Progetto Basi di Dati

Buglione Giuseppe[1], Calcagno Mario[2], and Calculli Francesco[3]

[1]Dipartimento delle Tecnologie dell'Informazione, Università Federico II

a.a. 2024/2025

Chapter 1

Introduzione

`ASTRADM` nasce con l'intento di fornire un sistema informatico per l'analisi e la gestione dei dati raccolti durante le missioni d'esplorazione spaziali. Il sistema permette da parte dei membri dell'equipaggio una gestione completa delle informazioni provenienti dai sensori e fornire un monitoraggio dettagliato delle operazioni in corso.

1.1 Specifica Informale

A.D.M. (Astra Data Management), un sistema informatico ,a supporto per i membri delle future missioni di esplorazioni lunari

1.1.1 Obiettivi Principali:

- La gestione e analisi dei dati raccolti da sensori sparsi sulla superficie lunare
- Gestione delle possibili anomalie riscontrabili e dei conseguenti interventi di riparazione.
- Gestori di robot autonomi per operazioni sul campo.
- Coordinazione in tempo reale della attivita dei membri della missione
- Monitoraggio delle operazioni in corso e dei relativi report.

1.2 Specifica sui dati

Come visto già nella specifica informale, è necessario poter gestire i seguenti dati:

Anomalie Possibili guasti o anomalie di un sensore

Intervento Interventi di riparazione, possono coinvolgere uno o più operatori

Membri dell'equipaggio Dati degli operatori assegnati ad una missione

Missione Missione di esplorazione lunare

Report Report sull'esito di una missione da parte di un operatore

Rilevazione Le rilevazioni fatte da un sensore

Robot Una delle risorse impiegate durante una missione

Sensore Una delle risorse impiegate durante una missione

1.3 Specifica delle funzionalità

La piattaforma deve mettere a disponibilità le seguenti funzionalità:

- Monitoraggio delle operazioni
- Gestione dei sensori sulla superficie lunare
- Supporto a diverse operazioni di analisi statistica

In particolare gli operatori devono essere in grado di:

- Registrare o cancellare le missioni
- Registrare o cancellare le anomalie
- Inserire i report

1.4 Specifica utenti e policy di sicurezza

Creato lo schema della basi di dati:

```
CREATE SCHEMA ASTRADM;
```

Si è deciso di creare i seguenti utenti:

Database Administrator DBA, con controllo completo della base di dati

```
CREATE USER astradm_dba IDENTIFIED BY astradm_dba;
GRANT DBA TO astradm_dba;
```

Admin Amministratore di sistema, ha i permessi CREATE, DROP, INSERT,UPDATE, DELETE e SELECT, oltre alla possibilità di creare nuove stored procedures

```
CREATE ROLE admin;
GRANT CONNECT TO admin;
GRANT RESOURCE TO admin;
```

Operatore Un operatore ha i permessi INSERT, UPDATE, DELETE, SELECT.

```
CREATE ROLE operator;
GRANT CONNECT TO operator;
GRANT SELECT ON ASTRADM.* TO operator;
GRANT INSERT,UPDATE, DELETE ON ASTRADM.* TO operator
```

Chapter 2

Progettazione Concettuale

In questa sezione si affronta la traduzione delle problematiche presentate in un modello **Entity-Relationship**, abbreviato in "E/R", definendo le entità e le relazioni che lo compongono.

2.1 Schema E/R portante

Analizzando le problematiche presentate si è arrivati all'individuazione delle seguenti entità fondamentali:

Membro Anagrafica dei membri dell'equipaggio

Missione Affidata dall'agenzia spaziale internazionale per l'esplorazione e analisi del terreno lunare

Report Report redatti dai membri sull'esito di una missione

Robot Una delle risorse utilizzabili in una missione

Sensore La risorsa principale utilizzata in una missione

Come visibile da 2.1 sono state individuate le seguenti relazioni principali:

Partecipazione Indica la partecipazione di un'**membro** ad una data **missione**

Resoconto Indica la **missione** a cui si riferisce un dato **report**

`Risorsa_1` Indica l'utilizzo di una data risorsa di tipo **sensore** in una data **missione**

`Risorsa_2` Indica l'utilizzo di una data risorsa di tipo **robot** in una data **missione**

Stesura Indica il **membro** autore di un dato **report**

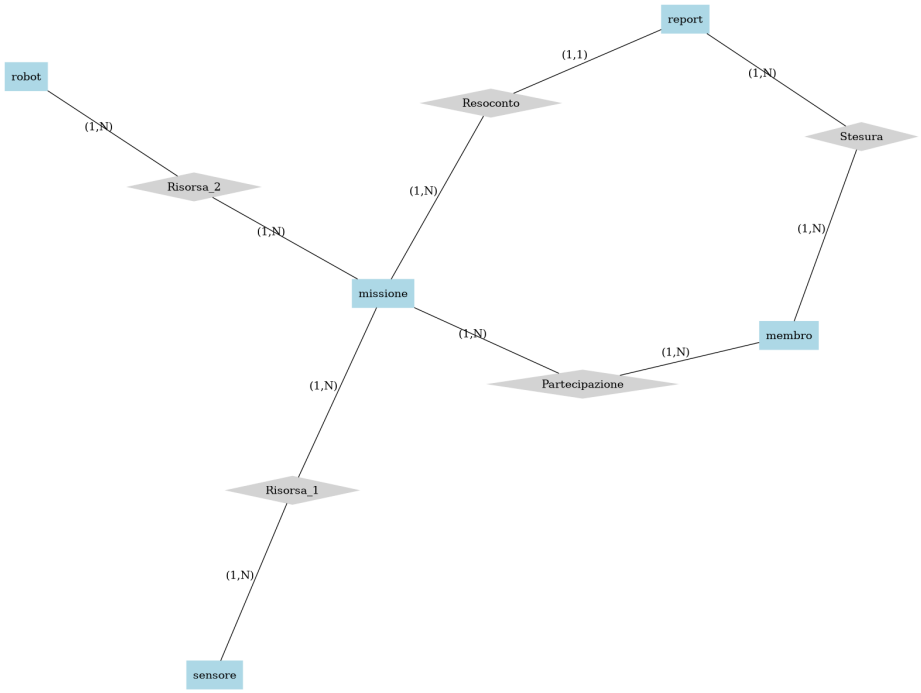

Figure 2.1: Modello E/R portante per `ASTRADM`

2.2 Schema E/R finale

Da una analisi più approfondia si vista la necessità di introdurre delle nuove entità:

Anomalia con cui si va ad indicare un Sensore non funzionante.

