Cos'è un database relazionale. 2
Il Modello dei Dati 2
Indipendenza fisica e logica e i livelli di descrizione dei dati 2
Progettare un Database (progettazione concettuale) 3
Schema Logico 4
Chiavi ed attributi di una relazione 5
Traduzione dei diagrammi E|R di uno schema concettuale in un modello relazionale 5
Regole di integrità 6
Integrità di entità. 6
Integrità di riferimento. 6
L'SQL (Structured Query Language) e Transact 8
1. Linguaggi di interrogazione Dati 8
I comandi SQL 9
DDL (Data Description Language) 9
DML (Data Manipulation Language) 9
DCL (Data Control Language) 9
Accedere a management studio 10
DML 11
Elements of a SELECT Statement 11
Logical Query Processing 11
Retrieving Columns from a Table or View 12
Displaying Columns 12
Using Calculations in the SELECT Clause 13
Querying Multiple Tables 14
The FROM Clause and Virtual Tables 14
Join Terminology: Cartesian Product 14
Overview of Join Types 15
Inner Join Syntax 15
Outer Join Examples 16
Filtering Data in the WHERE Clause With Predicates 17
WHERE Clause Syntax 17

Cos'è un database relazionale.

UNA RAPPRESENTAZIONE DELLA REALTÀ

Ma perché usare un Database relazionale?

I motivi sono tanti, fra questi si segnala:

- **MIGLIORE GESTIONE DELL'INSERIMENTO, MODIFICA E CANCELLAZIONE DEI DATI.**
- **RIDUZIONE DELLA DUPLICAZIONE DEI DATI.**
- **POSSIBILITÀ DI INTERROGARE I DATI .**

Il Modello dei Dati

Un modello di dati è uno strumento concettuale che consente al progettista di attribuire un certo significato (o interpretazione) ai dati e di manipolare i dati stessi. Si assegna una struttura ai dati attraverso appositi meccanismi di strutturazione previsti dal modello[2]

Il dato non dà di per se informazioni se non gli si fornisce una chiave d'interpretazione. Il modello di dati è uno strumento concettuale tramite il quale si può acquisire conoscenza da un insieme di dati altrimenti insignificante. L'organizzazione dei dati all'interno di questo schema attribuisce al dato una interpretazione che nel caso del modello relazionale viene definita come **rappresentazione della realtà di interesse** (Codd).

Indipendenza fisica e logica e i livelli di descrizione dei dati

Architettura a tre livelli di un DMBS

Lo schema logico descrive la struttura della base di dati non facendo nessun riferimento alla sua organizzazione fisica o al modo in cui vengono memorizzati i dati nelle memorie secondarie.

Lo schema interno è la descrizione di come sono organizzati fisicamente i dati nelle memorie di massa, e di quali strutture dati ausiliarie sono previste per facilitarne l'uso.

Lo schema esterno è una descrizione di come appare la struttura della base di dati ad una certa applicazione. Questo schema definisce quelle che si chiamano le viste, ovvero quelle porzioni della base di dati cui l'utente direttamente o tramite le applicazioni può accedere. In generale esistono più schemi esterni, uno per ogni gruppo omogeneo d'utenti. Essi però non sono indipendenti, poiché gli oggetti in comune hanno una rappresentazione unica nella base di dati e quindi le modifiche si riflettono su tutti gli utenti che ne fanno uso.

Fra questi livelli di descrizione dei dati devono esistere delle corrispondenze che vengono utilizzate dai DBMS per convertire le operazioni sugli oggetti virtuali accessibili da uno schema esterno in quelle sugli oggetti dello schema logico e, quindi, sui dati realmente presenti nel sistema, memorizzatisecondo lo schema interno. Questi schemi sono gestiti dal DBMS e non dai programmi applicativi i quali, per accedere alla base di dati, comunicano al sistema con particolari accorgimenti a quale schema fanno riferimento.

Con questo sistema dei tre livelli di descrizione dei dati si garantisce sia l'indipendenza fisica, perché i programmi applicativi fanno riferimento allo schema esterno che non contiene informazioni sul modo in cui i dati sono organizzati fisicamente, sia l'indipendenza logica, perché i programmi fanno riferimento allo schema esterno e non allo schema logico.

