Unità D3

Sicurezza e concorrenza nelle basi di dati

Sicurezza nelle basi di dati

- Una base di dati è sicura quando soddisfa i seguenti parametri:
  - regola l' accesso ai dati protetti;
  - evita la modifica o la manipolazione dei dati da parte di utenti non autorizzati;
  - è disponibile (nel momento in cui deve essere consultata è presente, consistente e coerente).

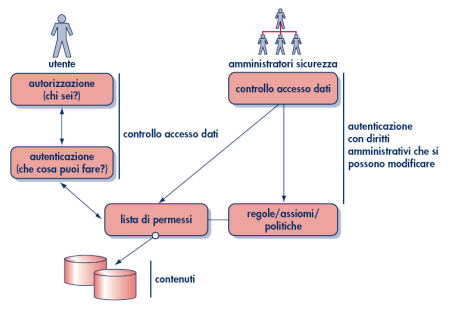
Controllo accesso

- **DAC** – controllo discrezionale
  - Il proprietario decide chi può accedere alle risorse
- **MAC** – a ogni risorsa viene associata una label (livello di privilegio che deve possedere l' utente per accedere)
- **RBAC** – gli utenti sono associati a uno o più ruoli (gruppi) le risorse sono accessibili solo a ruoli specificati

Protezione e integrità dati

- Autenticazione (login – password)
- Tracciabilità – registrazione delle operazioni effettuate da un utente (file di log)
- Integrità mediante "giornale" (file di log database) che registra i dati coinvolti e le operazioni effettuate su questi.
- Checkpoint e ripristino
- Backup

Autenticazione e autorizzazione



T-SQL (Esempio sicurezza)

```
CREATE DATABASE [ProvaSicurezza]
Use ProvaSicurezza
CREATE TABLE [dbo].[Studenti](
    [nome] [varchar](50) NOT NULL,
    [cognome] [varchar](50) NOT NULL
)

INSERT INTO Studenti VALUES ('Alberto', 'Paganuzzi')
INSERT INTO Studenti VALUES ('Alberto', 'Ferrari')
```

Nuovo utente

```
-- Creo un account di login sull'istanza DB Attuale
CREATE LOGIN Utente1 WITH password = 'PassUtente1'
USE ProvaSicurezza
SELECT * FROM dbo.Studenti -- funziona perchè eseguito da owner

-- Mi impersonifico in 'Utente1'
EXECUTE AS LOGIN = 'Utente1'
-- non è possibile: l'utente non è inserito in quelli che possono
-- accedere localmente al db ProvaSicurezza

-- Aggiungo l'utente a quelli autorizzati all'accesso di ProvaSicurezza
CREATE USER Utente1
EXECUTE AS LOGIN = 'Utente1' -- così funziona l'impersonificazione
SELECT * FROM dbo.Studenti -- non funziona l' utente non ha i diritti
```

Diritti all'utente

```
REVERT  -- il controllo torna all'utente precedente
-- autorizzo la select
GRANT SELECT ON ProvaSicurezza.dbo.Studenti TO Utente1
EXECUTE AS LOGIN = 'Utente1'
SELECT * FROM dbo.Studenti -- ora funziona utente autorizzato
REVERT
-- nego l'autorizzazione per la select
REVOKE SELECT ON ProvaSicurezza.dbo.Studenti TO Utente1
EXECUTE AS LOGIN = 'Utente1'
SELECT * FROM dbo.Studenti -- non funziona autorizzazione negata
PRINT USER  -- Visualizzazione utent
REVERT
PRINT USER
DROP DATABASE [ProvaSicurezza]
```

RBAC
(Role-based access control)

- Gli utenti sono associati a uno o più ruoli (gruppi) le risorse sono accessibili solo a ruoli specificati
- Esempio: associo Utente1 al ruolo che gli permette l'accesso in lettura al database
  EXEC sp_addrolemember db_datareader, Utente1
- Accesso in scrittura:
  EXEC sp_addrolemember db_datawriter, Utente1
- Negazione dell' accesso in scrittura
  EXEC sp_droprolemember db_datawriter, Utente1

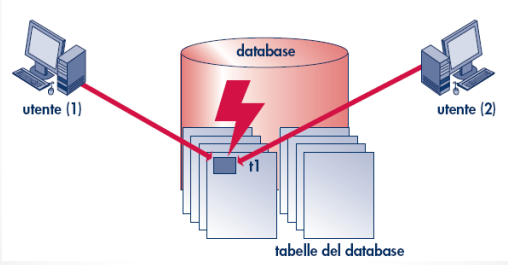
Consistenza di una Base di dati

- **Modello** : rappresentazione della realtà mediante un formalismo
- **Inconsistenza** : sfasamento fra realtà e modello che la rappresenta (database) (es. il DVD del film "…" è stato prestato ma risulta presente nel database)
- **Consistenza** : nessuna discrepanza tra la realtà fisica e il modello che la rappresenta.

Accesso concorrente

- **Accesso concorrente:** più utenti accedono a una stessa risorsa nello stesso istante.
- L' **accesso concorrente** è una delle cause principali dei problemi di inconsistenza delle basi di dati.
- Le soluzioni ai problemi di accesso concorrente sono basate su blocchi (***lock***) che operano come semafori e regolano il "traffico" verso le risorse.