Intervento per la riparazione di un Sensore non funzionante.

Rilevazione con cui si va indentificare i dati ralcolti dai sensori.

Caratterizate dalle seguenti relazioni:

Analisi che va a legare Rivelazioni e Sensore, con cui si va ad indicare i dati racolti dai vari sensori.

Malfunzionamento che va a legare Sensore ed Anomalia, tramite cui si individuano i sensori non funzionanti.

Risoluzione che va ad legare Anomalia ed Intervento, azine con cui si va ad riparere un sensore non funzionante.

Nel modello 2.1 sono riporta le **cardinalita**specificando il numero minimo e il numero massimo che, le occorrenze, possono assumere in ciascuna associazione, rispetto alle entità. Le cardinalità delle relazioni sono le seguenti:

Partecipazione di tipo **molti a molti (N,N)**; un membro dell'equipaggio può essere stato selezionato per la partecipazione di diverse missioni spaziali, mentre una missione richiede almeno un membro dell'equipaggio.

Stesura di tipo **molti a molti (N,N)**; un report può essere scritto da molteplici membri dell'equipaggio, mentre un membro dell'equipaggio può comporre diversi report.

Report di tipo **uno a molti (1,N)**; indica l'appartenenza di un report ad un'unica missione, quando invece una missione è composta da diversi report che informano dello stato della missione man mano.

Risorsa_1 e Risorsa_2 entrambe di tipo **molti a molti (N,N)**;in quanto una missione ha la possibilità di utilizzare molteplici sensori, quando un sensore viene ripiegato per una molteplicità di missioni. Analogalmente la relazione è identica per l'utilizzo dei robot nelle missioni.

Malfunzionamento di tipo **molti a molti (1,N)**; in quanto quando si presenta un'anomalia, essa si riferisce ad un determinato sensore. Invece per i sensori vi è la possibilità di trovare diverse anomalie.

Risoluzione di tipo **uno a molti (1,N)** quando si presenta un'anomalia, un intervento può risolverla. Un intervento può essere applicato su diverse anomalie quando hanno la stessa causa;.

Eseguito di tipo **molti a molti (N,N)**; un membro dell'equipaggio può eseguire innumerevoli interventi. Un intervento ha la possibilità di essere eseguito, sia da nessun membro, che da molteplici membri.

Analisi di tipo **uno a molti (1,N)**; una rilevazione viene analizzata da un determinato sensore, il sensore invece effettua diverse rilevazioni.

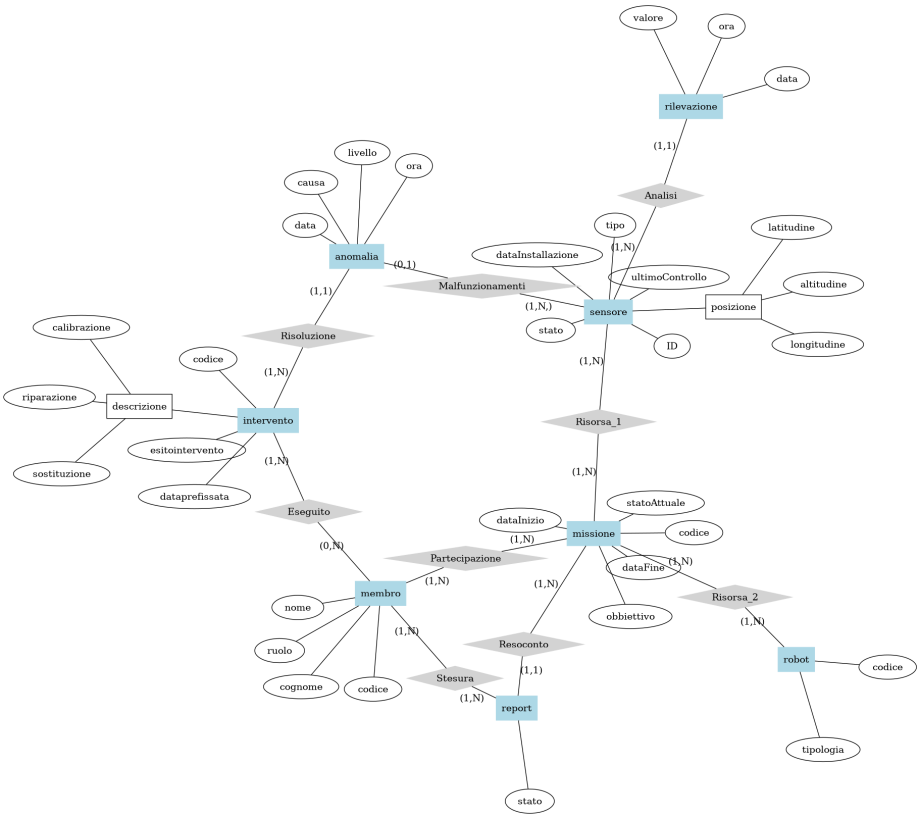

Figure 2.2: Modello E/R definitivo per `ASTRADM`

Chapter 3

Progettazione Logica

Di qui si tratta della progettazione logica del database per `ASTRADM`

3.1 Trasformazione del modello E/R

Riferendoci al modello raffigurato in 2.2 possiamo cominciare a discutera della semplificazione e della specializzazione del modello. Partendo dal modello 2.2 possiamo vedere come gli attributi:

descrizione dell'entità **Intervento**

posizione dell'entità **Sensore**

siano composti. Di conseguenza è possibili trasformarli nel seguente modo:

$$\text{descrizione} \rightarrow \text{sostituzione,riparazione, calibrazione}$$

e

$$\text{posizione} \rightarrow \text{latitudine, longitudine, altitudine}$$

Di conseguenza il modello diventa

3.2 Creazione Schema Relazionale

In questa sezione ci occupiamo della creazione dello schema relazionale partendo dal modello E/R in 3.1. Per la traduzione da modello E/R a schema relazionale si seguono le seguenti regole:

1. Ciascuna *entità* viene trasformata in una relazione, dove gli attributi dell'entità diventano gli attributi della relazione, con la chiave primaria dell'entità che diventa l'identificatore della relazione
2. Le relazioni vengono tradotte in base alla loro cardinalità:

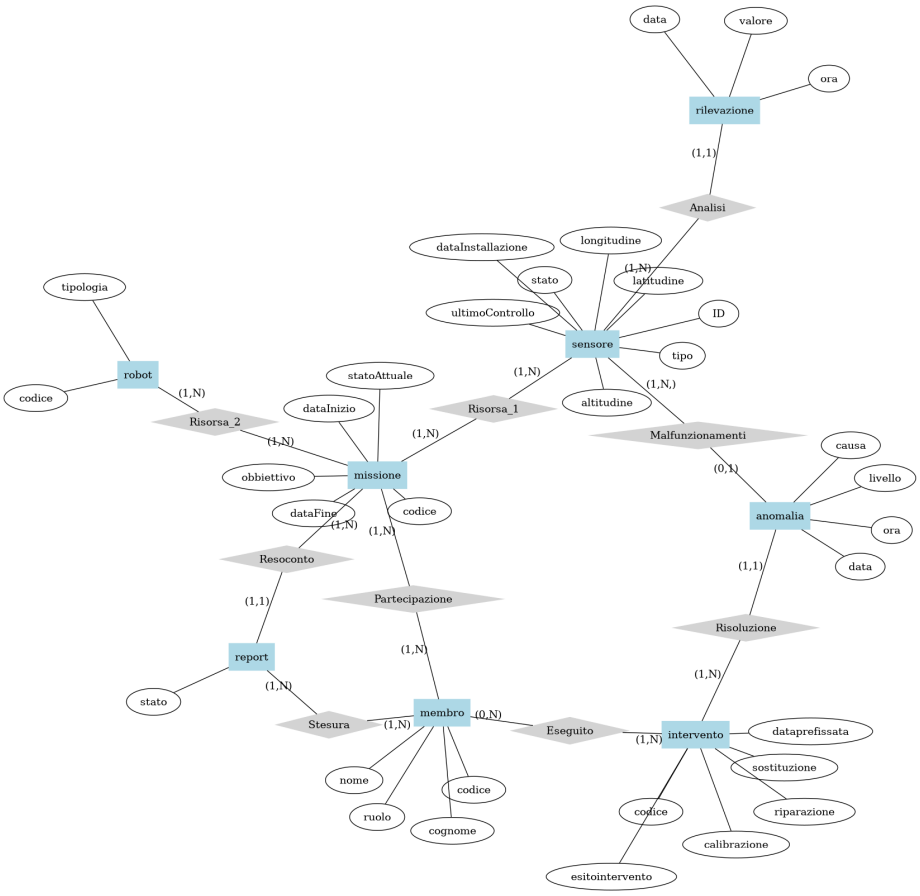

Figure 3.1: Modello ER finale per `ASTRADM`

uno a uno si traducono aggiungendo alle relazioni che traducono le entita dal lato uno gli attributi delle associazioni e gli identificatori delle entita lato molti. Questi ultimi saranno soggetti ad un vincolo di integrita referenziale con il corrispondente attributo delle entita dal lato molti

uno a molti si traducono aggiungendo alla relazione che traduce una delle due entita gli attributi dellassociazione, oltre che il suo identificatore che sara soggetto sia ad un vincolo di unicita che di integrita referenziale con il corrispondente attributo dellaltra entita.

molti a molti si traduce in un relazione con lo stesso nome avente come attributi, gli attributi dell'associazione e gli indentificativi dell'entita coinvolte, dove quest'ultime vanno a costituire un vincolo di integrita referenziale con l'attributo corrispondente dell'entità.

Seguendo queste regole si ottiene il seguente schema:

- `ANOMALIE`(data, ora, causa, livello,SENSORI: sensorie);
- `INTERVENTI`(codice, esito, calibrazione, riparazione, sostituzione, dataPrefissata);
- `MEMBRI`(codice, nome, cognome, ruolo);
- `MISSIONI`(codice, dataFine, obbiettivo, dataInizio, statoAttuale,REPORT:resoconto);
- $\texttt{RISORSA}_1$(SENSORI:sensore,MISSIONI:missione);
- `REPORT`(stato, MEMBRI:autore);
- `RILEVAZIONI`(data, ora, valore, SENSORI: sensore);
- `ROBOT`(codice, tipologia);
- $\texttt{RISORSA}_2$(Robot:robot,MISSIONI:missione);
- `SENSORI`(ID, stato, dataInstallazione, tipo, ultimoControllo, latitudine, altitudine, longitudine);

Chapter 4

Progettazione Fisica

4.1 Dimensionamento dei dati

Per andare fare una stima sul volume che adranno ad occupare i vari dati in base allla loro tipologia, ci baseremo su:

- Tabelle/Schemi ottenute in fase di progettazione logica (sezione 3.2)
- Il numero di missioni previste dall'agenzia spaziale

In questo caso ci baseremo sul popolamento fatto nella sezione 4.4 per fare il dimensionamento.