Progettare un Database (progettazione concettuale)

Aver chiaro lo scopo per cui si vuole costruire la Base di Dati.
Conoscere la realtà di interesse.

E' logico che occorre trarre un effettivo beneficiodalla realizzazione di una base di dati per gestire efficacemente la propria realtà di interesse.

Per conoscere la realtà di interesse occorre realizzare tutta una serie di attività che costituiscono la progettazione vera e propria. In particolare occorre realizzare:

- **Una analisi conoscitiva della realtà di interesse**

- **Codificare questa realtà in uno schema (progettazione concettuale)**

- **Rappresentare la realtà secondo il modello relazionale (progettazione logica).**

Entità-Relazioni.

Gli elementi essenziali di questo modello sono:*Dott. Maurizio Boghetto Introduzione a SQL Server* 7

Come si vede le entità sono rappresentate graficamente da quadrati mentre le relazioni da rombi. A volte le relazioni sono rappresentate da delle semplici linee**.**

Una volta realizzato lo schema concettuale lo si codificherà secondo le specifiche del modello relazionale.

Attraverso lo schema ER si arriva a comporre una rappresentazione della realtà oggetto del nostro interesse. Questa realtà così codificata dovrà essere trasformata secondo le regole di un modello di rappresentazione dei dati (in questo caso il modello Relazionale).

I metodi adottati per realizzare una progettazione concettuale sono solitamente l'incontro di questi due:

1. **Top-Down**. Si parte da una situazione estremamente sintetica per poi passare a gradi di analiticità maggiori fino a realizzare uno schema adeguatamente rappresentativo della nostra realtà.

2. **Bottom-Up**. Si parte da una situazione estremamente analitica che viene passo passo sintetizzata fino a realizzare uno schema adeguatamente rappresentativo della nostra realtà.

Schema Logico

La fase della progettazione concettuale dei dati porta a definire lo Schema Concettuale.

Il modello concettuale cerca di dare una struttura ai dati evitando il più possibile i dettagli realizzativi, per cui con lo Schema Concettuale si otterrà un documento di base che descrive la struttura del sistema informatico in maniera astratta, vale a dire astratta dal modello concettuale prescelto. Quindi, è un documento che guiderà i progettisti, nella fase della progettazione logica, alla realizzazione dello Schema logico secondo un particolare modello scelto. Un modello dei dati può essere definito come *un insieme di concetti, descritti tramite un preciso formalismo, il cui scopo è quello di permettere una rappresentazione ed una manipolazione delle informazioni che costituiscono il mondo della realtà*

Il modello scelto per realizzare lo schema concettuale è quello relazionale elaborato da Codd nel 1970. Si tratta di un modello matematico per la descrizione dello schema logico in maniera indipendente dalla realizzazione fisica il quale trae le sue origini dalla teoria degli insiemi.

Alla base del modello relazionale si trova la relazione detta anche tabella o schema poiché questa è costituita su di una struttura di righe dette tuple e colonne dette domini della tabella.

Le testate della tabella Ai si dicono **attributi,** mentre il numero n dei domini viene detto **grado o arietà** della tabella**.** Le tabelle che hanno un solo attributo si dicono unarie, quelle con due attributi binarie, con tre ternarie e via dicendo. Il numero delle tuple si dice **cardinalità** (m+1).

Le proprietà delle relazioni possono essere così riassunte[7]:

• *I valori di ogni colonna sono fra loro omogenei.* I valori di un attributo appartengono allo stesso dominio (interi, stringhe di caratteri, ecc..). Quindi, si può affermare che gli attributi rappresentano l'uso dei domini in una determinata tabella

• *l'ordinamento delle colonne è irrilevante.* Poiché sono sempre identificate per nome e non per posizione

• *l'ordinamento delle righe è irrilevante.* Poiché queste sono identificate per contenuto e non per posizione

Lo schema di una tabella R(A) è un'espressione che contiene il nome della tabella seguito dall'elenco di tutti gli attributi della tabella stessa. Può essere sia l'intestazione della tabella sia questa espressione : Prodotto (codice, descrizione, giacenza)

Chiavi ed attributi di una relazione

Il concetto di chiave viene elaborato nella teoria relazionale poiché le tabelle devono essere costruite in modo tale che ogni tupla deve essere distinta per mezzo di uno o più attributi. Quindi, scopo delle chiavi è identificare univocamente queste tuple. Si distingue fra superchiave, chiave candidate e chiave primaria. La **superchiave** è l'insieme degli attributi che identificano univocamente una tuple. La **chiave candidata** è in numero minimo di attributi necessari per identificare univocamente una tuple. Può essere definita come *una superchiave dalla quale possono essere eliminati attributi senza distruggere la proprietà della identificazione univoca della tuple. Per cui una chiave candidata è una superchiave con la proprietà di non ridondanza*[8].

