MPS Sqlserver

⚙️ Architettura di SQL Server

L'architettura di SQL Server si basa su due motori principali che lavorano in modo sinergico:

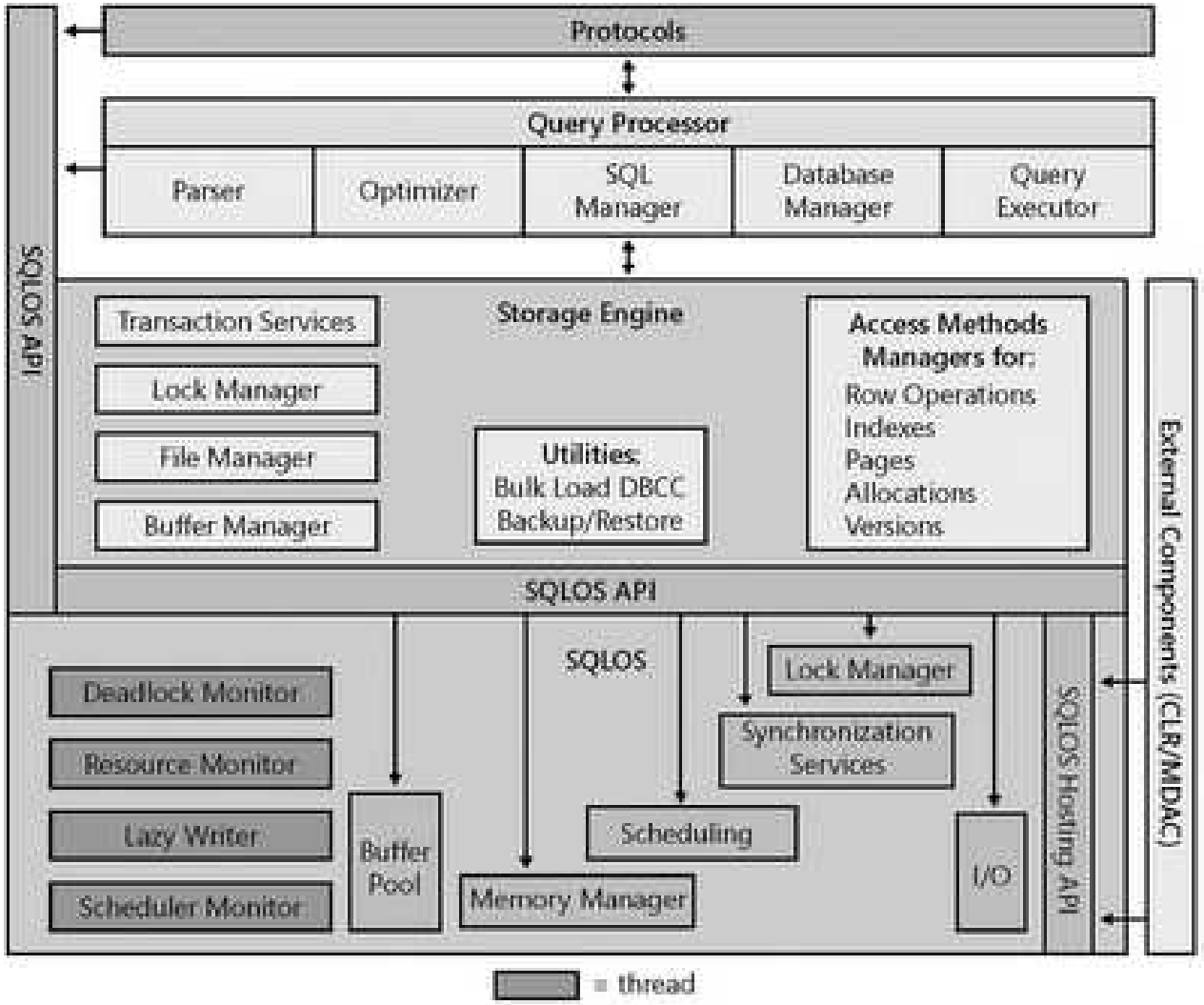
Database Engine (Motore del database): Questo è il nucleo di SQL Server. Gestisce la creazione, la manipolazione e il recupero dei dati. Si compone di:

Relational Engine (Motore relazionale): Accetta le query T-SQL (Transact-SQL) e determina il piano di esecuzione più efficiente. Si occupa dell'ottimizzatore di query, del gestore delle transazioni e della sicurezza.