4.1.1 Anomalia

ANOMALIA		
Attributo	Tipo	Spazio(Byte)
data	DATE	7
ora	VARCHAR(8)	8
causa	VARCHAR(30)	30
livello	INTEGER	4
sensore	INTEGER	4

$$\begin{aligned} D_{\text{anomalia}} &= (D_{\text{data}} + D_{\text{ora}} + D_{\text{causa}} + D_{\text{livello}} + D_{\text{sensore}}) \cdot N_{\text{anomalie}} = \\ &= (7 + 8 + 30 + 4 + 4) \cdot 6 = 318\mathbf{B} \end{aligned} \tag{4.1}$$

4.1.2 Interventi

INTERVENTI		
Attributo	Tipo	Spazio(Byte)
codice	INTEGER	4
esito	VARCHAR(20)	20
calibrazione	VARCHAR(20)	20
riparazione	VARCHAR(20)	20
sostituzione	VARCHAR(20)	20
dataPrefissata	DATE	7

$$D_{\text{interventi}} = (D_{\text{codice}} + D_{\text{esito}} + D_{\text{calibrazione}} + D_{\text{riparazione}} + D_{\text{sostituzione}} + D_{\text{dataPrefissata}}) \cdot N_{\text{int}} = \\ = (4 + 20 + 20 + 20 + 20 + 7) \cdot 5 = 455\mathbf{B} \tag{4.2}$$

4.1.3 Membri

MEMBRI		
Attributo	Tipo	Spazio(Byte)
codice	INTEGER	4
nome	VARCHAR(30)	30
cognome	VARCHAR(30)	30
ruolo	VARCHAR(30)	30

$$D_{\text{membri}} = (D_{\text{codice}} + D_{\text{nome}} + D_{\text{cognome}} + D_{\text{ruolo}}) \cdot N_{\text{membri}} = \\ = (4 + 30 + 30 + 30) \cdot 5 = 470\mathbf{B} \tag{4.3}$$

4.1.4 Missioni

MISSIONI		
Attributo	Tipo	Spazio(Byte)
codice	INTEGER	4
dataFine	DATE	7
obbiettivo	VARCHAR(100)	100
dataInizio	DATE	7
statoAttuale	VARCHAR(30)	30
resoconto	VARCHAR(30)	30

$$D_{\text{missioni}} = \\ = (D_{\text{codice}} + D_{\text{dataFine}} + D_{\text{obbietivo}} + D_{\text{dataInizio}} + D_{\text{statoA}} + D_{\text{dataFine}}) \cdot N_{\text{miss}} = \\ = (4 + 7 + 100 + 7 + 30 + 30) \cdot 5 = 890\mathbf{B} \tag{4.4}$$

4.1.5 Risorsa$_1$

RISORSA$_1$		
Attributo	Tipo	Spazio(Byte)
sensore	INTEGGER	4
missione	INTEGER	4

$$D_{\text{risorsa1}} = (D_{\text{sensore}} + D_{\text{missione}}) \cdot N_{\text{risorse1}} = \\ = (4+4) \cdot 5 = 40\mathbf{B} \tag{4.5}$$

4.1.6 Report

REPORT		
Attributo	Tipo	Spazio(Byte)
stato	VARCHAR(30)	30
autore	INTEGER	4

$$D_{\text{report}} = (D_{\text{stato}} + D_{\text{autore}}) \cdot N_{\text{report}} = \\ = (30+4) \cdot 5 = 170\mathbf{B} \tag{4.6}$$

4.1.7 Rilevazioni

RILEVAZIONI		
Attributo	Tipo	Spazio(Byte)
data	DATE	7
ora	VARCHAR(8)	8
valore	INTEGER	4
sensore	INTEGER	4

$$D_{\text{rivelazione}} = \\ = (D_{\text{data}} + D_{\text{ora}} + D_{\text{valore}} + D_{\text{sensore}}) \cdot N_{\text{rivelazione}} = \\ = (7+8+4+4) \cdot 5 = 115\mathbf{B} \tag{4.7}$$

4.1.8 Robot

ROBOT		
Attributo	Tipo	Spazio(Byte)
codice	INTEGER	4
tipologia	VARCHAR(30)	30

$$D_{\text{robot}} = (D_{\text{codice}} + D_{\text{tipologia}}) \cdot N_{\text{robot}} = \\ = (4+30) \cdot 5 = 170\mathbf{B} \tag{4.8}$$

4.1.9 Sensori

SENSORI		
Attributo	Tipo	Spazio(Byte)
ID	INTEGER	4
stato	VARCHAR(30)	30
dataInstallazione	DATE	7
tipo	VARCHAR(30)	30
ultimoControllo	DATE	7
latitudine	DECIMAL(10,4)	9
altitudine	DECIMAL(10,4)	9
longitudine	DECIMAL(10,4)	9

$$\begin{aligned} D_{\text{sensore}} = & \\ & = (D_{\text{id}} + D_{\text{stato}} + D_{\text{dataInst}} + D_{\text{ultCont}} + D_{\text{lat}} + D_{\text{alt}} + D_{\text{long}}) \cdot N_{\text{sen}} = \\ & = (4 + 30 + 7 + 30 + 7 + 9 + 9 + 9) \cdot 5 = 525\mathbf{B} \end{aligned} \tag{4.9}$$

4.1.10 Risorsa$_2$

RISORSA$_2$		
Attributo	Tipo	Spazio(Byte)
robot	INTEGGER	4
missione	INTEGER	4

$$\begin{aligned} D_{\text{risorsa2}} = (D_{\text{robot}} + D_{\text{missione}}) \cdot N_{\text{risorse2}} = \\ = (4 + 4) \cdot 6 = 48\mathbf{B} \end{aligned} \tag{4.10}$$

4.1.11 Gestione dello spazio e dell'affidabilità

Dalle stime fatte precedentemente possiamo stimare un volume di informazioni pari:

$$D_{\text{tot}} = (318+455+470+890+40+170+115+170+525+48) = 3201\mathbf{B} \tag{4.11}$$

4.2 Creazione del database

Delineato il modello logico, tramite i comandi DDL possiamo inserire all'interno del data base gli schemi di relazioni associate alle tabelle.

