**SUBQUERIES**

**La parola chiave IN**

SELECT \*

FROM Clienti

WHERE id **IN** (SELECT idCliente

FROM Incassi

WHERE importo > 1000)

**La parola chiave EXIST**

SELECT DISTINCT idCitta

FROM Citta

WHERE **EXISTS** (SELECT \*

FROM Citta\_SquadreCalcio

WHERE Citta.idCitta = Citta\_SquadreCalcio.idCitta)

*ALTERNATIVA MIGLIORE*

*SELECT DISTINCT idCitta*

*FROM Citta, Citta\_SquadreCalcio*

*WHERE Citta.idCitta = Citta\_SquadreCalcio.idCitta*

**Subqueries**

SELECT \*

FROM Studenti

WHERE voto > (SELECT AVG(voto)

FROM Studenti)

**UNION**

**Stored Procedure**

Una "Stored Procedure" è una sequenza di istruzioni SQL salvata all'interno del catalogo del DB e che può essere richiamata da:

- un programma

- un'altra Stored Procedure.

Una Stored Procedure che richiama se stessa si dice ricorsiva.

I vantaggi delle Stored Procedure sono:

- Le SP aumentano le performance delle applicazioni. Una volta creata la SP viene compilata e salvata nel catalogo del DB. E' quindi più veloce di un comando SQL equivalente non compilato.

- Le SP riducono il traffico tra Server DB ed applicazione client poiché invece di inviare una sequenza (più o meno lunga) di comandi SQL si limita a mandare solo il nome della procedura ed eventuali parametri.

- Le SP sono riusabili e trasparenti a qualsiasi applicazione. Con le SP è possibile creare un'interfaccia comune che consenta l'uso corretto del DB a tutte le applicazioni, senza che i programmatori delle applicazioni client debbano riscrivere le funzionalità fornite dalle SP disponibili.

- Le SP sono sicure grazie all'uso dei diritti di accesso forniti dal RDBMS.

Gli svantaggi delle SP sono:

- Le SP generano un carico ulteriore, sia per quanto riguarda la RAM che la CPU, del Server DB. Le SP non si limitano a richiedere il recupero dei dati ma solitamente svolgono una serie di operazioni logiche che non sempre si adattano alle funzionalità per le quali il server DB è stato progettato.

- Le SP contengono un set di istruzioni SQL non adatto a procedure caratterizzate da una logica complessa. In altri termini non hanno la stessa potenza computazionale di linguaggi come il C++, C# etc.

- Generalmente gli RDBMS forniscono funzionalità di debugging limitate o addirittura nessuna. Per chi sviluppa questo è un problema da non sottovalutare.

- La scrittura e la manutenzione di una SP richiede una competenza che non tutti gli sviluppatori hanno.

CREATE PROCEDURE nome\_funzione ([IN|OUT|INOUT parametro[,...]])

BEGIN

corpo;

END;

**Stored FUNCTION**

Le Stored Functions sono simili alle SP, ma hanno uno scopo più semplice, cioè quello di definire nuove funzioni, in aggiunta a quelle già fornite da MySQL. Esse restituiscono un valore, e non possono quindi restituire un resultset (insieme di record) come invece avviene per le SP.

CREATE FUNCTION nome\_funzione ([parametro[,...]])  
RETURNS tipo  
BEGIN  
   corpo;  
   RETURN Valore;  
END;

CREATE PROCEDURE SelectAllCustomers @City nvarchar(30), @PostalCode nvarchar(10)  
AS  
SELECT \* FROM Customers WHERE City = @City AND PostalCode = @PostalCode  
GO;

EXEC SelectAllCustomers @City = 'London', @PostalCode = 'WA1 1DP';

**BACKUP**

BACKUP DATABASE databasename  
TO DISK = 'filepath'  
WITH DIFFERENTIAL;

BACKUP DATABASE testDB  
TO DISK = 'D:\backups\testDB.bak'  
WITH DIFFERENTIAL;

