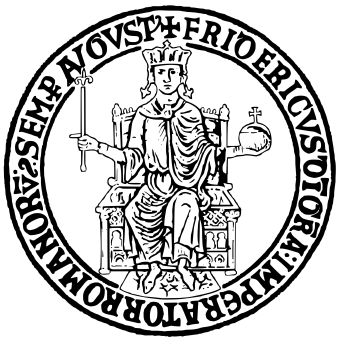
Elaborato di Basi di Dati

"nome progetto"



Professore: Vincenzo Moscato

Autori: Alessandro Cioffi N46006940
Antonio Cirino N46006930

a.a. 2024/2025

Contents

1 Creazione DB

In questa sezione verrà descritta la progettazione e la creazione del database. Il database è stato progettato per soddisfare i requisiti del sistema di monitoraggio e controllo di missioni spaziali e in modo da garantire la corretta memorizzazione e gestione dei dati relativi a missioni, membri dell'equipaggio, sensori, robot, anomalie, rilevazioni e report.

1.1 Progettazione Concettuale

La progettazione concettuale è la fase iniziale del processo di progettazione di un database, in cui si definiscono i requisiti del sistema e si identificano le entità coinvolte, le relazioni tra di esse e gli attributi che le caratterizzano. Questa fase è indipendente dal modello logico e si concentra sulla rappresentazione dei dati in modo astratto, senza considerare i dettagli implementativi.
La progettazione concettuale è stata realizzata attraverso la modellazione del sistema tramite il modello Entità/Relazione (E/R). Questo modello consente di rappresentare in modo chiaro e intuitivo le entità coinvolte, le relazioni tra di esse e gli attributi che le caratterizzano.

1.1.1 Modello E/R portante

Come prima fase della progettazione concettuale, è stato realizzato uno schema E/R portante. Lo schema portante rappresenta il nucleo centrale del modello Entità-Relazione (E/R) per il sistema in esame. Esso evidenzia le principali entità coinvolte: **RISORSA**, **MISSIONE** e **MEMBRO EQUIPAGGIO**, connesse tra loro tramite relazioni chiave.

- La **RISORSA** rappresenta gli strumenti o gli elementi utilizzati nelle missioni.
- La **MISSIONE** è il fulcro operativo, dove vengono assegnate risorse e membri dell'equipaggio.
- Il **MEMBRO EQUIPAGGIO** indica le persone coinvolte nelle missioni, con specifici ruoli e compiti.

Le relazioni descrivono i collegamenti logici tra queste entità, evidenziando l'assegnazione e l'utilizzo di risorse e personale in contesto missione. Questo schema costituisce la base per la progettazione logica e fisica del sistema.

Figure 1: Schema portante del modello E/R

1.1.2 Modello E/R completo

A partire dallo schema portante descritto in precedenza, è stato realizzato uno schema E/R completo che include tutte le entità e le relazioni coinvolte nel sistema. Lo schema completo rappresenta in modo esaustivo tutte le entità, le relazioni e gli attributi che caratterizzano il sistema di monitoraggio e controllo di missioni spaziali.

