**Università degli Studi di Modena e Reggio Emilia**

**Facoltà di Ingegneria "Enzo Ferrari"**

---

Corso di Laurea Specialistica in Ingegneria Informatica

# Progetto e realizzazione del software "Solar Data Manager" per la gestione di dati da sensori

**Relatore:**
Prof. Bergamaschi Sonia

**Candidato:**
Luca Magnotta

**Correlatore:**
Ing. Mirko Orsini

**Anno Accademico 2009/2010**

Key words:
sensore
wrapper
query
SQLite
MySQL

# Indice

# Indice delle figure

# CAP 1. Introduzione

La ricerca di energie alternative è da sempre stato un tema fonte di grande interesse, soprattutto negli ultimi anni, in cui si è riscontrato una sempre maggiore richiesta di energia "pulita" e a costi il più possibile contenuti.
Questo ha portato allo sviluppo di una miriade di progetti indipendenti molti dei quali riguardanti lo sfruttamento dell'energia solare.
È tra questi che si inserisce il progetto della ditta SUNGEN s.r.l. di realizzazione di un impianto di produzione di energia solare, a cui sta contribuendo anche l'Università di Modena e Reggio Emilia.

Questa tesi illustra la progettazione e realizzazione di "Solar Data Manager" (SDM) cioè del modulo software per la gestione e visualizzazione dei dati di tale impianto.

Essa si articola nei seguenti capitoli:

**Cap. 2 - Il dominio applicativo e architettura del sistema:**
in questo capitolo viene brevemente spiegato per quale ambito ambientale è stato progettato il software Solar Data Manager;

**Cap. 3 - Scelta delle tecnologie:**
in questo capitolo vengono illustrate le tecnologie utilizzate in Solar Data Manager e il relativo processo di scelta;

**Cap. 4 - Progetto e implementazione di Solar Data Manager:**
questo capitolo è diviso in 2 sezioni:

- **Il Network Wrapper**
- **L'interfaccia grafica**

Nella prima viene illustrato il network wrapper nel dettaglio, con una descrizione dettagliata sia della struttura che del funzionamento;
Nella seconda sezione viene mostrata l'interfaccia grafica di Solar Data Manager tramite la quale è possibile visionare i dati dell'impianto;

**Cap. 5 - Analisi prestazioni del sistema:**
in questo capitolo viene fatto un'analisi sintetica dei risultati sperimentali delle prestazioni del network wrapper di Solar Data Manager;

**Cap. 6 - Conclusioni e prospettive future**

# CAP 2. Il dominio applicativo e architettura del sistema

Il sistema presentato in questa tesi è un software che permette di monitorare il comportamento di un impianto di generazione di energia solare.
Questo software è pensato per essere utilizzato in una rete di sensori che raccolgono informazioni ambientali e di funzionamento dell'impianto che vengono regolarmente inviati ad nodo centrale dove vengono aggregati e resi visibili per l'utente.

In questo capitolo vengono analizzati:

- L'architettura e l'hardware utilizzato per la realizzazione dell' intero impianto definiti dall'azienda committente;
- i requisiti del software e la struttura a blocchi per la realizzazione del software "Solar Data Manager" (SDM).

## 2.1. L'architettura dell'impianto

Nella figura sottostante viene illustrato in maniera molto generale l'architettura dell'impianto di produzione di energia solare ottenuto dalla proposta iniziale dalla ditta SUNGEN s.r.l. e dall'attività della presente tesi:

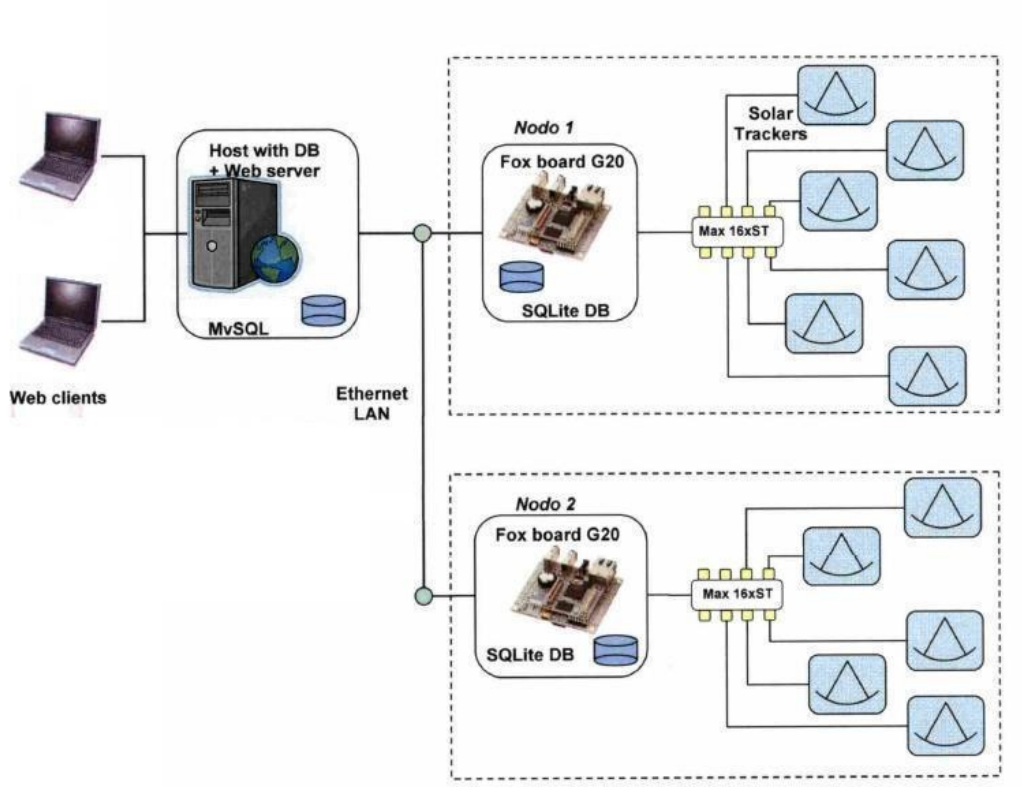



**Figura 1 - Schema della struttura dell'impianto**

L'impianto di produzione di energia solare è composto dai Sun Trackers (figura 2) ovvero da pannelli solari mobili che "inseguono" il sole in modo da avere sempre una angolazione normale rispetto all'azimuth corrente, così da massimizzare l'immagazzinamento di energia solare.

**Figura 2 - Foto di un prototipo di Sun Tracker**

I nodi gestiscono le informazioni prodotte dai sensori dei Sun Tracker (ST). La semantica proposta è 1:N, ovvero un nodo è capace di gestire più nodi, 16 per l'esattezza; pertanto l'impianto ha un nodo ogni 16 Sun Tracker. La figura 1 mostra come il nodo sia composto da una scheda dedicata e un database SQLite al cui interno sono raccolti i dati di funzionamento prodotti dai Sun Tracker: questo può essere definito il database di secondo livello o periferico.

I nodi non sono del tutto indipendenti ma sono monitorati da un server centrale MySQL che è composto da un host provvisto di web server e di un database centrale in cui sono memorizzati, in forma aggregata, i dati di funzionamento contenuti nei database periferici: questo database può essere definito come database di primo livello o centrale. Anche tra nodi e server centrale sussiste un rapporto di cardinalità 1:N.

I dati aggregati del database centrale vengono elaborati e resi visibili via web grazie al web server Tomcat; l'utente, quindi, utilizzando un qualsiasi web client, può prendere visione dei dati di funzionamento dell'impianto.

## 2.2. L'hardware dell'impianto

I Sun Trackers sono costituiti da hardware ideato e realizzato dalla ditta Sungen S.R.L.

I nodi invece sono composti da schede dedicate Fox Board G20 della ditta ACME Systems.
Questa scheda, che monta al proprio interno il micro-modulo Netus G20, ha le seguenti caratteristiche:

• ARM9 (e) CPU Atmel (tm) AT91SAM9G20 @ 400Mhz 64MB of 1.8 Volt BGA RAM at 32 bit
• 8Mbyte serial dataflash
• Two USB 2.0 host ports (12 Mbits per second)
• One Ethernet 10/100 port
• One USB device port (12 Mbits per second)
• One debug serial port (3.3v)
• Two serial ports (3.3v)
• One serial port for 4DSystems oLed displays (http://www.4dsvstems.com.au)
• On-board microSD socket (up to 8GB)
• 5VDC power supply input (compatible with PS5V1 A)
• Real Time Clock with on-board lithium backup battery
. GPIO lines (3.3v)
• 4 A/D converter lines
• Operative temperature range: -20° + 70°
• Image Sensor Interface (ITU-R BT. 601/656 12 bit).
•One Two-slot MultiMedia Card Interface (MCI). SDCard/SDIO and MultiMediaCard™ Compliant.

Sulla scheda è possibile installare diverse distribuzioni LINUX (Debian, Gentoo,...) ma l'unica in grado di assicurare tempi di startup rapidi a parere dell'azienda Sungen è risultato essere OpenWRT Crux 9.10 basato sul kernel Linux 2.6.30.

L'hardware del sistema del Server Centrale è stato definito solo in misura qualitativa: infatti sarà costituito da un Host ad alte prestazioni munito di un RDBMS (MySQL) e con una memoria di massa molto capiente in modo tale da non risultare essere un collo di bottiglia per il sistema di gestione dell'impianto.

Anche gli End-device da cui l'utente potrà visualizzare lo stato dell'impianto sono stati definiti in maniera qualitativa, infatti principalmente i dati devono essere visualizzabili su di un tablet-PC con uno schermo LCD di circa 10'', ma eventualmente anche netbook o laptop, ovvero HW caratterizzato da limitate prestazioni e uno schermo con media risoluzione.

## 2.3. Definizione dei requisiti del progetto “Solar Data Manager”

Il software Solar Data Manager (SDM) dovrà provvedere al recupero dei dati immagazzinati nei nodi sensore e alla visualizzazione degli stessi per mezzo di una apposita interfaccia utente (GUI).

Facendo riferimento alla figura 1 del paragrafo 1, si illustra come il software ricopra solo parte della struttura dell’intero impianto di produzione di energia solare.

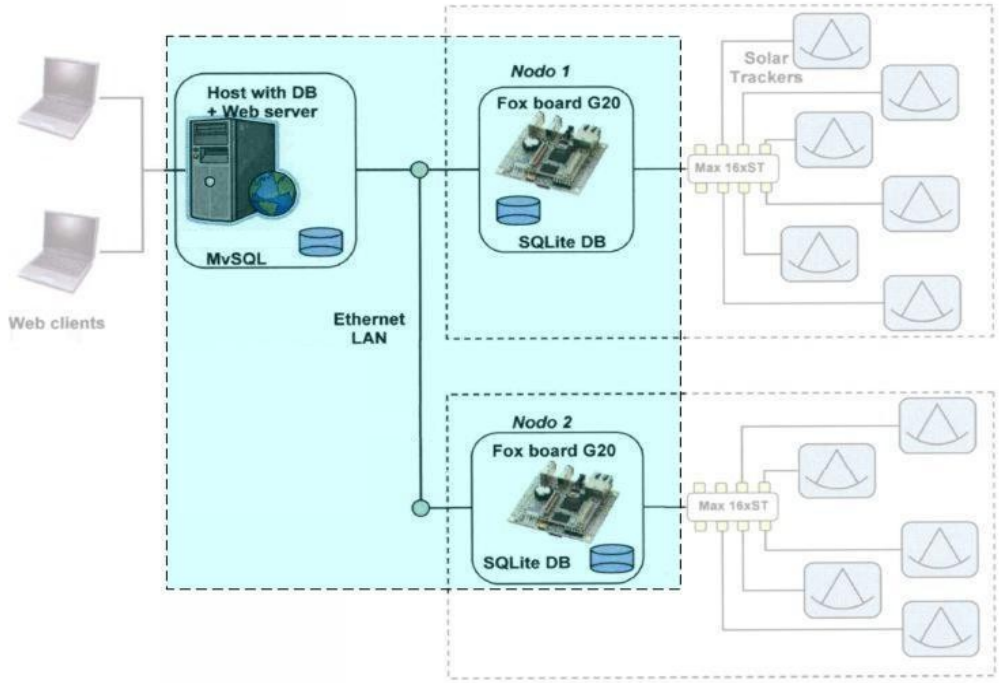



**Figura 3 - Schema della struttura di Solar Data Manager**

Nei paragrafi precedenti si è potuto notare la mancanza di indicazioni specifiche sulle caratteristiche del Server Centrale, pertanto si dovrà realizzare un software caratterizzato da una estrema flessibilità per poter adattare il sistema alle diverse ed eventuali esigenze che verranno introdotte durante lo sviluppo ed evoluzione del software.

### 2.3.1. Dettaglio dei requisiti di Solar Data Manager

- La modalità di reperimento dei dati deve essere trasparente all'utente finale;
- L'utente finale deve aver ben chiaro la stato di funzionamento del sistema: ovvero si richiede una GUI sintetica ma in grado di fornire una vista completa dei nodi dell'impianto;
- L'utente deve aver la possibilità di configurare il sistema;
- È necessario sviluppare il wrapping dei dati presenti nel nodo;
- I wrapper deve svolgere ogni tipo di operazione (query e trigger) sia sul Database Server che sul Database del nodo.

I vincoli che sono stati posti sono i seguenti:

- Il DBMS dei nodi sensore deve essere SQLite;
- I nodi devono usare il Sistema Operativo Linux;
- I nodi, che sono dispositivi remoti, vengono acceduti tramite Secure Shell, pertanto tramite una connessione protetta;
- I nodi sensore non possono svolgere operazioni computazionalmente pesanti;
- il Server centrale deve poter interagire con il database di ciascuno dei singoli sensori;
- I nodi sensore non possono mantenere i dati troppo a lungo a causa della limitata capacità della memoria interna, pertanto deve essere definita anche la modalità di eliminazione dei dati una volta caricati nel Database del server centrale
- Il sistema deve garantire buone performance: dato il contesto in cui il database dei nodi viene adoperato, le tabelle assumono abbastanza velocemente grandi dimensioni. Il database deve essere in grado di rispondere comunque in tempi ragionevoli alle query che riceve;

- Il sistema deve essere implementato con tecnologie open source: quest'ultimo requisito è fondamentale nell'ottica dell'abbattimento dei costi della componente Software dell'impianto;

## 2.4. Schema a blocchi del progetto

Il sistema visto in linee generali nel paragrafo 3 impone la realizzazione di 2 macroblocchi:

- **Nodo**
- **Server**

I 2 blocchi sono indipendenti e di diversa cardinalità: il Server è unico e i nodi invece sono N; il legame tra i 2 blocchi è costituito da una connessione LAN.
Per capire meglio la relazione tra i 2 blocchi si illustra di seguito lo schema del sistema che include le componenti del wrapper.

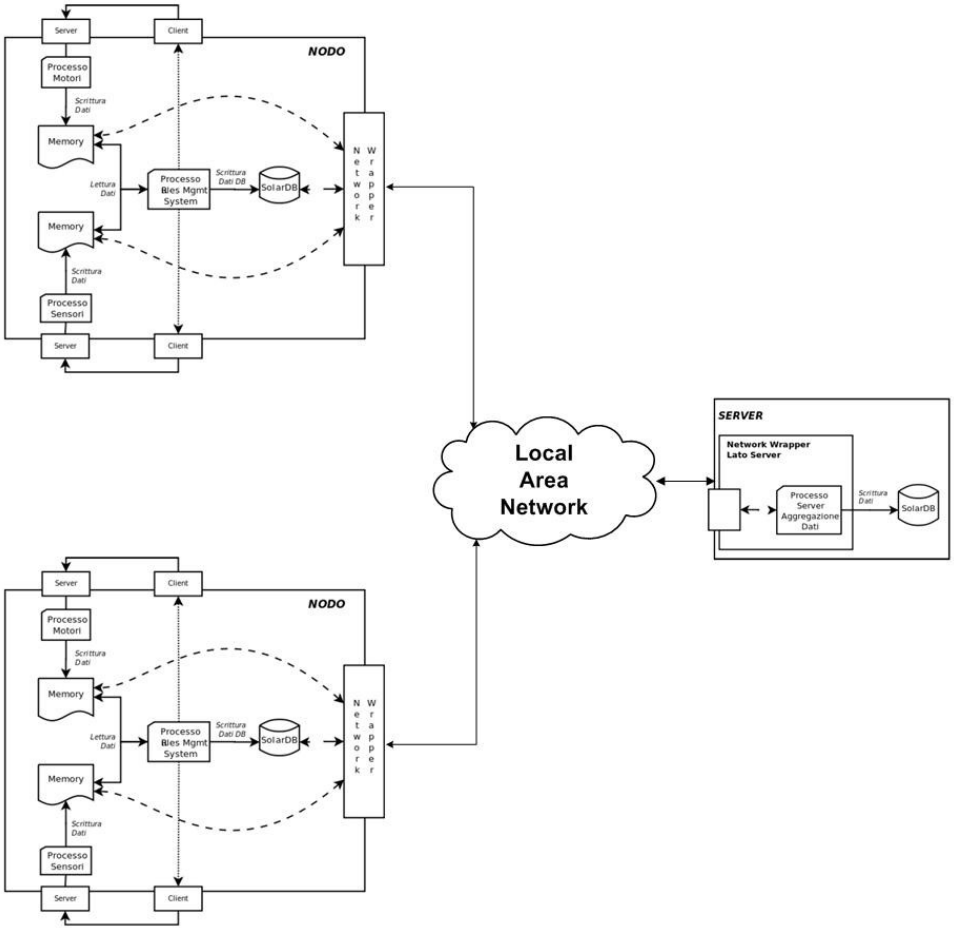



**Figura 4 - Schema a blocchi del sistema SDM**

La Figura 4 mostra tutti i blocchi che costituiscono il sistema; Si puntualizza che in questa tesi non verrà analizzata né la parte relativa all'Hardware dei nodi sensori ne la parte Software delegata all'immagazzinamento dei dati nel database locale del nodo poiché già realizzate dalla ditta SUNGEN s.r.l.

La figura 5 mostra in maniera più chiara quale sezione sarà oggetto di questa tesi:

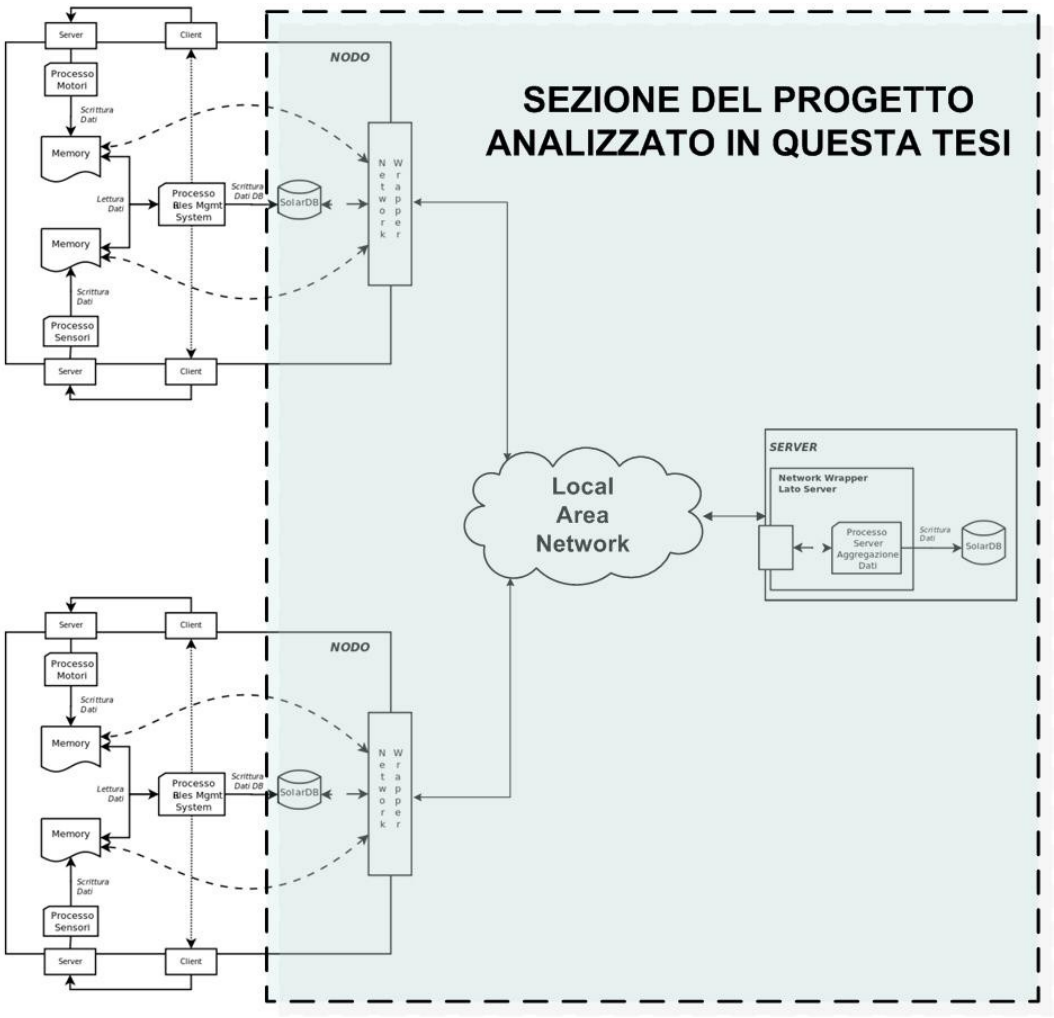



**Figura 5 - Schema a blocchi del sistema SDM: sezione all'oggetto di questa tesi**

# CAP 3. Scelta delle tecnologie

Per realizzare il software di controllo del sistema di gestione dell'impianto è stato necessario definire gli strumenti necessari per la realizzazione dello stesso, ovvero:

- **Il linguaggio di programmazione**
- **Il database (lato server poiché per i nodi è stato definito nell'analisi dei requisiti)**
- **I driver per interfacciare il database con il programma**
- **Il network wrapper** (cioè gli strumenti per l'interrogazione remota dei DBMS )
- **Il web server**

Nei paragrafi succedi verranno analizzati nella maniera più esaustiva possibile ciascuno dei componenti.

## *3.1. Il linguaggio di programmazione*

Il linguaggio di programmazione è ovviamente il punto di partenza per la realizzazione di qualsiasi software; in questo progetto è stato scelto Java, principalmente poiché esso è indipendente dalla piattaforma e per l'abbondanza di documentazione disponibile in rete.


Java è un linguaggio di programmazione orientato agli oggetti creato dalla Sun Microsystems. La piattaforma di programmazione Java è fondata sul linguaggio stesso, sulla Macchina virtuale Java (Java Virtual Machine o JVM) e sulle API Java. La peculiarità di Java consiste nel posti ad un livello intermedio tra interprete: infatti un programma Java viene compilato in un codice intermedio, detto

bytecode. I file di codice intermedio sono uguali per tutte le piattaforme e possono quindi essere scaricati da Internet ed eseguiti da un qualunque sistema capace di interpretarli. Qui sta il nocciolo delle capacità multipiattaforma di Java: è sufficiente implementare per ogni piattaforma un solo programma: l'interprete Java, meglio conosciuto come JVM o Java Virtual Machine, la cui definizione fa parte delle specifiche del linguaggio.
Java è inoltre un linguaggio molto usato in ambito accademico, e questo ha contribuito notevolmente nella scelta del linguaggio da utilizzare.

## 3.2. Il Database

### 3.2.1. Criteri di scelta del database

Il cuore del progetto è il network wrapper ovvero un sistema che gestisca reperimento delle informazioni contenute nei database dei sensori della rete. Le informazioni vengono generate da ogni singolo sensore immagazzinate nel proprio database dedicato e vengono inviate periodicamente al database del server centrale.
Pertanto una delle scelte fondamentali per la realizzazione del progetto è stata la definizione della struttura del database e in primo luogo la scelta DBMS utilizzare. Infatti esso dove rispondere a esigenze diverse.
Il server centrale deve rispondere alle seguenti caratteristiche:

- **buone performance**: per i motivi definiti nell'ambito della definizione dell'analisi dei requisiti di sistema;
- **pieno supporto al linguaggio SQL**: il DBMS non si può limitare al supporto di solo una parte del set d'istruzioni definite dallo standard SQL;
- **disponibilità API**: l'esistenza di API complete e ben documentate;
- **open source**: il DBMS deve essere open source.

Per i nodi sensori sono fondamentali sia **leggerezza** che **disponibilità API** , ma in questo caso non sono da prendere in considerazione poiché gia fissate nell'analisi dei requisiti del progetto.

In ambito Open Source le soluzioni che si sono affermate negli ultimi anni sono: MySQL[1], PostgreSQL[2] e SQLite[3]. Nei paragrafi successivi si analizzeranno le caratteristiche di ciascuno così da poterne fare un'analisi qualitativa e poter chiarire quale possa essere il più adatto per questo progetto

### 3.2.2. MySQL

MySql è il RDBMS Open Source che nell'ultimo decennio ha conosciuto la maggiore diffusione e popolarità cosa che si può constatare anche accedendo alla Homepage del sito ufficiale[4] in cui campeggia il sottotitolo "il database open source più famoso al mondo".

L'idea di questo RDBMS nasce nel 1975 ad opera di Michael Widenius per conto della svedese TcX aveva sviluppato uno strumento per la gestione di database chiamato "unireg". Nel 1994 la TcX, poi diventata MySQL AB, inizia a sviluppare applicazioni per il web utilizzando unireg, ma sfortunatamente il prodotto richiedeva troppe risorse per riuscire a generare dinamicamente pagine web. Si deve aspettare fino al 1996 per la realizzazione e rilascio di un prodotto adatto alle esigenze del mercato che supportasse SQL e che prendesse spunto dal DBMS mSQL (che in quegli anni era diventato la prima scelta dei programmatori open source[5]): MySQL 1.0 . Il prodotto si mostra subito vincente tant'è che nel giro i pochi anni ha preso il posto di mSQL il cui sviluppo è stato gradualmente abbandonato.

Il codice di MySQL era inizialmente di proprietà della società MySQL AB, veniva però distribuito con la licenza GNU GPL oltre che con una licenza commerciale. Fino alla versione 4.0, una buona parte del codice del client era licenziato con la GNU LGPL e poteva dunque essere utilizzato per applicazioni

commerciali. Dalla versione 4.1 in poi, anche il codice del client è distribuito sotto GNU GPL. Nonostante si tratti di software Open Source esso si rileva essere una grande fonte di introiti derivati dal supporto agli utilizzatori di MySQL tramite il pacchetto Enterprise, dalla vendita delle licenze commerciali e dall'utilizzo da parte di terzi del marchio MySQL.
Così nel 2008 la Sun Microsystem acquisì la MySQL AB per una cifra stimata di una miliardo di dollari.
Attualmente MySQL di proprietà della Oracle Corporation considerata da sempre una potenza di livello mondiale nel campo dei database, e come si può facilmente immaginare l'acquisizione della Sun da parte di Oracle datata 20 Aprile 2009 ha suscitato proteste nel mondo web "per il salvataggio di MySQL da Oracle" la cui petizione è stata sottoscritta dallo stesso Widenius.

