Il linguaggio SQL
Introduzione e tipi di dato

foglia@iet.unipi.it

Sommario

- Introduzione al linguaggio SQL
- I tipi di dato SQL e gli operatori
- I costrutti di interrogazione del linguaggio SQL

Il Linguaggio SQL

- Il linguaggio SQL (Structured Query Language) è di fatto lo standard tra i linguaggi per la gestione di basi di dati relazionali
- Standardizzato attraverso le attività di comitati di standardizzazione internazionali quali l'ANSI e l'ISO, SQL è oggi adottato dalla maggior parte dei sistemi DBMS commerciali e open-source
  - Standard
    - garantire la portabilità del codice
    - Diminuire il tempo di apprendimento nel migrare da un DBMS ad un altro
- Versioni principali: SQL-2, SQL-3 (comp. Con SQL-2)
  - Vedremo essenzialmente SQL-2 e pochi aspetti di SQL-3

Il Linguaggio SQL

- Nonostante l'SQL sia uno standard le varie implementazioni DBMS scelgono quali funzioni implementare. Un sistema DBMS quindi può non implementare tutte le funzioni definite nell'SQL

- Ciascuna implementazione (in un DBMS) può includere delle funzionalità aggiuntive del linguaggio che rendono l'implementazione non completamente aderente allo standard (non compatibili quindi con altri prodotti)
  - Implementazione non standard di funzionalità previste dallo standard
  - Implementazione di funzionalità non previste dallo standard

- Generalmente, manuali del DBMS hanno una sezione dedicata alla conformità allo standard ad alle funzionalità aggiuntive

- Molti sistemi mettono a disposizione interfacce amichevoli per l'interrogazione, la definizione e l'aggiornamento dei dati (ad esempio, interfacce di natura visuale)
- Tutti consentono, comunque, di utilizzare in modo esplicito il linguaggio SQL

Il Linguaggio SQL

- SQL è un linguaggio **dichiarativo** (non-procedurale), ovvero non specifica la sequenza di operazioni da compiere per ottenere il risultato
- SQL si dice essere “relazionalmente completo”, perché ogni espressione dell’algebra relazionale può essere tradotta in una corrispondente espressione in SQL
- Il linguaggio SQL contiene funzionalità di :
  - **Data Manipulation Language DML** - Linguaggio di manipolazione dei dati, è l'insieme di istruzioni dedicate alla manipolazione di dati, include istruzioni per l'inserimento, la cancellazione e la modifica di dati
  - **Data Definition Language DDL** - Linguaggio di definizione dei dati, è l'insieme di istruzioni dedicate alla definizione di dati e tabelle
  - **Data Control Language DCL** – Linguaggio di controllo degli accessi, è l’insieme di istruzioni che permette di definire I permessi necessari per accedere alle tabelle della base di dati

SQL: un po’ di storia

- L’attuale SQL è lo sviluppo di un linguaggio di interrogazione per il DBMS Relazionale **System R,** proposto originariamente presso i laboratori di ricerca dell’IBM nella seconda metà degli anni ‘70.
- il suo nome era **SEQUEL** (Structured English Query Language) - 1974
- nelle evoluzioni successive divenne **SQL**
- prime implementazioni in SQL/DS e Oracle (1981)
- dal 1983 ca. "standard di fatto"
- Oggetto di un’ intensa attività di standardizzazione ad opera sia dell’ANSI che dell’ISO (International Standard Organization).
- standard (1986, poi 1989, 1992, 1999, 2003, 2006, 2008, 2011 …)
  - recepito solo in parte (Vedi http://troels.arvin.dk/db/rdbms/ per un confronto)

Il linguaggio SQL
Principali Tipi di dato

foglia@iet.unipi.it

Principali Tipi di dato SQL

- Alla base del modello relazionale vi sono le tabelle con i relativi attributi. Ogni attributo appartiene ad un dominio che deve essere ben definito
- Al fine di definire i domini degli attributi per la creazione di tabelle (gli elementi che vengono manipolati dall' SQL appartengono tutti ad un tipo di dato e questo deve essere dichiarato) e per la loro interrogazione, SQL definisce vari tipi di dato
- Un **tipo di dato** è costituito:
  - da un insieme di valori
  - da un insieme di operazioni definite sul tipo