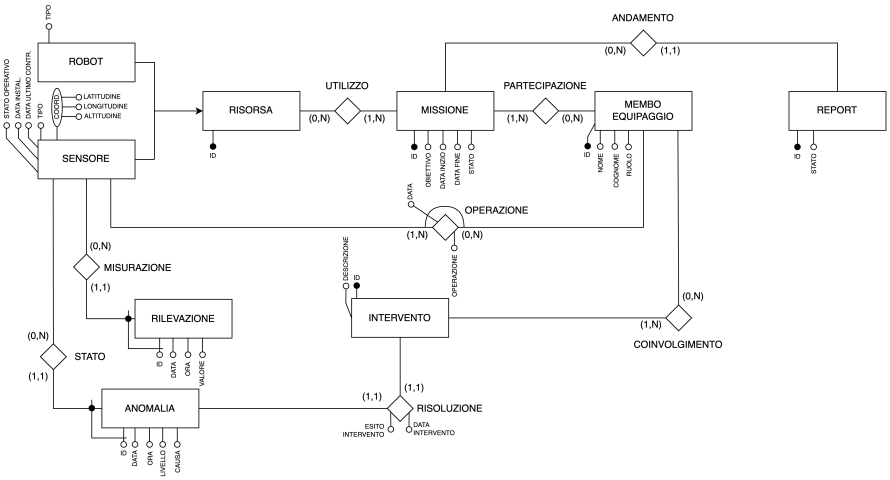


Figure 2: Modello E/R completo

Specifiche di Progettazione TODO

1.2 Progettazione Logica

La progettazione logica si articola in due fasi:

1. **Trasformazione**: in questa fase, vengono rimossi tutti i costrutti del modello Entità/Relazione (E/R) che non sono direttamente traducibili nel modello logico, come gli attributi composti e gli attributi multi-valore. Gli attributi multi-valore vengono associati direttamente all'entità di partenza, mentre gli attributi composti vengono scomposti nei loro componenti e, se necessario, trasferiti a una nuova entità collegata all'entità originale.

2. **Traduzione**: lo schema risultante dalla trasformazione viene convertito nel modello logico attraverso un insieme di regole predeterminate, che possono essere implementate anche tramite strumenti automatizzati. Questa fase non considera direttamente la semantica dei dati, ma si concentra sulla loro struttura.

1.2.1 Trasformazione

Durante la fase di trasformazione, vengono eliminati tutti gli attributi che non sono direttamente traducibili nel modello logico. Di seguito vengono descritti i casi specifici presenti nello schema:

- **Attributi multi-valore**: non sono presenti in questo caso, quindi non si rende necessaria alcuna operazione di trasformazione relativa a questa tipologia di attributi.
- **Attributi composti**: l'unico attributo composto identificato è **Coordinate**, associato all'entità *SENSORE*. Per conformarsi ai requisiti del modello logico, questo attributo è stato scomposto nei suoi componenti: **Latitudine**, **Longitudine** e **Altitudine**. Tali componenti sono stati direttamente associati all'entità di partenza senza creare una nuova entità.

Di seguito è riportato lo schema trasformato per l'entità *SENSORE*:

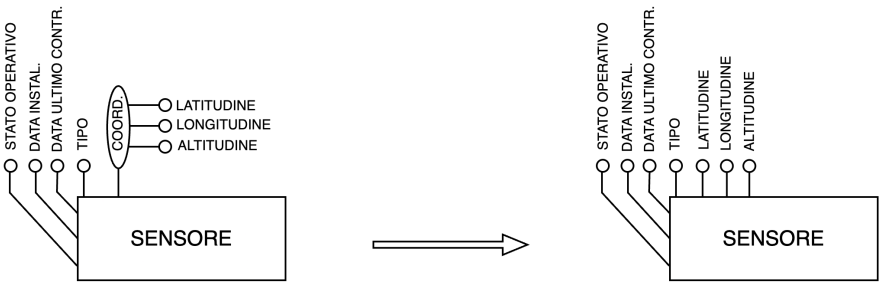


Figure 3: Entità *SENSORE* trasformata

1.2.2 Traduzione

TODO: DESCRIZIONE FASE TRADUZIONE

Traduzione Entità Ogni entità del modello E/R diventa una relazione/tabella.

- **Nome della tabella**: corrisponde al nome dell'entità.
- **Campi della tabella**: corrispondono agli attributi dell'entità.