4.2.1 Sensori

```
CREATE SEQUENCE sensori_id_seq START WITH 1;
CREATE TABLE SENSORI (
    ID INTEGER DEFAULT sensori_id_seq.nextval NOT NULL,
```

```
    STATO VARCHAR(30) NOT NULL,
    DATAINSTALLAZIONE DATE NOT NULL,
    TIPO VARCHAR(30) NOT NULL,
    UTLIMOCONTROLLO DATE,
    LATITUDINE DECIMAL(10,4) NOT NULL,
    LONGITUDINE DECIMAL(10,4) NOT NULL,
    ALTITUDINE DECIMAL(10,4) NOT NULL,
    CONSTRAINT PK_ID PRIMARY KEY(ID)
);
```

4.2.2 Anomalie

```
CREATE TABLE ANOMALIE(
    DATA DATE NOT NULL,
    ORA VARCHAR(8) NOT NULL,
    CAUSA VARCHAR(30) NOT NULL,
    LIVELLO INTEGER NOT NULL,
    SENSORE INTEGER NOT NULL,
    CONSTRAINT PK_ANOMALIE PRIMARY KEY (DATA,ORA),

     CONSTRAINT FK_ANOMALIE_SENSORI FOREIGN KEY
     ->(SENSORE) REFERENCES SENSORI (ID) ON DELETE CASCADE
);
```

4.2.3 Interventi

```
CREATE SEQUENCE interventi_codice_seq START WITH 1;
CREATE TABLE INTERVENTI(
    CODICE INTEGER DEFAULT interventi_codice_seq.nextval NOT
    ↪  NULL,
    ESITO VARCHAR(20) NOT NULL,
    CALIBRAZIONE VARCHAR(20) NOT NULL,
    RIPARAZIONE VARCHAR(20) NOT NULL,
    SOSTITUZIONE VARCHAR(20) NOT NULL,
    DATAPREFISSATA DATE NOT NULL,
    CONSTRAINT PK_INTERVENTI PRIMARY KEY(CODICE)
);
```

4.2.4 Membri

```
CREATE SEQUENCE membri_codice_seq START WITH 1;
CREATE TABLE MEMBRI(
    CODICE INTEGER DEFAULT membri_codice_seq.nextval NOT
    ↪  NULL,
```

```
    NOME VARCHAR(20) NOT NULL,
    COGNOME VARCHAR(20) NOT NULL,
    RUOLO VARCHAR(20) NOT NULL,
    CONSTRAINT PK_MEMBRI PRIMARY KEY(CODICE)
);
```

4.2.5 Report

```
CREATE TABLE REPORT(
    STATO VARCHAR(30) NOT NULL,
    AUTORE INTEGER NOT NULL,
    CONSTRAINT PK_REPORT PRIMARY KEY(STATO),
    CONSTRAINT FK_REPORT_MEMBRI FOREIGN KEY(AUTORE)
    ->REFERENCES MEMBRI(CODICE) ON DELETE CASCADE
);
```

4.2.6 Missioni

```
CREATE SEQUENCE missioni_codice_seq START WITH 1;
CREATE TABLE MISSIONI(
    CODICE INTEGER DEFAULT missioni_codice_seq.nextval NOT
    ↪  NULL,
    DATAFINE DATE NOT NULL,
    OBBIETTIVO VARCHAR(100) NOT NULL,
    DATAINIZIO DATE NOT NULL,
    STATOATTUALE VARCHAR(30) NOT NULL,
    RESOCONTO VARCHAR(30) NOT NULL,
    CONSTRAINT PK_MISSIONI PRIMARY KEY (CODICE),
    CONSTRAINT FK_MISSIONI_REPORT FOREIGN KEY
    ->(RESOCONTO) REFERENCES REPORT(STATO) ON DELETE CASCADE
);
```

4.2.7 Rilevazioni

```
CREATE TABLE RILEVAZIONI(
    DATA DATE NOT NULL,
    ORA VARCHAR(8) NOT NULL,
    VALORE INTEGER NOT NULL,
    SENSORE INTEGER NOT NULL,
    CONSTRAINT PK_RILEVAZIONI PRIMARY KEY(DATA,ORA),
    CONSTRAINT FK_RILEVAZIONI_SENSORI FOREIGN KEY(SENSORE)
    ->REFERENCES SENSORI (ID) ON DELETE CASCADE
    );
```

4.2.8 Robot

```
CREATE SEQUENCE robo_codice_seq START WITH 1;
CREATE TABLE ROBOT(
    CODICE INTEGER DEFAULT robot_codice_seq.nextval NOT
    ↪  NULL,
    TIPOLOGIA VARCHAR(30),
    CONSTRAINT PK_ROBOT PRIMARY KEY(CODICE)
);
```

4.2.9 Risorsa$_1$

```
CREATE TABLE RISORSA_1(
    ROBOT INTEGER NOT NULL,
    MISSIONE INTEGER NOT NULL,
    CONSTRAINT PK_RISORSA_1 PRIMARY KEY(ROBOT,MISSIONE),

    CONSTRAINT FK_RISORSA1_ROBOT FOREIGN KEY(ROBOT)
    ->REFERENCES ROBOT(CODICE) ON DELETE CASCADE,

    CONSTRAINT FK_RISORSA1_MISSIONE FOREIGN KEY(MISSIONE)
    ->REFERENCES MISSIONI(CODICE) ON DELETE CASCADE
    );
```

