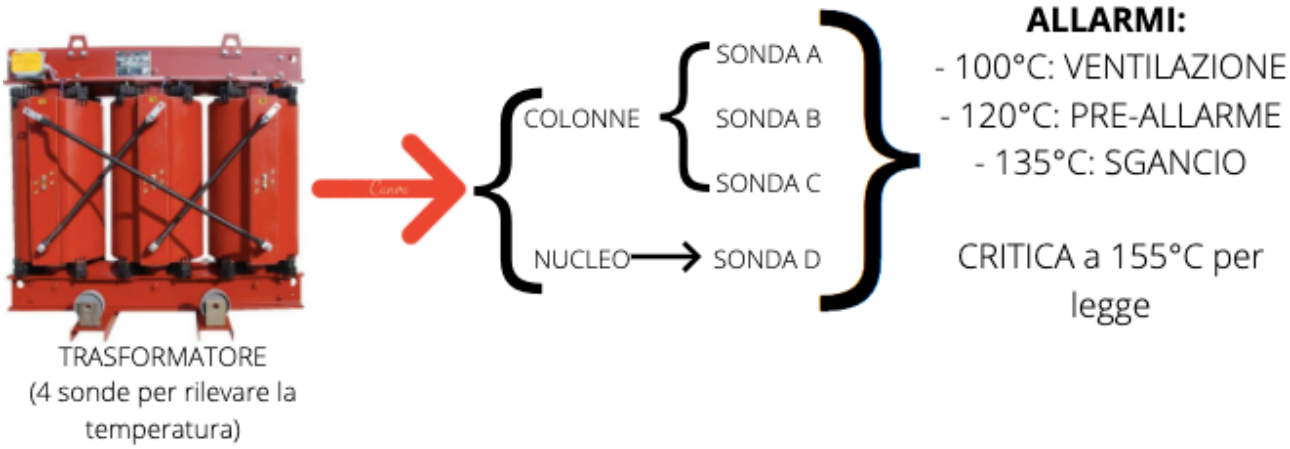


Figura 8 schema riassuntivo delle funzionalità di MONAS

Come scritto sopra, per poter usare Superset è necessario collegarlo al server di MySQL, per poi accedere al database di interesse e di conseguenza creare le *query* per selezionare ed analizzare i dati delle tabelle.

2.1. I dati

Gli elementi di interesse principale per creare le tabelle del database sono:

- **gli utenti**;
- **le centraline**: sono stati creati otto trasformatori, identificati da un ID univoco. Ogni centralina appartiene ad uno ed un solo utente, e i dati possono essere visti solo dall'utente proprietario o dall'admin. Ciascuna centralina contiene quattro sonde: A, B, C (le colonne), D (il nucleo).
- **le temperature**, con la relativa data di rilevazione.

Ciascuna di queste entità possiede degli attributi ed è collegata alle altre attraverso le relazioni. Così si crea lo schema Entità/Relazione (E/R) che descrive il database alla base dello studio.

2.1.1. Gli utenti

Gli utenti sono i proprietari delle cabine. Ogni utente può possedere più cabine ed ha l'accesso solo ed esclusivamente ai dati delle proprie. Perciò, una volta che uno *user* accede al proprio account Superset, può visualizzare solo le temperature rilevate dalle sue centraline. Per fare ciò si è reso necessario creare un nuovo ruolo per gli utenti non amministratori (gli *admin*). A partire dal ruolo *gamma*, sono stati revocati dei permessi e aggiunti altri, creando *Basic_user* [Figura 9]. Nonostante l'apparente non comprensibilità della lista, in generale un utente *Basic_user* può solo visualizzare (non può modificare) gli *slices* e le *dashboard* a cui ha accesso, e cioè solo quelle relative alle proprie centraline.

Basic_user	[can this form get on ResetMyPasswordView, can this form post on ResetMyPasswordView, can this form get on UserInfoEditView, can this form post on UserInfoEditView, can userinfo on UserDBModelView, can get on OpenApi, can show on SwaggerView, can get on MenuApi, can list on SliceModelView, can show on SliceModelView, can list on CssTemplateModelView, can show on CssTemplateModelView, can list on QueryView, can show on QueryView, can list on CssTemplateAsyncModelView, can this form get on CsvToDatabaseView, can list on DashboardModelViewAsync, can get on Datasource, can shortner on R, can list on SavedQueryView, can show on SavedQueryView, can list on SavedQueryViewApi, can show on SavedQueryViewApi, can list on SliceAsync, can extra table metadata on Superset, can csv on Superset, can slice on Superset, can stop query on Superset, can fetch datasource metadata on Superset, can available domains on Superset, can explore json on Superset, can schemas on Superset, can filter on Superset, can slice json on Superset, can user slices on Superset, can profile on Superset, can datasources on Superset, can search queries on Superset, can select star on Superset, can recent activity on Superset, can dashboard on Superset, can copy dash on Superset, can request access on Superset, can save dash on Superset, can fave dashboards by username on Superset, can created slices on Superset, can fave slices on Superset, can warm up cache on Superset, can created dashboards on Superset, can estimate query cost on Superset, can results on Superset, can list on TableColumnInlineView, can show on TableColumnInlineView, can expanded on TableSchemaView, can migrate query on TabStateView, can activate on TabStateView, can post on TabStateView, menu access on Charts, menu access on Query Search, can download dashboards on DashboardModelView, can show on DashboardModelView, can download on SliceModelView, menu access on Import Dashboards, all datasource access on all_datasource_access, menu access on Dashboards, can list on DashboardModelView]

Figura 9 lista dei permessi concessi al ruolo Basic_user.

Sono stati creati quattro utenti, ad ognuno dei quali è stato assegnato un numero identificativo univoco (da 1 a 4) che identifica, quindi, anche l'insieme di centraline di proprietà di uno *user*. Per effettuare l'accesso, essi devono essere in possesso della e-mail e di una password nel momento della registrazione [Figura 10]. Nel gestire gli utenti, è possibile anche disattivarne uno, oppure rimuoverlo del tutto, nel caso in cui non abbia più i diritti per accedere all'applicazione. Ogni cliente, nella pagina con le proprie informazioni ha la possibilità di modificare la propria password.

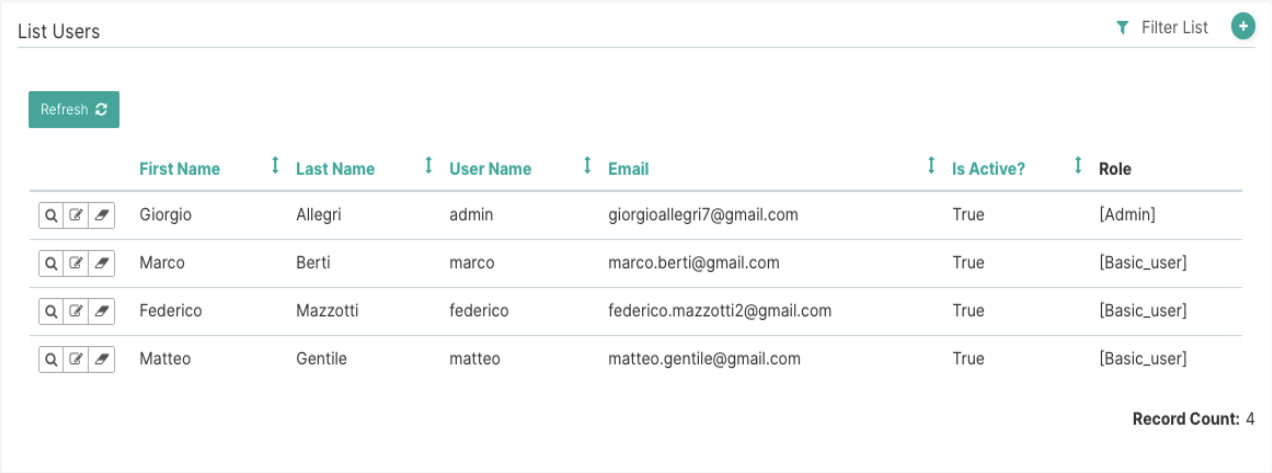


Figura 10 schermata della lista degli utenti creati; oltre ai dati si può notare la presenza del ruolo. Da Superset.

2.1.2. Le temperature

Sono stati creati otto trasformatori, identificati da un ID univoco. Ogni centralina appartiene ad uno ed un solo utente, e i dati possono essere visti solo dall'utente proprietario o dall'admin.

Ciascuna centralina contiene quattro sonde: A, B, C (le colonne), D (il nucleo).

Teoricamente, la rilevazione della temperatura deve avvenire ogni 5, 30 o 60 secondi, ma ai fini della simulazione è stato preso 24 ore come periodo di campionamento: la temperatura viene esaminata una volta al giorno, per un totale di 14 giorni. Le temperature sono state create casualmente, scegliendo un numero random compreso tra 10 e 150°C. Ogni temperatura è associata ad una sonda alla relativa centralina, e quindi ad un certo utente.

