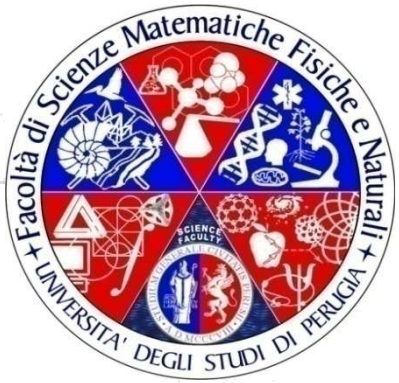
Università degli Studi di Perugia
Facoltà di Scienze MM.FF.NN
Corso di Informatica



Corso di Sicurezza Informatica
Prof. Stefano Bistarelli

Database Security

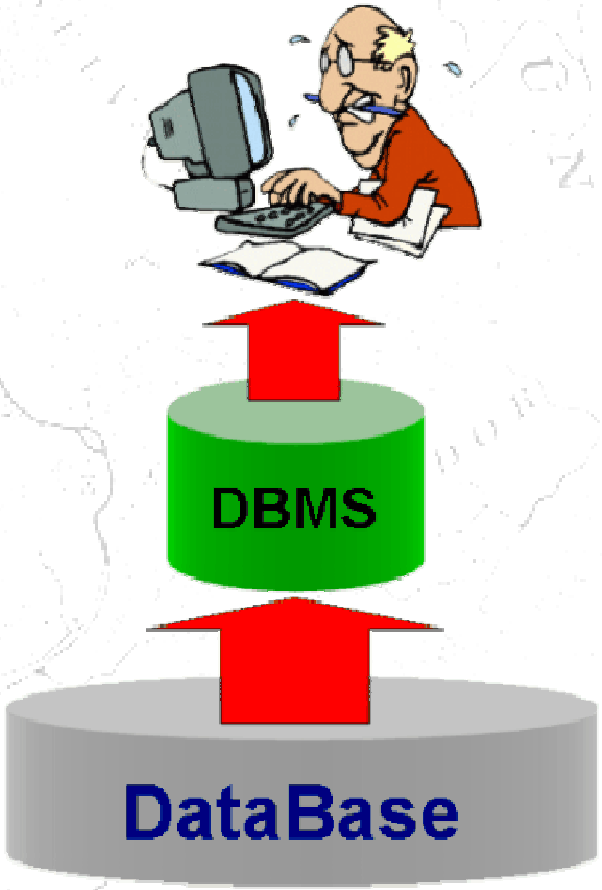
Seminario di Matteo Picciolini

Definizioni

Un Database è una collezione di dati legati secondo schemi logici

Un DBMS (database menagement system) è un applicazione software che consente la creazione e un'efficiente manipolazione di database

Le entrate di un database sono chiamate informazioni e sono gli obiettivi di un probabile attaccante



Definizioni 2

Le informazioni da proteggere sono classificabili in cinque categorie distinte:

•**Dati esatti** : il valore esatto salvato nella base di dati

•**Legami** : valore di limite superiore o inferiore su un campo del database di tipo numerico

•**Risultato negativo** : informazioni riguardo ai valori nulli di un database.

•**Esistenza**: la presenza o meno di un dato

•**Valore Probabile** : Valori che sono il risultato di query

Oltre a questa categoria di possibili minacce c'è anche la possibilità di proteggere il database dal creatore della base che se inesperto può causare involontariamente problemi.

Definizioni 3

Oggi i DBMS hanno al loro interno alcuni programmi che garantiscono il rispetto delle policy di sicurezza più elementari . In particolare :

Consistenza Interna : Permettendo di fare rispettare alle entrate del database certe regole precise (Per esempio le scorte di magazzino non devono essere sotto 0)

Consistenza Esterna: Controllando che le entrate di un database devo essere corrette. (Per esempio le giacenze di magazzino devono corrispondere al quantitativo ipotetico di merci solitamente prese).