MySQL è scritto in C e C++, la documentazione ufficiale offre anche un aiuto per chi avesse bisogno di provare a compilare il software su qualsiasi sistema operativo discretamente diffuso viene garantita anche un elevata compatibilità. Il linguaggio SQL di MySQL comprende anche numerose estensioni che sono tipiche di altri DBMS, quali PostgreSQL ed Oracle. In questo modo le query non standard scritte per altri DBMS saranno interpretate e salvo alcuni casi particolari, funzioneranno senza problemi.
MySQL è progettato e sviluppato per mettere in risalto le performance e la struttura interna del RDBMS ne è la dimostrazione: infatti sono stati sviluppati differenti motori interni di archiviazione (storage engines) il cui compito è di gestire i file che contengono le tabelle del database. I principali motori realizzano ognuno in maniera diversa un compromesso tra funzionalità (come il rispetto o meno delle regole ACID, definizione della strategia di docking, modalità di compressione dei record ecc.) e prestazioni. I SW di amministrazione MySQL (MySQL Managers) consentono all'utente di selezionare il motore più adatto alle proprie esigenze di archiviazione. Un'ulteriore punto di forza di MySQL è la grande compatibilità allo standard SQL, cosa che lo reso il RDBMS più adatto per i server web o di posta elettronica dove non sono richieste operazioni complesse quanto piuttosto dei tempi di risposta minimi anche in assenza di server dedicati

esclusivamente al servizio di database. Al giorno d'oggi MySQL è il riferimento per la realizzazione di contenuti web, infatti questo DBMS è una delle componenti fondamentali della principale piattaforma per lo sviluppo di applicazioni web ovvero la LAMP (acronimo di Lynux Apache MySQL Php/Perl); a questo si aggiunga che i server LAMP sono di comprovata stabilità e sicurezza infatti sia Wikipedia che Facebook (solo per citare n paio dei siti più famosi) sono basati su server LAMP

Un ultimo punto di forza di MySQL è il suo grado di sicurezza e stabilità garantito dal fatto che essendo un sistema aperto, e parzialmente libero, MySQL AB sfrutta la possibilità di analisi e rilascio di patch sia dalla community che segue lo sviluppo del codice sorgente del database sia dagli sviluppatori ufficiali. E' garantita in questo modo un' elevata disponibilità al rilascio di patch e questo contribuisce sicuramente ad un incremento totale della sicurezza del DBMS.

### 3.2.3. *PostgreSQL*

POSTGRESQL è un RDBMS che si pone sul mercato come principale alternativa sia rispetto ad altri prodotti liberi come MySQL che a quelli a codice chiuso come Oracle o DB2. Esso nasce in ambiente universitario pertanto il suo cardine va trovato nell'innovazione della tecnologia di gestione delle basi di dati, cosa sottolineata anche nella homepage del sito ufficiale con la frase "The world's most advanced open source database".


Il database è un progetto nato alla fine degli anni '70 nell'università di Berkeley per mano di Michael StoneBreaker con il nome di Ingres poi evoluto in Post-Ingres o più semplicemente Postgres (1985). Esso era stato ideato per rispettare i dettami della teoria classica dei RDBMS e puntava a fornire un supporto completo ai tipi di dati, in particolare la possibilità di definire nuovi tipi di dati, UDF (User Defined Function), User Defined Types. Vi era anche la possibilità di descrivere la relazione tra le entità (tabelle). Perciò non solo Postgres preservava l'integrità dei dati, ma era in grado di leggere informazioni da tabelle relazionate in modo naturale, seguendo le regole definite dall'utente. Il primo prototipo funzionante venne rilasciato nel 1988 e venne ufficialmente sviluppato fino al 1993 con il rilascio della versione 4 di Postgres che pose fine al progetto. In realtà il progetto non terminò poiché esso era stato rilasciato con licenza BSD, licenza che dà modo agli sviluppatori Open Source di ottenere una copia del software per poi migliorarlo a loro discrezione; così nel 1994 due studenti del Berkeley, Andrew Yu e Jolly Chen aggiunsero a Postgres un interprete SQL per rimpiazzare il vecchio QUEL (acronimo di QUEry Language) che risaliva ai tempi di Ingres. Il nuovo software venne quindi rilasciato sul web col nome di Postgres95. Nel 1996 cambiò nome di nuovo: per evidenziare il supporto al linguaggio SQL, venne chiamato PostgreSQL. Questa nuova versione ha conosciuto un successo ancora maggiore dei suoi predecessori soprattutto dopo il 2005 anno del rilascio della versione 8, la prima nativa per Windows.

PostrgreSQL attualmente è usato nei server di associazioni governative e non come CISCO, US Forestry Service e  American Chemical Society.
Va sottolineato nel frattempo Ingres non è stato abbandonato ma venne acquistato della Computer Associates. E' curioso anche come ancora oggi, per comodità di pronuncia si usi ancora il nome Postgres invece del nome ufficiale.

PostgreSQL nel mondo dei DBMS open source si può definire come il punto di unione tra ricerca pubblica e accademica. È rilasciato sul mercato con licenza BSD la quale potenzialmente permette a chiunque di prelevare il codice e di modificarlo e rilasciarlo con licenza proprietaria. Questo non garantisce dunque una continuità di libertà per i futuri utilizzatori di quel software. PostgreSQL è un database indipendente al 100%. Non soffre di "politiche aziendali" o problemi di mercato. Viene sviluppato da una grande comunità di programmatori ed è finanziato da molte aziende interessate al progetto. PostgreSQL è un DBMS open source. Funziona su tutti i sistemi operativi maggiormente noti.
Rispetta completamente il modello ACID (Atomicity, Consistency, Isolation, Durability). Include i tipi di dati di SQL92 e SQL99. Supporta anche la memorizzazione di grandi oggetti binari, immagini, suoni, video. È la gestione dei dati che lo diversifica maggiormente dagli altri prodotti presenti sul mercato: i database tradizionali infatti gestiscono i dati in "flat table", spetta quindi al programmatore tradurre i dati strutturati tipici di un linguaggio di alto livello in elementi di uno schema SQL (tabelle/entità e relazioni); PostgreSQL evita questo passaggio al programmatore (che si stima possa assorbire fino al 40% del tempo di sviluppo di un progetto) e permette agli utenti di definire nuovi tipi basati sui normali tipi di dato SQL, permettendo al database stesso di comprendere dati complessi. Per esempio, si può definire un indirizzo come un insieme di diverse stringhe di testo per rappresentare il numero civico, la città, ecc. pertanto si possono creare facilmente tabelle che contengono tutti i campi necessari a memorizzare un indirizzo con una sola linea di codice.
Le funzionalità di PostgreSQL si estendono al supporto ai triggers e alle stored procedures che possono essere realizzate in 4 linguaggi: pgSQL, Tcl, Perl e Phyton a discrezione del programmatore.

Questo RDBMS deve poter gestire delle basi di dati di dimensioni elevate (il database può avere dimensioni illimitate e una singola tabella sino a 32 TB di dimensione max[6]) pertanto esso viene comunque considerato "inferiore" rispetto ai suoi concorrenti commerciali ma esso viene considerato l'ideale in sistemi di produzione che gestiscono più TB di dati e dove le prestazioni sono secondarie rispetto all'efficienza computazionale. Da questo punto di vista PostreSQL è un rivale di Oracle piuttosto che di MySQL.
Quanto scritto finora mette in luce perché esso non spicchi per le prestazioni, nonostante questo vanno anche considerate le funzionalità che contribuiscono a incrementarne l'efficienza complessiva: la logica viene applicata direttamente dal server di database in una volta, riducendo il passaggio di informazioni tra il client ed il server; centralizzazione del codice di controllo sul server, permette di gestire la sincronizzazione della logica tra molteplici client e i dati memorizzati sul server contribuendo al miglioramento dell'affidabilità; infine inserendo livelli di astrazione dei dati direttamente sul server, il codice del client può essere più snello e semplice.

Per concludere va aggiunto che PostgreSQL è molto apprezzato soprattutto da coloro che lo usano e/o sviluppano, sia utenti che aziende. È stato anche considerato addirittura il miglior DBMS presente sul mercato, infatti ha ricevuto sia il "Linux New Media Award" e per tre volte il "The Linux Journal Editors' Choice Award".

### 3.2.4. SQLite

SQLite è una libreria software scritta in linguaggio C che implementa un DBMS SQL di tipo ACID e auto-contenuto incorporabile all'interno di applicazioni. Il suo creatore lo ha rilasciato nel pubblico dominio, rendendolo utilizzabile quindi senza alcuna restrizione, cosa che si stima ne abbia fatto il database engine più diffuso al mondo, dato tutto sommato attendibile se si pensa che ogni copia di Firefox, MacOS o iPhone (solo per citarne alcuni) utilizza un database interno SQLite.

La sua storia è molto breve: infatti nasce nella primavera del 2000 per mano di D. Richard Hipp nell'ambito di un progetto della General Dynamics commissionato dalla Marina degli Stati Uniti per la realizzazione di SW di gestione del sistema di gestione dei missili delle navi da guerra che fino ad allora facevano uso di database IBM Informix. L'idea su cui si basava SQLite è di permettere ai programmi di operare senza l'istallazione o l'amministrazione di un RDBMS. Nell'Agosto del 2000 viene rilasciata la versione 1.0 basata su gdbm (GNU Database Manager) sostituita da un più classico B-tree nella versione 2.0 che inoltre introduceva il supporto per le transazioni. Attualmente è disponibile la versione 3.7.2

SQLite è diverso dagli altri DBMS, infatti non è un processo stand alone ma può essere incorporato all'interno di un altro programma, non ha un processo server separato ed esegue le operazione di lettura e scrittura direttamente sul disco. Il database SQL completo di tabelle, indici, triggers e viste è composto da un singolo file indipendente dalla piattaforma utilizzata (viene permessa la copia diretta del database da un sistema a 32-bit a uno a 64-bit e viceversa ).

SQLite è una soluzione serverless: questo comporta un alleggerimento del carico computazionale della macchina su cui è istallato in quanto non si deve creare un processo server separato che dovrebbe essere configurato inizializzato e gestito anche in caso di errore; in questo modo si è riusciti ad ottenere un database

enigine a “configurazione-zero”: un programma che contiene SQLite al suo interno non ha bisogno di supporti amministrativi o di setup prima di poter essere messo in esecuzione, l’unico requisito richiesto è la capacita di eseguire un qualsiasi accesso sul disco. Sono presenti anche problemi dovuti alla mancanza del server che infatti garantirebbe una maggiore protezione dell’applicazione client dai bug, il processo server essendo unico e persistente è in grado di controllare gli accessi al database con più precisione garantendo la una migliore gestione della concorrenza.

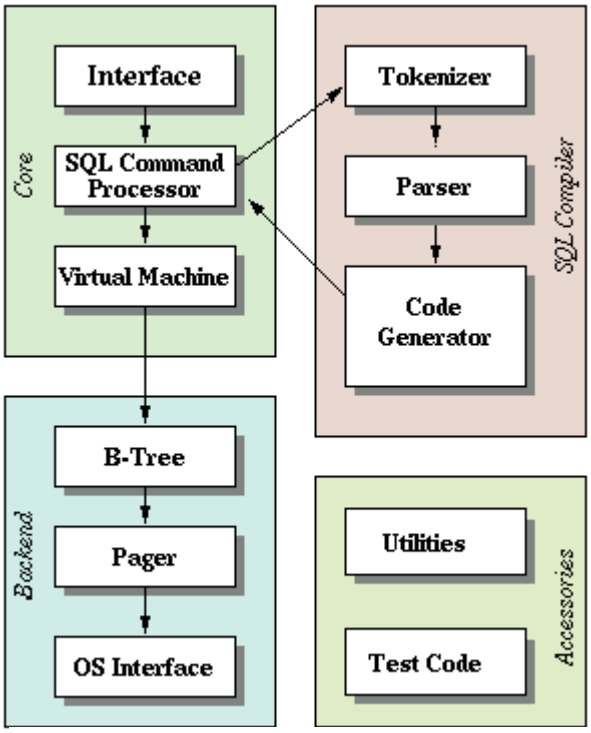



**Figura 6 - Schema a blocchi di SQLite**

Il fatto che sia autocontenuto comporta il supporto di numero minimo di librerie esterne. Questa scelta è stata fatta affinchè possa essere utilizzato in dispositivi embedded che in genere non sono provvisti della infrastruttura di supporto di un desktop computer ed inoltre in questo modo SQLite può essere usato su ampia gamma di computer che avranno configurazioni molto differenti.

Altri punti di forza di SQL sono: la compattezza della libreria che infatti è inferiore a 500KB, il supporto allo standard SQL e SQL92, la semplicità d’utilizzo dell’ API, la grande capienza (il limite dichiarato per la dimensione del database è di 2TB) e la mancanza di dipendenze esterne.

I limiti di cui soffre SQL invece sono: la mancanza di store procedures, la mancanza di protocolli di rete, la mancanza di gestione dei diritti di accesso e della concorrenza, la gestione dei lock del file di database che è integralmente demandata al gestore del file system della macchina su cui è in esecuzione SQLite, mancanza di una cache per le query, mancanza di supporto a costrutti SQL come la Right Join o la Full Outer Join, mancanza di possibilità di utilizzare

appieno il comando Alter Table che infatti è limitato alla modifica del nome della tabella e all'aggiunta di nuove colonne in coda alla stessa (non è possibile cancellare delle singole colonne), mancanza di supporto alla scrittura diretta delle viste e impossibilità di utilizzare trigger di tipo "For Each Statement".
I limiti elencati in precedenza sono dovuti principalmente all'utilizzo previsto per SQLite che infatti si propone come database di supporto alle applicazioni. Infatti si stima che siano state distribuite oltre 500 milioni di copie di questo database[7] ripartite principalmente in: 300 milioni copie di Mozilla Firefox, 20 milioni di PC della Apple (ognuno dei quali ha più copie al suo interno), 20 milioni di siti web scritti in PHP, 45 milioni di copie di Skype, 10 milioni di istallazioni di Solaris; a questi vanno aggiunte le copie di McAfee Antivirus, gli Smartphone che usano Symbian o iPhoneOS, oltre a tutti i progetti indipendenti che usano come database proprio SQLite.

### 3.2.5. *Analisi comparativa di MySQL, PostgreSQL, SQLite*

In questo paragrafo vengono analizzate le prestazioni dei 3 DBMS sulla base dei test svolti dal team di sviluppo di SQLite e disponibili online[8] .
Il test è stato realizzato utilizzando come piattaforma un PC con un processore Athlon da 1.6 GHz, RAM di 1GB e un sistema Operativo RedHat Linux 7.2 .

**Test 1: 1000 INSERT**

Questo caso è di particolare interesse poichè la funzione principale del server centrale è quella di raccolta dai dati provenienti dai nodi remoti, operazione che in linguaggio SQL si traduce nella esecuzione dello statement INSERT.

**Codice SQL**

```
CREATE TABLE t1(a INTEGER, b INTEGER, c VARCHAR(100));
INSERT INTO t1 VALUES(1,13153,'thirteen thousand one
hundred fifty three');
INSERT INTO t1 VALUES(2,75560,'seventy five thousand
five hundred sixty');
```

... 995 linee omesse

```
INSERT INTO t1 VALUES(998,66289,'sixty six thousand two
hundred eighty nine');
INSERT INTO t1 VALUES(999,24322,'twenty four thousand
three hundred twenty two');
INSERT INTO t1 VALUES(1000,94142,'ninety four thousand
one hundred forty two');
```

**Tempi di risposta**

| | |
|---|---|
| **PostgreSQL:** | ***4.373*** |
| **MySQL:** | ***0.114*** |
| **SQLite 2.7.6** | ***13.061*** |
| **SQLite 2.7.6 (nosync)** | ***0.223*** |

**Tabella 1 - Analisi prestazioni: 1000 INSERT**

**Analisi dei risultati ottenuti**

MySQL ha mostrato la sua grande efficienza risultando molto più veloce dei concorrenti.
SQLite è risultato essere nella configurazione sincronizzata il più lento, questo perché non è provvisto di un server centrale che coordini gli accessi, quindi deve chiudere e riaprire ogni volta il database file ed invalidare la cache per ogni transazione. In questo test ogni statement produce una transazione quindi il database file deve essere chiuso e riaperto 1000 volte.

D'altro canto la versione asincrona di SQLite si rivela molto più veloce di PostreSQL ma non ancora ai livelli di MySQL.
La grande differenza tra versione sincrona e asincrona è data dalla chiamata della funzione fsync() da parte di SQLite che verifica che per ogni transazione i dati siano stati scritti in maniera sicura sul disco prima di poter procedere con una nuova transazione; per tutta la durata del completamento dell'operazione di I/O SQLite è costretto a rimanere nello stato di "idle".

**Test 2: 25000 INSERT in un'unica transazione**

Questo test è stato aggiunto per completezza poichè nel progetto non è previsto questo approccio per l'inserimento dati.

**Codice SQL**

```
BEGIN;
      CREATE TABLE t2(a INTEGER, b INTEGER, c
       VARCHAR(100));
      INSERT INTO t2 VALUES(1,59672,'fifty nine thousand
       six hundred seventy two');

     ... 24997 linee omesse

      INSERT INTO t2 VALUES(24999,89569,'eighty nine
       thousand five hundred sixty nine');
      INSERT INTO t2 VALUES(25000,94666,'ninety four
       thousand six hundred sixty six');
COMMIT;
```

**Tempi di risposta**

| | |
|---|---|
| **PostgreSQL:** | ***4.900*** |
| **MySQL:** | ***2.184*** |
| **SQLite 2.7.6** | ***0.914*** |
| **SQLite 2.7.6 (nosync)** | ***0.757*** |

**Tabella 2 - Analisi prestazioni: 25000 INSERT in unica transazione**

**Analisi dei risultati ottenuti**

In questo caso è SQLite a mostrarsi il più efficiente: tutte le INSERT vengono eseguite in una singola transazione, quindi SQLite non deve come nel caso precedente eseguire numerosi cicli di accesso in I/O tra gli statement pertanto la finzione fsync() viene chiamata solo una volta.

In questo caso MySQL non si mostra tanto efficiente come nel caso precedente, ma il fanalino di coda rimane PostgreSQL .

**Test 3**: **100 SELECT senza indicizzazione dei dati**

Questo test è molto interessante poichè una ulteriore funzionalità richiesta al Database Server è quella di selezionare dei dati aggregati per la visualizzazione da parte dell'utente

**Codice SQL**

```
BEGIN;
     SELECT count(*), avg(b) FROM t2 WHERE b>=0 AND
       b<1000;
     SELECT count(*), avg(b) FROM t2 WHERE b>=100 AND
       b<1100;

     ... 96 linee omesse
```

```
      SELECT count(*), avg(b) FROM t2 WHERE b>=9800 AND
        b<10800;
      SELECT count(*), avg(b) FROM t2 WHERE b>=9900 AND
        b<10900;
COMMIT;
```

**Tempi di risposta**

| | |
|---|---|
| **PostgreSQL:** | ***3.629*** |
| **MySQL:** | ***2.760*** |
| **SQLite 2.7.6** | ***2.494*** |
| **SQLite 2.7.6 (nosync)** | ***2.526*** |

**Tabella 3 - Analisi prestazioni: 100 SELECT senza indicizzazione**

**Analisi dei risultati ottenuti**

Questo test esegue 100 query su una tabella composta da 25000 entry senza un indice per cui è richiesta la scansione dell'intera tabella. Come nei casi precedenti si riscontrano prestazioni simili per MySQL e SQLite mentre PostreSQL pecca in efficienza.

I 3 test analizzati sono stati sufficienti per fare un'analisi qualitativa dei 3 DBMS presi in esame e poter definire quello più adatto per la realizzazione del sistema di gestione dell'impianto.

### 3.2.6. Il database scelto e conclusioni

L'analisi delle pagine precedenti ha consentito una rapida selezione del DBMS più adatto per il progetto. Come spiegato in precedenza il database da scegliere deve avere buone doti di efficienza, affidabilità, disponibilità di API e rispetto dello standard SQL.
Dato come requisito principale la velocità (efficienza) l'opzione PostgreSQL è stata scartata.
La scelta si è dunque ristretta a MySQL ed SQLite, ma problemi come la non ottimale gestione della concorrenza oltre alla mancanza di supporto di rete che affliggono SQLite, hanno portato alla scelta finale di MySQL come DBMS per il Modulo Server.

L'utilizzo di MySQL garantisce infatti una ottima efficienza ed efficacia, abbondanza di API, rispetto dello standard SQL, una gestione ottimizzata per l'accesso concorrente alle tabelle, buona gestione dei permessi di accesso, facendo di questo DBMS la scelta più adatto per questo progetto.

In questo progetto vengono, quindi, utilizzati 2 differenti DBMS:

- **MySQL** come database centrale
- **SQLite** come database periferici

## 3.3. I driver

Definiti i Database e il linguaggio di programmazione che saranno utilizzati nella realizzazione del progetto, bisogna definire i driver.
Con il termine driver si intende uno strumento capace di rendere le diverse sorgenti di informazione conformi ad uno standard interno o esterno, in altre parole deve fornire una interfaccia comune per database eterogenei.

Considerando le strutture dati utilizzate, il driver è costituito dalle API fornite da JDBC[9] , la cui struttura è illustrata nella figura sottostante estratta dal sito della Sun:

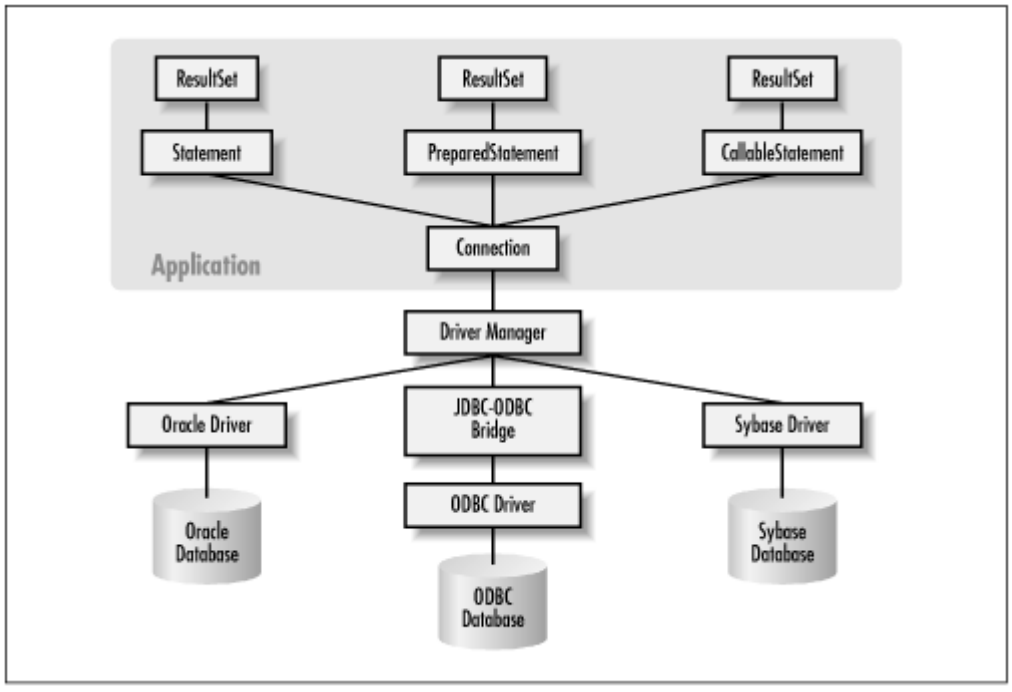



**Figura 7 - JDBC - Schema a blocchi**

La figura 7 mostra le diverse tipologie di driver, infatti i driver JDBC si dividono in 4 tipologie (la figura prende in considerazione solo tipo 1 e 2):

- **Tipo 1: bridge ODBC-JDBC**
- **Tipo 2: JDBC-API Native**
- **Tipo 3: JDBC-Net Java puro**
- **Tipo 4: 100% Java puro**

La ricerca si è concentrata sui driver di tipo 4 sia per per SQLite che MySQL.
SQLite propone una lista di diversi Wrapper per java tra cui:

- sqlite4java: un vero e proprio wrapper che estende le funzionalità di JDBC proponendo un'interfaccia che garantisce alte prestazioni e minimizzando la produzione di "spazzatura";
- Christian Werner Javasqlite: la libreria più diffusa e utilizzata, non immune però da difetti come possibili degradi delle prestazioni del database dovuti all'utilizzo del metodo finaline(), naming di metodi e classi non user-friendly, rallentamenti notevoli per i resultSet voluminosi;
- Xerial: Sqlite JDBC driver: driver JDBC considerato il migliore tra quelli disponibili;
- SQLiteJDBC: driver JBDC che si caratterizza dall'utilizzo di codice java puro, grazie alla traduzione delle librerie scritte in C di SQLite in codice java grazie all'emulatore NestedVM che funziona nel seguente modo:

    - Installa un **cross compilatore GCC** (che quindi accetta qualunque linguaggio da esso supportato, come **C, C++, Fortran**) per processori **MIPS**
    - Trascodifica il **codice macchina di Mips** in **bytecode Java**
    - Tramite il linker sostituisce le chiamate a librerie di I/O nelle corrispondenti **librerie di NestedVM e Java** (quindi I/O su files sono salvaguardati).

L'abbondanza di documentazione e semplicità d'utilizzo di quest'ultima libreria JDBC citata ha fatto propendere per l'utilizzo di questa soluzione per l'implementazione del wrapper dei database dei nodi-sensore.

Per MySQL invece la ricerca è stata molto più semplice in quanto il sito ufficiale di MySQL propone come soluzione (di tipo 4) MySQL Connector/J, il cui schema illustrato nella figura 8.

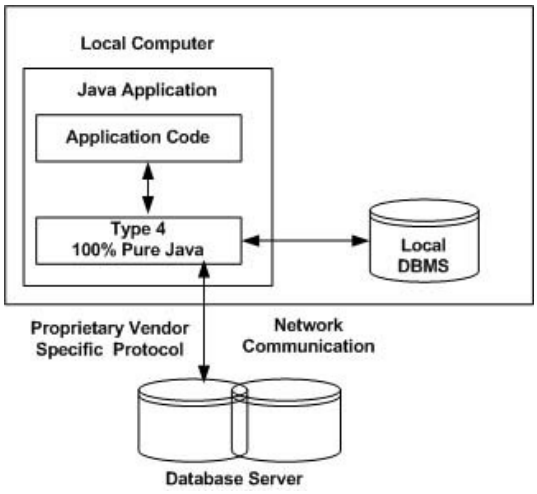



**Figura 8 - JDBC (Type 4) - Schema a blocchi**

Anche esso nei test effettuato si rivelato essere adatto per l'utilizzo nel SW realizzato.

## 3.4. Gli strumenti per il funzionamento di rete del DBMS

La realizzazione della sezione di network wrapping del Software ha richiesto infine la ricerca di soluzioni che realizzassero una architettura client-server per l'accesso ai database.
MySQL è pensato per essere utilizzato in ambito di strutture client-server, mentre la stessa cosa non si può dire di SQLite che invece è dichiaratamente serverless.

Il network wrapper che deve essere realizzato prevede che il Server centrale possa interagire con ciascuno dei singoli sensori: questo vincolo infatti ha posto il non semplice problema di rendere anche SQLite un DBMS accessibile da remoto.
In assenza di questo vincolo infatti sarebbe stato possibile creare una soluzione "unidirezionale": in altre parole il nodo-sensore invia autonomamente i dati contenuti nel database locale al database centrale MySQL, senza che il server centrale invii comandi si sincronizzazione o traduca delle query globali in query locali. Questa soluzione avrebbe garantito prestazioni superiori, ma l'impossibilità di avere una comunicazione dal Server al Nodo-Periferico ha portato dalla imposizione del vincolo di "bidirezionalità".

Dunque è stato necessario risolvere il problema dell'accesso ai dati dei nodi da parte del Server centrale.
Una soluzione sarebbe potuta essere quella di condividere in rete i file di database ma essa è consigliata dagli stessi sviluppatori di SQLite poiché troppo dipendente dalla struttura dei FileSystem delle macchine su cui risiedono i database, sempre ammesso che tali FS prevedano protocolli per la condivisione in rete. Pertanto questa soluzione è stata da subito accantonata a favore di software per il "networking reale".
Dalla nascita di SQLite sono stati realizzati alcuni progetti indipendenti accessibili direttamente dalla pagina di documentazione di SQLite[10]:

REAL SQL Server[11]: Software per sviluppatori per la conversione di database monoutente in multiutente. Tale soluzione però è stata scartata poiché la licenza non ne permette una applicazione in ambiti commerciali .