Risultato della Traduzione delle Entità:

```
MISSIONI(ID, Obiettivo, Data_Inizio, Data_Fine, Stato);
MEMBI(ID, Nome, Cognome, Ruolo);
REPORT(ID, Stato);
INTERVENTI(ID, Descrizione);
ANOMALIE(ID, Data, Ora, Livello, Causa);
RILEVAZIONI(ID, Data, Ora, Valore);
ROBOT(ID, Tipo);
SENSORI(ID, Data_Installazione, Data_Ultimo_Controllo, Tipo,
   Stato_Operativo, Latitudine, Longitudine, Altitudine)
```

Traduzione Relazioni

1. Relazioni **N a N**
   - Ogni associazione N a N diventa una tabella con:
     - **Nome**: corrisponde al nome dell'associazione, al plurale.
     - **Campi**: includono gli identificatori delle due entità che collega, più eventuali attributi dell'associazione.
     - **Chiave primaria**: composta dalla coppia dei due identificatori.
     - **Vincoli di integrità referenziale**: garantiscono la consistenza con le entità collegate.

```
UTILIZZO_SENSORI (Sensore: Sensori, Missione: Missioni);
UTILIZZO_ROBOT (Robot: Robot, Missione: Missioni);
PARTECIPAZIONI (Missione: Missioni, Membro_Equipaggio:
   Membri_Equipaggio);
OPERAZIONI (Membro_Equipaggio: Membri_Equipaggio, Sensore
   : Sensori, Operazione, Data);
COINVOLGIMENTI (Membro_Equipaggio: Membri_Equipaggio,
   Intervento: Interventi)
```

2. Relazioni **1 a N**
   - Gli **attributi dell'entità lato 1** e gli **attributi della relazione** vengono aggiunti come campi all'entità lato N.
   - **Chiave primaria**: rimane quella dell'entità lato N.

- Questa scelta consente di ridurre il numero di tabelle, evitando join complessi a 3 tabelle.

```
REPORT (ID, Stato, Data, Missione: Missioni);
RILEVAZIONI (ID, Data, Ora, Valore, Sensore: Sensori);
ANOMALIE (ID, Data, Ora, Livello, Causa, Sensore: Sensori
   );
```

3. Relazioni **1 a 1**

- Ogni associazione 1 a 1 diventa una tabella con:
  - **Campi**: includono gli identificatori delle entità che collega, più eventuali attributi.
  - **Chiave primaria**: si sceglie l'identificatore dell'entità con cardinalità minima e partecipazione obbligatoria, per evitare valori NULL.

```
RISOLUZIONI (Intervento: Interventi, Anomalia: Anomalie,
   Esito_Intervento, Data_Intervento)
```

1.2.3 Modello E/R avanzato

Le gerarchie di generalizzazione/specializzazione non possono essere direttamente rappresentate nel modello logico relazionale, poiché quest'ultimo non prevede un costrutto equivalente. Per superare questa limitazione, il modello E/R è stato esteso con l'introduzione del costrutto di **generalizzazione/specializzazione**.

Il costrutto di generalizzazione può essere trasformato in schemi traducibili nel modello logico seguendo tre modalità principali.

Per questo progetto è stata adottata la modalità di **accorpamento della superclasse nelle sottoclassi**:

- **Eliminazione dell'entità padre**: l'entità padre (superclasse) viene eliminata dallo schema.
- **Eredità degli attributi e delle relazioni**: grazie alla proprietà dell'eredità, gli attributi, l'identificatore e le relazioni a cui partecipava l'entità padre vengono trasferiti integralmente alle entità figlie (sottoclassi).

Di seguito è riportato il risultato dell'accorpamento relativo alla superclasse *RISORSA* e alle sottoclassi *ROBOT* e *SENSORE*

