**CHAPTER 2.A – DML QUERIES AND DATA MANIPULATION**

Simple queries with single table / multiple tables

**SQL** è un linguaggio per la definizione e la manipolazione di database relazionali. Comprende 3 sottolinguaggi:

* **DDL:** creazione e modifica di schema relazionali dei DB
* **DML:** aggiornamento e queries sui database
  + - SELECT: permette la dichiarazione di queries per ricavare le informazioni dal database.
    - INSERT compie l’inserzione di una o più righe in una tabella.
    - DELETE permette all’utente di eliminare una o più righe di una tabella.
    - UPDATE modifica i valori di una o più colonne (o righe) di una tabella.
* **DCL:** Cambi dinamici alle proprietà dei database

Chart, diagram

Description automatically generated

Esempio:

Elenca tutti i nomi e l’età di tutti i ciclisti

*SELECT name, age*

*FROM Cyclist*;

Elenca tutti i nomi, l’altezza e la categoria delle arrampicate (order by height and category)

*SELECT climbname, height, category*

*FROM Climb*

*ORDERED BY height, category;*

*SELECT* indica quali informazioni devono essere stampate

*FROM* indica da quale tabella devo prendere le informazioni

*ORDERED BY* indica in quale ordine devono essere mostrate le informazioni

*WHERE* indica quali condizioni devono essere applicate per le tuple

Il *SELECT* DISTINCT permette di stampare solo valori diversi nel caso ci fossero più valori uguali.

Il *SELECT* \* permette di stampare tutti i valori da una tabella.

Esempio:

Elenca tutte le informazioni dalla tabella Team

*SELECT* \*

FROM Team

L’istruzione LIKE è seguita da un pattern che viene usato per le strings wildcards in SQL:

‘%stringa%’ se ci sono altri caratteri prima o dopo la stringa

\_ rappresenta un singolo carattere

Esempio

Ottieni il nome e l’età dei ciclisti che appartengono ai team con nomi che contengono la parola ‘ciclisti’

*SELECT* name, age

FROM Cyclist

*WHERE* teamname LIKE ‘%ciclisti%’ ;

Elenca il numero degli stage dove il nome della città d’arrivo ha come prima lettera un ‘A’ oppure il nome della città di partenza ha due o più ‘s’

*SELECT* stagenum

FROM Stage

*WHERE* arrival LIKE ‘A%’ **OR** departure LIKE ‘%s%s%’;

Per selezionare elementi NULLI bisogna utilizzare il comando IS NULL (nell’istruzione WHERE).

*SELECT* **COUNT** (\*) serve per contare il numero di righe in una colonna

*SELECT* **AVG, MAX, MIN,** per stampare la media, il valore massimo o minimo di una tupla.

Se l’informazione è contenuta su più tabelle allora bisogna utilizzare il formato **FROM.** Una query su più tabelle corrisponde ad un prodotto cartesiano: bisogna porre molte condizioni altrimenti il numero di elementi selezionato sarebbe molto alto. Se ci sono delle chiavi aliene definite, spesso alcune condizioni sono formate come un’uguaglianza tra la chiave aliena e l’attributo corrispondente nella tabella alla quale si riferisce.

Con n tabelle avremo almeno n-1 connessioni.

Esempio

Elenca le paia di stage e arrampicate vinte dallo stesso ciclista

*SELECT* Stage.stagenum, climbname

FROM Stage, Climb

*WHERE* Stage.cnum = Climb.cnum;

Subqueries

Una **subqueries** è una query tra parentesi inclusa in un’altra query.

Esempio

Calcola il numero e la lunghezza degli stages con arrampicate

*SELECT DISTINCT* stagenum, km

FROM Stage

*WHERE* stagenum **IN** (*SELECT DISTINCT* stagenum

FROM Climb)

Esempio

Ottieni il nome delle arrampicate la cui altezza è più grande della media delle altezze delle arrampicate della seconda categoria.