Database Relazionali

640 K ought to be enough memory for anybody.

Bill Gates – Intervistato dalla BBC 1981

Formalmente una relazione (tabella) R è un sottoinsieme $D_1 x \ldots x D_n$ dove i vari D sono i domini di un attributo.

Gli elementi della relazione sono delle ennuple ($v_1..vn$) dove ogni vi appartiene a Di. Gli elementi di queste ennuple sono spesso chiamati campi.

Esiste un valore speciale chiamato "NULL". Questo valore non sta a significare che il valore non esiste ma piuttosto che quel valore è sconosciuto.

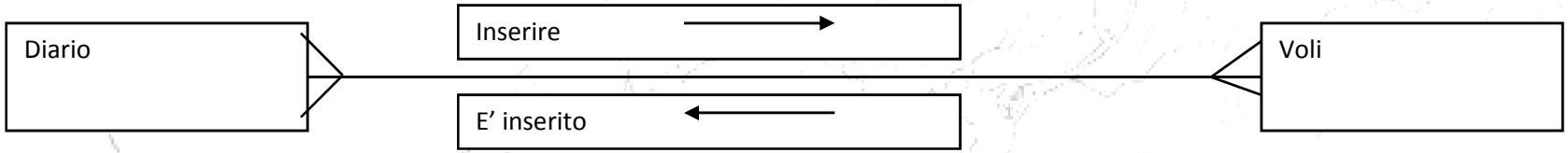
Relazione Diario

Nome	Giorno	Volo	Classe
Alice	Lunedi	GR123	Privata
Bob	Lunedi	YL011	Business
Bob	Mercoledi	BX201	*NULL*
Carol	Giovedi	BX201	Business
Alice	Giovedi	FL9700	Business

Di cui i domini sono:
Nome = Stringhe , tutti i nomi dei possibili dei clienti
Giorno = Stringhe, i giorni della settimana
Volo = Stringa, numero del volo costituito da due lettere e fino a quattro numeri
Status = Stringa, solo due valori permessi, privata e business

Relazioni Voli

Volo	Destinazione	Partenza	Gate
GR123	Tokio	7:55	1
YL011	Roma	8:10	22
BX201	Madrid	9:20	15
FL9700	Parigi	14:00	21
GR127	Londra	14:55	14



Modifica dello schema logico

Nome	**Giorno**	**Classe**
Alice	Lunedi	Privata
Bob	Lunedi	Business
Bob	Mercoledi	*NULL*
Carol	Giovedi	Business
Alice	Giovedi	Business

Campo Volo -> Ridondante

Volo	**Destinazione**	**Partenza**	**Gate**
GR123	Tokio	7:55	1
YL011	Roma	8:10	22
BX201	Madrid	9:20	15
FL9700	Parigi	14:00	21
GR127	Londra	14:55	14

ID	**Nome**	**Giorno**	**Volo**
1	Alice	Lunedi	GR123
2	Bob	Lunedi	YL011
3	Bob	Mercoledi	BX201
4	Carol	Giovedi	BX201
5	Alice	Giovedi	FL9700

La relazione molti a molti si trasforma in una tabella con le chiavi primarie di tutte le altre tabelle

Le chiavi

Come detto ogni tupla della tabella deve essere identificato in maniera univoca. Per questo uno o più campi di ogni tabella sono definiti chiave Primaria (Primary Key).

Dunque una chiava K primaria per la relazione R deve rispettare le seguenti condizioni:

Unica : allo stesso tempo nessuna tupla di R ha lo stesso valore per K

Minimale : se K è una chiave composta da più attributi, nessun componente di K può essere omesso senza distruggere la regola dell'unicità.

La chiave esterna (o foreign key) è un attributo della tabella collegata, ovvero nel caso della relazione molti a uno la chiave primaria del lato molti diventa un attributo della tabella del lato 1.

Regole d'integrità

Rinforzare la consistenza interna per cercare di mantenere la consistenza esterna.

Regola per l'integrità di una relazione : Nessun componente della chiave primaria di una relazione base può avere valori nulli.
Questa regola ci permette di trovare con certezza tutte le tuple di una relazione base.

