

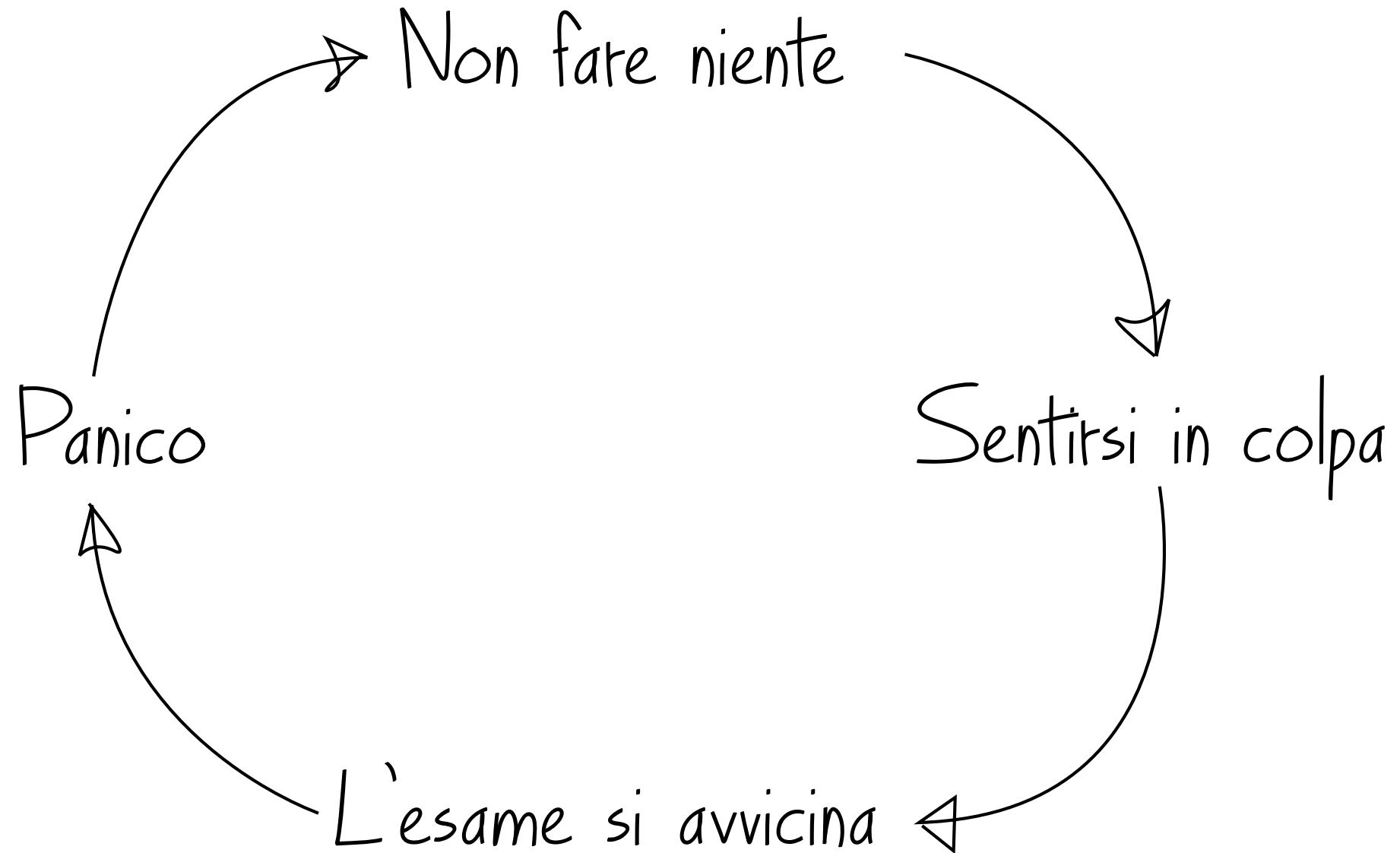
Basi di dati cod. 861II [9 CFU]

Corso di Laurea in Ingegneria Informatica

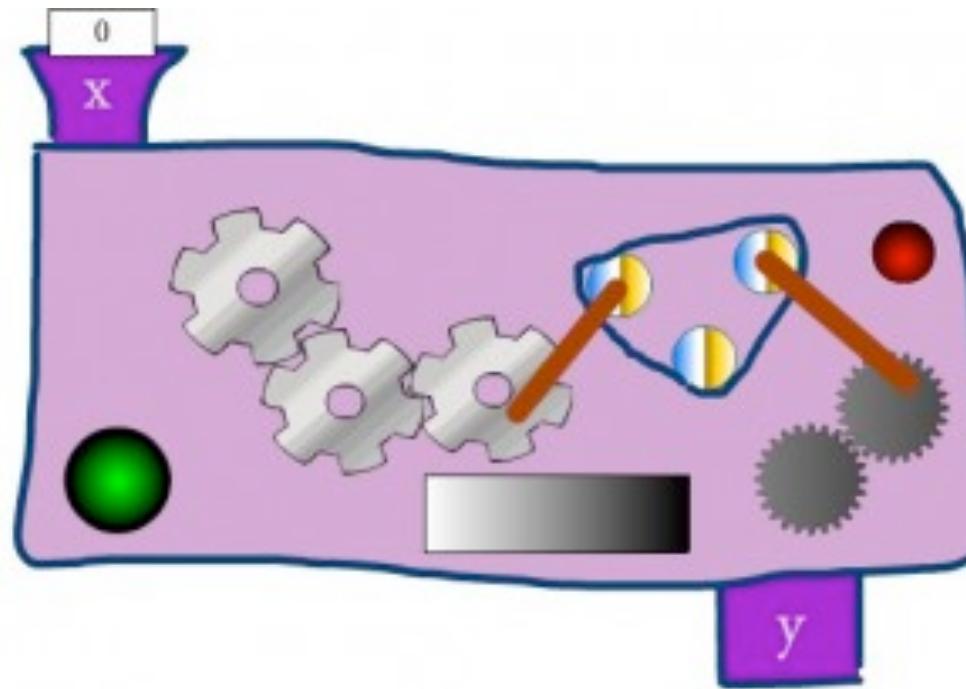
Oracle MySQL
A.A. 2022-2023

Francesco Pistolesi
Dipartimento di Ingegneria dell'Informazione
Università di Pisa
francesco.pistolesi@unipi.it

È Maggio...

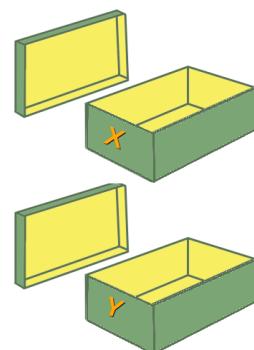


Parametri di una stored procedure



Tipologie di parametro

Una stored procedure MySQL accetta parametri di tipo
ingresso*, *uscita* e *ingresso-uscita



permettono di comunicare col chiamante

Ingresso

IN



equivalente al passaggio per valore



Un parametro in ingresso **può essere letto**, ma non modificato

i parametri sono in ingresso per default
(se non specificato diversamente)

Ingresso: esempio

Scrivere una stored procedure che stampi la parcella media di una specializzazione specificata come parametro

```
DROP PROCEDURE IF EXISTS parcella_media_spec;  
  
DELIMITER $$  
CREATE PROCEDURE parcella_media_spec(IN _specializzazione VARCHAR(100))  
BEGIN  
    SELECT AVG(M.Parcella)  
        FROM Medico M  
        WHERE M.Specializzazione = _specializzazione;  
END $$  
DELIMITER ;  
  
CALL parcella_media_spec('Ortopedia'); ← chiamata
```

Esecuzione

The screenshot shows the MySQL Workbench interface with the following details:

- Title Bar:** Untitled @db_root (localhost)
- Toolbar:** Run, Stop, Export Wizard, New, Load, Save, Save As, Grid View, Form View, Image, Text, Hex.
- Tab Bar:** clinicaNe... (closed), Untitled @ClinicaNe... (closed), Untitled @Clinica (lo... (closed), Untitled @Clinica (lo... (closed), medico @db_root (lo... (closed), Untitled @db_root (l... (closed).
- Query Editor:** The tab "Query Editor" is selected.
- SQL Script:** The script is as follows:

```
1 DROP PROCEDURE IF EXISTS parcella_media_spec;
2
3 DELIMITER $$

4
5 CREATE PROCEDURE parcella_media_spec(IN _specializzazione VARCHAR(100))
6 BEGIN
7     SELECT AVG(M.Parcella)
8     FROM Medico M
9     WHERE M.Specializzazione = _specializzazione;
10 END $$

11
12 DELIMITER ;
13
14 CALL parcella_media_spec('Ortopedia');
```

The output window shows the results of the query:

Message	Result
AVG(M.Parcella)	170.0000

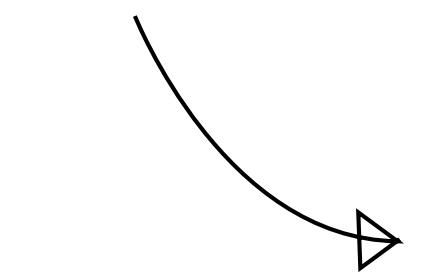
At the bottom, the status bar indicates: CALL parcella_media_spec('Ortopedia') 0.01 sec elapsed. The result pane shows 1 record.

Uscita

OUT



Un parametro di uscita **può essere modificato** per assumere il
valore del risultato della stored procedure



nella chiamata si possono usare
variabili user-defined (quelle che iniziano con '@')

Esempio

Scrivere una stored procedure che restituisca il numero di pazienti visitati da medici di una data specializzazione, ricevuta come parametro

```
DROP PROCEDURE IF EXISTS tot_pazienti_visitati_spec;

DELIMITER $$

CREATE PROCEDURE tot_pazienti_visitati_spec(
    IN _specializzazione VARCHAR(100),
    OUT totale_pazienti_ INT)

BEGIN
    SELECT COUNT(DISTINCT V.Paziente) INTO totale_pazienti_
    FROM Visita V
        INNER JOIN
            Medico M ON V.Medico = M.Matricola
    WHERE M.Specializzazione = _specializzazione;
END $$

DELIMITER ;
```

```
CALL tot_pazienti_visitati_spec('Neurologia', @quantiPazienti);
```

```
SELECT @quantiPazienti;
```

Esecuzione

The screenshot shows the MySQL Workbench interface with the following details:

- Title Bar:** Untitled @db_root (localhost)
- Toolbar:** Run, Stop, Export Wizard, New, Load, Save, Save As, Grid View, Form View, Image, Text, Hex.
- Tab Bar:** Untitled @ClinicaNe... (selected), Untitled @Clinica (lo...), Untitled @Clinica (lo...), medico @db_root (lo...), Untitled @db_root (l...).
- Query Editor:** Contains the following SQL code:

```
1 DROP PROCEDURE IF EXISTS tot_pazienti_visitati_spec;
2
3 DELIMITER $$
4 CREATE PROCEDURE tot_pazienti_visitati_spec(
5     _specializzazione VARCHAR(100),
6     totale_pazienti_ INT)
7 BEGIN
8     SELECT COUNT(DISTINCT V.Paziente) INTO totale_pazienti_
9     FROM Visita V
10    INNER JOIN
11        Medico M ON V.Medico = M.Matricola
12    WHERE M.Specializzazione = _specializzazione;
13 END $$
14 DELIMITER ;
15
16 CALL tot_pazienti_visitati_spec('Neurologia', @quantiPazienti);
17
18 SELECT @quantiPazienti;
```
- Message Panel:** Shows the result of the last query: @quantiPazienti | 18
- Result Panel:** Shows the result of the last query: SELECT @quantiPazienti | 1 record | 0.00 sec elapsed

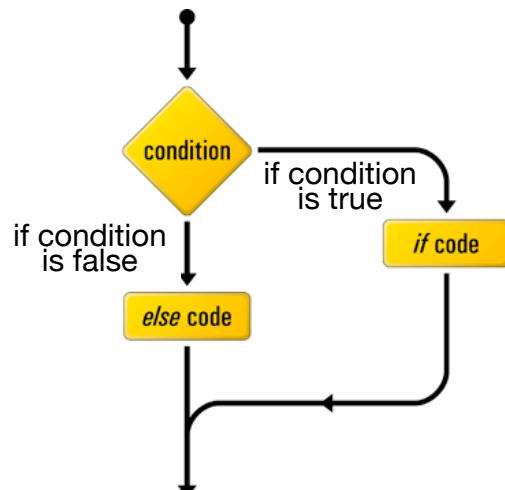
Istruzioni condizionali



Istruzioni condizionali

→ istruzione IF e istruzione CASE

Le istruzioni condizionali permettono di **esprimere condizioni**, modificando il flusso di esecuzione



possono contenere letterali, variabili e funzioni

Istruzione IF

parte facoltativa

IF if_condition **THEN**

-- blocco istruzioni if true

ELSEIF elseif_1_condition **THEN**

-- blocco istruzioni elseif_1

:

:

ELSEIF elseif_N_condition **THEN**

-- blocco istruzioni elseif_N

ELSE

-- blocco istruzioni else

END IF ;

obbligatorio il punto e virgola!

Esempio

Scrivere una stored procedure che riceva come parametro un intero t e una specializzazione s e restituisca in uscita `true` se il numero di visite della specializzazione s nel mese in corso è superiore a t , `false` se è inferiore, e `NULL` se è uguale.

Soluzione

```
1  DROP PROCEDURE IF EXISTS visite_sopra_soglia;
2
3  DELIMITER $$ 
4  CREATE PROCEDURE visite_sopra_soglia(IN _t INT, IN _s VARCHAR(100), OUT passed BOOLEAN)
5  BEGIN
6      DECLARE visite_mese_attuale INT DEFAULT 0;
7      SET visite_mese_attuale = (
8          SELECT COUNT(*)
9              FROM Visita V
10             INNER JOIN
11                 Medico M ON V.Medico = M.Matricola
12            WHERE M.Specializzazione = _s
13            AND MONTH(V.`Data`) = MONTH(CURRENT_DATE)
14            AND YEAR(V.`Data`) = YEAR(CURRENT_DATE)
15      );
16
17      IF visite_mese_attuale > _t THEN
18          SET passed = TRUE;
19      ELSEIF visite_mese_attuale < _t THEN
20          SET passed = FALSE;
21      ELSE
22          SET passed = NULL;
23  END IF;
24  END $$ 
25  DELIMITER ;
26
27  CALL visite_sopra_soglia(10, 'Otorinolaringoiatria', @controllo);
```

Istruzione CASE

CASE

WHEN condition_1 **THEN**

-- blocco istruzioni_1

:
:

WHEN condition_N **THEN**

-- blocco istruzioni_N

END CASE ;

equivale all'istruzione 'switch' del C++

Più variabili di uscita

Scrivere una stored procedure che restituisca la data in cui un paziente, il cui codice fiscale è passato come parametro, è stato visitato per la prima volta, e il nome e cognome del medico che lo ha visitato in tale circostanza. In caso di più medici, per semplicità, selezionarne uno.