*SELECT* climbname

FROM Climb

*WHERE* height **>** (*SELECT* AVG(height)

FROM Climb

WHERE category = ‘2’);

L’alternativa al predicato **IN** è l’**EXIST** che restituisce true se l’istruzione SELECT da come risultato almeno una riga.

Se sappiamo che esistono valori nulli bisogna usare il NOT EXIST mentre se non ci sono valori nulli (per via dei vincoli) si può usare il NOT IN (che altrimenti darebbe come risultato indefinitio).

Universal quantification

In SQL non esiste il ‘’per ogni con implicazione’’, infatti bisogna rimodellare questi tipi di richieste

ESEMPIO

‘’Tutti gli esseri umani sono essere viventi’’ diventa ‘’Non esistono essere umani che non sono esseri viventi’’

Ottieni il nome ed il codice dei dipartimenti tali che per tutti i professori di questo dipartimento la sua provincia sia ‘’Teruel’’

Ottieni il nome ed il codice dei dipartimenti tali che non esistono professori di questo dipartimento di cui la sua provincia non sia di ‘’Teruel’’

*SELECT* D.cod\_department, D.nombre

FROM Department D

*WHERE* **NOT EXIST** (*SELECT* \*

FROM Profesor P

*WHERE* P.provincia <> ‘Teruel’ AND P.cod\_department = D.cod\_department)

**AND EXIST** (*SELECT* \*

FROM Profesor P1

*WHERE* P1.cod\_department = D.cod\_department) <- importante per far uscire solo dipartimenti che hanno professori

Grouping

Un **gruppo** è un insieme di righe con lo stesso valore per il sott’insieme di colonne usato per raggruppare (**GROUP BY**)

Esempio

Ottieni il nome di tutte le squadre e la media dell’età dei ciclisti di ogni team

*SELECT* teamname, AVG(age)

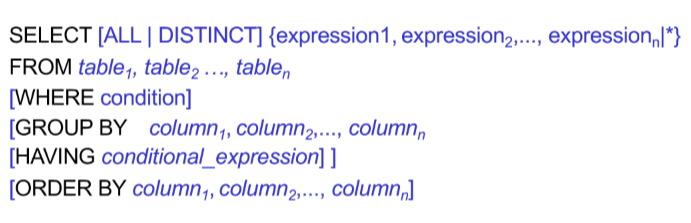
FROM Cyclist

GROUP BY teamname;

Table

Description automatically generated with medium confidence

Nel caso di query con GROUP la sintassi completa è:

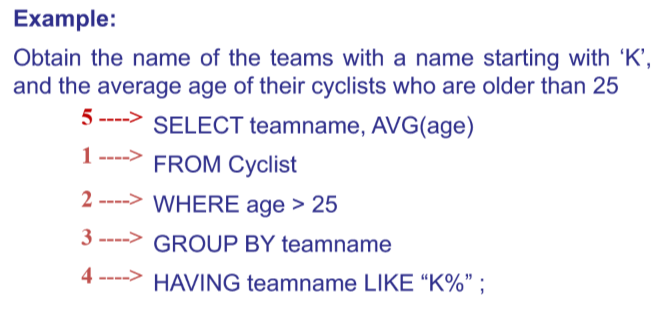




Ordine logico

La funzione HAVING può essere usata solo nelle query con GROUP BY ed ha lo scopo di fornire una condizione extra simile a WHERE ma da applicare solo al gruppo.

Sia in HAVING che in *SELECT* possono apparire solo attributi presenti nei gruppi.

La funzione *WHERE* è viene applicata prima dei gruppi. 

Set operations

Per creare delle queries tra più tabelle oltre ad utilizzare le subqueries o ad includere più tabelle con la funzione FROM è possibile utilizzare le **set tables operations** :

* UNION: corrisponde all’operatore unione dell’insiemistica.

Graphical user interface, text, application

Description automatically generated

* EXCEPT: corrisponde all’operatore differenza dell’insiemistica.