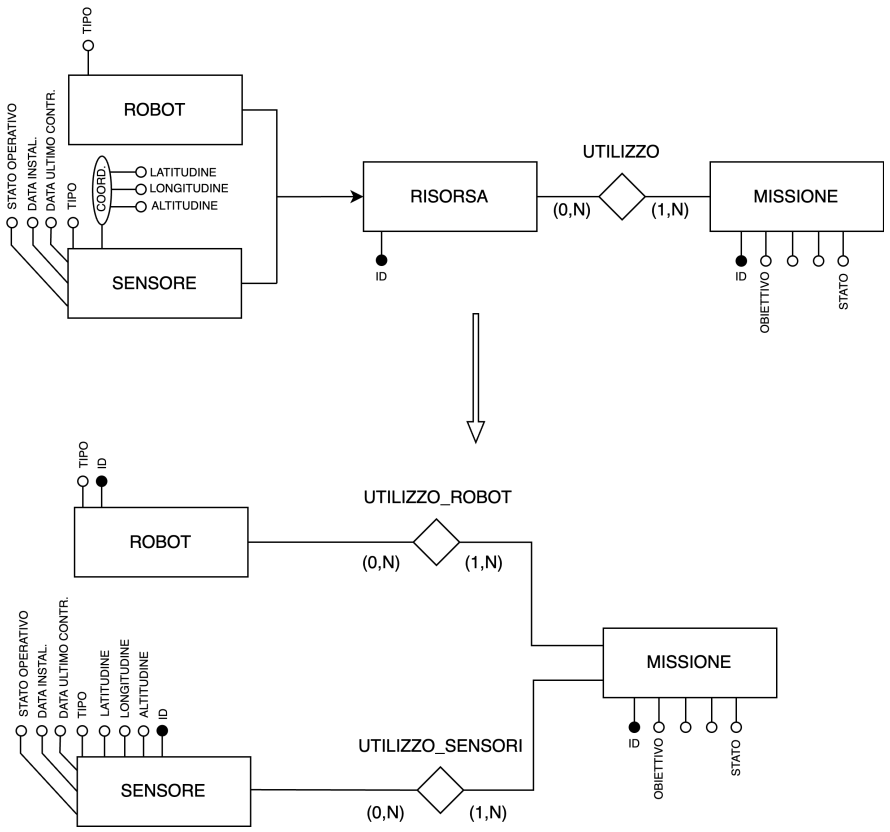


Figure 4: accorpamento di *RISORSA* in *ROBOT* e *SENSORE*

1.3 Progettazione Fisica

I tipi di dato utilizzati per la realizzazione di questo progetto sono:

- **DATE**: Utilizzato per tutte le date presenti.
- **INTEGER**: per tutti gli ID che quindi sono formati esclusivamente da numeri e non da lettere;
- **VARCHAR2(N)**: per tutti gli attributi di tipo testuali, come per esempio nome, cognome, descrizione, ecc..