Considerando che Data in Visita fa parte della chiave, in uno stesso giorno più medici possono visitare lo stesso paziente

Soluzione

Scrivere una stored procedure che restituisca la data in cui un paziente, il cui codice fiscale è passato come parametro, è stato visitato per la prima volta, e il nome e cognome del medico che lo ha visitato in tale circostanza. In caso di più medici, per semplicità, selezionarne uno.

```
1  DELIMITER $$  
2  ▼CREATE PROCEDURE primaVisitaPaziente(IN paziente VARCHAR(16),  
3                                         OUT dataPrimaVisita DATE,  
4                                         OUT cognomeMedico VARCHAR(100),  
5                                         OUT nomeMedico VARCHAR(100))  
6  ▲  
6  ▼BEGIN  
7      SELECT MIN(V.`Data`) INTO dataPrimaVisita  
8      FROM Visita V  
9      WHERE V.Paziente = paziente;  
10  
11     SELECT M.Cognome, M.Nome INTO cognomeMedico, nomeMedico  
12     FROM Visita V INNER JOIN Medico M ON V.Medico = M.Matricola  
13     WHERE V.Paziente = paziente  
14         AND V.`Data` = dataPrimaVisita  
15     LIMIT 1;  
16  ▲END $$  
17  DELIMITER ;
```

Istruzioni iterative



Istruzioni iterative

→ istruzioni WHILE, REPEAT e LOOP

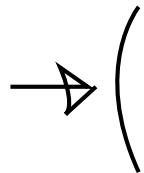
Le istruzioni iterative permettono di **ripetere blocchi di codice**,
dipendentemente dalla veridicità di una condizione



la condizione può valutare anche funzioni

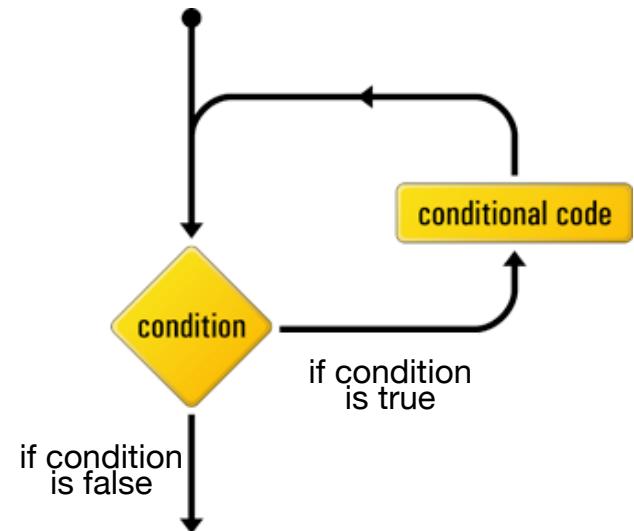
While

eseguito finché
la condizione è
TRUE



WHILE condition **DO**
-- blocco istruzioni
END WHILE;

la condizione è controllata prima di ogni iterazione



Repeat



eseguito fintantoché non →
si verifica la condizione

(UNTIL contiene
una condizione d'uscita)

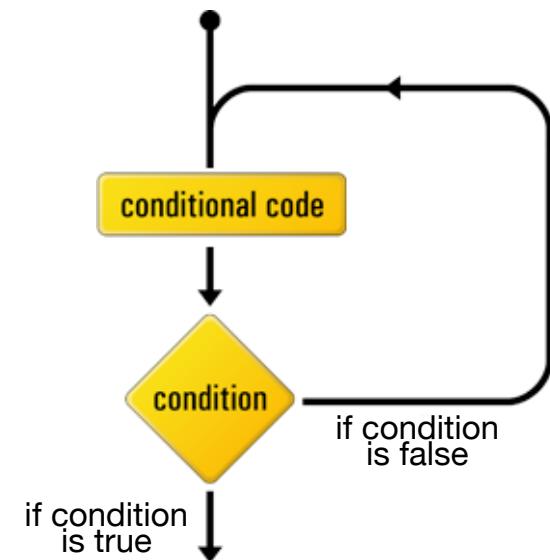
la condizione è controllata dopo ogni iterazione

REPEAT

-- blocco istruzioni

UNTIL condition

END REPEAT;



Esempio...programmazione spinta

Scrivere una stored procedure che riceve in ingresso un intero i e stampa a video i primi i interi separati da virgola, in ordine crescente



Esempio: se $i=3$, l'uscita deve essere 1 , 2 , 3

Soluzione con WHILE

The screenshot shows the MySQL Workbench interface with a query editor window titled "Untitled @db_root (localhost)". The code in the editor is:

```
1 DROP PROCEDURE IF EXISTS stampa_interi;
2
3 DELIMITER $$;
4 CREATE PROCEDURE stampa_interi(IN i INT)
5 BEGIN
6     DECLARE s VARCHAR(255) DEFAULT '1';
7     DECLARE counter INT DEFAULT 2;
8
9     WHILE counter <= i DO
10         SET s = CONCAT(s,',',counter);
11         SET counter = counter + 1;
12     END WHILE;
13
14     SELECT s;
15 END $$;
16 DELIMITER ;
17
18 CALL stampa_interi(3);
```

The result pane below shows the output of the query:

s
1,2,3

At the bottom of the result pane, it says "CALL stampa_interi(3)" and "0.00 sec elapsed".

Soluzione con REPEAT

The screenshot shows the MySQL Workbench interface with a query editor window. The query editor contains the following MySQL code:

```
1 DROP PROCEDURE IF EXISTS stampa_interi2;
2
3 DELIMITER $$;
4 CREATE PROCEDURE stampa_interi2(IN i INT)
5 BEGIN
6     DECLARE s VARCHAR(255) DEFAULT '1';
7     DECLARE counter INT DEFAULT 2;
8
9     REPEAT
10        SET s = CONCAT(s, ',', counter);
11        SET counter = counter + 1;
12    UNTIL counter > i
13    END REPEAT;
14
15    SELECT s;
16 END $$;
17 DELIMITER ;
18
19 CALL stampa_interi2(3);
```

The result pane shows the output of the stored procedure:

s
1,2,3

Below the result pane, the command was `CALL stampa_interi2(3)` and it took `0.01 sec elapsed`.

Loop

loop_label: **LOOP**

-- blocco istruzioni e check di condizioni

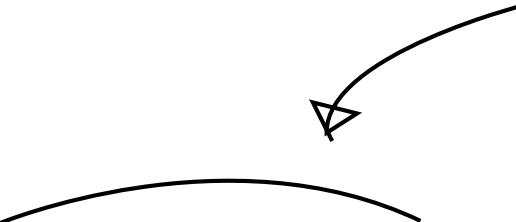
END LOOP;



le condizioni di uscita sono gestite dal programmatore

Istruzioni di salto

sono sempre usate in concomitanza con blocchi if o case



Le istruzioni **LEAVE** e **ITERATE** permettono di **interrompere** un ciclo o **passare all'iterazione successiva**, rispettivamente



leave è l'equivalente di break, mentre iterate di continue...