Gli allarmi si presentano quando la temperatura supera le seguenti soglie:

- 100°C per VENTILAZIONE;
- 120°C si è in PRE-ALLARME;
- 135°C si va in SGANCIO;
- 155°C è la temperatura CRITICA.

Dato che le temperature vengono rilevate nelle sonde, ogni rilevazione è identificata da: ID utente, ID centralina, e dal nome della sonda.

2.2. Schema E/R

Il modello E/R è un metodo descrittivo che utilizza una simbologia dettagliata. Le *entità* sono elementi di interesse che hanno caratteristiche comuni, ma esistono indipendentemente dagli altri oggetti dello stesso insieme; ognuna di esse è composta da più *attributi*, che sono le proprietà elementari della entità, e la descrivono. Inoltre, gli elementi che compongono le entità sono le istanze. Vengono simboleggiate da un rettangolo.

Le *relazioni* sono i legami tra più entità; vengono distinte con un rombo.

Ogni relazione può avere un *vincolo di cardinalità*: è una coppia di numeri (N, M) che identificano il numero minimo e massimo di entità che possono partecipare a quella relazione.

Una volta create le colonne delle tabelle, vanno inseriti i dati. Quindi, con MySQL, sono stati aggiunti gli elementi per simulare il più possibile la realtà. Gli utenti realizzati sono quattro, di cui solo uno ricopre il ruolo di *admin*, mentre gli altri tre possiedono i permessi di *Basic_user*. Le centraline simulate sono quaranta, localizzate in cinque paesi del mondo: in Italia, nello specifico, sono state posizionate in cinque province diverse. Le temperature sono state rilevate per otto cabine, ed ogni utente ne ha a disposizione due, di conseguenza può vedere solo i dati delle proprie centraline, se non è *admin*. Tra il 01/01/2020 e il 12/01/2020 sono state effettuate una misurazione al giorno per ogni sonda (A, B, C e D) di ciascuna centralina, per un totale di 384 rilevazioni.

In **Errore. L'origine riferimento non è stata trovata.** è mostrato lo schema Entità/Relazione del d atabase per la situazione in esame:

- ogni utente, per poter essere registrato nel database, deve essere il proprietario di *almeno* una centralina (1:n); ogni centralina, però, è gestita da uno e un solo utente (1:1).
- ogni centralina contiene sempre quattro sonde.

"*PK*" indica la chiave primaria della tabella (Primary Key); anche detto *identificatore*, gode della proprietà della *minimalità,* e permette l'identificazione univoca delle istanze di un'entità.

Il diagramma E/R è stato creato con il programma "draw.io" [23][24][**Errore. L'origine riferimento non è stata trovata.**].

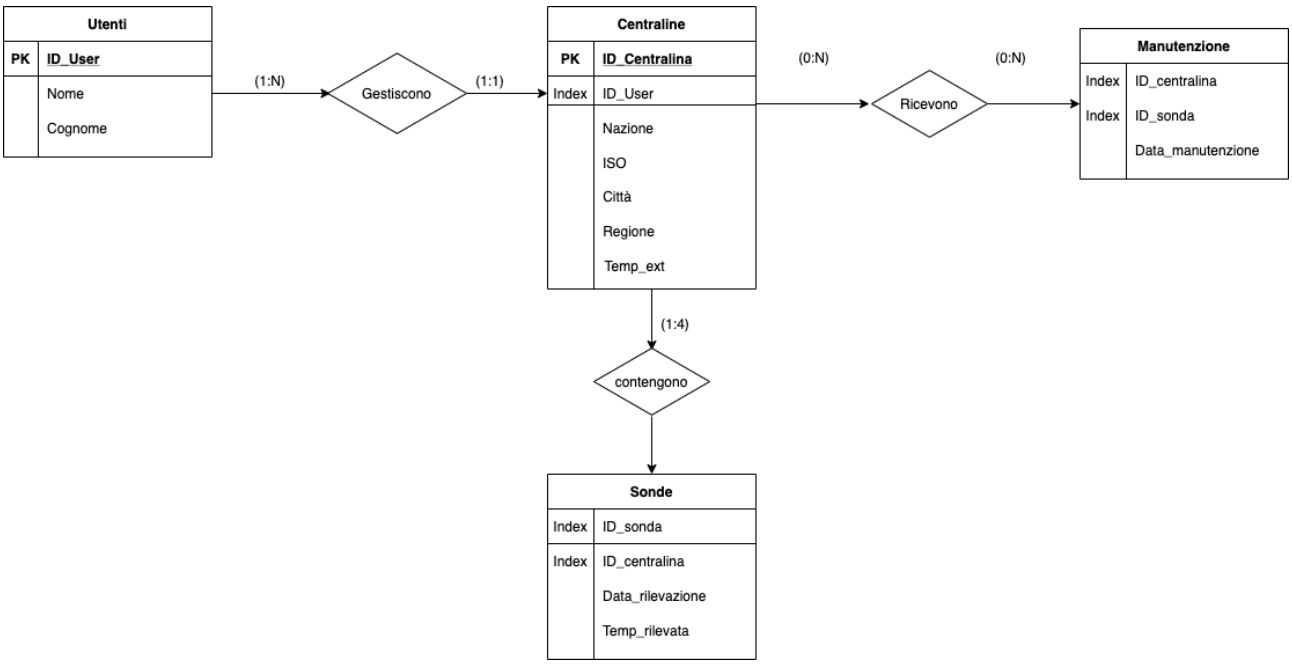


Figura 11: schema E/R dello scenario in esame.

Con tale schema è stato possibile gestire 8 centraline diverse, con rilevazioni che iniziano il 01/01 e terminano il 12/01. I valori di temperatura inseriti sono stati scelti in modo casuale con il codice:

“update tesi.sonda set temp_riv = FLOOR(RAND()*151)+10 WHERE id_sonda='…' ;”

2.3. Le tabelle

Dopo la creazione di una prima tabella di prova, “riv_temp”, che conteneva solo le colonne con un ID della rilevazione, la data e la temperatura rilevata, sono state create le tabelle che più si avvicinano ad un’organizzazione efficace dei dati: “centraline”, “sonda” e “utenti”. Infine, per poter inserire le date delle manutenzioni, così da visualizzarle sui grafici, è stata aggiunta la tabella “manutenzione”.

2.3.1. Centraline

La *table* “centraline” è formata dalle informazioni generali sui trasformatori, quindi il codice identificativo, l’ID del proprietario, il luogo in cui sono ubicati e anche le rispettive temperature esterne. La *chiave primaria* è l’ID della centralina (“ID_c”), mentre il codice utente (“ID_user”) è un *indice*. [Tabella 1].

Tabella 1: colonne della table "centraline", con variabili.

ID_C (int, PK)	ID_user (int, Index)	Nazione (varchar)	ISO (varchar)	Città (varchar)	Regione (varchar)	Temp_ext (int)

Centraline (ID_C%, ID_user%, Nazione, ISO, Città, Regione, Temp_ext)

FK: ID_user REFERENCES utenti.

La colonna “ISO” è servita per la visualizzazione della posizione delle colonnine sulla mappa; ci si è serviti dei codici *ISO 3166-1 alpha-3,* che identifica un sistema a tre lettere per descrivere una nazione; in modo particolare per l’Italia sono stati usati anche i codici delle province. [25].

2.3.2. Sonda

La tabella in questione contiene i dati sulle temperature rilevate dalle varie sonde delle centraline. Sono presenti l’ID della sonda, che può essere A, B, C, o D, il codice della centralina a cui appartengono le sonde, la temperatura rilevata e la data del monitoraggio. Sono indici l’ID della sonda e quello della centralina. [Tabella 2].

Tabella 2: colonne della table "sonda", con variabili.

ID_sonda (int, Index)	ID_centralina (int, Index)	Data_rilevazione (int)	Temp_rilevata (int)

Sonda (ID_Sonda, ID_centralina%, Data_rilevazione, Temp_rilevata)

FK: ID_centralina REFERENCES centraline.

2.3.3. Utenti

“Utenti” comprende gli elementi che distinguono coloro che gestiscono una (o più) centraline: i dati anagrafici, e un codice univoco per identificarli. [Tabella 3].

Tabella 3: colonne della table "utenti", con variabili.

ID_User (int, PK)	**Nome (varchar)**	**Cognome (varchar)**

Utenti (ID_User%, Nome, Cognome).

È stata creata, inoltre, una tabella “**Manutenzione**”, per poter inserire nei grafici le date delle manutenzioni e correlarle con gli effettivi andamenti delle temperature [Tabella 4].

Tabella 4: table “manutenzione”, con variabili.

ID_Centralina (int, Index)	**ID_Sonda (int, varchar)**	**Data_manutenzione (date)**

Manutenzione (ID_Centralina%, ID_Sonda%, Data_manutenzione);

FK: ID_Centralina REFERENCES centraline;

FK: ID_Sonda REFERENCES sonda.