Tipi di dato

- I Tipi di Dato o (Domini) possono essere classificati in:
  - Tipi di dato predefiniti
    - **Numerici**
      - **Esatti**: rappresentano valori interi e valori decimali in virgola fissa (es. 75,4.5,-4.6...)
      - **Approssimati**: rappresentano valori reali in virgola mobile
    - **Carattere** (singoli caratteri o stringhe, anche di lunghezza variabile)
    - **Temporali**
    - **Bit** (singoli booleani o stringhe)
    - **Booleani** (introdotti in SQL3)
  - Tipi di dato definiti dall'utente a partire dai domini elementari

Tipi numerici esatti

- Questa famiglia contiene i domini numerici che permettono di rappresentare valori numerici esatti, es. numeri interi o numeri con parte decimale di lunghezza prefissata (e.g. valori monetari)
  - **NUMERIC(precisione, scala)** : rappresenta i valori decimali (interi o decimali, positivi e negativi):
    - **Precisione** è il numero totale di cifre (>=0)
    - **Scala** (>0 e <precisione) è il numero di cifre alla destra del separatore decimale
    - Sia dimensione che precisione sono parametri facoltativi: se non vengono esplicitate, sono impostati automaticamente (scala=0 di default)
  - **DECIMAL(precisione, scala) abbr DEC**: identico a NUMERIC, con l'unica differenza che la precisione rappresenta il valore minimo con cui poi I numeri verranno memorizzati
    - La precisione massima per entrambi dipende dall'implementazione. Quella dei decimal >= numeric (possono essere sinonimi)
  - **BIGINT** : Rappresenta i valori interi (senza cifre decimali).
  - **INTEGER abbr INT**: Rappresenta i valori interi (senza cifre decimali). Non è possibile dimensionarlo (cioè specificarne la lunghezza in byte).
  - **SMALLINT**: Rappresenta i valori interi. I valori di questo tipo sono usati per eventuali ottimizzazioni poichè richiedono minore spazio di memorizzazione.
    - (possono essere sinonimi)

esempi

- In mysql il separatore è il .
- **In mysql decimal e numeric sono sinomini**
  - **Nel senso di Numeric (la precisione non può essere aumentata)**

- considerato il tipo decimal(4,2)
  - 100 non appartiene al tipo
  - -2.2 appartiene al tipo
  - 22 appartiene al tipo
  - 22.122 viene approssimato a 22.12 (approssimazione all'intero più vicino)

- Considerato il tipo decimal(3)
  - Equivale a decimal(3,0)
  - 1000 non appartiene al tipo
  - -23 appartiene al tipo
  - -998.8 viene approssimato a -999

esempi

- In MYSQL
  - INT sono i numeri rappresentati in C2 su 32 bit
  - SMALLINT sono i numeri rappresentati in C2 su 16 bit
  - BIGINT sono i numeri rappresentati in C2 su 64 bit

  - Quali sono i valori ammessi?

Tipi numerici approssimati

- Per le grandezze reali approssimate invece (es. grandezze fisiche) SQL definisce:
  - **REAL**: rappresenta valori a singola precisione in virgola mobile. La precisione dipende dalla specifica implementazione di SQL.
  - **DOUBLE PRECISION o DOUBLE**: rappresenta valori a doppia precisione in virgola mobile. La precisione dipende dalla specifica implementazione di SQL.
  - **FLOAT**: permette di richiedere la precisione che si desidera (il parametro p indica il numero di cifre da rappresentare per la mantissa). Il formato è FLOAT(p).

Tipi numerici approssimati

- In mysql REAL e DOUBLE sono sinonimi
  - Implementazione non standard

- FLOAT sono rappresentati su 32 bit
- DOUBLE sono rappresentati su 64 bit
- FLOAT(P) ha una implementazione non standard
  - Sceglie fra 32 e 64 bit e non la dimensione

- Numeri validi:
  - -0.1E-1 (-0.01)
  - 9
  - 9.322

Tipi caratteri

- SQL definisce diversi tipi carattere utilizzati per rappresentare stringhe (sequenze) di caratteri (inclusa la rappresentazione a carattere di un numero)
- Assegnare un valore più lungo della lunghezza specificata nella definizione del tipo di dato costituisce un errore a tempo di esecuzione (run time error)
  - La sintassi per la definizione è:
    - tipo_di_dato(dimensione)
    - esempio: CHAR, CHAR(100),
    - in alcuni casi è obbligatorio indicare la dimensione
    - se non si indica, viene assunto un valore di default

