

**NAPOLLON**



# **NApollon**

## **PROGETTO DI BASI DI DATI**

**Alberto Bonventre & Salvatore Bosco & Ernesto Cifuni**

NApollon

ALBERTO BONVENTRE, SALVATORE BOSCO, ERNESTO CIFUNI

December 2024

# Contents

<b>1 Creazione DB</b>	<b>3</b>
1.1 Introduzione . . . . .	3
1.2 Progettazione Concettuale . . . . .	3
1.2.1 Modello E/R portante . . . . .	3
1.2.2 Modello E/R completo . . . . .	4
1.2.3 Trasformazione per progettazione Logica . . . . .	6
1.3 Progettazione Logica . . . . .	7
1.4 Progettazione Fisica . . . . .	8
<b>2 Ottimizzazione DB</b>	<b>12</b>
2.1 Indici . . . . .	12
2.2 Gestione della concorrenza . . . . .	12
2.2.1 PL2 Stretto . . . . .	12
2.2.2 Lock . . . . .	13
2.3 Affidabilità e Backup . . . . .	13
2.3.1 Raid 6 . . . . .	13
2.4 Recovery . . . . .	14
2.4.1 File di log . . . . .	14
2.4.2 Ripresa a caldo e ripresa a freddo . . . . .	14
<b>3 Funzionalità</b>	<b>15</b>
3.1 Introduzione . . . . .	15
3.2 Stored Procedure . . . . .	15
3.3 Query . . . . .	18
3.4 Viste . . . . .	22
3.5 Trigger . . . . .	23
3.6 Utenti . . . . .	29
<b>4 Oracle APEX</b>	<b>31</b>
4.1 Introduzione . . . . .	31
4.2 Home . . . . .	31
4.3 Mappa Lunare Real-Time . . . . .	32
4.4 Anomalie . . . . .	32
4.5 Manutenzioni . . . . .	33

4.6 Report . . . . .	34
4.7 Dashboard . . . . .	35
4.8 Riparazioni . . . . .	36
4.9 Risorse . . . . .	36
4.10 Sensori . . . . .	37
4.11 Sicurezza . . . . .	37

# Chapter 1

## Creazione DB

### 1.1 Introduzione

Il progetto ”NApollon” nasce dall’esigenza di sviluppare un sistema informatico avanzato per la gestione e l’analisi dei dati provenienti dalle missioni di esplorazione lunare, ispirato dal contesto descritto nel corso di Basi di Dati. Questo sistema si propone di coordinare in tempo reale le attività degli equipaggi, monitorare lo stato operativo di sensori e robot autonomi, e gestire le anomalie attraverso interventi specifici. Con un focus sull’implementazione di una piattaforma affidabile e performante, il progetto combina la tradizione e l’innovazione tecnologica, unendo l’identità partenopea al fascino dell’esplorazione spaziale.

### 1.2 Progettazione Concettuale

La creazione di una base di dati parte con la progettazione preliminare di un modello concettuale che descriva la base di dati stessa, detto **Modello Entity-Relationship** o **Modello E/R**.

#### 1.2.1 Modello E/R portante

La prima fase della progettazione del modello E/R vede l’individuazione delle entità ed associazioni fondamentali, centrali all’interno dell’applicazione che svilupperemo, e dunque la creazione di un cosiddetto **Modello E/R portante**. Di seguito lo schema del modello:

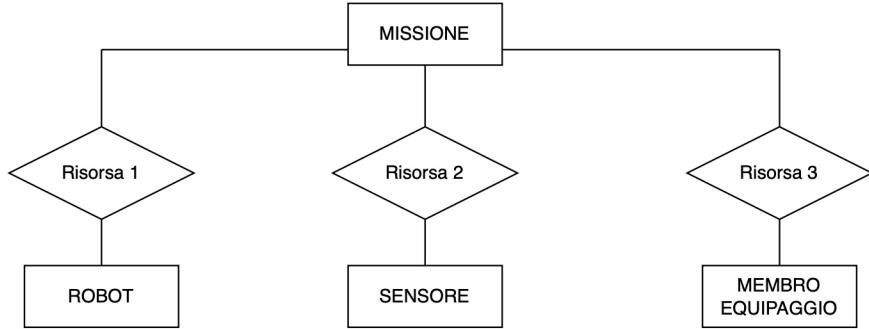


Figure 1.1: Modello E/R portante

### 1.2.2 Modello E/R completo

A partire dal modello E/R portante illustrato precedentemente, si deriva il modello concettuale completo della base di dati attraverso un'operazione di **raffinamento** del modello portante, in cui andiamo ad aggiungere tutte le altre entità e associazioni specificate dal progetto, così come i vari attributi. In questo modo, otteniamo il seguente diagramma completo:

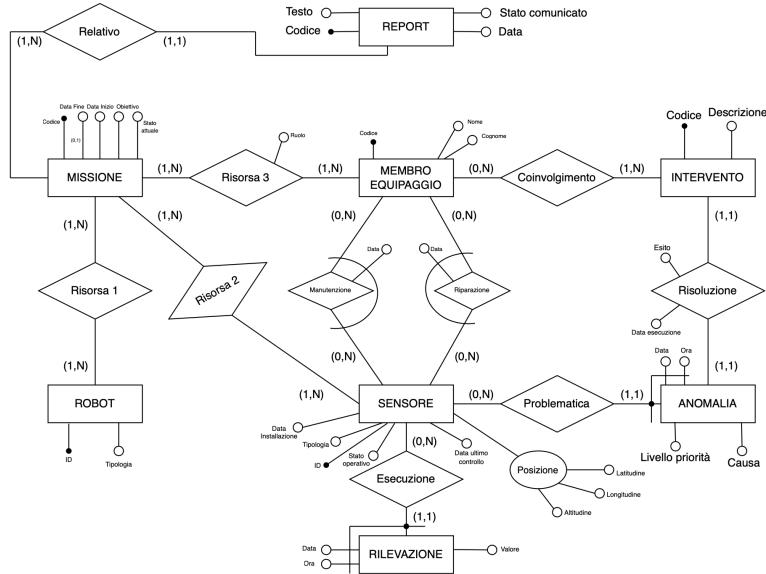


Figure 1.2: Modello E/R

## Specifiche di progettazione

Nell'individuazione del modello E/R completo, sono state effettuate varie considerazioni e ipotesi, così come varie scelte per garantire efficienza e generalità di utilizzo dell'applicazione. Tutte le varie specifiche verranno qui di seguito commentate:

- **RILEVAZIONE:** è stata utilizzata una chiave esterna che lega l'entità RILEVAZIONE all'entità SENSORE mediante la relazione ESECUZIONE sugli attributi Data e Ora. Sono sufficienti gli attributi Data e Ora poiché, insieme all'ID del sensore, riescono univocamente ad identificare una Rilevazione eseguita da un Sensore supposto che un Sensore non possa eseguire multiple rilevazioni in uno stesso istante di tempo identificato da (Data,Ora)
- **ANOMALIA:** è stata utilizzata una chiave esterna che collega l'entità ANOMALIA all'entità SENSORE mediante la relazione PROBLEMATICA sugli attributi Data e Ora. L'utilizzo di questa chiave esterna è utile per rendere più restrittivo il legame che c'è tra ANOMALIA e SENSORE posto che un Sensore non possa entrare in molteplici stati di anomalia in uno stesso istante di tempo identificato da (Data,Ora)
- **MANUTENZIONE:** è necessario un attributo all'interno della relazione MANUTENZIONE e che questo faccia parte dell'identificatore che lega le entità MEMBRO EQUIPAGGIO e SENSORE in modo tale che un MEMBRO EQUIPAGGIO possa eseguire molteplici manutenzioni su uno stesso SENSORE utilizzando come discriminante la Data in cui il MEMBRO EQUIPAGGIO esegue la MANUTENZIONE
- **RIPARAZIONE:** è necessario un attributo all'interno della relazione RIPARAZIONE e che questo faccia parte dell'identificatore che lega le entità MEMBRO EQUIPAGGIO e SENSORE in modo tale che un MEMBRO EQUIPAGGIO possa eseguire molteplici riparazioni su uno stesso SENSORE utilizzando come discriminante la Data in cui il MEMBRO EQUIPAGGIO esegue la RIPARAZIONE
- **REPORT:** l'entità REPORT è legata esclusivamente all'entità MISSIONE mediante la relazione RELATIVO per evitare LOOP. Contestualmente a ciò non è necessario mantenere le informazioni sul MEMBRO EQUIPAGGIO che ha stilato tale REPORT
- **MISSIONE-REPORT:** non è stata scelta una cardinalità 0-N dal lato dell'entità MISSIONE ma 1-N poiché supposto che all'atto di Inizio di una MISSIONE venga generato un REPORT che sottolinei che la MISSIONE sia effettivamente iniziata; conseguentemente Stato Comunicato in REPORT sarà identificato con 'Iniziata'.
- **SENSORE-RILEVAZIONI:** è stata scelta una cardinalità minima pari a 0 dal lato dell'entità SENSORE poiché non è necessario che un SEN-

SORE all'atto della sua installazione (identificata dall'attributo 'Data Installazione') esegua una RILEVAZIONE.

- **MISSIONE-SENSORE:** è stata scelta una cardinalità di tipo 'Molti a Molti' per rappresentare il fatto che un SENSORE possa essere utilizzato anche in multiple missioni: l'entità SENSORE rappresenta il SENSORE in sé e non indica il suo impiego nella MISSIONE di riferimento; Inoltre non abbiamo aggiunto un timestamp (sia in RISORSE\_2 che nelle altre) perché una singola risorsa può essere associata ad una singola missione una sola volta, perché banalmente dopo un tot tempo essa finisce
- **MISSIONE-MEMBRO EQUIPAGGIO:** è stata scelta una cardinalità di tipo 'Molti a Molti' per rappresentare il fatto che un MEMBRO EQUIPAGGIO possa partecipare anche a multiple missioni. La scelta di inserire l'attributo Ruolo nella relazione RISORSA 3 è utile per evitare ridondanza di informazioni: in questo modo un MEMBRO EQUIPAGGIO può partecipare a diverse missioni anche con ruoli differenti essendo che il ruolo non dipende dal MEMBRO EQUIPAGGIO ma dalla MISSIONE a cui il MEMBRO EQUIPAGGIO partecipa
- **MISSIONE-ROBOT (N-N):** è stata scelta una cardinalità di tipo 'Molti a Molti' per rappresentare il fatto che un ROBOT possa essere utilizzato anche in multiple missioni: l'entità ROBOT rappresenta il ROBOT in sé e non indica il suo impiego nella MISSIONE di riferimento; conseguentemente non è necessario utilizzare attributi aggiuntivi come identificatori all'interno della relazione RISORSA 1

### 1.2.3 Trasformazione per progettazione Logica

Prima di passare alla progettazione logica del database, il modello Entità-Relazione (ER) deve essere trasformato in una forma che possa essere facilmente tradotta in un sistema di gestione di database (DBMS). Alcuni elementi del modello ER non sono direttamente traducibili in tabelle o relazioni nel database relazionale, quindi è necessario effettuare delle modifiche per rendere il modello compatibile con la progettazione logica. Di seguito il diagramma trasformato:

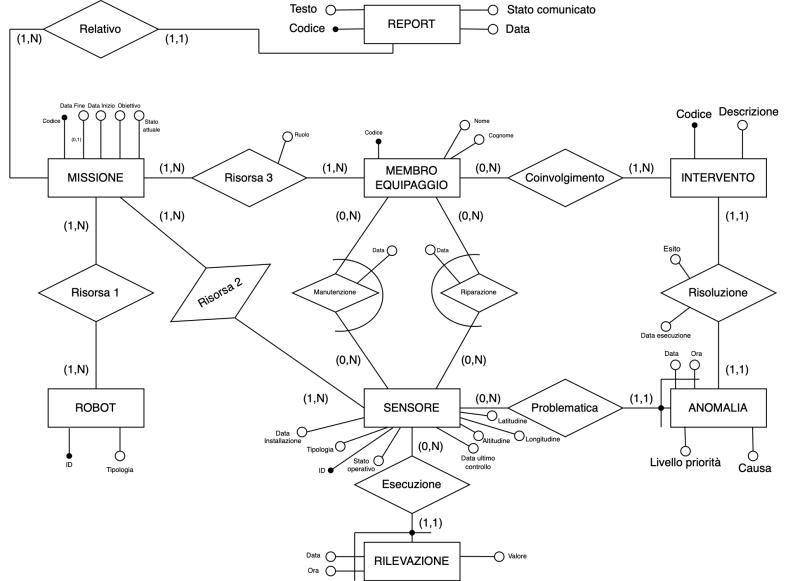


Figure 1.3: Modello E/R

### 1.3 Progettazione Logica

Nella traduzione logica del modello Entità-Relazione (ER) in un modello relazionale, abbiamo seguito delle regole standard che sono ampiamente accettate nella progettazione dei database. Si elenca di seguito il risultato di tale operazione:

```

MISSIONI(Codice, Data Fine, Data Inizio, Obbiettivo, Stato attuale)
MEMBRI_EQUIPAGGIO(Codice, Nome, Cognome)
RISORSE_3(Missione : MISSIONI, Membro : MEMBRI_EQUIPAGGIO, Ruolo)
INTERVENTI(Codice, Descrizione)
ANOMALIE(Data, Ora, Sensore : SENSORI, Livello priorità, Causa)
SENSORI(ID, Data installazione, Data ultimo controllo, Tipologia, Stato
operativo, Latitudine, Longitudine, Altitudine)
RILEVAZIONI(Data, Ora, Sensore : SENSORI, Valore)
REPORT(Codice, Testo, Data, Stato comunicato, Missione : MISSIONI)
ROBOT(ID, Tipologia)
RISORSE_1(Missione : MISSIONI, Robot : ROBOT)
RISORSE_2(Missione : MISSIONI, Sensore : SENSORI)
MANUTENZIONI(Data, Membro : MEMBRI_EQUIPAGGIO, Sensore : SENSORI)
RIPARAZIONI(Data, Membro : MEMBRI_EQUIPAGGIO, Sensore : SENSORI)
COINVOLGIMENTI(Membro : MEMBRI_EQUIPAGGIO, Intervento : INTERVENTI)
RISOLUZIONI(Intervento : INTERVENTI, (AnomaliaData, AnomaliaOra,
AnomaliaSensore)* : ANOMALIE, Esito, DataEsecuzione)
    
```

Figure 1.4: Progettazione Logica

### Specifiche di progettazione

1. Per le relazioni binarie 1 a N è stata scelta la soluzione B vista a lezione
2. Per le relazioni binarie N a N c'è un'unica soluzione
3. Per le relazioni binarie 1 a 1 è stata scelta la soluzione A vista a lezione

**Nota:** In RISOLUZIONI la PK poteva essere data a una delle due indipendentemente. Scelta una l'altra deve essere unique (\*) vincolo di unicità

## 1.4 Progettazione Fisica

L'ultima fase di progettazione rimanente riguarda la progettazione fisica, che si concentra sulla definizione e implementazione della struttura fisica del database<sup>1</sup>, ossia su come i dati vengono memorizzati e organizzati all'interno del sistema. Tale implementazione è stata realizzata mediante uno script SQL apposito, qui mostrato a corredo.

```
-- Tabella MISSIONI
CREATE TABLE MISSIONI (
    Codice NUMBER PRIMARY KEY,
    Data_Fine DATE,
    Data_Inizio DATE,
    Obiettivo VARCHAR2(255),
    Stato_Attuale VARCHAR2(100)
);

-- Tabella MEMBRI_EQUIPAGGIO
CREATE TABLE MEMBRI_EQUIPAGGIO (
    Codice NUMBER PRIMARY KEY,
    Nome VARCHAR2(100),
    Cognome VARCHAR2(100)
);
```

```
-- Tabella INTERVENTI
CREATE TABLE INTERVENTI (
    Codice NUMBER PRIMARY KEY,
    Descrizione VARCHAR2(255)
);

-- Tabella ANOMALIE
CREATE TABLE ANOMALIE (
    Data DATE,
    Ora VARCHAR2(5),
    Sensore NUMBER,
    Livello_Priorita NUMBER,
    Causa VARCHAR2(255),
    PRIMARY KEY (Data, Ora, Sensore)
);
```

<sup>1</sup> Nota: Da qui in poi, per semplicità, ci riferiremo al termine Database con l'acronimo DB

```
-- Tabella SENSORI
CREATE TABLE SENSORI (
    ID NUMBER PRIMARY KEY,
    Data_Installazione DATE,
    Data_Ultimo_Controllo DATE,
    Tipologia VARCHAR2(100),
    Stato_Operativo VARCHAR2(100)
    CHECK (Stato_Operativo='Operativo'
        OR Stato_Operativo='Non operativo'
        OR Stato_Operativo='In manutenzione'
        OR Stato_Operativo='Malfunzionante'),
    Latitudine FLOAT,
    Longitudine FLOAT,
    Altitudine FLOAT
);
```

```
-- Tabella RILEVAZIONI
CREATE TABLE RILEVAZIONI (
    Data DATE,
    Ora VARCHAR2(5),
    Sensore NUMBER,
    Valore FLOAT,
    PRIMARY KEY (Data, Ora, Sensore)
);

-- Tabella REPORT
CREATE TABLE REPORT (
    Codice NUMBER PRIMARY KEY,
    Testo CLOB,
    Data DATE,
    Stato_Comunicato VARCHAR2(100),
    Missione NUMBER
);
```

```
-- Tabella ROBOT
CREATE TABLE ROBOT (
    ID NUMBER PRIMARY KEY,
    Tipologia VARCHAR2(100)
);

-- Tabella RISORSE_1
CREATE TABLE RISORSE_1 (
    Missione NUMBER,
    Robot NUMBER,
    PRIMARY KEY (Missione, Robot)
);
```

```
-- Tabella RISORSE_2
CREATE TABLE RISORSE_2 (
    Missione NUMBER,
    Sensore NUMBER,
    PRIMARY KEY (Missione, Sensore)
);

-- Tabella RISORSE_3
CREATE TABLE RISORSE_3 (
    Missione NUMBER,
    Membro NUMBER,
    Ruolo VARCHAR2(100),
    PRIMARY KEY (Missione, Membro)
);
```

```
-- Tabella MANUTENZIONI
CREATE TABLE MANUTENZIONI (
    Data DATE,
    Membro NUMBER,
    Sensore NUMBER,
    PRIMARY KEY (Data, Membro, Missione, Sensore)
);

-- Tabella RIPARAZIONI
CREATE TABLE RIPARAZIONI (
    Data DATE,
    Membro NUMBER,
    Sensore NUMBER,
    PRIMARY KEY (Data, Membro, Sensore)
);
```

```
-- Tabella COINVOLGIMENTI
CREATE TABLE COINVOLGIMENTI (
    Membro NUMBER,
    Intervento NUMBER,
    PRIMARY KEY (Membro, Intervento)
);

-- Tabella RISOLUZIONI
CREATE TABLE RISOLUZIONI (
    Esito VARCHAR2(5) CHECK (Esito='TRUE' OR Esito='FALSE'),
    DataEsecuzione DATE,
    Intervento NUMBER,
    AnomaliaData DATE,
    AnomaliaOra VARCHAR2(5),
    AnomaliaSenso NUMBER,
    PRIMARY KEY (Intervento),
    UNIQUE (AnomaliaData, AnomaliaOra, AnomaliaSenso)
);
```

```
-- Tabella MANUTENZIONI
CREATE TABLE MANUTENZIONI (
    Data DATE,
    Membro NUMBER,
    Sensore NUMBER,
    PRIMARY KEY (Data, Membro, Missione, Sensore)
);

-- Tabella RIPARAZIONI
CREATE TABLE RIPARAZIONI (
    Data DATE,
    Membro NUMBER,
    Sensore NUMBER,
    PRIMARY KEY (Data, Membro, Sensore)
);
```

```
-- Tabella COINVOLGIMENTI
CREATE TABLE COINVOLGIMENTI (
    Membro NUMBER,
    Intervento NUMBER,
    PRIMARY KEY (Membro, Intervento)
);

-- Tabella RISOLUZIONI
CREATE TABLE RISOLUZIONI (
    Esito VARCHAR2(5) CHECK (Esito='TRUE' OR Esito='FALSE'),
    DataEsecuzione DATE,
    Intervento NUMBER,
    AnomaliaData DATE,
    AnomaliaOra VARCHAR2(5),
    AnomaliaSensore NUMBER,
    PRIMARY KEY (Intervento),
    UNIQUE (AnomaliaData, AnomaliaOra, AnomaliaSensore)
);
```

Per ultimare la trattazione di questo paragrafo, si elencano anche i vincoli di chiave esterna, che permettono la referenza tra le varie tabelle appena create. Nello sviluppo della struttura fisica del DB si è preferito riservare a queste una sezione a parte, sia per rendere lo script più pulito e leggibile , ma anche per garantire un corretto funzionamento dell'applicativo, infatti l'inserimento di chiavi esterne segue un ordine lessicografico secondo cui risulta impossibile referenziare tabelle che non sono state ancora create (che vengono cioè create in un momento successivo all'interno dello script).