SQLiteServer[12]: TCP/IP server per SQLite, progetto I cui sorgenti sono reperibili in rete ma purtroppo non più sviluppato e fermo alla versione 0.2, inoltre privo di una documentazione specifica.

uSQLiteServer and Client[13]: SW che implementa un protocollo ASCII per l'accesso a database SQLite per mezzo di socket. È provvisto di un client scritto in C che consente l'accesso al Server SQLite usando funzioni di tipo exec/callback. Implementa anche controlli di sicurezza molto semplici per l'accesso al database. Inoltre è Open Source.
Queste caratteristiche ne hanno fatto un valido candidato per l'utilizzo nell'ambito del progetto, ma anche questa soluzione è stata accantonata a causa della grande difficoltà riscontrata nel trovare un'adeguata documentazione delle API di questo SW.

SQL4Sockets[14]: SW che implementa il funzionamento client-server per SQLite utilizzando delle socket TCP. Rilasciato con licenza freeware, non vengono forniti i codici sorgenti ma solo i file compilati per win32 e unix. Anch'esso pertanto scartato.

SQLite ODBC Driver[15]: API per SQLite che ne estendono le funzionalità per garantirne anche l'utilizzo in rete, ancora in fase sperimentale e quindi non sfruttabile a per il progetto da realizzare

A seguito dei problemi presentati dai SW già presenti, è stato scelto di creare una soluzione personalizzata per la gestione da remoto dei file presenti nel database SQLite che sfruttasse il protocollo SSH per la comunicazione remota.

## 3.5. Il protocollo di comunicazione remota: Secure Shell

L'esigenza generare un canale di comunicazione bidirezionale tra nodo e server ha portato all' utilizzo di Secure Shell[16] (SSH) come protocollo per il trasporto dati all'interno della Local Area Network.

SSH è un protocollo di comunicazione nato per rimpiazzare i comandi Berkeley r* (rsh, rlogin, rcp) con le rispettive versioni sicure (ssh, slogin, scp). SSH fornisce una infrastruttura per connessioni crittografate nonché autenticazione forte tra host e host e tra utente e host. Risolve molti dei problemi di sicurezza noti dei protocolli TCP/IP come l'IP spoofing (falsificazione dell'indirizzo IP del mittente), il DNS spoofing (falsificazione delle informazioni contenute nel DNS) e il routing spoofing (falsificazione delle rotte intraprese dai pacchetti e instradamento su percorsi diversi).

### 3.5.1. L'architettura di SSH

Il funzionamento[17] di SSH è molto simile a quello di SSL[18]. Entrambi possono instaurare una comunicazione cifrata autenticandosi usando "host key" ed eventualmente certificati che possono essere verificati tramite una autorità fidata.

La figura 9 illustra come si instaura una connessione:


**Figura 9 - SSH - Instaurazione connessione**

1. Il client (Host A) ed il server (Host B) si scambiano le loro chiavi pubbliche. Se la macchina del client riceve per la prima volta una data chiave pubblica, SSH chiede all'utente se accettare o meno la chiave.

2. Successivamente client e server negoziano una chiave di sessione che sarà usata per cifrare tutti i dati seguenti attraverso un cifrario a blocchi come il Triplo DES o Blowfish. Il Transport Layer Protocol si occupa di questa prima parte, cioè di fornire autenticazione host, confidenzialità, integrità e opzionalmente compressione. Tale protocollo gira tipicamente su una connessione TCP, ma potrebbe essere usato anche su un qualsiasi altro flusso di dati affidabile.

   Lo User Authentication Protocol autentica l'utente del client sul server. Sono attualmente supportati tre tipi di autenticazione:

   2.1. **Con chiave pubblica:** il client invia un pacchetto firmato con la propria chiave privata. La firma è verificata dal server tramite la chiave pubblica del client, che il server può possedere da eventuali connessioni precedenti. Nel caso in cui il server non possieda la chiave pubblica del client, questo metodo fallisce;
   2.2. **Con password:** all'utente sul client viene presentato il solito prompt per l'inserimento della password. Generalmente questo metodo viene utilizzato solo alla prima connessione ad un server;
   2.3. **Host based:** simile al metodo rhosts di UNIX, ma più sicuro, in quanto aggiunge ad esso la verifica dell'identità dell'host tramite la "Host key".

   Il server guida l'autenticazione, inviando al client la lista contenente i metodi di autenticazione supportati. Il client sceglie quello che considera più conveniente.

3. Infine, dopo una autenticazione avvenuta con successo, tramite il Connection Protocol comincia la sessione: il client può richiedere una shell remota, l'esecuzione di un comando , un trasferimento di file sicuro, ecc. Il protocollo

divide la connessione in canali logici; tutti questi canali sono multiplexati in una singola connessione. In questo modo è possibile accedere a più servizi con un singolo "tunnel cifrato".

L'architettura di SSH e la sua divisione in protocolli è schematizzata nell'immagine seguente:

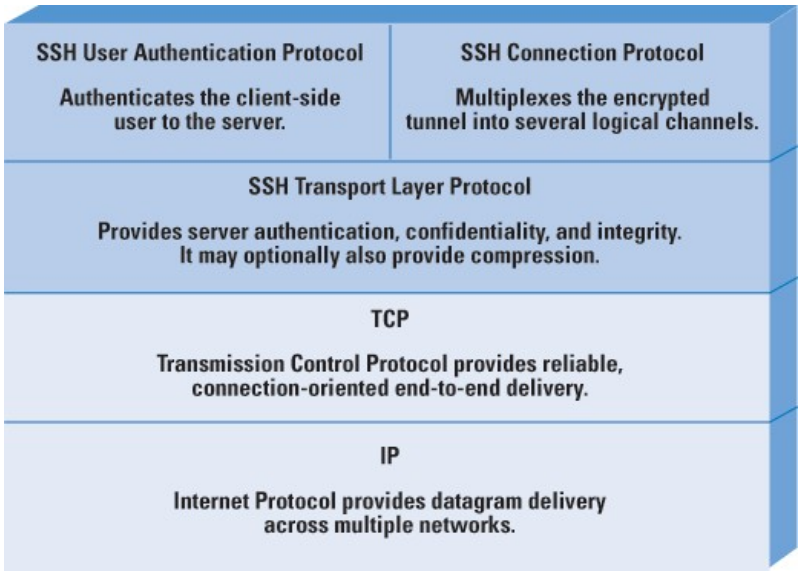



**Figura 10 - SSH - Stack dei protocolli**

I **Layer TCP/IP** garantiscono la comunicazione end-to-end in maniera affidabile.
L' **SSH Transport Layer Protocol** è un protocollo di trasporto sicuro a basso livello. Fornisce crittografia forte, autenticazione degli host crittografica e protezione dell'integrità. Questo protocollo non si occupa dell'autenticazione: questo compito è demandato al protocollo di livello superiore. Il protocollo è stato progettato per essere semplice, flessibile, per permettere la negoziazione dei parametri e per minimizzare il numero di round-trip. Tutti gli algoritmi (scambio di chiavi, chiave pubblica, cifratura simmetrica ed hash) sono negoziati. Ci si aspetta che in molti ambienti siano necessari solo 2 round-trip per completare lo scambio delle chiavi, l'autenticazione del server, la richiesta dei servizi e la notifica dell'accettazione della richiesta dei servizi. Il caso peggiore è 3 round-trip.

L'**SSH Authentication Protocol** è un protocollo di autenticazione dell'utente general-purpose. Gira al di sopra dell'SSH Transport Layer Protocol. Questo protocollo assume che il protocollo sottostante fornisca protezione dell'integrità e confidenzialità.
Quando questo protocollo parte, riceve l'identificatore di sessione dal protocollo al livello inferiore che identifica univocamente questa sessione.
Il **Connection Protocol** è il protocollo che deve gestire sessioni interattive di login, esecuzione remota di comandi, inoltro di connessioni TCP/IP. Il service name per questo protocollo è "ssh-connection".
La connessione è divisa in canali logici; tutti questi canali sono multiplexati in un singolo tunnel cifrato.Tutti gli pseudo-terminali, connessioni inoltrate, ecc. sono considerati canali, essi possono essere aperti sia dal client che dal server, sono identificati da numeri(ad entrambe le parti), che possono essere differenti tra lato client e lato server.
Le richieste di apertura di un canale conterranno il numero di canale del mittente ed ogni altro messaggio relativo ad un canale conterrà il numero del canale del ricevente. I canali hanno un meccanismo di controllo del flusso, questo significa che non possono essere inviati dati su di un canale se prima non si è ricevuto messaggio circa la disponibilità della "window space".

Si possono avere richieste che coinvolgono globalmente lo stato della parte remota e che sono indipendenti dai canali. Un esempio è una richiesta di iniziare un TCP/IP forwarding per una porta specifica. Tutte queste richieste usano il seguente formato:

```
byte      SSH_MSG_GLOBAL_REQUEST
string    request name
boolean   want reply
... dati di risposta specifici della
request
```

Il ricevente risponderà a questo messaggio con SSH_MSG_REQUEST_SUCCESS oppure SSH_MSG_REQUEST_FAILURE se il campo “want reply” è stato posto a true.
Nel caso si richieda l’apertura di una sessione viene inviato il seguente messaggio:

```
byte      SSH_MSG_CHANNEL_OPEN
string    "session"
uint32    sender channel
uint32    initial window size
uint32    maximum packet size
```

Una volta che la sessione è stata settata, può essere richiesto al server l’esecuzione di un programma. Il programma può essere una shell, un programma applicativo oppure un servizio. Il seguente messaggio richiede la shell di default dell’utente(tipicamente definita nel file /etc/passwd):

```
byte      SSH_MSG_CHANNEL_REQUEST
uint32    recipient channel

string    "shell"
boolean   want reply
```

Se invece si richiede al server di iniziare l’esecuzione di un dato comando, il messaggio che viene inviato è il seguente:

```
byte      SSH_MSG_CHANNEL_REQUEST
uint32    recipient channel

string    "exec"
boolean   want reply
```

La stringa di comando può contenere anche il path del comando e quindi si rende necessario prendere precauzioni per prevenire l'esecuzione di comandi non autorizzati.
Poiché non considerate nel corso del progetto non si analizzeranno altri tipi di funzioni di SSH come l'X.11 forwarding o il TCP/IP port forwarding.

### 3.5.2. OpenSSH e Jsch

Definito il protocollo è stato necessario definire i SW per l'implementazione delle politiche stabilite dal protocollo. In questo progetto sono stati utilizzati:

- **OpenSSH**[19]: per la creazione del server SSH
- **Jsch:** per implementare il client SSH all'interno dell'applicazione Java .

**OpenSSH** è una suite di protocolli gratuita che fornisce cifratura per servizi di rete, come il login remoto o il trasferimento di file remoto, le cui principali caratteristiche sono:

- Essere un SW OpenSuorce (requisito necessario nel criterio di scelta): il codice è disponibile gratuitamente per chiunque via Internet. Questo incoraggia il riutilizzo e la verifica del codice. Licenza Gratuita
- È coperto da una licenza libera. Questo significa che OpenSSH essere usato per qualsiasi scopo, incluso quello commerciale. Tutte le componenti soggette a brevetto, sono state rimosse dal codice sorgente. Tutte le componenti soggette a restrizioni sulla licenza o brevetto, sono state scelte da librerie esterne (ad esempio OpenSSL).
- Utilizzo di algoritmi di crittografia forte: 3DES, Blowfish
- Utilizzo di autenticazione forte: protegge contro alcuni problemi di sicurezza, come IP spoofing, rotte false e DNS spoofing.
- Interoperabilità: Le versioni di OpenSSH anteriori alla 2.0 supportano i protocolli SSH 1.3 e SSH 1.5, permettendo la comunicazione con molte

versioni di UNIX, Windows ed altre implementazioni commerciali di SSH.

- Compressione Dati: La compressione dati prima della cifratura migliora le prestazioni su link lenti.

OpenSSH si è mostrato essere un tool ideale per essere usato con il sistema operativo dei nodi sensori (che si ricorda è una versione di base della distribuzione Linux Debian).

**Jsch** (acronimo di **J**ava **S**ecure **Ch**annel) è una libreria jar rilasciata dalla Jcraft[20]. Essa è una pura implementazione Java di SSH2 che permette la connessioni a server ssh e consente l'integrazione con qualsiasi programma java. È abbastanza diffuso facendo parte dei pacchetti forniti di base dall' ambiente di sviluppo Eclipse a partire dalla versione 3.0. Inoltre è lasciato sotto licenza di tipo BSD ed è ben documentato poiché largamente usato in ambito di programmazione Java.

## 3.6. Il Web Server

Esaurita la definizione dei componenti che costituiscono il Network Wrapper; rimane da definire quale web server utilizzare per la visualizzazione delle pagine web per la visualizzazione dei dati di funzionamento del sistema.

La scelta è ricaduta sin da subito su Apache poiché esso è il più diffuso sul mercato dei web server.
Di questo è stato scelto il sottoprodotto Apache Tomcat: un web container open source che implementa le specifiche JSP e Servlet di Sun Microsystems, fornendo una piattaforma per l'esecuzione di applicazioni Web sviluppate nel linguaggio Java. La sua distribuzione standard include anche le funzionalità di web server tradizionale, che corrispondono al prodotto Apache.
In passato, Tomcat era gestito nel contesto del Jakarta Project, ed era pertanto identificato con il nome di Jakarta Tomcat; attualmente è oggetto di un progetto indipendente.
Tomcat è rilasciato sotto la Apache license, ed è scritto interamente in Java; può quindi essere eseguito su qualsiasi architettura su cui sia installata una JVM.

Le pagine web sono state scritte in linguaggio JSP, così da ottenere un prodotto finale realizzato con sola tecnologia Java.

Nei capitoli successivi si viene analizzata la struttura e il funzionamento del componente principale del progetto realizzato, ovvero il Network Wrapper

# CAP 4. Progetto ed implementazione di Solar Data Manager

## 4.1. Il Network Wrapper

### 4.1.1. Network Wrapper: un sistema basato su query

Nel seguente capitolo di analizzerà l'architettura e il funzionamento del modello di sistema sviluppato per l'interrogazione di una rete di sensori .
Il modello parte dal presupposto della semplicità dal punto di vista dell'utente, ovvero la possibilità di interrogare l'intera rete eseguendo una semplice query.
La query è pertanto l'elemento fondamentale per l'interrogazione della rete; E' doveroso puntualizzare che in futuro la definizione delle query sarà un processo trasparente all'utente finale, che infatti disporrà della sola interfaccia grafica per interrogare il sistema.
L'utilizzo di un sistema di querying SQL-like (cioè basato sullo standard SQL) rende il sistema potenzialmente in grado di gestire i flussi di dati provenienti da sorgenti eterogenee svincolandosi così dalle piattaforme hardware utilizzate, poiché l'obbiettivo primario che ci si è posti sin dal momento della progettazione è stata la più completa flessibilità del sistema.

Il sistema realizzato è costituito da 2 blocchi principali:

- **modulo centralizzato: Query Wrapper Module**
- **modulo remoto: Remote Module**

Il primo è unico e si deve occupare della creazione delle query per i moduli remoti e l'elaborazione dei risultati ottenuti per permetterne una visualizzazione user-friendly. Il Query Wrapper Module (QWM) risiede sul Server centrale e pertanto

ad esso sono affidate le operazioni più onerose come la creazione/eliminazione di tabelle, il controllo di coerenza e l'aggregazione dei dati, l'attivazione di trigger/segnalazioni al sistema.
Il secondo invece è costituto dai diversi nodi distribuiti sulla che ricevono le informazioni direttamente dai sensori a cui sono collegati: esso si occupa di ricevere ed eseguire le query inviate dal Query Wrapper Module.

La figura 11 mostra lo schema a blocchi del sistema nel suo complesso:

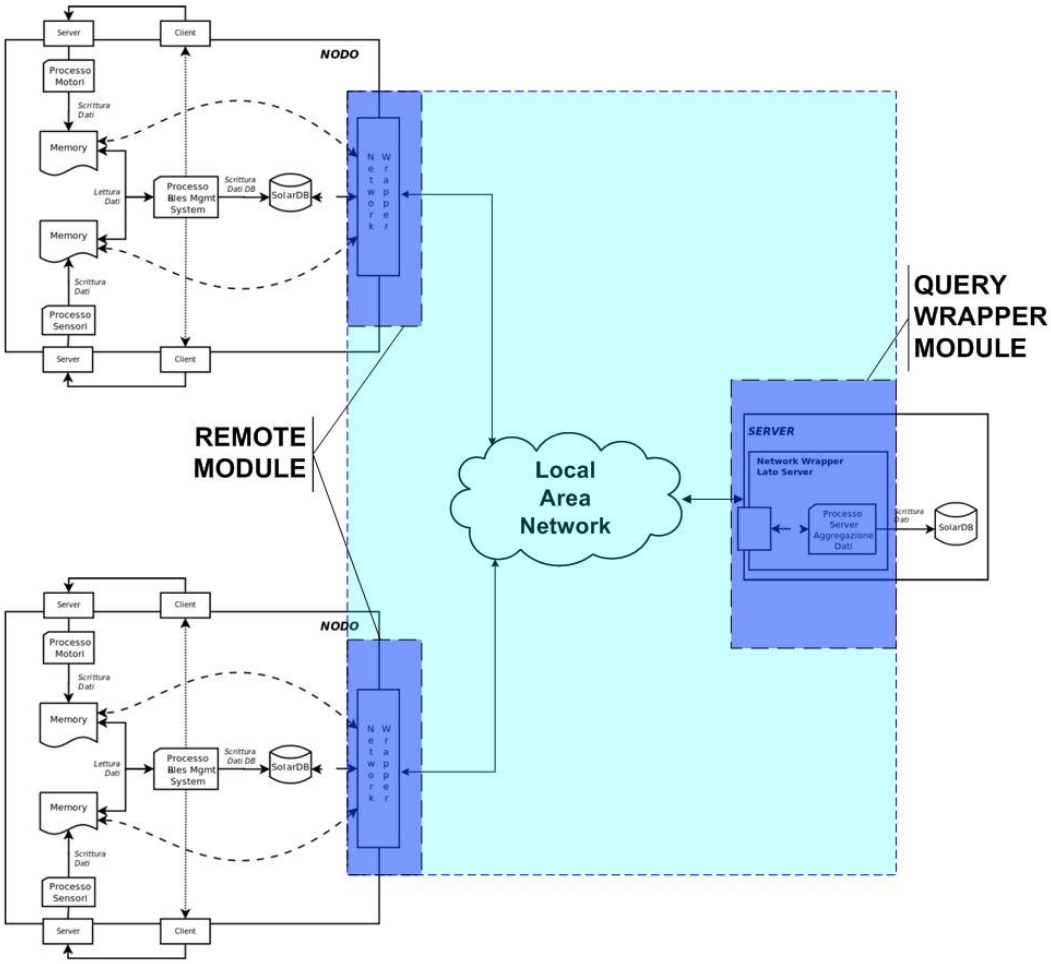



**Figura 11 - Sistema complessivo**

### 4.1.2. Network Wrapper: funzionamento e struttura

Il network wrapper come già spiegato è il cuore del progetto realizzato.
Esso realizza l'interazione tra il Server Centrale e i nodi cui è composta la rete di sensori, al fine di consentire l'invio dei dati presenti sui database SQLite dei nodi al database MySQL che risiede sul server, ovvero eseguendo un set di operazioni che hanno come risultato finale l'invio e l'esecuzione degli statement visti nel capitolo precedente.

La figura 12 mostra le fasi di cui si compone il wrapping dei dati:


**Figura 12 - schema delle fasi di wrapping**

Lo schema evidenzia come il trasferimento dei dati dalle tabelle del Database del nodo alla relativa tabella del Database centrale non è diretto, poiché vi è un passaggio intermedio di inserimento dei dati in tabelle temporanee che vengono create e distrutte ad ogni ciclo di interrogazione dei nodi periferici. Il dato finale inserito nelle tabelle (persistenti) del server centrale non proviene direttamente dal nodo ma dalla tabella temporanea, che ha sua volta contiene al suo interno dati presenti nel nodo.
L'inserimento di questa fase è stata fatta in funzione della ricerca di una maggiore robustezza del sistema: infatti senza l'utilizzo di tabelle temporanee si sarebbe potuta creare una situazione di stallo dovuta ad eventuali malfunzionamenti del nodo-periferico in fase di scrittura del dato nella tabella (persistente) del server. Con questa soluzione il problema non viene a verificarsi poiché nel peggiore dei

casi il problema di accesso alle tabelle sarebbe circoscritto alla singola tabella temporanea dove il processo scrittore del nodo stava operando al momento del malfunzionamento.

Le prossime figure illustrano graficamente come il Network Wrapper in un singolo ciclo vada ad agire sui Database che compongono il sistema.

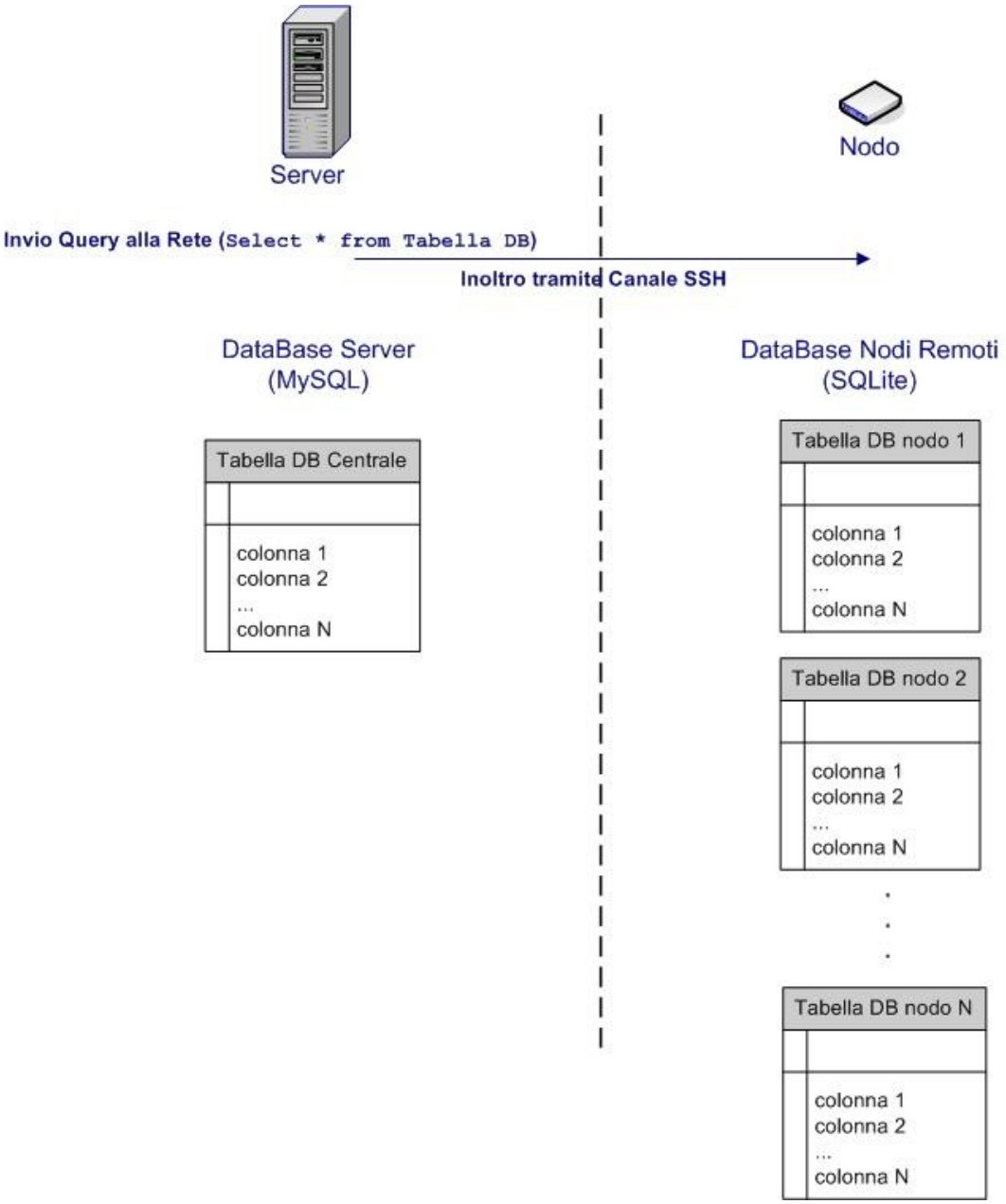



**Figura 13 - Stato iniziale delle Tabelle del sistema**

La prima figura mostra lo stato iniziale del sistema in forma esemplificata: infatti per esigenze di chiarezza viene mostrata una sola tabella del database centrale e una sola tabella per database degli N-nodi remoti.A sinistra viene rappresentato il server e a destra i le tabelle dei nodi remoti.

Nella fase iniziale sono presenti solo tabelle persistenti.

Tramite i canali SSH (uno per nodo) viene inoltrata la richiesta di informazioni, ai nodi remoti; successivamente vengono eseguiti gli statement per l'eliminazione dei dati obsoleti dalle tabelle locali

Come nella fase precedente non avvengono modifiche strutturali ai database che compongono i sistema.

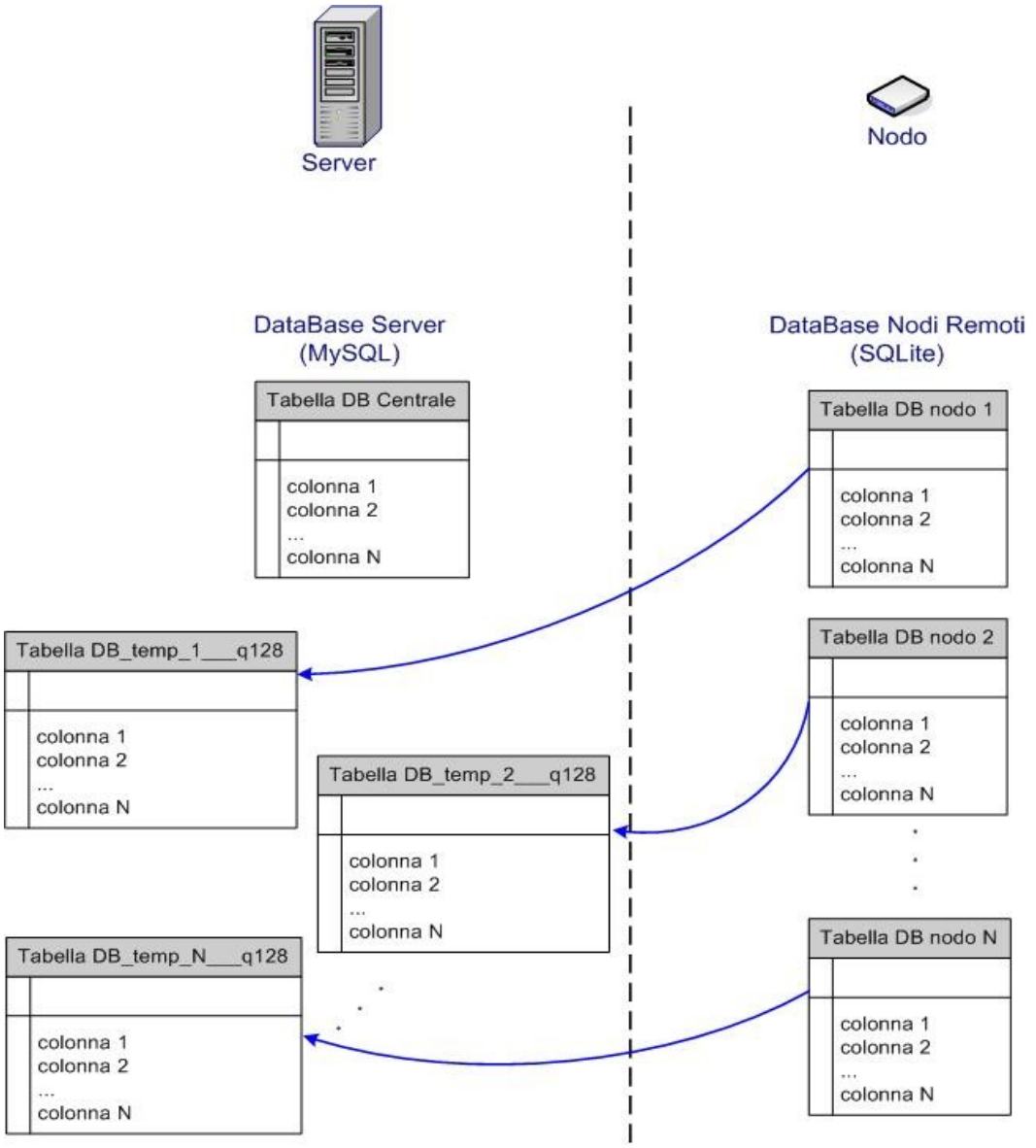



**Figura 14 - Trasferimento dati a tabelle temporanee**

In questa fase le tabelle locali hanno solo dati "nuovi" cioè non inviati nei precedenti cicli di interrogazione. Quindi si procede con la creazione sul database centrale delle tabelle temporanee (un per nodo per ogni query globale), l'interrogazione delle tabelle locali e l'invio dei dati ottenuti alle tabelle temporanee appena create.