Dove “FK” sono le *chiavi esterne* (“foreign key”): identifica una o più colonne di una tabella, che si riferisce a una o più colonne di un’altra tabella.

3. CAPITOLO 3 – LA CREAZIONE DELLE DASHBOARD

Dopo tutto il processo di introduzione, di creazione del database e delle tabelle, di inserimento dei dati, si è pronti ora per interrogare gli elementi e costruire i grafici che più si adeguano al tipo di visualizzazione che si sta cercando.

Effettuando l'accesso come utente *Admin*, si ha l'accesso a tutte le dashboard create (in blu), e a tutte le opzioni di lavoro nella barra con i vari tab (in rosso) [Figura 12].

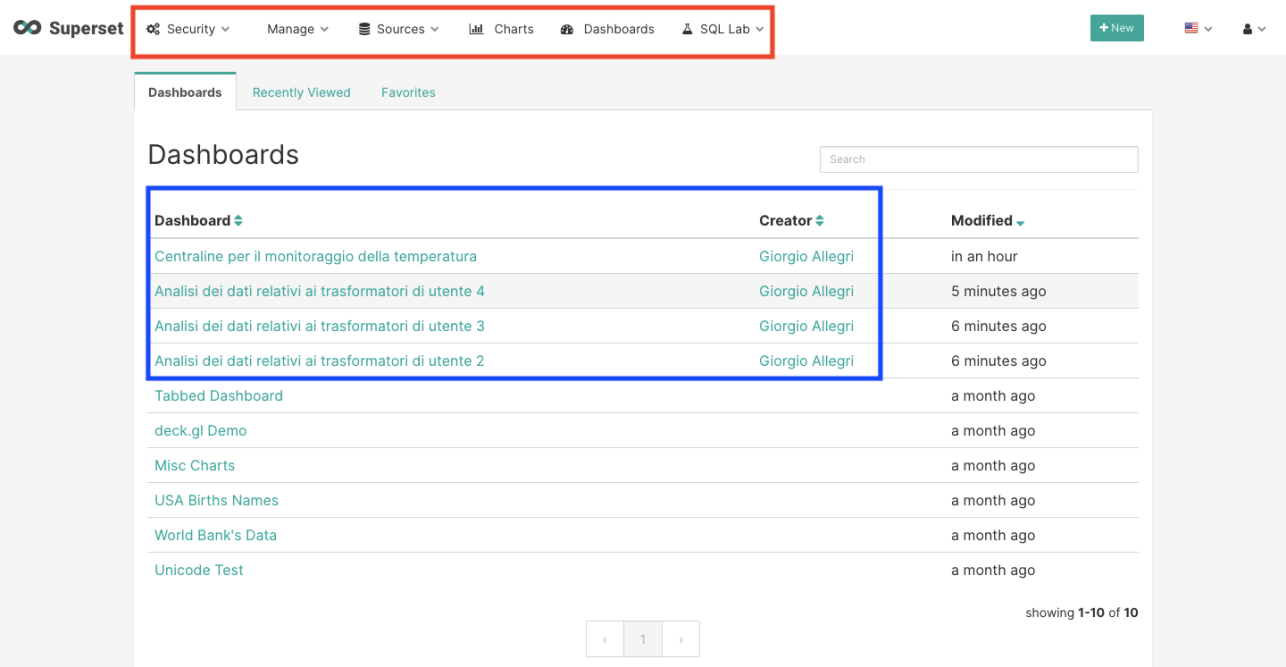


Figura 12: schermata di accesso di un utente Admin. Sono evidenziati in rosso i tab con le opzioni di lavoro disponibili; in blu le dashboard di proprietà.

3.1. I grafici

È necessario utilizzare il linguaggio SQL per interrogare le tabelle; l'area di Superset adibita a questo tipo comunicazione col database è SQL Lab. Selezionando lo schema e le tabelle di interesse, si vanno poi a scrivere nella *command line* gli ordini in SQL.

3.1.1. Temperature (tutti gli utenti)

Il primo grafico è stato creato per fornire una prima visione d'insieme: vengono visualizzate, infatti, tutte le temperature rilevate da ogni sonda di ciascuna centralina nell'intero arco temporale. Perciò, con la *query*, si selezionano i dati degli utenti (nome, cognome e ID), il codice identificativo delle sonde e delle centraline e la data e la temperatura rilevata in tale giorno. Sono stati aggiunte le date delle manutenzioni e i limiti di temperatura consentiti, con i vari allarmi [Figura 13].

È stato scelto un *chart* che fosse efficace per la visualizzazione della variazione di una grandezza (la temperatura) nel tempo: il *line chart*. Dal grafico in figura è possibile osservare l'andamento delle temperature (generate casualmente) confrontate tra quelle rilevate da tutte le centraline. Inoltre, sono

stati aggiunti i layer per le temperature di allarme (VENTILAZIONE, SGANCIO e PRE-ALLARME), e per le date della manutenzione.

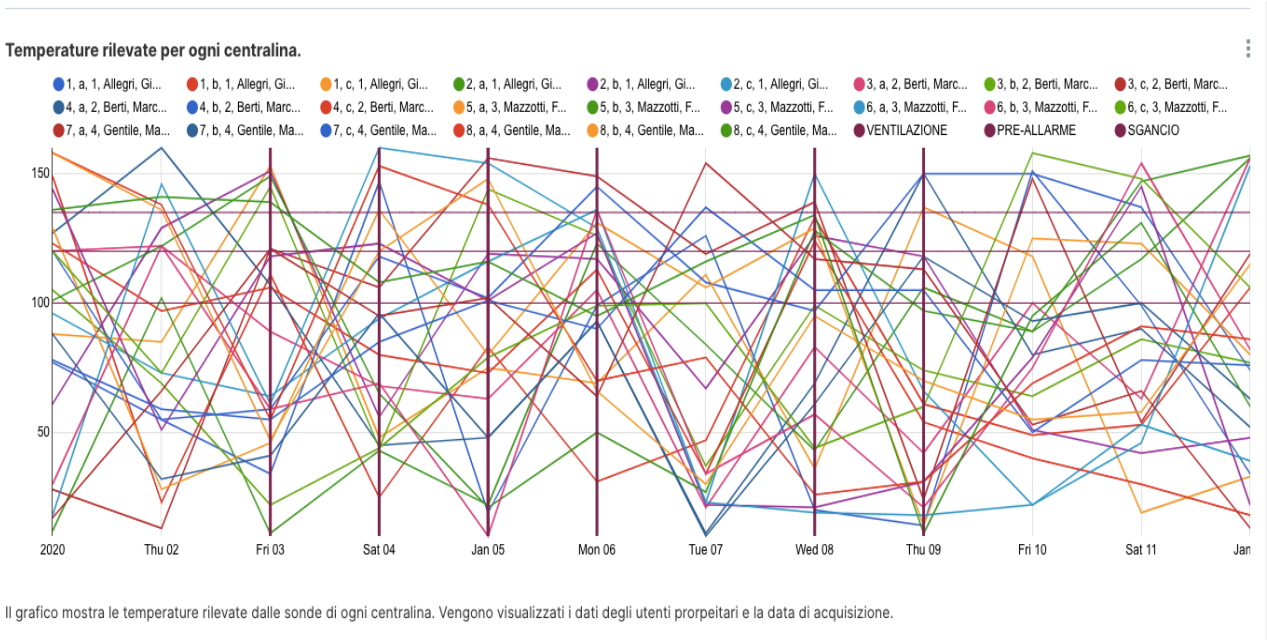


Figura 13: andamento temporale di tutte le temperature rilevate

Dalla schermata è possibile la visualizzazione dell'andamento di temperature a scelta, per esaminare solo i dati che interessano. Per esempio, è possibile osservare i dati di un singolo utente, o di una centralina, o una sola sonda [Figura 14].

