Politecnico di Torino

Costrutti avanzati

Linguaggio SQL

Linguaggio SQL: costrutti avanzati

- Viste
- Transazioni
- Controllo del'accesso
- Gestione degli indici
- Progettazione fisica

Viste

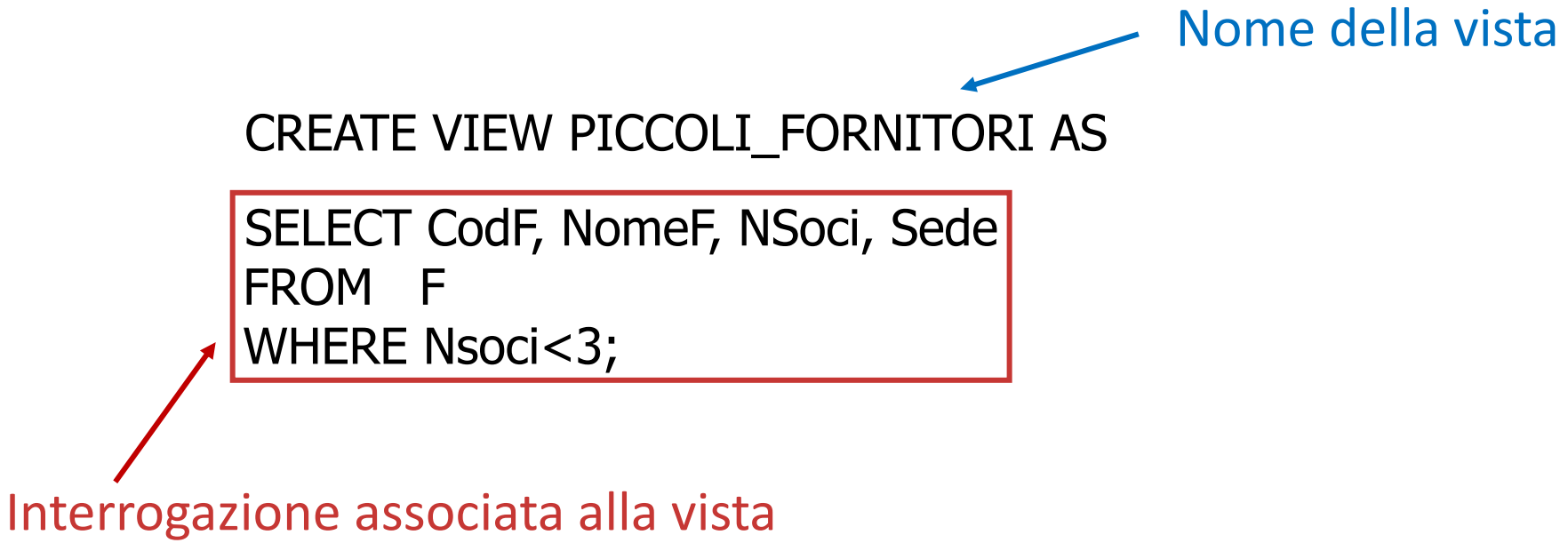
Costrutti avanzati

Concetto di vista

- La vista è una *tabella "virtuale"*
  - il contenuto (tuple) è definito mediante un'interrogazione SQL sulla base di dati
    - il contenuto della vista dipende dal contenuto delle altre tabelle presenti nella base di dati
  - il contenuto *non* è memorizzato fisicamente nella basi di dati
    - è ricalcolato tutte le volte che si usa la vista eseguendo l'interrogazione che la definisce
- La vista è un oggetto della base di dati
  - è utilizzabile nelle interrogazioni come se fosse una tabella
- Se l'interrogazione fa riferimento a una vista, deve essere riscritta dal DBMS prima dell'esecuzione
- La riscrittura è svolta automaticamente
  - si sostituiscono i riferimenti alla vista con la sua definizione

Esempio n.1: definizione della vista

- Definizione della vista *piccoli fornitori*
  - i fornitori che hanno meno di 3 soci sono considerati “piccoli fornitori”
- La vista piccoli fornitori
  - contiene il codice, il nome, il numero di soci e la sede dei fornitori che hanno meno di 3 soci



Esempio n.1: interrogazione

- Visualizzare il codice, il nome, la sede e il numero di soci dei piccoli fornitori di Torino
- L'interrogazione può essere risolta senza l'uso di viste

```
SELECT *
FROM   F
WHERE NSoci<3 AND Sede='Torino';
```