In questa fase ho pertanto una modifica della struttura del database centrale, come si può notare dalla figura 14.

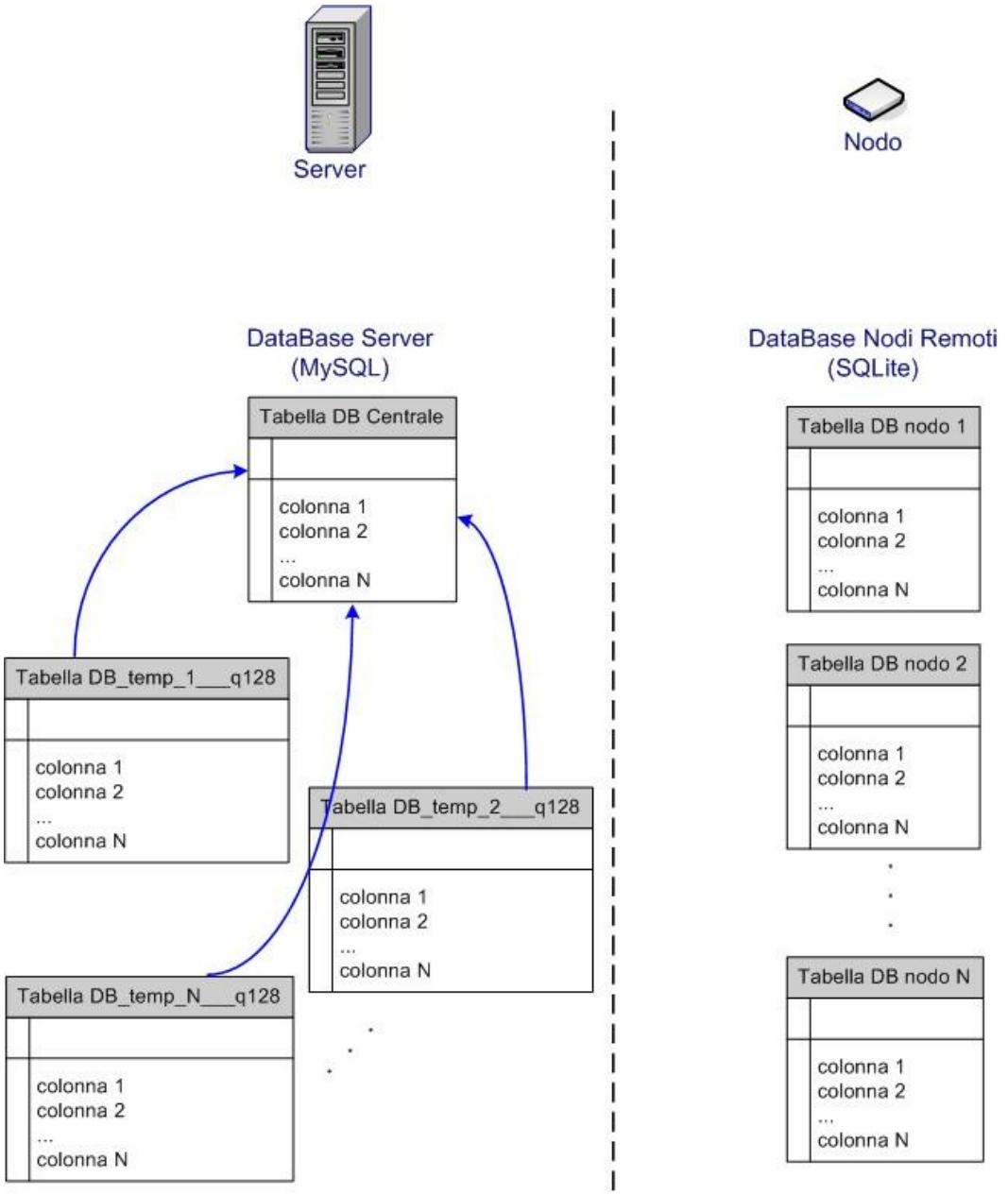



**Figura 15 - Trasferimento dati a tabella persistente**

Terminato il processo di popolamento  delle tabelle temporanee, si concludono le operazioni sui database remoti.

I dati contenuti nelle tabelle temporanee vengono trasferiti (secondo le opportune politiche) alla relativa tabella persistente

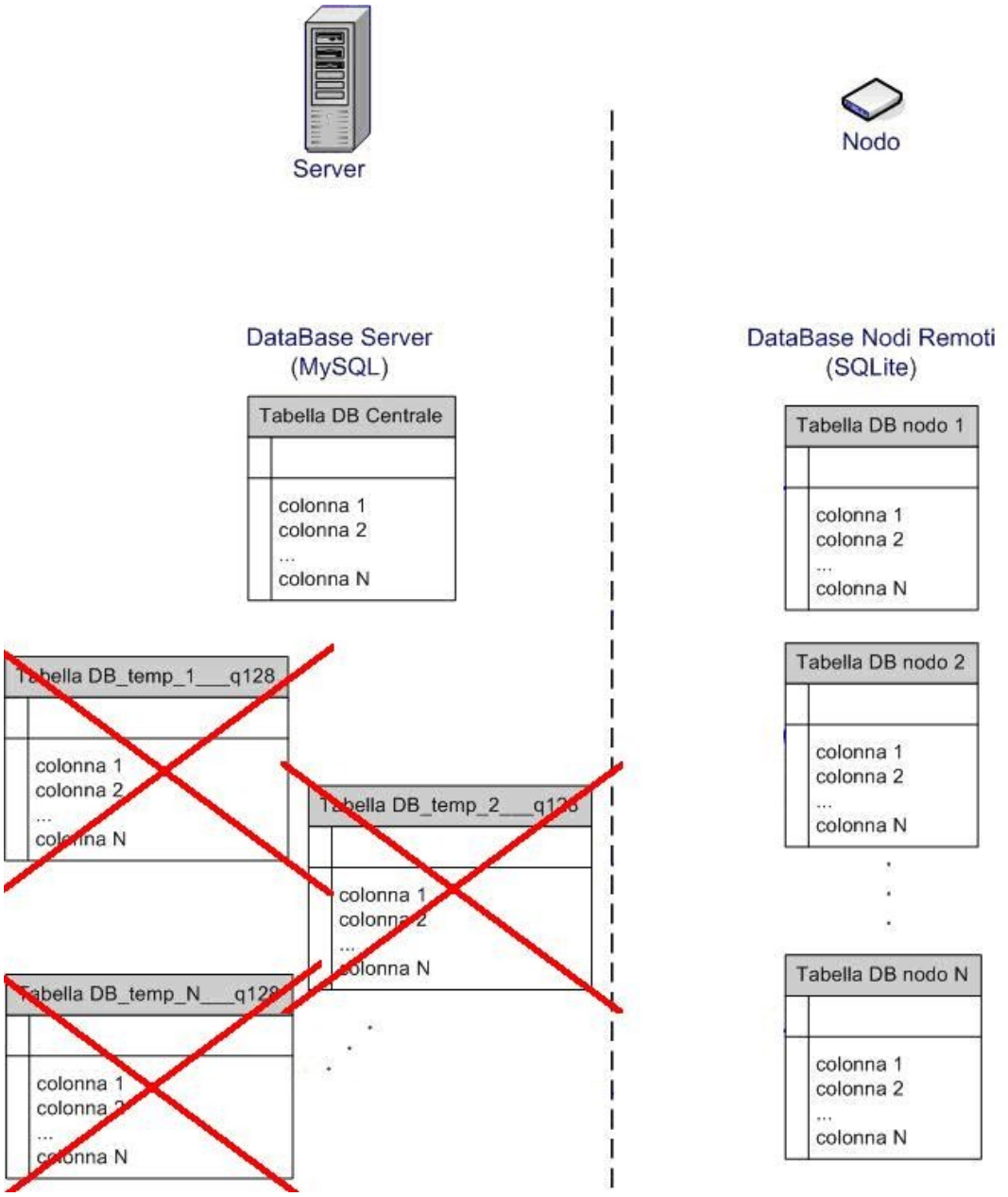



**Figura 16 - Eliminazione tabelle temporanee**

Infine vengono eliminate le tabelle temporanee, così facendo ho ripristinato la struttura iniziale del sistema.

Con queste operazioni si conclude il funzionamento del Network Wrapper

Nei paragrafi successivi si analizzano i processi che realizzano il network wrapper.

### 4.1.3. Query Wrapper Module: interrogazione della rete

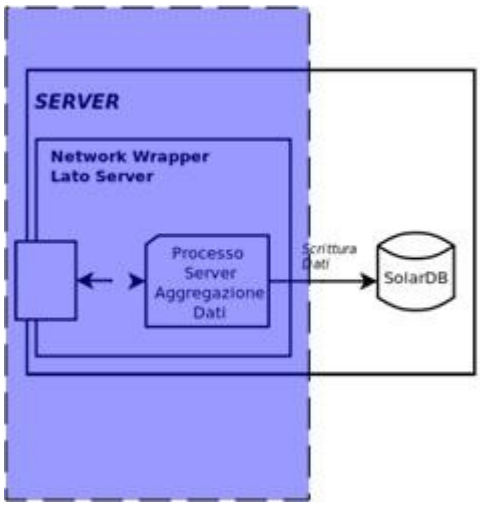



**Figura 17 – Schema del Query Wrapper Module**

L'utente interagisce con il sistema tramite interfaccia grafica presente sul server i cui contenuti vengono periodicamente aggiornati dal sistema di query automatizzate. Quest'ultime sono query che permettono di interrogare i singoli nodi periferici ed ottenere da essi sia dei dati aggregati che dei dati istantanei.

Il Query Wrapper Module (QWM) è il modulo che provvede a inviare le query ai nodi periferici e a gestirne i dati ottenuti dall'esecuzione di quest'ultime.

Il Query Wrapper Module è configurabile e grazie ad esso è possibile definire e personalizzare le query da iniettare alla rete indicando, oltre al dato che si vuole selezionare, la tabella del Server centrale dove si vuole inserire il (gli) risultato(i) della query e con quale periodicità si vogliono interrogare i nodi remoti.

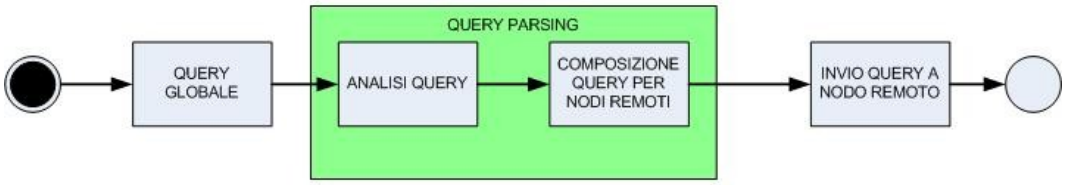



**Figura 18 - Query Parsing - Schema a blocchi**

Inserita la query ed impostato l'intervallo di interrogazione e definita la tabella di destinazione, ovvero 3 parametri distinti impostabili dall'utente, la query viene

elaborata (figura 18) e ne viene eseguito il Query Parsing: viene controllata la struttura della query immessa, chiamata anche query globale, per poi tradurla in una nuova query che viene inviata ai nodi, chiamata anche query remota. Questa operazione si divide in 2 sottoperazioni uno di analisi della query e uno per di composizione delle query che si occupa della creazione di query elaborata e specifica per i nodi.

Per capire meglio come si comporta il sistema, si riporta di seguito l'esempio del monitoraggio dei pannelli solari del sistema.
Ogni pannello invia i dati raccolti dal sensore al database locale SQLite il quale funge da magazzino di informazioni fino a quando esse non vengono "raccolte" dal processo wrapper che le invia al server centrale.
L'utente in genere può monitorare lo stato dell'impianto senza dovere fare uso dell'immissione diretta di query nel sistema ma utilizzare i dati che vengono già forniti nell'interfaccia utente. Come già accennato vi è però l'opportunità di configurare nuove query manualmente;
Poniamo il caso che si sia deciso di utilizzare questa seconda via: una volta definito il Polling Period (intervallo di interrogazione) e la tabella del Server dove inserire i risultati della query, è possibile inserire la query, ad es.

```
SELECT * FROM SENSORS_REV WHERE TIME < 13:05:00
```

Che corrisponde alla richiesta di reperimento delle informazioni dei dati contenuti nella tabella `SENSORS_REV` fino alle 13:05:05
La query viene passata all' Analizer Module che verifica la presenza di eventuali clausole WHERE. La query viene passata al Composer Module che prima cerca modifica il la sintassi della query, e successivamente divide la query in 2 parti: la `SENSOR_QUERY` (ovvero la query priva della clausola WHERE ) e la `QUERY_TAIL` contente la "coda" della query (quello che viene definito nella clausola WHERE)
La `SENSOR_QUERY` viene inviata ai nodi remoti e prendendo l'esempio corrente, il nodo dovrà soddisfare la query:

```
SELECT '*' FROM SENSORS_REV
```

Si noti la presenza di 2 apici singoli attorno al carattere *, necessari per evitare errori di interpretazione del comando da parte del RM.

### 4.1.4. La Query Globale

La Query Globale come è stato visto nel capitolo precedente rispetta la sintassi SQL, e necessita però di essere configurata.

La configurazione delle query globali è impostabile dalla tabella `globalqueries`.

Questa tabella è composta dai seguenti campi:

- `Qry_id`
- `Qry_active`
- `Qry_text`
- `Qry_intervall`
- `Qry_table`
- `Qry_nodes`
- `Qry_descr`

**Qry_id:** codice univoco della query. Necessario per la creazione delle tabelle temporanee;

**Qry_active:** intero che indica se i parametri di configurazione della riga corrente sono attivi (valore 1) o non devono essere eseguiti (valore 0);

**Qry_text:** testo della query. Rispetta la sintassi SQL;

**Qry_intervall:** periodicità con il quale deve essere eseguita la query;

**Qry_table:** tabella del Server MySQL dove deve essere salvato il risultato della query;

**Qry_nodes:** nodi della rete che devono essere interrogati dal sistema (quasi sempre saranno tutti ma è possibile che alcune query debbano essere riferiti a solo una parte dei nodi della rete)

**Qry_descr:** descrizione della query in linguaggio naturale, più facilmente comprendibile all'utente

Pertanto una nuova query da inserire nel sistema deve avere la seguente sintassi:

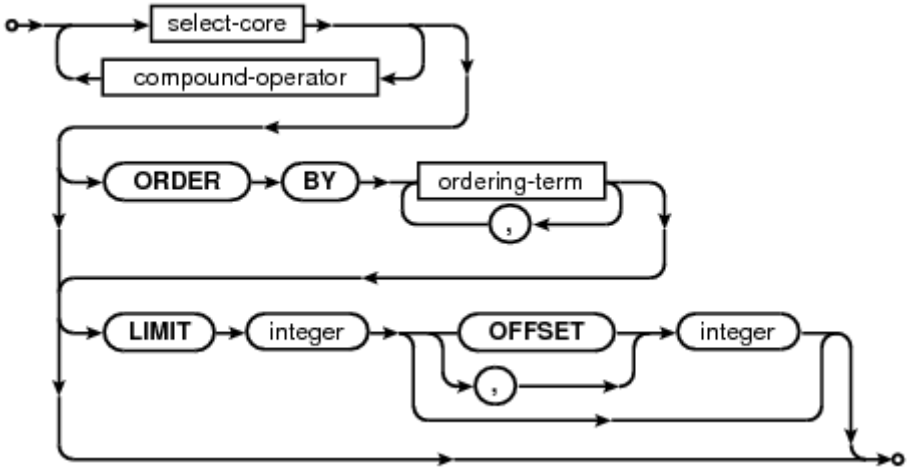
```
INSERT INTO GLOBALQUERIES VALUES (Qry_id ,(0|1),
Qry_text,Qry_intervall , Qry_table, Qry_nodes)
```

Dove `Qry_id` è un valore intero, `Qry_intervall` è un valore intero che esprime il tempo di attesa in secondi, nel campo Qry_active vale 0 (query inattiva) oppure 1 (query attiva), `Qry_table` è una semplice stringa, `Qry_nodes` invece è una sequenza di interi intervallati dal carattere '-': ad esempio se voglio interrogare solo i nodi 1,4,7,8,15 la sintassi relativa alla colonna `Qry_nodes` è: `1-4-7-8-15;` se si vogliono interrogare tutti i nodi dell'impianto si può usare il carattere "*" (asterisco) invece che indicare l'intera sequenza dei nodi che compongono l'impianto

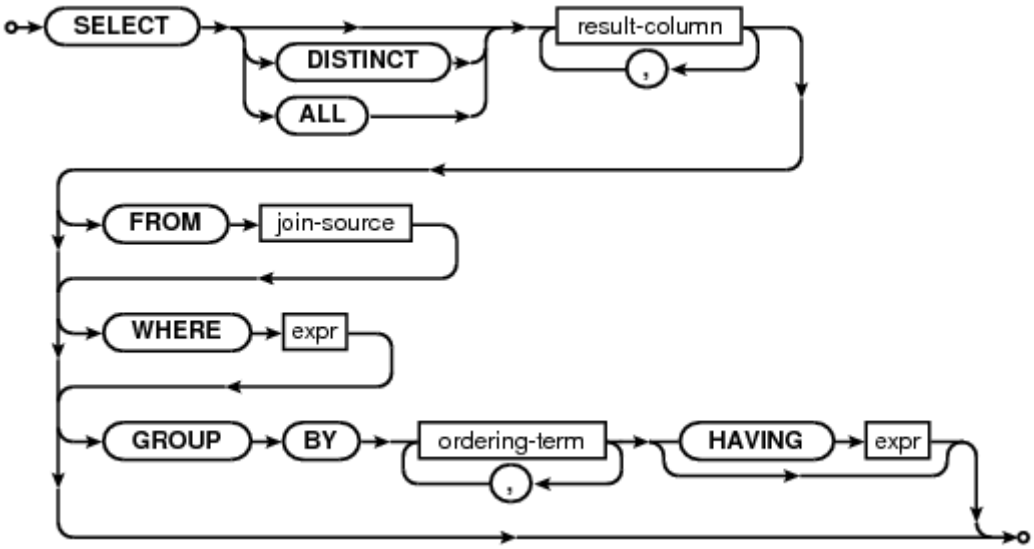
Qry_text come già detto rispetta le regole dello standard SQL, questo perché in fase di progetto si è dovuto optare per un sistema che fosse caratterizzato dalla massima flessibilità; quindi sono possibili tutte le operazioni disponibili per il DBMS SQLite, anche se principalmente le operazioni utilizzate sono delle SELECT e delle DELETE, ma non le INSERT che invece sono realizzate dai sensori poiché il sistema centrale si deve limitare alla raccolta dei dati prodotti dai sensori e alla cancellazione dei dati già raccolti nei database dei nodi.

Pertanto nelle figure seguenti mostra in forma grafica la struttura della query che caratterizza il contenuto della colonna Qry_text (si fa riferimento alla documentazione fornita da www.sqlite.org).

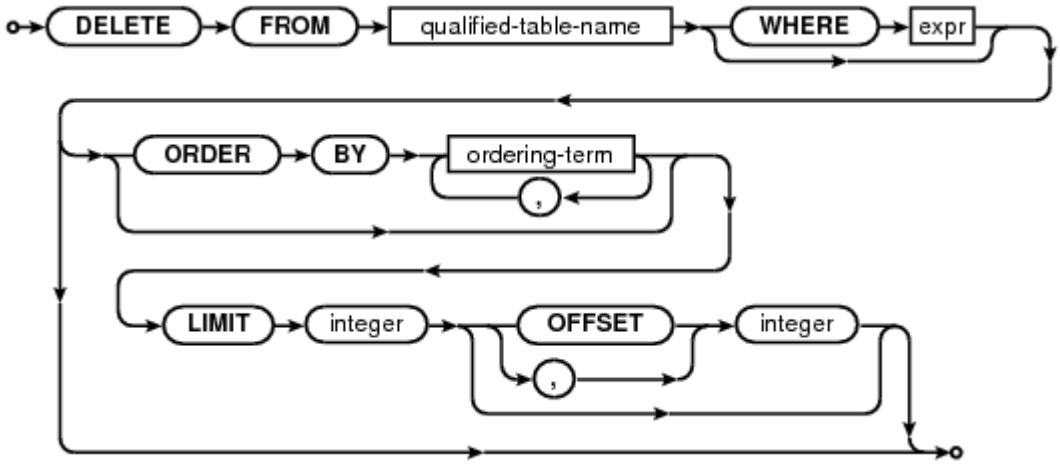
La SELECT




Il cui core deve rispettare la seguente sintassi



E la DELETE




Riprendendo l'esempio precedente la query string per l'inserimento di nuove query di sistema potrebbe essere la seguente:

```
INSERT INTO GLOBALQUERIES VALUES (128 , 1 , SELECT *
FROM SENSORS_REV WHERE REV_TIME > 13:05:00, 30 ,
SENSORS_REV , 1-4-7-8-15 , "get all Sun Tracker data
from 13:05:00" )
```

Questa query aggiunge una esecuzione di query che ha il seguente significato: attiva da subito la seguente query: ogni 30 secondi interroga i nodi 1,4,7,8,15 per cercare il contenuto della tabella SENSORS_REV successivo all'ora 13:05:00 presente su ciascuno sei nodi selezionati, e invia il risultato, caratterizzato da codice identificativo 128, alla tabella SENSORS_REV.

### 4.1.5. Remote Module: la query per la cancellazione dati

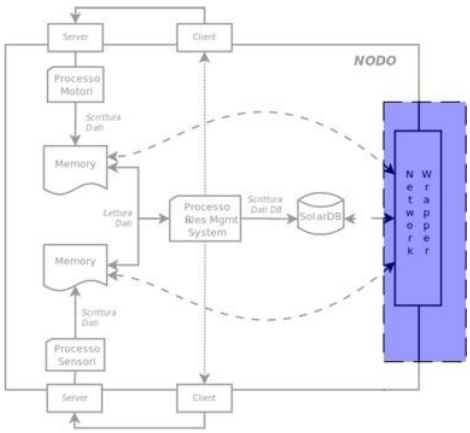



**Figura 19 – Blocco Nodo Remoto**

Il sistema creato è query based per cui così come l'interrogazione del nodo remoto avviene attraverso delle query globali anche il trasferimento dei dati al Server centrale avviene attraverso query che vengono generate in risposta all'input ricevuto dal Query Wrapper Module.

In questo paragrafo si discutono solo le query generate, mentre nel capitolo successivo analizzeranno i processi che portano alla generazione degli statement eseguiti da ognuno dei nodi.

In ordine cronologico il primo statement eseguito è una cancellazione dei dati contenuti nelle tabelle temporanee; questa operazione è necessaria poiché il sensore produce dati con frequenza elevata (anche 5 inserimenti al secondo) e come è facile prevedere senza una politica di cancellazione dei dati obsoleti, si avrebbe un rapido degrado delle prestazioni dei nodi causati dall'esaurimento della memoria disponibile.
La query eseguita ha una sintassi del tipo:

```
DELETE FROM [NOME TABELLA] WHERE TIMESTAMP > [VALORE_MIN] AND TIMESTAMP < [VALORE_MAX]
```

Dove `NOME TABELLA`, `VALORE_MIN` e `VALORE_MAX` sono dati forniti dal processo in esecuzione sul server (e che verranno analizzati nel capitolo successivo), ricavati dalle tabelle temporanee esistenti sul Database del Server.

### 4.1.6. Remote Module: le query per il trasferimento dati

Il secondo statement che viene eseguito è la creazione di specifiche tabelle temporanee sul Database Server e il popolamento delle stesse con i dati specificati nella query globale.

I 3 statement di update(), 2 necessari per la creazione della tabella e un terzo per l'inserimento dati, sono:

- `DROP TABLE IF EXIST [NOME TABELLA TEMPORANEA]`
- `CREATE TABLE [NOME TABELLA TEMPORANEA] ([STRUTTURA DATI])`
- `INSERT INTO [NOME TABELLA TEMPORANEA] VALUES ([VALORI])`

Dove `NOME TABELLA TEMPORANEA` è costruito così:

```
[NOME DELLA TABELLA] + "_temp_" +[NUMERO NODO] + "___q" +[QUERY_ID]
```

Con NOME DELLA TABELLA che sarà dato dal nome di tabella specificato nella QRY_TEXT specificato nella QUERY GLOBALE numero nodo è il numero intero che identica l'i-esimo nodo e QUERY_ID.
Prendendo l'esempio visto nel capitolo precedente il nome di una delle tabella temporanee sarebbe: SENSORS_REV_TEMP_1___Q128.

La struttura dati della tabella creata è identica a quella della tabella del nodo da cui vengono prelevati i dati, a meno delle chiavi primarie.
Nella versione attualmente implementata, si suppone chi immette nuove query globale conosca il database e le tabelle del sistema.

### 4.1.7. Network Wrapper: Class Diagram

Le figure presentate in questo capitolo illustrano il diagramma delle classi che compongono il network wrapper. Esse sono state inserite per dare maggiore completezza alla documentazione del progetto realizzato.

#### 4.1.7.1. Class Diagram: Query Wrapper Module

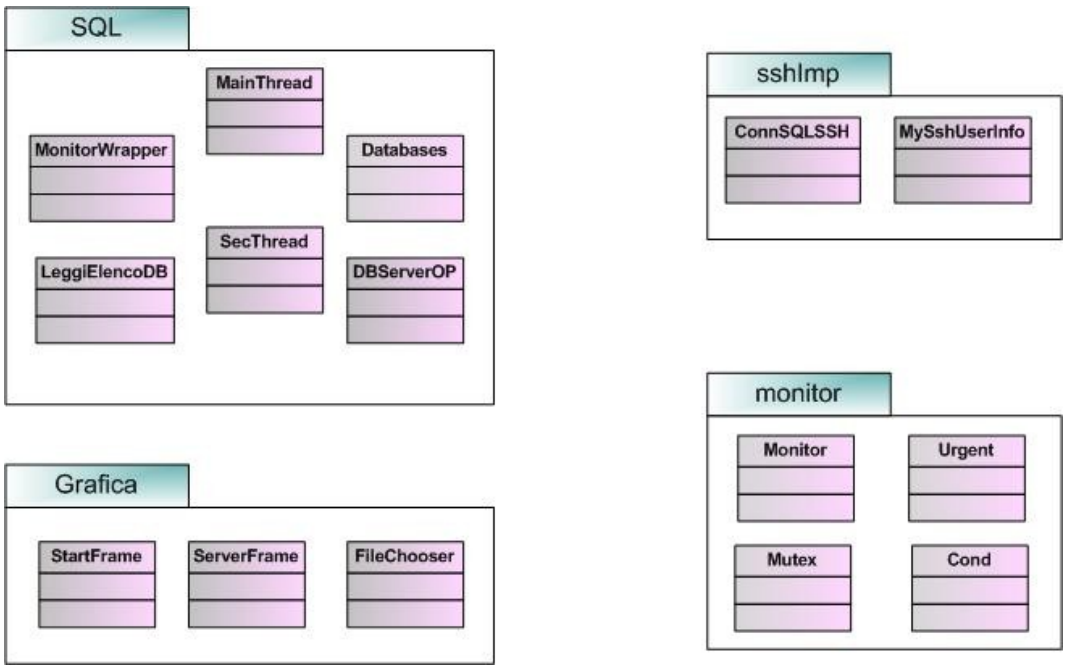



**Figura 20 - Query Wrapper Module – Class Diagramm – base**

Il Query Wrapper Module, cioè il modulo del Network wrapper che risiede nel server, è composto da 15 classi suddivise in 4 package:

- **SshImp:** che contiene le classi necessarie alla gestione dei canali Secure Shell;

- **Monitor:** che contiene le classi per la gestione della sincronizzazione tra processi
- **Grafica:** package che si è rivelato necessario in fase di testing del sistema per la generazione di interfacce utente alternative per la valutazione del funzionamento del sistema
- **SQL:** Package che contiene le classi che implementano le funzioni principali del Network wrapper, principalmente quelle relative a connessione e interrogazione dei database, lettura e scrittura dati dei database.

