Joel Bosio

SM3201517

Esame di Basi di Dati 2025

Descrizione del progetto

Un ente che si occupa di gestire missioni esplorative nello spazio vuole organizzare e tracciare la struttura e l'attrezzatura delle varie vetture usate per le varie missioni.

Ogni missione, oltre ad una data, ha una piattaforma da cui avviene il lancio e i dati riguardo al razzo: la serie (ordinata) di stadi di cui è composto, la strumentazione e componentistica di ogni stadio, assieme ad eventuali allegati (schemi, misurazione, fotografie, video) relativi alla missione stessa. Solo foto e video sono disponibili al pubblico.

Un razzo può teoricamente avere qualsiasi numero di stadi, ma dato il numero ci devono essere esattamente quella quantità di stadi distinti e ordinati. Ogni stadio è progettato specificamente per la missione nel quale è impiegato: è impossibile che si ricicli perfettamente un design fino alla strumentazione di bordo e le versioni dei software, quindi gli stadi sono da considerarsi assolutamente unici. Il modello dello stadio è ciò che rappresenta la generica struttura del componente e questo è invece usato come categoria per tutti gli stadi date dalle iterazioni di uno stesso progetto.

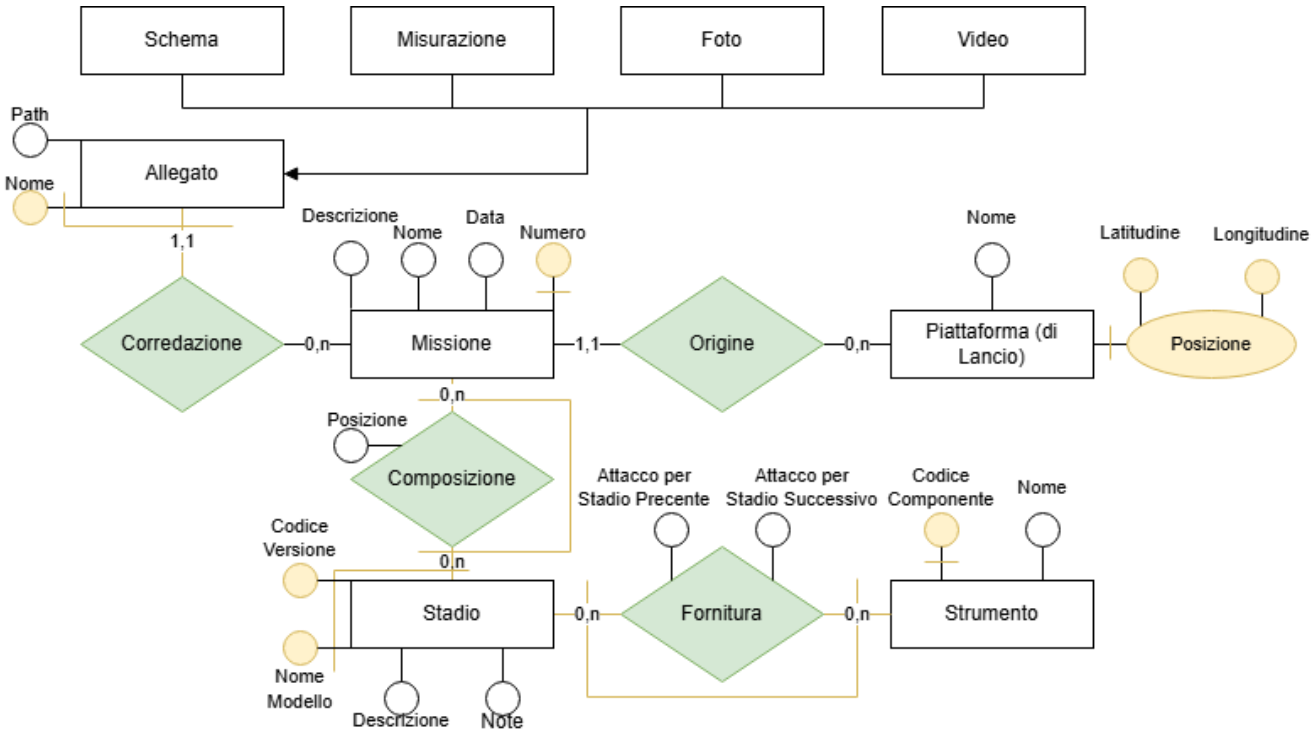
Uno stadio ha un nome di modello, un codice versione che indica l'iterazione di sviluppo del modello secondo uno standard numerico, una descrizione, delle note specifiche alla versione e un inventario di strumenti, oggetti e altre parti che contiene e/o che lo compongono, che può variare con le versioni dello stadio. Gli oggetti sono tracciati esternamente e noti solo attraverso un generico codice identificativo che li rappresenta solo in maniera astratta e un nome che funziona in maniera analoga, non individualmente per ogni copia (eg: due sedili identici avranno lo stesso codice identificativo e nome se sono dello stesso modello).