1.3.1 Tabelle

```
-- Tabella MISSIONI
CREATE TABLE MISSIONI (
    ID INT,
    Obiettivo VARCHAR2(255) NOT NULL,
```

```
    Data_Inizio DATE NOT NULL, --NOT NULL o no ?
    Data_Fine DATE,
    Stato VARCHAR2(50) NOT NULL
);

-- Tabella MEMBRI
CREATE TABLE MEMBRI (
    ID INT,
    Nome VARCHAR2(100) NOT NULL,
    Cognome VARCHAR2(100) NOT NULL,
    Ruolo VARCHAR2(100) NOT NULL
);

-- Tabella SENSORI
CREATE TABLE SENSORI (
    ID INT,
    Data_Installazione DATE NOT NULL,
    Data_Ultimo_Controllo DATE,
    Tipo VARCHAR2(100) NOT NULL,
    Latitudine FLOAT NOT NULL,
    Longitudine FLOAT NOT NULL,
    Altitudine FLOAT NOT NULL
);

-- Tabella ROBOT
CREATE TABLE ROBOT (
    ID INT,
    Tipo VARCHAR2(100) NOT NULL
);

-- Tabella ANOMALIE
CREATE TABLE ANOMALIE (
    ID INT,
    Data DATE NOT NULL,
    Ora TIMESTAMP NOT NULL,
    Livello VARCHAR2(50) NOT NULL,
    Causa VARCHAR2(255) NOT NULL,
    Sensori INT
);

-- Tabella INTERVENTI
CREATE TABLE INTERVENTI (
    ID INT,
    Descrizione VARCHAR2(255) NOT NULL
);

-- Tabella RISOLUZIONI
CREATE TABLE RISOLUZIONI (
    Anomalie INT,
```

```
    Interventi INT,
    Esito_Intervento VARCHAR2(255),
    Data_Intervento DATE
);

-- Tabella RILEVAZIONI
CREATE TABLE RILEVAZIONI (
    ID INT,
    Data DATE NOT NULL,
    Ora TIMESTAMP NOT NULL,
    Valore FLOAT NOT NULL,
    Sensori INT
);

-- Tabella REPORT
CREATE TABLE REPORT (
    ID INT,
    Stato VARCHAR2(50) NOT NULL,
    Missioni INT
);

-- Tabella UTILIZZO_ROBOT
CREATE TABLE UTILIZZO_ROBOT (
    Robot INT,
    Missioni INT
);

-- Tabella UTILIZZO_SENSORI
CREATE TABLE UTILIZZO_SENSORI (
    Sensori INT,
    Missioni INT
);

-- Tabella COINVOLGIMENTI
CREATE TABLE COINVOLGIMENTI (
    Membri INT,
    Interventi INT
);

-- Tabella OPERAZIONI
CREATE TABLE OPERAZIONI (
    Membri INT,
    Sensori INT,
    Stato_Operativo VARCHAR2(50) NOT NULL,
    Operazione VARCHAR2(255)
);

-- Tabella PARTECIPAZIONI
CREATE TABLE PARTECIPAZIONI (
```

```
    Missione INT,
    Membri INT
);

-- Tabella SENSORI_MISSIONI
CREATE TABLE SENSORI_MISSIONI (
    Sensore_ID INT,
    Missione_ID INT,
    PRIMARY KEY (Sensore_ID, Missione_ID),
    FOREIGN KEY (Sensore_ID) REFERENCES SENSORI(ID),
    FOREIGN KEY (Missione_ID) REFERENCES MISSIONI(ID)
);

-- Tabella MEMBRI_MISSIONI
CREATE TABLE MEMBRI_MISSIONI (
    Membro_ID INT,
    Missione_ID INT,
    PRIMARY KEY (Membro_ID, Missione_ID),
    FOREIGN KEY (Membro_ID) REFERENCES MEMBRI(ID),
    FOREIGN KEY (Missione_ID) REFERENCES MISSIONI(ID)
);
```

1.3.2 Chiavi primarie

```
ALTER TABLE MISSIONI ADD CONSTRAINT PK_MISSIONI PRIMARY KEY (ID);
ALTER TABLE MEMBRI ADD CONSTRAINT PK_MEMBRI PRIMARY KEY (ID);
ALTER TABLE SENSORI ADD CONSTRAINT PK_SENSORI PRIMARY KEY (ID);
ALTER TABLE ROBOT ADD CONSTRAINT PK_ROBOT PRIMARY KEY (ID);
ALTER TABLE ANOMALIE ADD CONSTRAINT PK_ANOMALIE PRIMARY KEY (ID);
ALTER TABLE INTERVENTI ADD CONSTRAINT PK_INTERVENTI PRIMARY KEY (
    ID);
ALTER TABLE RILEVAZIONI ADD CONSTRAINT PK_RILEVAZIONI PRIMARY KEY
    (ID);
ALTER TABLE REPORT ADD CONSTRAINT PK_REPORT PRIMARY KEY (ID);
ALTER TABLE RISOLUZIONI ADD CONSTRAINT PK_RISOLUZIONI PRIMARY KEY
    (Anomalie, Interventi);
ALTER TABLE UTILIZZO_ROBOT ADD CONSTRAINT PK_UTILIZZO_ROBOT
    PRIMARY KEY (Robot, Missioni);
ALTER TABLE UTILIZZO_SENSORI ADD CONSTRAINT PK_UTILIZZO_SENSORI
    PRIMARY KEY (Sensori, Missioni);
ALTER TABLE COINVOLGIMENTI ADD CONSTRAINT PK_COINVOLGIMENTI
    PRIMARY KEY (Membri, Interventi);
ALTER TABLE OPERAZIONI ADD CONSTRAINT PK_OPERAZIONI PRIMARY KEY (
    Membri, Sensori);
ALTER TABLE PARTECIPAZIONI ADD CONSTRAINT PK_PARTECIPAZIONI
    PRIMARY KEY (Missione, Membri);
```