NOTA: nella creazione della tabella MANUTENZIONI, la primary key è (Data, Membro, Sensore) e non quella indicata nella foto

```
-- Chiave esterna per RISORSE_3
ALTER TABLE RISORSE_3
ADD CONSTRAINT FK_Missione_Risorse_3 FOREIGN KEY (Missione) REFERENCES MISSIONI(Codice)
ON DELETE CASCADE;
ALTER TABLE RISORSE_3
ADD CONSTRAINT FK_Membro_Risorse_3 FOREIGN KEY (Membro) REFERENCES MEMBRI_EQUIPAGGIO(Codice)
ON DELETE CASCADE;

-- Chiave esterna per ANOMALIE
ALTER TABLE ANOMALIE
ADD CONSTRAINT FK_Sensore_Anomalie FOREIGN KEY (Sensore) REFERENCES SENSORI(ID)
ON DELETE CASCADE;

-- Chiave esterna per RILEVAZIONI
ALTER TABLE RILEVAZIONI
ADD CONSTRAINT FK_Sensore_Rilevazioni FOREIGN KEY (Sensore) REFERENCES SENSORI(ID)
ON DELETE CASCADE;
```

```
-- Chiave esterna per REPORT
ALTER TABLE REPORT
ADD CONSTRAINT FK_Missione_Report FOREIGN KEY (Missione) REFERENCES MISSIONI(Codice)
ON DELETE SET NULL;

-- Chiave esterna per RISORSE_1
ALTER TABLE RISORSE_1
ADD CONSTRAINT FK_Missione_Risorse_1 FOREIGN KEY (Missione) REFERENCES MISSIONI(Codice)
ON DELETE CASCADE;
ALTER TABLE RISORSE_1
ADD CONSTRAINT FK_Robot_Risorse_1 FOREIGN KEY (Robot) REFERENCES ROBOT(ID)
ON DELETE CASCADE;
```

```
-- Chiave esterna per RISORSE_2
ALTER TABLE RISORSE_2
ADD CONSTRAINT FK_Missione_Risorse_2 FOREIGN KEY (Missione) REFERENCES MISSIONI(Codice)
ON DELETE CASCADE;
ALTER TABLE RISORSE_2
ADD CONSTRAINT FK_Sensore_Risorse_2 FOREIGN KEY (Sensore) REFERENCES SENSORI(ID)
ON DELETE CASCADE;

-- Chiave esterna per MANUTENZIONI
ALTER TABLE MANUTENZIONI
ADD CONSTRAINT FK_Membro_Manutenzioni FOREIGN KEY (Membro) REFERENCES MEMBRI_EQUIPAGGIO(Codice)
ON DELETE CASCADE;
ALTER TABLE MANUTENZIONI
ADD CONSTRAINT FK_Sensore_Manutenzioni FOREIGN KEY (Sensore) REFERENCES SENSORI(ID)
ON DELETE CASCADE;
```

```
-- Chiave esterna per RIPARAZIONI
ALTER TABLE RIPARAZIONI
ADD CONSTRAINT FK_Membro_Riparazioni FOREIGN KEY (Membro) REFERENCES MEMBRI_EQUIPAGGIO(Codice)
ON DELETE CASCADE;
ALTER TABLE RIPARAZIONI
ADD CONSTRAINT FK_Sensore_Riparazioni FOREIGN KEY (Sensore) REFERENCES SENSORI(ID)
ON DELETE CASCADE;

-- Chiave esterna per COINVOLGIMENTI
ALTER TABLE COINVOLGIMENTI
ADD CONSTRAINT FK_Membro_Coinvolgimenti FOREIGN KEY (Membro) REFERENCES MEMBRI_EQUIPAGGIO(Codice)
ON DELETE CASCADE;
ALTER TABLE COINVOLGIMENTI
ADD CONSTRAINT FK_Intervento_Coinvolgimenti FOREIGN KEY (Intervento) REFERENCES INTERVENTI(Codice)
ON DELETE CASCADE;
```

```
-- Chiave esterna per RISOLUZIONI
ALTER TABLE RISOLUZIONI
ADD CONSTRAINT FK_Intervento_Risoluzioni FOREIGN KEY (Intervento) REFERENCES INTERVENTI(Codice)
ON DELETE CASCADE;
ALTER TABLE RISOLUZIONI
ADD CONSTRAINT FK_Anomalia_Risoluzioni FOREIGN KEY (AnomaliaData, AnomaliaOra, AnomaliaSensore)
REFERENCES ANOMALIE(Data, Ora, Sensore)
ON DELETE CASCADE;
```

**NOTA:** In Oracle la clausola ON UPDATE non funziona e va implementata esplicitamente mediante l'uso di trigger<sup>3.5</sup> (qui trattati al capitolo 3)

# Chapter 2

## Ottimizzazione DB

### 2.1 Indici

Nel processo di progettazione del database, abbiamo creato determinati indici con l'obiettivo di velocizzare le query sulle tabelle. Sebbene il sistema attuale non richieda frequenti interrogazioni su determinati campi, abbiamo ipotizzato quelli che secondo noi verranno utilizzati frequentemente in query successive. L'adozione di questo approccio proattivo ci permette di migliorare la performance complessiva del sistema e di ottimizzare l'esperienza utente, riducendo i tempi di attesa e migliorando l'efficienza del database man mano che il carico di lavoro cresce.

```
CREATE INDEX Data_Report ON REPORT(DATA);
CREATE INDEX Stato_Missioni ON MISSIONI(STATO_ATTUALE);
CREATE INDEX Tipologia_Sensori ON SENSORI(TIPOLOGIA);
CREATE INDEX Stato_Sensori ON SENSORI(STATO_OPERATIVO);
CREATE INDEX Tipologia_Stato_Sensori ON SENSORI(TIPOLOGIA, STATO_OPERATIVO);
```

Figure 2.1: SQL degli indici in Apex

### 2.2 Gestione della concorrenza

#### 2.2.1 PL2 Stretto

Il metodo di gestione della concorrenza PL2 stretto (o Strict Two-Phase Locking, 2PL stretto) viene scelto in molti sistemi transazionali per garantire la consistenza dei dati e prevenire anomalie. Questo approccio estende il Two-Phase Locking (2PL) tradizionale aggiungendo un vincolo: i lock (blocchi) acquisiti da una transazione non vengono rilasciati fino a quando la transazione non termina (commit o abort).

Perchè sceglierlo?

- Garanzia di serializzabilità: PL2 stretto garantisce che l'ordine delle transazioni sia equivalente a un'esecuzione sequenziale (seriale), evitando conflitti nei dati condivisi
- Previene gli stati inconsistenti: i lock sono mantenuti fino al termine della transazione, nessun'altra può accedere ai dati in uno stato intermedio o non ancora confermato. (ad esempio evita problemi di lettura sporca)

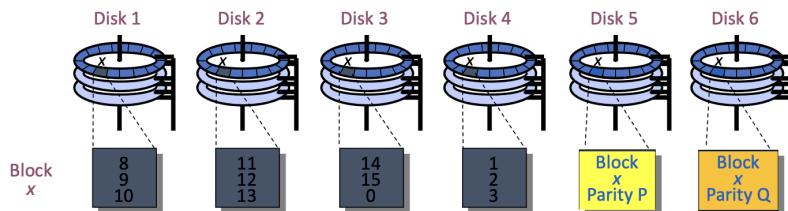
### 2.2.2 Lock

Ogni oggetto della base di dati è protetto da un apposito lock (blocco). Ogni transazione che vuole effettuare una lettura di un oggetto deve richiederne l'autorizzazione attraverso un'operazione di `read_lock(x)` e, solo una volta acquisito il lock, può eseguire la lettura. Il lock in lettura su un dato oggetto può essere condiviso da più transazioni. Ogni transazione che vuole effettuare una scrittura su un oggetto deve richiederne l'autorizzazione attraverso un'operazione di `write_lock(x)` e, solo una volta acquisito il lock, può eseguire la scrittura. Il lock in scrittura su un dato oggetto è esclusivo, ovvero può essere acquisito da una sola transazione per volta.

## 2.3 Affidabilità e Backup

### 2.3.1 Raid 6

- Ridondanza dei dati con doppio bit di parità per blocco distribuiti su più dischi
- Tra le soluzioni presentate RAID-6 è quella che ha il miglior livello di affidabilità
- In lettura la capacità di trasferimento è simile al RAID 0, la scrittura è penalizzata (è il peggioro di tutti per le prestazioni in scrittura)
- Dischi richiesti =  $N+2$
- A differenza di altri tipi di RAID, come il 3 (il quale ha striping a livello di bit), il RAID 6 ha uno striping a livello di blocco il che lo rende particolarmente adatto alle basi di dati per massimizzare il throughput di richieste servite



## 2.4 Recovery

### 2.4.1 File di log

Un file di log è come un diario dove il sistema registra tutte le operazioni eseguite su un database, come modifiche, inserimenti o cancellazioni. È una traccia che serve a sapere cosa è successo, quando e come, ed'è utile per il ripristino di una base dati in caso di guasti.

```
DP, B(T1,-, -, -, -), U(T1,-,qtaP,100,90), U(T1,-,qtaC,NULL,10), C(T1,-, -, -, -),
B(T2,-, -, -, -), CK(T2), U(T2,-,qtaP,90,70), U(T1,-,qtaC,NULL,20), C(T2,-, -, -
,-), B(T3,-, -, -, -), U(T3,-,qtaP,100,90), U(T3,-,qtaC,NULL,10), C(T3,-, -, -, -),
...
```

Figure 2.2: Esempio di file log

### 2.4.2 Ripresa a caldo e ripresa a freddo

Nel caso di guasti soft (es. errori di programma, crash di sistema, caduta di tensione, ecc.) si perde il contenuto della sola memoria centrale (mentre rimangono intatte la memoria secondaria e quella stabile). In tale situazioni, viene effettuata la cosiddetta ripresa a caldo (warm restart).

Nel caso di guasti hard sui dispositivi di memoria di massa si perde la memoria centrale e quella secondaria ma non quella stabile. ). In tale situazioni, viene effettuata la cosiddetta ripresa a freddo (cold restart).

In entrambi i casi, la procedura di ripristino avviene nelle seguenti fasi (modello fail-stop):

- si forza l'arresto completo delle transazioni attive sul sistema di basi di dati;
- viene ripristinato il corretto corretto funzionamento del sistema operativo;
- viene effettuata la procedura di ripristino.

# Chapter 3

## Funzionalità

### 3.1 Introduzione

L'implementazione di stored procedure, query, viste e trigger in SQL è un passaggio fondamentale per garantire l'efficienza, l'affidabilità e l'automazione delle operazioni all'interno di un sistema basato su database. Questi elementi sono stati progettati per soddisfare le necessità specifiche delle operazioni richieste, migliorando la gestione dei dati e l'esecuzione delle attività di business. Ad esempio, un trigger potrebbe essere utilizzato per monitorare la tabella delle anomalie e, quando viene registrata una nuova anomalia, inserire automaticamente la data di esecuzione dell'intervento in una tabella separata, migliorando l'efficienza operativa e garantendo che i dati siano sempre coerenti.

### 3.2 Stored Procedure

Le stored procedure sono blocchi di codice SQL che vengono eseguiti dal database per eseguire operazioni complesse o ripetitive. In questo caso, abbiamo implementato stored procedure per automatizzare operazioni frequenti, come l'inserimento, l'aggiornamento o la cancellazione di record, minimizzando così il rischio di errori manuali e migliorando le prestazioni delle operazioni.