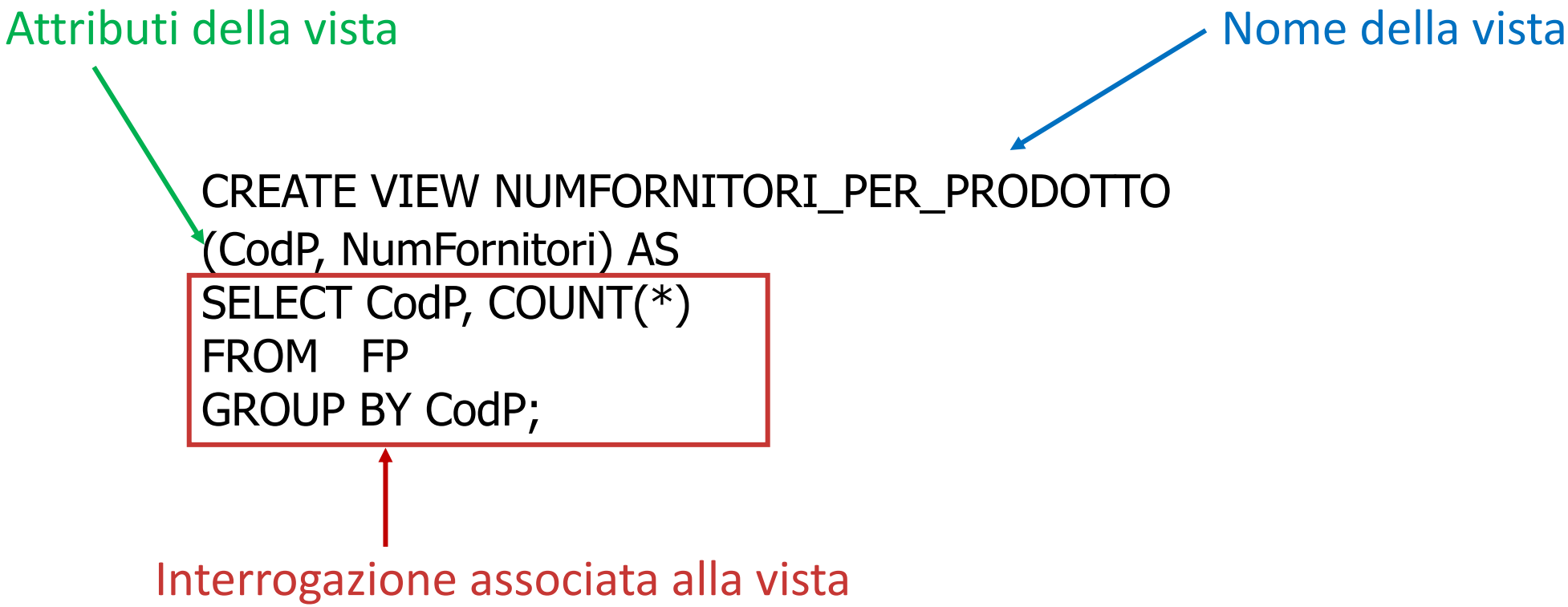
- L'interrogazione può essere risolta usando la vista definita in precedenza

```
SELECT *
FROM   PICCOLI_FORNITORI
WHERE Sede='Torino';
```

- La vista PICCOLI_FORNITORI è usata come se fosse una tabella

Esempio n.2: definizione della vista

- Definizione della vista numero di fornitori per prodotto
  - per ogni prodotto, contiene il codice prodotto e il numero di fornitori diversi che l'hanno fornito



Vantaggi offerti dalle viste

- Semplificazione delle interrogazioni
  - scomposizione di un'interrogazione complessa in sottointerrogazioni associate alle viste
- Gestione della sicurezza
  - è possibile introdurre meccanismi di protezione della privatezza diversi per ogni utente o gruppo
    - la vista diviene l'elemento a cui sono associate le autorizzazioni di accesso
    - ogni utente, o gruppo, accede alla base di dati solo mediante viste appropriate per le operazione che è abilitato a svolgere
- Evoluzione della base di dati
  - in caso di ristrutturazione di una base di dati, è sufficiente ridefinire le viste
    - non si devono riscrivere le interrogazioni scritte prima della ristrutturazione e presenti nelle applicazioni già sviluppate

