

BASI DI DATI

TECNOLOGIA DEI SERVER PER BASI DI DATI

Prof. Fabio A. Schreiber
Dipartimento di Elettronica e Informazione
Politecnico di Milano

tratto da: Atzeni, Ceri, Paraboschi, Torlone - Basi di Dati -
McGraw-Hill



COMPONENTI DI UN SERVER PER DB

- **ANALIZZATORE DELLE INTERROGAZIONI**
- **OTTIMIZZATORE**
 - SCEGLIE LE STRATEGIE DI ACCESSO
- **GESTORE DEI METODI DI ACCESSO**
 - GESTISCE LE STRATEGIE
 - RSS (RELATIONAL STORAGE SYSTEM)
 - OM (OBJECT MANAGER)
- **GESTORE DEI BUFFER**
 - GESTISCE GLI SCAMBI TRA DISCO E MEMORIA CENTRALE

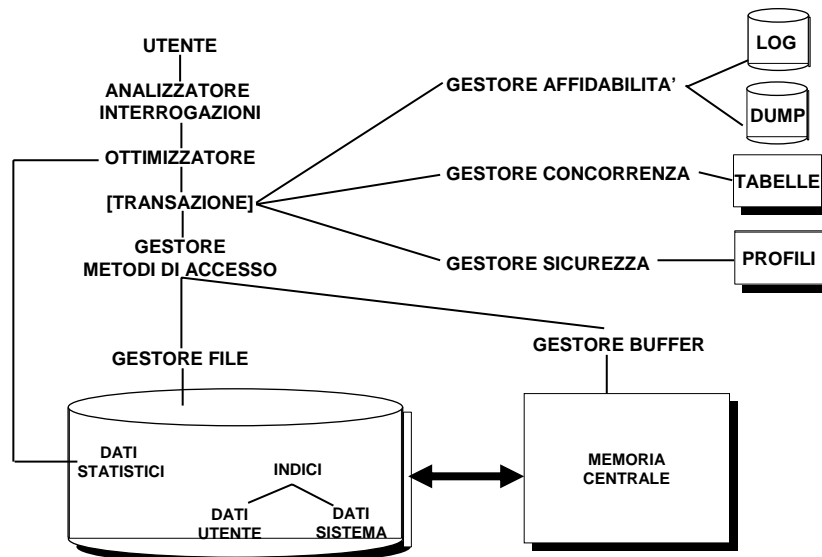
COMPONENTI DI UN SERVER PER DB

- **GESTORE DELL'AFFIDABILITA'**
 - GESTISCE I PROBLEMI LEGATI AI GUASTI E ALLA RIPARTENZA DEL SISTEMA
- **GESTORE DEL CONTROLLO DELLA CONCORRENZA**
 - GESTISCE LE INTERFERENZE DOVUTE ALLA MULTIUTENZA
- **GESTORE DELLA SICUREZZA**
 - GESTISCE GLI ACCESSI IN MODO CHE VENGANO RISPETTATE LE AUTORIZZAZIONI

© Fabio A. Schreiber

Tecnologia server 2

COMPONENTI DI UN SERVER PER DB



© Fabio A. Schreiber

Tecnologia server 3

DEFINIZIONE DI TRANSAZIONE

UNITA' ELEMENTARE DI LAVORO EFFETTUATO DA UN'APPLICAZIONE, TALE DA POSSEDERE CARATTERISTICHE DI

- CORRETTEZZA**
 - ROBUSTEZZA**
 - ISOLAMENTO**
-
- OGNI TRANSAZIONE E' INCAPSULATA TRA DUE COMANDI**
 - BEGIN TRANSACTION (bot)**
 - END TRANSACTION (eot)**

© Fabio A. Schreiber

Tecnologia server 4

DEFINIZIONE DI TRANSAZIONE

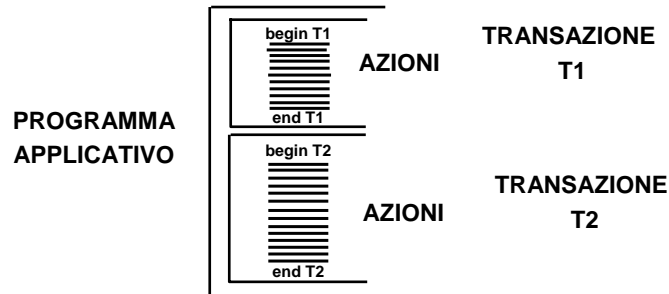
- ENTRO UNA TRANSAZIONE DEVE ESSERE ESEGUITO UNA E UNA SOLA VOLTA UNO DEI SEGUENTI COMANDI**
 - COMMIT WORK PER TERMINARE CORRETTAMENTE**
 - ROLLBACK WORK PER ABORTIRE LA TRANSAZIONE**

UN SISTEMA TRANSAZIONALE (OLTP) E' IN GRADO DI DEFINIRE ED ESEGUIRE TRANSAZIONI PER CONTO DI UN CERTO NUMERO DI APPLICAZIONI CONCORRENTI