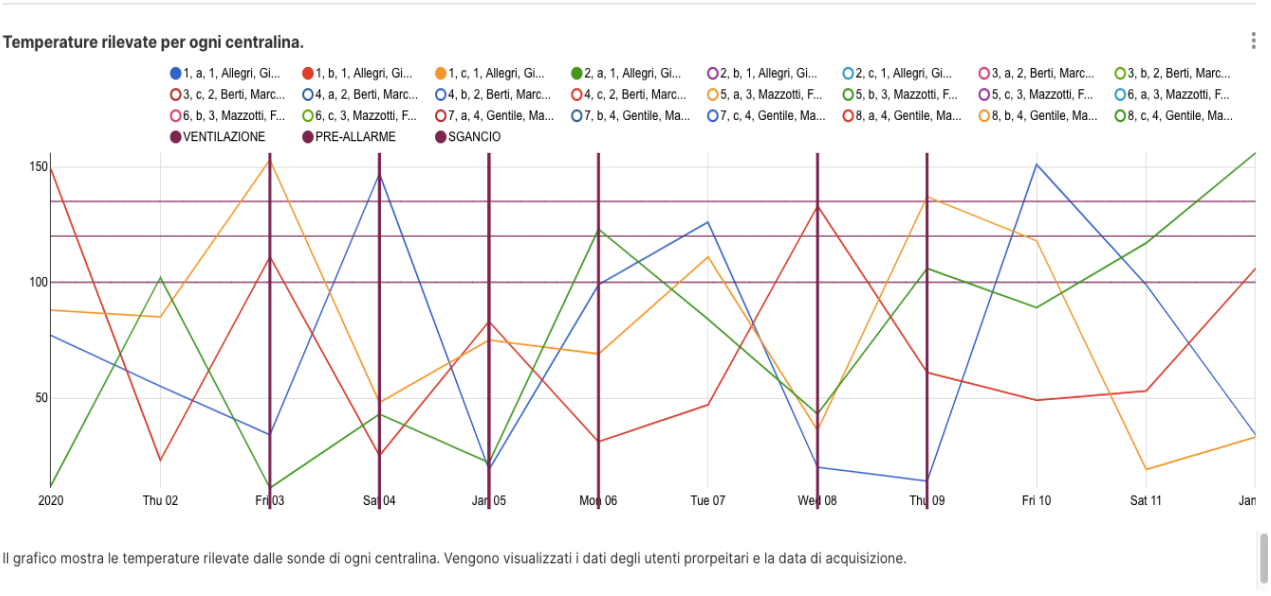


Figura 14 visualizzazione dei dati di una sola centralina

Un altro tipo di visualizzazione utilizzato per confrontare le temperature rilevate consiste nell'inserire gli stessi dati di cui sopra in tabella, in ordine decrescente di temperatura. In questo modo è possibile cercare un determinato evento, o andando a indagare le date, o trovando una determinata temperatura, legandola poi alla centralina ed al suo proprietario [Figura 15].

Figura 15 tabella della schermata di Superset con le temperature in ordine decrescente.

3.1.2. Posizione (tutti gli utenti)

Sempre dalla dashboard dell'*admin* si può cambiare la schermata, andando a osservare la posizione delle cabine nel mondo ed in Italia, divise per province. La quantità di trasformatori nella *slice* mondiale viene visualizzata con delle bolle di grandezza crescente.

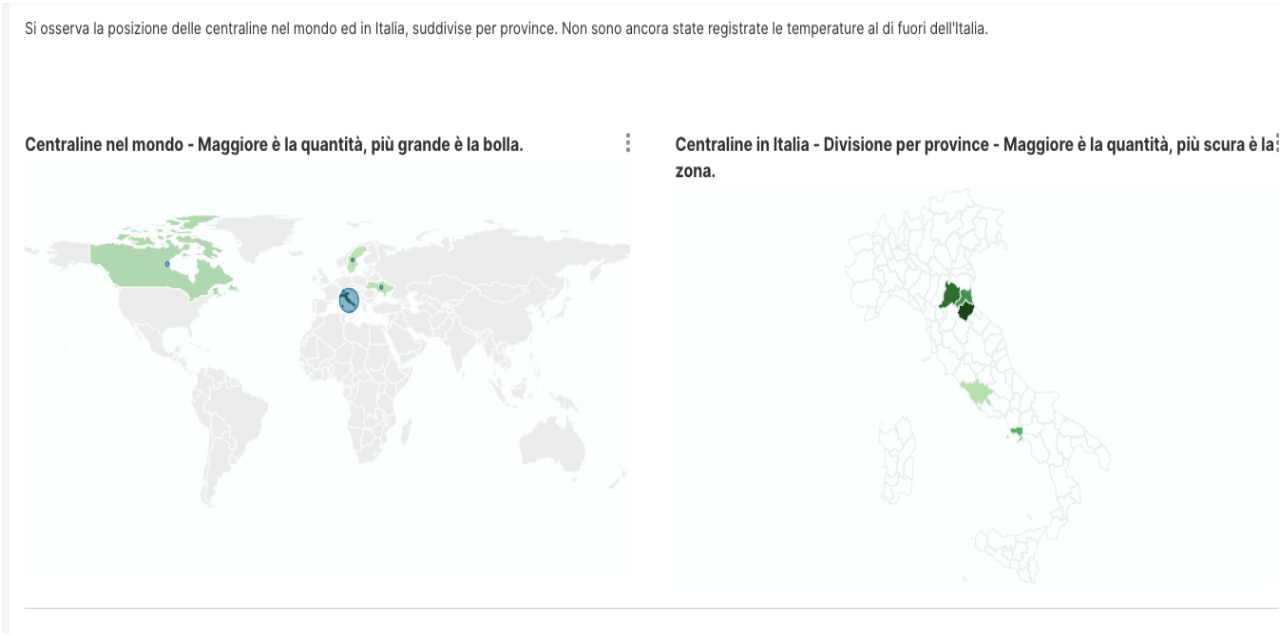


Figura 16 schermata che visualizza la distribuzione delle centraline nel mondo e in Italia.

Per rendere più evidente la differente quantità di centraline in Italia e nel mondo, sono stati creati due diagrammi a torta che esemplificano meglio la situazione [Figura 17], [Figura 18]. I grafici a torta rendono visibilmente più comprensibile la differenza nel numero delle centraline nei paesi del mondo

e, in modo più specifico, nelle province italiane. Sono stati utilizzati i codici "ISO 3166 [40].

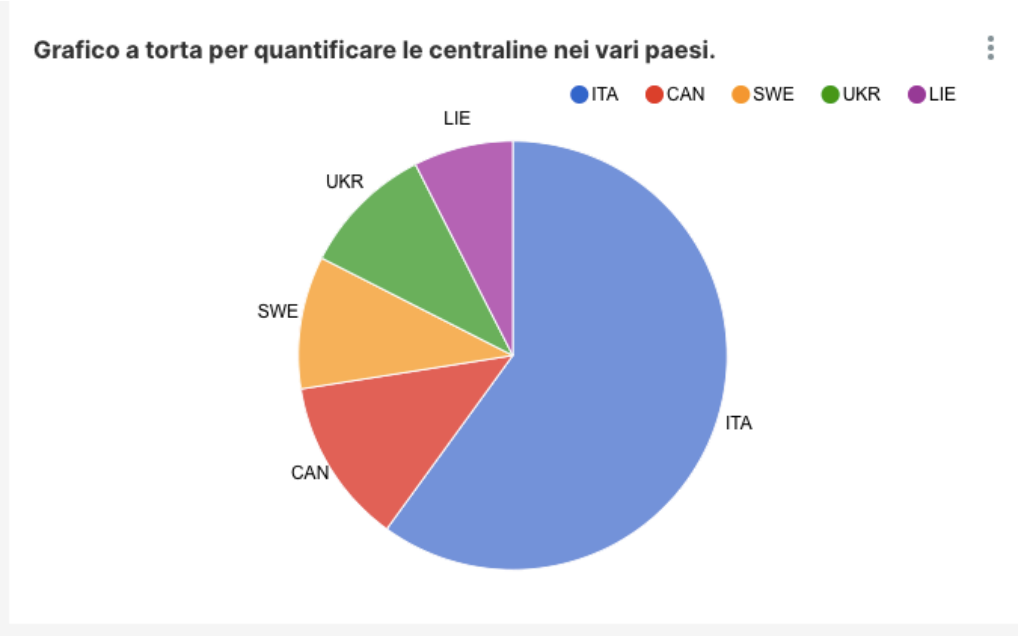


Figura 17: grafico a torta che rappresenta la quantità di centraline nel mondo

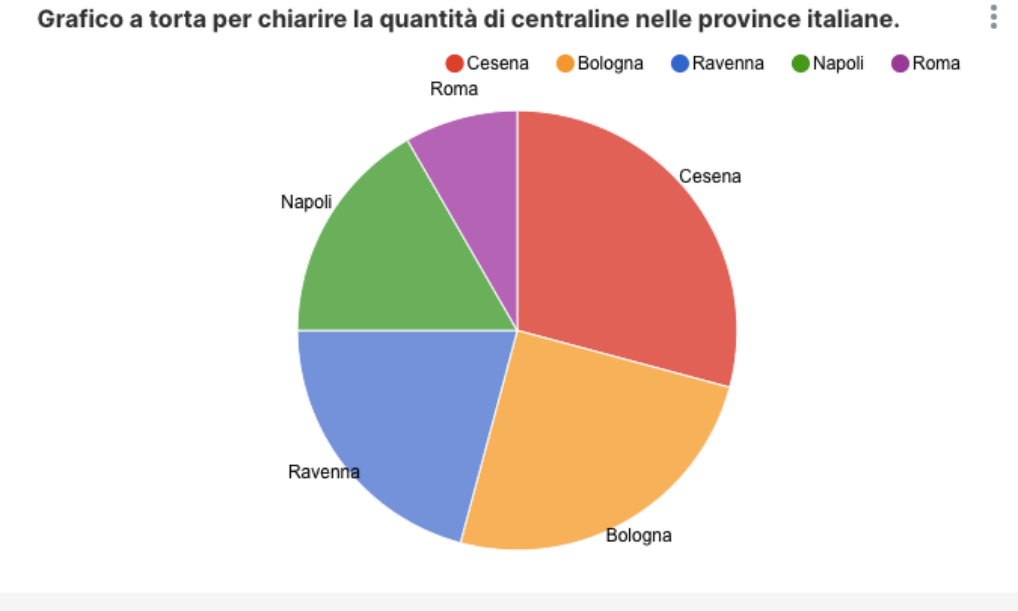


Figura 18: diagramma a torta che mostra la quantità di centraline nelle province italiane

3.1.3. Allarmi (tutti gli utenti)

È possibile, inoltre, controllare la quantità e la distribuzione degli allarmi, con le rispettive temperature – viene contato un allarme quando la temperatura supera la soglia di ventilazione posta

a 100°C – e le città in cui essi si sono verificati attraverso un *Pie Chart* [Figura 19] e una *Country Map* [Figura 20].