4.2.10 Risorsa$_2$

```
CREATE TABLE RISORSA_2(
    SENSORE INTEGER NOT NULL,
    MISSIONE INTEGER NOT NULL,
    CONSTRAINT PK_RISORSA_2 PRIMARY KEY (SENSORE,MISSIONE),

      CONSTRAINT FK_RISORSA2_SENSORE FOREIGN KEY(SENSORE)
   ->REFERENCES SENSORI(ID) ON DELETE CASCADE,

      CONSTRAINT FK_RISORSA2_MISSIONE FOREIGN KEY(MISSIONE)
     ->REFERENCES MISSIONI(CODICE) ON DELETE CASCADE,
    );
```

4.3 Gestione degli indici

Gli indici sono struttura dati ordinata impiegata nel DBMS per la localizzazione di un specifico record andando ad velocizzare le operazioni di ricerca, sono carat-

terizati da una data entry (record memorizato in un file indice) che può essere costituito:

- Intero record di dati
- Coppia costituita da una `chiave` e `rid`
- Coppia costituita da una `chiave` e `lista di rid`

Dove in `DBMS Oracle` va creare in maniera automati indici di tipo `B+Tree` per ogni chiave primaria di tutte le tabelle.

```
CREATE INDEX INTERVENTI_CODICE ON INTERVENTI (codice);
CREATE INDEX REPORT_STATO ON REPORT (stato);
CREATE INDEX ANOMALIE_DATA_ORA ON ANOMALIE (data, ora);
CREATE INDEX RIVELAZIONI_DATA_ORA ON RIVELAZIONI (data, ora);
CREATE INDEX MISSIONI_CODICE ON MISSIONI (codice);
```

4.4 Popolamento della base di dati

In tale sezioni ci occuperemo del popolamento della base dati con l'ausilio di Chatgpt per la creazione di dati fittizie con il fini di testare la nostra base dati. E otteniamo:

- 5 Sensori
- 6 Anomalie
- 5 Interventi
- 5 Membri
- 5 Report
- 5 Missioni
- 5 Rilevazioni
- 5 Robot
- 5 Risorsa_1
- 6 Risorsa_2

4.4.1 Sensori

```
INSERT INTO SENSORI (STATO, DATAINSTALLAZIONE, TIPO,
↪  ULTIMOCONTROLLO, LATITUDINE, LONGITUDINE, ALTITUDINE) VALUES
('Attivo', TO_DATE('2023-01-15', 'YYYY-MM-DD'), 'Temperatura',
↪  TO_DATE('2024-02-01', 'YYYY-MM-DD'), 45.1234, 12.5678,
↪  100.5),
('Guasto', TO_DATE('2022-06-20', 'YYYY-MM-DD'), 'Pressione',
↪  TO_DATE('2024-01-10', 'YYYY-MM-DD'), 44.8765, 13.6789,
↪  200.3),
('Attivo', TO_DATE('2023-03-10', 'YYYY-MM-DD'), 'Umidità',
↪  TO_DATE('2024-02-15', 'YYYY-MM-DD'), 46.2345, 11.3456,
↪  150.0),
('Manutenzione', TO_DATE('2023-05-22', 'YYYY-MM-DD'),
↪  'Temperatura', TO_DATE('2024-02-28', 'YYYY-MM-DD'), 45.6789,
↪  12.4567, 180.7),
('Attivo', TO_DATE('2022-11-05', 'YYYY-MM-DD'), 'Pressione',
↪  TO_DATE('2024-02-10', 'YYYY-MM-DD'), 46.7890, 13.7890,
↪  220.1);
```

4.4.2 Anomalie

```
INSERT INTO ANOMALIE (DATA, ORA, CAUSA, LIVELLO, SENSORE) VALUES
(TO_DATE('2024-02-10', 'YYYY-MM-DD'),'10:30:00', 'Sbalzo di temperatura', 3, 1),
(TO_DATE('2024-01-25', 'YYYY-MM-DD'),'08:15:00', 'Guasto elettronico', 5, 2),
(TO_DATE('2024-02-20', 'YYYY-MM-DD'),'14:45:00', 'Errore di calibrazione', 2, 2),
(TO_DATE('2024-02-22', 'YYYY-MM-DD'),'18:30:00', 'Sbalzo di temperatura', 4, 1),
(TO_DATE('2024-02-28', 'YYYY-MM-DD'),'20:00:00','Sensore non risponde', 3, 1),
(TO_DATE('2024-01-09', 'YYYY-MM-DD'),'10:00:00', 'Danno Hardware', 2, 3);
```

4.4.3 Interventi

```
INSERT INTO INTERVENTI (ESITO, CALIBRAZIONE, RIPARAZIONE,
↪  SOSTITUZIONE, DATAPREFISSATA) VALUES
('Completato', 'Sì', 'No', 'No', TO_DATE('2024-02-20',
↪  'YYYY-MM-DD')),
('In corso', 'No', 'Sì', 'No', TO_DATE('2024-03-05',
↪  'YYYY-MM-DD')),
('Pianificato', 'No', 'No', 'Sì', TO_DATE('2024-03-15',
↪  'YYYY-MM-DD')),
('Completato', 'Sì', 'Sì', 'No', TO_DATE('2024-02-25',
↪  'YYYY-MM-DD')),
('Annullato', 'No', 'No', 'No', TO_DATE('2024-04-01',
↪  'YYYY-MM-DD')),
```

4.4.4 Membri

```
INSERT INTO MEMBRI (NOME, COGNOME, RUOLO) VALUES
('Mario', 'Rossi', 'Tecnico'),
('Laura', 'Bianchi', 'Supervisore'),
('Giovanni', 'Verdi', 'Operatore'),
('Elena', 'Neri', 'Analista'),
('Paolo', 'Moretti', 'Ingegnere'),
```

4.4.5 Report

```
INSERT INTO REPORT (STATO, AUTORE) VALUES
('Confermato', 1),
('In revisione', 2),
('Approvato', 3),
('Respinto', 4),
('In attesa', 5);
```