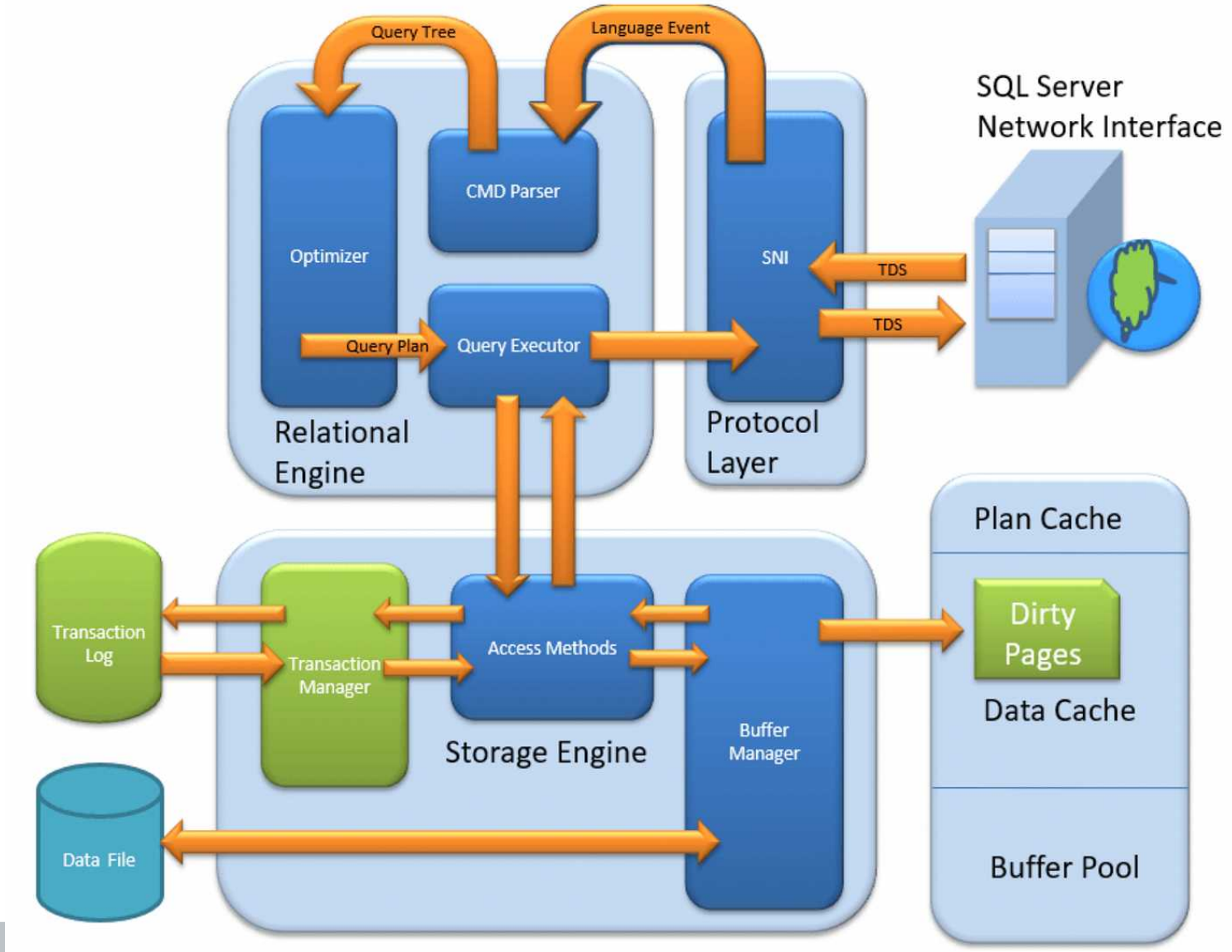
Storage Engine (Motore di archiviazione): Si occupa della gestione fisica dei dati, interagendo direttamente con i dischi per leggere e scrivere le informazioni. Include il gestore dei buffer, il gestore dei file e il gestore dei blocchi.