Uno stadio può avere a bordo qualsiasi numero di copie di qualsiasi oggetto: l'effettiva plausibilità della cosa è responsabilità degli ingegneri di competenza, in quanto non viene fatta distinzione tra strumenti portatili (torce, kit di pronto soccorso, manuali, etc) e componenti rigide non strutturali (telecamere esterne, paracadute, sedili, etc): il software richiesto traccia solo gli i pezzi in maniera astratta.

Ogni modulo può avere o meno un attacco per un modulo precedente e/o per uno successivo. Questi sono contati tra le parti a bordo e identificate da un flag, univoco per il pezzo rispetto al modulo. La presenza di un modulo successivo/precedente in un razzo che usa un dato stadio è subordinata alla presenza e compatibilità dei punti d'attacco, che devono combaciare tra gli stadi adiacenti.

Gli allegati della missione possono essere accessibili o meno al pubblico in base alla tipologia. Ne va inoltre tracciata la posizione su disco in quanto saranno salvati internamente al sistema per essere poi resi disponibili alle varie interfacce pubbliche e interne che faranno uso del sistema.

Schema concettuale



Glossario dei termini specifici

Termine	***Definizione***	***Sinonimi***	***Collegamenti***
Missione	Un'operazione di lancio di un razzo avvenuta in una certa data	Lancio	Allegati, Stadio, Piattaforma
Stadio	Un componente di un razzo, solitamente corrispondente ad una sezione orizzontale. Più stadi vengono "impilati" per comporre il razzo	Modulo	Strumento, Missione
Strumento	Un pezzo di attrezzatura, portatile o fissa, attaccata o caricata a bordo di un modulo	Pezzo	Stadio
Allegato	Un file relativo ad una missione. Possono essere di vari tipi		Missione
Piattaforma	Un sito di lancio per missioni. Corrisponde ad uno spazioporto o un altro comprensorio equiparabile	Piattaforma di lancio	Missione

Operazioni richieste

Descrizione	***Frequenza Stimata***
Dato un sito di lancio, l'estrazione di una pagina (skip/take) di missioni in ordine cronologico, partendo dalle più recenti	5000/giorno
Data una missione, l'estrazione della lista ordinata di moduli usati, includendo il nome del modello. Includere una stringa in formato JSON di coppie nomeStrumentazione/quantità per ogni modulo	2500/giorno
L'aggiunta di un modulo ad una missione con inserimento automatico del campo "posizione"	10/mese
La rimozione di uno stadio da una missione data una qualsiasi posizione con aggiornamento delle posizioni degli stadi rimanenti, se possibile	1/mese
L'estrazione di tutti gli allegati data una missione dato un flag che indica se i risultati sono intesi per il pubblico o per uso interno	1000/giorno
Impostazione di una certa quantità di copie di uno strumento da un modulo dato il suo codice univoco	25/mese