Creazione viste

```
CREATE VIEW NomeVista [(ElencoAttributi) ]
AS InterrogazioneSQL;
```

- Se i nomi degli attributi della vista non sono specificati
  - sono utilizzati quelli presenti nella select dell'interrogazione SQL
- I nomi degli attributi devono essere specificati se
  - rappresentano il risultato di una funzione interna
  - rappresentano il risultato di un'espressione
  - sono costanti
  - due colonne (provenienti da tabelle diverse) hanno lo stesso nome

Cancellazione viste

```
DROP VIEW NomeVista;
```

- La cancellazione di una tabella a cui fa riferimento una vista può avere effetti diversi
  - eliminazione automatica delle viste associate
  - invalidazione automatica delle viste associate
  - divieto di esecuzione dell'operazione di cancellazione della tabella
- L'effetto dipende dal DBMS utilizzato

Aggiornabilità delle viste

- È possibile eseguire operazioni di aggiornamento dei dati presenti in una vista *solo* per alcune tipologie di viste
- Sono aggiornabili le viste in cui una sola riga di ciascuna tabella di base corrisponde a una sola riga della vista (SQL-92)
  - corrispondenza univoca tra le tuple della vista e le tuple della tabella su cui è definita
  - è possibile propagare senza ambiguità le modifiche apportate sulla vista verso ogni tabella su cui è definita
- *Non è aggiornabile* una vista che, nel blocco più esterno dell'interrogazione che la definisce
  - non contiene la chiave primaria della tabella su cui è definita
  - contiene join che rappresentano corrispondenze uno a molti o molti a molti
  - contiene funzioni aggregate
  - contiene DISTINCT
- Alcune viste non aggiornabili possono diventare aggiornabili modificando l'espressione SQL associata alla vista
  - può essere necessario ridurre il contenuto informativo della vista

Esempio n.1

- Vista FORNITORE_SEDE

```
CREATE VIEW FORNITORE_SEDE AS
SELECT CodF, Sede
FROM   F;
```

Esempio n.1: inserimento

- Inserimento in FORNITORE_SEDE di

('F10', 'Roma')

corrisponde all'inserimento in F di

('F10', NULL, NULL, 'Roma')

- gli attributi NomeF, NSoci devono ammettere il valore NULL

Esempio n.1: cancellazione

- Cancellazione da FORNITORE_SEDE di

('F1', 'Torino')

➡ cancellazione da F di

('F1', 'Andrea', 2, 'Torino')

- l'identificazione della tupla da cancellare è permessa dalla chiave primaria

Esempio n.1: modifica

- Modifica in FORNITORE_SEDE di

('F1', 'Torino') in ('F1', 'Milano')

➡ modifica in F di

('F1', 'Andrea', 2, 'Torino') in ('F1', 'Andrea', 2, 'Milano')

- l'identificazione della tupla da modificare è permessa dalla chiave primaria

Esempio n.1: aggiornabilità

- La vista FORNITORE_SEDE *è aggiornabile*
  - ogni tupla della vista corrisponde a una sola tupla della tabella F
  - le operazioni di modifica effettuate sulla vista possono essere propagate alla tabella su cui è definita

Esempio n.2

- Vista NUMSOCI_SEDE

```
CREATE VIEW NUMSOCI_SEDE AS
SELECT DISTINCT NSoci, Sede
FROM   F;
```

Esempio n.2: inserimento

- Inserimento in NUMSOCI_SEDE di

(40, ‘Napoli’)

➡ impossibile inserire in F

(NULL, NULL, 40, ‘Napoli’)

- manca il valore della chiave primaria

Esempio n.2: cancellazione

- Cancellazione da NUMSOCI_SEDE di

(2, ‘Torino’)

- più tuple sono associate alla coppia (2, ‘Torino’)
  - quale tupla deve essere cancellata da F?

Esempio n.2: modifica

- Modifica in NUMSOCI_SEDE di

(2, ‘Torino’) in (3, ‘Milano’)

- più tuple sono associate alla coppia (2, ‘Torino’)
  - quale tupla deve essere modificata in F?