4.4.6 Missioni

```
INSERT INTO MISSIONI (DATAFINE, OBBIETTIVO, DATAINIZIO,
↪  STATOATTUALE, RESOCONTO) VALUES
(TO_DATE('2024-03-10', 'YYYY-MM-DD'), 'Manutenzione sensori',
↪  TO_DATE('2024-02-25', 'YYYY-MM-DD'), 'Attiva', 'Confermato'),
(TO_DATE('2024-04-01', 'YYYY-MM-DD'), 'Installazione nuovi
↪  dispositivi', TO_DATE('2024-03-15', 'YYYY-MM-DD'),
↪  'Pianificata', 'In revisione'),
(TO_DATE('2024-05-12', 'YYYY-MM-DD'), 'Riparazione anomalie',
↪  TO_DATE('2024-04-20', 'YYYY-MM-DD'), 'In corso',
↪  'Approvato'),
(TO_DATE('2024-06-10', 'YYYY-MM-DD'), 'Test nuovi sensori',
↪  TO_DATE('2024-05-01', 'YYYY-MM-DD'), 'Pianificata', 'In
↪  attesa'),
(TO_DATE('2024-07-05', 'YYYY-MM-DD'), 'Upgrade firmware',
↪  TO_DATE('2024-06-20', 'YYYY-MM-DD'), 'Attiva', 'Confermato');
```

4.4.7 Rilevazioni

```
INSERT INTO RILEVAZIONI (DATA, ORA, VALORE, SENSORE) VALUES
(TO_DATE('2024-02-12', 'YYYY-MM-DD'),'12:00:00', 25, 1),
(TO_DATE('2024-02-12', 'YYYY-MM-DD'),'12:30:00', 27, 1),
(TO_DATE('2024-02-14', 'YYYY-MM-DD'),'09:45:00', 30, 4),
(TO_DATE('2024-02-15', 'YYYY-MM-DD'),'16:20:00', 22, 3),
(TO_DATE('2024-07-05', 'YYYY-MM-DD'),'19:30:00', 22, 5);
```

4.4.8 Robot

```
INSERT INTO ROBOT (TIPOLOGIA) VALUES ('Ricognizione'),
('Riparazione'),
('Supporto logistico'),
('Monitoraggio'),
('Emergenza');
```

4.4.9 Risorsa 1

```
INSERT INTO RISORSA_1 (ROBOT, MISSIONE) VALUES
(1, 1),
(2, 2),
(3, 4),
(2, 3),
(4, 5);
```

4.4.10 Risorsa 2

```
INSERT INTO RISORSA_2 (SENSORE, MISSIONE) VALUES
(1, 1),
(2, 2),
(3, 3),
(4, 5),
(1, 4),
(2, 1);
```

Chapter 5

Operazioni

Per l'implementazione di funzionalità che la base dati deve offrire agli utento aviene mediante l'utilizzo di:

- Viste, sono delle tabelle, che vengono descritte in termini di altre tabelle, una **relazione virtuale**, dove sue tuple non sono memorizzate nella base dati ma ricavabili medianti interrogazioni, deliniando un iterfaccia messa a disposizione di utenti e applicazioni. La sua utilita si trova nei casi in cui si debbano vare tabelle che mettono in relazione campi di tabelle distanti nel modello logico.
- Query, per verificare il corretto funzionamento della base di dati.
- Trigger, sono procedure che si attivano quando vengono eseguite ogni volta che si fa un operazione stabilitasu un oggetto specifico della base dati.

5.1 Viste

5.1.1 Vista dei sensori attivi con ultima rilevazione

Creare una vista che unisca le informazioni sui sensori attivi con la loro ultima rilevazione registrata.

```
CREATE VIEW SENSORI_ATTIVI_RILEVAZIONI AS
SELECT SENSORI.ID AS SENSORI, RILEVAZIONI.DATA AS ULTIME_RILEVAZIONI
FROM SENSORI JOIN RILEVAZIONI ON SENSORI.ID = RILEVAZIONI.SENSORE
WHERE SENSORI.STATO='Attivo' AND RILEVAZIONI.DATA = (
    SELECT MAX(RILEVAZIONI.DATA)
    FROM RILEVAZIONI
    WHERE SENSORI.ID=RILEVAZIONI.SENSORE
);
```

5.1.2 Vista degli interventi completati e il loro esito

Mostrare tutti gli interventi completati con dettagli su calibrazione, riparazione e sostituzione.

```
CREATE VIEW INTERVENTI-COMPLETATI AS
SELECT INTERVENTI.CODICE AS
↪  INTERVENTI,INTERVENTI.ESITO,INTERVENTI.CALIBRAZIONE,
->INTERVENTI.RIPARAZIONE,INTERVENTI.SOSTITUZIONE
FROM INTERVENTI
WHERE INTERVENTI.ESITO = 'Completato';
```

5.1.3 Vista dei membri e dei report creati

Unire la tabella MEMBRI con REPORT per ottenere un elenco dei membri che hanno scritto report e il loro stato.

```
CREATE VIEW MEMBR_REPORT AS
SELECT MEMBRI.CODICE AS MEMBRO_CODE, REPORT.STATO AS STATO_REPORT
FROM MEMBRI JOIN REPORT ON MEMBRI.CODICE = REPORT.AUTORE;
```

5.1.4 Vista delle missioni con i robot assegnati

Creare una vista che visualizzi le missioni attive e i robot assegnati tramite RISORSA_1.

```
CREATE VIEW MISSIONI-ROBOT AS
SELECT MISSIONI.CODICE AS MISSIONI, RISORSA_1.ROBOT
FROM MISSIONI JOIN RISORSA_1 ON MISSIONI.CODICE = RISORSA_1.MISSIONE
WHERE MISSIONI.STATOATTUALE = 'Attiva';
```