Fra le chiavi candidate, che possono essere più di una, se ne sceglierà una che ha il minor numero di attributi e che, a parità di attributi, ha il minor numero di caratteri. Questa è la **chiave primaria** e sarà la chiave scelta fra le chiavi candidate a rappresentare univocamente una tuple.

Rispetto alla chiave della tabella si possono distinguere gli **attributi primi**, cioè quelli che fanno parte di almeno una chiave candidata, e i restanti **attributi non primi**. Infine, quegli attributi che possono costituire la chiave di altre tabelle sono detti **chiave esterna** e permettono di stabilire associazioni fra le tabelle che rappresentano diverse entità.

Però dopo tutto questo parlare è meglio che vi faccia vedere una tabella.

Traduzione dei diagrammi E|R di uno schema concettuale in un modello relazionale

Si può tradurre abbastanza agevolmente gli schemi E\R secondo le strutture delle basi di dati relazionali seguendo, però, alcune regole.

Entità e gerarchie IS-A

Le **entità** degli schemi E|R vengono tradotte in altrettante tabelle. Gli attributi delle entità vengono riportati nelle colonne mentre le occorrenze delle entità saranno tradotte in tuples.

Associazioni 1 a 1 e 1 a n

Per tradurre il legame tra due entità che viene realizzato da un'associazione 1 a 1, si deve inserire tra gli attributi di almeno una delle entità la chiave esterna, cioè quell'attributo o insieme di attributi che identificano l'entità ad essa collegata. Le associazioni 1 a n sono tradotte in modo analogo. Occorre però prestare attenzione a non inserire però **attributi ripetuti**, siano chiavi esterne o attributi normali, perché possono essere causa di problemi nella gestione delle tabelle. Ad esempio, in una relazione 1 a n fra due entità, *cliente* e *numero d'ordine,* inserisco la chiave primaria della tabella cliente nella tabella ordini divenendo chiave esterna. Se facessi l'inverso, ovvero se inserissi come chiave esterna nella relazione clienti la chiave della tabella ordini, dovrei inserire per ogni cliente tutti gli ordini creando molte tuples. Le associazioni tra entità, del tipo 1 a 1 o 1 a n, riportate sui diagrammi E\R possono aver attributi propri. Questi attributi possono essere riportati in altrettante colonne di una delle tabelle in cui sono state tradotte le entità.

Associazioni n a m

Per tradurre le associazioni n a m (molti a molti) è necessario introdurre una nuova tabella. Questo procedimento, sconsigliato per le associazioni 1 a 1 e 1 a n , è la strada obbligata per questo tipo di associazione (poiché col metodo precedente si avrebbe una grande proliferazione di attributi multipli). La nuova tabella includerà gli attributi propri della associazione e le chiavi esterne che permettono i collegamenti con le due tabelle traduzione delle entità che tramite essa sono associate. La chiave primaria della nuova tabella è costituita dalla coppia primaria di chiavi esterne. Si spezza così un'associazione m a n in due associazioni 1 a n.

- **Le entità**: L'entità è un "qualcosa" che nel contesto in esame ha una certa importanza. Ad esempio, se si vuol studiare una realtà scolastica, un'entità descriverà lo studente, del quale si potranno considerare le singole proprietà che interessano (detti attributi), come nome cognome, facoltà cui è iscritto, esami sostenuti e punteggio. Un'altra entità potrà essere il corso caratterizzato da un suo codice, descrizione numero ore annue ecc.

- **Gli attributi**: Con attributi si indicano dei dati che appartengono ad una entità.
- **Le Associazioni**: Indicano il tipo di relazioni fra le varie entità.
- **Le Relazioni**: Sono le relazioni fra le diverse entità. **Anche le relazioni possono avere i loro attributi**

Regole di integrità

Il modello relazionale prevede due regole generali di integrità. Queste sono **la integrità di entità** e **la integrità referenziale** (o di riferimento).