Esempio

Scrivere una stored procedure che riceve in ingresso un intero i e stampa a video i numeri dispari da 1 a i , separati da virgola

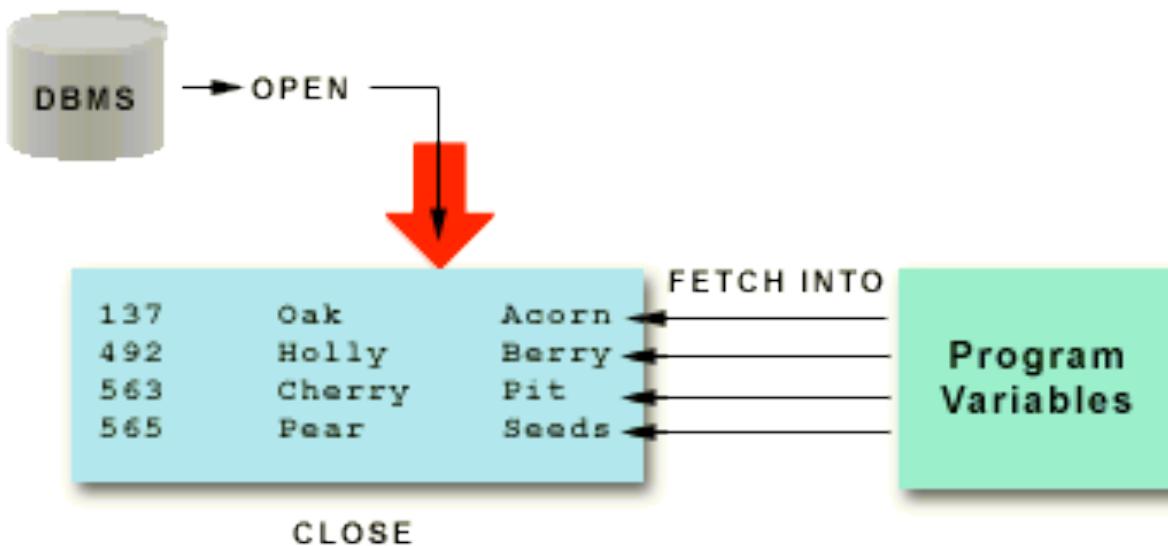
Esempio: se $i=5$, l'uscita deve essere 1, 3, 5

Soluzione

```
1  DROP PROCEDURE IF EXISTS stampa_dispari;
2  DELIMITER $$ 
3  CREATE PROCEDURE stampa_dispari(IN i INT)
4  BEGIN
5      DECLARE s VARCHAR(255) DEFAULT '';
6      DECLARE counter INT DEFAULT 1;
7
8      IF i>0 THEN
9          SET s = '1';
10     END IF;
11
12     scan: LOOP
13         SET counter = counter + 1;
14
15         IF counter = i+1 THEN
16             LEAVE scan;
17         END IF;
18
19         IF (counter % 2) = 0 THEN
20             ITERATE scan;
21         ELSE
22             SET s = CONCAT(s, ', ', counter);
23         END IF;
24     END LOOP;
25
26     SELECT s;
27 END $$ 
28 DELIMITER ;
```

```
graph TD
    L12[12 scan: LOOP] --> L13[13 SET counter = counter + 1]
    L13 --> L15[15 IF counter = i+1 THEN]
    L15 --> L16[16 LEAVE scan]
    L16 --> L24[24 END LOOP]
    L24 --> L20[20 ITERATE scan]
    L20 --> L12
```

Cursori



Cursori

come farebbe un puntatore

Scorrono i record di un result set, solo in avanti,
per effettuare delle azioni all'interno di istruzioni iterative



Cursori: dichiarazione

```
DECLARE NomeCursore CURSOR FOR  
SQL query ;
```

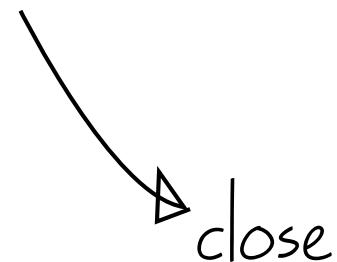


i cursori si possono dichiarare solo immediatamente dopo
la dichiarazione di tutte le variabili

Cursori: apertura, fetch e chiusura



Per usare un cursore lo si deve **aprire**, poi è possibile **effettuare il prelievo** riga per riga, infine lo si deve **chiudere**



Azioni su un cursore



apertura

OPEN NomeCursore;



prelievo

FETCH NomeCursore **INTO** ListaVariabili



chiusura

CLOSE NomeCursore;

Handler

Sono **gestori di situazioni**, utili *tra l'altro* quando si usano i cursori per riconoscere la fine del result set



possono essere definiti *subito dopo* le definizioni delle variabili e dei cursori

Esempio: il not found handler

esprime l'evento "oggetto non trovato", dove
l'oggetto è un record



DECLARE CONTINUE HANDLER FOR NOT FOUND
SET finito = 1;



È il tipico handler usato per gestire il "fine corsa"
nella scansione di un result set effettuata all'interno di
un ciclo dove si scorre un cursore

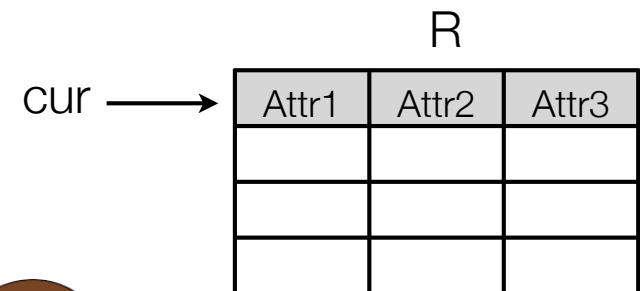
Funzionamento del *not found* handler

```
DECLARE finito INTEGER DEFAULT 0;
```

finito

0

```
DECLARE cur CURSOR FOR  
-- query che restituisce il result set R;
```

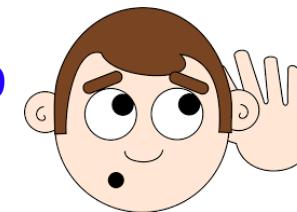


```
DECLARE CONTINUE HANDLER FOR NOT FOUND  
SET finito = 1;
```

/*

ciclo di fetch dove si scorre il cursore di una posizione
si preleva un record, e lo si processa. Se, avanzando
di una posizione, non c'è un nuovo record la campanella
suona, l'handler la sente, e setta la variabile finito a 1

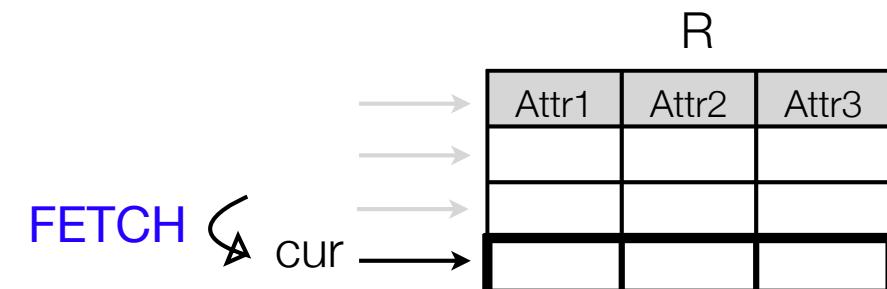
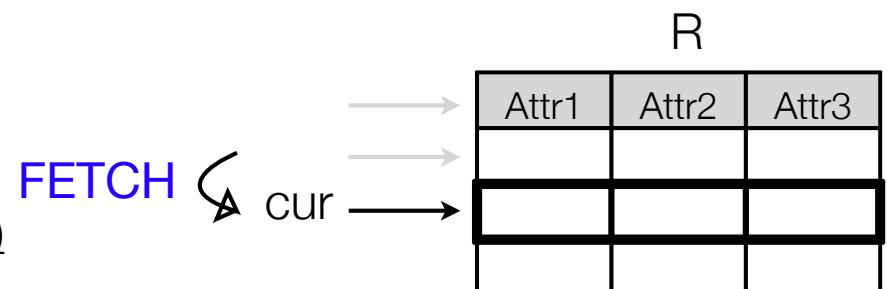
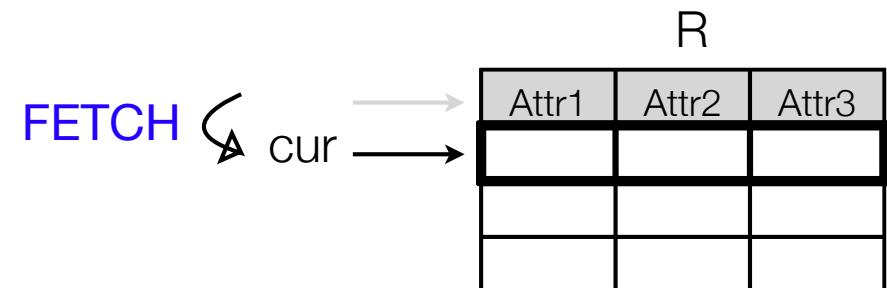
*/