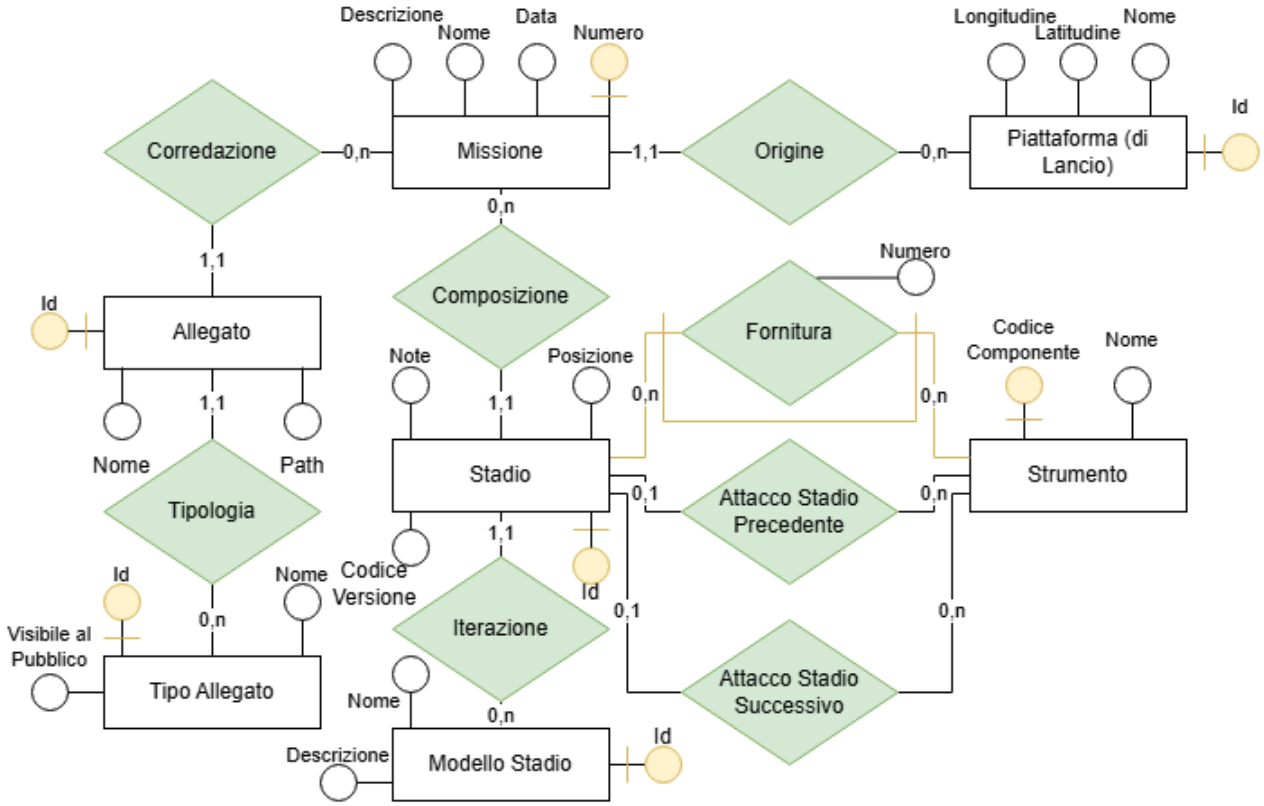
Ristrutturazione

- I vari tipi di allegati sono stati raggruppati in una sola tabella dove ogni allegato ha un tipo e ogni tipo ha un indicazione riguardo alla visibilità dei dati. I tipo sono fissi e verranno modificati pochissime volte se non addirittura mai, quindi si prevede che questa riorganizzazione non causi un aumento dei costi delle operazioni.
- Seguendo una logica simile i modelli di Stadio sono stati estratti ad una tabella separata per evitare duplicazione e i rischi che ne derivano per i campi "NomeModello" e "Descrizione". Per la nuova tabella è stato predisposto l'uso di un Id automatico progressivo come chiave primaria al posto che il campo NomeModello in quanto non conoscendo il formato dei nomi non è stato considerato sicuro affidarsi ai dati inseriti. In questo caso, anche se il numero non è alto, si prevede ci possano essere aggiunte alla tabella, detto ciò la modifica si apporta comunque per favorire un'organizzazione ordinata, coerente con le forme normali
- La relationship tra un modulo e la sua strumentazione è stata ristrutturata per identificare e gestire più facilmente gli attacchi per stadi precedenti/successivi. Alla relationship molti-a-molti per la strumentazione generica è stato aggiunto un campo "numero" che permette di gestire la molteplicità senza duplicazione di righe
- Per le relazioni con chiavi primarie composte poco maneggevoli (posizione, allegato, stadio e la tabella di sponda generata dalla relationship "Fornitura" tra Stadio e Strumento) e la nuova