Regola per l'Integrità referenziale : Il database non può contiene chiavi esterne che non corrispondono a nessuna chiave primaria.

Controllo dei campi. Utile per prevenire errori sui tipi di dato inseriti. Come ad esempio nel nostro caso il campo Classe che può essere sono business o privato.

Controllo dello Scope. In un database di tipo statistico pensare alcune regole per verificare l'esattezza di una query. Questo include ad esempio di far tornare dei valori di default per evitare di incorrere in dati inesatti.

Controllo della consistenza. Si tratta di uno speciale controllo sulle informazioni che debbono risultare coerenti. Per esempio se Alice viaggia il Lunedi su GR123 è consistente con il fatto che il volo GR123 parta il Lunedi e il Giovedi.

SQL

Select

Select [nome campo o lista dei campi]
from [tabelle di provenienza dei campi]
where [condizione]

Come esempio supponiamo di dover selezionare tutti i clienti che partono Lunedi e di far apparire come risultato della query anche la classe di appartenenza . Avremo quindi

Select Nome,Classe
from Diario
where Giorno = Lunedi

Ad esempio se volessimo vedere chi sono le persone che viaggiano verso Londra potremmo interrogare il database in due modi :

Senza Join

```
Select Nome
  From Diario
    Where Volo in (
        Select Volo
        From Voli
       Where Destinazione = Londra
)
```

Con Join

```
Select  Nome
From   Diario,Voli,Relazione
Where Voli.Volo=Relazione.Volo and Relazione.Nome=Diario.Nome and
Relazione.Nome=Diario.Nome and Destionazione=Londra
```

Oggetti del DB

Relazioni Base : Chiamate anche relazioni reali, o pure esistono dalla creazione della base di dati e sono salvate con i vari diritti di accesso

Viste : hanno un nome che può essere diverso dalla relazioni coinvolte nella creazione. Sono costruite a partire da altre relazioni, non vengono salvate in memoria, ma vengono coinvolte nel meccanismo di protezione di diritti.

Snapshot : sono come le viste ma possono essere salvati in memoria

Risultati di Query: Sono semplici risultati di query le quali possono essere rinominate oppure no e non esistono come oggetto a se stante, se non dopo essere state eseguite.

```
Create snapshot Viaggiatore
As select Nome,Giorno
From Diario
```

Controllo degli Accessi

Qualsiasi politica si voglia intraprendere si deve tener conto di due proprietà:

•**Completezza** : tutti i campi del database devono essere protetti

•**Consistenza** : non ci devono essere regole che risultino in conflitto per l'accesso agli oggetti del db.

Il controllo degli accessi è diviso in tre entità:

Utenti : Sono gli utenti che vorrebbero consultare il database. L'identità dell'utente è garantita dalle politiche di controllo accessi che solitamente coincidono con una sessione di login. Qui il dbms si incarica di effettuare il controllo oppure delega al sistema operativo il compito.

Azioni : che includono SELECT,UPDATE,DELETE,INSERT

Oggetti: tabelle, viste e attributi, SQL3 include anche strutture dati definite dall'utente.

Controllo degli Accessi

MySQL mette a disposizione quattro tipologie di privilegi che si differenziano a seconda del contesto di validita che assumono:

privilegi a livello globale: riguardano tutti i database del server e vengono memorizzati nella tabella **user**;

privilegi a livello di database: riguardano tutti gli oggetti di un database e sono memorizzati nella tabella **db**;

privilegi a livello di tabella: sono relativi ad una tabella di un database e vengono memorizzati nella tabella **tables_priv**;

privilegi a livello di colonna: riguardano una singola colonna di una tabella e sono memorizzati nella tabella **columns_priv**;

Privilegi

GRANT e **REVOKE** assegnano e revocano i privilegi

Se per esempio si volesse garantire la consultazione e l'aggiornamento, ad una parte della tabella DIARIO a due agenti di viaggio Art e Zoe dovremmo scrivere :

```
GRANT SELECT,UPDATE (Giorno,Volo)
ON TABLE Diario
TO Art,Zoe
```