Graphical user interface, text, application

Description automatically generated

* INTERSECT: corrisponde all’operatore intersezione dell’insiemistica

Graphical user interface, text, application

Description automatically generated

Join

L’operatore join corrisponde alla concatenazione nell’algebra relazionale, abbiamo 3 tipi di join:

1. Cross Join

Table\_reference1 cross join Table\_reference2 == SELECT\* FROM table\_reference1, table\_reference2

1. Inner Join, 3 forme differenti:
   * table\_reference1 JOIN table\_reference2 ON conditional expression ==

SELECT \* FROM table1, table2 WHERE conditional expression

Text

Description automatically generated

* + table1 JOIN table2 USING (c1, c2, …, c­\_n) == SELECT \* FROM table1, table2

WHERE table1.c1 = table2.c2 AND table1.c\_n = table2.c\_n

Graphical user interface, text, application

Description automatically generated

* + table1 NATURAL INNER JOIN table2 è uguale al JOIN USING però in questo caso gli attributi in comune tra le tabelle sono più di uno

Graphical user interface, text

Description automatically generated

1. Outer Join combina tutte le righe di una delle tabelle (anche se non ci sono corrispondenze per alcune righe dell’altra tabella)
   * table1 NATURAL LEFT/RIGHT/FULL JOIN table2

Timeline

Description automatically generated

* + table1 LEFT/RIGHT/FULL JOIN table2 ON conditional expression

Graphical user interface, text, application

Description automatically generated

Database updates

Il modulo **DML** (data manipulation languages) si occupa dell’update dei database.

Funzione *INSERT INTO* permette l’inserzione di una o più righe in una tabella, la sintassi è:

*INSERT INTO* table [(column1, column2, …, column n)]

{**DEFAULT VALUES** / OR /

**VALUES (atom1, atom2, atom n)** / OR /

**table expression**}

Se non includiamo la lista di colonne dovremo specificare tutti gli attributi della tabella che inseriamo tramite VALUES, se inseriamo una colonna con tutti gli attributi non occorre specificare le colonne.

Se utilizziamo l’opzione **DEFAULT VALUES**inseriremo una singola riga con tutti i valori default stabiliti nella definizione della tabella.

Se utilizziamo l’opzione **VALUES(atom1,atom2,atom n)** , gli atomi sono dati da espressioni scalari.

Se utilizziamo l’opzione **table expression** inseriamo delle righe che sono il risultato di un’esecuzione di un comando (quindi un *SELECT*).

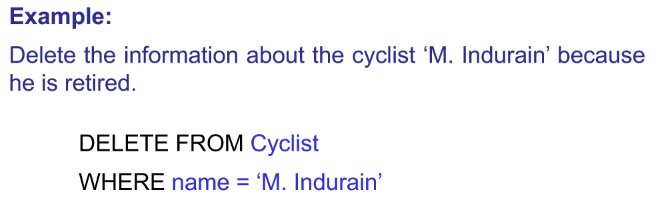
Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

L’istruzione **DELETE** **FROM** table [WHERE conditional expression] serve per eliminare una riga da una tabella, tramite l’istruzione where eliminiamo solo le righe che rispettano quella condizione, altrimenti tutte le righe verranno eliminate.



L’istruzione **UPDATE** table

**SET** assignment1, assignment2, … ,assignment n

[WHERE conditional expression]

Serve per modificare righe di una tabella.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Commands for handling transactions

Una **transaction** è un’unità logica di lavoro composta da uno o più dichiarazioni SQL che è garantita *atomica* rispetto al ripristino.

L’inizializzazione della **transaction** è implicito, inizia con la prima dichiarazione SQL in una sessione, oppure quando la transaction precedente termina.

Il completamento di una transazione può avvenire in due modi

* COMMIT se la transizione termina con successo, in questo caso il cambio del database è permanente.
* ROLLBACK se la transizione abortisce mandando indietro ogni cambiamento fatto con la transizione.