Per esigenze di chiarezza interpretativa a figura 20 mostra solo i package i e classi in esse contenute.

La figura 21 mostra inoltre le relazioni che sussistono tra le classi del Query Wrapper Module .

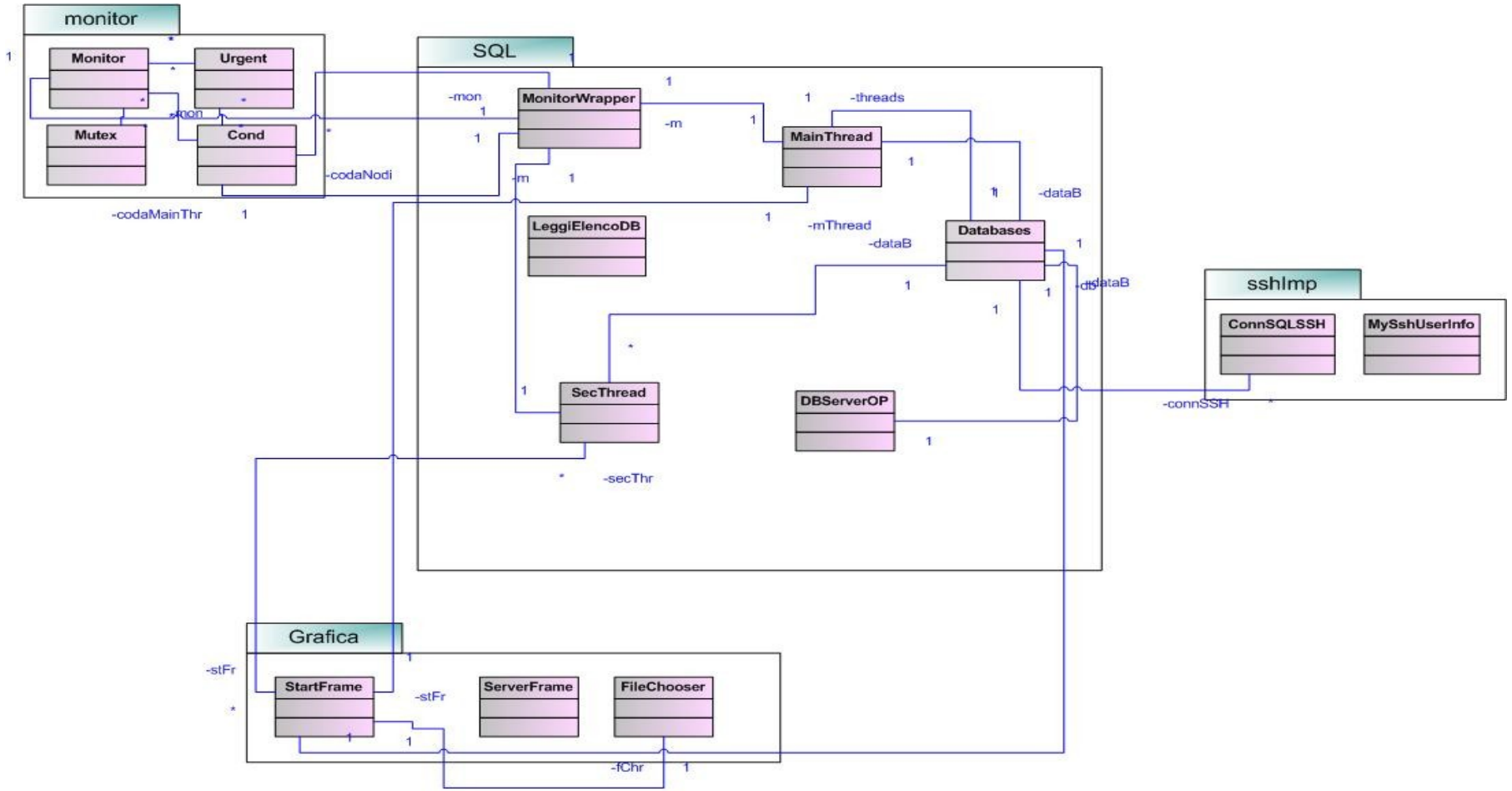



**Figura 21 – Query Wrapper Module - Class Diagram Completo**

Il package sshImp si compone di 2 classi

- **ConnSQLSSH**
- **MySshUserInfo**

ConnSQLSSH è la classe principale tramite la quale è possibile creare e gestire i client Secure Shell, necessari per la comunicazione tra server centrale e nodi remoti.

MySshUserInfo è la classe di supporto che gestisce le informazioni del host della connessione SSH.

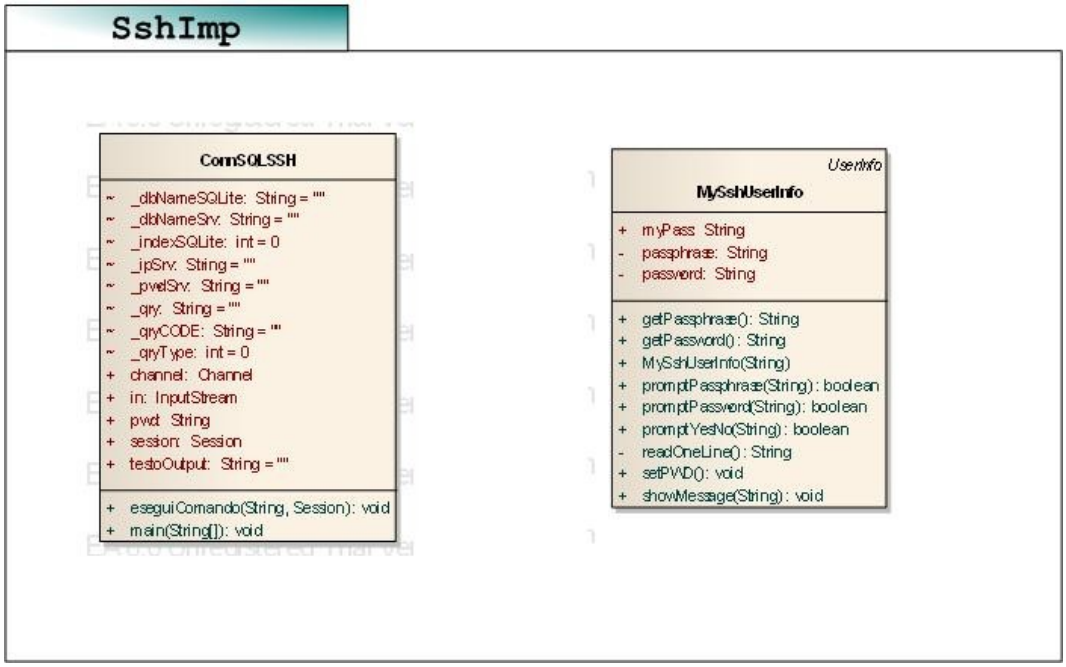



**Figura 22 - QWM - Class Diagram - Package sshImp**

Il package monitor di compone di 4 classi:

- **Monitor**
- **Urgent**
- **Mutex**
- **Cond**

Le caratteristiche del monitor realizzato per questo progetto verrà descritta nel paragrafo 4.1.8;

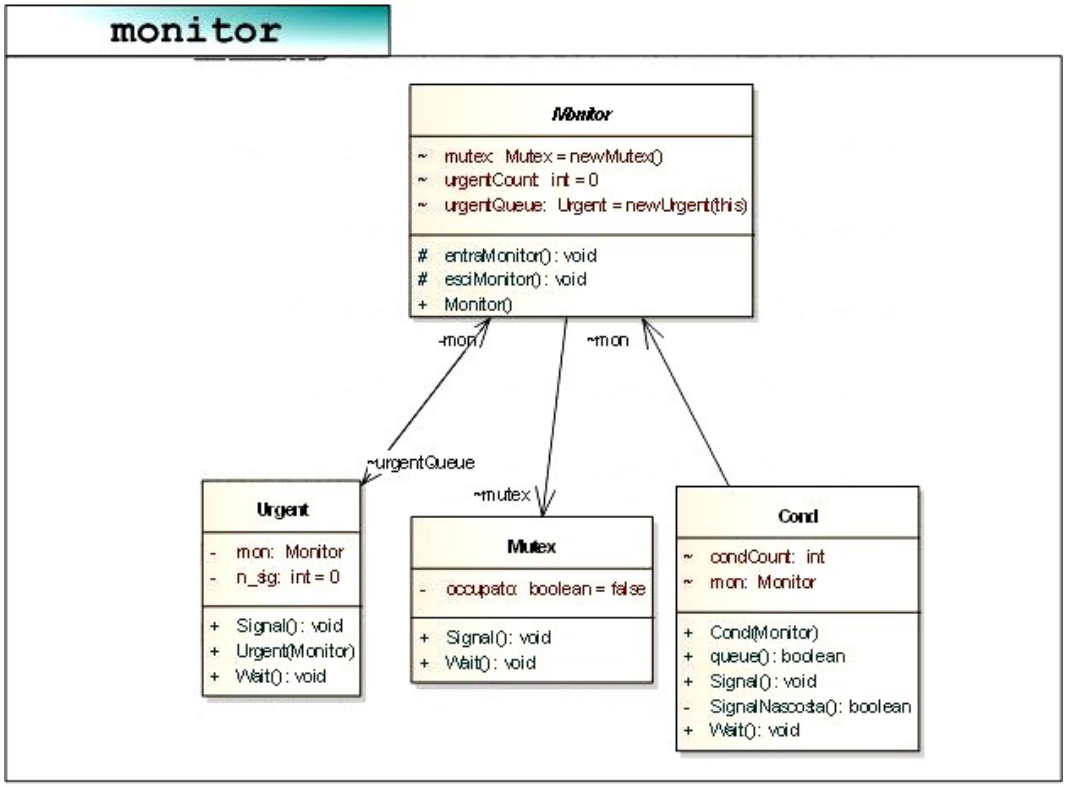



**Figura 23 - QWM - Class Diagram - Package monitor**

Nel package Grafica si trovano le classi per la visualizzazione a schemo di frame pensati per visualizzare il funzionamento del Network wrapper. Le 3 classi rilevanti che sono rimaste anche nella versione finale per SW realizzato sono:

- FileChooser: usato per selezionare il file di configurazione del sistema
- ServerFrame: frame per visualizzare le tabelle del database centrale
- StartFrame: frame principale del programma

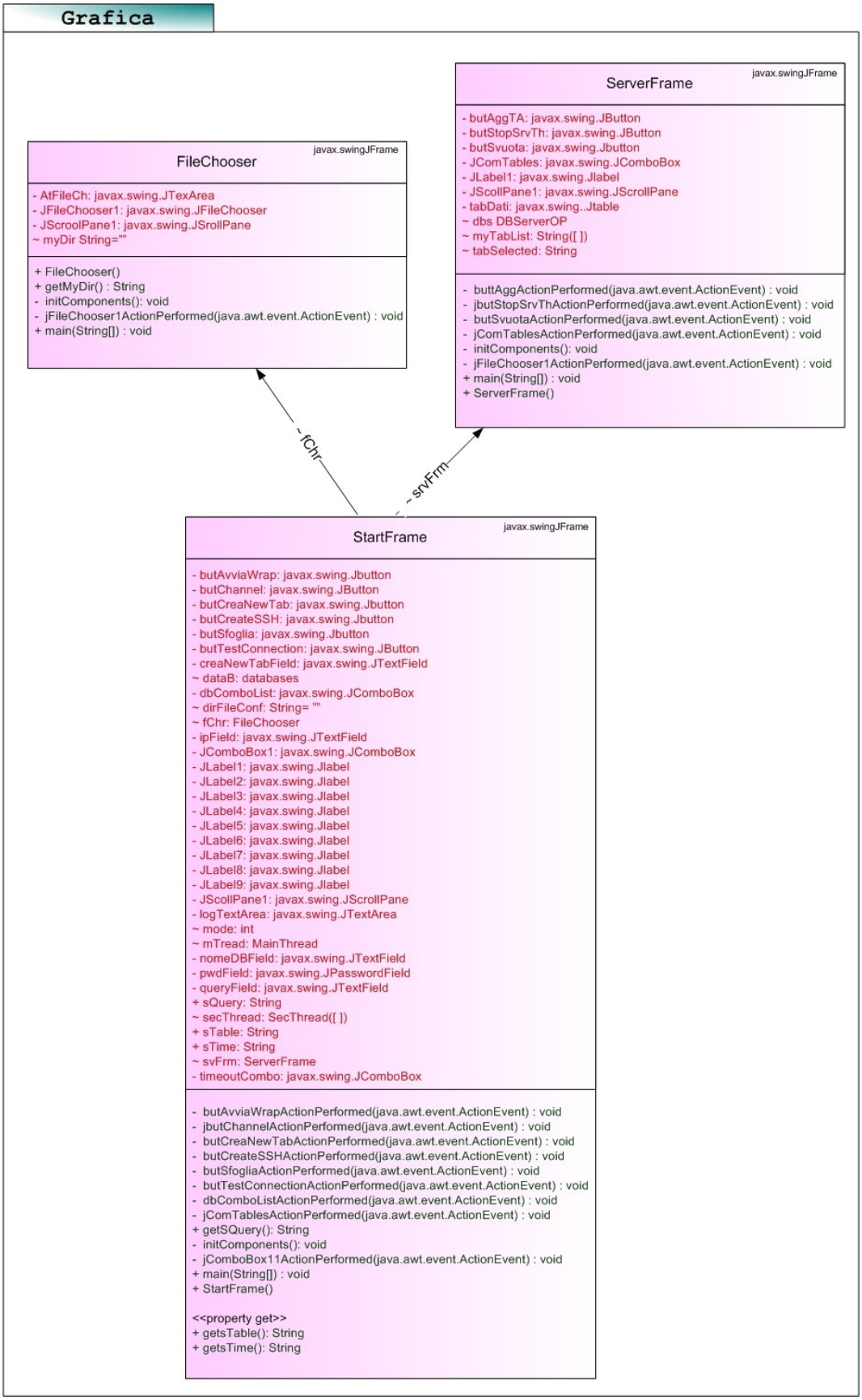



**Figura 24 - QWM - Class Diagram - Package Grafica**

Nel package SQL risiedono le classi principali del Network Wrapper tra cui MainThread SecThread e MonitorWrapper, classi che verranno analizzate nel dettaglio nei capitoli successivi.

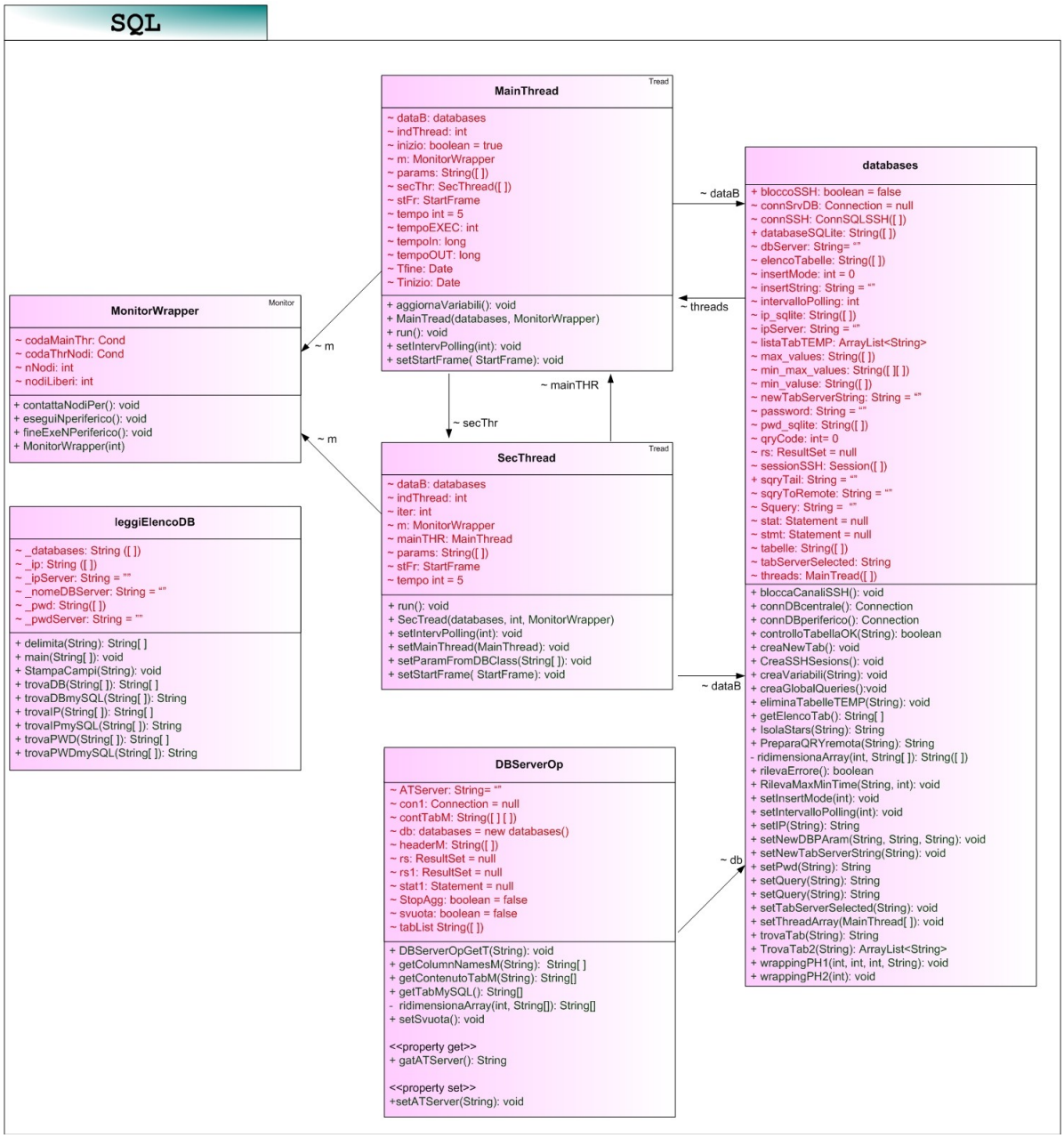



**Figura 25 - QWM - Class Diagram - Package SQL**

`Leggi elenco DB` è una classe che serve a ricavare le informazioni relative alla configurazione del sistema a partire da un documento di configurazione;

questa funzione può essere sostituita dalla lettura dei dati dal database nella tabella su configurazione.

`DBServerOP` implementa le funzioni per la visualizzazione dei contenuti del server del database Centrale.

`Databases` è la classe fondamentale per il funzionamento del Network wrapper: essa contiene le funzioni principali del programma per eseguire il wrapping dei dati tra cui la connessione ai database remoti e quello centrale, la politica di gestione delle query globali per la generazione delle query locali, la politica di aggregazione dei dati contenuti nelle tabelle del server.

Le funzioni contenute da questa classe sono sfruttate sia dal MainThread che dal SecThread.

### 4.1.7.2. Class Diagram: Remote Module

Il class diagram del Remote Module è composto da una singola classe (MySQLite.class) tramite il quale viene esguito il comando inviato tramite il canale SSH dal processo SecThread.
Il Remote Module, essendo la parte del Network Wrapper risiedente nei nodi remoti, è di modeste dimensioni (il file .class non arriva a 50 Kbyte di spazio occupato sulla memoria di massa), rispettando così le esigenze di leggerezza, imposte in fase di progetto.

L'immagine sottostante illustra il Class Diagram del Remote Module.


**Figura 26 - Remote Module - Class Diagram**

Nei capitoli successivi vengono analizzati il funzionamento con particolare attenzione alla struttura a blocchi dei 2 thread che compongono il NW (MainThread e SecThread) .

### 4.1.8. Network Wrapper: i thread per il trasferimento dati

Il processo di wrapping, caratterizzato dalle 3 fasi descritte nel paragrafo 4.1.2, viene realizzato da 2 thread differenti, uno in esecuzione sul server centrale e un altro in esecuzione su ciascuno dei nodi. Essi rispettivamente sono:

- **MainThread();**
- **SecThread();**

questi 2 tipi di thread mutuamente esclusivi, sono istanziati all'avvio e permettono di eseguire un ciclo di interrogazione per ogni Qry_intervall (l'intervallo di interrogazione specificato nella query globale).

La mutua esclusività è garantita dal monitor di sistema implementato dalla classe `MonitorWrapper.class` che estende la classe `Monitor` del package `monitor` presente nel software realizzato.
MonitorWrapper si ispira alla logica di Hoare[21] il cui schema è illustrato nella figura 27. Questa logica prevede la riattivazione immediata di un processo in attesa sulla variabile per cui la notifica è chiamata, mentre il processo che compie la notifica viene accodato alla coda di attesa della stessa, per essere riattivato quando il processo risvegliato lascia il monitor o si mette in attesa su una variabile condizionale; questa soluzione è anche chiamata Signal Urgent in contrapposizione alle logiche "signal and return" e "signal and continue".

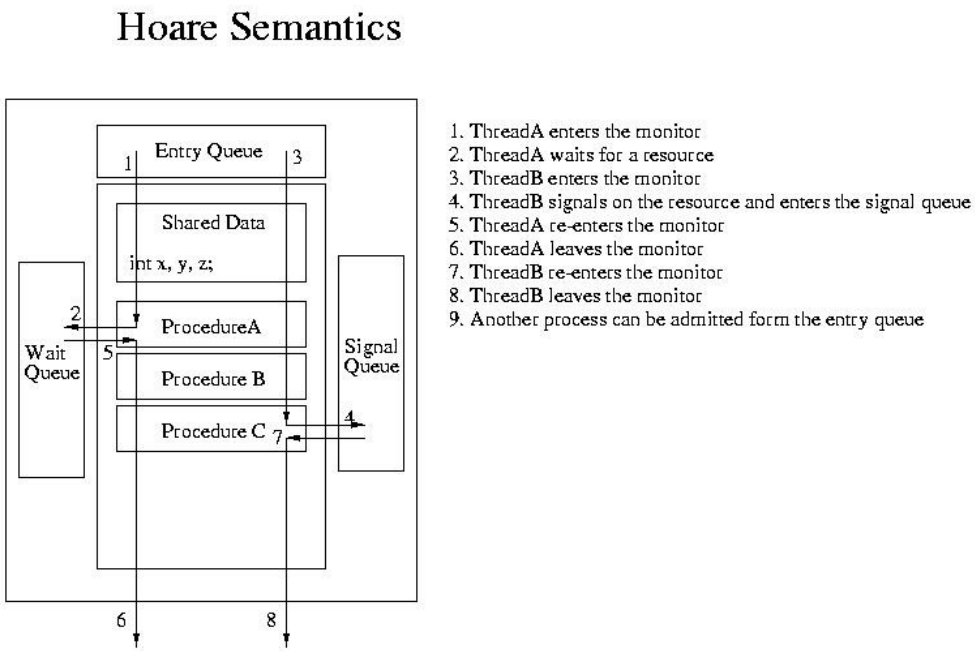



**Figura 27 - Logica di Hoare**

Nel caso SW oggetto di questa tesi il MonitorWrapper composto da 2 code (le variabili condition) e 3 metodi: le code sono differenti poiché una è "riservata" al MainThread e sulla seconda vengono accodati i SecThread; i metodi implementati invece sono:

- `contattaNodiPer()`: tramite il quale si gestisce il MainThread e vengono generate le signal per i SecThread
- `eseguiNPeriferico()`: tramite il quale viene acquisita la risorsa "nodo di rete"
- `fineExeNPeriferico()`: tramite il quale viene rilasciata la risorsa "nodo di rete" e in caso si verifichi la condizione di risveglio del MainThread, viene mandata la signal() sull' elemento della coda del MainThread.

Con questa semplice struttura è stato possibile sincronizzare i 2 tipi di processi del Network Wrapper, che nei paragrafi successivi saranno analizzati in dettaglio.

### 4.1.8.1. Network Wrapper: MainThread

Il MainThread è il thread che viene eseguito sul server centrale che realizza la funzione del Query Wrapper Module.
Ovvero:

1. traduzione della Query Globale nelle Query Locali che devono essere eseguite su ogni nodo
2. inoltro delle Query Locali ai nodi
3. Acquisizione dei dati presenti nelle tabelle temporanee
4. Eliminazione delle tabelle temporanee

Nella figura successiva viene mostrato lo schema a blocchi del funzionamento del MainThread:

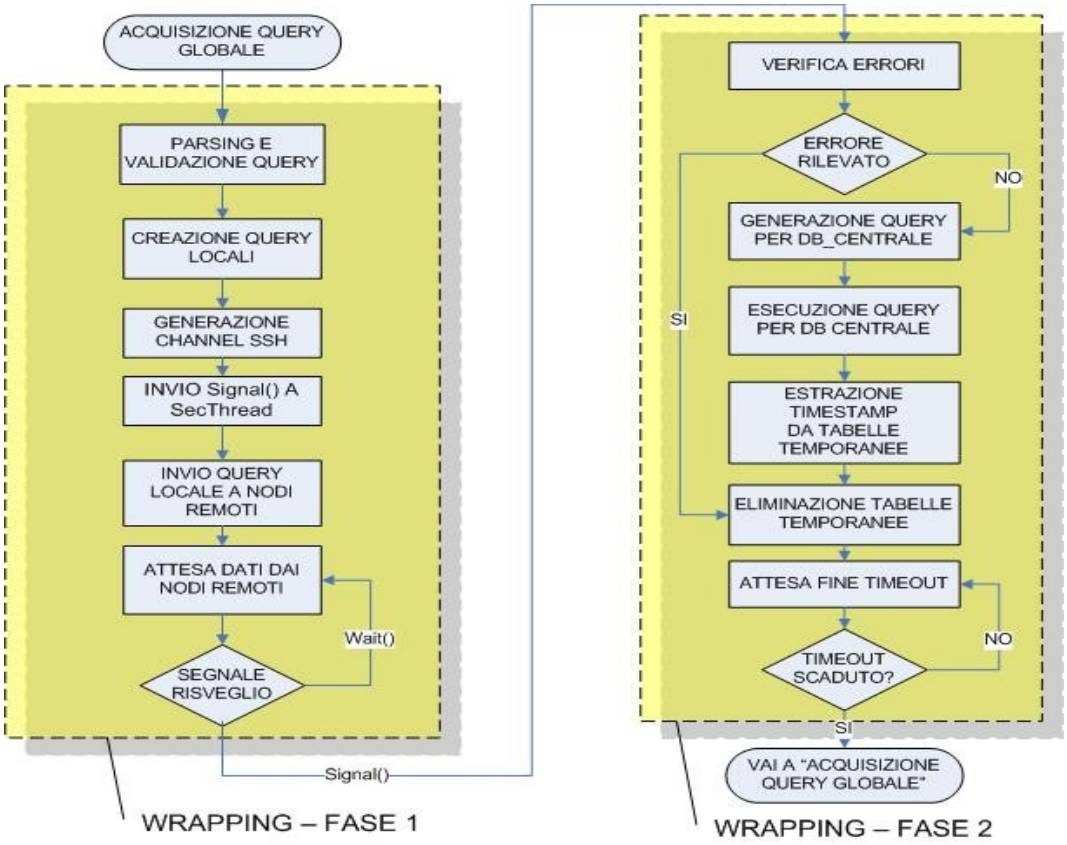



**Figura 28 - MainThread - Schema a Blocchi**

La figura 28 evidenzia come il MainThread si componga di 2 macroblocchi:

- **Wrapping – fase 1**
- **Wrapping – fase 2**

Nella prima fase viene elaborata la Query Globale e viene inoltrata ai nodi del sistema la richiesta di informazioni
Nella seconda fase invece vengono elaborati i dati "grezzi" ricevuti dai nodi.