Se invece volessimo revocare il permesso di aggiornamento dato a Art potremmo utilizzare questa istruzione:

```
REVOKE UPDATE
ON TABLE Diario
FROM Art
```

Privilegi 2

Interessante è l'opzione di permettere ad un altro utente di potere concedere permessi.

```
GRANT SELECT
ON TABLE Diario
TO Art
WITH GRANT OPTION
```

E quindi a sua volta Art potrà concedere I permessi a Zoe nella forma

```
GRANT SELECT
ON TABLE Diario
TO Zoe
WITH GRANT OPTION
```

Questa è una struttura gerarchica, in effetti revocare I permessi di Art significherà revocare a cascata tutti I permessi da lui concessi.

Le Viste

```
CREATE VIEW view_name
AS subquery
[WITH CHECK OPTION]
```

```
CREATE VIEW viaggi_d_affari AS
SELECT * FROM Diario
WHERE Stato = 'business'
WITH CHECK OPTION
```

L'opzione check option permette un meccanismo interessante di difesa. In effetti d'ora in avanti qualsiasi query sulla vista viaggi_d_affari che comporterebbe l'eliminazione di record non verrebbe eseguita.

Le Viste 2

Pro :

•Le viste sono flessibili e permettono policies di controllo degli accessi ad layer diversi.
•Le viste rinforzano le policies di sicurezza sulla consistenza e integrità di dati.
•Le viste sono istruzioni sql e pertanto possono essere controllate sin dalla loro invocazioni.
•I dati possono essere facilmente riclassificati.

Contro :

•I controlli sugli accessi possono risultare complicati e lenti.
•Le definizioni di viste devono essere controllate e risultare corrette.
• La completezza e la consistenza dei dati non sono automaticamente ottenute; le viste infatti possono fallire l'elaborazione della richiesta.
•La parte del DBMS incaricata del controllo delle viste (TCB) potrebbe occupare molta memoria.

Database Statistici

Le query di natura statistica sono anche definite aggregate, e si esprimono nel linguaggio SQL mediante:

COUNT: il numero dei valori in una colonna,
SUM: la somma dei valori di una colonna,
AVG: la media dei valori in una colonna,
MAX: il massimo dei valori di una colonna,
MIN: minimo dei valori di una colonna.

Il predicato di una query statistica specifica quali tuple sono coinvolte per il calcolo della funzione aggregante . I problemi che possono sorgere sono i seguenti:

✓ Il database contiene oggetti che sono sensibili singolarmente per cui l'accesso diretto a questi dati non deve essere permesso.

✓Le query statistiche sono permesse. Per loro natura devono leggere dati individuali.

Relazione Studenti

NOME	SESSO	PROGRAMMA	UN ITA'	MEDIA VOTI
Alma	F	MBA	8	63
Bill	M	CS	15	58
Carol	F	CS	16	70
Don	M	MIS	22	75
Eroll	M	CS	8	66
Gandalf	M	MIS	16	81
Frodo	M	MBA	23	68
Homer	M	CS	7	50
Alex	M	MIS	21	70

Questa tabella rappresenta la relazione studenti. Le query di tipo statistico sono tutte permesse, invece l'accesso diretto ai campi UNITA' e MEDIA VOTI non sono accessibili individualmente.

Query Aggregata

Un esempio di query statistica su questa relazione è la seguente, supponiamo di voler calcolare la media voti di tutti gli studenti iscritti al programma MBA.

```
SELECT AVG(Media Voti)
FROM studenti
WHERE Programma = 'MBA'
```

E' importante da subito sottolineare come il predicato della query sia

PROGRAMMA = 'MBA'.