Accesso concorrente



Transazioni

- Una **transazione** è un insieme di operazioni
  - indivisibili (atomiche)
  - corrette anche in presenza di concorrenza
  - con effetti definitivi.

Caratteristiche di una transazione ACID

- *atomicità* (***A****tomicity*). Una transazione viene portata a termine completamente o non viene effettuata;
- *consistenza* (***C****onsistency*). Prima e dopo la transazione la base di dati è sempre in uno stato consistente;
- *isolamento* (***I****solation*). Il database non viene modificato finchè la transazione non è conclusa (nessuno può vedere un risultato intermedio);
- *permanenza* (***D****urability*). Una volta conclusa la transazione i dati sono in uno stato consistente e non possono essere ripristinati allo stato precedente.

commit - rollback

- Transazione terminata con successo (COMMIT)
- Transazione abortita (ROLLBACK)

Anomalie derivanti da accessi concorrenti

- Perdita di aggiornamenti
  - T1 aggiorna X
  - T2 aggiorna X
  - T1 fallisce (rollback)
  - T2 termina (commit)
- Letture non riproducibili
  - T1 legge X
  - T2 aggiorna X
  - T2 commit
  - T1 rilegge X
- Letture fantasma
  - T1 aggiorna X
  - T2 legge X
  - T1 fallisce (rollback)

Soluzioni ai problemi

- Lock in lettura (compatibile con altri lock in lettura)
- Lock in scrittura (blocco esclusivo della risorsa)

Utilizzo delle transazioni

- Inizio transazione:
  BEGIN TRANSACTION <nome Transazione>
- Transazione terminata con successo:
  COMMIT TRANSACTION <nome Transazione>
- Transazione abortita:
  ROLLBACK TRANSACTION <nome Transazione>

Esempi transazioni in SQLServer

```
CREATE DATABASE [ProvaTransazioni]
Use ProvaTransazioni
CREATE TABLE [dbo].[Studenti](
    [nome] [varchar](50) NOT NULL,
    [cognome] [varchar](50) NOT NULL
)
```

Transazione senza errori

```
-- Primo esempio (in questo caso la transazione ha successo)
BEGIN TRY                                   -- costrutto try catch
  BEGIN TRANSACTION                         -- inizio transazione
  INSERT INTO Studenti VALUES ('Alberto', 'Paganuzzi')
  INSERT INTO Studenti VALUES ('Alberto', 'Ferrari')
  COMMIT TRANSACTION -- se non ci sono errori transazione completata
END TRY
BEGIN CATCH
  ROLLBACK TRANSACTION -- se ci sono errori transazione abortita
END CATCH
```

Transazione con errori

```
-- Secondo esempio di transazione
-- (la transazione NON ha successo perché non viene rispettato
-- il vincolo not null)
BEGIN TRY
  BEGIN TRANSACTION
  INSERT INTO Studenti VALUES ('Giuseppe', null)
  COMMIT TRANSACTION
END TRY
BEGIN CATCH
  ROLLBACK TRANSACTION
END CATCH
```

Transazione con errori (2)

```
-- Terzo esempio di transazione
-- (la transazione NON ha successo)
-- Non viene eseguita neppure al prima INSERT
-- (anche se questa è corretta)
BEGIN TRY
  BEGIN TRANSACTION
  INSERT INTO Studenti VALUES ('Paolo', 'Rossi')
  INSERT INTO Studenti VALUES ('Giuseppe', null)
  COMMIT TRANSACTION
END TRY
BEGIN CATCH
  ROLLBACK TRANSACTION
END CATCH
```

HOLDLOCK

```
/* Permette di bloccare (in questo caso la tabella) in modo SHARE
   non permette che altri eseguano UPDATE ma permette altre
     SELECT
*/
BEGIN TRANSACTION
SELECT *
  FROM Studenti WITH (HOLDLOCK)
  WHERE cognome = 'Paganuzzi'
-- visualizza lo stato di lock
SELECT resource_type,
    resource_associated_entity_id,request_mode,request_status
  FROM sys.dm_tran_locks dml INNER JOIN
       sys.dm_tran_current_transaction dmt
      ON dml.request_owner_id = dmt.transaction_id
COMMIT TRANSACTION
```

Prova HOLDLOCK

```
/* In parallelo alla precedente transazione PRIMA di effettuare il
     COMMIT
*/
-- per la SELECT  non ci sono problemi
SELECT * FROM Studenti

-- l' UPDATE rimane bloccata fino al commit della transazione
     precedente
UPDATE Studenti
  SET nome = 'Jack'
  WHERE cognome = 'Ferrari'
```

Blocco esclusivo

```
-- UPDATE blocca la tabella in modo esclusivo
BEGIN TRANSACTION
UPDATE Studenti
  SET nome = 'Alberto'
  WHERE cognome = 'Ferrari'
-- visualizza lo stato di lock
SELECT resource_type,
     resource_associated_entity_id,request_mode,request_status
  FROM sys.dm_tran_locks dml INNER JOIN
       sys.dm_tran_current_transaction dmt
      ON dml.request_owner_id = dmt.transaction_id
COMMIT TRANSACTION
```

Prova blocco esclusivo

- /* in parallelo testare le operazioni */
- SELECT * FROM Studenti-- attende la conclusione della transazione

- -- Possibile stabilire il tempo di attesa prima di abortire una transazione
- SET LOCK_TIMEOUT 3000 -- aspetta 3 secondi
- SELECT * FROM Studenti