**SQL INJECTION**

**Sql function**

**Funzioni STRINGHE**

|  |  |
| --- | --- |
| **Function** | **Description** |
| ASCII | Returns the ASCII value for the specific character |
| CHAR\_LENGTH | Returns the length of a string (in characters) |
| CONCAT | Adds two or more expressions together |
| FORMAT | Formats a number to a format like "#,###,###.##", rounded to a specified number of decimal places |
| INSTR | Returns the position of the first occurrence of a string in another string |
| LCASE | Converts a string to lower-case |
| LEFT | Extracts a number of characters from a string (starting from left) |
| LENGTH | Returns the length of a string (in bytes) |
| LOWER | Converts a string to lower-case |
| LTRIM | Removes leading spaces from a string |
| MID | Extracts a substring from a string (starting at any position) |
| POSITION | Returns the position of the first occurrence of a substring in a string |
| REPLACE | Replaces all occurrences of a substring within a string, with a new substring |
| REVERSE | Reverses a string and returns the result |
| RIGHT | Extracts a number of characters from a string (starting from right) |
| RTRIM | Removes trailing spaces from a string |
| STRCMP | Compares two strings |
| SUBSTR | Extracts a substring from a string (starting at any position) |
| TRIM | Removes leading and trailing spaces from a string |
| UCASE | Converts a string to upper-case |
| UPPER | Converts a string to upper-case |

**Funzioni NUMERICHE**

|  |  |
| --- | --- |
| [ABS](https://www.w3schools.com/sql/func_mysql_abs.asp) | Returns the absolute value of a number |
| [AVG](https://www.w3schools.com/sql/func_mysql_avg.asp) | Returns the average value of an expression |
| [COUNT](https://www.w3schools.com/sql/func_mysql_count.asp) | Returns the number of records returned by a select query |
| [EXP](https://www.w3schools.com/sql/func_mysql_exp.asp) | Returns e raised to the power of a specified number |
| [LOG](https://www.w3schools.com/sql/func_mysql_log.asp) | Returns the natural logarithm of a number, or the logarithm of a number to a specified base |
| [MAX](https://www.w3schools.com/sql/func_mysql_max.asp) | Returns the maximum value in a set of values |
| [MIN](https://www.w3schools.com/sql/func_mysql_min.asp) | Returns the minimum value in a set of values |
| [MOD](https://www.w3schools.com/sql/func_mysql_mod.asp) | Returns the remainder of a number divided by another number |
| [POW](https://www.w3schools.com/sql/func_mysql_pow.asp) | Returns the value of a number raised to the power of another number |
| [RAND](https://www.w3schools.com/sql/func_mysql_rand.asp) | Returns a random number |
| [ROUND](https://www.w3schools.com/sql/func_mysql_round.asp) | Rounds a number to a specified number of decimal places |
| [SIGN](https://www.w3schools.com/sql/func_mysql_sign.asp) | Returns the sign of a number |
| [SQRT](https://www.w3schools.com/sql/func_mysql_sqrt.asp) | Returns the square root of a number |
| [SUM](https://www.w3schools.com/sql/func_mysql_sum.asp) | Calculates the sum of a set of values |
| [TRUNCATE](https://www.w3schools.com/sql/func_mysql_truncate.asp) | Truncates a number to the specified number of decimal places |