Possibili Attacchi

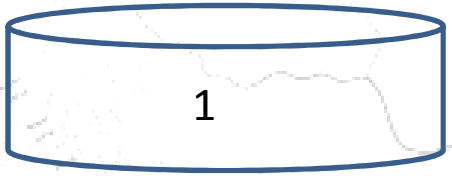
- **Attacco diretto:** l'aggregato è calcolato su un piccolo campione di dati trapelati durante l'attacco

- **Attacco indiretto**: questo tipo di attacco combina le informazioni relative al calcolo di altri aggregati

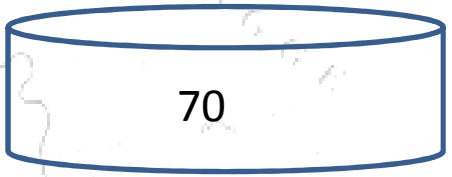
- **Tracker attack**: è un particolare ed efficiente attacco indiretto

- **Linear system vulnerability**: questo attacco utilizza le relazioni algebriche tra un set di query per costruire equazioni da utilizzare per ottenere le informazioni desiderate.

```
SELECT COUNT(*)
FROM Studenti
WHERE Sesso='F' AND Programma = 'CS'
```



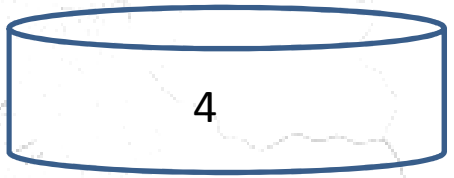

```
SELECT AVG (MEDIA VOTI)
FROM Studenti
WHERE Sesso='F' AND Programma = 'CS'
```



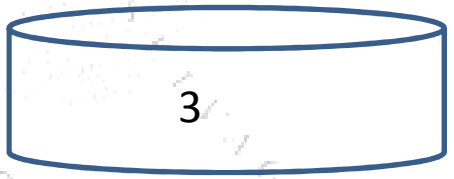
Attacco Scoperta Media Voti Di Carol

Relazione Studenti

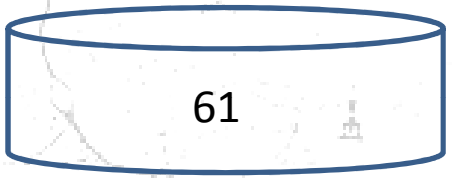
```
SELECT COUNT (*)
FROM Studenti
WHERE Programma = 'CS'
```



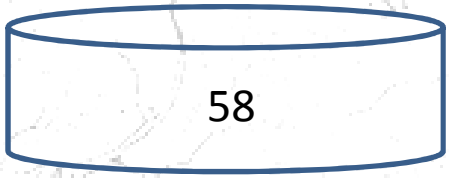

```
SELECT COUNT (*)
 FROM Studenti
WHERE Programma = 'CS' AND Sesso='M'
```




```
SELECT AVG(Media Voti)
 FROM Studenti
WHERE Programma = 'CS'
```




```
SELECT AVG (Media Voti)
FROM Studenti
WHERE Programma = 'CS' AND Sesso='M'
```



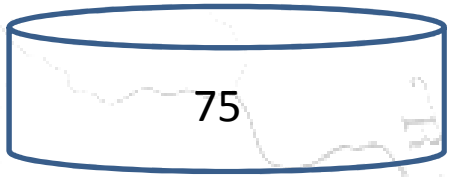
Attacco Scoperta Media Voti di Carol -> 4*61-3*58=70

Tracker Attack

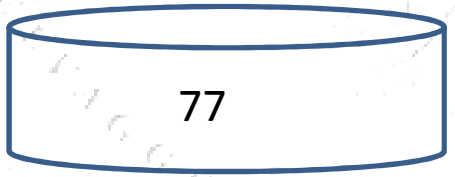
Un predicato della query T che permette di tracciare informazioni di una singola tupla è chiamato **“individual tracker”** per quella tupla. Un **“general tracker”** è un predicato che può essere usato per trovare la risposta ad una query che altrimenti sarebbe inammissibile.