NOTA: La stored numero 4 perde di significato se è attivo il trigger numero 11

```
-- 1 Creazione di una nuova missione
CREATE OR REPLACE PROCEDURE Nuova_Missione(
    codiceM MISSIONI.CODICE%TYPE,
    datafineM MISSIONI.DATA_FINE%TYPE,
    datainizioM MISSIONI.DATA_INIZIO%TYPE,
    obiettivoM MISSIONI.OBIETTIVO%TYPE,
    statoattuale MISSIONI.STATO_ATTUALE%TYPE
)
AS
BEGIN
    INSERT INTO MISSIONI VALUES (codiceM, datafineM, datainizioM, obiettivoM, statoattuale);
END;
BEGIN
    Nuova_Missione(69, TO_DATE('18-OCT-2024', 'DD-MON-YYYY'), TO_DATE('05-MAY-2023', 'DD-MON-YYYY'), 'Estabilire una base lunare permanente', 'In corso');
END;
```

```
-- 2 Aggiornamento dello stato della missione
CREATE OR REPLACE PROCEDURE Aggiorna_Stato_Missione(
    codiceM MISSIONI.CODICE%TYPE,
    nuovostato MISSIONI.STATO_ATTUALE%TYPE
)
AS
BEGIN
    UPDATE MISSIONI
    SET STATO_ATTUALE = nuovostato
    WHERE CODICE = codiceM;
END;
BEGIN
    Aggiorna_Stato_Missione(69,'Finito');
END;
```

```
-- 3 Elenco delle missioni attive (stato = In corso/Iniziata)
CREATE OR REPLACE PROCEDURE Stampa_Missioni_Attive
AS
CURSOR missioni_cur IS
    SELECT CODICE, STATO_ATTUALE, DATA_FINE, DATA_INIZIO, OBIETTIVO
    FROM MISSIONI
    WHERE STATO_ATTUALE = 'In corso' OR STATO_ATTUALE = 'Iniziata';

    v_cod MISSIONI.CODICE%TYPE;
    datafineM MISSIONI.DATA_FINE%TYPE;
    datainizioM MISSIONI.DATA_INIZIO%TYPE;
    obiettivoM MISSIONI.OBIETTIVO%TYPE;
    v_stato MISSIONI.STATO_ATTUALE%TYPE;

BEGIN
    OPEN missioni_cur;
    LOOP
        FETCH missioni_cur INTO v_cod, v_stato, datafineM, datainizioM, obiettivoM;
        EXIT WHEN missioni_cur%NOTFOUND;

        DBMS_OUTPUT.PUT_LINE('Codice: ' || v_cod || ', Stato: ' || v_stato || ', Inizio: ' || datainizioM || ', Fine: ' || datafineM || ', Obiettivo: ' || obiettivoM);
    END LOOP;
    CLOSE missioni_cur;
END;
```

```
Codice: 1, Stato: In corso, Inizio: 06/01/2023, Fine: 10/10/2024, Obiettivo: Estabilire una base lunare permanente
Codice: 4, Stato: In corso, Inizio: 09/01/2025, Fine: 12/12/2026, Obiettivo: Creare un laboratorio scientifico sulla luna
Codice: 6, Stato: In corso, Inizio: 06/10/2024, Fine: 11/01/2025, Obiettivo: Esplorare i poli lunari
Codice: 10, Stato: In corso, Inizio: 04/01/2025, Fine: 08/25/2025, Obiettivo: Creare un avamposto di ricerca per la biologia lunare
Codice: 13, Stato: In corso, Inizio: 12/01/2024, Fine: 05/20/2025, Obiettivo: Sviluppare tecnologie per la propulsione lunare
Codice: 17, Stato: In corso, Inizio: 06/01/2025, Fine: 10/15/2026, Obiettivo: Instaurare collegamenti tra la Luna e Marte
Codice: 20, Stato: In corso, Inizio: 09/01/2025, Fine: 12/05/2025, Obiettivo: Esplorare le possibilità di costruire città lunari autosufficienti
Codice: 69, Stato: In corso, Inizio: 05/05/2023, Fine: 10/18/2024, Obiettivo: Etablire una base lunare permanente
Codice: 2, Stato: Iniziata, Inizio: 11/01/2024, Fine: 02/28/2025, Obiettivo: Ricercare risorse minerali sulla Luna
Codice: 7, Stato: Iniziata, Inizio: 01/15/2026, Fine: 06/30/2027, Obiettivo: Testare la terraformazione lunare
Codice: 12, Stato: Iniziata, Inizio: 09/01/2026, Fine: 01/10/2027, Obiettivo: Esplorare le grotte lunari per rifugi naturali
Codice: 15, Stato: Iniziata, Inizio: 11/01/2026, Fine: 03/01/2027, Obiettivo: Sperimentare la creazione di energia solare lunare
Codice: 19, Stato: Iniziata, Inizio: 12/15/2023, Fine: 03/30/2024, Obiettivo: Lanciare una missione per la raccolta di dati sismici lunari
```

```
-- 4 Lista anomalie la cui risoluzione è fallita
CREATE OR REPLACE PROCEDURE Risoluzioni_Fallite
AS
CURSOR anomalie_cur IS
    SELECT A.Sensore, A.Causa, A.Livello_Priorita, R.DataEsecuzione
    FROM RISOLUZIONI R JOIN ANOMALIE A ON R.AnomaliaData = A.Data AND R.AnomaliaOra = A.Ora AND R.AnomaliaSensore = A.Sensore
    WHERE R.Esito = 'FALSE';

    sensore ANOMALIE.SENSORE%TYPE;
    causa ANOMALIE.CAUSA%TYPE;
    LP ANOMALIE.LIVELLO_PRIORITA%TYPE;
    dataE RISOLUZIONI.DATAESECUIZIONE%TYPE;

BEGIN
    OPEN anomalie_cur;
    LOOP
        FETCH anomalie_cur INTO sensore, causa, LP, dataE;
        EXIT WHEN anomalie_cur%NOTFOUND;

        DBMS_OUTPUT.PUT_LINE('Sensore: ' || sensore || ', Causa_Anomalia: ' || causa || ', Livello Priorità: ' || LP || ', Data esecuzione fallita: ' || dataE);
    END LOOP;
    CLOSE anomalie_cur;
END;
```

```
Sensore: 2, Causa_Anomalia: Interferenze nel sistema di comunicazione, Livello Priorità: 2, Data esecuzione fallita: 11/18/2024
Sensore: 1, Causa_Anomalia: Perdita di pressione nella camera di equilibrio, Livello Priorità: 3, Data esecuzione fallita: 11/21/2024
Sensore: 8, Causa_Anomalia: Segnale irregolare nel radar di prossimità, Livello Priorità: 3, Data esecuzione fallita: 11/24/2024
Sensore: 1, Causa_Anomalia: Guasto del sistema di filtraggio dell'aria, Livello Priorità: 3, Data esecuzione fallita: 11/27/2024
Sensore: 4, Causa_Anomalia: Errore nei segnali del sistema di monitoraggio lunare, Livello Priorità: 2, Data esecuzione fallita: 11/30/2024
Sensore: 9, Causa_Anomalia: Interruzione nel flusso di dati geologici, Livello Priorità: 2, Data esecuzione fallita: 12/03/2024
Sensore: 10, Causa_Anomalia: Guasto nei motori del sistema robotico, Livello Priorità: 2, Data esecuzione fallita: 12/05/2024
```

```
-- 5 Inserimento report
CREATE OR REPLACE PROCEDURE Nuovo_Report(
    codiceR REPORT.CODICE%TYPE,
    testo REPORT.TESTO%TYPE,
    dataR REPORT.DATA%TYPE,
    stato REPORT.STATO_COMUNICATO%TYPE,
    missione REPORT.MISSIONE%TYPE
)
AS
    v_count NUMBER;
BEGIN
    SELECT COUNT(*)
    INTO v_count
    FROM REPORT
    WHERE CODICE = codiceR;

    IF v_count > 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Codice già esistente nel database!');
    END IF;

    INSERT INTO REPORT VALUES (codiceR, testo, dataR, stato, missione);
END;
```

```
-- 6 Lista tutti report per una data missione
CREATE OR REPLACE PROCEDURE Lista_report_per_missione(
    codiceM REPORT.MISSIONE%TYPE
)
AS
    CURSOR report_cur IS
        SELECT R.CODICE, R.TESTO, R.DATA, R.STATO_COMUNICATO
        FROM REPORT R
        WHERE R.MISSIONE = codiceM;

    codiceR REPORT.CODICE%TYPE;
    testoR REPORT.TESTO%TYPE;
    dataR REPORT.DATA%TYPE;
    statoR REPORT.STATO_COMUNICATO%TYPE;
BEGIN
    OPEN report_cur;
    LOOP
        FETCH report_cur INTO codiceR, testoR, dataR, statoR;
        EXIT WHEN report_cur%NOTFOUND;

        DBMS_OUTPUT.PUT_LINE('Codice Report: ' || codiceR || ', Testo: ' || testoR || ', Data: ' || dataR || ', Stato: ' || statoR);
    END LOOP;
    CLOSE report_cur;
END;
```

```
Codice Report: 1, Testo: Il sensore di temperatura ha registrato una lettura inferiore ai livelli di sicurezza, suggerendo una possibile avaria al sistema di riscaldamento. Dopo un'attenta analisi, si è determinato che la causa principale è un guasto parziale al modulo di riscaldamento. La missione è stata temporaneamente sospesa per le necessarie riparazioni e per garantire la sicurezza dell'equipaggio., Data: 11/01/2024, Stato: In corso
Iniziata
Codice Report: 99, Testo: is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry. Data: 12/05/2024, Stato: In corso
Statement processed.
```

### 3.3 Query

Le query SQL sono utilizzate per estrarre, manipolare e visualizzare i dati in modo efficiente.

```
--1. Visualizzare lo STATO_ATTUALE di una Missione:
SELECT Codice, Stato_Attuale
FROM MISSIONI
WHERE Codice = 1;
```

```
--2. Elenicare i membri dell'equipaggio assegnati a una missione:
SELECT R.Missione AS CODICEMISSIONE, ME.Codice AS CODICEMEMBRO, ME.Nome, ME.Cognome, R.Ruolo
FROM MEMBRI_EQUIPAGGIO ME
JOIN RISORSE_3 R
ON ME.Codice = R.Membro
WHERE R.Missione = 1;
```

CODICEMISSIONE	CODICEMEMBRO	NOME	COGNOME	RUOLO
1	1	Luca	Rossi	Comandante
1	2	Maria	Bianchi	Specialista Tecnico
1	3	Giovanni	Verdi	Ingegner delle Comunicazioni

3 rows returned in 0.05 seconds [Download](#)

```
--3. Elenicare rilevazioni di un sensore in un intervallo di tempo: //PL/SQL
Select S.ID, S.Tipologia, R.Data, R.Ora, R.Valore
FROM SENSORI S
JOIN RILEVAZIONI R
ON S.ID = R.Sensore
WHERE S.ID = 2 AND R.DATA > TO_DATE('18-OCT-2022', 'DD-MON-YYYY') AND R.DATA < TO_DATE('18-OCT-2025', 'DD-MON-YYYY');
```

ID	TIPOLOGIA	DATA	ORA	VALORE
2	Sensore di pressione	11/01/2024	08:30	1013.2

1 rows returned in 0.01 seconds [Download](#)

```
--4. Elenicare rilevazioni di un sensore in un intervallo di tempo: //PL/SQL
SELECT S.ID, S.Tipologia, R.Data, R.Ora, R.Valore
FROM SENSORI S
JOIN RILEVAZIONI R
ON S.ID = R.Sensore
WHERE S.ID = 1
AND R.DATA > TO_DATE('31/12/2023', 'DD/MM/YYYY')
AND R.DATA < TO_DATE('31/12/2099', 'DD/MM/YYYY');
```

ID	TIPOLOGIA	DATA	ORA	VALORE
1	Sensore di temperatura	11/01/2024	08:00	23.5
1	Sensore di temperatura	11/01/2024	17:30	24.1

2 rows returned in 0.02 seconds [Download](#)

```
--4.1. Elencare le anomalie che si sono scatenate fino ad una certa data su uno spec. sensore
SELECT S.ID, S.Tipologia, S.Stato_Operativo AS STATO_OPERATIVO_SENSERE, A.Data AS DATA_ANOMALIA, A.Ora, A.Livello_Priorita, A.Causa
FROM SENSORI S
JOIN ANOMALIE A
ON S.ID = A.Sensore
WHERE S.ID = 1 AND A.Data<TO_DATE('10/11/2024','DD/MM/YYYY');
```