Esempio n.2: aggiornabilità

- La vista NUMSOCI_SEDE *non è aggiornabile*
  - non è presente la chiave primaria della tabella F nella vista
    - l'inserimento di nuove tuple nella vista non è propagabile a F
  - alcune tuple della vista corrispondono a più tuple della tabella F
    - l'associazione tra tuple nella vista e tuple nella tabella è ambigua
    - non è possibile propagare le modifiche effettuate su tuple della vista alle tuple della tabella su cui è definita

Esempio n.3: vista non aggiornabile

```
CREATE VIEW FORNITORI_TORINO AS
SELECT *
FROM F
WHERE Sede='Torino';
```

- La vista non è aggiornabile
  - non seleziona in modo esplicito la chiave primaria della tabella F
- È sufficiente sostituire al simbolo "*" il nome degli attributi

Esempio n.4: vista non aggiornabile

```
CREATE VIEW FORNITORI_IMPORTANTI (CodF,
NomeF) AS
SELECT DISTINCT CodF, NomeF
FROM    F, FP
WHERE F.CodF=FP.CodF AND
            Qta>100;
```

- La vista non è aggiornabile
  - è presente un join
  - è presente la parola chiave DISTINCT

Esempio n.4: vista modificata

```
CREATE VIEW FORNITORI_IMPORTANTI (CodF, NomeF) AS
SELECT CodF, NomeF
FROM   F
WHERE CodF IN (SELECT CodF
                      FROM   FP
                      WHERE Qta>100);
```

- La vista è aggiornabile
  - il join è stato realizzato mediante IN
  - la parola chiave DISTINCT non è più necessaria

Non sempre è possibile riscrivere la query per rendere la vista aggiornabile

Transazioni

Costrutti avanzati

Transazione

- Necessaria quando più utenti possono accedere contemporaneamente ai dati
- Offre meccanismi efficienti per
  - gestire l'accesso concorrente ai dati
  - effettuare il recovery a seguito di un malfunzionamento
- E' un'unità logica di lavoro, non ulteriormente scomponibile
  - una sequenza di operazioni (istruzioni SQL) di modifica dei dati, che porta la base di dati da uno stato consistente a un altro stato consistente
  - non è necessario conservare la consistenza negli stati intermedi
- Un sistema che mette a disposizione un meccanismo per la definizione e l'esecuzione di transazioni viene detto *sistema transazionale*
- I DBMS contengono blocchi architetturali che offrono servizi di gestione delle transazioni

Inizio di una transazione

- Per definire l'inizio di una transazione, il linguaggio SQL prevede l'istruzione
  - START TRANSACTION
- Di solito l'istruzione di inizio della transazione è omessa
  - l'inizio è implicito
    - prima istruzione SQL del programma che accede alla base di dati
    - prima istruzione SQL successiva all'istruzione di termine della transazione precedente

Fine di una transazione

- Il linguaggio SQL prevede istruzioni per definire la fine di una transazione
  - Transazione terminata con successo
    - COMMIT [WORK]
    - l'azione associata all'istruzione si chiama *commit*
  - Transazione terminata con insuccesso
    - ROLLBACK [WORK]
    - l'azione associata all'istruzione si chiama *abort*

Commit

- Azione eseguita quando una transazione termina con successo
- La base di dati è in un nuovo stato (finale) corretto
- Le modifiche dei dati effettuate dalla transazione divengono
  - permanenti
  - visibili agli altri utenti

Rollback

- Azione eseguita quando una transazione termina a causa di un errore
  - per esempio, di un errore applicativo
- Tutte le operazioni di modifica dei dati eseguite durante la transazione sono “annullate”
- La base di dati ritorna nello stato precedente l’inizio della transazione
  - i dati sono nuovamente visibili agli altri utenti

Esempio

- Trasferire la somma 100
  - dal conto corrente bancario IT92X0108201004300000322229
  - al conto corrente bancario IT32L0201601002410000278976

```
START TRANSACTION;

UPDATE Conto-Corrente
    SET Saldo = Saldo - 100
    WHERE IBAN='IT92X0108201004300000322229';

UPDATE Conto-Corrente
    SET Saldo = Saldo + 100
    WHERE IBAN= 'IT32L0201601002410000278976';

COMMIT;
```

Proprietà delle transazioni

- Le proprietà principali delle transazioni sono
  - Atomicity – atomicità
  - Consistency – consistenza
  - Isolation – isolamento
  - Durability – persistenza (o durabilità)
- Sono riassunte dall'acronimo (inglese) *ACID*