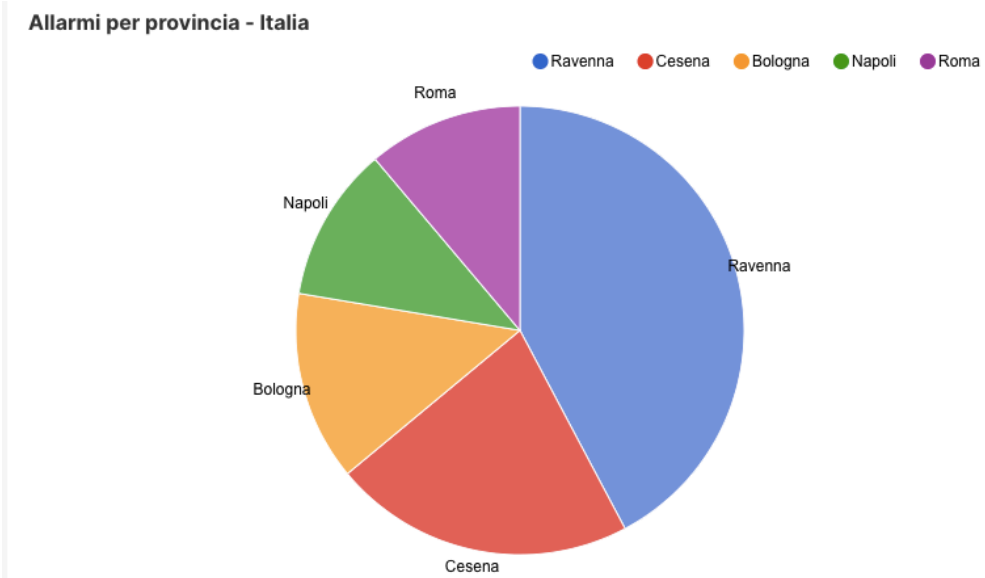


Figura 19: grafico a torta che rappresenta la quantità di allarmi in ogni provincia italiana

Non sono stati rilevati allarmi al di fuori dell'Italia perché le temperature simulate erano site solo in quella nazione. Le centraline aggiunte in altre zone sono state utilizzate solo per verificare il funzionamento del *chart* con la mappa della Terra.

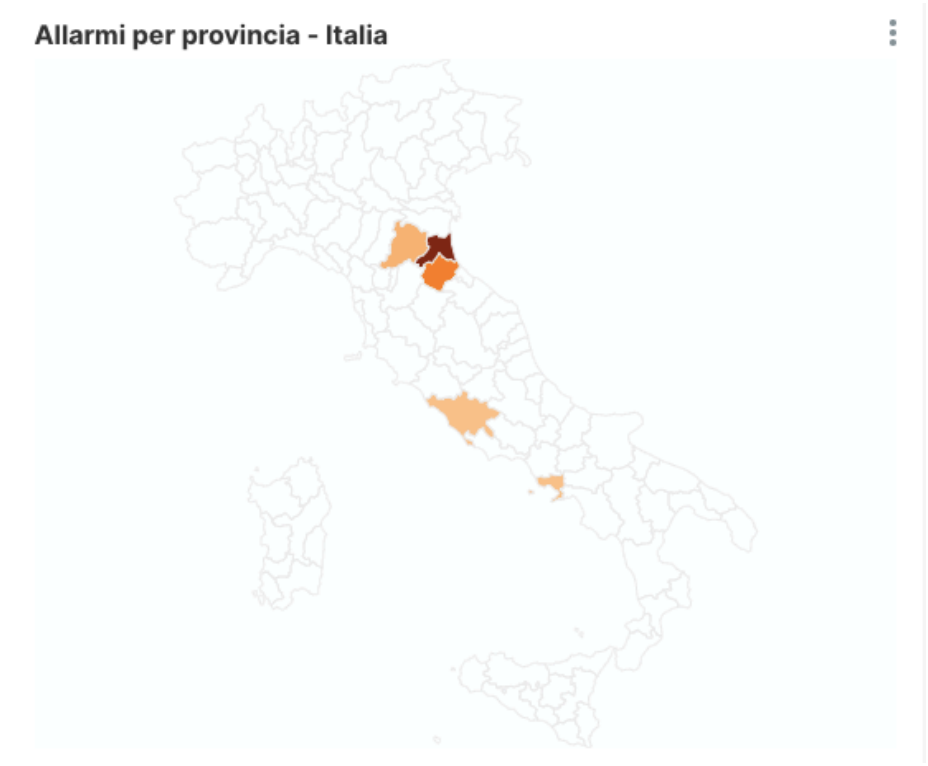


Figura 20: mappa con la localizzazione degli allarmi - più è alto il numero di allarmi, più è scura la provincia

Inoltre, è possibile osservare il numero di allarmi registrato da ogni centralina; vengono considerate tutte le temperature rilevate da ogni sonda (sono 4 in ogni trasformatore) [Figura 21].

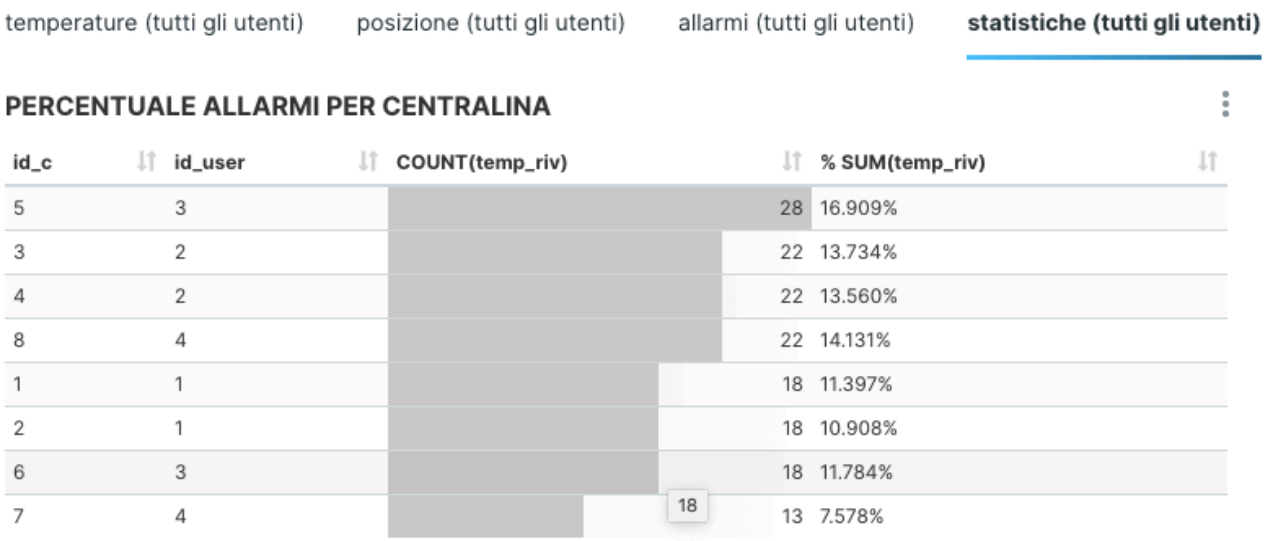
Numero di guasti per ogni centralina

id_user	id_c	citta	COUNT(temp_riv>100)
3	5	Ravenna	28
2	3	Bologna	22
2	4	Cesena	22
4	8	Ravenna	22
1	1	Ravenna	18
1	2	Roma	18
3	6	Napoli	18
4	7	Cesena	13

Figura 21: allarmi registrati in ogni centralina

3.1.4. Statistiche (tutti gli utenti)

Infine, l'utente amministratore può osservare la percentuale di allarmi per ogni centralina, per avere una visione chiara delle cabine più problematiche e avere così la possibilità di pianificare le manutenzioni in maniera più efficace. È visibile anche una *slice* che mostra le temperature massime in ogni trasformatore per ogni giorno [Figura 22][Figura 23].



temperature (tutti gli utenti) posizione (tutti gli utenti) allarmi (tutti gli utenti) **statistiche (tutti gli utenti)**

PERCENTUALE ALLARMI PER CENTRALINA

id_c	id_user	COUNT(temp_riv)	% SUM(temp_riv)
5	3	28	16.909%
3	2	22	13.734%
4	2	22	13.560%
8	4	22	14.131%
1	1	18	11.397%
2	1	18	10.908%
6	3	18	11.784%
7	4	13	7.578%

18

Figura 22: viene mostrata la percentuale di allarmi rilevati in ogni trasformatore.