Ciclo di fetch

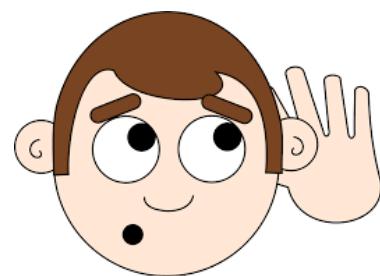
- 1: scan: **LOOP**
- 2: **FETCH cur** -- prelievo di un record da R
- 3: -- controllo *finito*
- 4: **IF finito è ancora 0**
- 5: -- il cursore ha prelevato un record
- 6: -- che può essere qui processato
- 7: **ELSE LEAVE scan;** -- altrimenti esco dal loop (*)
- 8: **END LOOP scan;**

(*) vedi slide successiva per l'ultimo fetch

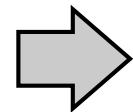


Ultimo fetch

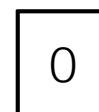
7: *finito* è stato messo a 1 dall'handler
e quindi non è stato possibile prelevare
un nuovo record da R, perché non c'è.
Se *finito*=,1 uscire dal ciclo!



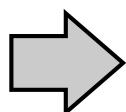
1



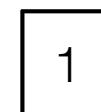
finito



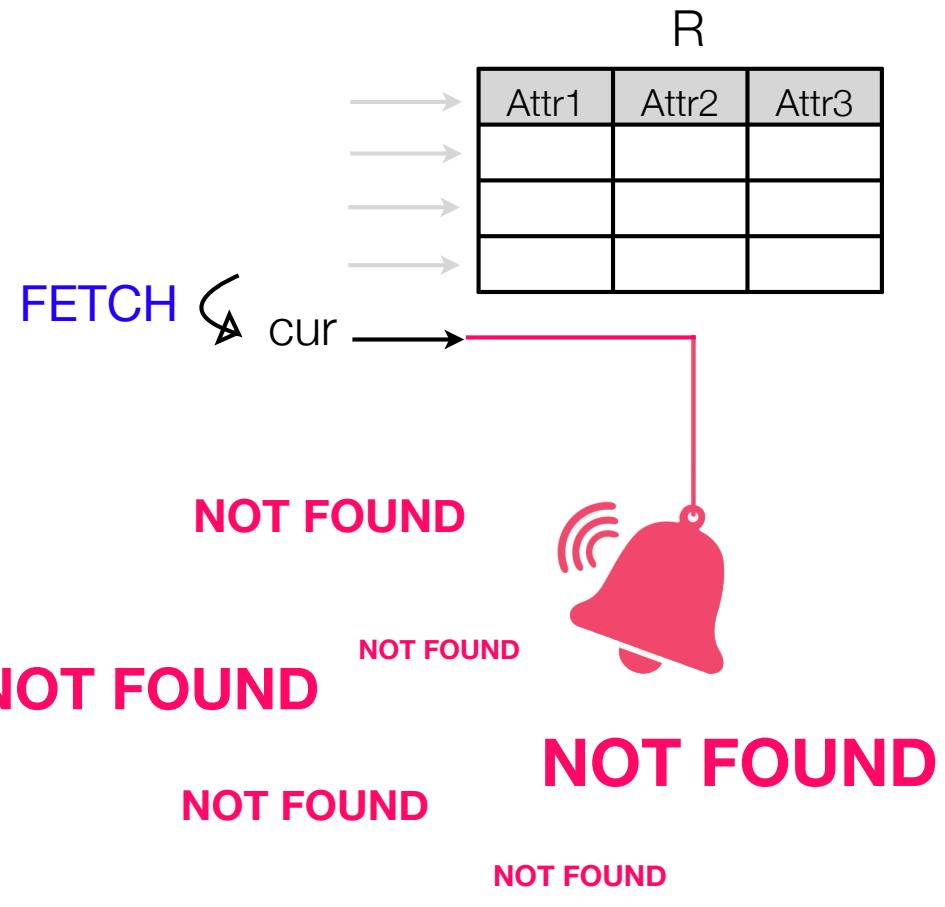
0



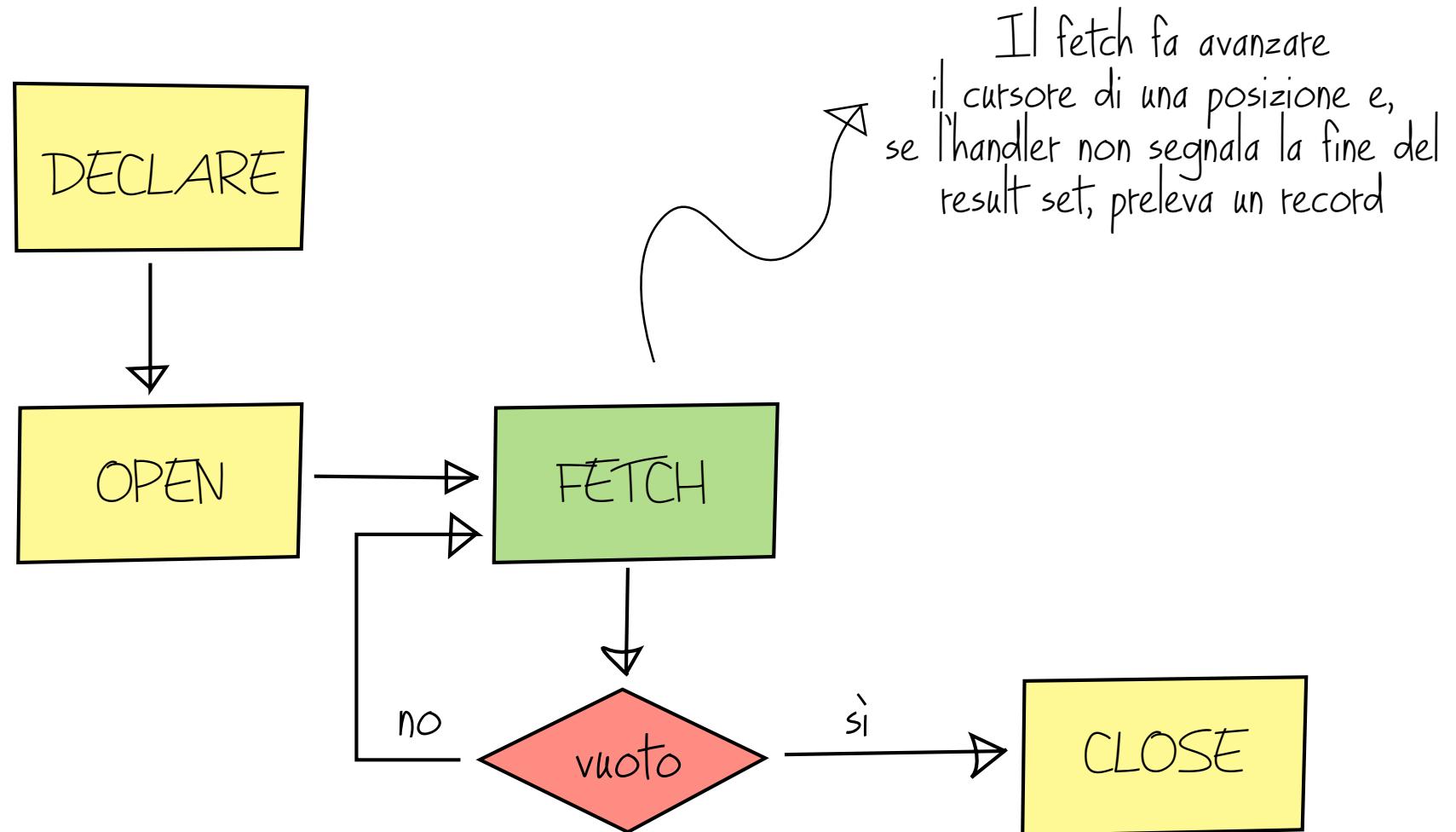
finito



1



Come funziona un cursore?