Atomicità

- Una transazione è un’unità indivisibile (atomo) di lavoro
  - devono essere eseguite tutte le operazioni contenute nella transazione
  - oppure nessuna delle operazioni contenute nella transazione deve essere eseguita
    - la transazione non ha nessun effetto sulla base di dati
- La base di dati non può rimanere in uno stato intermedio assunto durante l’esecuzione di una transazione

Consistenza

- L’esecuzione di una transazione deve portare la base di dati
  - da uno stato iniziale consistente (corretto)
  - a uno stato finale consistente
- La correttezza è verificata dai vincoli di integrità definiti sulla base di dati
- Quando si verifica la violazione di un vincolo di integrità il sistema interviene
  - per annullare la transazione
  - oppure, per modificare lo stato della base di dati
    eliminando la violazione del vincolo

Isolamento

- L’esecuzione di una transazione è indipendente dalla contemporanea esecuzione di altre transazioni
- Gli effetti di una transazione non sono visibili dalle altre transazioni fino a quando la transazione non è terminata
  - si evita la visibilità di stati intermedi non stabili
    - uno stato intermedio può essere annullato da un rollback successivo
    - in caso di rollback, è necessario effettuare rollback delle altre transazioni che hanno osservato lo stato intermedio (effetto domino)

Persistenza

- L’effetto di una transazione che ha effettuato il commit è memorizzato in modo permanente
  - le modifiche dei dati eseguite da una transazione terminata con successo sono permanenti dopo il commit
- Garantisce l’affidabilità delle operazioni di modifica dei dati
  - i DBMS offrono meccanismi di ripristino dello stato corretto della base di dati dopo che si è verificato un guasto

Controllo dell'accesso

Costrutti avanzati

Sicurezza dei dati

- Protezione dei dati da
  - letture non autorizzate
  - alterazione o distruzione
- Il DBMS fornisce strumenti per realizzare le protezioni, che sono definite dall'amministratore della base dati (DBA)
- Il controllo della sicurezza verifica che gli utenti siano autorizzati a eseguire le operazioni che richiedono di eseguire
- La sicurezza è garantita attraverso un insieme di vincoli
  - specificati dal DBA in un opportuno linguaggio
  - memorizzati nel dizionario dei dati del sistema

Risorse

- Qualsiasi componente dello schema di una base di dati è una risorsa
  - tabella
  - vista
  - attributo all'interno di una tabella o di una vista
  - dominio
  - procedura
  - …
- Le risorse sono protette mediante la definizione di *privilegi di accesso*

Privilegi di accesso

- Descrivono i diritti di accesso alle risorse del sistema
- SQL offre meccanismi di controllo dell'accesso molto flessibili mediante i quali è possibile specificare
  - le risorse a cui possono accedere gli utenti
  - le risorse che devono essere mantenute private

Privilegi: caratteristiche

- Ogni privilegio è caratterizzato dalle seguenti informazioni
  - la risorsa a cui si riferisce
  - il tipo di privilegio
    - descrive l'azione permessa sulla risorsa
  - l'utente che concede il privilegio
  - l'utente che riceve il privilegio
  - la facoltà di trasmettere il privilegio ad altri utenti

Tipi di privilegi

- INSERT
  - permette di inserire un nuovo oggetto nella risorsa
  - vale per le tabelle e le viste
- UPDATE
  - permette di aggiornare il valore di un oggetto
  - vale per le tabelle, le viste e gli attributi
- DELETE
  - permette di rimuovere oggetti dalla risorsa
  - vale per le tabelle e le viste
- SELECT
  - permette di utilizzare la risorsa all'interno di un'interrogazione
  - vale per le tabelle e le viste
- REFERENCES
  - permette di far riferimento a una risorsa nella definizione dello schema di una tabella
  - può essere associato solo a tabelle e attributi
- USAGE
  - permette di utilizzare la risorsa (per esempio, un nuovo tipo di dato) nella definizione di nuovi schemi

Privilegi del creatore della risorsa

Creatore della risorsa

- Alla creazione di una risorsa, il sistema concede tutti i privilegi su tale risorsa all'utente che ha creato la risorsa
- Solo il creatore della risorsa ha il privilegio di eliminare una risorsa (DROP) e modificarne lo schema (ALTER)
  - il privilegio di eliminare e modificare una risorsa non può essere concesso a nessun altro utente

Amministratore del sistema

- L'amministratore del sistema (utente system) possiede tutti i privilegi su tutte le risorse