Prendiamo T come general traker e consideriamo R come un predicato che identifica unicamente la tupla r che noi vogliamo trovare. Nel nostro esempio il predicato in questione è Nome = ‘Carol’. Noi dobbiamo fare due query sul database usando il predicato (R or T) e (R or not(T)). Il nostro target r è la sola tupla che soddisfa entrambe le query. Per essere sicuri che le due query risultano ammissibili, noi dobbiamo scegliere T in modo che, sia lei che il suo complemento, siano ammissibili. Inoltre dobbiamo concludere con una query che effettui una ricerca sull’intero database

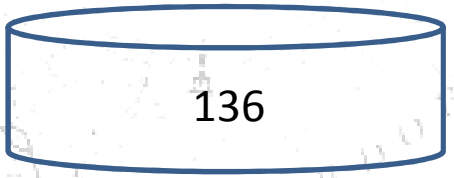
```
SELECT SUM(Unità)
FROM Studenti
WHERE Nome = 'Carol' OR Programma='MIS'
```




```
SELECT SUM(Unità)
FROM Studenti
WHERE Nome = 'Carol' OR NOT(Programma='MIS')
```




```
SELECT SUM(Unità)
FROM Studenti
```



Attacco Scoperta Unità di Carol -> (75+77)-136=16

Contromisure

togliere le informazione sensibili dal database

controllare il quantitativo di query statistiche che vengono richieste a distanza di tempo molto breve

mascherare i dati

aggiungere delle piccole perturbazioni sui risultati di query

Migliorare schema logico del database

Audit log

Sicurezza nei DBMS

Una situazione tipica da considerare è in cui due utenti leggono lo stesso dato con l'intenzione di aggiornarlo, evidentemente uno dei due lo farà per primo e di conseguenza il secondo utente quando lo vorrà aggiornarlo troverà una situazione diversa rispetto a quando aveva letto i dati, rischiando di incorrere in situazioni incongruenti. Le soluzioni offerte dal gestore sono due i lock sulle tabelle e la gestione delle transazioni.

Lock

Il lock può essere considerato come un vincolo di “uso esclusivo” e può essere richiesto in lettura o in scrittura.

Vediamo la sintassi delle operazioni di lock, ricordando che esse richiedono il privilegio **LOCK TABLES**:

lock tables nome_tabella1 READ|WRITE,
nome_tabella2 READ|WRITE;

Per sbloccare i lock su tutte le tabelle, si usa il comando seguente:

unlock tables;

Transazioni

L'uso delle transazioni permette di "consolidare" le modifiche alla base dati solo in un momento ben preciso: dal momento in cui avviamo una transazione, gli aggiornamenti rimangono sospesi (e invisibili ad altri utenti) fino a quando non li confermiamo (**commit**); in alternativa alla conferma è possibile annullarli (**rollback**).

Riprendendo la nostra analisi con il dbms MySql, esso gira automaticamente in modalità **autocommit mode**: questo significa che tutti gli aggiornamenti vengono automaticamente consolidati nel momento in cui sono eseguiti. Se siamo in autocommit, per iniziare una transazione dobbiamo usare l'istruzione **START TRANSACTION**; da questo punto in poi tutti gli aggiornamenti rimarranno sospesi. La transazione può essere chiusa con l'istruzione **COMMIT**, che consolida le modifiche, oppure con **ROLLBACK**, che annulla tutti gli aggiornamenti effettuati nel corso della transazione. Possiamo utilizzare anche **COMMIT AND CHAIN** o **ROLLBACK AND CHAIN**, che provocano l'immediata apertura di una nuova transazione, oppure **COMMIT RELEASE** o **ROLLBACK RELEASE**, che oltre a chiudere la transazione chiudono anche la connessione al server.

Transazioni

```
START TRANSACTION
...istruzioni di aggiornamento (1)...
SAVEPOINT sp1;
...istruzioni di aggiornamento (2)...
ROLLBACK TO SAVEPOINT sp1;
...istruzioni di aggiornamento (3)...
COMMIT
```

Anomalie nelle Transazioni

Write-Write - Lost Update : Perdite di Aggiornamento

1. T1 effettua un UPDATE del dato Y.
2. T2 effettua nuovamente un UPDATE dello stesso dato Y.
3. La transazione T1 viene abortita a causa di un errore e va in ROLLBACK portando il database allo stato di origine.
4. La transazione T2 va in COMMIT e termina con successo.