Integrità di entità. *Se A è un attributo della relazione R e partecipa alla chiave primaria, non può avere valori Null. Questo vuol dire che tutti gli attributi che partecipano alla chiave primaria non possono essere definiti su di un dominio che contempli valori Null.*

Integrità di riferimento. *Se nella relazione R la chiave primaria è rappresentata dall'attributo (o insieme di attributi) A, definito sul dominio D, allora in tutte le relazioni Sj, in cui appare come attributo A, le tuples devono contenere un valore di A definito sul dominio D oppure un valore Null. L'attributo A nelle relazioni Sj prende il nome di chiave esterna.*

Per una chiave esterna va verificato se :

• Può assumere valori Null

• Cosa succede nella tabella collegata tramite chiave esterna se viene cancellato o modificato un valore della chiave primaria della tabella che ad essa si collega se questo valore è presente (nella tabella collegata) in una sua occorrenza.

La possibilità che ha una chiave esterna di assumere valori Null dipende dai tipi di Associazione. Se, ad esempio, l'associazione è quella fra Impiegato e Reparto la chiave esterna corrispondente al reparto può assumere il valore Null (un impiegato non è assegnato ad un reparto), mentre nella associazione Fattura - Movimenti la chiave esterna corrispondente alla fattura non può mai assumere il valore Null (un movimento non può esistere senza fattura).

Gli effetti dell'aggiornamento o della cancellazione di una chiave primaria, presente come chiave esterna in altre tabelle, si possono ricondurre a tre casi :

Effetto cascata. Una cancellazione o un aggiornamento della chiave primaria provoca una cancellazione o un aggiornamento delle occorrenze presenti nelle tabelle collegate tramite chiave esterna.

Effetto restrizione. La cancellazione o l'aggiornamento non devono essere permessi se sono presenti occorrenze per il valore considerato nelle tabelle collegate da chiave esterna.

Effetto annullamento. La cancellazione o l'aggiornamento di un valore della chiave primaria provoca un annullamento (Null) dei corrispondenti valori presenti nelle chiavi secondarie delle tabelle collegate da chiave esterna.

Benché il modello relazionale tramite normalizzazione crea sia l'integrità di entità sia l'ntegrità di riferimento spetterà al RDBMS (Relational DBMS) mantenere l'integrità durante il procedimento di inserimento dei dati.

In questo i diversi Rdbms differiscono molto, applicando le tecniche più disparate.

L'SQL (Structured Query Language) e Transact

1. Linguaggi di interrogazione Dati

Il linguaggio SQL dialetto SQLServer (Transact SQL)

Il linguaggio SQL è un linguaggio di programmazione di database la cui origine è strettamente correlata all'invenzione del database relazionale, creato da E. F. Codd all'inizio degli anni '70. Il linguaggio SQL deriva dal linguaggio *Sequel*, nome con cui viene talvolta definito anche oggi.

Il moderno linguaggio SQL si è evoluto in uno standard largamente utilizzato per i database relazionali e definito dalle norme ANSI. Le diverse implementazioni del linguaggio SQL, compresa la versione supportata dal modulo di gestione di SQLServer, sono leggermente diverse dallo standard. Più avanti in questo capitolo verranno descritte le diverse versioni del linguaggio SQL standard, che comunque hanno in comune la stessa struttura di base e la stessa funzionalità. Chi abbia precedentemente utilizzato qualsiasi versione del linguaggio SQL non incontrerà pertanto alcuna difficoltà nel passaggio alla versione di Transact-SQL.

I Componenti del linguaggio SQL

Il linguaggio SQL è costituito da comandi, proposizioni, operatori e funzioni di aggregazione. Tali elementi vengono combinati in istruzioni che consentono di creare, aggiornare e gestire i database. Nelle sezioni che seguono verranno descritti brevemente tutti gli elementi costitutivi del linguaggio SQL, mentre nelle rimanenti parti del capitolo verranno presentati esempi specifici del loro utilizzo.

