SQL

Estrae

```
SELECT column_name1, column_name2
 FROM table_name
```

```
SELECT * FROM table_name
```

```
SELECT column_name(s)
 FROM table_name
 WHERE column_name operator value
```

possibile **Join**:

```
SELECT column_name(s)
 FROM table_name1, table_name2
 WHERE column_name operator value
```

equivale a **INNER JOIN** (equijoin)

```
SELECT column_name(s)
 FROM table_name1
 INNER JOIN table_name2
 ON table_name1.column_name=table_name2.column_name
```

```
SELECT column_name(s)
 FROM table_name
 ORDER BY column_name(s) ASC|DESC
```

```
SELECT column_name, aggregate_function(column_name)
 FROM table_name
 WHERE column_name operator value
 GROUP BY column_name
```

```
SELECT column_name, aggregate_function(column_name)
 FROM table_name
 WHERE column_name operator value
 GROUP BY column_name
 HAVING aggregate_function(column_name) operator value
```

```
SELECT column_name(s)
 FROM table_name
 WHERE column_name
 BETWEEN value1 AND value2
```

```
SELECT DISTINCT column_name(s)
 FROM table_name
```

```
SELECT column_name(s)
 FROM table_name
 WHERE column_name LIKE pattern
```

```
SELECT column_name(s)
 FROM table_name AS alias_name;
```

oppure

```
SELECT column_name AS alias_name
 FROM table_name;
```

SQL Server / MS Access per selezionare i primi N record della query SELECT **TOP** number|percent column_name(s) FROM table_name;	***MySQL*** (diversa sintassi per paginazione) SELECT column_name(s) FROM table_name **LIMIT** number;

Inserisce

INSERT INTO table_name
VALUES (value1, value2, value3,...)

INSERT INTO table_name (column1, column2, column3,...)
VALUES (value1, value2, value3,...)

Aggiorna

UPDATE table_name
SET column1=value, column2=value2,...

UPDATE table_name
SET column1=value, column2=value2,...
WHERE some_column=some_value

Cancella

DELETE FROM table_name
WHERE some_column=some_value

DDL: Crea ed elimina

CREATE DATABASE database_name

CREATE TABLE table_name
(
column_name1 data_type,
column_name2 data_type,
column_name3 data_type,
....
) tipo di dati *diversi a seconda del DBMS*

CREATE **VIEW** view_name AS
SELECT column_name(s)
FROM table_name
WHERE condition

CREATE **INDEX** index_name
ON table_name (column_name)

CREATE ***UNIQUE*** INDEX index_name
ON table_name (column_name) *per duplicati non permessi*

MS Access	***MYSQL***
CREATE [UNIQUE]INDEX index_name ON table_name ({nome campo [ASC | DESC]} [,..]) [WITH [PRIMARY | DISALLOW NULL | IGNORE NULL]]	CREATE [UNIQUE | FULLTEXT] INDEX index_name ON table_name ((nome campo [(**lunghezza**)] [ASC | DESC]) [,..]) NB: lunghezza anche per interi/float etc ..

TRUNCATE TABLE table_name *per cancellare dati in tabella, non la tabella*

DROP DATABASE database_name *per eliminare lo schema*

DROP TABLE[1] table_name *per eliminare la struttura*

DROP VIEW view_name

[1] *Attenzione:* non è possibile eliminare una tabella a cui fa riferimento un vincolo FOREIGN KEY. È prima necessario eliminare il vincolo FOREIGN KEY o la tabella di riferimento.

DROP INDEX

MS Access	*MS SQL Server*	*MySQL*
DROP INDEX index_name ON table_name	DROP INDEX table_name.index_name	ALTER TABLE table_name DROP INDEX index_name

ALTER TABLE table_name
ADD column_name datatype
ALTER TABLE table_name
DROP COLUMN column_name

MySQL	*MS Access /MS SQL Server*
ALTER TABLE table_name **MODIFY** COLUMN column_name datatype	ALTER TABLE table_name **ALTER** COLUMN column_name datatype

Aggiornare VIEW

MySQL	*MS Access /MS SQL Server [Transact-SQL]*

CREATE OR **REPLACE** view_name AS
SELECT column_name(s)
FROM table_name
WHERE condition
	`ALTER VIEW view_name AS`
	`SELECT column1, column2, ...`
	`FROM table_name`
	`WHERE condition`

Esempi a confronto nell'uso di diversi RDBMS:

Definizione di chiave primaria e vincoli di unicità

MySQL:	**SQL Server / Oracle / MS Access:**
CREATE TABLE Persons (P_Id int NOT NULL, LastName varchar(255) NOT NULL, FirstName varchar(255), Address varchar(255), City varchar(255), **PRIMARY KEY** (P_Id))	CREATE TABLE Persons (P_Id int NOT NULL **PRIMARY KEY**, LastName nvarchar(255) NOT NULL, FirstName nvarchar(255), Address nvarchar(255), City nvarchar(255)) nb: da SQL Server 2012 (11.x) → nvarchar UNICODE

MySQL:	**SQL Server / Oracle / MS Access:**
CREATE TABLE Persons (P_Id int NOT NULL, LastName varchar(255) NOT NULL, FirstName varchar(255), Address varchar(255), City varchar(255), **UNIQUE (P_Id)**)	CREATE TABLE Persons (P_Id int NOT NULL **UNIQUE**, LastName varchar(255) NOT NULL, FirstName varchar(255), Address varchar(255), City varchar(255))

MySQL: visualizzare tipi numerici es: **int [(M)]**	**SQL Server / Oracle / MS Access: int** o **integer**
CREATE TABLE NomeTabella (…… NomeCampo **int (11),** ⇦ visualizza tutte le cifre se valore maggiore ……)	CREATE TABLE NomeTabella (…… NomeCampo **int** ⇦ senza indicare eventuale diversa visualizzazione ……)

Definizione di chiave esterna

MySQL:

```
CREATE TABLE Orders
(
O_Id int NOT NULL,
OrderNo int NOT NULL
P_Id int,
PRIMARY KEY (O_Id),
FOREIGN KEY (P_Id) REFERENCES Persons(P_Id)
ON DELETE CASCADE
ON UPDATE CASCADE
)
```

Possibile engine InnoDB *che permette di applicare* ***vincoli di integrità referenziale*** *con* modalità*:*

- CASCADE: propaga la modifica anche alla tabella figlia;
- SET NULL: annulla l'attributo portandolo al valore Null;
- SET DEFAULT: assegna all'attributo il valore di default;
- RESTRICT: impedisce che la modifica possa avvenire.

SQL Server / Oracle / MS Access:

```
CREATE TABLE Orders
(
O_Id int NOT NULL PRIMARY KEY,
OrderNo int NOT NULL,
P_Id int FOREIGN KEY REFERENCES Persons(P_Id)
)
CREATE TABLE Orders
   (O_Id integer PRIMARY KEY,
    ….,
    P_Id integer FOREIGN KEY REFERENCES Persons
    ON DELETE CASCADE
    ON UPDATE CASCADE )
```

nb: **SQL Server da SQL Server 2012 (11.x)**

```
CREATE TABLE Orders
( O_Id int NOT NULL PRIMARY KEY,
OrderNo int NOT NULL,
P_Id int,
FOREIGN KEY (P_Id) REFERENCES Persons(P_Id)
)
```

Esempio MySQL

```
CREATE TABLE Ordini
( ID_ordine INT(5) NOT NULL AUTO_INCREMENT,
id_cliente INT(5) NOT NULL,
data DATE,
id_pagamento INT(5) NOT NULL DEFAULT '1',
importo FLOAT (5,2) DEFAULT '0',
conferma ENUM('S','N'),
PRIMARY KEY(ID_ordine),
INDEX(id_cliente),
FOREIGN KEY(id_cliente) REFERENCES Clienti(ID_cliente) ON DELETE
RESTRICT,
INDEX(id_pagamento),
FOREIGN KEY(id_pagamento) REFERENCES Pagamenti(ID_pagamento) ON
DELETE RESTRICT)
TYPE=INNODB;
```

lunghezza int, float
AUTO_INCREMENT
uso enum

impedisce modifica

specifica il tipo di DB

USE *nome database* per aprire un DB tra molti ed evitare di specificarne il nome nell'uso di tabelle

MySQL:

SHOW
(TABLES | DATABASE | COLUMNS)
[IN | FROM]
[nome tabella| noma database]

ad esempio SHOW character set;

ad esempio USE rubrica;
SHOW columuns FROM elenco
USE scuola;
SHOW columuns FROM studenti

Vincolo di CHECK

MySQL:	**SQL Server / Oracle / MS Access:**
non accetta* Vincolo di CHECK** ***ma è possibile emularlo: ***con View*** ***oppure*** ***con Stored Procedure e Trigger***	CREATE TABLE Persons (P_Id int NOT NULL **CHECK (P_Id>0)**, LastName varchar(255) NOT NULL, FirstName varchar(255), Address varchar(255), City varchar(255))

