EPICODE

Modificare ed eliminare i dati

AGENDA

- ❑ Modificare i dati
- ❑ Eliminare i dati

la fine di questo modulo sarai in grado di:

- ✔ Eseguire correttamente operazioni di update
- ✔ Eseguire correttamente operazioni di eliminazione record

UPDATE

UPDATE è un'istruzione DML che modifica le righe di una tabella.

In generale, la sintassi è questa:

UPDATE *NomeTabella*
SET *colonna1* = *valore1*, *colonna2* = *valore2*, ...
WHERE *condizione di ricerca*;

UPDATE

UPDATE è un'istruzione DML che modifica le righe di una tabella.

Supponiamo di dover correggere un valore del campo ProductName nell'anagrafica prodotti.
Ipotizziamo pertanto che ci sia un errore in fase di inserimento dati e bisogna correggere il valore del campo ProductName del codice prodotto 349 da 'Motain-100 Black, 42' a 'Mountain-100 Black, 42'

ProductID	ProductName	Size
349	Motain-100 Black, 42	42
350	Mountain-100 Black, 44	44
351	Mountain-100 Black, 48	48

```
UPDATE Product
SET ProductName = 'Mountain-100 Black, 42'
WHERE ProductID = 349
```

ProductID	ProductName	Size
349	Mountain-100 Black, 42	42
350	Mountain-100 Black, 44	44
351	Mountain-100 Black, 48	48

UPDATE

UPDATE è un'istruzione DML che modifica le righe di una tabella.

Supponiamo di dover correggere un valore del campo Size nell'anagrafica prodotti.
Ipotizziamo pertanto che ci sia un errore in fase di inserimento dati nel campo Size. In particolare, il codice prodotto 350 ha una valore in Size di 45 quando dovrebbe essere 44

ProductID	ProductName	Size
349	Mountain-100 Black, 42	42
350	Mountain-100 Black, 44	45
351	Mountain-100 Black, 48	48

```
UPDATE Product
SET Size = 44
WHERE ProductID = 350
```

ProductID	ProductName	Size
349	Mountain-100 Black, 42	42
350	Mountain-100 Black, 44	44
351	Mountain-100 Black, 48	48

UPDATE

UPDATE è un'istruzione DML che modifica le righe di una tabella.

Supponiamo di dover correggere i valori di più campi contemporaneamente.
In questo caso, occorre correggere il valore 43 di Size in 42 e il nome del prodotto per il codice prodotto 349.

ProductID	ProductName	Size
349	Motain-100 Black, 42	43
350	Mountain-100 Black, 44	44
351	Mountain-100 Black, 48	48

```
UPDATE Product
SET ProductName = 'Mountain-100 Black, 42', Size = 42
WHERE ProductKey = 349
```

ProductID	ProductName	Size
349	Mountain-100 Black, 42	42
350	Mountain-100 Black, 44	44
351	Mountain-100 Black, 48	48

ACID

I database relazioni sono progettati per garantire:

- **Atomicità** delle transazioni: le transazioni vengono eseguite come singola unità di lavoro.
- **Coerenza (o consistenza)** del db: le transazioni lasciano il db in uno stato coerente.
- **Isolamento**: le transazioni non possono interferire una con l'altra.. Le transazioni simultanee sono gestite in maniera sequenziale
- **Durabilità**: il risultato delle transazioni è persistente

ProductID	ProductName	Size
349	Motain-100 Black, 42	43
350	Mountain-100 Black, 44	44
351	Mountain-100 Black, 48	48

```
UPDATE Product
SET ProductName = 'Mountain-100 Black, 42', Size = 42
WHERE ProductKey = 349
```

ProductID	ProductName	Size
349	Mountain-100 Black, 42	42
350	Mountain-100 Black, 44	44
351	Mountain-100 Black, 48	48

ACID

I database relazioni sono progettati per garantire:

- L'**Atomicità** delle transazioni: le transazioni vengono eseguite come singola unità di lavoro (con un approccio 'o tutto o niente').
- La **Coerenza (o consistenza)** del db: le transazioni lasciano il db in uno stato coerente.
- **Isolamento**: le transazioni non possono interferire una con l'altra.. Le transazioni simultanee sono gestite in maniera sequenziale.
- **Durabilità**: il risultato delle transazioni è persistente (non volatile).