5.1.5 Vista delle anomalie recenti

Mostrare tutte le anomalie registrate nellultimo mese con i dettagli del sensore coinvolto.

```
CREATE VIEW ANOMALIE_RECENTI AS
SELECT SENSORI.ID AS SENSORI, ANOMALIE.DATA AS
↪  DATA_ANOMALIA,SENSORI.LATITUDINE
↪  ,SENSORI.LONGITUDINE,SENSORI.ALTITUDINE
FROM SENSORI JOIN ANOMALIE ON SENSORI.ID = ANOMALIE.SENSORE
WHERE TO_DATE(ANOMALIE.DATA, 'YYYY-MM-DD') >= SYSDATE - 30
```

5.2 Query

5.2.1 Recupero anomalie gravi

Selezionare tutte le anomalie con un livello maggiore o uguale a una certa soglia (es. 4), indicando il sensore coinvolto e la data.

```
SELECT SENSORI.ID, ANOMALIE.LIVELLO
FROM SENSORI JOIN ANOMALIE ON SENSORI.ID = ANOMALIE.SENSORE
WHERE ANOMALIE.LIVELLO >= 4
GROUP BY SENSORI.ID,ANOMALIE.LIVELLO;
```

5.2.2 Monitoraggio degli interventi pianificati

Recuperare tutti gli interventi con stato "Pianificato" e la data prevista per l'esecuzione.

```
SELECT INTERVENTI.CODICE, INTERVENTI.ESITO
FROM INTERVENTI
WHERE INTERVENTI.ESITO = 'Pianificato'
GROUP BY INTERVENTI.CODICE, INTERVENTI.ESITO;
```

5.2.3 Lista delle missioni attive con risorse assegnate

Visualizzare tutte le missioni con stato "Attiva", mostrando i robot e i sensori associati.

```
SELECT MISSIONI.STATOATTUALE, RISORSA_1.ROBOT, RISORSA-2.SENSORE
FROM MISSIONI JOIN RISORSA-1 ON MISSIONI.CODICE =
↪  RISORSA-1.MISSIONE JOIN RISORSA_2 ON RISORSA_1.MISSIONE =
↪  RISORSA_2.MISSIONE
WHERE MISSIONI.STATOATTUALE = 'Attiva'
GROUP BY MISSIONI.STATOATTUALE, RISORSA_1.ROBOT,
↪  RISORSA_2.SENSORE;
```

5.2.4 Sensori con più anomalie registrate

Contare il numero di anomalie per ciascun sensore e restituire quelli con il maggior numero di problemi.

```
CREATE VIEW ANOMALIA_SENSORI AS
SELECT SENSORI.ID AS SENSORI,COUNT(ANOMALIE.SENSORE) AS NUM_ANOMALIE
FROM SENSORI JOIN ANOMALIE ON SENSORI.ID = ANOMALIE.SENSORE
GROUP BY SENSORI.ID;

SELECT SENSORI,NUM_ANOMALIE
FROM ANOMALIE-SENSORI
WHERE NUM_ANOMALIE=(
    SELECT MAX(NUM_ANOMALIE)
    FROM ANOMALIE_SENSORI
);
```

5.2.5 Ultime rilevazioni per ogni sensore

Ottenere l'ultima misurazione registrata per ogni sensore.

```
SELECT R1.SENSORE AS SENSORE,R1.VALORE,R1.DATA
FROM RILEVAZIONI R1
WHERE (DATA,ORA) = (
    SELECT MAX(R2.DATA),MAX(R2.ORA)
    FROM RILEVAZIONI R2
    WHERE R1.SENSORE = R2.SENSORE
    RAISE_APPLICATION_ERROR(-20001,'ERRORE: Sensore non trovato');
);
```

5.3 Trigger

5.3.1 Trigger per aggiornare automaticamente la data dell'ultimo controllo di un sensore

Quando viene registrata una nuova rilevazione per un sensore, aggiornare il campo ULTIMOCONTROLLO.

```
CREATE OR REPLACE TRIGGER UPDATE_ULTIMO_CONTROLLO
AFTER INSERT ON RIVELAZIONI
FOR EACH ROW
BEGIN
    UPDATE SENSORI
    SET ULTIMOCONTROLLO = :NEW.DATA
    WHERE ID = :NEW.SENSORE
END;
```

5.3.2 Trigger per impedire l'inserimento di anomalie con data futura

Controllare che la data delle anomalie non sia successiva alla data odierna.

```
CREATE OR REPLACE TRIGGER ANOMALIE_FUTURE
BEFORE INSERT ON ANOMALIE
FOR EACH ROW
DECLARE
     DATA_FUTURA EXCEPTION;
BEGIN
IF :NEW.DATA > SYSDATE THEN
        RAISE DATA-FUTURA;
    END IF;
EXCEPTION
        WHEN DATA-FUTURA THEN
        RAISE_APPLICATIONERROR(-20002,'DATA INSERITA NON POSSIBILE');
END;
```

5.3.3 Trigger per aggiornare lo stato di un sensore in caso di anomalie gravi

e un'anomalia ha livello 4, modificare automaticamente lo stato del sensore in "Guasto" o "Manutenzione".

```
CREATE OR REPLACE TRIGGER STATO_ANOMALI
AFTER INSERT ON ANOMALIE
FOR EACH ROW
BEGIN
    IF :NEW.LIVELLO >=4 THEN
    UPDATE SENSORI
    SET STATO = 'GUASTO'
    WHERE ID = :NEW.SENSORE;
    END IF;
END;
```

5.3.4 Trigger per impedire la cancellazione di membri che hanno scritto report

Evitare la rimozione di un membro se ha creato almeno un report.

```
CREATE OR REPLACE TRIGGER NO_DELETE_MEMBER
BEFORE DELETE ON MEMBRI
FOR EACH ROW
DECLARE
    COUNTER NUMBER;
BEGIN
    SELECT COUNT(*) INTO COUNTER
    FROM REPORT
    WHERE AUTORE = :OLD.CODICE;
    IF COUNTER > 0 THEN
        RAISE_APPLICATION_ERROR(-20004, 'ERRORE: Impossibile
        ↪  eliminare il membro perché ha scritto uno o più
        ↪  report.');
    END IF;
END;
```