Nel **Wrapping – fase 1** vengono svolte le seguenti operazioni:

*PARSING E VALIDAZIONE QUERY*: La generazione delle Query locali è già stata analizzata nel capito relativo alla gestione della Query Globale da parte del Query Wrapper Module. Pertanto l'output di questo passo è la Query Locale che deve essere inviata al nodo;
*CREAZIONE DELLE QUERY LOCALI:* una volta analizzata e modificata la query occorre aggiungere anche altri dati necessari per la corretta esecuzione delle interrogazioni del nodo remoto:

- Codice della query viene eseguita
- Indice del nodo su cui deve essere eseguita la query
- Parametri di connessione al Server centrale
- URL del database SQLite locale
- Eventuali MAXTIMESTAP e MINTIMESTAMP delle tabelle che devono essere svuotate
- Nome dell'eventuale tabella contente il Timestamp

Terminato questo blocco si ottengono tutte le informazioni di cui necessita il nodo sensore.
*GENERAZIONE CHANNEL SSH*: per contattare i nodi-sensore della rete occorre generare la connessione SSH; le sessioni (una per nodo) vengono generate in fase

di inizializzazione ed assieme a queste un canale fittizio di tipo “shell” necessario per avere una sessione permanente. Ad ogni ciclo di interrogazioni viene generato il relativo canale “exec” tramite il quale è possibile eseguire il comando di trasferimento dati sui nodi remoti. Questo channel rimane aperto finché l’InputStream non risulta vuoto cioè finché c’è passaggio di dati attraverso il canale. La prima implementazione del programma in questa fase prevedeva l’istanziazione dell’intero client SSH, ma la soluzione è stata quasi da subito scartata poiché la fase di creazione e di autenticazione della sessione ad ogni ciclo di richiesta dati, comportava un eccessivo spreco di risorse: la versione del programma con creazione dell’intero client SSH ha un tempo medio di esecuzione in condizioni ottimali di 7,8 secondi mentre la versione con la creazione dei soli Exec-Channel ha un tempo medio pari a 2,8 secondi riducendo così a quasi un terzo i tempi di esecuzione.

*INVIO Signal() A SecThread():* come accennato in precedenza SecThread() è istanziato su ogni nodo della rete, e posto in stato di Wait() in attesa della ricezione del segnale di risveglio,Signal() , da parte del thread in esecuzione sul Server centrale, operazione che avviene appunto in questo passo.

*INVIO QUERY A NODI REMOTI*: una volta terminate le operazioni descritte in precedenza si può procedere con l’invio dei dati ai nodi.

*ATTESA DEI DATI DAI NODI REMOTI*: il MainThread si pone in stato di Wait() in attesa che i nodi a cui è stato inviato il comando di recupero dei dati eseguano, eseguano le operazioni richieste.

In **Wrapper – fase 2** vengono svolte le seguenti operazioni:

*VERIFICA ERRORI:* la prima operazione che viene fatta al risveglio del MainThread è la verifica dell’output fornito dai nodi remoti: Infatti viene fatto un controllo sul contenuto dell’inputStream dell’ExecChannel() se questo contiene il codice d’errore allora viene considerato non valido il contenuto delle tabelle temporanee presenti sul database e si esegue il jump al passo *ELIMINAZIONE TABELLE TEMPORANEE*;

*GENERAZIONE QUERY PER DB CENTRALE*: in questa fase vengono creati gli statement necessari l'esecuzione del corretto "wrapping secondario": con questo termine si intende il trasferimento dei dati presenti nelle n-tabelle temporanee (dove n indica il numero di nodi a cui è stata inoltrata in precedenza la richiesta dei dati); esso sarà diretto ma dipenderà dalla struttura della query tail;

ESECUZIONE QUERY PER DB CENTRALE: una volta create le query per le tabelle temporanee, viene eseguito l'update delle tabelle persistenti del database centrale; esso avviene in maniera sequenziale per evitare di raggiungere il numero massimo di connessioni simultanee: Ad esempio se l'update avvenisse contattando contemporaneamente tutte le tabelle temporanee il database MySQL avesse un Engine capace di gestire al massimo 100 connessioni simultanee il sistema fosse impostato per eseguire 10 query globali basterebbero 11 nodi per eccedere il numero massimo di connessioni consentite (il numero di connessioni = n° nodi * n° di query globali attive, 11 * 10 = 110)

*ESTRAZIONE TIMESTAMP DA TABELLE TEMPORANEE*: terminato l'invio dei dati dal nodo al Server Centrale (seppur nella forma grezza contenuta nelle tabelle temporanee), questi dati diventano obsoleti: è necessario pertanto eliminare i dati dalle tabelle dei nodi. Questa operazione non può coinvolgere tutti i dati presenti nella tabella del nodo ma deve essere circoscritta a tutti ai soli dati inviati: alcune rilevazioni dati vengono eseguite ogni 0,2 secondi se venisse eseguito semplicemente la query: `DELETE FROM [TABELLA LOCALE DEL NODO]` una volta inviati i dati presenti in tabella nel momento della richiesta dati da parte del Server, di sicuro verrebbero cancellati anche dei dati "nuovi".

La soluzione consiste nel recuperare il timestamp delle righe inserite nelle tabelle temporanee per poi cancellare tutte le righe il cui timestamp sia compreso tra il valore min e il valore massimo rilevato. In questa fase il wrapper cerca per ogni tabella temporanea questi 2 valori che al ciclo di richiesta dati successivo verranno inseriti tra i dati per la query locale del passo CREAZIONE QUERY LOCALI.

*ELIMINAZIONE TABELLE TEMPORANEE*: è l'ultimo passo che prevede modifiche al contenuto del database centrale; si arriva a questo passo sia se il recupero dati remoti è andato a buon fine sia che vi siano stati problemi. In questa

fase vengono cercate dall'elenco delle tabelle temporanee presenti nel database quelle che hanno il Qry_code (l'identificativo della Query) corrispondente alla Query globale mandata in esecuzione, ed elimina queste tabelle. Il controllo del Qry_code mi consente infatti di poter svolgere più query globali contemporaneamente: senza questo controllo il sistema non potrebbe sapere a quale query globale si riferisce il contenuto della tabella temporanea presente sul database, inoltre se si sono verificati errori nel recupero dei dati dei sensori tutte e sole le tabelle temporanee che aggregate che comporrebbero il resultSet della query globale vengono cancellate.

*ATTESA DI FINE TIMEOUT*: A questo punto il trasferimento dati è da considerarsi completo per cui il thread si mette in stato di "sleeping" fino al raggiungimento della fine del timeout impostato nella query globale (campo Qry_intervall). Il tempo effettivo di sleeping ovviamente sarà pari a:

```
T.Sleep=(qry_intervall-T.Esecuzione Wrapping dati nodi)
SE T.Sleep > 0 ALLORA sleep (t.Sleep);
SE T.Sleep <= 0 ALLORA sleep(0);
```

Scaduto (l'eventuale) timeout il sistema esegue nuovamente il ciclo di raccolta dati, ritornando ad "*ACQUISIZIONE QUERY GLOBALE*"

### 4.1.8.2. Network Wrapper: SecThread

SecThread è il thread "secondario" cioè quello che viene eseguito su ogni nodo del sistema.

La figura 29 mostra lo schema a blocchi del SecThread:

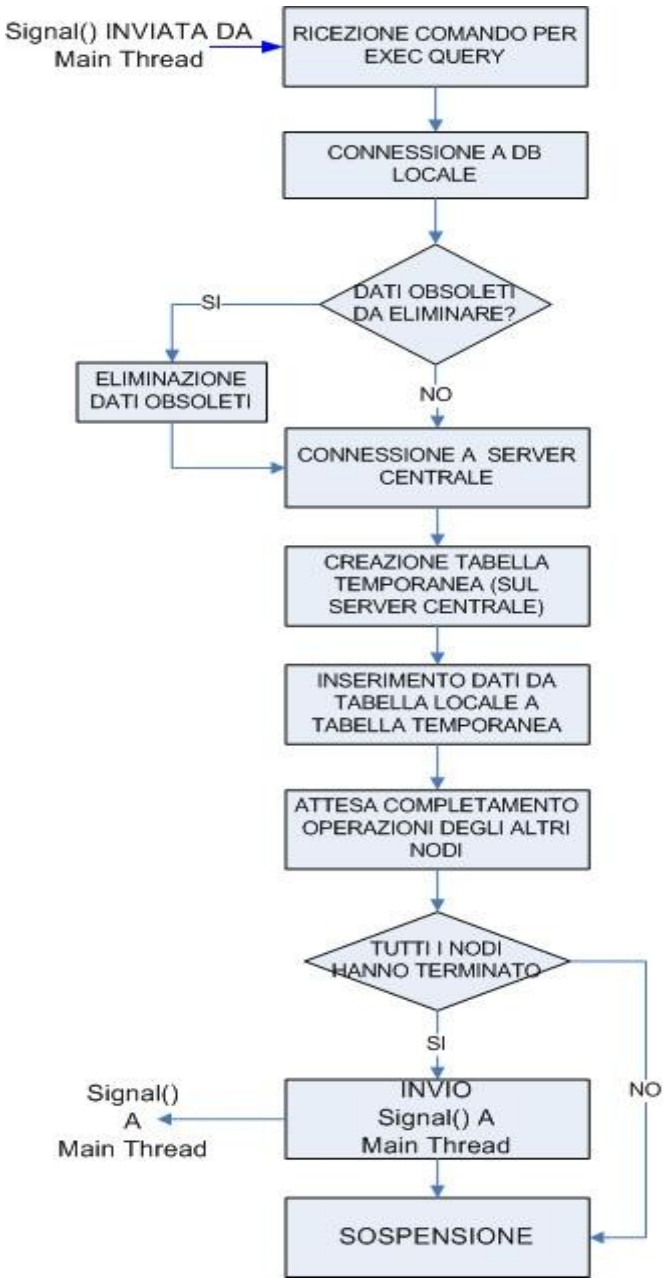



**Figura 29 - Schema a blocchi di SecThread**

Lo schema mette in evidenza come il thread per essere messo in esecuzione necessita di una Signal(), ovvero di un segnale di risveglio, da parte del MainThread: come nella struttura più tipica di un sistema di sincronizzazione gestito da monitor, il processo rimane in stato di "Wait" fino alla ricezione di una Signal(), una volta "risvegliato" esegue le operazioni richieste e solo dopo aver completate si metterà in sospensione, non prima di aver verificato che vi siano le condizioni di risveglio del MainThread, che nel frattempo è in sospensione.
*RICEZIONE COMANDO PER EXEC QUERY*: l'operazione immediatamente successiva al "risveglio" è la ricezione e l'esecuzione del comando ricevuto dal server; in questa fase vengono analizzati gli argomenti del comando di esecuzione della Classe Java che risiede sul nodo. Da questa devono essere estrapolati i dati creati dal MainThread al passo "*CREAZIONE QUERY LOCALI*" che sono sempre:

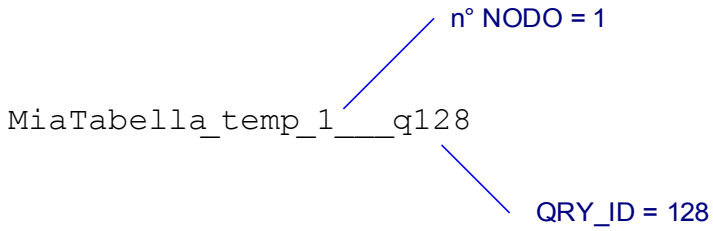
- Codice della query da eseguire
- Indice del nodo su cui deve essere eseguita la query
- Parametri di connessione al Server centrale
- URL del database SQLite locale
- Eventuali MAXTIMESTAP e MINTIMESTAMP delle tabelle che devono essere svuotate
- Nome della tabella contente il TimeStamp
- Query Locale

Ovviamente questa operazione comporta la verifica della coerenza dei dati passati, e l'eventuale generazione del codice di errore in caso di rilevazione anomalie.
*CONNESSIONE A DB LOCALE:* connessione al database locale SQLite la cui stringa di connessione deve riportare il percorso assoluto o relativo al Database locale;
*ELIMINAZIONE DATI OBSOLETI:* qualora la terna MINTIMESTAMP, MAXTIMESTAMP, TABTIMESTAMP (tabella contente il Timestamp) non contenga valori "null", il programma può provvedere alla generazione dello Statement per la cancellazione dei dati con la modalità descritta nei capitoli

precedenti. Terminata questa operazione, le tabelle dei database periferici contengono solo dati "nuovi" ovvero non presenti in alcuna riga delle tabelle (persistenti) presenti sul database centrale.

*CONNESSIONE AL SERVER CENTRALE*: connessione al Server MySQL. In questa fase sono attive 2 connessioni differenti: quella al database locale e quella al database Server: è quindi il SecThread a realizzare effettivamente il trasferimento delle informazioni a database Centrale.

*CREAZIONE TABELLE TEMPORANEE:* se connessione al database centrale è andata a buon fine, vengono create le tabelle tabelle temporanee il cui nome denota a quale coppia (nodo, query) appartiene, ad esempio:




A seconda dell'intervallo di interrogazione impostato per le varie query globali, a seguito di questa fase in un dato istante t sul database Centrale possono essere presenti da `1` a (`n°nodi * n° query globali)` tabelle temporanee.

*INSERIMENTO DATI DA DB LOCALE A TABELLA TEMPORANEA*: in questa fase viene eseguita la query locale e per ogni riga del resultSet ottenuto viene eseguita la `INSERT` sulla tabella temporanea. Quindi per ogni tabella temporanea devono essere eseguite un numero di transazioni pari al numero delle righe del resultSet ottenuto sulla tabella locale; Si sarebbe potuta eseguire una singola transazione ma anche in questa configurazione non si riscontra degrado di prestazioni poiché MySQL garantisce ottime prestazioni anche svolgendo delle transazioni singole (vedi capitolo 2 sulla analisi comparativa dei DBMS)

ATTESA COMPLETAMENTO DEGLI ALTRI NODI: completata l'operazione di inserimento dei dati nei nodi le connessioni al database locale e al database centrale vengono chiuse, viene eseguito il controllo sul numero di processi SecThread ancora attivi, se risulta che il thread corrente è l'ultimo in esecuzione

allora viene mandato il comando di risveglio, cioè Signal(), al MainThread, prima di porsi nello stato di Wait().

INVIO SIGNAL() A MAINTHREAD: stadio intermedio che precede la sospensione del processo in cui si viene inviata la Signal();

SOSPENSIONE: stadio finale del SecThread in cui si mette in attesa del segnale di risveglio per poter cominciare un nuovo ciclo di invii dati al Server centrale.

## 4.2. L'interfaccia Grafica di Solar Data Manager

L'ultima parte del progetto consiste nella realizzazione della interfaccia web per la visualizzazione e modifica dei dati dell'impianto.
L'obiettivo è di creare una Graphical User Interface (SDM-GUI) tramite la quale l'utente possa interagire con il sistema: tramite questa interfaccia utente è possibile non solo conoscere lo stato dell'impianto e dei singoli nodi, ma anche inviare direttive per il funzionamento di questi ultimi.

Le pagine web come decritto nel capitolo 2 sono state realizzate con tecnologia "Java Server Pages" (JSP), portando così alla realizzazione di un SW integralmente java-based.

Gli elementi caratterizzanti della SDM-GUI sono:

- **Il grafo solare:** grafico che permette all'utente di conoscere la posizione del sole in relazione all'istante di visualizzazione, latitudine e longitudine dell'osservatore;
- **Sintesi dei dati di funzionamento:** l'utente deve poter avere una visione d'insieme dell'impianto, immediata e di facile comprensione, che sintetizzi le informazioni principali riguardanti lo stato di funzionamento dell'impianto;
- **Pagine di dettaglio:** oltre ai dati sintetici devono essere fornite le informazioni nel dettaglio sul funzionamento dei singoli Sun Tracker, così da fornire all'utente ogni tipo di informazione utile per la valutazione dello stato dell'impianto (anche quelle che non si potrebbero ricavare da una visione sintetica e d'insieme) e poter intervenire in maniera mirata e in tempo reale sul sistema.

### 4.2.1. Diagramma a stati della SDM-GUI

La figura 30 mostra diagramma a stati della user interface, così come è stato definito dalla azienda Sungen srl.

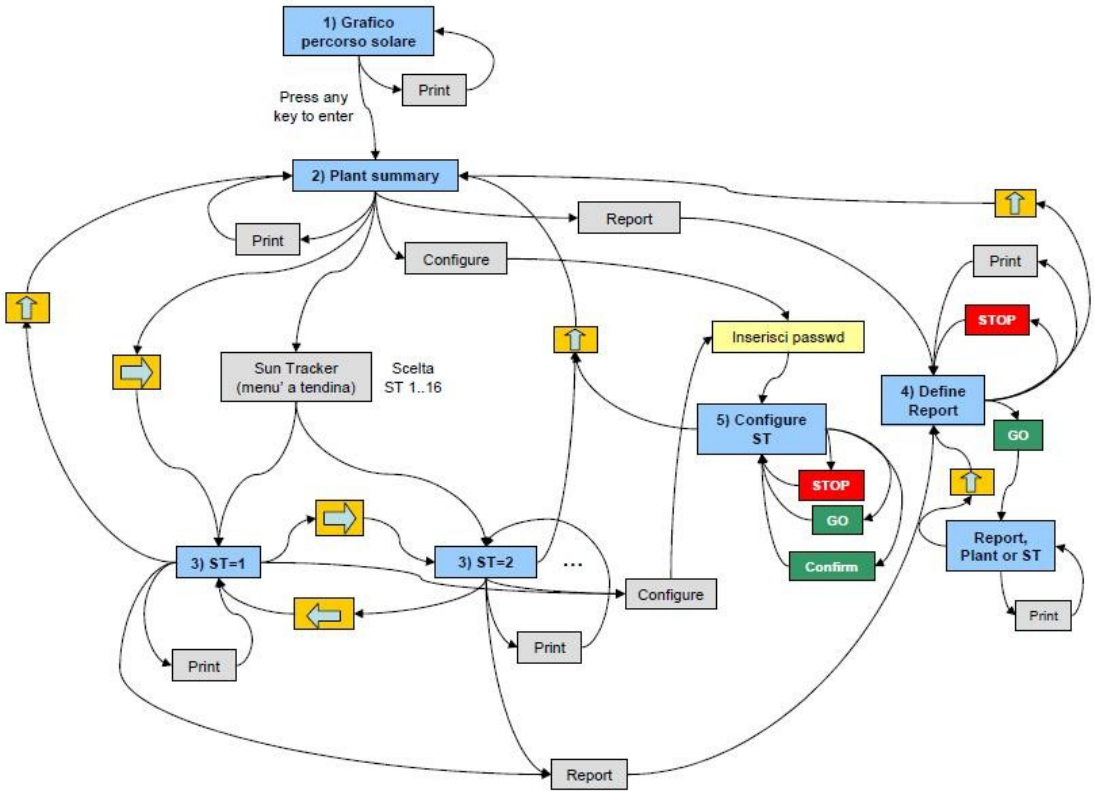



**Figura 30 – SDM Graphical User Interface – Diagramma a stati**

L'interfaccia grafica è composta da 5 stati:

- Home Page – Visualizzazione grafico percorso solare
- Panoramica impianto
- Stato del sun tracker
- Report (impianto/sun tracker)
- Configurazione

Nel paragrafo successivo vengono mostrate le schermate e definite le principali funzioni.

### 4.2.2. Le pagine web della SDM-GUI

La prima schermata a cui si accede (e pertanto nominata "index.jsp" ) è la panoramica la Home Page, il cui elemento principale è il grafico del percorso solare.

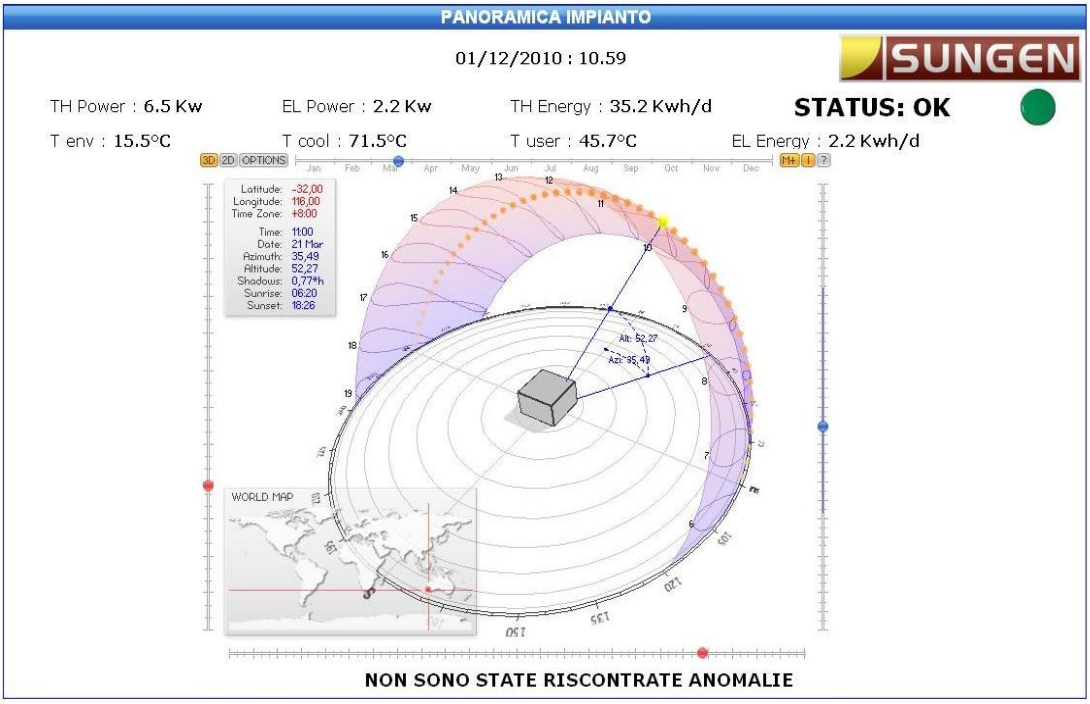



**Figura 31 - SDM-GUI - Schermata iniziale**

Questa pagina web è molto sintetica e, oltre all'elemento principale costituito dal tool grafico "Solar Path", mostra solo i dati essenziali relativi allo stato dell'impianto come l'energia prodotta, le temperature di funzionamento, lo stato complessivo e, in fondo alla pagina, un breve messaggio riassuntivo.

La pagina realizzata permette già ad un primo sguardo di comprendere se l'impianto sta funzionando correttamente o se invece è necessaria una ispezione più approfondita del sistema a causa di eventuali anomalie.

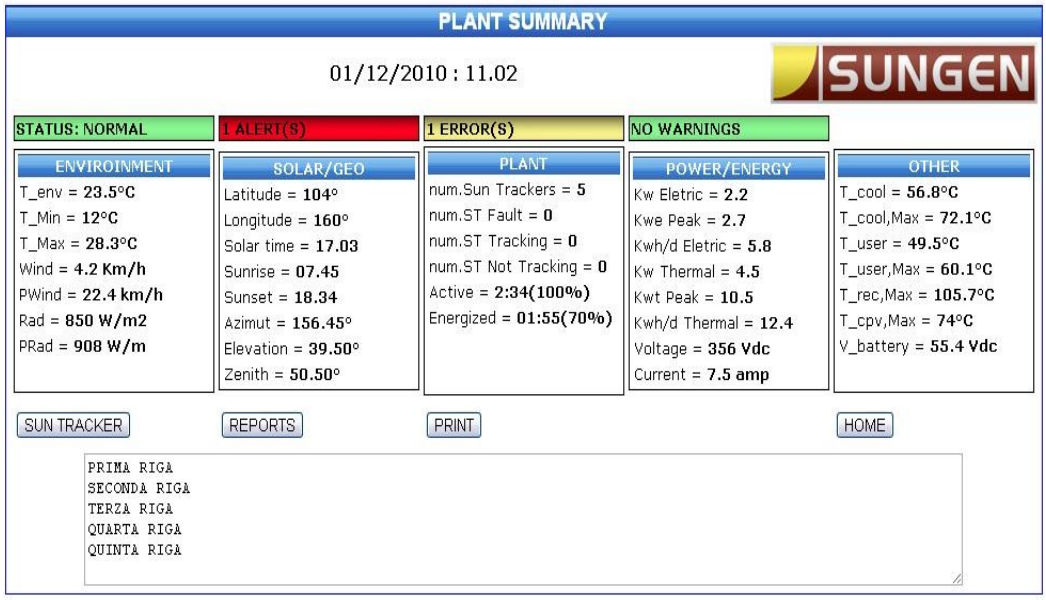



**Figura 32 - SDM-GUI – Panoramica impianto**

La figura 32 mostra la schermata relativa al Plant Summary (Panoramica dell'impianto). Scorrendo la schermata dall'alto al basso si può notare come essa sia divisa in 4 sezioni:

- Box-status
- Tabelle di stato
- Pulsanti di funzione
- Area di notifica

Le box-status permettono di mostrare all'utente l'eventuale presenza di problemi con l'ausilio di codici cromatici molto semplici: il verde indica la condizione di regolarità e il rosso e il giallo indicano la condizione di anomalia.
Con i tasti di funzioni è possibile spaziare in qualunque degli stati definiti al paragrafo 5.1

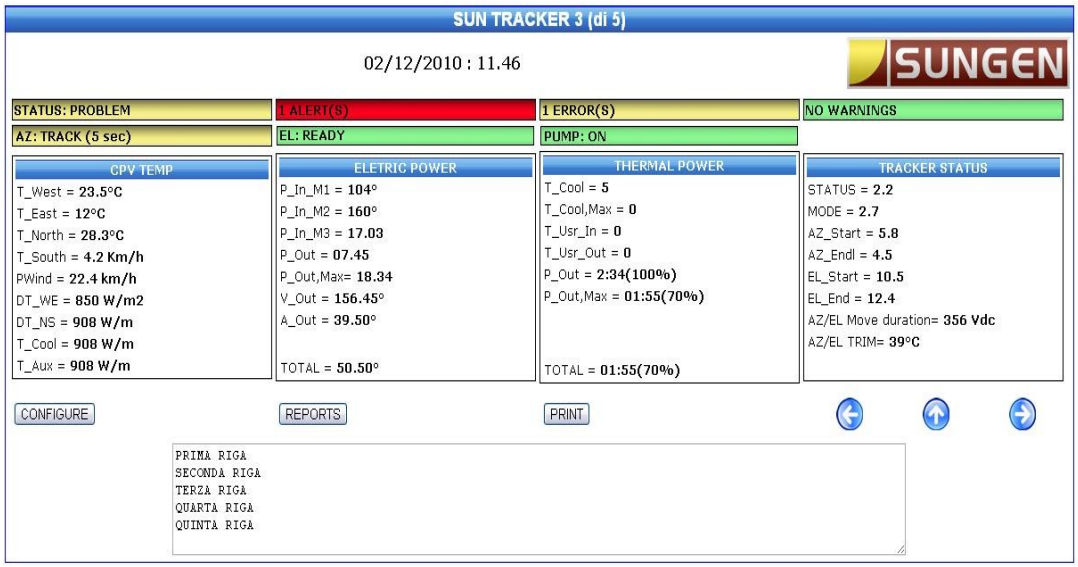



**Figura 33 - SDM-GUI - Sun Tracker Status**

La schermata di visualizzazione dei singoli Sun Tracker è molto simile a quella relativa all'intero impianto; essa non fornisce dati sulla produzione di energia, ma soltanto dati istantanei sul funzionamento del dispositivo di raccolta di energia solare.

La figura 34 mostra invece la schermata di reportistica. Qui è possibile ottenere i dati di funzionamento sia dell'intero impianto che di un singolo sun tracker; i dati ottenuti sono personalizzabili ottenibili su un orizzonte temporale massimo di 1 anno.