I comandi SQL

Il linguaggio SQL comprende i comandi del linguaggio di definizione dei dati (DDL) ,i comandi del linguaggio di gestione dei dati (DML) e del controllo delle transazioni e accesso (DCL). Benché in alcuni casi i due linguaggi siano sovrapponibili, i comandi DDL consentono di creare e definire nuovi database, campi e indici, mentre utilizzando i comandi DML è possibile creare query che consentono di ordinare, filtrare ed estrarre dati dal database.

DDL (Data Description Language)

Le istruzioni DDL nel linguaggio SQL sono espressioni costruite in base ai seguenti comandi.

CREATE Consente di creare nuove tabelle, campi e indici.

DROP Consente di eliminare tabelle e indici dal database.

ALTER Consente di modificare tabelle aggiungendovi campi o modificandone la definizione.

DML (Data Manipulation Language)

Le istruzioni DML sono espressioni costruite in base ai seguenti comandi.

SELECT Consente di richiedere i record del database corrispondenti a criteri specifici.

INSERT Consente di caricare gruppi di dati nel database con una sola operazione.

UPDATE Consente di modificare i valori di particolari record e campi.

DELETE Consente di rimuovere record da una tabella del database.

DCL (Data Control Language)

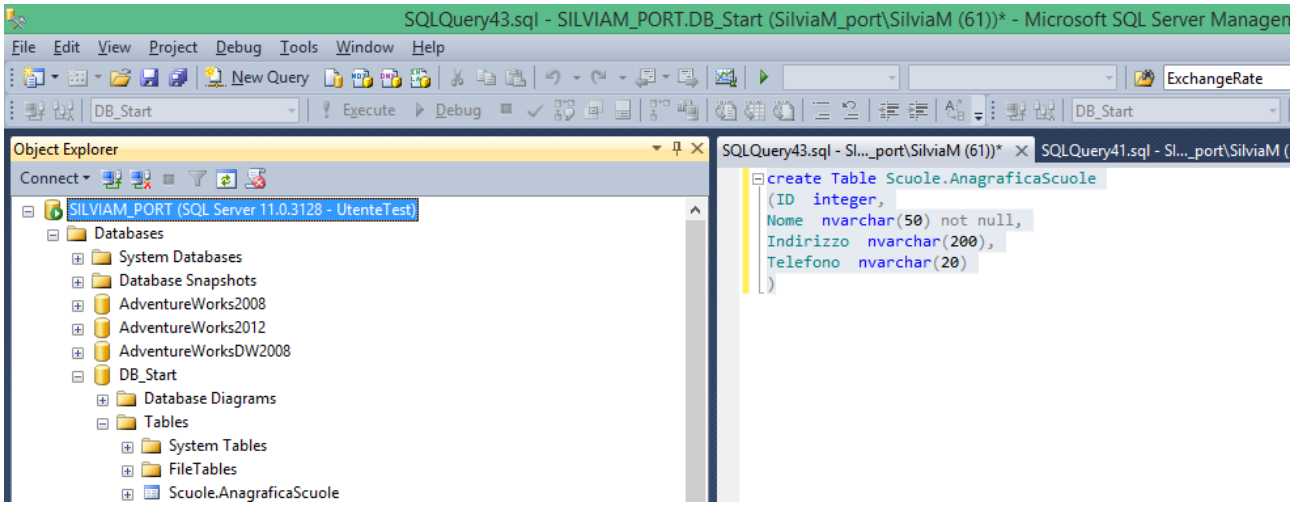
Istruzioni DCL che permettono di controllare l'esatta esecuzione delle operazioni o gruppi di operazioni.

• COMMIT • ROLLBACK

Istruzioni DCL che permettono di implementare l'accesso e la sicurezza.

• GRANT • REVOKE

Accedere a management studio



DML

Elements of a SELECT Statement

Element	Expression	Role
SELECT	<select list>	Defines which columns to return
FROM	<table source>	Defines table(s) to query
WHERE	<search condition>	Filters rows using a predicate
GROUP BY	<group by list>	Arranges rows by groups
HAVING	<search condition>	Filters groups using a predicate
ORDER BY	<order by list>	Sorts the output

Logical Query Processing

```
5: SELECT     <select list>
```

```
1: FROM     <table source>
```

```
2: WHERE    <search condition>
```

```
3: GROUP BY <group by list>
```

```
4: HAVING <search condition>
```

```
6: ORDER BY <order by list>
```