1.3.3 Chiavi esterne

```
ALTER TABLE ANOMALIE ADD CONSTRAINT FK_ANOMALIE_SENSORI FOREIGN
   KEY (Sensori) REFERENCES SENSORI(ID);
ALTER TABLE RISOLUZIONI ADD CONSTRAINT FK_RISOLUZIONI_ANOMALIE
   FOREIGN KEY (Anomalie) REFERENCES ANOMALIE(ID);
ALTER TABLE RISOLUZIONI ADD CONSTRAINT FK_RISOLUZIONI_INTERVENTI
   FOREIGN KEY (Interventi) REFERENCES INTERVENTI(ID);
ALTER TABLE RILEVAZIONI ADD CONSTRAINT FK_RILEVAZIONI_SENSORI
   FOREIGN KEY (Sensori) REFERENCES SENSORI(ID);
ALTER TABLE REPORT ADD CONSTRAINT FK_REPORT_MISSIONI FOREIGN KEY (
   Missioni) REFERENCES MISSIONI(ID);
ALTER TABLE UTILIZZO_ROBOT ADD CONSTRAINT FK_UTILIZZO_ROBOT_ROBOT
   FOREIGN KEY (Robot) REFERENCES ROBOT(ID);
ALTER TABLE UTILIZZO_ROBOT ADD CONSTRAINT
   FK_UTILIZZO_ROBOT_MISSIONI FOREIGN KEY (Missioni) REFERENCES
   MISSIONI(ID);
ALTER TABLE UTILIZZO_SENSORI ADD CONSTRAINT
   FK_UTILIZZO_SENSORI_SENSORI FOREIGN KEY (Sensori) REFERENCES
   SENSORI(ID);
ALTER TABLE UTILIZZO_SENSORI ADD CONSTRAINT
   FK_UTILIZZO_SENSORI_MISSIONI FOREIGN KEY (Missioni) REFERENCES
   MISSIONI(ID);
ALTER TABLE COINVOLGIMENTI ADD CONSTRAINT FK_COINVOLGIMENTI_MEMBRI
    FOREIGN KEY (Membri) REFERENCES MEMBRI(ID);
ALTER TABLE COINVOLGIMENTI ADD CONSTRAINT
   FK_COINVOLGIMENTI_INTERVENTI FOREIGN KEY (Interventi)
   REFERENCES INTERVENTI(ID);
ALTER TABLE OPERAZIONI ADD CONSTRAINT FK_OPERAZIONI_MEMBRI FOREIGN
    KEY (Membri) REFERENCES MEMBRI(ID);

ALTER TABLE OPERAZIONI ADD CONSTRAINT FK_OPERAZIONI_SENSORI
   FOREIGN KEY (Sensori) REFERENCES SENSORI(ID);

ALTER TABLE PARTECIPAZIONI ADD CONSTRAINT
   FK_PARTECIPAZIONI_MISSIONE FOREIGN KEY (Missione) REFERENCES
   MISSIONI(ID);

ALTER TABLE PARTECIPAZIONI ADD CONSTRAINT FK_PARTECIPAZIONI_MEMBRI
    FOREIGN KEY (Membri) REFERENCES MEMBRI(ID);
```