AUTO INCREMENTO di un campo

MySQL:	**SQL Server:**
CREATE TABLE Persons (ID int NOT NULL **AUTO_INCREMENT**, LastName varchar(255) NOT NULL, FirstName varchar(255), Address varchar(255), City varchar(255), PRIMARY KEY (ID)) *default: valore iniziale 1 ed incremento di 1*	CREATE TABLE Persons (ID int **IDENTITY(1,1)** PRIMARY KEY, LastName varchar(255) NOT NULL, FirstName varchar(255), Address varchar(255), City varchar(255)) ad es: *con valore iniziale 1 ed incremento di 1*

MS Access:	CREATE TABLE Persons (ID Integer PRIMARY KEY **AUTOINCREMENT**, LastName varchar(255) NOT NULL, FirstName varchar(255), Address varchar(255), City varchar(255))

SQL DEFAULT Constraint (vincolo) **on ALTER TABLE**

MySQL:	**SQL Server / MS Access:**
ALTER TABLE Persons ALTER City SET DEFAULT 'SANDNES'	ALTER TABLE Persons ALTER **COLUMN** City SET DEFAULT 'SANDNES'

DROP a DEFAULT Constraint

MySQL:	**SQL Server / Oracle / MS Access:**
ALTER TABLE Persons ALTER City DROP DEFAULT	ALTER TABLE Persons ALTER **COLUMN** City DROP DEFAULT

MySQL ***controllo di esistenza / non esistenza***	**SQL Server** ***controllo di esistenza / non esistenza***
DROP TABLE *IF EXISTS* table_name CREATE TABLE *IF NOT EXISTS* table_name *(per qualsiasi oggetto: database, view etc..)*	IF OBJECT_ID ('table_name[2]') *IS NOT NULL* DROP TABLE table_name IF OBJECT_ID ('table_name') *IS NULL* CREATE TABLE table_name

Test su valore nullo: funzioni diverse a seconda del DBMS

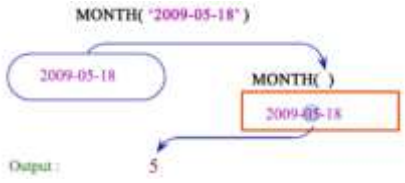
>> SQL SERVER: built-in function

Date: formato e funzioni diverse a seconda del DBMS

[2] Opzionale specificare il tipo di oggetto IF OBJECT_ID ('view_name', 'V') … per view etc…

Esempio nell'uso di Month

Sintassi Month(*date*)

MySQL	**SQL Server**	**MS ACCESS Data Functions**
SELECT MONTH(CURDATE());	SELECT **Month** (GetDate());	SELECT **Month** (Date());
altro es: 		

Il formato data

Access è in grado, almeno teoricamente, di gestire correttamente almeno 3 formati di data: lo standard yyyy/mm/dd, l'inglese mm/dd/yyyy e l'italiano dd/mm/yyyy. Di fatto, l'ultimo formato, viene gestito correttamente solo fino a quando il valore del mese e della data non risultano una combinazione tale da poter essere confusi col valore inglese e quindi scambiati con conseguenti problemi gestionali.

Per quanto riguarda ***MySQL*** la soluzione è tanto più drastica quanto efficiente. L'unico formato supportato è infatti quello **ISO**, ovvero la struttura **yyyy/mm/dd** dove lo / può essere comunemente rimpiazzato con un -.

Vediamo un semplice esempio legato ad un aggiornamento di un record:

Access strSQL = "UPDATE nometabella SET campodata = #09/24/2005# "

MySQL strSQL = "UPDATE nometabella SET campodata = '2005-09-24' "

Per poter gestire correttamente la migrazione da Access a MySQL è quindi necessario rivedere tutti gli inserimenti di date e prevedere una funzione che automaticamente splitti la data e la ricomponga come necessario. Ciò è facilmente ottenibile semplicemente usando le funzioni Year(), Month() e Day() messe a disposizione da vbscript oppure la più generica DatePart().

Ancor meglio, siccome Access è in grado di gestire correttamente anche questo formato, peraltro standard anche per ***SQL Server*** sarebbe opportuno abituarsi già dal principio all'uso di questa struttura.

A questo è poi possibile accodare l'ora nel formato consueto hh:mm:ss per un risultato finale del tipo
yyyy-mm-dd hh:mm:ss

MySQL o SQL Server? 6 differenze su cui riflettere

Esempi con uso MySQL

```
CREATE TABLE tabella
 (stringa CHAR(20),              stringa di lunghezza massima 20 caratteri
numero_int INT,                  numero intero
numero_dec DECIMAL(5,2),         numero con 3 cifre intere (+ segno e punto che funge da virgola) e 2 decimali
data DATE,                       data in formato AAAA-MM-GG
testo TEXT                       testo lungo e di lunghezza variabile (fino a 65.000 caratteri)
);
```