ID	TIPOLOGIA	STATO_OPERATIVO_SENSERE	DATA_ANOMALIA	ORA	LIVELLO_PRIORITA	CAUSA
1	Sensore di temperatura	Operativo	11/01/2024	08:30	3	Fluttuazione di potenza nei pannelli solari
1	Sensore di temperatura	Operativo	11/05/2024	11:30	5	Perdita di pressione nella camera di equilibrio

2 rows returned in 0.03 seconds [Download](#)

```
--5. Controllare le anomalie attive per un sensore:
SELECT S.ID, S.Tipologia, S.Stato_Operativo AS STATO_OPERATIVO_SENSERE, A.Data AS DATA_ANOMALIA, R.DataEsecuzione, R.Esito, A.Livello_Priorita, A.Causa
FROM SENSORI S
JOIN ANOMALIE A
ON S.ID = A.Sensore
JOIN RISOLUZIONI R
ON R.Anomaliasensore = A.Sensore
WHERE S.ID = 1 AND (R.DataEsecuzione > SYSDATE OR ( R.DataEsecuzione < SYSDATE AND R.Esito = 'FALSE' ));
```

ID	TIPOLOGIA	STATO_OPERATIVO_SENSERE	DATA_ANOMALIA	DATAESECUSIONE	ESITO	LIVELLO_PRIORITA	CAUSA
1	Sensore di temperatura	Operativo	11/01/2024	11/21/2024	FALSE	5	Fluttuazione di potenza nei pannelli solari
1	Sensore di temperatura	Operativo	11/01/2024	11/21/2024	FALSE	5	Fluttuazione di potenza nei pannelli solari
1	Sensore di temperatura	Operativo	11/05/2024	11/21/2024	FALSE	5	Perdita di pressione nella camera di equilibrio
1	Sensore di temperatura	Operativo	11/05/2024	11/21/2024	FALSE	5	Perdita di pressione nella camera di equilibrio
1	Sensore di temperatura	Operativo	11/11/2024	11/21/2024	FALSE	5	Attiva Windows Guarda dal sistema di filtraggio dell'aria

```
--6.Consultare i report compilati per una missione:
SELECT R.Codice AS CODICE_REPORT, R.Testo, R.Data, R.Stato_Comunicato
FROM REPORT R
WHERE R.Missione = 19;
```

```
--7.Identificare i sensori con il maggior numero di anomalie:
SELECT S.ID AS CODICE_SENSEORE, COUNT(*) AS NUM_ANOMALIE
FROM SENSORI S
JOIN ANOMALIE A
ON S.ID = A.Sensore
GROUP BY S.ID
HAVING COUNT(*) >= ALL(SELECT COUNT(A2.Sensore)
FROM ANOMALIE A2
GROUP BY A2.Sensore
);
```

CODICE_SENSEORE	NUM_ANOMALIE
1	3
3	3
4	3
10	3

4 rows returned in 0.03 seconds    [Download](#)

Attiva Windows  
Passa a Impostazioni per attivare Windows.

```
--8.Calcolare il numero di rilevazioni effettuate per ciascun tipo di sensore in una missione:
SELECT S.Tipologia, COUNT(S.Tipologia) AS NUM_RILEVAZIONI
FROM SENSORI S
JOIN RILEVAZIONI R
ON S.ID = R.Sensore
GROUP BY S.Tipologia;
```

TIPOLOGIA	NUM_RILEVAZIONI
Sensore di temperatura	2
Sensore di pressione	1
Sensore di radiazioni	2
Sensore di ossigeno	1
Sensore di velocità vento	2

Attiva Windows  
Passa a Impostazioni per attivare Windows.

```
--9.Analizzare il tempo medio (in giorni) di risoluzione delle anomalie:
SELECT AVG( R.DataEsecuzione - A.Data) AS TEMPO_MEDIO_RISOLUZIONE
FROM RISOLUZIONI R
JOIN ANOMALIE A
ON R.AnomaliaData = A.Data AND R.AnomaliaOra = A.Ora AND R.AnomaliaSensore = A.Sensore
WHERE R.Esito = 'TRUE';
```

TEMPO_MEDIO_RISOLUZIONE
16

```
--10. Trovare i membri dell'equipaggio con il maggior numero di interventi effettuati:
SELECT C.Membro AS CODICE_MEMBRO, COUNT(*) AS NUM_INTERVENTI
FROM COINVOLGIMENTI C
JOIN MEMBRI_EQUIPAGGIO ME
ON C.Membro = ME.Codice
GROUP BY C.MEMBRO
HAVING COUNT(*) >= ALL(SELECT COUNT(C2.Membro)
FROM COINVOLGIMENTI C2
GROUP BY C2.Membro
);
```

CODICE_MEMBRO	NUM_INTERVENTI
1	1
2	1
3	1
4	1
5	1

Attiva Windows  
Passa a Impostazioni per attivare Windows.

```
--11. Verificare la percentuale di sensori attivi rispetto al totale:
SELECT
| COUNT(CASE WHEN S.stato_operativo='Operativo' THEN 'ForzaNapoli' END) / COUNT(*) * 100 AS Percentuale
FROM SENSORI S;
```

PERCENTUALE
65

```
--12. Contare il numero di risoluzioni fallite su sensori per missione
SELECT E.Codice_Missione AS Missione, COUNT(*) AS Risoluzioni_Falliste
FROM Esiti_per_Anomalie_su_Sensori E
WHERE E.Esito_Riparazione = 'FALSE'
GROUP BY E.Codice_Missione;
```

MISSIONE	RISOLUZIONI_FALLISTE
1	5
2	1
4	2
5	1

### 3.4 Viste

Le viste sono state implementate per semplificare, la creazione di query non banali e l'analisi dei dati. Una vista può aggregare, filtrare e unire dati provenienti da più tabelle, riducendo la complessità per l'utente finale. Esistono due tipi di viste, viste materializzate e non materializzate. Una vista non materializzata è una finestra che non memorizza i dati da nessuna parte. Ogni volta che la usi, il database calcola al momento i risultati in base ai dati presenti nelle tabelle. I

vantaggi sono che è sempre aggiornata con i dati più recenti e non occupa spazio aggiuntivo nel database, ma lo svantaggio è che è più lenta perché deve rifare i calcoli ogni volta. Una vista materializzata, invece, salva i risultati dei dati in una tabella fisica. È come fare una foto: i dati vengono copiati e memorizzati in quel momento. I vantaggi sono che è più veloce perché i risultati sono già pronti e utile per analisi complesse o dataset molto grandi, ma gli svantaggi sono che deve essere aggiornata manualmente o automaticamente per riflettere eventuali cambiamenti nei dati originali e occupa spazio nel database. Quelle realizzate per la gestione delle missioni sono non materializzate.

-- Unisce MISSIONI RISORSE_2 SENSORI			
CREATE VIEW Sensori_per_missione AS			
SELECT			
M.CODICE AS Codice_Missione,	S.ID AS ID_Sensore,	S.STATO_OPERATIVO AS Stato_Sensore,	S.DATA_ULTIMO_CONTROLLO AS Ultimo_controllo
FROM (MISSIONI M JOIN RISORSE_2 R2 ON M.CODICE = R2.MISSIONE) JOIN SENSORI S ON R2.SENSORE = S.ID;			
CODICE_MISSED	ID_SENSORE	STATO_SENSORE	ULTIMO_CONTROLLO
1	1	Operativo	1/10/2024
1	2	Operativo	1/05/2024
1	3	In manutenzione	1/08/2024
2	4	Operativo	1/07/2024
2	5	Non operativo	1/01/2024 Attiva Windows Passa a Impostazioni per attivare Windows.

-- Unisce Anomalie Risoluzione			
CREATE VIEW Esiti_per_Anomalie AS			
SELECT			
A.SENSORE AS ID_Sensore,	A.CAUSA AS Causa_Anomalia,	R.ESITO AS Esito_Riparazione	
FROM RISOLUZIONI R JOIN ANOMALIE A ON R.ANOMALIADATA = A.DATA AND R.ANOMALIAORA = A.ORA AND R.ANOMALIASENSORE = A.SENSORE;			
ID_SENSORE	CAUSA_ANOMALIA	ESITO_RIPARAZIONE	
1	Fluttuazione di potenza nei pannelli solari	TRUE	
2	Interferenze nel sistema di comunicazione	FALSE	
3	Surrocaldamento del modulo abitativo	TRUE	
4	Malfunzionamento del sensore di ossigeno	TRUE	
1	Perdita di pressione nella camera di equilibrio	FALSE Attiva Windows Passa a Impostazioni per attivare Windows.	

-- Unisce le viste Sensori_per_missione Esiti_per_Anomalie			
CREATE VIEW Esiti_per_Anomalie_su_Sensori AS			
SELECT			
S.Codice_Missione AS Codice_Missione,	S.ID_Sensore AS ID_Sensore,	E.Esito_Riparazione AS Esito_Riparazione	
FROM Sensori_per_missione S JOIN Esiti_per_Anomalie E ON S.ID_Sensore=E.ID_Sensore;			
CODICE_MISSED	ID_SENSORE	ESITO_RIPARAZIONE	
1	1	TRUE	
1	2	FALSE	
1	3	TRUE	
2	4	TRUE	
3	1	FALSE Attiva Windows Passa a Impostazioni per attivare Windows.	

### 3.5 Trigger

I trigger sono stati implementati per automatizzare operazioni che devono essere eseguite in risposta a determinati eventi nel database, come inserimenti, aggior-

namenti o eliminazioni di dati, assicurando che le informazioni siano sempre aggiornate senza la necessità di intervento manuale.

Di seguito vengono illustrati tutti i trigger implementati, con in allegato una breve descrizione della funzionalità che implementano.

**Trigger n.1:** Inserimento automatico dell'intervento per un'anomalia critica: Quando viene registrata un'anomalia con priorità "critica", il trigger inserisce automaticamente un intervento con una descrizione predefinita ("Intervento urgente richiesto") e assegna una data di esecuzione immediata.

```

1 CREATE OR REPLACE TRIGGER trg_insert_intervento_critico
2 AFTER INSERT ON Anomalie
3 FOR EACH ROW
4 BEGIN
5   IF :NEW.LIVELLO_PRIORITA = 3 THEN
6     -- Dichiarazione della variabile
7     DECLARE
8       ULTIMO_ID INTERVENTI.CODICE%TYPE;
9       V_COUNT    NUMBER;
10    BEGIN
11      -- Primo step: creo l'ID del nuovo intervento
12      SELECT COUNT(*) INTO V_COUNT FROM INTERVENTI ;
13      IF V_COUNT=0 THEN
14        ULTIMO_ID:=1;
15      ELSE
16        SELECT (MAX(CODICE) + 1) INTO ULTIMO_ID FROM
17          INTERVENTI;
18        END IF;
19      -- Creo l'intervento
20      INSERT INTO INTERVENTI (CODICE, DESCRIZIONE)
21      VALUES (ULTIMO_ID, 'Anomalia critica rilevata,
22              intervento urgente!!');
23      -- Creazione delle risoluzioni
24      INSERT INTO RISOLUZIONI VALUES (NULL, :NEW.DATA,
25          ULTIMO_ID,:NEW.DATA, :NEW.ORA, :NEW.SENSORE)
26          ;
27    END;
28  END IF;
29 END;
```