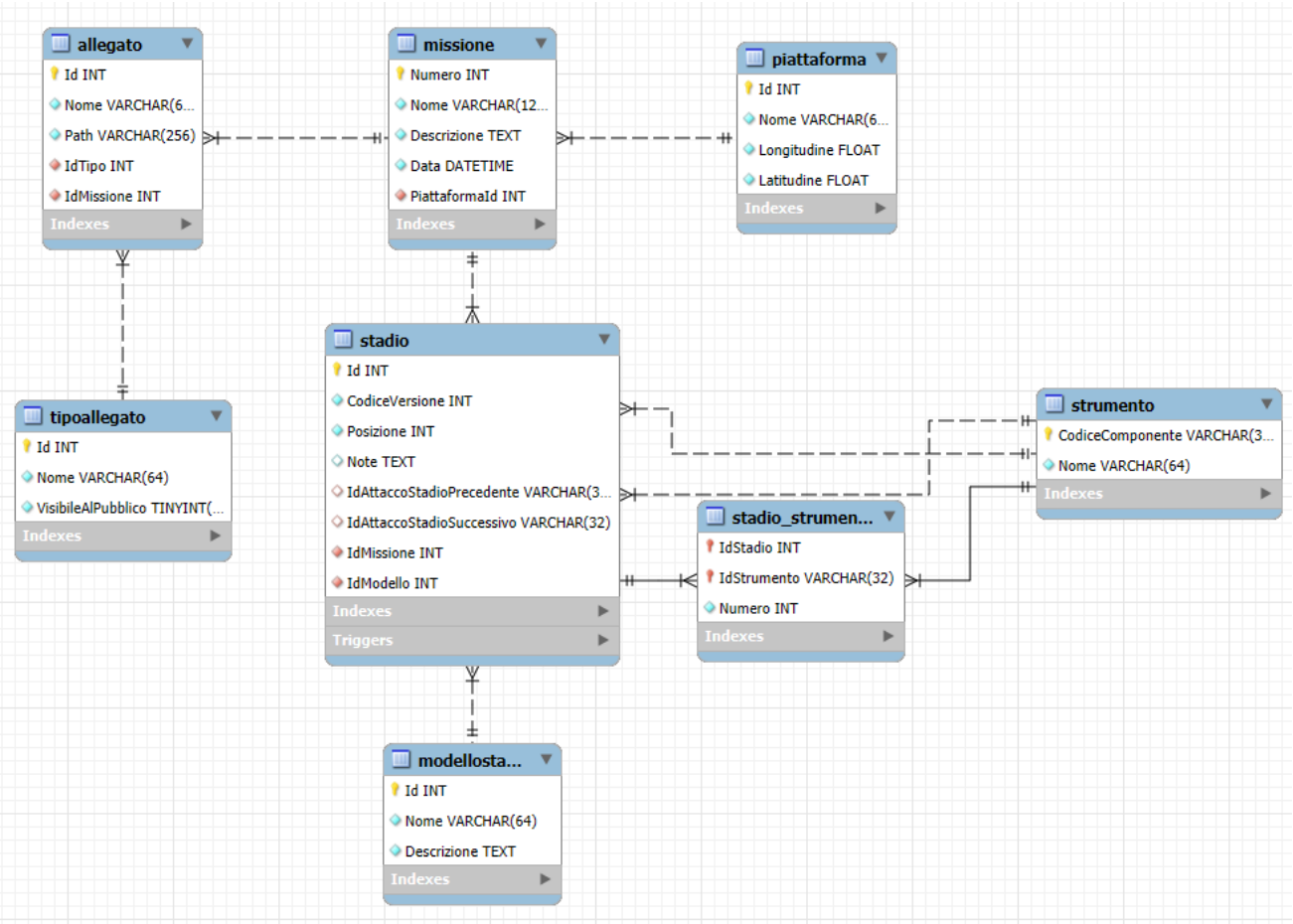
tabella TipoAllegato è stato predisposto l'uso di Id automatici progressivi per garantire univocità assoluta e semplicità di accesso ai dati

- A seguito delle ristrutturazioni sopra descritte non sono presenti ridondanze

Schema ristrutturato



Schema logico



- Piattaforma(Id, Nome, Longitudine, Latitudine)
- TipoAllegato(Id, Nome, VisibileAlPubblico)
- Missione(Numero, Nome, Descrizione, Data, PiattaformaId)
- Allegato(Id, Nome, Path, IdTipo, IdMissione)
- Stadio(Id, CodiceVersione, Posizione, Note, IdAttaccoStadioPrecedente, IdAttaccoStadioSuccessivo, IdMissione, IdModello)
- Strumento(CodiceComponente, Nome)
- stadio_strumento(IdStadio, IdStrumento, Numero)