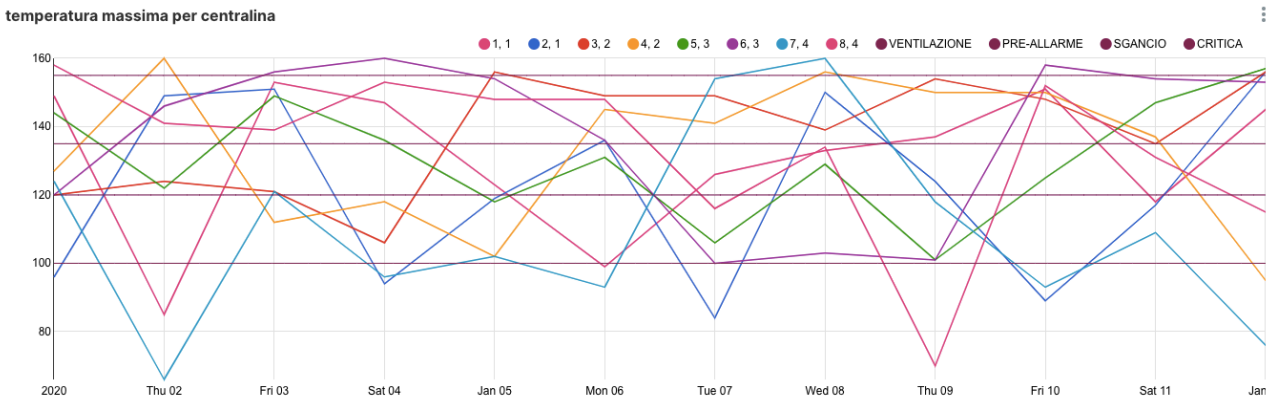


Figura 23: sono mostrate le temperature massime registrate in ogni centralina

3.2. Grafici per utenti non amministratori

Per raggiungere uno degli obiettivi prefissati, è stato necessario creare delle dashboard che hanno come proprietari l'utente *admin* e un altro utente; in queste dashboard sono presenti i grafici relativi alle centraline di proprietà dell'utente che può accedervi. Ogni utente infatti, se non è amministratore, ha il permesso di osservare le rilevazioni e l'analisi dei dati che provengono solo ed esclusivamente dai propri trasformatori. I permessi concessi a questi *users* (*Basic_user*) sono mostrati in Figura 9.

Gli utenti *Basic_user* sono Marco Berti (ID: 2), Federico Mazzotti (ID: 3) e Matteo Gentile (ID: 4).

Prendendo come esempio l'utente Marco Berti (ID 2), deve inserire nella propria pagina di sign-in i propri dati di accesso (nome utente e password) [Figura 24].

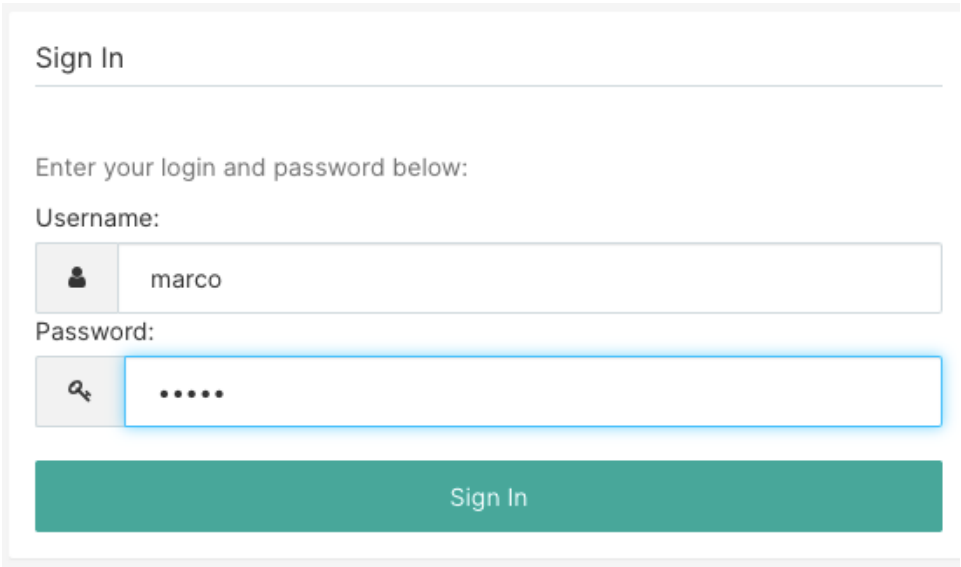


Figura 24: schermata di sign in.

Effettuato l'accesso, nella pagina principale si trovano solo le opzioni che un utente *Basic_user* può selezionare, oltre alle dashboard che può visualizzare, e quindi quelle pubbliche insieme a quelle di cui è proprietario [Figura 25][Figura 26].

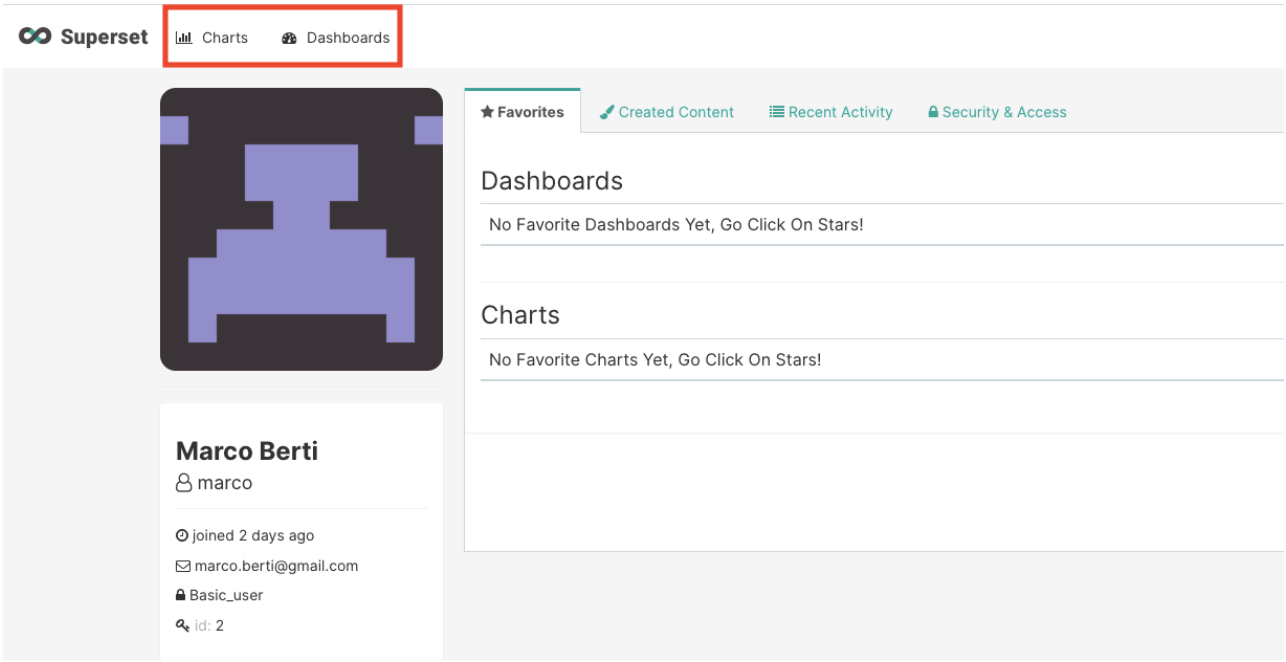


Figura 25: schermata della pagina con le informazioni di un utente Basic_user. Sono evidenziate in rosso le opzioni di lavoro.

In Figura 25 è possibile notare che la barra del menù di un utente *Basic_user* contiene molte meno opzioni rispetto a quella di un utente *Admin* [Figura 12].