SQLOS (SQL Server Operating System): Un livello intermedio tra il motore di database e il sistema operativo sottostante (Windows o Linux). Gestisce le risorse di basso livello come la schedulazione dei thread, la gestione della memoria, l'I/O dei file e la sincronizzazione. Funziona come un sistema operativo virtuale per SQL Server, permettendo al Database Engine di funzionare in modo efficiente indipendentemente dal SO.

Architettura



Architettura



🧠 Aree di Memoria di SQL Server

SQL Server utilizza la memoria di sistema per ottimizzare le operazioni, riducendo la necessità di leggere e scrivere continuamente su disco, che è un'operazione lenta. Le aree di memoria principali sono:

Buffer Pool (Pool dei buffer): Questa è l'area di memoria più importante e più grande. Contiene le pagine di dati e le pagine di indice lette dai file di database (mdf/ndf). L'obiettivo è mantenere in memoria le pagine più utilizzate per un accesso rapido, minimizzando l'I/O su disco.

Quando una pagina viene richiesta, SQL Server la cerca prima nel Buffer Pool; se non la trova, la carica dal disco.

🧠 Aree di Memoria di SQL Server

Plan Cache (Cache dei piani): Qui vengono memorizzati i piani di esecuzione delle query. Quando una query viene eseguita per la prima volta, l'ottimizzatore crea un piano di esecuzione, che viene poi salvato nella Plan Cache.

Se la medesima query viene eseguita di nuovo, SQL Server può riutilizzare il piano esistente, evitando il costoso processo di ricompilazione migliorando così le prestazioni.

Procedure Cache (Cache delle procedure): Questa è un'area specifica del Plan Cache. Memorizza i piani di esecuzione per le stored procedure, i trigger e le funzioni.

🧠 Aree di Memoria di SQL Server

TempDB: Sebbene sia un database fisico su disco, una parte significativa delle sue operazioni avviene in memoria. TempDB viene utilizzato per:

- Oggetti temporanei creati dagli utenti (tabelle temporanee, variabili di tabella).
- Operazioni interne al motore di query (ad esempio, ordinamenti complessi o l'uso di cursori).

Workspace Memory (Memoria di lavoro): che viene utilizzata per eseguire operazioni specifiche delle query, come l'ordinamento (SORT), gli hash join e la costruzione di indici. La quantità di memoria allocata a queste operazioni è limitata e gestita eparatamente per evitare di monopolizzare tutte le risorse del sistema.

Schema

In SQL Server, uno schema è un container logico che raggruppa oggetti di database, come:

table,

view,

stored procedure,

index,

function...

È un livello di astrazione che separa la proprietà e la gestione degli oggetti all'interno di un database, fornendo una struttura organizzativa e di sicurezza.

Schema

Organizzazione: gli schema permettono di raggruppare logicamente oggetti correlati. Ad esempio, in un database per un'azienda, si possono creare schemi separati per diverse aree di business come Vendite, RisorseUmane e Contabilità. Questo evita conflitti di denominazione (es. Vendite.Clienti e RisorseUmane.Clienti possono coesistere).

Sicurezza: Gli schema sono la base per implementare un modello di sicurezza granulare. È possibile assegnare a utenti o ruoli specifici i permessi di accesso a un intero schema, piuttosto che dover gestire i permessi per ogni singolo oggetto.

Schema e Utenti

Proprietario dello Schema: Uno schema è di proprietà di un utente o di un ruolo del database. Il proprietario ha il pieno controllo sugli oggetti all'interno di quello schema.