1.3.4 Vincoli di check

```
    ALTER TABLE SENSORI ADD CONSTRAINT CK_SENSORI_TIPO CHECK (Tipo
        IN ('Temperatura', 'Pressione', 'Gas', 'Radiazioni', '
       Geologia'));
```

```
ALTER TABLE OPERAZIONI ADD CONSTRAINT CK_OPERAZIONI_STATO
    CHECK (Stato_Operativo IN ('Attivo', 'Standby', '
    Manutenzione', 'Malfunzionante'));

ALTER TABLE ANOMALIE ADD CONSTRAINT CK_ANOMALIE_LIVELLO CHECK
    (Livello IN ('Bassa', 'Media', 'Alta', 'Critica'));

ALTER TABLE MISSIONI ADD CONSTRAINT CK_MISSIONI_STATO CHECK (
    Stato IN ('Pianificata', 'In␣corso', 'Completata', '
    Annullata'));
```

2 Ottimizzazione DB

2.1 Indici

Un **indice** in un sistema di gestione di database (DBMS) è una struttura dati organizzata che consente di individuare rapidamente un determinato record all'interno di un file di dati. Uno dei principali vantaggi dell'utilizzo degli indici è il miglioramento delle prestazioni delle query: gli indici permettono di ridurre il tempo necessario per cercare i dati, evitando una scansione completa della tabella.

Creare indici sui campi che vengono frequentemente utilizzati nei filtri delle query (ad esempio, con condizioni `WHERE`, `JOIN`, `ORDER BY` o `GROUP BY`) può velocizzare significativamente l'elaborazione delle richieste, ottimizzando il sistema nel suo complesso.

Tuttavia, è importante bilanciare l'uso degli indici per evitare costi aggiuntivi durante le operazioni di scrittura come `INSERT`, `UPDATE` e `DELETE`, poiché gli indici devono essere aggiornati ogni volta che i dati della tabella vengono modificati.

Tralasciando gli indici sulla chiave primaria, in quanto il DBMS crea un indice per ogni chiave primaria della tabella, abbiamo pensato di aggiungere questi indici:

```
CREATE INDEX idx_missioni_stato ON MISSIONI(Stato);
-- utile per filtrare missioni sullo stato

CREATE INDEX idx_membri_ruolo ON MEMBRI(Ruolo);
-- utile per filtrare sul ruolo dei membri

CREATE INDEX idx_robot_tipo ON ROBOT(Tipo);
-- utile per filtrare sul tipo di robot

CREATE INDEX idx_report_data ON REPORT(Data);

CREATE INDEX idx_rilevazioni_sensori_data ON RILEVAZIONI(Sensori,
   Data);
```

2.2 Concorrenza

Per la gestione della concorrenza abbiamo scelto di adottare il protocollo **2PL stretto**, una variante del **2PL**. Entrambi i protocolli si basano sul meccanismo del **lock**, che consente di gestire la concorrenza e garantire la **serializzabilità delle transazioni**.

Il suo funzionamento prevede:

- `lock()`: ogni oggetto è protetto da un lock;

- `read_lock()`: se una transazione vuole effettuare una lettura su uno oggetto. Più transazioni possono leggere contemporaneamente lo stesso oggetto (condivisione).

- `write_lock()`: è esclusivo e consente a una sola transazione di modificare l'oggetto per volta.

- `unlock()`: Ogni lock, una volta terminata l'operazione, deve essere rilasciato (unlock).

Gli oggetti possono trovarsi in tre stati: **libero**, **bloccato in lettura**, o **bloccato in scrittura**.

La scelta del **2PL stretto** rispetto al **2PL** è dovuta al fatto che il **2PL stretto** evita l'anomalia delle **letture sporche** (dirty reads), che possono verificarsi in altri approcci di gestione della concorrenza.

Il 2PL stretto prevede due fasi:

- **fase crescente**: è la fase in cui una transazione acquisisce mediante `read_lock` e `write_lock` tutte le risorse su cui dovrà effettuare operazioni di lettura e scrittura.

- **fase decrescente**: è la fase in cui attraverso l'`unlock` si vanno a rilasciare le risorse. In particolare questa fase nel caso di 2PL stretto può essere effettuata solo se la transazione termina con una commit o con un abort.