Gestione dei privilegi in SQL

- I privilegi sono concessi o revocati mediante le istruzioni SQL
  - GRANT
    - concede privilegi su una risorsa a uno o più utenti
  - REVOKE
    - toglie a uno o più utenti i privilegi che erano stati loro concessi

GRANT

GRANT *ElencoPrivilegi* ON *NomeRisorsa*
TO *ElencoUtenti*
[WITH GRANT OPTION]

- *ElencoPrivilegi*
  - specifica l'elenco dei privilegi
  - ALL PRIVILEGES
    - parola chiave per identificare tutti i privilegi
- *NomeRisorsa*
  - specifica la risorsa sulla quale si vuole concedere il privilegio
- *ElencoUtenti*
  - specifica gli utenti a cui viene concesso il privilegio
- WITH GRANT OPTION
  - facoltà di trasferire il privilegio ad altri utenti

Esempi

```
GRANT ALL PRIVILEGES
ON PRODOTTI TO Neri, Bianchi
```

- Agli utenti Neri e Bianchi sono concessi tutti i privilegi sulla tabella PRODOTTI

```
GRANT SELECT ON FORNITORI TO
Rossi
WITH GRANT OPTION
```

- All'utente Rossi è concesso il privilegio di SELECT sulla tabella FORNITORI
- L'utente Rossi ha facoltà di trasferire il privilegio ad altri utenti

REVOKE

REVOKE *ElencoPrivilegi* ON *NomeRisorsa* FROM *ElencoUtenti*
[RESTRICT|CASCADE]

- Può togliere
  - tutti i privilegi che erano stati concessi
  - un sottoinsieme dei privilegi concessi *NomeRisorsa*
- RESTRICT
  - il comando non deve essere eseguito qualora la revoca dei privilegi all'utente comporti qualche altra revoca di privilegi
    - Esempio: l'utente ha ricevuto i privilegi con GRANT OPTION e ha propagato i privilegi ad altri utenti
  - valore di default
- CASCADE
  - revoca anche tutti i privilegi che erano stati propagati
    - genera una reazione a catena
  - per ogni privilegio revocato sono
    - revocati in cascata tutti i privilegi concessi
    - rimossi tutti gli elementi della base di dati che erano stati creati sfruttando questi privilegi

Esempi

REVOKE UPDATE ON PRODOTTI
FROM Bianchi

- All'utente Bianchi è revocato il privilegio di UPDATE sulla tabella PRODOTTI
  - il comando non è eseguito se comporta la revoca del privilegio ad altri utenti

REVOKE SELECT ON FORNITORI
FROM Rossi CASCADE

- All'utente Rossi è revocato il privilegio di SELECT sulla tabella FORNITORI
- L'utente Rossi aveva ricevuto il privilegio con GRANT OPTION
  - se Rossi ha propagato il privilegio ad altri utenti, il privilegio è revocato in cascata
  - se Rossi ha creato una vista utilizzando il privilegio di SELECT, la vista è rimossa

Concetto di ruolo

- Il ruolo è un profilo di accesso
  - definito dall'insieme di privilegi che lo caratterizzano
- Ogni utente ricopre un ruolo predefinito
  - gode dei privilegi associati al ruolo
- Vantaggi
  - controllo dell'accesso più flessibile
    - possibilità che un utente ricopra ruoli diversi in momenti diversi
  - semplificazione dell'attività di amministrazione
    - possibilità di definire un profilo di accesso in un momento diverso dalla sua attivazione
    - facilità nella definizione del profilo di nuovi utenti

CREATE ROLE

CREATE ROLE *NomeRuolo*

- Definizione dei privilegi di un ruolo e del ruolo di un utente
  - istruzione GRANT
- Un utente in momenti diversi può ricoprire ruoli diversi
  - associazione dinamica di un ruolo a un utente

SET ROLE *NomeRuolo*

Gestione degli indici

Costrutti avanzati

Organizzazione fisica dei dati

- All'interno di un DBMS relazionale, i dati sono rappresentati come collezioni di record memorizzati in uno o più file
  - l'organizzazione fisica dei dati all'interno di un file influenza il tempo di accesso alle informazioni
  - ogni organizzazione fisica dei dati rende alcune operazioni efficienti e altre onerose
- Non esiste un'organizzazione fisica dei dati che sia efficiente per qualunque tipo di lettura e scrittura dei dati