Vincoli non esprimibili graficamente

- Gli n Stadi associati ad una Missione devono presentare esattamente una volta nel campo “Posizione” ogni numero da 1 a n: deve essere definito attraverso il campo un ordinamento coerente tra gli stadi
- Stadi adiacenti devono presentare lo stesso punto di attacco per potersi collegare correttamente (es: il secondo modulo deve presentare come attacco per il modulo successivo lo stesso pezzo usato come pezzo d’attacco per il modulo precedente del terzo stadio)
- Il numero di strumenti di un certo tipo associati ad un modulo non potrà mai essere negativo o nullo, in quanto nel caso il conteggio sia 0 non si traccia l’associazione

Normalizzazione

- Ogni calore è in un campo dedicato e non ci sono duplicati, quindi il database è già in 1NF
- È presente una chiava composta nella tabella di sponda usata per la relazione molti-a-molti tra Stadio e Strumento, ma il campo “Numero” è comunque dipendente da entrambi gli attributi che compongono la chiave, quindi essendo rispettata la 1NF ed essendo er il resto rispettati tutti i criteri della 2NF il database è in 2NF
- I criteri della 3NF sono rispettati quindi essendo il database in 2NF è anche in 3N

Dizionario Entità

Entità	***Descrizione***	***Attributi***	***Chiave Primaria***
Piattaforma	Una piattaforma di lancio di uno spazioporto	Id, Nome, Latitudine, Lognitudine	Id
Allegato	File associato ad una Missione	Id, Nome, Path	Id
TipoAllegato	Una tipologia di allegato (immagine, video, etc)	Id, Nome, VisibileAlPubblico	Id
Missione	Un razzo lanciato in una certa data per una certa missione	Nome, Descrizione, Data, Numero	Numero
Stadio	Un modulo di un razzo	Id, CodiceVersione, Pozione, Note	Id
ModelloStadio	Una tipologia e/o una catena di moduli di natura uniforme	Id, Nome, Descrizione	Id
Strumento	Un pezzo di attrezzatura fornito ad un modulo in una data quantità	CodiceComponente, Nome	CodiceComponente

Dizionario Relationship

Entità	***Descrizione***	***Componenti***	***Attributi***	***Chiave Primaria***
Origine	Una missione ha origine su una piattaforma di lancio	Piattaforma, Missione	N/A	

			Ogni Missione ha una Piattaforma di lancio	
Corredazione	Una Missione può essere corredata da un qualsiasi numero di Allegati	Missione, Allegato	N/A Ogni Allegato è associato ad una Missione	
Tipologia	Un Allegato ha una tipologia di file/contenuto	Allegato, TipoAllegato	N/A Ogni Allegato ha un TipoAllegato	
Composizione	Ogni Missione è caratterizzata dagli stadi utilizzati	Missione, Stadio	N/A Ogni Stadio è associato alla Missione di cui è stato parte	
Iterazione	Ogni stadio è un'interazione di un Modello	Stadio, ModelloStadio	N/A Ogni Stadio è derivato da un ModelloStadio	
Fornitura	Ogni Stadio può essere fornito con un certo numero di copie di uno Strumento	Stadio, Strumento	Stadio, Strumento, Numero	Stadio, Strumento
Attacco Stadio Precedente	Ogni Stadio può avere o meno un apparato dedicato al collegamento con uno Stadio precedente	Stadio, Strumento	N/A Ogni stadio può avere uno Strumento per attaccarsi ad uno stadio precedente	
Attacco Stadio Successivo	Ogni Stadio può avere o meno un apparato dedicato al collegamento con uno Stadio successivo	Stadio, Strumento	N/A Ogni stadio può avere uno Strumento per attaccarsi ad uno stadio successivo	

Tavola dei Volumi

Concetto	***Tipo***	***Volume Stimato***
Piattaforma	E	50

Allegato	E	5000
TipoAllegato	E	4
Missione	E	1000
ModelloStadio	E	100
Stadio	E	5000
Strumento	E	10000
Origine	R	1000
Corredazione	R	5000
Tipologia	R	5000
Composizione	R	5000
Iterazione	R	5000
Fornitura	R	20000
Attacco Stadio Precedente	R	4000
Attacco Stadio Successivo	R	4000