Schema Predefinito (Default Schema): Ogni utente del database ha uno schema predefinito (default). Quando un utente crea un oggetto (es. una tabella) senza specificare uno schema, l'oggetto viene creato nel suo schema predefinito.

Lo schema predefinito per un nuovo database è **dbo** (database owner).

Questo sistema permette la separazione tra la proprietà degli oggetti (chi li ha creati/gestisce) e i permessi su di essi (chi può leggerli, modificarli, ecc.).

Schema

Creare un nuovo schema:

```
CREATE SCHEMA abc AUTHORIZATION MarioRossi;
```

Creare una tabella all'interno di uno schema:

```
CREATE TABLE abc.Dipendenti (
    ID INT PRIMARY KEY,
    Nome VARCHAR(100)
);
```

Assegnare un permesso a un utente su uno schema:

```
GRANT SELECT ON SCHEMA::Vendite TO GiovanniBianchi;
```

Oggetti di database

I tipi di oggetto più comuni includono:

Tabelle (Tables): Sono la base di un database, utilizzate per archiviare dati in formato righe e colonne.

Viste (Views): Tabelle “virtuali” basate su un set di risultati di una query. Le viste semplificano l'accesso a dati complessi e possono essere usate per la sicurezza, limitando l'accesso a determinate colonne o righe.

Stored Procedure: Blocchi di codice T-SQL precompilati e salvati nel database. Sono utili per eseguire operazioni complesse, migliorare le performance e aumentare la sicurezza.

Funzioni (Functions): Simili alle stored procedure, ma restituiscono un valore. Possono essere usate nelle query e non possono modificare i dati del database.

Oggetti di database

Indici (Indexes): Strutture che migliorano le performance delle query, rendendo più veloce la ricerca dei dati nelle tabelle.

Trigger: Codice T-SQL che si attiva automaticamente in risposta a un evento (es. un'operazione di INSERT, UPDATE o DELETE).

Vincoli (Constraints): Regole applicate alle colonne di una tabella per limitare il tipo di dati che possono essere inseriti:

- PRIMARY KEY: Garantisce l'univocità di ogni riga.
- FOREIGN KEY: Assicura l'integrità referenziale tra le tabelle.
- UNIQUE: Assicura che tutti i valori in una colonna siano diversi.
- CHECK: Limita i valori che possono essere inseriti in una colonna.
- NOT NULL: Forza una colonna a non accettare valori nulli.

Sinonimi (Synonyms): Alias per altri oggetti del database, utili per semplificare nomi lunghi o complessi.

Utilizzatori

Per creare un utente in SQL Server, occorrono le autorizzazioni a due livelli distinti:

Login: È un'entità di sicurezza a livello di istanza del server. Permette a una persona o un'applicazione di connettersi a un'istanza di SQL Server. I permessi assegnati a un login si applicano a livello di server (es. creare un database).

Utente: È un'entità di sicurezza a livello di database. Un utente è sempre associato a un login. Permette a un login di accedere e interagire con gli oggetti (tabelle, viste, ecc.) all'interno di un database specifico. I permessi assegnati a un utente si applicano solo a quel database.

Login

Ruoli a livello di istanza del server (per il Login):

sysadmin: Questo ruolo è il più potente e ha il controllo totale sull'intera istanza di SQL Server. I membri di questo ruolo possono creare, modificare ed eliminare qualsiasi login e utente.

securityadmin: Questo ruolo è specificamente designato per la gestione della sicurezza, inclusa la creazione e la gestione dei login e dei loro permessi a livello di server.

Utente

Ruoli a livello di database (per l'Utente):

db_owner: Questo ruolo è il più potente a livello di database. I suoi membri hanno il pieno controllo sul database e possono creare, modificare ed eliminare utenti, oltre che gestire i permessi su tutti gli oggetti al suo interno.

db_securityadmin: Questo ruolo è specifico per la gestione della sicurezza all'interno del database e i suoi membri possono gestire permessi, ruoli e utenti.

dbo (Database Owner): È un utente speciale che ha pieni permessi sul database.

In sintesi, per creare un nuovo utente in un database, un utente (o il login a cui è associato) deve avere il permesso CREATE USER su quel database.