Tipi carattere

- **CHARACTER** (abbreviato **CHAR**): permette di definire stringhe di caratteri di lunghezza fissata n. La specifica è della forma CHAR(n).
  - la dimensione massima è 2000 (255 in MYSQL)
  - la dimensione minima e di default è1
  - Essendo una lunghezza fissa, nel caso che venga inserita una stringa di m caratteri (dove m è minore di n), il DBMS aggiunge in coda alla stessa n-m caratteri ‘ ‘ (spazio) in modo da raggiungere la lunghezza massima dichiarata.
- **CHARACTER VARYING** (abbreviato **VARCHAR**): permette di definire stringhe di caratteri a lunghezza variabile con lunghezza massima predefinita n. La specifica è della forma VARCHAR(n).
  - la dimensione è obbligatoria e va da 1 a 4000 (65536 in mysql)
  - se si immette un dato di lunghezza inferiore, non si effettua il blank padding

esempi

- CHAR o CHAR(1)
  - ‘a’ // valore valido
  - ‘Aa’ //valore non valido genera errore
- CHAR(4)
  - ‘a’ // valore valido
  - ‘a a’ // valore valido (3 caratteri)
  - ‘aaaaa’ // valore non valido
- Lo stesso vale per i VARCHAR
- I letterali caratteri sono racchiusi fra ‘’ (apici) o doppi apici
  - Ed esistono le sequenze di escape
  - Esempio ‘1’ o ‘a’ sono i caratteri 1 e a

Tipi temporali

- SQL definisce dei tipi temporali per permettere la gestione di date e ore
  - Il formato usato è quello anglosassone:
  - Data ‘anno-mese-giorno’
  - Ora ‘ore:minuti:secondi’
  - Nella maggior parte dei DBMS date e ore sono inserite sotto forma di stringhe di caratteri, per cui vanno racchiuse tra apici ‘anno-mese-giorno’.
- **DATE**: rappresenta le date espresse come anno (4 cifre), mese (2 cifre comprese tra 1 e 12), giorno (2 cifre comprese tra 1 e 31 con ulteriori restrizioni a seconda del mese)
  - 'YYYY-MM-DD'
- **TIME**: rappresenta I tempi espressi come ora (2 cifre), minuti (2 cifre) e secondi (2 cifre) - sono utilizzati sia per rappresentare tempi che intervalli intervalli temporali
  - 'HH:MM:SS'
- **DATETIME**: rappresenta una concatenazione dei due tipi di dato precedenti con una precisione al microsecondo, sono specificati nella forma ‘anno-mese-giorno ore:minuti:secondi.microsecondi’
  - 'YYYY-MM-DD HH:MM:SS.UUUUUU‘
  - in mysql 'YYYY-MM-DD HH:MM:SS’

Tipi temporali - DATE

- Letterali DATE
  - ‘1999-11-01’ // valido
  - ‘-1999-11-01’ // non valido: la data non può essere negativa
  - ‘11-11-2011’ // non valido formato sbagliato
  - ‘2011-13-11’ // non valido mese sbagliato
  - ‘2011-12-41’ // non valido giorno sbagliato
  - ‘2011-04-31’ // non valido – perché?

Tipi temporali - TIME

- Usati per rappresentare sia l'ora del giorno, ma anche intervalli temporali
  - Formato: 'HH:MM:SS' o 'HHH:MM:SS'
  - Valori ammissibili: da '-838:59:59' a '838:59:59'

- Esempi di letterali
  - '54:45:00' // corretto
  - '154:45:01' //corretto
  - '-154:45:01' // negativo ma corretto
  - '01:70:00' // scorretto: i minuti sono > 59
  - '01:59:61' // scorretto: i secondi sono > 59