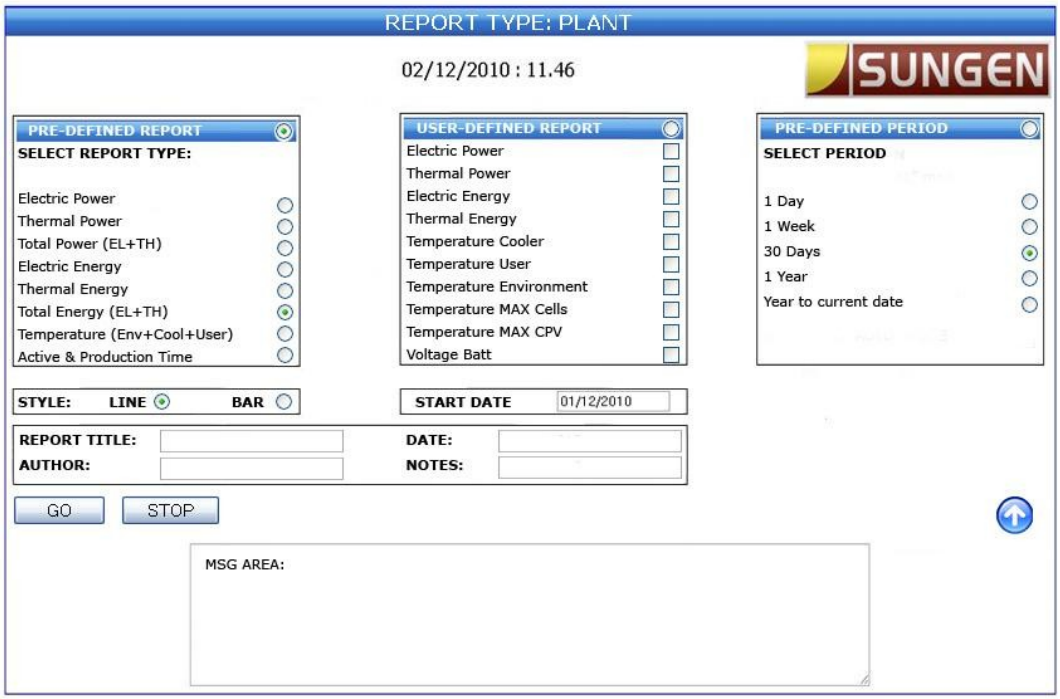



**Figura 34 - SDM-GUI - Report impianto**

L'ultima schermata che viene illustrata è il pannello di configurazione. Da qui è possibile impostare i parametri di funzionamento di ogni Sun Tracker (ST), effettuare test di funzionamento definire la modalità di funzionamento del dispositivo ("AUTO" o "MANUAL").

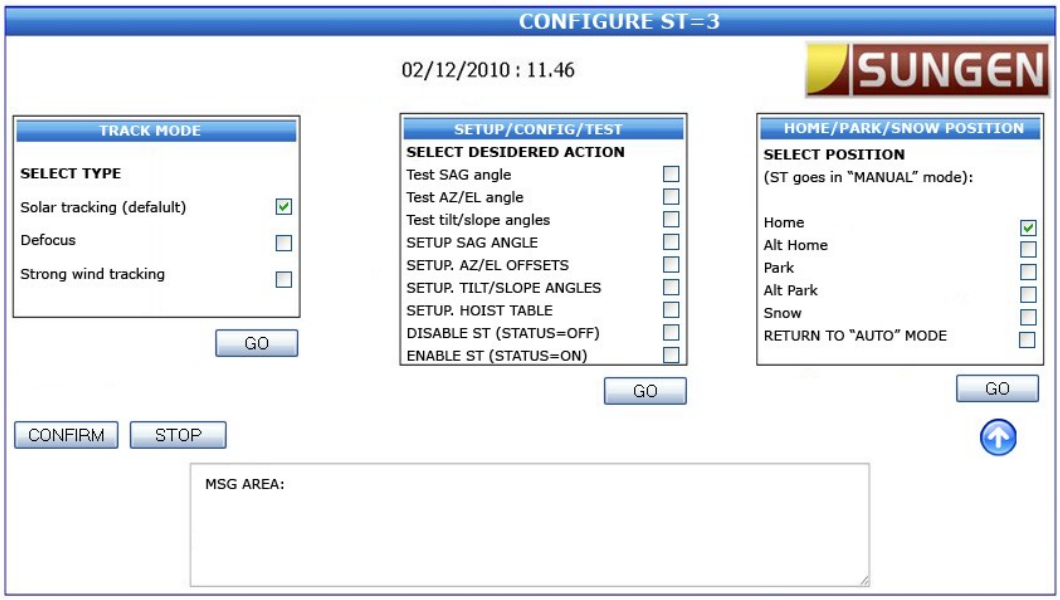



**Figura 35 - SDM-GUI - Configurazione impianto**

Come è stato accennato ad inizio capitolo i dati attualmente visualizzati sono solo simulati.

Per dovere di completezza si aggiunge l'SDM-GUI una volta completa permetterà non solo di visualizzare i dati immagazzinati nel database grazie all'utilizzo del network wrapper, darà occasione di monitorare i dati prodotti in real-time da ciascun Sun Tracker e contenuti all'interno del proprio Shared Memory Module (elemento ancora in fase di sviluppo da parte di Sungen).

Sarà la combinazione di dati persistenti e storici (database centrale) e di dati istantanei (Shared Memory Module) a fornire all'utente uno strumento completo per la visualizzazione e reportistica dell'impianto di generazione di energia solare.

# CAP 5. Analisi prestazioni del sistema

Nel seguente capitolo si illustreranno brevemente i test effettuati durante la realizzazione del network wrapper.
Si specifica che durante lo sviluppo della tesi non è stato possibile avere accesso all'HW del dominio applicativo. Pertanto in mancanza dell'HW specifico, i test sono stati effettuati su PC general purpose.

Come ambiente di test sono state usate 2 configurazioni:

- **Server:** Processore: AMD Athlon 2.4 Ghz, 1 GB RAM, Spazio HD: 120Gb, S.O. Windows XP Professional, Scheda di rete: Realtek RTL8169/8110 Gigabit LAN;
  **Nodo**: Processore: AMD Athlon 3.0 Ghz , 2 GB RAM, Spazio HD: 80Gb, 80Gb, S.O. Ubuntu Linux 10.04(Lucid Lynx), Scheda di rete: Broadcom 802.11g WLAN
- **Server:** Processore: AMD Turion64 1.8 Ghz, 512MB RAM, Spazio HD: 80Gb, S.O. Windows XP Professional, Scheda di rete: Realtek RTL8169/8110 Gigabit LAN;
  **Nodo**: Processore: AMD Athlon 3.0 Ghz , 2 GB RAM, Spazio HD: 80Gb, 80Gb, S.O. Ubuntu Linux 10.04(Lucid Lynx), D-Link wireless Adapter DWL G122

I test sono stati effettuati in 3 diverse condizioni di efficienza di ricezione del segnale wireless da parte del PC (notebook) che simula il nodo remoto: potenza del segnale 100%, potenza del segnale 60%, potenza del segnale 20%; il PC che simula in server è invece connesso direttamente al router.
Nella simulazione fatta si doveva eseguire il wrappping dei dati contenuti in tre database diversi contenuti i folder differenti ma all'interno dello stesso notebook la cui connessione era garantita da 3 differenti channel SSH.

I risultati ottenuti sono stati i seguenti:

| | TEMPO DI ESCUZIONE MEDIO WRAPPING (sec) | |
|---|---|---|
| | CONFIGURAZIONE 1 | CONFIGURAZIONE 2 |
| Segnale:100% | 2.8 | 5.3 |
| Segnale: 60% | 4.5 | 6.2 |
| Segnale: 20% | 8.1 | 9.7 |

**Figura 36 - Tabella test network wrapper**


**Figura 37 - Grafico test network wrapper**

**Analisi dei risultati ottenuti**

Lo svolgimento di questi test ha messo permesso di valutare le prestazioni del wrapper sia in relazione alla potenza del segnale che in base alla efficienza del Server.

Dal punto di vista dell'efficienza in rapporto all'efficienza della wireless LAN si nota come essa condizioni fortemente le prestazioni portando ad un forte degrado delle prestazioni, fino ad arrivare alla impossibilità di scambio di dati tra nodo e Server. Per motivi di sicurezza le connessioni SSH in questo SW sono state impostate con un timeout di 5 secondi pertanto se le condizioni della rete non

consentono di inviare dei pacchetti in questo periodo avviene una chiusura automatica dei channel SSH, con conseguente impossibilità di inviare dati.

Il secondo fattore che si è potuto testare è l'efficienza complessiva del sistema il relazione all'efficienza del server. Il risultato ottenuto è stato soddisfacente poiché in condizioni ottimali (configurazione 1 + potenza del segnale=100% ) è stato possibile avere un passaggio di dati completo in tempi molto rapidi, considerando che comunque per la realizzazione della WLAN si è utilizzato un router domestico. La forbice tra configurazione 1 e configurazione 2 si assottiglia al diminuire dell'efficienza della WLAN.

# CAP 6. Conclusioni e prospettive future

Per quanto questa tesi abbia affrontato solo una parte del complesso problema della realizzazione di un sistema di produzione di energia solare, la realizzazione del programma Solar Data Manager si è rivelata altrettanto complessa.
Questo ha portato alla scomposizione del problema principale (la realizzazione del SW) in tanti sottoproblemi e alla soluzione a piccoli passi degli stessi.
Un altro problema non banale è stata l'impossibilità di non aver occasione di poter visionare di persona il dispositivo (sun tracker) al centro della realizzazione del SW, fino a poche settimane prima del completamento di questa tesi.
Nonostante questo è stato realizzato un prototipo di SW capace di venire incontro alle esigenze del committente, basato un linguaggio multipiattaforma come Java e su un RDBMS "universale" come MySQL.

**Figura 38 - Foto prototipo di Sun Tracker**

Ovviamente quanto realizzato è solo una bozza di quello potrà diventare Solar Data Manager nella sua versione definitiva.
Infatti mancano ancora alcune parti come:

- Definizione della struttura definitiva del database;
- Interfacciamento di SDM-UI con il wrapper e definizione degli esatti contenuti della user interface;

- Definizione di protocolli di interfacciamento con il modulo Shared Memory Area: questo modulo sarà fondamentale per la visualizzazione di dati real time dello stato dei Sun Tracker;
- Definizione della struttura della stessa Shared Memory Area.

Inoltre vi sono ancora margini di miglioramento anche per quanto riguarda il network wrapper, esso potrà spaziare tra gestione e analisi delle query globali e protocolli di comunicazione tra nodo e Server.

# APPENDICE A. Il Database del Sistema

In questa sezione viene illustrata la struttura del database definito da SUNGEN per Solar Data Manager.

## A.1. Diagramma del Database

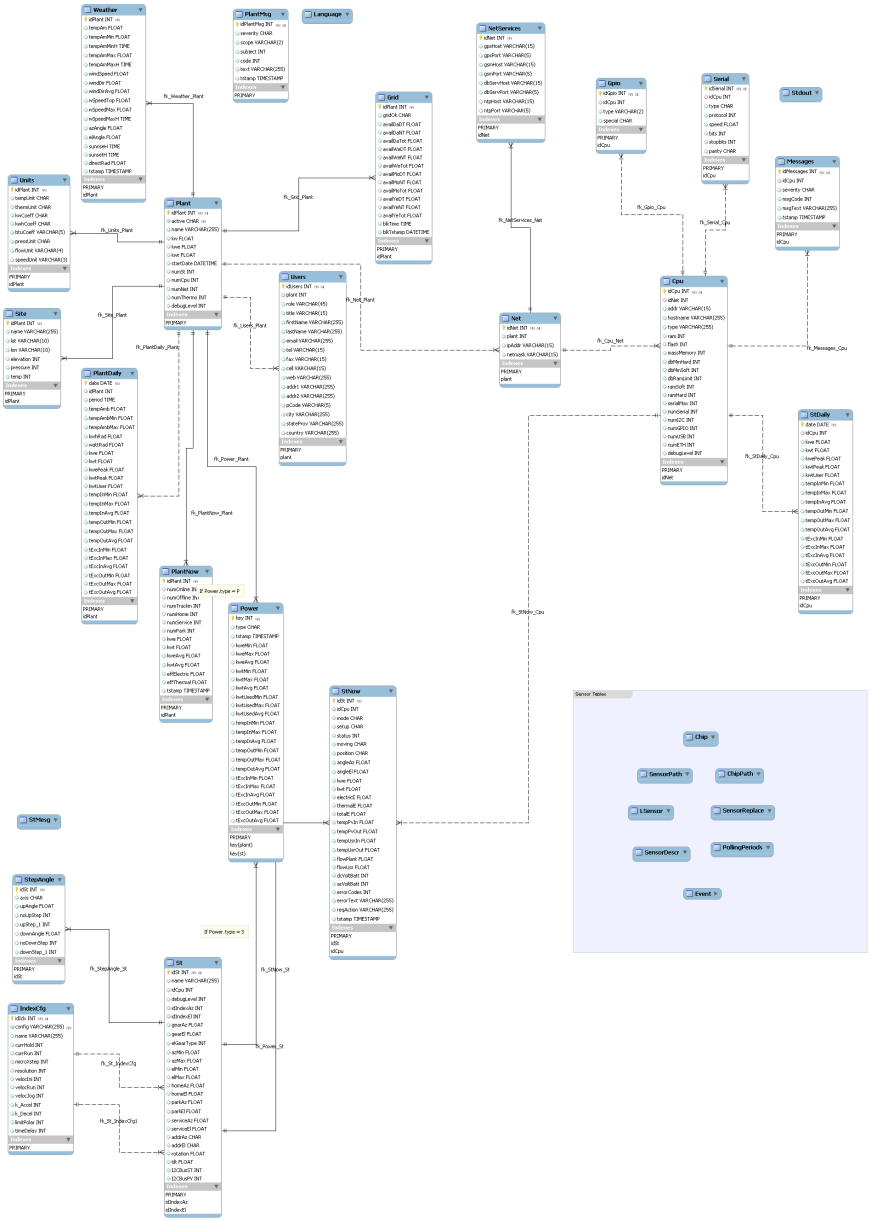


**Figura 39 - Diagramma del Database**

## A.2. Tabelle

### A.2.1. Panoramica

In questa sezione viene fatta una panoramica di tutte le tabelle con la loro funzione.

Le tabelle SQL sono di due tipi diversi, evidenziati con diversi colori:

Le tabelle in **NERO** descrivono la **struttura di impianto;**

La tabelle in **ROSSO** accumulano **dati di funzionamento**, perciò non hanno valori iniziali e sono popolate con i valori misurati durante il funzionamento impianto.

| **SolarSW DB Tables** | |
|---|---|
| **Table Name** | **Description** |
| *Tabelle Impianto* | |
| Plant | Dati identificativi dell'impianto |
| db_config | Dati relativi all'ubicazione dei database remoti |
| Globalqueries | Dati delle global queries |
| Units | Unità di misura |
| Languages | Localizzazione stringhe messaggi. Da valutare se utilizzare database oppure file di testo esterni |
| Users | Nomi ed indirizzi dei contatti tecnici, amministrativi, manutentori per l'impianto |
| Site | Parametri geografici sito installazione impianto |
| PlantNow | Stato corrente dell'impianto. Aggiornata ogni 1-2 secondi. |
| Weather | Informazioni meteo rilevate. Aggiornata ogni minuto tranne che per le informazioni che variano repentinamente (vento, ecc..). Quest'ultime aggiornate ogni 1-2 |

| | |
|---|---|
| | secondi |
| Power | Energia elettrica/termica totale prodotta dall'impianto e energia termica effettivamente prelevata dall'utente |
| Grid | Availability della rete elettrica |
| PlantMsg | Logging messaggi impianto |
| PlantDaily | Sommario attività rilevanti singola giornata |
| *Tabelle di Rete* | |
| Net | Descrizione delle reti IP utilizzate |
| NetService | Servizi disponibili sulla rete (GPS, NTP, GSM, DB) |
| *Tabelle Centralina* | |
| Cpu | Descrizioni centraline componenti l'impianto |
| Serial | Descrizione porte seriali configurabili per centralina |
| Gpio | Descrizione porte I/O configurabili per centralina |
| Messages | Logging Cpu |
| Stdout | Messaggi del S.O. Valutare la reale necessità |
| *Tabelle Inseguitori(Sun trackers)* | |
| St | Descrizione singoli Sun Tracker |
| IndexCfg | Configurazioni Indexer/Driver motori |
| StepAngle | Mappatura tra passi del motore e angoli |
| StNow | Stato corrente singolo Sun Tracker |
| StMsg | Logging Sun Tracker |
| StDaily | Sommario attività rilevanti singola giornata |
| *Tabelle Sensori* | |
| PollingPeriod | Durata dei cicli di polling |
| LSensor | Dichiara i sensori dal punto di vista logico |
| SensorDescr | Caratteristiche dello specifico sensore |
| SensorReplace | Tabella di sostituzione sensori |
| Sensors_P1 | Lettura dati sensori Polling Period 1 |
| Sensors_P2 | Lettura dati sensori Polling Period 2 |
| Sensors_P3 | Lettura dati sensori Polling Period 3 |
| Sensors_P4 | Lettura dati sensori Polling Period 4 |
| *Tabelle Bus e Chip* | |
| Chip | Descrizione chip |

| | |
|---|---|
| SensorPath | Descrizione topologia bus; posizione sensore |
| ChipPath | Descrizione topologia bus; posizione sensore |
| *Tabelle gestione Eventi* | |
| Event | Eventi semplici |

**Tabella 4 – Panoramica tabelle del Sistema**

## A.2.2. Dettaglio Tabelle

### Table Plant

**Columns**

| Key | Column Name | Datatype | Not Null | Default | Comment |
|---|---|---|---|---|---|
| PK | idPlant | INT | Yes | | |
| | Active | CHAR | Yes | 'Y' | impianto attivo Y/N |
| | Name | VARCHAR(255) | No | 'Sungen' | nome dell'impianto |
| | Kw | FLOAT | No | 9 | potenza totale impianto (elettrica+termica) |
| | Kwe | FLOAT | No | 3 | potenza elettrica |
| | Kwt | FLOAT | No | 6 | potenza termica |
| | startDate | DATETIME | No | | data di avvio impianto |
| | numSt | INT | No | 1 | numero inseguitori solari |
| | numCpu | INT | No | 1 | numero centraline di controllo |
| | numNet | INT | No | 1 | numero di reti locali |
| | numThermo | INT | No | 1 | numero termo box |
| | debugLevel | INT | No | 1 | verbosità debug: 0/1/2/3 |

**Indices**

| Index Name | Columns | Primary | Unique | Type | Kind | Comment |
|---|---|---|---|---|---|---|
| PRIMARY | | Yes | No | PRIMARY | | |

**Relationships**

| Relationship Name | Relationship Type | Parent Table | Child Table | Card. |
|---|---|---|---|---|
| IdPlant | Non-Identifying | Plant | Units | 1:n |

| | | | | |
|---|---|---|---|---|
| Plant | Non-Identifying | Plant | Users | 1:n |
| IdPlant | Non-Identifying | Plant | Site | 1:n |
| IdPlant | Non-Identifying | Plant | PlantNow | 1:n |
| IdPlant | Non-Identifying | Plant | Weather | 1:n |
| key(plant) | Non-Identifying | Plant | Power | 1:n |
| IdPlant | Non-Identifying | Plant | Grid | 1:n |
| IdPlant | Non-Identifying | Plant | PlantDaily | 1:n |
| Plant | Non-Identifying | Plant | Net | 1:n |

### Table db_config

**Columns**

| Key | Column Name | Datatype | Not Null | Default | Comment |
|---|---|---|---|---|---|
| PFK | id | INT | Yes | | Id database |
| | type | INT | Yes | | Tipo database (0=server; 1=nodo) |
| | name | VARCHAR(50) | No | | Nome del database |
| | user | VARCHAR(50) | No | | Nome utente del database |
| | pass | VARCHAR(50) | No | | Password per accedere a database |
| | IP | VARCHAR(50) | No | | IP del database |

**Indices**

| Index Name | Columns | Primary | Unique | Type | Kind | Comment |
|---|---|---|---|---|---|---|
| ID | | Yes | Yes | PRIMARY | | |

### Table globalqueries

**Columns**

| Key | Column Name | Datatype | Not Null | Default | Comment |
|---|---|---|---|---|---|
| PK | Qry_id | INT | Yes | | Id della query |
| | Active | INT | | | Query attiva(0=no; 1=sì) |
| | qry_txt | VARCHAR(50) | No | | Testo della query |
| | Nodes | VARCHAR(50) | No | | Nodi da contattare |
| | Tables_srv | VARCHAR(50) | No | | Tablella del |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | server |
| | Interval | INT | No | | Intervallo tra 2 richieste successive |
| | description | VARCHAR(50) | | | Descrizione |

**Indices**

| Index Name | Columns | Primary | Unique | Type | Kind | Comment |
|---|---|---|---|---|---|---|
| qry_id | | Yes | Yes | PRIMARY | | |

## Table Units

**Columns**

| Key | Column Name | Datatype | Not Null | Default | Comment |
|---|---|---|---|---|---|
| PFK | idPlant | INT | Yes | | idPlant |
| | tempUnit | CHAR | No | 'C' | C/F - Unità temperatura: Celsius, Fahreneit |
| | thermUnit | CHAR | No | 'W' | W/C/B/J - Unità energia termica: Wh, KCal, Btu, J |
| | kwCoeff | CHAR | No | 'K' | 1/K/M - Coeff moltiplicativo potenza: Watt, Kw, Mw |
| | kwhCoeff | CHAR | No | 'K' | 1/K/M - Coeff moltiplicativo energia: Watth, Kwh, Mwh |
| | btuCoeff | VARCHAR(5) | No | 'M' | 1/M/MM/therm - Coeff moltiplicativi Btu: Btu, MBtu (Btu x 1000), MMBtu (Btu x 1M), therm (Btu x 100000) |
| | pressUnit | CHAR | No | 'B' | B/P/M - Unità pressione: Bar, Psi, Mpascal |
| | flowUnit | VARCHAR(4) | No | 'Lm' | Lm/Ls/Mc/Gpm/Gph/lgpm/lgph - Unità flusso: litri/min, litri/sec, mc/h, galloni/min, galloni/h, galloni/min (imperial), galloni/ora (imperial) |
| | speedUnit | VARCHAR(3) | No | 'Kmh' | Kmh/ms/Mph - Unità velocità: km/h, m/sec, miles/hour |

**Indices**

| Index Name | Columns | Primary | Unique | Type | Kind | Comment |
|---|---|---|---|---|---|---|
| PRIMARY | | Yes | No | PRIMARY | | |
| idPlant | | No | No | FOREIGN | | |

**Relationships**

| Relationship Name | Relationship Type | Parent Table | Child Table | Card. |
|---|---|---|---|---|
| IdPlant | Identifying | Plant | Units | 1:n |

## Table Users

**Columns**

| Key | Column Name | Datatype | Not Null | Default | Comment |
|---|---|---|---|---|---|
| PK | idUsers | INT | Yes | | |
| | plant | INT | No | | |
| | role | VARCHAR(45) | No | | |
| | title | VARCHAR(15) | No | | |
| | firstName | VARCHAR(255) | No | | |
| | lastName | VARCHAR(255) | No | | |
| | email | VARCHAR(255) | No | | |
| | tel | VARCHAR(15) | No | | |
| | fax | VARCHAR(15) | No | | |
| | cell | VARCHAR(15) | No | | |
| | web | VARCHAR(255) | No | | |
| | addr1 | VARCHAR(255) | No | | |
| | addr2 | VARCHAR(255) | No | | |
| | pCode | VARCHAR(5) | No | | |
| | city | VARCHAR(255) | No | | |
| | stateProv | VARCHAR(255) | No | | |
| | Country | VARCHAR(255) | No | | |

**Indices**

| Index Name | Columns | Primary | Unique | Type | Kind | Comment |
|---|---|---|---|---|---|---|
| PRIMARY | | Yes | No | PRIMARY | | |
| plant | | No | No | FOREIGN | | |

**Relationships**

| Relationship Name | Relationship Type | Parent Table | Child Table | Card. |
|---|---|---|---|---|
| Plant | Non-Identifying | Plant | Users | 1:n |

## Table Site

**Columns**

| Key | Column Name | Datatype | Not Null | Default | Comment |
|---|---|---|---|---|---|
| PFK | idPlant | INT | Yes | | idPlant |
| | Name | VARCHAR(255) | No | | nome del sito |

| | | | | | |
|---|---|---|---|---|---|
| | Lat | VARCHAR(10) | No | | latitudine |
| | Lon | VARCHAR(10) | No | | longitudine |
| | elevation | INT | No | | altezza in metri slm |
| | pressure | INT | No | | pressione media reale in mBar (non normalizzata) |
| | Temp | INT | No | | temperatura media annua in C |

**Indices**

| Index Name | Columns | Primary | Unique | Type | Kind | Comment |
|---|---|---|---|---|---|---|
| PRIMARY | | Yes | No | PRIMARY | | |
| idPlant | | No | No | FOREIGN | | |

**Relationships**

| Relationship Name | Relationship Type | Parent Table | Child Table | Card. |
|---|---|---|---|---|
| idPlant | Identifying | Plant | Site | 1:n |

## Table PlantNow

**Columns**

| Key | Column Name | Datatype | Not Null | Default | Comment |
|---|---|---|---|---|---|
| PFK | idPlant | INT | Yes | | identificatore impianto |
| | numOnline | INT | No | | numero inseguitori attivi |
| | numOffline | INT | No | | numero inseguitori non attivi (spenti, offline) |
| | numTrackin | INT | No | | numero inseguitori in tracking automatico (fra alba e tramonto) |
| | numHome | INT | No | | numero inseguitori in posizione "Home" |
| | numService | INT | No | | numero inseguitori in posizione |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | "Service" |
| | numPark | INT | No | | numero inseguitori in posizione "Park" |
| | kwe | FLOAT | No | | potenza istantanea elettrica |
| | kwt | FLOAT | No | | potenza istantanea termica |
| | kweAvg | FLOAT | No | | energia elettrica media nei 10 minuti precedenti, in kwh |
| | kwtAvg | FLOAT | No | | energia termica media nei 10 minuti precedenti, in kwh |
| | effElectric | FLOAT | No | | efficienza elettrica istantanea (se disponibile sensore) |
| | effThermal | FLOAT | No | | efficienza termica istantanea (se disponibile sensore) |
| | Tstamp | TIMESTAMP | No | | timestamp aggiornamento tabella |

**Indices**

| Index Name | Columns | Primary | Unique | Type | Kind | Comment |
|---|---|---|---|---|---|---|
| PRIMARY | | Yes | No | PRIMARY | | |
| idPlant | | No | No | FOREIGN | | |

**Relationships**

| Relationship Name | Relationship Type | Parent Table | Child Table | Card. |
|---|---|---|---|---|
| idPlant | Identifying | Plant | PlantNow | 1:n |

## Table Weather

**Columns**

| Key | Column Name | Datatype | Not Null | Default | Comment |
|---|---|---|---|---|---|
| PFK | idPlant | INT | Yes | | identificatore |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | impianto |
| | tempAm | FLOAT | No | | temperatura ambiente in C, corrente |
| | tempAmMin | FLOAT | No | | temperatura minima in C, del giorno |
| | tempAmMinH | TIME | No | | Orario temperatura minima |
| | tempAmMax | FLOAT | No | | temperatura massima in C, del giorno |
| | tempAmMaxH | TIME | No | | orario temperatura massima |
| | windSpeed | FLOAT | No | | velocità media del vento, corrente |
| | windDir | FLOAT | No | | direzione del vento, corrente |
| | windDirAvg | FLOAT | No | | angolo, misurato da N in direzione oraria |
| | wSpeedTop | FLOAT | No | | velocità di picco del vento, corrente |
| | wSpeedMax | FLOAT | No | | picco di velocità del giorno |
| | wSpeedMaxH | TIME | No | | orario di velocità massima del vento |
| | azAngle | FLOAT | No | | angolo solare Azimuth |
| | elAngle | FLOAT | No | | angolo solare elevazione |
| | sunriseH | TIME | No | | ora alba |
| | sunsetH | TIME | No | | ora tramonto |
| | directRad | FLOAT | No | | radiazione diretta (se presente strumento) |
| | Tstamp | TIMESTAMP | No | | timestamp tabella |