Creazione utente

- Un amministratore del server (membro di sysadmin o securityadmin), in genere *sa*, crea un nuovo login a livello di istanza.
- Un amministratore del database (membro di db_owner o db_securityadmin) crea un nuovo utente nel proprio database, associandolo al login appena creato.

Questo processo in due fasi garantisce la gestione della sicurezza, separando le responsabilità tra l'amministrazione del server e quella dei singoli database.

Creazione utente

Supponendo di essere al primo avvio di sql server e potendo utilizzare *sa*, come creo un nuovo utente ed un nuovo database associato a quell'utente?

vedere procedura 001-CreateUser.sql

Modello Relazionale (in breve)

Il modello relazionale, ideato da Edgar F. Codd nel 1970, è un modello logico per l'organizzazione e la gestione dei dati in un database.

La forza del modello relazionale sta nel combinare un concetto matematico formalizzato (la relazione) con un'idea intuitiva (la tabella).

Modello Relazionale (in breve)

Relazione (Tabella): È la struttura fondamentale del modello relazionale. Una relazione è un insieme non ordinato di tuple (righe), e ogni tupla è un insieme non ordinato di attributi (colonne).

Attributo (Colonna): Rappresenta un tipo di dato specifico (es. Nome, Età, Codice Fiscale). Ogni attributo ha un dominio, che definisce l'insieme di valori che può assumere (es. il dominio dell'attributo "Età" è l'insieme dei numeri interi positivi).

Tupla (Riga, Npla, Record, Row): Rappresenta un'istanza o un record. Ogni tupla è unica all'interno della relazione.

Chiavi: Sono essenziali per stabilire le relazioni tra le tabelle e garantire l'integrità dei dati.

Chiave Primaria: Un attributo o un insieme di attributi che identifica in modo univoco ogni tupla in una relazione. Il suo valore non può essere nullo.

Chiave Esterna: Un attributo (o un insieme di attributi) in una tabella che fa riferimento alla chiave primaria di un'altra tabella. Serve a collegare le relazioni tra loro, creando un riferimento logico.

Modello Relazionale (in breve)

Indipendenza dei Dati: il modello relazionale separa la visione logica dei dati (come l'utente li vede e li manipola) dalla loro implementazione fisica (come sono memorizzati). Ciò significa che è possibile modificare la struttura di memorizzazione senza alterare le applicazioni che accedono ai dati.

Ridondanza Ridotta: grazie alla possibilità di collegare le tabelle tramite chiavi, si evitano dati duplicati, migliorando l'integrità e riducendo lo spazio di archiviazione.

Flessibilità e Semplicità: i dati vengono gestiti con operatori basati sull'algebra relazionale (come la selezione, la proiezione e il join), che sono implementati in linguaggi come l'SQL (Structured Query Language).

Modello Relazionale (in breve)

Per garantire la coerenza dei dati, il modello relazionale si basa su regole precise:

I valori degli attributi devono essere atomici: ovvero non possono essere ulteriormente scomponibili.

Ogni riga deve essere unica: non ci possono essere due righe identiche in una tabella.

L'ordine delle righe e delle colonne è irrilevante: non influisce sulla validità dei dati.

I nomi degli attributi devono essere univoci all'interno della stessa relazione.

Il valore della chiave primaria non può essere nullo.

Il valore di chiave esterna deve corrispondere a un valore di chiave primaria esistente nella tabella a cui fa riferimento, oppure essere nullo.

Modello Relazionale (in breve)

Le forme normali sono un insieme di regole che servono a ridurre la ridondanza dei dati e a migliorare l'integrità in un database relazionale.

Prima Forma Normale (1FN)

La 1FN è il requisito di base e ogni database relazionale deve rispettarla. Una tabella è in 1FN se:

Ogni colonna contiene valori atomici: Non ci sono valori multipli o liste all'interno di una singola cella.