Tipi temporali - DATETIME

- Formato:'YYYY-MM-DD HH:MM:SS.UUUUUU’
  - In mysql i letterali hanno il formato di sopra, ma sono memorizzati nel formato
    - 'YYYY-MM-DD HH:MM:SS’
    - Con arrotondamento dei microsecondi
  - HH:MM:SS devono essere un tempo e non un intervallo temporale
- Letterali
  - ‘2018-02-09 16:00:00’ // valido
  - ‘2018-01-10’ // valido
    - ‘2018-01-10 00:00:00’ così viene memorizzato
  - ‘2018-01-10 10:20’ // valido
    - ‘2018-01-10 10:20:00’
  - ‘2010-04-31’ // non valido
  - ‘2010-04-30 18:75:00’ // non valido

Tipi booleani

- SQL definisce un tipo BOOLEAN per rappresentare appositamente i valori booleani
  - Generalmente un dato booleano occupa soltanto un bit.
  - Esempi di valori booleani (letterali):
  - TRUE, FALSE, '1', ‘0‘
  - UNKNOWN è un valore che può essere utilizzato solo per verificare i valori (tramite is e is not) – come null
  - Presente in mysql come sinonimo di bit(1)
    - letterali
      - 0 vale FALSE
      - 1 vale TRUE
    - in realtà TRUE e FALSE sono alias per 1 e 0
      - Qui, non in Java

Valore NULL

- Il valore NULL è una speciale costante presente in tutti i DBMS
  - Rappresenta l' assenza di informazione
  - è compatibile con ogni tipo di dato
  - Non va confuso con lo 0, non sono la stessa cosa, anche se internamente possono avere la stessa rappresentazione
  - Utilizzato nelle espressioni aritmetiche, restituisce sempre il valore NULL
- Confronti con NULL:
  - **Per verificare se un valore è NULL si deve usare l'operatore IS NULL o IS NOT NULL e non l'operatore di uguaglianza**
    - se A vale NULL
      - A = 0 è equivalente a falsa (è null, si comporta da false nella where)
      - A > 0 è equivalente a falsa (è null, si comporta da false nella where)
      - A < 0 è equivalente a falsa (è null, si comporta da false nella where)
      - A <> 0 è equivalente a falsa (è null, si comporta da false nella where)
      - **A = NULL è equivalente a falsa** (è null, si comporta da false nella where)
      - **A IS NULL è vera**
  - **operatori o funzioni su valori NULL restituiscono NULL**
    - **1+NULL restituisce null**

Il linguaggio SQL
Operatori e Funzioni

foglia@iet.unipi.it

Operatori e funzioni

- Sui vari tipi sono definiti un certo numero di operatori
  - In generale, gli operandi devono essere dello stesso tipo
  - In caso opposto, a meno delle conversioni implicite di tipo alla Java, non viene segnalato errore ma il risultato ha poco/nullo significato
  - Gli operatori hanno precedenza ed associatività
- Inoltre sono definiti un numero elevato di funzioni (funzioni scalari) che consentono la manipolazione dei tipi
  - Ad esempio la somma di due date o la concatenazione di due stringhe
  - In generale hanno più argomenti, e restituiscono un valore di un dominio elementare
- **Vedremo di seguito i principali. Altri verranno introdotti nella presentazione degli statement SQL**

Operatori aritmetici

- Si usano con i tipi numerici
  - + - * / %
  - DIV divisione intera
    - A div b => int(a/b)
  - Esempi
    - 2+3
      - 5
    - 3/2
      - 1.5
    - 5 DIV 2
      - 2
    - 5 DIV 2.9
      - 1 -> int (5/2.9)= int(1.7241) = 1

Operatori di confronto

- = (verifica l'uguaglianza)
- != oppure <>
- >, >=, <, <=
- IS boolean – test per boolean
- IS NOT boolean – test per boolean
- …

- Il confronto si applica:
  - Ai tipi numerici
  - Alle stringhe – ma è case insensitive in mysql (di default)
  - Alle date

esempi

- 1=2 -> 0 (ossia FALSE)
- ‘casa’ > ‘a’ -> 1 (TRUE)
- ‘casa’ > ‘A’ -> 1 (TRUE)
- ‘casa’ > ‘CASA’ -> 0 (FALSE in mysql) – case insensitive
- ‘casa’ > ‘casal’ -> 0 (FALSE) – ordinamento alfabetico
- ‘1’ > ‘a’ -> 0 (FALSE) – i numeri sono più piccoli delle lettere
- ‘1aa’> ‘9b’ -> 0 – ordinamento alfabetico

- '1991:11:10' > '1991:10:10‘ (TRUE)
- '1991:11:10 00:00:00' = '1991:11:10 00:00:00‘ (TRUE)

- '1991:11:10' > '1991:11:10 01:00:00‘ (FALSE OK)
- Ma attenzione:
  - '1991:11:10' = '1991:11:10 00:00:00' (FALSE)
  - Non «mischiare i tipi»

- '80:00:00' >= '80:00:00.01‘ (FALSE)

Operatori logici

- AND &&
- OR ||
- NOT !