Stored procedure con cursore

Scrivere una stored procedure che riceve in ingresso una specializzazione s e restituisca i codici fiscali dei pazienti visitati da un solo medico di s, in una stringa del tipo “codFiscale1, ... , codFiscaleN”

```
1  DELIMITER $$  
2  CREATE PROCEDURE PazientiSingoloMedico(IN specializzazione CHAR,  
3 										  OUT codiciFiscali VARCHAR (255))  
4  BEGIN  
5      DECLARE finito INTEGER DEFAULT 0;  
6      DECLARE codiceFiscale VARCHAR(255) DEFAULT '';  
7  
8      -- dichiarazione del cursore  
9      DECLARE cursoreCodici CURSOR FOR  
10         SELECT V.Paziente  
11             FROM Visita V INNER JOIN Medico M ON V.Medico = M.Matricola  
12             WHERE M.Specializzazione = specializzazione  
13             GROUP BY V.Paziente  
14             HAVING COUNT(DISTINCT V.Medico) = 1;  
15  
16      -- dichiarazione handler  
17      DECLARE CONTINUE HANDLER  
18          FOR NOT FOUND SET finito = 1;  
19  
20      OPEN cursoreCodici;  
21  
22      -- ciclo di fetch per il prelievo  
23      preleva: LOOP  
24          FETCH cursoreCodici INTO codiceFiscale;  
25          IF finito = 1 THEN  
26              LEAVE preleva;  
27          END IF;  
28          SET codiciFiscali = CONCAT(codiceFiscale, ';', codiciFiscali);  
29      END LOOP preleva;  
30      CLOSE cursoreCodici;  
31  END $$  
32  DELIMITER ;
```

Chiamata

```
SET @codiciPazienti = '';
CALL PazientiSingoloMedico('Ortopedia', @codiciPazienti);
SELECT @codiciPazienti;
```

il risultato viene inserito in `@codiciPazienti`
come unica stringa formattata

Data Manipulation



Data Manipulation

Parte del linguaggio che permette di **inserire**, **cancellare** e **aggiornare** interi record o valori di attributi delle tabelle di un database

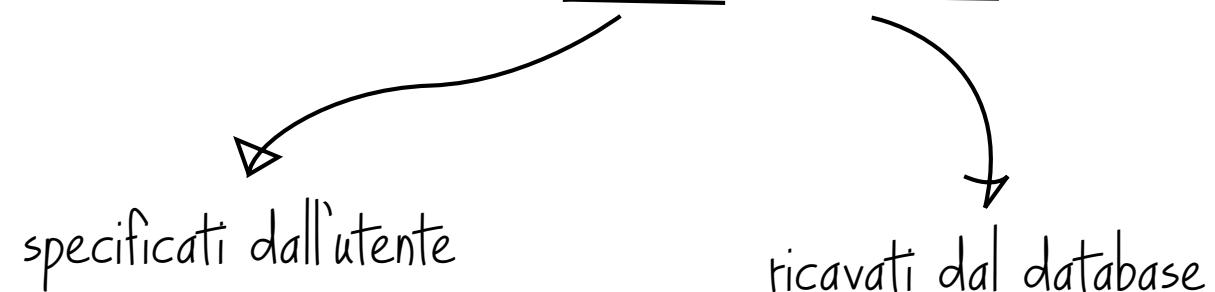
comandi **INSERT – DELETE – UPDATE**

Inserimento



Inserimento

Considerata una tabella, permette di **inserire un nuovo record** i cui valori degli attributi possono essere sia statici che ricavati



Inserimento con valori statici: esempio

Inserire nel database un nuovo paziente, le cui informazioni sono:

- Nome: Elvira
- Cognome: Passerotti
- Sesso: F
- Data di nascita: 27 Ottobre 1965
- Città: Pisa
- Reddito: 1500 €
- Codice fiscale: PSSLVR65R67G702U

INSERT INTO Paziente

VALUES ('PSSLVR65R67G702U', 'Passerotti', 'Elvira', 'F',
'1965-10-27', 'Pisa', 1500);

Non importa specificare
lo schema se inserisco tutti i campi
rispettando l'ordine dello schema

Inserimento: mancanza di informazioni

Tre giorni fa, la dottoressa Clelia Celesti ha visitato il paziente Edoardo Lepre, codice fiscale 'slq6'. La visita non è stata inserita. Lo si vuole fare in ritardo ma non si sa più se la visita era mutuata.

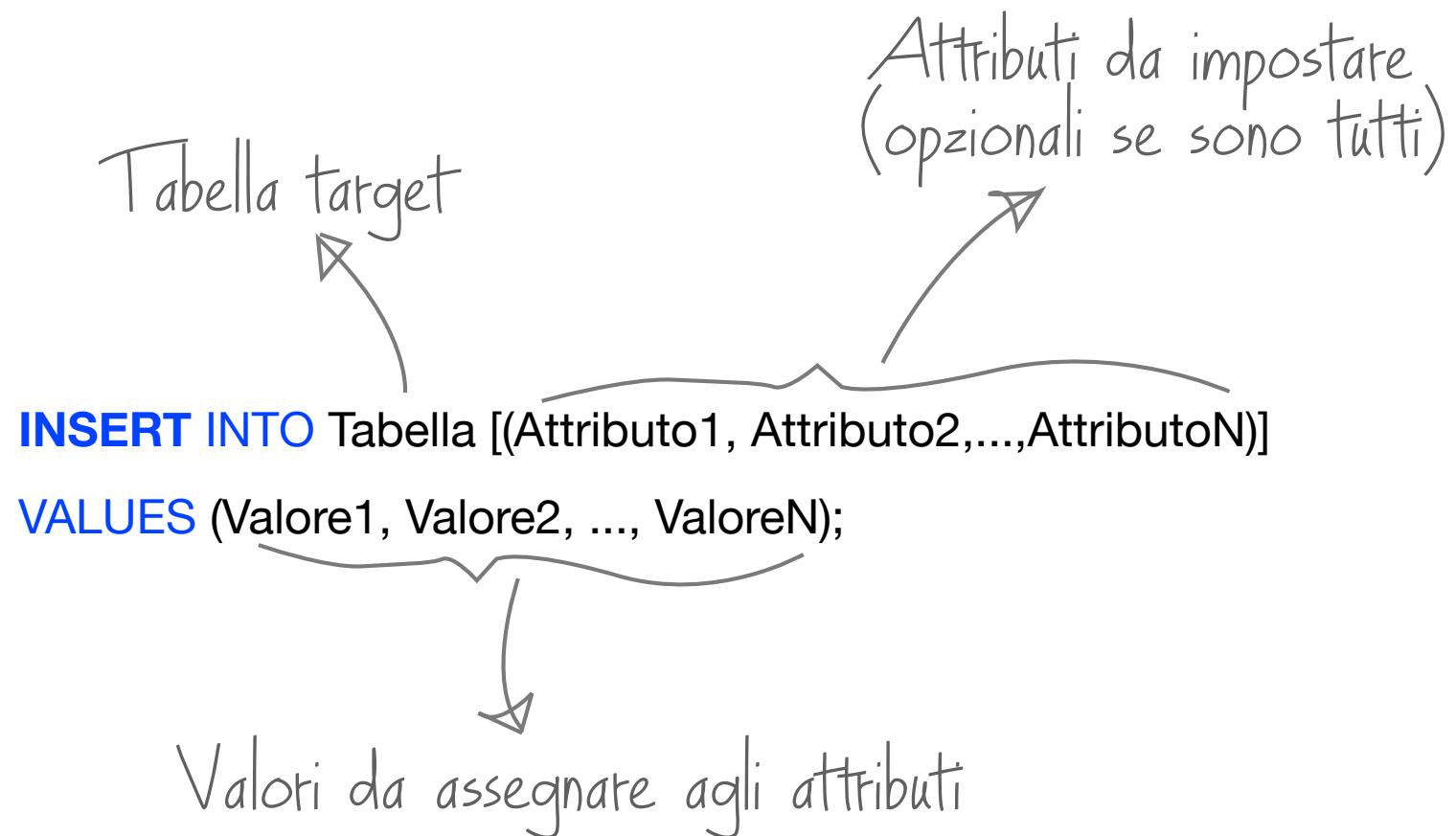
Attributi oggetto dell'inserimento



```
INSERT INTO Visita(Medico, Paziente, Data)
VALUES (010, 'slq6', CURRENT_DATE - INTERVAL 3 DAY);
```