**Trigger n.2** Aggiornamento dello stato del sensore: Quando viene registrata un'anomalia, il trigger aggiorna automaticamente lo stato operativo del sensore associato a "malfunzionante".

```

1 CREATE OR REPLACE TRIGGER aggiorna_stato_sensore
2 AFTER INSERT ON ANOMALIE
3 FOR EACH ROW
4 BEGIN
5   UPDATE SENSORI
```

```

6      SET STATO_OPERATIVO = 'Malfunzionante'
7      WHERE ID = :NEW.SENSORE;
8  END;

```

**Trigger n.3** Aggiornamento automatico dello stato operativo: Se il numero di anomalie registrate per un sensore supera una certa soglia, il trigger cambia il suo stato operativo a 'In manutenzione'.

```

1 CREATE OR REPLACE TRIGGER
2   aggiorna_stato_in_manutenzione_soglia
3 AFTER INSERT ON ANOMALIE
4 FOR EACH ROW
5 DECLARE
6   v_num_anomalie NUMBER;
7   v_soglia NUMBER := 5; -- Soglia predefinita per il
8   numero di anomalie
9 BEGIN
10   -- Conta il numero di anomalie per il sensore associato
11   SELECT COUNT(*)
12     INTO v_num_anomalie
13     FROM anomalie
14     WHERE sensore = :NEW.sensore;
15
16   -- Se il numero di anomalie supera la soglia, aggiorna
17   -- lo stato del sensore
18   IF v_num_anomalie > v_soglia THEN
19     UPDATE SENSORI
20       SET STATO_OPERATIVO = 'In manutenzione',
21           WHERE ID = :NEW.sensore;
22   END IF;
23 END;

```

**Trigger n.4** Inserimento della data prefissata di esecuzione di un intervento al verificarsi di un'anomalia (*per anomalie non critiche*)

```

1 CREATE OR REPLACE TRIGGER trg_insert_intervento
2 AFTER INSERT ON Anomalie
3 FOR EACH ROW
4 DECLARE
5   ULTIMO_ID INTERVENTI.CODICE%TYPE;
6   V_COUNT NUMBER;
7   V_DATE DATE;
8 BEGIN
9   -- Esegui solo per anomalie non critiche
10  IF :NEW.LIVELLO_PRIORITA < 3 THEN
11    -- Calcolo del prossimo ID per la tabella INTERVENTI
12    SELECT COUNT(*) INTO V_COUNT FROM INTERVENTI;
13
14  IF V_COUNT = 0 THEN

```

```

15          ULTIMO_ID := 1; -- Primo intervento
16      ELSE
17          SELECT MAX(CODICE) + 1 INTO ULTIMO_ID FROM
18              INTERVENTI;
19      END IF;
20
21          -- Calcolo della data di risoluzione predefinita (
22          -- una settimana dopo l'anomalia)
23          V_DATE := :NEW.DATA + 7;
24
25          -- Inserimento dell'intervento nella tabella
26          -- INTERVENTI
27          INSERT INTO INTERVENTI (CODICE, DESCRIZIONE)
28              VALUES (ULTIMO_ID, 'Intervento default_automatizzato
29                  ');
30
31          -- Inserimento della risoluzione nella tabella
32          -- RISOLUZIONI
33          INSERT INTO RISOLUZIONI
34              VALUES (NULL, V_DATE, ULTIMO_ID, :NEW.DATA, :NEW.ORA
35                  , :NEW.SENSORE);
36      END IF;
37  END;

```

**Trigger n.5** Verifica che le date di fine delle missioni siano successive alla data di inizio (altrimenti annulla l'INSERT)

```

1 CREATE OR REPLACE TRIGGER verifica_date_missioni
2 BEFORE INSERT ON MISSIONI
3 FOR EACH ROW
4 BEGIN
5     --Verifica se la data di fine      precedente alla data
6     --di inizio
7     IF :NEW.DATA_FINE < :NEW.DATA_INIZIO THEN
8         --Lancia un errore se la data di fine
9         --antecedente alla data di inizio
10        RAISE_APPLICATION_ERROR(-20001, 'La data di fine
11            deve essere successiva alla data di inizio.');
12    END IF;
13 END;

```

**Trigger n.6** Verifica che le date di pubblicazione di un report non siano precedenti alla data dell'ultimo report e alla data di inizio della missione (altrimenti annulla INSERT)

```

1 --6.Verifica che le date di pubblicaz. di un report non
2 --siano precedenti alla data dell'ultimo report e alla data
3 --di inizio della missione (altrimenti annulla INSERT)
4 CREATE OR REPLACE TRIGGER verifica_date_report

```

```

3 BEFORE INSERT ON report
4 FOR EACH ROW
5 DECLARE
6     v_data_ultimo_report DATE;
7     v_data_inizio_missione DATE;
8 BEGIN
9     -- Recupera la data dell'ultimo report pubblicato
10    SELECT MAX(DATA)
11        INTO v_data_ultimo_report
12        FROM report
13        WHERE MISSIONE = :NEW.MISSIONE;
14
15    -- Recupera la data di inizio della missione associata
16    -- al nuovo report
17    SELECT DATA_INIZIO
18        INTO v_data_inizio_missione
19        FROM missioni
20        WHERE CODICE = :NEW.CODICE;
21
22    -- Verifica che la data di pubblicazione non sia
23    -- precedente all'ultimo report
24    IF :NEW.DATA < v_data_ultimo_report THEN
25        RAISE_APPLICATION_ERROR(-20001, 'La data di
26            pubblicazione del report non puo essere
27            precedente a quella dell''ultimo report.');
28    END IF;
29
30    -- Verifica che la data di pubblicazione non sia
31    -- precedente alla data di inizio della missione
32    IF :NEW.DATA < v_data_inizio_missione THEN
33        RAISE_APPLICATION_ERROR(-20002, 'La data di
34            pubblicazione del report non puo essere
35            precedente alla data di inizio della missione.');
36    END IF;
37 END;

```

**Trigger n.7** Quando viene inserita una Risoluzione, Esito deve essere posto a NULL (Modifica l'INSERT) (Before)

```

1 CREATE OR REPLACE TRIGGER imposta_risoluzioni_esito_null
2 BEFORE INSERT ON risoluzioni
3 FOR EACH ROW
4 BEGIN
5     -- Imposta il campo Esito a NULL prima dell'inserimento
6     :NEW.esito := NULL;
7 END;

```

**Trigger n.8** Verifica che la data di un'anomalia sia successiva alla data di installazione sensore (altrimenti annulla INSERT)

```

1 CREATE OR REPLACE TRIGGER verifica_data_anomalia
2 BEFORE INSERT ON ANOMALIE
3 FOR EACH ROW
4 DECLARE
5     v_data_installazione DATE;
6 BEGIN
7     -- Recupera la data di installazione del sensore
8         associato all'anomalia
9     SELECT data_installazione
10    INTO v_data_installazione
11   FROM sensori
12  WHERE ID = :NEW.SENSORE;
13
14  -- Verifica che la data dell'anomalia non sia precedente
15      alla data di installazione del sensore
16  IF :NEW.DATA < v_data_installazione THEN
17      RAISE_APPLICATION_ERROR(20001, 'La data dell'
18          anomalia non puo essere precedente alla data di
19          installazione del sensore.');
20  END IF;
21
22 END;

```

**Trigger n.9** Verifica che la data di esecuzione di una risoluzione sia successiva rispetto la data dell'anomalia (altrimenti annulla INSERT)

```

1 CREATE OR REPLACE TRIGGER verifica_data_risoluzione
2 BEFORE INSERT ON RISOLUZIONI
3 FOR EACH ROW
4 DECLARE
5     v_data_anomalia DATE;
6 BEGIN
7     -- Recupera la data dell'anomalia associata alla
8         risoluzione
9     SELECT DATA
10    INTO v_data_anomalia
11   FROM ANOMALIE A
12  WHERE A.DATA= :NEW.ANOMALIADATA AND A.ORA=:NEW.
13          ANOMALIAORA AND A.SENSORE=:NEW.ANOMALIASSENSORE;
14
15  -- Verifica che la data di esecuzione della risoluzione
16      sia >= alla data dell'anomalia
17  IF :NEW.DATASECUZIONE < v_data_anomalia THEN
18      RAISE_APPLICATION_ERROR(-20002, 'La data di
19          esecuzione della risoluzione deve essere uguale o
20          successiva alla data dell''anomalia.');
21  END IF;
22
23 END;

```

**Trigger n.10** Implementazione della politica di ON UPDATE CASCADE per il codice di una missione

```
1 CREATE OR REPLACE TRIGGER trg_update_cascade_missione
2 AFTER UPDATE OF CODICE ON MISSIONI
3 FOR EACH ROW
4 BEGIN
5     UPDATE RISORSE_1
6     SET MISSIONE=:NEW.CODICE
7     WHERE MISSIONE=:OLD.CODICE;
8
9     UPDATE RISORSE_2
10    SET MISSIONE=:NEW.CODICE
11    WHERE MISSIONE=:OLD.CODICE;
12
13    UPDATE RISORSE_3
14    SET MISSIONE=:NEW.CODICE
15    WHERE MISSIONE=:OLD.CODICE;
16
17    UPDATE REPORT
18    SET MISSIONE=:NEW.CODICE
19    WHERE MISSIONE=:OLD.CODICE;
20 END;
```

**Trigger n.11** Imposta una nuova data di esecuzione per un intervento se questo genera fallimento (esito = false)

```
1 CREATE OR REPLACE TRIGGER trg_nuova_data_esec_per_fallimento
2 BEFORE UPDATE ON risoluzioni
3 FOR EACH ROW
4 WHEN (NEW.ESITO = 'FALSE')
5 BEGIN
6     -- Aggiorna la data prefissata di esecuzione per l'
7     -- intervento
8     -- Ad esempio, una settimana dopo
9     :NEW.DATAESECUZIONE := :OLD.DATAESECUZIONE + 7;
10    :NEW.ESITO := NULL;
11 END;
```

### 3.6 Utenti

Creare utenti diversi con permessi specifici in una base dati è fondamentale per garantire sicurezza, controllo e organizzazione. Questo approccio permette di limitare l'accesso ai dati sensibili solo a chi ne ha bisogno, riducendo il rischio di errori o abusi. Ad esempio, un amministratore può avere accesso completo per gestire tabelle e utenti, mentre un analista può solo leggere i dati e un addetto all'inserimento può aggiungere nuove informazioni senza modificarne

altre. In questo modo si tutela l'integrità del database e si rispettano i principi di responsabilità e minimizzazione degli accessi.

```
CREATE ROLE MembroEQ;  
GRANT SELECT ON SENSORI TO MembroEQ;  
GRANT INSERT ON REPORT TO MembroEQ;  
CREATE USER NApollon IDENTIFIED BY Napoli;  
GRANT MembroEQ TO NApollon;
```

Nell'applicazione è stato implementato un controllo di sicurezza di questo tipo, consultabile a questo [4.11](#) paragrafo