Note Implementative

- Si ritiene opportuno aggiungere indici secondari sui seguenti campi:
    - La foreign key riferita a Piattaforma nella tabella Missione, in quanto la maggior parte delle estrazioni avverrà in base a quel campo
    - Il campo Data della tabella missione, per permettere un ordinamento più efficiente per lo stesso motivo
    - La foreign key riferita a Missione nella tabella Stadio, in quanto la maggior parte delle estrazioni avverrà in base a quel campo
- Si considera ma NON si ritiene utile aggiungere indici secondari sulle foreign key riferite alle tabelle categoriali ModelloStadio e TipoAllegato rispettivamente nelle entità Stadio e Allegato in

quanto nonostante rappresentino un raggruppamento naturale dei dati non favorirebbero nessuna operazione di frequenza degna di nota

- Si definiscono Unique:
    - La coppia Nome/Path di Allegato

Codice SQL per l'imposizione di vincoli, chiavi e constraint aggiuntivi non banali:

```
ALTER TABLE `allegato`
  ADD UNIQUE KEY `uq_location` (`Nome`,`Path`),
  ADD KEY `fk_corredazione` (`IdMissione`); --*

ALTER TABLE `missione`
  ADD KEY `idx_data` (`Data`) USING BTREE,
  ADD KEY `fk_origine` (`PiattaformaId`) USING BTREE; --*

ALTER TABLE `stadio`
  ADD KEY `fk_composizione` (`IdMissione`) USING BTREE,
  ADD CONSTRAINT uq_missione_posizione UNIQUE (IdMissione, Posizione);

ALTER TABLE `stadio_strumento`
  ADD KEY `idx_stadio` (`IdStadio`) USING BTREE, --*
  ADD CONSTRAINT `chk_numero_positivo` CHECK (Numero > 0);
```

*Hash potrebbe essere meglio se supportato dal motore

--Constraint per assicurare che le posizioni degli stadi all'interno di una missione siano inserite in maniera coerente. Non si applicano restrizione in UPDATE per semplificare le operazioni di riordinamento nel caso fossero necessarie

```
CREATE DEFINER=`root`@`localhost` TRIGGER `trgchk_posizione_stadio` AFTER INSERT
ON `stadio` FOR EACH ROW BEGIN
DECLARE posizioni_ok BOOLEAN;

SELECT
    COUNT(DISTINCT Posizione) = MAX(Posizione)
        AND MIN(Posizione) = 1
INTO posizioni_ok FROM
    stadio
WHERE
    IdMissione = NEW.IdMissione;

  IF NOT posizioni_ok THEN
    SIGNAL SQLSTATE '45001'
      SET MESSAGE_TEXT = 'Una missione non può avere due stadi nella stessa
posizione!';
  END IF;
END
```

--Controllo di compatibilità con lo stadio precedente o successivo rispetto a quello inserito. Il controllo non avviene in UPDATE per lo stesso motivo di quello precendete. Inoltre non si rifiutano valori nulli per lo stesso motivo

```
CREATE DEFINER=`root`@`localhost` TRIGGER `trgchk_stadio_adj` BEFORE INSERT ON
`stadio` FOR EACH ROW BEGIN
  DECLARE next_precedente VARCHAR(32);
  DECLARE prev_successivo VARCHAR(32);
```

```
SELECT
    IdAttaccoStadioPrecedente
INTO next_precedente FROM
    stadio
WHERE
    IdMissione = NEW.IdMissione
        AND Posizione = NEW.Posizione + 1
LIMIT 1;

SELECT
    IdAttaccoStadioSuccessivo
INTO prev_successivo FROM
    stadio
WHERE
    IdMissione = NEW.IdMissione
        AND Posizione = NEW.Posizione - 1
LIMIT 1;

  IF (next_precedente IS NOT NULL AND
    next_precedente COLLATE utf8mb4_general_ci !=
    NEW.IdAttaccoStadioSuccessivo COLLATE utf8mb4_general_ci)
  OR (prev_successivo IS NOT NULL AND
    prev_successivo COLLATE utf8mb4_general_ci !=
    NEW.IdAttaccoStadioPrecedente COLLATE utf8mb4_general_ci)
  THEN
    SIGNAL SQLSTATE '45002'
      SET MESSAGE_TEXT = 'Lo stadio che si vuole inserire non ha attacchi
compatibili con gli stadi adiacenti già presenti';
  END IF;
END
```