Funzioni scalari per le stringhe

- CONCAT(*str1*,*str2*,...)
  - Restituisce una stringa che è la concatenazione degli argomenti
  - Es:
    - **CONCAT('My', 'S', 'QL')**
      - **Restituisce MySQL**
- LOWER(*str*) and UPPER(str)
  - Restituiscono una stringa con i caratteri tutti minuscoli o maiuscoli
- CHAR_LENGTH(*str*)
  - Restituisce il numero di caratteri della stringa
  - CHAR_LENGTH(*'pippo'*) -> 5
  - CHAR_LENGTH(*'pippo '*) ->7
- E molte altre

Funzioni scalari per le date – data ed ora corrente

- NOW()
  - Restituisce la data e l'ora corrente, nel formato 'YYYY-MM-DD HH:MM:SS' - datetime
  - Esempio: '2018-02-10 09:10:26'

- CURTIME()
  - Restituisce l'ora corrente, nella forma 'HH:MM:SS' - time
  - Esempio: '09:10:26'

- CURDATE()
  - Restituisce la data corrente, nella forma 'YYYY-MM-DD' - date
  - Esempio: 2018-02-10

Funzioni scalari per le date – differenza fra 2 date in giorni

- DATEDIFF(*expr1*,*expr2*)
  - Restituisce la differenza in giorni fra due date (expr1-expr2)
  - Esempi:
    - datediff('2018:03:10', '2018:02:01') -> 37
    - datediff('2018:02:01', '2018:03:10') -> -37
    - datediff('2019:02:01', '2018:02:01') -> 365
    - Vale anche per valori negativi

- TIMEDIFF(*expr1*,*expr2*) (expr1-expr2)
  - Restituisce la differenza fra due intervalli temporali
  - Esempio:
    - timediff('23:28:10','11:27:01') ->12:01:09

Funzioni scalari per le date – differenza espressa in unit

- TIMESTAMPDIFF(*unit*,*datetime_expr1*,*datetime_expr2*)
  - *datetime_expr1*,*datetime_expr2 sono di tipo datetime*
  - *Il risultato è espresso in unit, dove unit può valere:*
    - MICROSECOND , SECOND, MINUTE, HOUR, DAY, WEEK, MONTH, or YEAR
  - *Calcola datetime_expr2-datetime_expr2 in unit*
  - *Esempi:*
    - timestampdiff(day, '2018:02:01', '2018:03:10') -> 37 come sopra
    - timestampdiff(month, '2018:02:01', '2018:03:10')->1
    - timestampdiff(hour, '2018:02:01', '2018:03:10') -> 888

Funzioni scalari per le date – aggiungere/sottrarre intervalli ad una data

- ADDTIME(*expr1*,*expr2*)
- SUBTIME(*expr1*,*expr2*)
  - expr1 è di tipo time o datetime
  - expr2 è di tipo time
    - È ammesso il formato ‘D HH:MM:SS’
  - Restituisce expre2+/-expr1 come datetime o time
  - Esempi:
    - addtime('01:01:59.01', '00:0:01.99') -> 01:02:01
    - addtime('01:01:59', '-00:00:01'); -> 01:01:58 // come sub
    - subtime('01:01:59', '00:00:01'); -> 01:01:58
    - addtime('2018:10:02 11:18:00', '14:00:00');
      - .> 2018-10-03 01:18:00
    - addtime('2018:10:02 11:18:00', '1 14:00:00')
      - -> 2018-10-04 01:18:00