**Indices**

| Index Name | Columns | Primary | Unique | Type | Kind | Comment |
|---|---|---|---|---|---|---|

| | | | | | | |
|---|---|---|---|---|---|---|
| PRIMARY | | Yes | No | PRIMARY | | |
| idPlant | | No | No | FOREIGN | | |

**Relationships**

| Relationship Name | Relationship Type | Parent Table | Child Table | Card. |
|---|---|---|---|---|
| idPlant | Identifying | Plant | Weather | 1:n |

## Table Power

**Columns**

| Key | Column Name | Datatype | Not Null | Default | Comment |
|---|---|---|---|---|---|
| PFK | key | INT | Yes | | puntatore a indice ST o PLANT |
| | type | CHAR | No | | Tipo di chiave: S(t) oppure (P)lant |
| | tstamp | TIMESTAMP | No | | timestamp tabella |
| | kweMin | FLOAT | No | | kw elettrici, min |
| | kweMax | FLOAT | No | | kw elettrici, max |
| | kweAvg | FLOAT | No | | kw elettrici, valore medio |
| | kwtMin | FLOAT | No | | kw termici, min |
| | kwtMax | FLOAT | No | | kw termici, max |
| | kwtAvg | FLOAT | No | | kw termici, valore medio |
| | kwtUsedMin | FLOAT | No | | kw termici effettivamente prelevati dall'utente, min |
| | kwtUsedMax | FLOAT | No | | kw termici effettivamente prelevati dall'utente, max |
| | kwtUsedAvg | FLOAT | No | | kw termici effettivamente prelevati dall'utente, valore medio |
| | tempInMin | FLOAT | No | | temperatura fluido ingresso moduli FV, min |
| | tempInMax | FLOAT | No | | temperatura fluido ingresso moduli FV, max |
| | tempInAvg | FLOAT | No | | temperatura fluido ingresso |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | moduli FV, valore medio |
| | tempOutMin | FLOAT | No | | temperatura fluido uscita moduli FV, min |
| | tempOutMax | FLOAT | No | | temperatura fluido uscita moduli FV, max |
| | tempOutAvg | FLOAT | No | | temperatura fluido uscita moduli FV, valore medio |
| | tExcInMin | FLOAT | No | | temperatura ingresso scambiatore utente, min |
| | tExcInMax | FLOAT | No | | temperatura ingresso scambiatore utente, max |
| | tExcInAvg | FLOAT | No | | temperatura ingresso scambiatore utente, valore medio |
| | tExcOutMin | FLOAT | No | | temperatura uscita scambiatore utente, min |
| | tExcOutMax | FLOAT | No | | temperatura uscita scambiatore utente, max |
| | tExcOutAvg | FLOAT | No | | temperatura uscita scambiatore utente, valore medio |

**Indices**

| Index Name | Columns | Primary | Unique | Type | Kind | Comment |
|---|---|---|---|---|---|---|
| PRIMARY | | Yes | No | PRIMARY | | |
| key(plant) | | No | No | FOREIGN | | |
| key(st) | | No | No | FOREIGN | | |

**Relationships**

| Relationship Name | Relationship Type | Parent Table | Child Table | Card. |
|---|---|---|---|---|
| key(plant) | Identifying | Plant | Power | 1:n |
| key(st) | Identifying | St | Power | 1:n |

## Table Grid

**Columns**

| Key | Column Name | Datatype | Not Null | Default | Comment |
|---|---|---|---|---|---|
| PFK | idPlant | INT | Yes | | identificatore impianto |
| | gridOk | CHAR | No | | presenza rete elettrica Y/N |
| | availDaDT | FLOAT | No | | % disponibilità rete elettrica giorno corrente, daytime |
| | availDaNT | FLOAT | No | | % disponibilità rete elettrica giorno corrente, nighttime |
| | availDaTot | FLOAT | No | | % disponibilità rete elettrica giorno corrente, totale |
| | availWeDT | FLOAT | No | | % disponibilità rete elettrica settimana corrente, daytime |
| | availWeNT | FLOAT | No | | % disponibilità rete elettrica settimana corrente, nighttime |
| | availWeTot | FLOAT | No | | % disponibilità rete elettrica settimana corrente, totale |
| | availMoDT | FLOAT | No | | % disponibilità rete elettrica mese corrente, daytime |
| | availMoNT | FLOAT | No | | % disponibilità rete elettrica mese corrente, nighttime |
| | availMoTot | FLOAT | No | | % disponibilità rete elettrica mese corrente, totale |
| | availYeDT | FLOAT | No | | % disponibilità rete elettrica anno corrente, daytime |
| | availYeNT | FLOAT | No | | % disponibilità rete elettrica anno corrente, |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | nighttime |
| | availYeTot | FLOAT | No | | % disponibilità rete elettrica anno corrente, totale |
| | blkTime | TIME | No | | durata ultimo blackout registrato |
| | blkTstamp | DATETIME | No | | chiusura ultimo blackout |

**Indices**

| Index Name | Columns | Primary | Unique | Type | Kind | Comment |
|---|---|---|---|---|---|---|
| PRIMARY | | Yes | No | PRIMARY | | |
| idPlant | | No | No | FOREIGN | | |

**Relationships**

| Relationship Name | Relationship Type | Parent Table | Child Table | Card. |
|---|---|---|---|---|
| idPlant | Identifying | Plant | Grid | 1:n |

## Table PlantMsg

**Columns**

| Key | Column Name | Datatype | Not Null | Default | Comment |
|---|---|---|---|---|---|
| PK | idPlantMsg | INT | Yes | | identificatore messaggo |
| | severity | CHAR | No | | tipo msg: Fatal, Error, Warning, Information, Alert |
| | scope | VARCHAR(2) | No | | Plant, ST, Cpu, Board, Network, Sensor |
| | subject | INT | No | | indice entità |
| | code | INT | No | | codice numerico |
| | text | VARCHAR(255) | No | | stringa messaggio |
| | tstamp | TIMESTAMP | No | | timestamp messaggio |

**Indices**

| Index Name | Columns | Primary | Unique | Type | Kind | Comment |
|---|---|---|---|---|---|---|
| PRIMARY | | Yes | No | PRIMARY | | |

### Table PlantDaily

**Columns**

| Key | Column Name | Datatype | Not Null | Default | Comment |
|---|---|---|---|---|---|
| PK | Date | DATE | Yes | | data registrazione tupla, chiave primaria, una al giorno |
| | idPlant | INT | No | | identificatore impianto |
| | Period | TIME | No | | periodo di funzionamento |
| | tempAmb | FLOAT | No | | temperatura ambiente, media |
| | tempAmbMin | FLOAT | No | | temperatura ambiente, minima |
| | tempAmbMax | FLOAT | No | | temperatura ambiente, massima |
| | kwhRad | FLOAT | No | | kwh radiazione diretta |
| | wattRad | FLOAT | No | | radioazione diretta max |
| | Kwe | FLOAT | No | | kw elettrici prodotti nella giornata |
| | Kwt | FLOAT | No | | kw termici prodotti nella giornata |
| | kwePeak | FLOAT | No | | potenza elettrica istantanea, valore max |
| | kwtPeak | FLOAT | No | | potenza termica istantanea, valore max |
| | kwtUser | FLOAT | No | | kw termici effettivamente prelevati dall'utente |
| | tempInMin | FLOAT | No | | temperatura fluido ingresso moduli FV, min |
| | tempInMax | FLOAT | No | | temperatura fluido ingresso moduli FV, max |
| | tempInAvg | FLOAT | No | | temperatura fluido ingresso moduli FV, valore medio |

| | | | | | |
|---|---|---|---|---|---|
| | tempOutMin | FLOAT | No | | temperatura fluido uscita moduli FV, min |
| | tempOutMax | FLOAT | No | | temperatura fluido ingresso moduli FV, max |
| | tempOutAvg | FLOAT | No | | temperatura fluido ingresso moduli FV, valore medio |
| | tExcInMin | FLOAT | No | | temperatura ingresso scambiatore utente, min |
| | tExcInMax | FLOAT | No | | temperatura ingresso scambiatore utente, max |
| | tExcInAvg | FLOAT | No | | temperatura ingresso scambiatore utente, valore medio |
| | tExcOutMin | FLOAT | No | | temperatura uscita scambiatore utente, min |
| | tExcOutMax | FLOAT | No | | temperatura uscita scambiatore utente, max |
| | tExcOutAvg | FLOAT | No | | temperatura uscita scambiatore utente, valore medio |

**Indices**

| Index Name | Columns | Primary | Unique | Type | Kind | Comment |
|---|---|---|---|---|---|---|
| PRIMARY | | Yes | No | PRIMARY | | |
| idPlant | | No | No | FOREIGN | | |

**Relationships**

| Relationship Name | Relationship Type | Parent Table | Child Table | Card. |
|---|---|---|---|---|
| idPlant | Non-Identifying | Plant | PlantDaily | 1:n |

## Table Net

**Columns**

| Key | Column Name | Datatype | Not Null | Default | Comment |
|---|---|---|---|---|---|

| | | | | | |
|---|---|---|---|---|---|
| PK | idNet | INT | Yes | | indice |
| | plant | INT | No | | idPlant |
| | ipAddr | VARCHAR(15) | No | | indirizzo IP della rete |
| | netmask | VARCHAR(15) | No | | netmask rete |

**Indices**

| Index Name | Columns | Primary | Unique | Type | Kind | Comment |
|---|---|---|---|---|---|---|
| PRIMARY | | Yes | No | PRIMARY | | |
| plant | | No | No | FOREIGN | | |

**Relationships**

| Relationship Name | Relationship Type | Parent Table | Child Table | Card. |
|---|---|---|---|---|
| plant | Non-Identifying | Plant | Net | 1:n |
| idNet | Non-Identifying | Net | NetServices | 1:n |
| idNet | Non-Identifying | Net | Cpu | 1:n |

## Table NetServices

**Columns**

| Key | Column Name | Datatype | Not Null | Default | Comment |
|---|---|---|---|---|---|
| PFK | idNet | INT | Yes | | |
| | gpsHost | VARCHAR(15) | No | | |
| | gpsPort | VARCHAR(5) | No | | |
| | gsmHost | VARCHAR(15) | No | | |
| | gsmPort | VARCHAR(5) | No | | |
| | dbServHost | VARCHAR(15) | No | | |
| | dbServPort | VARCHAR(5) | No | | |
| | ntpHost | VARCHAR(15) | No | | |
| | ntpPort | VARCHAR(5) | No | | |

**Indices**

| Index Name | Columns | Primary | Unique | Type | Kind | Comment |
|---|---|---|---|---|---|---|
| PRIMARY | | Yes | No | PRIMARY | | |
| idNet | | No | No | FOREIGN | | |

**Relationships**

| Relationship Name | Relationship Type | Parent Table | Child Table | Card. |
|---|---|---|---|---|
| idNet | Identifying | Net | NetServices | 1:n |

## Table Cpu

**Columns**

| Key | Column Name | Datatype | Not Null | Default | Comment |
|---|---|---|---|---|---|
| PK | idCpu | INT | Yes | | indice progressivo |
| | idNet | INT | No | | indice NET corrispondente |
| | Addr | VARCHAR(15) | No | | indirizzo CPU sulla rete |
| | hostname | VARCHAR(255) | No | | hostname |
| | Type | VARCHAR(255) | No | | tipo di scheda usata |
| | Ram | INT | No | | Mbyte RAM |
| | Flash | INT | No | | Mbyte Flash Memory |
| | massMemory | INT | No | | Mbyte memoria di massa |
| | dbMmHard | INT | No | | % Hard limit occupazione spazio database |
| | dbMmSoft | INT | No | | % Soft limit occupazione spazio database |
| | dbRamLimit | INT | No | | % max occupazione RAM database |
| | ramSoft | INT | No | | % soft limit occupazione RAM tutti i programmi |
| | ramHard | INT | No | | % hard limit occupazione RAM tutti i programmi |
| | serialMax | INT | No | | velocità max porte seriali |
| | numSerial | INT | No | | numero porte seriali |
| | numI2C | INT | No | | numero bus I2C |
| | numGPIO | INT | No | | numero porte GPIO |
| | numUSB | INT | No | | numero porte USB |
| | numETH | INT | No | | numero porte ETH |
| | debugLevel | INT | No | | debug Level |

**Indices**

| Index Name | Columns | Primary | Unique | Type | Kind | Comment |
|---|---|---|---|---|---|---|
| PRIMARY | | Yes | No | PRIMARY | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| idNet | | No | No | FOREIGN | | |

**Relationships**

| Relationship Name | Relationship Type | Parent Table | Child Table | Card. |
|---|---|---|---|---|
| idNet | Non-Identifying | Net | Cpu | 1:n |
| idCpu | Non-Identifying | Cpu | Serial | 1:n |
| idCpu | Non-Identifying | Cpu | Gpio | 1:n |
| idCpu | Non-Identifying | Cpu | Messages | 1:n |
| idCpu | Non-Identifying | Cpu | StNow | 1:n |
| idCpu | Non-Identifying | Cpu | StDaily | 1:n |

## Table Serial

**Columns**

| Key | Column Name | Datatype | Not Null | Default | Comment |
|---|---|---|---|---|---|
| PK | idSerial | INT | Yes | | indice tabella |
| | idCpu | INT | No | | puntatore alla cpu |
| | type | CHAR | No | | tipo porta: (D)iretta, (U)sb |
| | protocol | INT | No | | protocollo: 232,485,422 |
| | Speed | FLOAT | No | | velocità porta in Kbps |
| | Bits | INT | No | | |
| | stopbits | INT | No | | |
| | Parity | CHAR | No | | |

**Indices**

| Index Name | Columns | Primary | Unique | Type | Kind | Comment |
|---|---|---|---|---|---|---|
| PRIMARY | | Yes | No | PRIMARY | | |
| idCpu | | No | No | FOREIGN | | |

**Relationships**

| Relationship Name | Relationship Type | Parent Table | Child Table | Card. |
|---|---|---|---|---|
| idCpu | Non-Identifying | Cpu | Serial | 1:n |

## Table Gpio

**Columns**

| Key | Column Name | Datatype | Not Null | Default | Comment |
|---|---|---|---|---|---|
| PK | idGpio | INT | Yes | | indice tabella |
| | idCpu | INT | No | | identificatore cpu |

| | | | | | |
|---|---|---|---|---|---|
| | Type | VARCHAR(2) | No | | tipo porta: (I)nput, (O)utput, (IO) InOut |
| | Special | CHAR | No | | (N)o, (P)wm,(A)dc, (C)ounter |

**Indices**

| Index Name | Columns | Primary | Unique | Type | Kind | Comment |
|---|---|---|---|---|---|---|
| PRIMARY | | Yes | No | PRIMARY | | |
| idCpu | | No | No | FOREIGN | | |

**Relationships**

| Relationship Name | Relationship Type | Parent Table | Child Table | Card. |
|---|---|---|---|---|
| idCpu | Non-Identifying | Cpu | Gpio | 1:n |

## Table Messages

**Columns**

| Key | Column Name | Datatype | Not Null | Default | Comment |
|---|---|---|---|---|---|
| PK | idMessages | INT | Yes | | |
| | idCpu | INT | No | | identificatore cpu |
| | severity | CHAR | No | | Fatal, Error, Warning, Informational |
| | msgCode | INT | No | | codice messaggio |
| | msgText | VARCHAR(255) | No | | testo messaggio |
| | tstamp | TIMESTAMP | No | | timestamp |

**Indices**

| Index Name | Columns | Primary | Unique | Type | Kind | Comment |
|---|---|---|---|---|---|---|
| PRIMARY | | Yes | No | PRIMARY | | |
| idCpu | | No | No | FOREIGN | | |

**Relationships**

| Relationship Name | Relationship Type | Parent Table | Child Table | Card. |
|---|---|---|---|---|
| idCpu | Non-Identifying | Cpu | Messages | 1:n |

## Table St

**Columns**

| Key | Column Name | Datatype | Not Null | Default | Comment |
|---|---|---|---|---|---|

| PK | idSt | INT | Yes | | indice |
|---|---|---|---|---|---|
| | name | VARCHAR(255) | No | | nome inseguitore |
| | idCpu | INT | No | | puntatore alla cpu centralina |
| | debugLevel | INT | No | | debug level |
| | idIndexAz | INT | No | | puntatore alla configurazione indexer azimut |
| | idIndexEl | INT | No | | puntatore alla configurazione indexer elevazione |
| | gearAz | FLOAT | No | | rapporto riduzione fisso asse AZ |
| | gearEl | FLOAT | No | | rapporto riduzione fisso asse EL |
| | elGearType | INT | No | | 0 = VSF(z fisso), 1 = martinetto (z variabile) |
| | azMin | FLOAT | No | | angolo AZ minimo in gradi (Nord = 0°) |
| | azMax | FLOAT | No | | angolo AZ massimo in gradi (Nord = 0°) |
| | elMin | FLOAT | No | | angolo EL minimo in gradi (Horiz = 0°) |
| | elMax | FLOAT | No | | angolo EL massimo in gradi (Horiz = 0°) |
| | homeAz | FLOAT | No | | angolo AZ posizione HOME |
| | homeEl | FLOAT | No | | angolo EL posizione HOME |
| | parkAz | FLOAT | No | | angolo AZ posizione SNOW |
| | parcel | FLOAT | No | | angolo EL posizione SNOW |
| | serviceAz | FLOAT | No | | angolo AZ posizione SERVICE |
| | serviceEl | FLOAT | No | | angolo EL posizione |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | SERVICE |
| | addrAz | CHAR | No | | indirizzo RS-422 indexer asse AZ |
| | addrEl | CHAR | No | | indirizzo RS-422 indexer asse EL |
| | Rotation | FLOAT | No | | Angolo orario piano normale all'asse di rotazione, da Sud |
| | Tilt | FLOAT | No | | Inclinazione sull'orizzonte del piano normale all'asse di rotazione |
| | I2CBusST | INT | No | | Indice [0..15] del bus I2C che controlla l'inseguitore |
| | I2CBusPV | INT | No | | Indice [0..15] del bus I2C che controlla il ricevitore |

**Indices**

| Index Name | Columns | Primary | Unique | Type | Kind | Comment |
|---|---|---|---|---|---|---|
| PRIMARY | | Yes | No | PRIMARY | | |
| idIndexAz | | No | No | FOREIGN | | |
| idIndexEl | | No | No | FOREIGN | | |

**Relationships**

| Relationship Name | Relationship Type | Parent Table | Child Table | Card. |
|---|---|---|---|---|
| idIndexAz | Non-Identifying | IndexCfg | St | 1:n |
| idIndexEl | Non-Identifying | IndexCfg | St | 1:n |
| key(st) | Non-Identifying | St | Power | 1:n |
| idSt | Non-Identifying | St | StepAngle | 1:n |
| idSt | Non-Identifying | St | StNow | 1:n |

## Table IndexCfg

**Columns**

| Key | Column Name | Datatype | Not Null | Default | Comment |
|---|---|---|---|---|---|
| PK | idIdx | INT | Yes | | |
| | config | VARCHAR(255) | Yes | | nome configurazione |
| | name | VARCHAR(255) | No | | nome indexer/driver |

| | | | | | |
|---|---|---|---|---|---|
| | currHold | INT | No | | Hold Peak current % |
| | currRun | INT | No | | Run Peak current % |
| | microXstep | INT | No | | numero di microsteps per step |
| | resolution | INT | No | | 0 = Fixed resolution, 1 = Variable resolution |
| | velocIni | INT | No | | Initial velocity in microstep per secondo |
| | velocRun | INT | No | | Run (slew) velocity in microstep per secondo |
| | velocJog | INT | No | | Jog velocity in microstep per secondo |
| | k_Accel | INT | No | | acceleration slope |
| | k_Decel | INT | No | | decelaration slope |
| | limitPolar | INT | No | | limit polarity |
| | timeDelay | INT | No | | setting time delay |

**Indices**

| Index Name | Columns | Primary | Unique | Type | Kind | Comment |
|---|---|---|---|---|---|---|
| PRIMARY | | Yes | No | PRIMARY | | |

**Relationships**

| Relationship Name | Relationship Type | Parent Table | Child Table | Card. |
|---|---|---|---|---|
| idIndexAz | Non-Identifying | IndexCfg | St | 1:n |
| idIndexEl | Non-Identifying | IndexCfg | St | 1:n |

## Table StepAngle

**Columns**

| Key | Column Name | Datatype | Not Null | Default | Comment |
|---|---|---|---|---|---|
| PFK | idSt | INT | Yes | | identificatore sun tracker |
| | Axis | CHAR | No | | (A)zimut o (Elevazione) |
| | upAngle | FLOAT | No | | angolo incrementale |

| | | | | | |
|---|---|---|---|---|---|
| | noUpStep | INT | No | | numero di passi corrispondente |
| | upStep_1 | INT | No | | numero di passi corrispondente all'incremento di 1 grado |
| | downAngle | FLOAT | No | | angolo decrementale |
| | noDownStep | INT | No | | numero di passi corrispondente al decremento |
| | downStep_1 | INT | No | | numero di passi corrispondente al decremento di 1 grado |

**Indices**

| Index Name | Columns | Primary | Unique | Type | Kind | Comment |
|---|---|---|---|---|---|---|
| PRIMARY | | Yes | No | PRIMARY | | |
| idSt | | No | No | FOREIGN | | |

**Relationships**

| Relationship Name | Relationship Type | Parent Table | Child Table | Card. |
|---|---|---|---|---|
| idSt | Identifying | St | StepAngle | 1:n |

**Table StNow**

**Columns**

| Key | Column Name | Datatype | Not Null | Default | Comment |
|---|---|---|---|---|---|
| PFK | idSt | INT | Yes | | identificatore solar tracker |
| | idCpu | INT | No | | identificatore cpu |
| | mode | CHAR | No | | (M)anual, (A)uto |
| | setup | CHAR | No | | setup attivo Y/N |
| | status | INT | No | | stato inseguitore on(1) / off (0) |
| | moving | CHAR | No | | (R)eady, (M)oving, (H)oming, (P)arking, (S)ervice, (T)uning |
| | position | CHAR | No | | (T)racking, (H)ome, (P)ark, (S)ervice, (M)anual |

| | | | | | |
|---|---|---|---|---|---|
| | angleAz | FLOAT | No | | angolo az |
| | angleEl | FLOAT | No | | angolo el |
| | Kwe | FLOAT | No | | potenza istantanea elettrica |
| | Kwt | FLOAT | No | | potenza istantanea termica |
| | electricE | FLOAT | No | | efficienza elettrica istantanea |
| | thermalE | FLOAT | No | | efficienza termica istantanea |
| | totalE | FLOAT | No | | efficienza totale |
| | tempPvIn | FLOAT | No | | temperatura ingresso moduli FV |
| | tempPvOut | FLOAT | No | | temperatura uscita moduli FV |
| | tempUsrIn | FLOAT | No | | temperatura ingresso scambiatore utente |
| | tempUsrOut | FLOAT | No | | temperatura uscita scambiatore utente |
| | flowPlant | FLOAT | No | | flusso liquido raffreddamento, litri/minuto |
| | flowUsr | FLOAT | No | | flusso scambiatore utente, litri/minuto |
| | dcVoltBatt | INT | No | | voltaggio batterie, DC |
| | acVoltBatt | INT | No | | voltaggio batterie, AC |
| | errorCodes | INT | No | | 0 se non ho errori,altrimenti somma codici errore |
| | errorText | VARCHAR(255) | No | | stringa esplicativa codici |
| | reqAction | VARCHAR(255) | No | | azioni richieste dall'utente o richieste in automatico |
| | Tstamp | TIMESTAMP | No | | timestamp tabella |

**Indices**

| Index Name | Columns | Primary | Unique | Type | Kind | Comment |
|---|---|---|---|---|---|---|
| PRIMARY | | Yes | No | PRIMARY | | |
| idSt | | No | No | FOREIGN | | |
| idCpu | | No | No | FOREIGN | | |

**Relationships**

| Relationship Name | Relationship Type | Parent Table | Child Table | Card. |
|---|---|---|---|---|
| idSt | Identifying | St | StNow | 1:n |
| idCpu | Non-Identifying | Cpu | StNow | 1:n |

## Table StDaily

**Columns**

| Key | Column Name | Datatype | Not Null | Default | Comment |
|---|---|---|---|---|---|
| PK | Date | DATE | Yes | | data registrazione tupla, chiave primaria, una al giorno |
| | idCpu | INT | No | | identificatore cpu |
| | Kwe | FLOAT | No | | kw elettrici prodotti nella giornata |
| | Kwt | FLOAT | No | | kw termici prodotti nella giornata |
| | kwePeak | FLOAT | No | | potenza elettrica istantanea, valore max |
| | kwtPeak | FLOAT | No | | potenza termica istantanea, valore max |
| | kwtUser | FLOAT | No | | kw termici effettivamente prelevati dall'utente |
| | tempInMin | FLOAT | No | | temperatura fluido ingresso moduli FV, min |
| | tempInMax | FLOAT | No | | temperatura fluido ingresso moduli FV, max |
| | tempInAvg | FLOAT | No | | temperatura |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | fluido ingresso moduli FV, valore medio |
| | tempOutMin | FLOAT | No | | temperatura fluido uscita moduli FV, min |
| | tempOutMax | FLOAT | No | | temperatura fluido ingresso moduli FV, max |
| | tempOutAvg | FLOAT | No | | temperatura fluido ingresso moduli FV, valore medio |
| | tExcInMin | FLOAT | No | | temperatura ingresso scambiatore utente, min |
| | tExcInMax | FLOAT | No | | temperatura ingresso scambiatore utente, max |
| | tExcInAvg | FLOAT | No | | temperatura ingresso scambiatore utente, valore medio |
| | tExcOutMin | FLOAT | No | | temperatura uscita scambiatore utente, min |
| | tExcOutMax | FLOAT | No | | temperatura uscita scambiatore utente, max |
| | tExcOutAvg | FLOAT | No | | temperatura uscita scambiatore utente, valore medio |

**Indices**

| Index Name | Columns | Primary | Unique | Type | Kind | Comment |
|---|---|---|---|---|---|---|
| PRIMARY | | Yes | No | PRIMARY | | |
| idCpu | | No | No | FOREIGN | | |

**Relationships**

| Relationship Name | Relationship Type | Parent Table | Child Table | Card. |
|---|---|---|---|---|
| idCpu | Non-Identifying | Cpu | StDaily | 1:n |

# Bibliografia e riferimenti


1. http://www.mysql.it/
2. http://www.postgresql.org/
3. http://www.sqlite.org/
4. http://www.mysql.it/
5. http://it.wikipedia.org/wiki/MSQL
6. http://www.postgresql.org/about/
7. http://www.sqlite.org/mostdeployed.html
8. http://sqlite.org/speed.html
9. http://java.sun.com/products/jdbc
10. Using SQLite on a Network http://www.sqlite.org/cvstrac/wiki?p=SqliteNetwork
11. http://www.realsoftware.com/realsqlserver/
12. http://sqliteserver.xhost.ro/index.html
13. http://users.iol.it/irwin/
14. http://www.oneledger.co.uk/sql4sockets.html
15. http://www.ch-werner.de/sqliteodbc/
16. Ylonen, T. C. Lonvick, Ed., "The Secure Shell (SSH) Authentication Protocol", RFC 4252, Gennaio 2006
17. Ylonen, T. C. Lonvick, Ed., "The Secure Shell (SSH) Protocol Architecture", RFC 4251, Gennaio 2006
18. A. Freier, P. Karlton P. Kocher, ”The SSL Protocol Version 3.0", Marzo 1996
19. http://www.openssh.com/
20. http://www.jcraft.com/jsch/
21. Monitors: an operating system structuring concept, C. A. R. Hoare - Communications of the ACM, v.17 n.10, p. 549-557, Oct. 1974