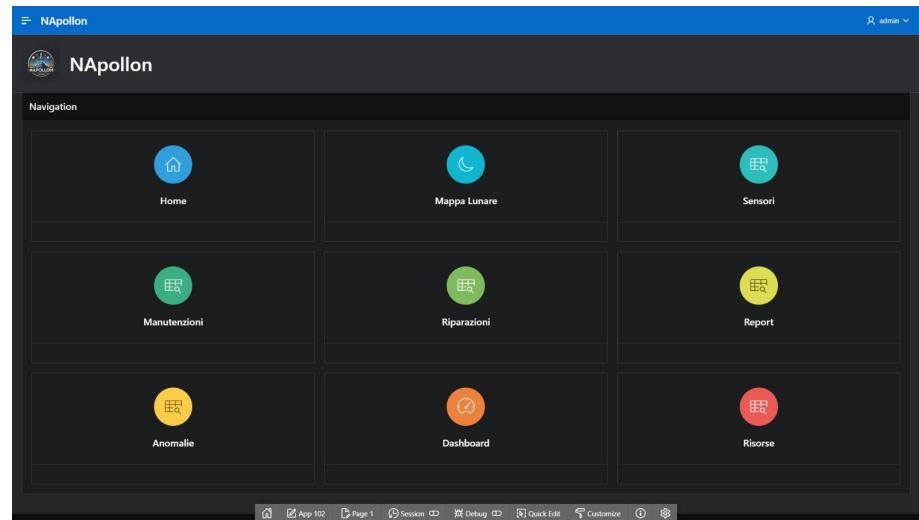
# Chapter 4

## Oracle APEX

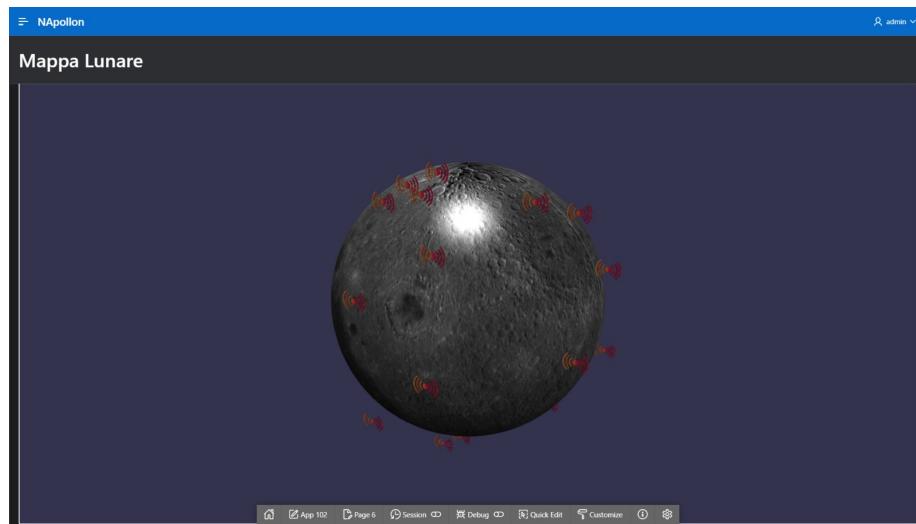
### 4.1 Introduzione

Oracle APEX (Application Express) è una piattaforma di sviluppo rapido di applicazioni (RAD) che consente di creare applicazioni web basate su database Oracle in modo semplice e veloce, senza necessità di scrivere molta codifica. È una delle soluzioni più potenti e popolari per lo sviluppo di applicazioni aziendali, e permette di creare interfacce utente interattive per interagire con i dati archiviati in un database Oracle.

### 4.2 Home



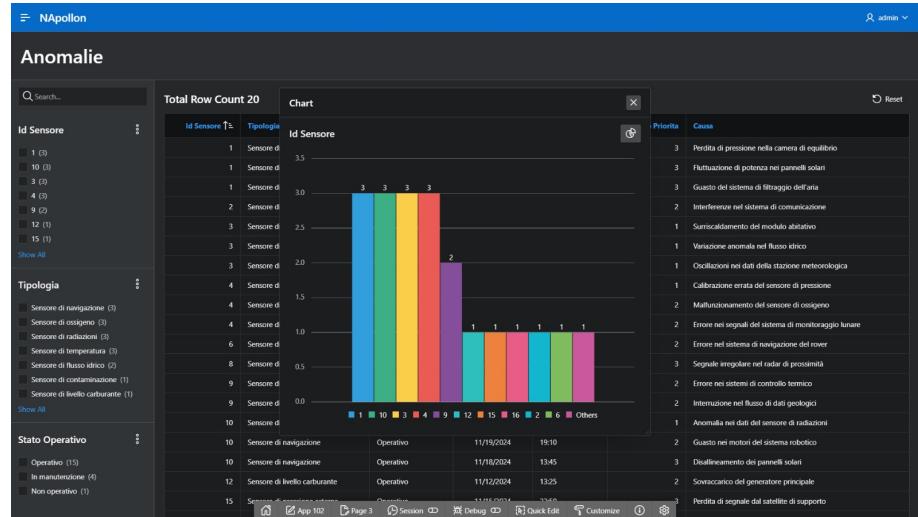
## 4.3 Mappa Lunare Real-Time



## 4.4 Anomalie

The screenshot displays a table titled "Anomalie" with 20 rows of data. The columns are: Id Sensore, Tipologia, Stato Operativo, Data Anomalia, Ora Anomalia, Livello Priorità, and Causa. The data includes various sensor types like temperature, pressure, oxygen, and radiation sensors, along with their operational status (Operativo, In manutenzione), the date and time of the anomaly, the priority level (1-3), and the cause of the anomaly.

Total Row Count 20						
Id Sensore	Tipologia	Stato Operativo	Data Anomalia	Ora Anomalia	Livello Priorità	Causa
1 (3)	Sensore di temperatura	Operativo	11/5/2024	11:30	3	Perturbazione di pressione nella camera di equilibrio
10 (3)	Sensore di temperatura	Operativo	11/1/2024	08:30	3	Fluttuazione di potenza nei pannelli solari
3 (3)	Sensore di temperatura	Operativo	11/1/2024	06:45	3	Guasto del sistema di filtraggio dell'aria
4 (3)	Sensore di pressione	Operativo	11/2/2024	10:15	2	Interferenze nel sistema di comunicazione
9 (2)	Sensore di radiazioni	In manutenzione	11/3/2024	14:45	1	Suriscaldamento del modulo abitativo
12 (1)	Sensore di radiazioni	In manutenzione	11/7/2024	07:05	1	Variazione anomala nel flusso selenio
15 (1)	Sensore di radiazioni	In manutenzione	11/13/2024	15:00	1	Oscillazioni nei dati della stazione meteorologica
Show All						
<b>Tipologia</b>						
Sensore di navigazione (3)						
Sensore di ossigeno (3)						
Sensore di radiazioni (3)						
Sensore di temperatura (3)						
Sensore di flusso idrico (2)						
Sensore di contaminazione (1)						
Sensore di livello carburante (1)						
Show All						
<b>Stato Operativo</b>						
Operativo (15)						
In manutenzione (4)						
Non operativo (1)						
15	Sensore di navigazione	Operativo	11/15/2024	13:00	3	Perturbazione nel segnale dal satellite di supporto



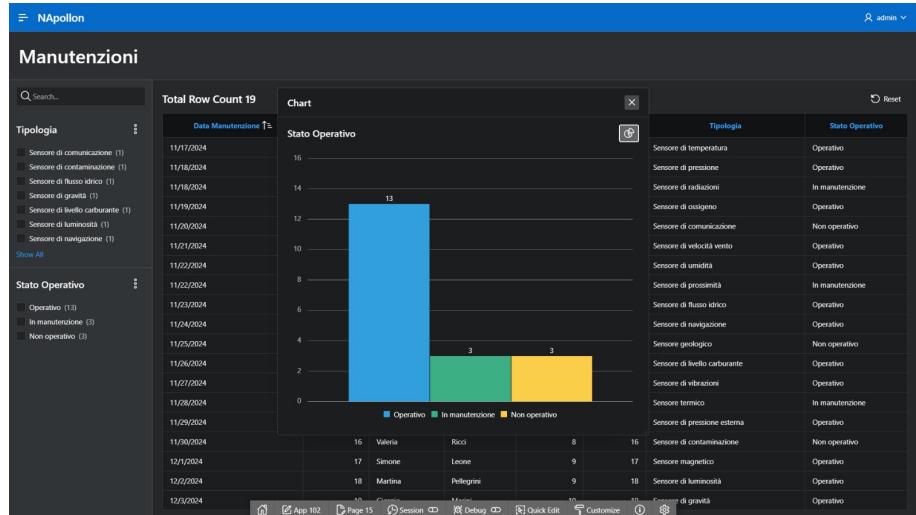
## 4.5 Manutenzioni

**NApollon**

### Manutenzioni

Total Row Count 19

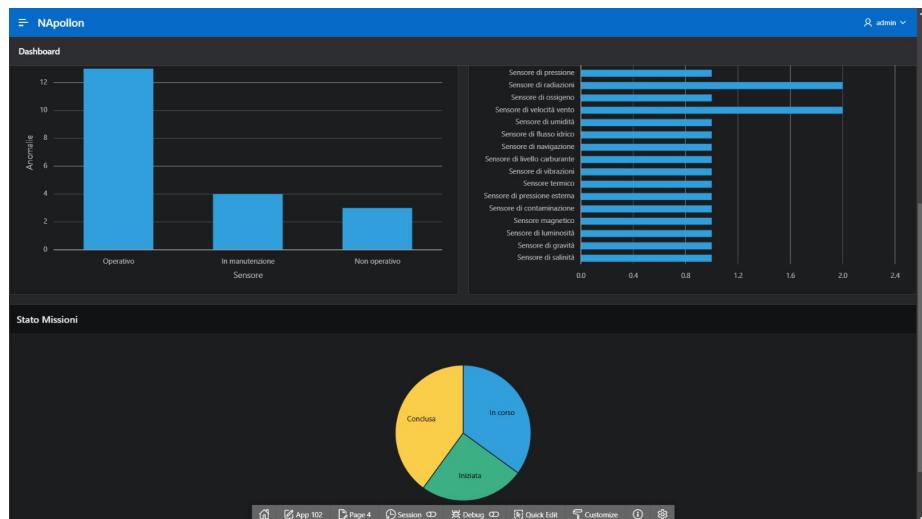
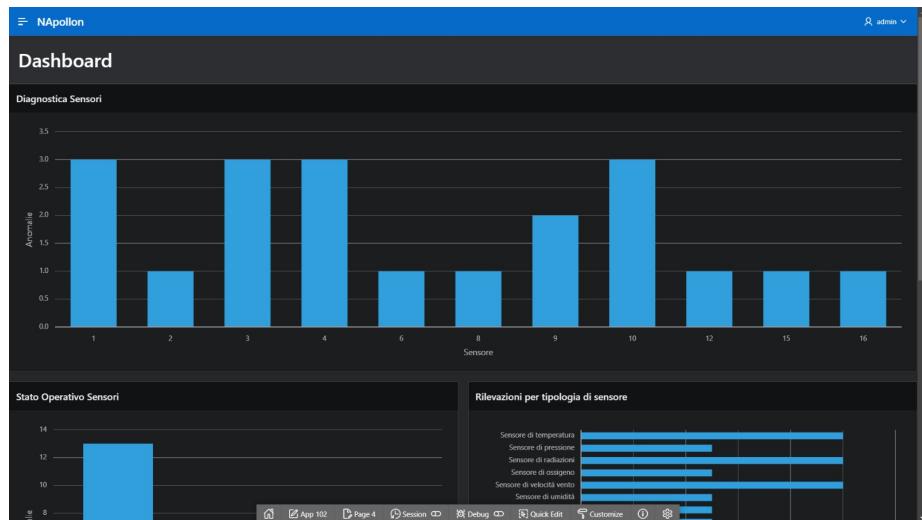
Data Manutenzione ↑	Membro	Nome	Cognome	Misilone	Sensore	Tipologia	Stato Operativo
11/17/2024	1	Luca	Rossi	1	1	Sensore di temperatura	Operativo
11/18/2024	2	Maria	Bianchi	1	2	Sensore di pressione	Operativo
11/18/2024	3	Giovanni	Verdi	2	3	Sensore di radiazioni	In manutenzione
11/19/2024	4	Anna	Neri	2	4	Sensore di ossigeno	Operativo
11/20/2024	5	Marco	Ferrari	3	5	Sensore di comunicazione	Non operativo
11/21/2024	6	Sara	Romanò	3	6	Sensore di velocità vento	Operativo
11/22/2024	7	Deide	Gallo	4	7	Sensore di umidità	Operativo
11/22/2024	8	Chiara	Fontana	4	8	Sensore di prossimità	In manutenzione
11/23/2024	9	Francesco	Barberi	5	9	Sensore di flusso idrico	Operativo
11/24/2024	10	Elena	De Luca	5	10	Sensore di navigazione	Operativo
11/25/2024	11	Alessandro	Greco	6	11	Sensore geologico	Non operativo
11/26/2024	12	Federica	Santini	6	12	Sensore di livello carburante	Operativo
11/27/2024	13	Paolo	Moretti	7	13	Sensore di vibrazioni	Operativo
11/28/2024	14	Giulia	Cattaneo	7	14	Sensore termico	In manutenzione
11/29/2024	15	Matteo	Serra	8	15	Sensore di pressione esterna	Operativo
11/30/2024	16	Valeria	Ricci	8	16	Sensore di contaminazione	Non operativo
12/1/2024	17	Simone	Leone	9	17	Sensore magnetico	Operativo
12/2/2024	18	Martina	Pellegrini	9	18	Sensore di luminosità	Operativo
12/3/2024						Correttore di gravità	Operativo