Questo tipo di anomalia causa la perdita dell'aggiornamento della transazione T2.

Read-Write - Unrepeatable Read : Letture non riproducibili

1. T1 effettua una SELECT sul dato Y.
2. T2 effettua un UPDATE dello stesso dato Y e va in COMMIT.
3. La transazione T1 rieffettua una SELECT nello stesso dato Y senza effettuare una modifica.

In questo caso T1 effettua due SELECT che restituiscono valori diversi di Y perchè la transazione T2, essendo concorrente, ne ha modificato il risultato durante l'esecuzione.

Anomalie nelle Transazioni 2

Write-Read - Dirty Read : Letture Fantasma

1. T1 effettua un UPDATE sul dato Y.
2. T2 effettua una SELECT dello stesso dato Y.
3. La transazione T1 abortisce andando in ROLLBACK.

In quest'ultimo caso,il dato letto dalla transazione T2 non sarà valido perchè non è persistente nel database.

Soluzione del Problema

Il DBMS nel rispettare questo concetto adotta un sistema di lock a livello di record (e non di tabella come nei sistemi operativi). I blocchi adottati sono:

Shared Locks - Blocchi Condivisi
Vengono richiesti in caso di lettura di un record e la loro acquisizione può essere concorrente ad altre acquisizioni dello stesso tipo

Exclusive Lock - Blocchi esclusivi
Vengono richiesti in caso di scrittura di un record e sono eseguiti in isolamento

Protocollo Strict 2PL

Lo Strict 2PL permette solo esecuzioni seriali delle transazioni, snellisce l'esecuzione di aborto delle transazioni e risolve i problemi relativi alle perdite di aggiornamento (Lost Update) e alle letture non riproducibili (Unrepeatable Read).

1. Ogni transazione deve ottenere uno Shared Lock su un oggetto prima di effettuare una lettura

2. Ogni transazione deve ottenere un Exclusive Lock su un oggetto prima di effettuare una scrittura

3. Se una transazione ha acquisito un Exclusive Lock su un oggetto nessuna altra transazione può acquisire Lock di qualsiasi tipo su quell'oggetto.

4. Ogni Lock acquisito da una transazione deve essere rilasciato al suo termine

Livelli di Isolamento

Risolve il problema del Dirty Read, adotta un'altra tipologia di Lock che nell'arco della durata di una transazione non permette la modifica del numero degli elementi che soddisfano le condizioni. I Lock vengono adottati sui predicati utilizzati nelle clausole WHERE e in concreto vengono acquisiti per via della presenza dei **livelli di isolamento** che il DBMS mette a disposizione.

I livelli di isolamento previsti sono i seguenti:

•**READ UNCOMMITED**

Con questo livello non si possono verificare solo Lost Update. Consente transazioni in sola lettura, senza bloccare la lettura dei dati.

•**READ COMMITTED**

Con questo livello di isolamento si possono verificare Unrepeatable Read ma non Dirty Read. Con il suo utilizzo è previsto il rilascio immediato dei dati in lettura e il rilascio ritardato dei dati in scrittura.

•**REPEATABLE READ**

Con questo livello si possono verificare solo Dirty Read. In pratica rappresenta lo Strict 2PL. Utilizzandolo vengono bloccati sia i dati in lettura che in scrittura dei record coinvolti.

•**SERIALIZABLE**

Grazie a questo livello si raggiunge un isolamento perfetto evitando le anomalie viste.

Livelli di Isolamento 2

Il livello **SERIALIZABLE non risolve tutti i problemi**: la sua dichiarazione richiede :

un numero elevato di blocchi e aumenta la competizione tra le transazioni per l'acquisizione di blocchi.

Questo peggiora le prestazioni e aumenta drasticamente il rischio di stallo.

La migliore soluzione sta nell'utilizzare un pò tutti i livelli di isolamento in base alle esigenze del dato o dell'applicazione

Domande a cui so rispondere?