Indici

- Gli *indici* sono le strutture fisiche accessorie offerte dai DBMS relazionali per migliorare l'efficienza delle operazioni di accesso ai dati
  - sono realizzati mediante strutture fisiche di tipo diverso
    - alberi
    - hash table
- Le istruzioni per la gestione degli indici non fanno parte dello standard SQL

Definizione di indici in SQL

- Il linguaggio SQL offre le seguenti istruzioni per la definizione degli indici
  - creazione di un indice
    - CREATE INDEX
  - cancellazione di un indice
    - DROP INDEX
- Le istruzioni per le gestione degli indici non fanno parte dello standard SQL

CREATE INDEX

CREATE INDEX *NomeIndice*
ON *NomeTabella (ElencoAttributi)*

- L'ordine in cui compaiono gli attributi in *ElencoAttributi* *è importante*
- Le chiavi dell'indice sono ordinate
  - prima in base al primo attributo in *ElencoAttributi*
  - a pari valore del primo attributo sui valori del secondo attributo
  - e così via, in ordine, fino all'ultimo attributo

Usare il numero minimo di attributi, di solito uno

Esempio

- Creazione di un indice sulla combinazione di attributi Cognome e Nome della tabella DIPENDENTE

```
CREATE INDEX IndiceCognomeNome
ON DIPENDENTE(Cognome,Nome)
```

- L'indice è definito congiuntamente sui due attributi
- Le chiavi dell'indice sono ordinate
  - prima in base al valore dell'attributo Cognome
  - a pari valore dell'attributo Cognome, sul valore dell'attributo Nome

DROP INDEX

DROP INDEX *NomeIndice*

- Elimina l'indice con nome *NomeIndice*
- Il comando è utilizzato quando
  - l' indice non è più utilizzato
  - il miglioramento delle prestazioni non è sufficiente
    - ridotta riduzione del tempo di risposta per le interrogazioni
    - rallentamento degli aggiornamenti causato dal mantenimento dell'indice

Progettazione fisica

Costrutti avanzati

Progettazione fisica: dati di ingresso

- Schema logico della base di dati
- Caratteristiche del DBMS prescelto
  - opzioni disponibili a livello fisico
    - strutture fisiche di memorizzazione
    - indici
- Volume dei dati
  - cardinalità delle tabelle
  - cardinalità e distribuzione dei valori del dominio degli attributi
- Stima del carico applicativo
  - interrogazioni più importanti e loro frequenza
  - operazioni di aggiornamento più importanti e loro frequenza
  - requisiti sul tempo di risposta per interrogazioni/aggiornamenti importanti

Progettazione fisica: risultato

- Schema fisico della base di dati
  - organizzazione fisica delle tabelle
  - indici
- Parametri di memorizzazione e funzionamento
  - dimensioni iniziali dei file, possibilità di espansione, spazio iniziale libero, ...

Procedimento

- La progettazione fisica è svolta in modo empirico, con un approccio per tentativi
  - non esistono metodologie di riferimento
- Caratterizzazione del carico applicativo
  - per ogni interrogazione rilevante è necessario definire
    - relazioni a cui accede
    - attributi da visualizzare
    - attributi coinvolti in selezioni/join
    - grado di selettività delle condizioni di selezione
  - per ogni aggiornamento rilevante è necessario definire
    - tipo di aggiornamento (inserimento, cancellazione, modifica)
    - relazione ed eventuali attributi coinvolti
    - grado di selettività delle condizioni di selezione

Procedimento: scelte da operare

- strutturazione fisica dei file che contengono le tabelle
  - ordinati, non ordinati
- scelta degli attributi da indicizzare
  - pilotata dalla stima del carico applicativo e dal volume dei dati
- per ogni indice definizione del tipo
  - per esempio, hash oppure B-tree
- eventuali variazioni dello schema
  - partizionamenti orizzontali in memoria secondaria
  - denormalizzazione di tabelle
    - utilizzata nei data warehouse

Tuning

- Se il risultato non è soddisfacente
  - *Tuning*, aggiungendo e togliendo indici
- Procedimento guidato dalla disponibilità di strumenti che permettano di
  - verificare il piano di esecuzione adottato dal DBMS prescelto
    - il piano di esecuzione definisce la sequenza di attività svolte dal DBMS per eseguire un'interrogazione
      - metodi di accesso ai dati
      - metodi di join
  - valutare il costo di esecuzione di alternative diverse