ACID

I database relazioni sono progettati per garantire:

- L'**Atomicità** delle transazioni: le transazioni vengono eseguite come singola unità di lavoro ('o tutto o niente')
- La **Coerenza (o consistenza)** del db: le transazioni lasciano il db in uno stato coerente.
- **Isolamento**: le transazioni non possono interferire una con l'altra.. Le transazioni simultanee sono gestite in maniera sequenziale
- **Durabilità**: il risultato delle transazioni è persistente

```
CREATE TABLE SalesTerritory (
TerritoryID INT
, RegionName VARCHAR(25)
, CONSTRAINT PK_TerritoryID PRIMARY KEY (TerritoryID))

INSERT INTO SalesTerritory
VALUES (1, 'Italy'), ('Germany', 2)

SELECT *
FROM SalesTerritory
```

La transazione viene eseguita come singola unità di lavoro: non viene scritto nessun record e l'intera transazione ha esito negativo.

Il result set è vuoto.

ACID

```
CREATE TABLE SalesTerritory (
TerritoryID INT
, RegionName VARCHAR(25)
, CONSTRAINT PK_TerritoryID
    PRIMARY KEY (TerritoryID))

INSERT INTO SalesTerritory
VALUES (1, 'Italy'), ('Germany', 2)

SELECT *
FROM SalesTerritory
```

TerritoryID	RegionName
1	Italy
2	Germany

```
CREATE TABLE SalesPerson (
SalespersonID INT
, TerritoryID INT
, FirstName VARCHAR(25)
, LastName VARCHAR (25)
, CONSTRAINT PK_SalespersonID PRIMARY KEY
(SalespersonID)
, CONSTRAINT
FK_SalesTerritory_SalesPerson_TerritoryID
FOREIGN KEY (TerritoryID)
REFERENCES SalesTerritory (TerritoryID))

INSERT INTO SalesPerson
VALUES (1, 3, 'xyz', 'zzz')
```

La transazione fallisce. Il database è lasciato in uno stato consistente!

ACID

```
CREATE TABLE SalesTerritory (
TerritoryID INT
, RegionName VARCHAR(25)
, CONSTRAINT PK_TerritoryID
    PRIMARY KEY (TerritoryID))

INSERT INTO SalesTerritory
VALUES (1, 'Italy'), ('Germany', 2)

SELECT *
FROM SalesTerritory
```

TerritoryID	RegionName
1	Italy
2	Germany

```
CREATE TABLE SalesPerson (
SalespersonID INT
, TerritoryID INT
, FirstName VARCHAR(25)
, LastName VARCHAR (25)
, CONSTRAINT PK_SalespersonID PRIMARY KEY
(SalespersonID)
, CONSTRAINT
FK_SalesTerritory_SalesPerson_TerritoryID
FOREIGN KEY (TerritoryID)
REFERENCES SalesTerritory (TerritoryID))

INSERT INTO SalesPerson
VALUES (1, 2, 'xyz', 'zzz')
```

L'integrità referenziale non è violata. La transazione si conclude correttamente.

ACID

```
CREATE TABLE SalesPerson (
SalespersonID INT
, TerritoryID INT
, FirstName VARCHAR(25)
, LastName VARCHAR (25)
, CONSTRAINT PK_SalespersonID PRIMARY KEY (SalespersonID)
, CONSTRAINT FK_SalesTerritory_SalesPerson_TerritoryID
FOREIGN KEY (TerritoryID)
REFERENCES SalesTerritory (TerritoryID))

INSERT INTO SalesPerson
VALUES (1, 2, 'xyz', 'zzz')
```

SalespersonID	TerritoryID	FirstName	LastName
1	2	xyz	zzz

ACID

SalespersonID	TerritoryID	FirstName	LastName
1	2	xyz	zzz
2	2	abc	yyy
3	1	luc	cnc

TerritoryID	RegionName
1	Italy
2	Germany

```
UPDATE SalesPerson
SET TerritoryID = 4
WHERE SalespersonID = 1
```

Il numero di record affette da aggiornamento è 0!
Il database è lasciato in uno stato consistente. Non c'è alcuna violazione di integrità referenziale.