Codice SQL per l'implementazione delle operazioni richieste:

--Fetch delle missioni per un sito di lancio:

```
DELIMITER $$

CREATE PROCEDURE GetMissioniBySitoLancio (
    IN sitoId INT,
    IN skipCount INT,
    IN takeCount INT
)
BEGIN
    SELECT Numero,Nome,Descrizione,Data
Data
    FROM missione
    WHERE PiattaformaId = sitoId
    ORDER BY missione.Data DESC
    LIMIT takeCount OFFSET skipCount;
END $$

DELIMITER ;
```

--Fetch dei moduli di una missione:

```
DELIMITER $$

CREATE PROCEDURE GetModuliByMissione (
    IN missioneId INT
)
BEGIN
    SELECT
        stadio.Id,
        stadio.Posizione,
        modello.Nome as Modello,
        (
            SELECT
                JSON_OBJECTAGG(strumento.Nome, stadio_strumento.Numero)
            FROM stadio_strumento
            JOIN strumento ON strumento.CodiceComponente =
stadio_strumento.IdStrumento
            WHERE stadio_strumento.Idstadio = stadio.Id
        ) AS Strumentazione
    FROM stadio
    INNER JOIN modellostadio AS modello ON stadio.IdModello = modello.Id
    WHERE stadio.IdMissione = missioneId
    ORDER BY stadio.Posizione ASC;
END $$

DELIMITER ;
```

--Aggiunta di modulo:

```
DELIMITER $$

CREATE PROCEDURE AggiungiStadioMissione (
    IN IdMissioneParam INT,
    IN IdModello INT,
    IN Versione INT,
    IN Note TEXT,
    IN IdAttaccoPrecedente VARCHAR(32)
)
BEGIN
    DECLARE nuovaPosizione INT;

    SELECT COALESCE(MAX(Posizione), 0) + 1
    INTO nuovaPosizione
    FROM stadio
    WHERE stadio.IdMissione = IdMissioneParam;

    INSERT INTO stadio (
        CodiceVersione,
        Posizione,
        Note,
        IdMissione,
        IdModello,
        IdAttaccoStadioPrecedente
    )
    VALUES (
```

```
        Versione,
        nuovaPosizione,
        Note,
        IdMissioneParam,
        IdModello,
        IdAttaccoPrecedente
    );
END $$

DELIMITER ;
```

--Cancellazione stadio:

```
DELIMITER $$

CREATE PROCEDURE RimuoviStadioMissione (
    IN IdMissioneTarget INT,
    IN PosizioneTarget INT
)
BEGIN
    DELETE FROM stadio
    WHERE IdMissione = IdMissioneTarget AND Posizione = PosizioneTarget;

    UPDATE stadio
    SET Posizione = Posizione - 1
    WHERE IdMissione = IdMissioneTarget AND Posizione > PosizioneTarget;
END$$

DELIMITER ;
```

--Estrazione Allegati:

```
DELIMITER $$

CREATE PROCEDURE EstraiAllegatiMissione (
    IN IdMissione INT,
    IN VisibilePubblico BOOL
)
BEGIN
    SELECT
        allegato.Id,
        allegato.Nome,
        allegato.Path,
        tipo.Nome AS TipoAllegato,
        tipo.VisibileAlPubblico
    FROM
        allegato
        INNER JOIN tipoallegato AS tipo ON allegato.IdTipo = tipo.Id
    WHERE
        allegato.IdMissione = IdMissione
        AND tipo.VisibileAlPubblico = VisibilePubblico;
END $$

DELIMITER ;
```

--Aggiornamento conteggio stumenti:

```
DELIMITER $$

CREATE PROCEDURE `ImpostaQuantitaStrumentoStadio`(
  IN IdModulo INT,
  IN IdStrumento VARCHAR(32),
  IN Numero INT
)
BEGIN
  IF Numero = 0
  THEN
    DELETE FROM stadio_strumento
    WHERE IdStadio = IdModulo AND IdStrumento = IdStrumento;
  ELSE
    INSERT INTO stadio_strumento (IdStadio, IdStrumento, Numero)
    VALUES (IdModulo, IdStrumento, Numero)
    ON DUPLICATE KEY UPDATE Numero = VALUES(Numero);
  END IF;
END$$

DELIMITER ;
```