Retrieving Columns from a Table or View

- Use SELECT with column list to display columns
- Use FROM to specify a source table or view
  - Specify both schema and table names
- Delimit names if necessary
- End all statements with a semicolon

Keyword	Expression
SELECT	**<select list>**
FROM	**<table source>**

```
SELECT companyname, country
FROM Sales.Customers;
```

Displaying Columns

- Displaying all columns
  - This is not a best practice in production code!

```
SELECT *
FROM Sales.Customers;
```

- Displaying only specified columns

```
SELECT companyname, country
FROM Sales.Customers;
```

Using Calculations in the SELECT Clause

- Calculations are scalar, returning one value per row

Operator	Description
+	Add or concatenate
-	Subtract
*	Multiply
/	Divide
%	Modulo

- Using scalar expressions in the SELECT clause

```
SELECT unitprice, qty, (unitprice * qty)
 FROM sales.orderdetails;
```

Querying Multiple Tables

- Querying with Inner Joins
- Querying with Outer Joins
- Querying with Cross Joins and Self-Joins

The FROM Clause and Virtual Tables

- FROM clause determines source tables to be used in SELECT statement
- FROM clause can contain tables and operators
- Result set of FROM clause is virtual table
  - Subsequent logical operations in SELECT statement consume this virtual table
- FROM clause can establish table aliases for use by subsequent phases of query

Join Terminology: Cartesian Product

- Characteristics of a Cartesian product
  - Output or intermediate result of FROM clause
  - Combine all possible combinations of two sets
  - In T-SQL queries, usually undesired
    - Special case: table of numbers

Name
Davis
Funk
King

×

Product
Alice Mutton
Crab Meat
Ipoh Coffee

=

Name	Product
Davis	Alice Mutton
Davis	Crab Meat
Davis	Ipoh Coffee
Funk	Alice Mutton
Funk	Crab Meat
Funk	Ipoh Coffee
King	Alice Mutton
King	Crab Meat
King	Ipoh Coffee

Overview of Join Types

- Join types in FROM clause specify the operations performed on the virtual table:

Join Type	Description
Cross	Combines all rows in both tables (creates Cartesian product).
Inner	Starts with Cartesian product; applies filter to match rows between tables based on predicate.
Outer	Starts with Cartesian product; all rows from designated table preserved, matching rows from other table retrieved. Additional NULLs inserted as placeholders.

Inner Join Syntax

- List tables in FROM Clause separated by JOIN operator
- Table aliases preferred
- Table order does not matter

```
FROM t1 JOIN t2
    ON t1.column = t2.column
```

```
SELECT o.orderid,
       o.orderdate,
       od.productid,
       od.unitprice,
       od.qty
FROM Sales.Orders AS o
     JOIN Sales.OrderDetails AS od
     ON o.orderid = od.orderid;
```

Outer Join Examples

- All customers with order details if present:

```
SELECT c.custid, c.contactname, o.orderid, o.orderdate
FROM Sales.Customers AS C
LEFT OUTER JOIN Sales.Orders AS O
     ON c.custid = o.custid;
```

- Customers that did not place orders:

```
SELECT c.custid, c.contactname, o.orderid, o.orderdate
FROM Sales.Customers AS C LEFT OUTER JOIN Sales.Orders AS O
     ON c.custid = o.custid
WHERE o.orderid IS NULL;
```

Filtering Data in the WHERE Clause With Predicates

- WHERE clauses use predicates
  - Must be expressed as logical conditions
  - Only rows for which predicate evaluates to TRUE are accepted
  - Values of FALSE or UNKNOWN filtered out
- WHERE clause follows FROM, precedes other clauses
  - Can't see aliases declared in SELECT clause
- Can be optimized by SQL Server to use indexes
- Data filtered server-side
  - Can reduce network traffic and client memory usage

WHERE Clause Syntax

- Filter rows for customers from Spain

```
SELECT contactname, country
FROM Sales.Customers
WHERE country = N'Spain';
```

- Filter rows for orders before July 1, 2007

```
SELECT orderid, orderdate
FROM Sales.Orders
WHERE orderdate > '20070101';
```

- Filter orders within a range of dates

```
SELECT orderid, custid, orderdate
FROM Sales.Orders
WHERE orderdate >= '20070101' AND orderdate < '20080101';
```