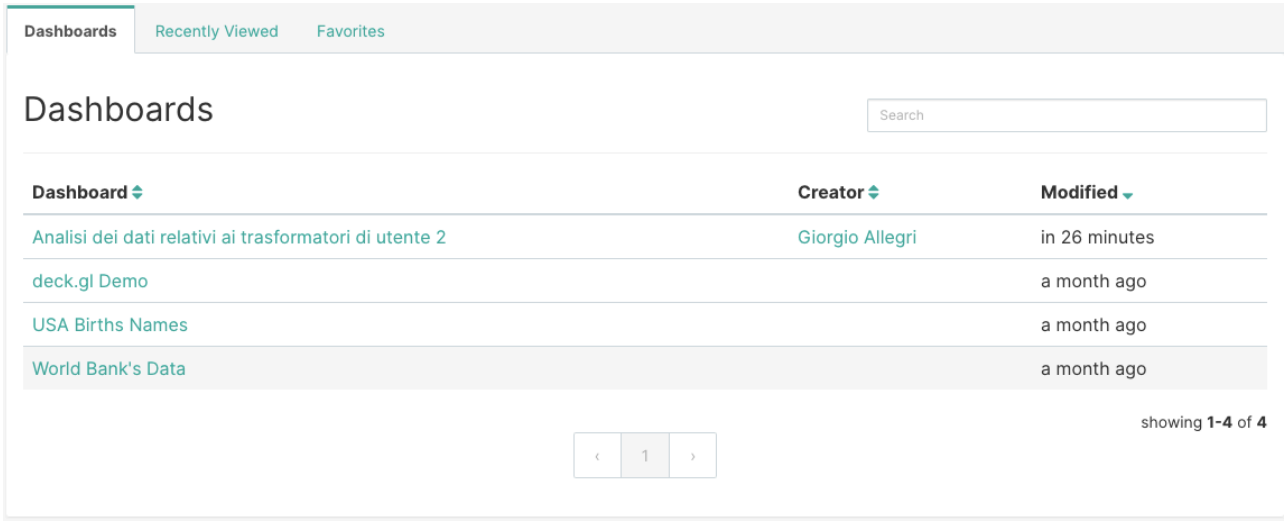


Figura 26: dashboard visualizzabili dagli utenti non amministratori.

Selezionando la dashboard “Analisi dei dati relativi ai trasformatori di utente 2”, si osservano i dati delle centraline di cui è proprietario l’utente 2. Nella prima pagina è presente l’analisi delle temperature rilevate [Figura 27]; poi, passando nella tab “POSIZIONE CENTRALINE” vengono esaminate i trasformatori per posizione [Figura 28], e infine per gli allarmi rilevati [Figura 29].

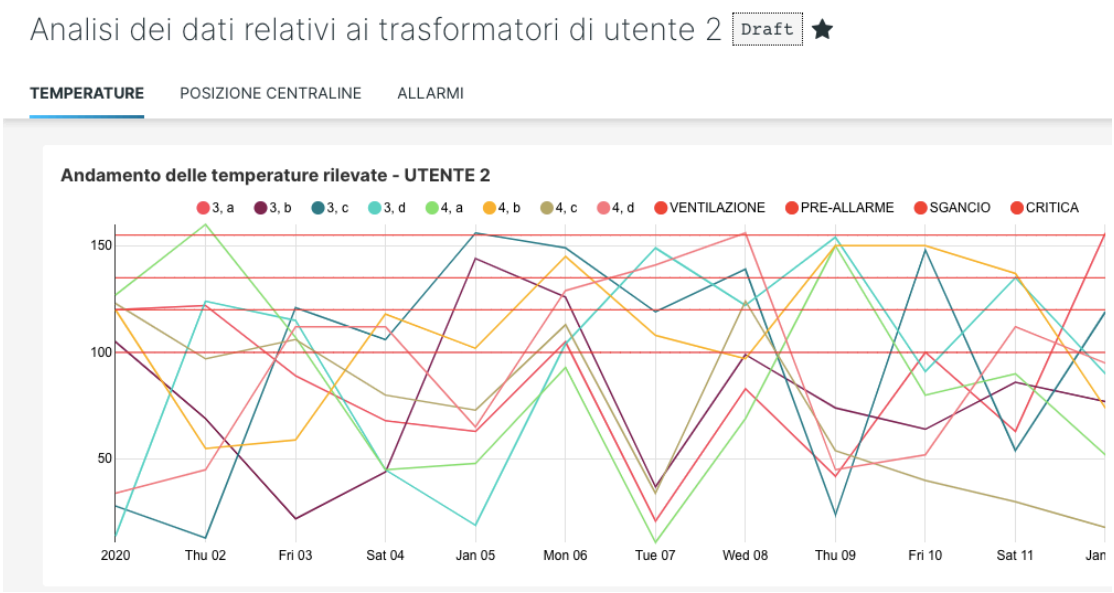


Figura 27: schermata con l'andamento temporale delle temperature delle centraline di utente 2.

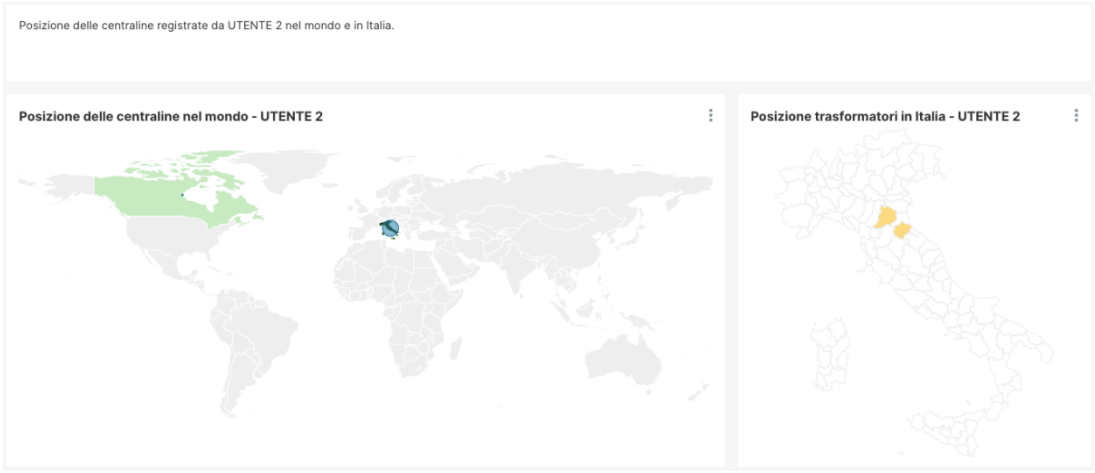


Figura 28: schermata della localizzazione dei trasformatori di utente 2

Figura 29: schermate dell'analisi degli allarmi rilevati dalle sonde di utente2

3.3. Le query utilizzate

SQL è un potente strumento per interrogare i database relazionali, cioè quei tipi di database che possono analizzare enormi quantità di dati, finché le relazioni tra le tabelle sono limitate. Per creare le *dashboard* è necessario inserire delle *query* in *SQL Lab* da Superset, che comunica direttamente con il database di MySQL.

Per selezionare le colonne di una tabella si usa il comando SELECT seguito, generalmente, da altre specifiche:

SELECT *elementi da analizzare*
FROM *tabella di origine*
WHERE *condizione*
GROUP BY *elementi di riferimento*
ORDER BY *espressione* **ASC/DESC**

Dove, FROM indica la tabella, o le tabelle, di origine dei dati. In caso di più tabelle selezionate viene restituito il loro prodotto cartesiano. Inoltre, se si scelgono più tabelle, si può imporre il comando "INNER JOIN … ON", che permette l'unione di due tabelle sulla base di un elemento in comune.

WHERE impone una condizione alla ricerca: gli elementi che non la soddisfano non vengono selezionati.

Infine, si possono raggruppare i dati sotto certe condizioni col comando GROUP BY, e si possono poi porre in ordine crescente (ASC) o decrescente (DESC).

Nel caso di studio, per visualizzare l'andamento delle temperature [Figura 13][Figura 15] prima di tutto, sono state selezionate le 3 tabelle (Centraline, Utenti e Sonde), i dati di tutti gli utenti, le temperature rilevate e la data delle analisi [Query 1]. Da Superset, poi, è possibile aggiungere i filtri per andare a esaminare solo i dati relativi ad un certo utente. In questo modo sono stati creati anche gli andamenti temporali visibili dai singoli utenti, come mostrato in Figura 27.

Per visualizzare la distribuzione spaziale delle centraline (sulla mappa e nel grafico a torta) è stata scelta la sola colonna "Nazione" dalla tabella "Centraline"; per concentrarsi, invece, sulla posizione in Italia, si sono presi solo i dati dalla colonna "ISO", con la condizione che la nazione fosse l'Italia ("ITA") [Query 2], [Query 3].

Per poter controllare la situazione globale degli eventi di allarme, è stata utilizzata la query [Query 4] ed è stato aggiunto un *filtro,* direttamente da Superset, che conta solo le temperature maggiori di 100°C: "SUM (temp_riv) > 100". Così si sommano solo le temperature maggiori di 100°C e, raggruppando gli elementi per data, ID della centralina e ID della sonda, si crea la tabella mostrata in Figura 29. Per portare i risultati anche sul grafico a torta e nella mappa, i dati sono stati raggruppati per città. Infine, sono stati contati i guasti di ogni centralina tramite l'operatore COUNT che, appunto, conta il numero di elementi di un determinato insieme, poi raggruppati per città e ID degli utenti [Query 4][Query 5].

La pagina con le statistiche comprende la tabella con la percentuale di allarmi per ogni trasformatore e le temperature massime rilevate.