Funzioni scalari per le date – aggiungere/sottrarre intervalli

- date + INTERVAL *expr unit*
- date - INTERVAL *expr unit*
  - *Date è di tipo date o datetime*
  - *unit definisce come expr (che è una stringa in formato assegnato) deve essere interpretata*
  - *Alcuni valori per unit (e formato di expr)*
  - *DAY ‘day’*
  - YEAR_MONTH 'YEARS-MONTHS‘
  - DAY_SECOND 'DAYS HOURS:MINUTES:SECONDS‘
  - *Sono disponibili anche YEAR, MONTH, WEEK, MICROSECOND (singoli valori)*

esempi

- '2018-02-10' + interval '365' day
  - -> ‘2019-02-10’
- '2018-02-10' + interval '1-2' year_month
  - ‘2019-04-10’
- '2018-02-10' + interval '1 01:00:00' day_second
  - 2018-02-11 01:00:00
- '2018-02-10 23:00:00' + interval '1 01:00:00' day_second
  - 18-02-12 00:00:00

Estrarre i campi da una data

- Esistono varie funzioni
- DATE_FORMAT(*date*,*format*)
  - Format è una stringa che specifica il formato

Specificatore	Descrizione
%d	giorno del mese numerico 00…31
%M	nome del mese January…December
%m	mese numerico 00…12
%H	ora 00…23
%i	minuti 00…59
%s	secondi 00…59
%Y	anno di quattro cifre
%y	anno di due cifre

esempi

- date_format('2018-02-10 12:03:00', '%d')
  - -> 10
- date_format('2018-02-10 12:03:00', '%d %M')
  - -> 10 February
- date_format('2018-02-10 12:03:00', '%d %m %Y')
  - -> 10 02 2018 // il ns formato
- date_format('2018-02-10 12:03:00', '%H %i %s')
  - 12 03 00

Funzioni «veloci»

- Accettano il tipo date e datetime
  - year(date) estrae l'anno
  - month(date) estrae il mese
  - day(date) estrae il giorno

- Esempi:
  - year('1999-12-28')
    - ->1999
  - month('1999-02-01 13:01:01')
    - ->02
  - day('2018-02-10 12:03:00')
    - ->10

Per TIME

- TIME_FORMAT(*date*,*format*)
  - Funziona come DATE_FORMAT
  - Si applicano solo gli specificatori validi per il formato
  - If the *time* value contains an hour part that is greater than 23, the %H format specifiers produce a value larger than the usual range of 0..23

- Esempio:
  - time_format('12:00:10','%H %i %s')
    - -> 12 00 10
  - time_format('12:00:10','%s')
    - -> 12 00 00

Trasformare stringhe in date

- STR_TO_DATE(*str*,*format*)
  - Converte la stringa str in una date, datetime o time a seconda del formato specificato.
  - Gli specificatori di formato sono analoghi ai precedenti
  - Il formato deve consentire il «matching»

- Esempi
  - str_to_date('10-11-2012','%d-%m-%Y')
    - ->2012-11-10
  - str_to_date('10/11/2012','%d-%m -%Y');
    - -> NULL // formato sbagliato
  - str_to_date('10/11/2012','%d/%m /%Y');
    - 2012-11-20
  - str_to_date('10-11-2012','%d-%m %Y')
    - -> NULL // formato sbagliato
  - str_to_date('10-11-2012','%d-%m');
    - -> 0000-11-10
  - str_to_date('00-11-11','%H-%i-%s')
    - -> 00:11:11 // è di tipo time
  - str_to_date('10-11-2012 12:03:00','%d-%m-%Y %H:%i:%s')
    - -> 2012-11-10 12:03:00 // è di tipo datetime

Commenti in SQL

- -- commento
- /* */

Tipi di dato definiti dall'utente

- SQL permette di definire domini e utilizzarli nella definizione delle tabelle.
- Un Dominio viene creato tramite istruzione CREATE DOMAIN

- Sintassi:
- **CREATE DOMAIN NomeDominio AS Tipo [DEFAULT value] [CHECK(Vincolo)]**

- Esempi:
- **CREATE DOMAIN TipoEta AS INTEGER CHECK (VALUE BETWEEN 1 AND 120);**
- **CREATE DOMAIN Occupazione AS VARCHAR(30) DEFAULT 'Disoccupato' ;**
- **CREATE DOMAIN Voto AS SMALLINT DEFAULT NULL CHECK (VALUE >=18 AND VALUE <= 30);**