Non ci sono gruppi di colonne ripetitive: Le colonne non devono ripetersi (es. Telefono1, Telefono2, Telefono3). Se un'entità ha più attributi simili, questi dovrebbero essere spostati in una tabella separata.

Modello Relazionale (in breve)

Seconda Forma Normale (2FN)

Una tabella è in 2FN se è già in 1FN e ogni attributo non chiave dipende completamente dalla chiave primaria intera. Questo si applica solo a tabelle con chiavi primarie composte (formate da più colonne).

L'anomalia da evitare è la dipendenza parziale. Se un attributo non chiave dipende solo da una parte della chiave primaria composta, la tabella non è in 2FN.

Esempio di tabella non in 2FN:

Ordine (ID_Ordine, ID_Prodotto, Quantità, Nome_Prodotto)

Dove Nome_Prodotto dipende solo da ID_Prodotto, non dalla chiave composta (ID_Ordine, ID_Prodotto).

Correzione: split della tabella

Ordine_Dettagli (ID_Ordine, ID_Prodotto, Quantità)

Prodotti (ID_Prodotto, Nome_Prodotto).

Modello Relazionale (in breve)

Terza Forma Normale (3FN)

Una tabella è in 3FN se è in 2FN e non ha dipendenze transitive. Si ha una dipendenza transitiva se un attributo non chiave dipende da un altro attributo non chiave.

In altre parole l'anomalia da evitareé: se A determina B e B determina C, allora C ha una dipendenza transitiva da A.

Esempio di tabella non in 3FN:

Dipendente (ID_Dipendente, Nome, ID_Reparto, Nome_Reparto)

Dove Nome_Reparto dipende da ID_Reparto, che a sua volta dipende da ID_Dipendente. Nome_Reparto ha una dipendenza transitiva.

Correzione split della tabella:

Dipendenti (ID_Dipendente, Nome, ID_Reparto)

Reparti (ID_Reparto, Nome_Reparto).

Modello Relazionale (in breve)

Forma Normale di Boyce-Codd (BCNF)

La BCNF è una versione più rigorosa della 3FN. Una tabella è in BCNF se, per ogni dipendenza funzionale X -> Y (dove X determina Y), X è una superchiave. Una superchiave è un insieme di attributi che identifica in modo univoco una tupla.

La BCNF elimina alcune anomalie che la 3FN potrebbe non coprire, specialmente in casi complessi con più chiavi candidate. Spesso 3FN è sufficiente, ma BCNF è la norma per database ben progettati.

L'importanza delle forme normali risiede nel fatto che garantiscono un design del database logico, coerente e senza ridondanze inutili, rendendo più efficiente la gestione dei dati nel tempo.

Modello Relazionale (in breve)

Esempio: Tabella "Istruttori-Corsi"

Immaginiamo una tabella che gestisce l'assegnazione di istruttori a corsi specifici in una scuola.

ID_Studente , ID_Istruttore, Nome_Istruttore, Nome_Corso

La chiave primaria è (ID_Studente, ID_Istruttore). Questo significa che ogni riga è identificata in modo univoco dalla combinazione di un determinato studente e un istruttore.

Dipendenze Funzionali:

{ID_Studente, ID_Istruttore} -> {Nome_Corso}: La combinazione di uno studente e un istruttore determina il corso che stanno seguendo insieme.

{ID_Studente, ID_Istruttore} -> {Nome_Istruttore}: La combinazione di uno studente e un istruttore determina il nome dell'istruttore.

{Nome_Corso} -> {ID_Istruttore}: Un corso è tenuto da un solo istruttore.

{ID_Istruttore} -> {Nome_Istruttore}: Ogni istruttore ha un nome.

Modello Relazionale (in breve)

È in 3FN: perchè non ci sono dipendenze transitive ovvero nessun attributo non-chiave dipende da un altro attributo non-chiave, ma non è in BCNF:

La dipendenza Nome_Corso -> ID_Istruttore non rispetta la regola BCNF.

La regola dice che per ogni dipendenza X -> Y, X deve essere una superchiave.

In questo caso, Nome_Corso non è una superchiave per la tabella, ma determina un'altra colonna (ID_Istruttore).