L'informazione perduta (mutuata o no)
sarà impostata automaticamente al default value
(il default value di un attributo si imposta all'atto della creazione
della tabella, lo vedremo più avanti...)

Inserimento: sintassi MySQL



Inserimento con valori ricavati



Inserimento con valori ricavati: esempio

Considerata la tabella VISITAVECCHIA avente lo schema riportato sotto, inserire in VISITAVECCHIA tutte le visite effettuate prima di due anni fa.

VISITAVECCHIA (CognomePaziente, CognomeMedico, DataVisita, Specializzazione)

```
INSERT INTO VisitaVecchia
    SELECT P.Cognome AS CognomePaziente,
           M.Cognome AS CognomeMedico,
           V.Data AS DataVisita,
           M.Specializzazione
      FROM Visita V INNER JOIN Medico M ON V.Medico = M.Matricola
                INNER JOIN Paziente P ON V.Paziente = P.CodFiscale
     WHERE YEAR(V.Data) < YEAR(CURRENT_DATE) - 2;
```

Aggiornamento



Aggiornamento

Permette di **modificare il valore di uno o più attributi** di uno o più record
con valori statici oppure ricavati

l'insieme di record si seleziona
mediante una condizione

Aggiornamento: esempio

Modificare in “mutuata” tutte le visite del mese corrente, effettuate da
pazienti nati prima del 1925

UPDATE Visita

SET Mutuata = 1

WHERE MONTH(Data) = MONTH(CURRENT_DATE)

AND YEAR(Data) = YEAR(CURRENT_DATE)

AND Paziente IN (SELECT CodFiscale

FROM Paziente

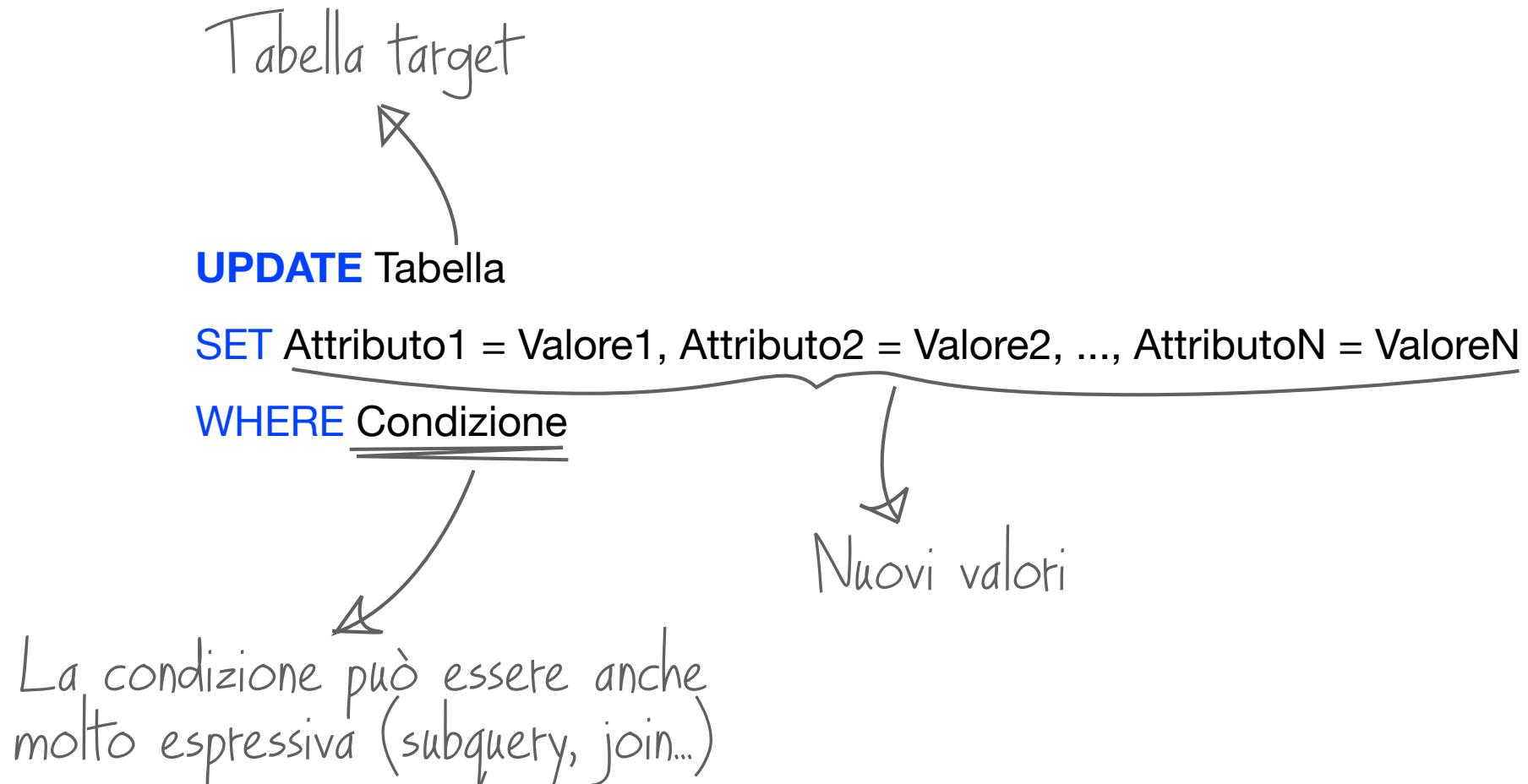
WHERE YEAR(DataNascita) < 1925);