**CHAPTER 2.B – DATA DEFINITION LANGUAGE DDL**

DDL si occupa della creazione, modifica o eliminazione dei componenti dello schema relazionale dei DB, le operazioni possibili sono:

create schema: fornisce il nome ad uno schema relazionale e dichiara l’utente che è il proprietario di quello schema.

create domain: definisce un nuovo dominio di dati.

create table: definisce una tabella, il suo schema e le sue restrizioni.

create view: definisce una view o relazioni derivate nello schema relazionale.

create assertions: definisce restrizioni d’integrità generali.

grant: definisce le autorizzazioni all’utente per le operazioni sugli oggetti DB.

Tutti questi comandi hanno l’operazione inversa (DROP / REVOKE) e la modificazione (ALTER).

La funzione **create scheme** ha la seguente sintassi:

create scheme [scheme name][AUTHORIZATION user][list of scheme elements]

scheme elements sono:

* Domain definition
* Table definition
* View definition
* Constraint definition
* Authorization definition

Per eliminare uno schema bisogna utilizzare:

DROP SCHEMA schema\_name {RESTRICT / CASCADE};

L’opzione CASCADE serve ad eliminare automaticamente gli oggetti contenuti nello schema.

La funzione **create table** segue la seguente sintassi:

**create table** table\_name

(column definition list

[table constraints definition list]

);

La column definition avviene in questo modo:

column\_name datatype (VARCHAR (n), CHAR(n), NUMBER (n, n), DATE)

[DEFAULT {literal | system\_function | NULL }]

[column constraints definition list]

I column constraints sono:

NOT NULL

UNIQUE

REFERENCE l’operazione non è permessa se la chiave aliena è violata (operazione default)

ON DELETE ↓

{CASCADE, SET NULL, SET DEFAULT, NO ACTION}

ON UPDATE

{CASCADE, SET NULL, SET DEFAULT, NO ACTION}

CHECK (conditional expression)

Graphical user interface, text, application, email

Description automatically generated

I table constraints sono:

A screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface, text, application, email

Description automatically generated

In Oracle l’unico tipo di *integrità referenziale* supportato è quello **WEAK**

Se in una tupla di R tutti i valori per gli attributi di FK hanno valore non nullo, deve esistere una tupla di S che ha gli stessi valori per gli attributi UK come i valori per gli attributi FK.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, chat or text message

Description automatically generated

Graphical user interface, text

Description automatically generated

Una view è una tabella virtuale che è derivata da altre tabelle (base o virtuali anche esse), può essere consultata come le altre tabelle.

Graphical user interface, diagram, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Un update della view si può fare solo se non c’è AMBIGUITA’.

Una view non è modificabile se:

Contiene operatori set (UNION, INTERSECT, …)

Contiene operatore DISTINCT

Contiene funzioni aggregate (SUM, AVG, …)

Contiene GROUP BY

Se una view contiene più di una tabella le modifiche sono permesse solo per la tabella che contiene la chiave primaria che potrebbe essere la primary key della view.

Graphical user interface, text, application

Description automatically generated

Per ELIMINARE una view bisogna utilizzare il comando

DROP VIEW view\_name {CASCADE CONSTRAINTS}

↘ Per eliminare tutte le chiavi aliene che si riferiscono a questa view.

Access control

Ogni oggetto in SQL ha un **proprietario**, un proprietario di uno schema database è anche il proprietario di tutti i suoi componenti.

Il proprietario è l’unica persona che può compiere qualsiasi operazione sugli oggetti.

Graphical user interface, text

Description automatically generatedPer dare permessi ad altri utenti il **proprietario** deve usare la funzione **GRANT**.

*Operazioni da consentire all’utente.*



Per revocare i permessi ad un utente bisogna utilizzare il comando REVOKE

Text

Description automatically generated

E’ necessario conoscere l’accesso alle autorizzazioni di tutti gli utenti, un’autorizzazione può essere trasferita ad un altro utente in modo trasferibile o non. Quando un’autorizzazione è revocata se l’operazione era transferibile è necessario revocare tutte le autorizzazioni fatte da quest’ultimo utente.

Diagram

Description automatically generated

Se un utente riceve più di un’autorizzazione, ciascuna di esse può essere revocata in maniera indipendente.