Questo approccio garantisce che le transazioni siano serializzabili, impedendo conflitti e mantenendo la consistenza dei dati.

2.3 Affidabilità

Il controllo di **affidabilità** in un sistema di basi di dati ha come obiettivo principale il **ripristino** dello stato corretto del sistema (recovery) in seguito a guasti accidentali o intenzionali, che possano compromettere la funzionalità del sistema stesso. I guasti possono essere legati sia a malfunzionamenti hardware (ad esempio, guasti su disco o memoria) che software (ad esempio, crash di applicazioni o errori di sistema).

Il sistema di affidabilità si basa sulla gestione delle **transazioni**, che sono le unità fondamentali delle operazioni nel database, garantendo **atomicità** (le transazioni sono eseguite in modo completo o non eseguite affatto) e **persistenza** (i dati delle transazioni devono essere memorizzati in modo permanente una volta che la transazione è stata completata correttamente).

2.3.1 Backup

Per garantire un alto livello di affidabilità, il nostro sistema di database implementa la strategia di backup RAID 1, che offre una soluzione di mirroring. In un sistema RAID 1, ogni dato scritto sul disco primario viene duplicato in tempo reale su un disco secondario, chiamato "mirror". Questa tecnica garantisce che, in caso di guasto di uno dei dischi, i dati siano ancora disponibili sull'altro disco, riducendo il rischio di perdita di informazioni e migliorando la disponibilità del sistema.

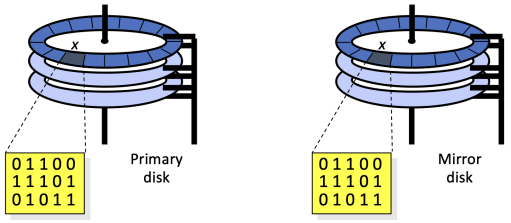


Figure 5: RAID 1 Mirroring

2.3.2 Recovery

Il gestore dell'affidabilità deve gestire l'esecuzione dei comandi transazionali di `begin transaction`, `commit`, `rollback` e tutte le operazioni di ripristino dopo i guasti.

Per poter effettuare ciò, il gestore deve possedere un file di log: un file presente su memoria stabile che registra tutte le operazioni svolte dalle transazioni nel loro ordine di esecuzione.

Il log è quindi una sorta di "diario di bordo" che, in un qualsiasi istante, permette di ricostituire il contenuto corretto della base dei dati a seguito di malfunzionamenti.

DP, B(T_1,-,-,-,-), U(T_1,-,qtaP,100,90), U(T_1,-,qtaC,NULL,10), C(T_1,-,-,-,-), B(T_2,-,-,-,-), CK(T_2), U(T_2,-,qtaP,90,70), U(T_1,-,qtaC,NULL,20), C(T_2,-,-,-,-), B(T_3,-,-,-,-), U(T_3,-,qtaP,100,90), U(T_3,-,qtaC,NULL,10), C(T_3,-,-,-,-), ...

Figure 6: File di log

Tecniche di recovery

- **Ripresa a freddo**: Nel caso di guasti hard sui dispositivi di memoria di massa (es. guasti ai dischi rigidi) si perde sia la memoria centrale che quella

secondaria, ma la memoria stabile (come i dispositivi di backup) rimane intatta. In queste situazioni, viene effettuata la ripresa a freddo (cold restart), che richiede un ripristino più approfondito, attingendo ai backup e ai log per recuperare i dati persi.

- **Ripresa a caldo**: 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 tali situazioni, viene effettuata la cosiddetta ripresa a caldo (warm restart).

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

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

3 Stored Procedures

4 Oracle APEX

4.1 Introduzione

Oracle APEX è un applicativo...

4.2 Home

Nella home...

4.3 Procedure

Tramite la seguente view...

4.4 Analisi dati

Questa view...