**Funzioni DATA**

|  |  |
| --- | --- |
| [DATEADD](https://www.w3schools.com/sql/func_sqlserver_dateadd.asp) | Adds a time/date interval to a date and then returns the date |
| [DATEDIFF](https://www.w3schools.com/sql/func_sqlserver_datediff.asp) | Returns the difference between two dates |
| [DATEFROMPARTS](https://www.w3schools.com/sql/func_sqlserver_datefromparts.asp) | Returns a date from the specified parts (year, month, and day values) |
| [DATEPART](https://www.w3schools.com/sql/func_sqlserver_datepart.asp) | Returns a specified part of a date (as integer) |
| [DAY](https://www.w3schools.com/sql/func_sqlserver_day.asp) | Returns the day of the month for a specified date |
| [GETDATE](https://www.w3schools.com/sql/func_sqlserver_getdate.asp) | Returns the current database system date and time |
| [ISDATE](https://www.w3schools.com/sql/func_sqlserver_isdate.asp) | Checks an expression and returns 1 if it is a valid date, otherwise 0 |
| [MONTH](https://www.w3schools.com/sql/func_sqlserver_month.asp) | Returns the month part for a specified date (a number from 1 to 12) |
| [SYSDATETIME](https://www.w3schools.com/sql/func_sqlserver_sysdatetime.asp) | Returns the date and time of the SQL Server |
| [YEAR](https://www.w3schools.com/sql/func_sqlserver_year.asp) | Returns the year part for a specified date |

**Migliora le performance scrivendo condizioni Sargable**

CREATE INDEX IX\_DataFattura

ON dbo.Fatture (DataFattura)

SELECT COUNT(\*)

FROM dbo.Fatture

WHERE DataFattura >=

'2020-01-01'

AND DataFattura <

'2021-01-01';

SELECT COUNT(\*)

FROM dbo.Fatture

WHERE YEAR(DataFattura) =

2020;

SELECT COUNT(\*)

FROM dbo.Clienti

WHERE Nome LIKE 'N%'

SELECT COUNT(\*)

FROM dbo.Clienti

WHERE LEFT(Nome,1)='N'

**Sostituire dove possibile Union con Union All**

Link: [Come migliorare le performance di una query SQL (yimp.it)](https://www.yimp.it/performance-database-tecniche-per-query-piu-veloci/)

**Monitoraggio e ottimizzazione delle prestazioni**

L'obiettivo del monitoraggio dei database consiste nella valutazione delle prestazioni di un server. Un monitoraggio efficace implica l'esecuzione di snapshot periodici delle prestazioni correnti al fine di isolare i processi che causano problemi, nonché la raccolta continua di dati nel tempo per tenere traccia delle tendenze delle prestazioni.

La valutazione continuativa delle prestazioni del database consente di ridurre al minimo i tempi di risposta e di aumentare al massimo la velocità effettiva, ottimizzando pertanto le prestazioni. Traffico di rete, operazioni di I/O su disco e utilizzo della CPU efficienti sono fattori fondamentali per ottenere prestazioni ottimali. È necessario analizzare accuratamente i requisiti delle applicazioni, comprendere la struttura logica e fisica dei dati, valutare l'utilizzo del database e raggiungere compromessi adeguati tra tipi di utilizzo in conflitto, ad esempio elaborazione delle transazioni online (OLTP) e supporto decisionale.

**Monitoraggio e ottimizzazione del database per le prestazioni**

In Microsoft SQL Server e nel sistema operativo Microsoft Windows sono disponibili utilità che consentono di visualizzare la condizione corrente del database e di tenere traccia delle prestazioni in caso di variazioni. Per il monitoraggio di Microsoft SQL Server è disponibile una vasta gamma di strumenti e tecniche. Il monitoraggio di SQL Server consente di eseguire queste operazioni:

Determinare se è possibile migliorare le prestazioni. Il monitoraggio dei tempi di risposta delle query più frequenti consente, ad esempio, di determinare se sono necessarie modifiche alle query o agli indici nelle tabelle.

Valutare le attività degli utenti. Il monitoraggio dei tentativi di connessione a un'istanza di SQL Server consente ad esempio di determinare se il sistema di sicurezza è adeguatamente impostato e di testare applicazioni e sistemi di sviluppo. Il monitoraggio dell'esecuzione di query SQL consente ad esempio di determinare se le query sono formulate in modo corretto e se producono i risultati previsti.

Risolvere problemi o eseguire il debug dei componenti di applicazione, ad esempio di stored procedure.