## 4.6 Report

Total Row Count 20						
	Codice Missione	Data Fine	Data Inizio	Obiettivo	Code	Testo
1	1	10/15/2024	5/1/2023	Estabilire una base lunare permanente	1	Il sensore di temperatura ha registrato una lettura inferiore ai livelli di sicurezza, suggerendo una possibile avaria al sistema di riscaldamento. Dopo un'attenta analisi, si è determinato che la causa principale è un guasto all'unità di controllo termico. La missione è stata temporaneamente sospesa per le necessarie riparazioni e per garantire la sicurezza dell'equipaggio.
2	2	2/28/2025	11/1/2024	Ritrovare risorse minerali sulla Luna	2	Il sistema di navigazione del rover ha mostrato segnali di malfunzionamento. Durante l'ultimo percorso, si è rilevato un errore nel software di controllo che ha causato una deviazione dalla rotta programmata. Il team tecnico ha avviato un intervento per ripristinare la piena funzionalità del sistema. Nonostante questo, la missione continua a procedere, seppur con lievi ritardi.
3	3	9/10/2023	1/5/2022	Installare un osservatorio astronomico sulla Luna	3	Un malfunzionamento è stato riscontrato nel sensore di gravità. I dati raccolti hanno mostrato variazioni imprevedibili, indicando una possibile interferenza da raggi X. Dopo un controllo esauriente, si è concluso che il sensore è ancora in grado di fornire dati precisi. La missione non è stata influenzata gravemente, ma è stata avviata una revisione del sistema di monitoraggio.
4	4	12/31/2026	7/1/2025	Costituire un laboratorio scientifico sulla Luna	4	Sono stati riscontrati malfunzionamenti nei sensori di vibrazione a causa di un urto nel loro allineamento. Le vibrazioni anomale potrebbero compromettere l'integrità del rover durante il suo movimento. Sono stati avvati interventi per ricalibrare i sensori e minimizzare i rischi per la missione. La missione prosegue, ma sono stati presi provvedimenti per monitorare la situazione più da vicino.
5	5	3/15/2023	1/1/2022	Sperimentare la produzione di ossigeno sulla Luna	5	Il sensore di ossigeno ha registrato una lettura al di sotto della soglia di sicurezza, suggerendo un malfunzionamento. Dopo un controllo dettagliato, si è determinato che il sensore ha bisogno di una manutenzione urgente. Il sistema di allarme ha segnalato il problema, ma la missione ha subito un rallentamento dovuto all'emergenza. Tuttavia, il rischio è stato contenuto grazie all'intervento tempestivo del team tecnico.
6	6	11/1/2024	8/10/2024	Esplorare i poli lunari	6	Durante un controllo di routine, il sensore di pressione ha mostrato dei valori elevati che indicavano un possibile guasto al sistema di controllo delle operazioni. Il team ha isolato il problema e ridotto il tasso di raffreddamento del circuito. La missione ha subito una breve interruzione, ma le operazioni sono riprese senza ulteriori complicazioni.
7	7	6/30/2027	1/15/2026	Testare la terraformazione lunare	7	Il sensore di luminosità ha mostrato un comportamento strano, con letture che oscillano tra valori estremamente alti e bassi, suggerendo una possibile interruzione nei circuiti di alimentazione. È stato necessario un intervento tecnico immediato per isolare il guasto. Dopo aver ripristinato il sistema, il sensore ha ripreso a funzionare correttamente, ma la missione ha subito alcune ritardate.

## 4.7 Dashboard



## 4.8 Riparazioni

**Riparazioni**

Total Row Count 19

Data Riparazione	Membro	Nome	Cognome	Missons	Sensore	Tipologia	Stato Operativo
11/17/2024	1	Luca	Rossi	1	1	Sensore di temperatura	Operativo
11/18/2024	2	Maria	Bianchi	1	2	Sensore di pressione	Operativo
11/19/2024	3	Giovanni	Verdi	2	3	Sensore di radiazioni	In manutenzione
11/20/2024	4	Anna	Neri	2	4	Sensore di ossigeno	Operativo
11/21/2024	5	Marcos	Ferrari	3	5	Sensore di comunicazione	Non operativo
11/22/2024	6	Sara	Romano	3	6	Sensore di velocità vento	Operativo
11/23/2024	7	Davide	Gallo	4	7	Sensore di umidità	Operativo
11/24/2024	8	Chiara	Fontana	4	8	Sensore di prossimità	In manutenzione
11/25/2024	9	Francesco	Barberini	5	9	Sensore di flusso idrico	Operativo
11/26/2024	10	Elena	De Luca	5	10	Sensore di navigazione	Operativo
11/27/2024	11	Alessandro	Greco	6	11	Sensore geologico	Non operativo
11/28/2024	12	Federica	Santelli	6	12	Sensore di livello carburante	Operativo
11/29/2024	13	Paolo	Moretti	7	13	Sensore di vibrazioni	Operativo
11/30/2024	14	Giulia	Cattaneo	7	14	Sensore termico	In manutenzione
12/1/2024	15	Matteo	Sera	8	15	Sensore di pressione esterna	Operativo
12/2/2024	16	Valeisa	Ricci	8	16	Sensore di contaminazione	Non operativo
12/3/2024	17	Simone	Leone	9	17	Sensore magnetico	Operativo
12/4/2024	18	Marina	Pellegrini	9	18	Sensore di luminosità	Operativo

## 4.9 Risorse

**Risorse**

Total Row Count 3

Codice	Nome Membro	Cognome Membro	Codice Missione	Id Robot	Tipo Robot	Codice Missione	Id Sensore	Stato Sensore	Ultimo Controllo
1	Luca	Rossi	1	1	Rover Lunare	1	1	Operativo	11/10/2024
1	Maria	Bianchi	1	2	Drone di Sorveglianza	1	2	Operativo	11/5/2024
1	Giovanni	Verdi	1	3	Braccio Robotico per Riparazioni	1	3	In manutenzione	11/8/2024

## 4.10 Sensori

Total Row Count 20						
Data Installazione	Data Ultimo Controllo	Tipologia	Stato Operativo	Latitude	Longitude	Altitude
1/10/2020	11/10/2024	Sensore di temperatura	Operativo	12.3456	98.7654	10.5
2/15/2021	11/10/2024	Sensore di pressione	Operativo	-23.4567	87.6543	20
3/20/2022	11/8/2024	Sensore di radiazioni	In manutenzione	34.5678	-76.5432	30
4/10/2020	11/7/2024	Sensore di ossigeno	Operativo	45.6789	65.4321	5
5/25/2019	11/1/2024	Sensore di comunicazione	Non operativo	56.789	-54.321	15.5
6/1/2023	11/2/2024	Sensore di velocità vento	Operativo	-67.8901	43.2109	50
7/15/2020	11/1/2024	Sensore di umidità	Operativo	78.9012	32.1098	12
8/30/2018	11/6/2024	Sensore di prossimità	In manutenzione	-89.0123	-21.0987	0
9/5/2019	11/1/2024	Sensore di flusso idrico	Operativo	12.3456	98.7654	22
10/10/2021	11/5/2024	Sensore di navigazione	Operativo	23.4567	-87.6543	18
11/1/2020	11/3/2024	Sensore geologico	Non operativo	34.5678	76.5432	7.5
12/15/2019	11/1/2024	Sensore di livello carburante	Operativo	45.6789	65.4321	25
1/1/2022	11/4/2024	Sensore di vibrazioni	Operativo	56.789	-54.321	33
2/20/2021	11/6/2024	Sensore termico	In manutenzione	-67.8901	43.2109	42.5
3/10/2018	11/1/2024	Sensore di pressione esterna	Operativo	78.9012	32.1098	14.5
4/5/2020	11/16/2024	Sensore di contaminazione	Non operativo	-89.0123	-21.0987	8.5
5/15/2022	11/18/2024	Sensore magnetico	Operativo	12.3456	98.7654	3
6/1/2021	11/1/2024	Sensore di luminosità	Operativo	-23.4567	87.6543	9
7/20/2019	11/20/2024	Sensore di temperatura	Operativo	34.5678	-76.5432	19.5

## 4.11 Sicurezza

L'utente è un membro dell'equipaggio. Non puoi accedere a questa sessione. Autorizzazioni insufficienti

Access denied by Page security check

Technical Info (only available for developers)

```

1. is_internal_error: true
2. apex_error_code: APEX.AUTHORIZATIONACCESS_DENIED
3. component_type: APEX_APPLICATION_AUTHORIZATION
4. component_id: 10434568665752279
5. component_name: Diritti Utenti Normali
6. error_backtrace:
----- PL/SQL Call Stack -----
object    line object
handle    name      number
000077F7C3F13A0 985  package body APEX_Z2020B_MAX_FLOW_ERROR_INTERNAL.GET_ERROR
000077F7C3F13A0 1049  package body APEX_Z2020B_MAX_FLOW_ERROR_INTERNAL.ADD_ERROR
000077F7C3F13A0 1532  package body APEX_Z2020B_MAX_FLOW_ERROR_INTERNAL.FINISH_ERROR
000077F7C3F13A0 933  package body APEX_Z2020B_MAX_FLOW_ERROR_INTERNAL.AUTHORIZED
000077F7C350E7B 3485  package body APEX_Z2020B_MAX_FLOW_ERROR_INTERNAL.AUTHORIZED
000077F7C350E7B 5393  package body APEX_Z2020B_MAX_FLOW_ERROR_INTERNAL.FINISH_ERROR
000077F7C3F13A0 2 anonymous block

```

OK