esprimibile con subquery



per le condizioni, stessa sintassi delle query di selezione

Aggiornamento: sintassi MySQL



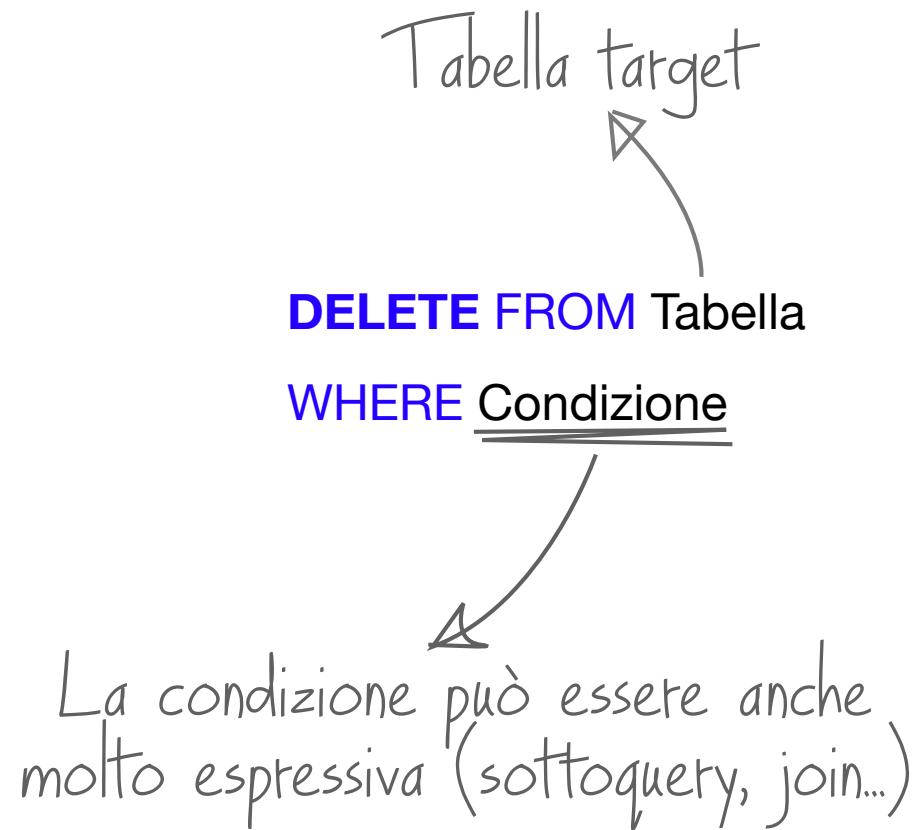
Cancellazione



Cancellazione

Permette di **cancellare uno o più record** dipendentemente dalla veridicità di una condizione, anche articolata

Cancellazione: sintassi MySQL



Cancellazione: esempio

Il dottor Ettore Grigi ha lasciato la clinica. Rimuovere tutte le sue visite dal database, supponendo che il medico non avesse omonimi nella clinica.

```
DELETE FROM Visita  
WHERE Medico = (SELECT Matricola  
                    FROM Medico  
                    WHERE Nome = 'Ettore' AND  
                          Cognome = 'Grigi');
```

Riferimento alla target table nella condizione

Rimuovere dal database tutti i medici di Pisa che non hanno effettuato visite mutuate il mese scorso.

Soluzione

Rimuovere dal database tutti i medici di Pisa che non hanno effettuato visite mutuate il mese scorso.

```
DELETE FROM Medico
WHERE Matricola IN
(
    SELECT M1.Matricola
    FROM Medico M1
    LEFT OUTER JOIN
    (
        SELECT V2.Medico AS MedicoMutuato Medici con almeno una visita il mese scorso
        FROM Visita V2
        INNER JOIN
            Medico M2 ON V2.Medico = M2.Matricola
        WHERE M2.Citta = "Pisa"
            AND V2.Mutuata IS TRUE
            AND YEAR(V2.Data) = YEAR(CURRENT_DATE)
            AND MONTH(V2.Data) = MONTH(CURRENT_DATE)-1
    ) AS D
    ON M1.Matricola = D.MedicoMutuato
    WHERE D.MedicoMutuato IS NULL
);
```

...e il delfino si arrabbia!

error : You can't specify target table 'MEDICO' for update
in FROM clause

Non si può mettere la tabella target
nel FROM della query contenuta nella
clausola WHERE!!!



Cosa non gli piace?

Rimuovere dal database tutti i medici di Pisa che non hanno effettuato visite mutuate il mese scorso.

```
DELETE FROM Medico
WHERE Matricola IN (
    SELECT M1.Matricola
    FROM Medico M1
    LEFT OUTER JOIN
    (
        SELECT V2.Medico AS MedicoMutuato
        FROM Visita V2 INNER JOIN Medico M2
        ON V2.Medico = M2.Matricola
        WHERE M2.Citta = "Pisa" AND V2.Mutuata IS TRUE
            AND YEAR(V2.Data) = YEAR(CURRENT_DATE)
            AND MONTH(V2.Data) = MONTH(CURRENT_DATE)-1
    ) AS D
    ON M1.Matricola = D.MedicoMutuato
    WHERE D.MedicoMutuato IS NULL );
```



Al mattino, pane e volpe!





Derived table

Rimuovere dal database tutti i medici di Pisa che non hanno effettuato visite mutuate il mese scorso.

```
DELETE FROM Medico
WHERE Matricola IN (
    SELECT * FROM (
        SELECT M1.Matricola
        FROM Medico M1
        LEFT OUTER JOIN(
            SELECT V2.Medico AS MedicoMutuato
            FROM Visita V2 INNER JOIN Medico M2
            ON V2.Medico = M2.Matricola
            WHERE M2.Citta = 'Pisa' AND V2.Mutuata IS TRUE
            AND YEAR(V2.Data) = YEAR(CURRENT_DATE)
            AND MONTH(V2.Data) = MONTH(CURRENT_DATE)-1
        ) AS D
        ON M1.Matricola = D.MedicoMutuato
        WHERE D.MedicoMutuato IS NULL
    ) AS D2 );
```

Occhio non vede, cuore non duole...

Join anticipato

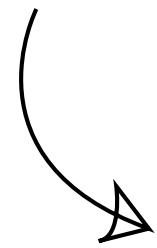


Rimuovere dal database tutti i medici di Pisa che non hanno effettuato visite mutuate il mese scorso.

```
DELETE M1.*  
FROM Medico M1  
    LEFT OUTER JOIN (  
        SELECT V2.Medico AS MedicoMutuato  
        FROM Visita V2 INNER JOIN Medico M2 ON V2.Medico = M2.Matricola  
        WHERE M2.Citta = 'Pisa' AND V2.Mutuata IS TRUE  
            AND YEAR(V2.Data) = YEAR(CURRENT_DATE)  
            AND MONTH(V2.Data) = MONTH(CURRENT_DATE)-1  
    ) AS D  
    ON M1.Matricola = D.MedicoMutuato  
WHERE D.MedicoMutuato IS NULL;
```

...e anche in aggiornamento

Anche quando si effettua un aggiornamento (comando update) MySQL
vieta l'inserimento della tabella target nella condizione WHERE.



si possono usare le derived table
oppure il join anticipato

Aggiornamento con join anticipato

Aumentare del 5% la parcella ai medici che hanno effettuato nell'ultimo anno più della metà delle visite della loro specializzazione.

```
UPDATE Medico M
    NATURAL JOIN
    (
        SELECT V1.Medico, COUNT(*) AS TotaleVisiteMedico,
        (
            SELECT COUNT(*)
            FROM Visita V2 INNER JOIN Medico M2
                ON V2.Medico = M2.Matricola
            WHERE M2.Specializzazione = M1.Specializzazione
        ) AS TotaleVisiteSpecializzazione
        FROM Visita V1
        INNER JOIN
            Medico M1 ON V1.Medico = M1.Matricola
        GROUP BY V1.Medico ) AS D
    SET M.Parcella = M.Parcella*1.05
    WHERE D.TotaleVisiteMedico > D.TotaleVisiteSpecializzazione*0.5);
```