ACID

Supponiamo di dover aggiornare l'anagrafica agenti (Salesperson) per assegnare la regione *Italy* all'agente *abc yyy*

SalespersonID	TerritoryID	FirstName	LastName
1	2	xyz	zzz
2	2	abc	yyy
3	1	luc	cnc

TerritoryID	RegionName
1	Italy
2	Germany

```
UPDATE SalesPerson
SET TerritoryID = 1
WHERE SalespersonID = 1
```

SalespersonID	TerritoryID	FirstName	LastName
1	1	xyz	zzz
2	2	abc	yyy
3	1	luc	cnc

Come ci comportiamo se abbiamo sbagliato la transazione?
... Il risultato della transazione è persistente

ACID

Supponiamo di dover aggiornare l'anagrafica agenti (Salesperson) per assegnare la regione *Italy* all'agente *abc yyy*

SalespersonID	TerritoryID	FirstName	LastName
1	2	xyz	zzz
2	2	abc	yyy
3	1	luc	cnc

TerritoryID	RegionName
1	Italy
2	Germany

```
UPDATE SalesPerson
SET TerritoryID = 1
WHERE SalespersonID = 1
```

SalespersonID	TerritoryID	FirstName	LastName
1	1	xyz	zzz
2	2	abc	yyy
3	1	luc	cnc

Come ci comportiamo se abbiamo sbagliato la transazione?
... Il risultato della transazione è persistente

ACID

```
START TRANSACTION

UPDATE SalesPerson
SET TerritoryID = 4
WHERE SalespersonID = 1

SELECT *
FROM SalesPerson

ROLLBACK
```

con **START TRANSACTION** è possibile indicare l'inizio di una transazione esplicita

UPDATE sui dati

È possibile eseguire una query per visualizzare il risultato ottenibile prima di 'cancellarlo' facendo il rollback della transazione

ROLLBACK consente di cancellare tutte le modifiche dei dati eseguite a partire dall'inizio della transazione. I risultati della transazione non sono persistenti.

ACID

```
START TRANSACTION

UPDATE SalesPerson
SET TerritoryID = 1
WHERE SalespersonID = 2

SELECT *
FROM SalesPerson

COMMIT
```

con **START TRANSACTION** è possibile indicare l'inizio di una transazione esplicita

UPDATE sui dati

È possibile eseguire una query per visualizzare il risultato ottenibile prima di 'cancellarlo' facendo il rollback della transazione

COMMIT rende permanenti nel database tutte le modifiche apportate ai dati dall'inizio della transazione.

DELETE

UPDATE è un'istruzione DML che elimina le righe di una tabella.

In generale, la sintassi è questa:

DELETE FROM *NomeTabella*
WHERE *condizione di ricerca*

È necessario sempre specificare una condizione di ricerca per filtrare solo i record che occorre eliminare.
Se non si specifica una condizione di ricerca (o più condizioni di ricerca) vengono cancellati tutti i record della tabella (a meno che non si verificano errori dovuti a violazione di vincoli).

L'operazione DELET FROM NomeTabella cancella tutti i record della tabella

DELETE

Supponiamo di dover eliminare un record a seguito, per esempio, di un inserimento errato.
Supponiamo pertanto di dover eliminare il record relativo al SalespersonID = 2

SalespersonID	TerritoryID	FirstName	LastName
1	2	xyz	zzz
2	1	abc	yyy
3	1	luc	cnc

```
BEGIN

DELETE FROM SalesPerson
WHERE SalespersonID = 2

SELECT *
FROM SalesPerson

ROLLBACK
```

SalespersonID	TerritoryID	FirstName	LastName
1	2	xyz	zzz
3	1	luc	cnc

Risultato visualizzabile ma non persistente fintantoché non si esegue il COMMIT della transazione.

L'utilità della SELECT nella transazione è proprio nel poter aver un controllo sul risultato prima che venga scritto sul db! Conclusa la transazione il db resta allo stato precedente..!

DELETE

Supponiamo di dover eliminare un record a seguito, per esempio, di un inserimento errato.
Supponiamo pertanto di dover eliminare il record relativo al SalespersonID = 2

SalespersonID	TerritoryID	FirstName	LastName
1	2	xyz	zzz
2	1	abc	yyy
3	1	luc	cnc

```
BEGIN

DELETE FROM SalesPerson
WHERE SalespersonID = 2

SELECT *
FROM SalesPerson

COMMIT
```

SalespersonID	TerritoryID	FirstName	LastName
1	2	xyz	zzz
3	1	luc	cnc

Il risultato è scritto in maniera persistente sul database.

SQL

LET'S TAKE A LOOK!

EPICODE

GRAZIE

Epicode