Questo crea una potenziale ridondanza e anomalia. Ad esempio, se due studenti diversi seguono lo stesso corso, il Nome_Istruttore e ID_Istruttore verranno ripetuti.

Modello Relazionale (in breve)

Per portare la tabella in BCNF, è necessario eliminare la dipendenza che causa il problema, creando una nuova tabella per l'informazione ridondante.

Tabella 1: Istruttori_Corsi che gestisce la relazione tra corsi e istruttori.

ID_Corso (Chiave Primaria), ID_Istruttore (Chiave Esterna a Istruttori)

Tabella 2: Iscrizioni che gestisce le iscrizioni degli studenti ai corsi.

ID_Studente, ID_Corso

La relazione tra istruttori e corsi è ora gestita in una tabella separata e ogni dipendenza funzionale è rispettata.

SQL

SQL, acronimo di Structured Query Language, è un linguaggio ””standard”” per la gestione e l'interazione con i database relazionali. Non è un linguaggio di programmazione generico, ma un linguaggio dichiarativo, il che significa che l'utente specifica "cosa" vuole fare, e il database si occupa di determinare "come" farlo.

È il linguaggio universale utilizzato per creare, interrogare, modificare e controllare la struttura e i dati all'interno di un database. Viene utilizzato per definire schemi di database (CREATE, ALTER), manipolare i dati (INSERT, UPDATE, DELETE), recuperare informazioni (SELECT) e gestire i permessi degli utenti (GRANT, REVOKE).

La sua forza risiede nella sua capacità di operare su insiemi di dati, rendendolo estremamente efficiente per l'elaborazione di grandi volumi di informazioni.

È alla base di tutti i principali sistemi di gestione di database relazionali (RDBMS) come SQL Server, MySQL, PostgreSQL e Oracle...

T-SQL

Il T-SQL (Transact-SQL) è l'estensione proprietaria del linguaggio SQL di Microsoft per SQL Server.

A differenza dell'SQL standard, che si concentra sulle query dichiarative, il T-SQL aggiunge funzionalità procedurali, di gestione degli errori e nuovi tipi di dato.

SQL

L'SQL (Structured Query Language) di SQL Server, come l'SQL standard, può essere suddiviso in diversi sotto-linguaggi, ognuno con uno scopo specifico.

1. Data Definition Language (DDL) che viene usato per definire la struttura del database e dei suoi oggetti, non manipola i dati, ma gestisce lo schema.

CREATE: Crea oggetti nel database (es. CREATE TABLE, CREATE VIEW, CREATE INDEX).

ALTER: Modifica la struttura di un oggetto esistente (es. ALTER TABLE per aggiungere o rimuovere una colonna).

DROP: Cancella un oggetto dal database (es. DROP TABLE, DROP VIEW).

SQL

2. Data Manipulation Language (DML) che è usato per manipolare i dati all'interno degli oggetti del database, come tabelle e viste.

INSERT: Inserisce nuove righe in una tabella.

UPDATE: Modifica i valori delle righe esistenti.

DELETE: Rimuove righe da una tabella.

SELECT: Recupera i dati dal database in base a criteri specificati.

3. Data Control Language che gestisce i permessi e il controllo di accesso agli oggetti del database.

GRANT: Concede permessi agli utenti (es. GRANT SELECT su una tabella).

REVOKE: Revoca i permessi precedentemente concessi.

DENY: Nega esplicitamente i permessi, impedendo a un utente di ereditarli da un ruolo.

4. Transaction Control Language che controlla le transazioni, ovvero sequenze di istruzioni DML eseguite come una singola unità logica. Questo assicura che tutte le modifiche avvengano con successo o che nessuna venga applicata.

COMMIT: Salva in modo permanente tutte le modifiche fatte in una transazione.

ROLLBACK: Annulla tutte le modifiche fatte nella transazione, ripristinando lo stato precedente.

SAVEPOINT: Imposta un punto di salvataggio all'interno di una transazione, permettendo di fare un rollback parziale.