Per la prima *slice*, la percentuale viene calcolata automaticamente in Superset, scegliendo in "Percentage Metrics" quale grandezza è da considerare per il calcolo [Figura 30].

Figura 30: schermata della selezione per il calcolo della percentuale

Invece, le temperature massime sono state calcolate attraverso il comando MAX, che trova il valore massimo in un certo insieme, che in questo caso è la colonna delle temperature rilevate, raggruppate per *user* e *centralina* [Query 6].

Per gli utenti non amministratori, che possono vedere soltanto i grafici e i dati relativi alle proprie centraline, sono state create delle dashboard apposite, con query simili a quelle appena descritte, ma con la particolarità che i risultati sono stati filtrati, inserendo, appunto, solo i dati relativi all'utente proprietario.

Alla fine del processo di scrittura delle *query* e creazione dei grafici, vengono pubblicate le dashboard che hanno come proprietari soltanto l'utente *admin* e coloro che sono in possesso dei trasformatori analizzati. L'utente *admin* è l'unico che ha accesso alla dashboard con i dati di tutti gli utenti.

```
SELECT id_user,
           id_c,
           data_riv,
           temp_riv,
           id_sonda,
           nome,
           cognome
    from (sonda S
          inner join centraline C on S.id_centr = C.id_c)
    inner join utenti U on U.id_u=C.id_user)
GROUP BY nome,
          id_user,
          id_sonda,
          id_c,
          cognome,
          DATE(data_riv)
ORDER BY `SUM(temp riv)` DESC;
```

Query 1: creazione del grafico dell'andamento delle temperature nel tempo.

```
SELECT * #l'asterisco indica che vanno selezionati tutti gli attributi di una tabe
lla.
    from centraline
GROUP BY nazione
ORDER BY count DESC;
```

Query 2: selezione degli elementi per la localizzazione dei trasformatori nel mondo.

```
SELECT *
    FROM centraline
WHERE nazione = 'ITA'
GROUP BY `ISO`
ORDER BY count DESC;
```

Query 3: query per la mappa delle centraline in Italia.

```
select nome,
           cognome,
           id_u,
           id_c,
           id_user,
           id_centr,
           id_sonda,
           data_riv,
           temp_riv
   from (utenti U
         inner join centraline C on U.id_u = C.id_user)
   inner join sonda S on C.id_c = S.id_centr) AS expr_qry
WHERE data_riv >= STR_TO_DATE('1920-09-21', '%Y-%m-%d')
GROUP BY id_u,
         nome,
         id_sonda,
         cognome,
         data_riv,
         id_c,
         id_centr,
         id_user
HAVING ((SUM(temp_riv) > 100))
ORDER BY `SUM(temp_riv)` DESC
LIMIT 1000;
```

Query 4: creazione dei grafici relativi agli allarmi.

```sql
select id_c,
           id_user,
           citta,
           ISO,
           temp_ext,
           id_sonda,
           id_centr,
           data_riv,
           temp_riv
    from (centraline C
          inner join sonda S on id_c=id_centr)
    where temp_riv>100) AS expr_qry
WHERE data_riv >= STR_TO_DATE('1920-09-21', '%Y-%m-%d')
GROUP BY id_c,
         citta,
         id_user
ORDER BY `COUNT(temp_riv>100)` DESC;
```

Query 5: conto degli allarmi per ogni centralina.

```sql
SELECT temp_riv,
           id_centr,
           id_sonda,
           data_riv,
           id_c,
           id_user
    from centraline C
    inner join sonda S on S.id_centr = C.id_c) AS expr_qry
WHERE data_riv >= STR_TO_DATE('1920-09-21', '%Y-%m-%d')
GROUP BY id_user,
         id_centr,
         DATE(data_riv)
ORDER BY `MAX(temp_riv)` DESC;
```

Query 6: temperature massime rilevate per ogni centralina ogni giorno

CONCLUSIONI

Come scritto nell'INTRODUZIONE, Apache Superset è stato sviluppato per andare incontro ai bisogni delle aziende per analizzare e visualizzare i *big data*. Grazie al lavoro svolto per questa tesi, è stato possibile studiare e scoprire molte delle funzionalità che offre questa applicazione. Gli obiettivi iniziali, come scritto prima, erano: riuscire a visualizzare i dati provenienti dai trasformatori; esaminarli ed organizzarli in modo da renderli facili e veloci da interpretare, creando un nuovo ruolo che dia permessi precisi ai clienti non *admin*.

Dopo il primo problema, poi risolto, legato all'installazione di Superset ed alla sua connessione con il database di MySQL, sono state provate le funzionalità disponibili. Grazie alle prove tentate è stato possibile comprendere al meglio il funzionamento di Superset, arrivando, infine, alla creazione delle *dashboard* più utili per lo studio dei dati creati per simulare la realtà. Così facendo, sono stati creati un certo numero di grafici che osservano il problema da più punti di vista: con i *Line Chart* vengono visualizzate le temperature rilevate nel tempo, dando una visione istantanea della situazione passata e presente; le tabelle sono utili per poter osservare una lista con i dati dei trasformatori; inoltre, sono stati utilizzati i grafici a torta per, i *Pie Chart*, per mostrare la quantità degli allarmi rilevati da ogni centralina; infine, la *Country Map* e la *World Map* sono state adoperate per studiare, nel modo più semplice possibile, la posizione e la distribuzione dei trasformatori nel mondo e, in modo specifico, nelle varie province italiane. È stato creato un nuovo ruolo, *Basic_user,* che era fondamentale per il raggiungimento degli obiettivi prefissati.

Complessivamente è stata un'esperienza molto interessante, che mi ha concesso l'opportunità di conoscere e studiare un'applicazione di grande importanza, applicandola ad una situazione verosimilmente reale. Sono molto soddisfatto del lavoro svolto, in modo particolare per essere riuscito a sviluppare il ruolo utile per la situazione ed aver creato i grafici con le mappe, perché sono state delle problematiche che mi hanno messo in difficoltà per buona parte del lavoro svolto con Superset.

INDICE DELLE FIGURE

BIBLIOGRAFIA

1. L. Duan and L. D. Xu, "Business Intelligence for Enterprise Systems: A Survey," in *IEEE Transactions on Industrial Informatics*, vol. 8, no. 3, pp. 679-687, Aug. 2012, doi: 10.1109/TII.2012.2188804.

2. M. Tvrdikova, "Support of Decision Making by Business Intelligence Tools," 6th International Conference on Computer Information Systems and Industrial Management Applications (CISIM'07), Minneapolis, MN, 2007, pp. 364-368, doi: 10.1109/CISIM.2007.64.

3. Microsoft Power BI: https://powerbi.microsoft.com/it-it/

4. Tableau, https://www.tableau.com/it-it

5. Google Data Studio, https://datastudio.google.com/overview

6. Apache Superset, https://superset.incubator.apache.org/

7. Metabase, https://www.metabase.com/

8. Redash, https://redash.io/

9. M. Petito, F. Fallucchi, and E. De Luca. "Create Dashboards and Data Story with the Data & Analytics Frameworks.", 2019, doi: 10.1007/978-3-030-36599-8_24.

10. Preset, https://preset.io/

11. GitHub, https://github.com/

12. Apache HTTP Server, https://httpd.apache.org/

13. Apache OpenOffice, https://www.openoffice.org/it/

14. Apache Software Foundation, https://www.apache.org/

15. MySQL, https://www.mysql.com/it/

16. Docker, https://www.docker.com/

17. SQLAlchemy, https://www.sqlalchemy.org/

18. Michael Bayer. SQLAlchemy. In Amy Brown and Greg Wilson, editors, **The Architecture of Open Source Applications Volume II: Structure, Scale, and a Few More Fearless Hacks** 2012 http://aosabook.org

19. DB-API 2.0, https://www.python.org/dev/peps/pep-0249/

20. Engine, https://docs.sqlalchemy.org/en/13/core/engines.html

21. Copeland, Rick. *Essential sqlalchemy*. " O'Reilly Media, Inc.", 2008.

22. Bayer, Mike. "SQLAlchemy Documentation." (2010).

23. Slide del corso di *Laboratorio di Ingegneria Biomedica* – Università di Bologna

24. Ling, Tok Wang. "A Normal Form For Entity-Relationship Diagrams." *ER* 1985 (1985): 24-35.

25. "Codici ISO 3166 delle Nazioni del mondo.", glossari.it, 2005-2007.

26. C. Fehily, "SQL: Visual quickstart guide." *Peachpit Press*, 2010.

27. Melton, Jim. "Sql language summary." *Acm Computing Surveys (CSUR)* 28.1 (1996): 141-143

28. Vicknair, Chad, et al. "A comparison of a graph database and a relational database: a data provenance perspective." *Proceedings of the 48th annual Southeast regional conference*. 2010.

29. Fan, Ju, Guoliang Li, and Lizhu Zhou. "Interactive SQL query suggestion: Making databases user-friendly." *2011 IEEE 27th International Conference on Data Engineering*. IEEE, 2011.