© Fabio A. Schreiber

Tecnologia server 5

STRUTTURA DELLE TRANSAZIONI



UNA TRANSAZIONE CHE COMINCIA CON `begin transaction` E TERMINA CON `end transaction`, NELLA QUALE VIENE ESEGUITO UNO SOLO DEI DUE COMANDI `commit work` O `rollback work` DOPO IL QUALE NON VIENE ESEGUITA ALCUN'ALTRA OPERAZIONE SUI DATI E' UNA TRANSAZIONE BEN FORMATA

© Fabio A. Schreiber

Tecnologia server 6

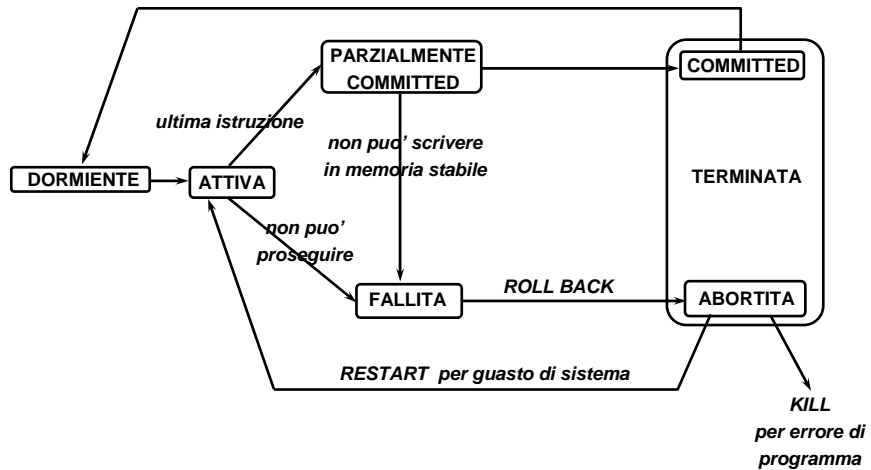
ESEMPIO DI TRANSAZIONE

```
begin transaction
x:=x-10
y:=y+10
commit work
end transaction
```

© Fabio A. Schreiber

Tecnologia server 7

STATI DI UNA TRANSAZIONE



© Fabio A. Schreiber

Tecnologia server 8

PROPRIETA' DELLE TRANSAZIONI E MECCANISMI

ATOMICITA'

- PROTOCOLLI DI COMMIT
- ABORT-ROLLBACK-RESTART

CONSISTENZA

- CONTROLLO DELLA CONCORRENZA
- VERIFICHE DI INTEGRITA' DEL DBMS

ISOLAMENTO

- CONTROLLO DELLA CONCORRENZA

DURABILITA' (PERSISTENZA)

- PROTOCOLLI DI COMMIT
- RECOVERY MANAGEMENT



© Fabio A. Schreiber

Tecnologia server 9

ATOMICITA'

UNA TRANSAZIONE E' UN'UNITA' DI LAVORO ATOMICA

- **NON PUO' LASCIARE LA BASE DI DATI IN UNO STATO INTERMEDIO (INCONSISTENTE)**
 - **UN ERRORE O UN GUASTO CHE INTERVENGA PRIMA DEL COMMIT PROVOCA L'ANNULLAMENTO (UNDO) DEL LAVORO FATTO FINO A QUEL PUNTO**
 - **UN ERRORE O UN GUASTO CHE INTERVENGA DOPO IL COMMIT PUO' RICHIEDERE IL RIFACIMENTO (REDO) DI QUANTO GIA' SVOLTO QUALORA NON SIA GARANTITA LA REGISTRAZIONE SULLA BASE DI DATI**
- **COMPORAMENTI POSSIBILI**
 - **COMMIT = CASO NORMALE (99.9%)**
 - **ROLLBACK RICHIESTO DALL'APPLICAZIONE = SUICIDIO**
 - **ROLLBACK RICHIESTO DAL SISTEMA = OMICIDIO**

© Fabio A. Schreiber

Tecnologia server 10

CONSISTENZA

NON DEVE ESSERE VIOLATO ALCUN VINCOLO DI INTEGRITA'

- **LA VERIFICA DEI VINCOLI DI INTEGRITA' PUO' ESSERE**
 - **IMMEDIATA - L'OPERAZIONE CHE CAUSA LA VIOLAZIONE VIENE RIGETTATA**
 - **DIFFERITA - SE VIENE VIOLATO QUALCHE VINCOLO TUTTA LA TRANSAZIONE VIENE RIGETTATA**

© Fabio A. Schreiber

Tecnologia server 11

ISOLAMENTO

**OGNI TRANSAZIONE VIENE ESEGUITA
INDIPENDENTEMENTE DALL'ESECUZIONE DI
QUALSIASI ALTRA TRANSAZIONE CONCORRENTE**

- **L'ISOLAMENTO RICHIEDE CHE L'ESECUZIONE
CONCORRENTE DI UN INSIEME DI TRANSAZIONI
TERMINATE CORRETTAMENTE (COMMITTED)
FORNISCA LO STESSO RISULTATO DI
UN'ARBITRARIA ESECUZIONE SEQUENZIALE DELLE
STESSE TRANSAZIONI**

PERSISTENZA

**L'EFFETTO DI UNA TRANSAZIONE CHE SIA
TERMINATA CORRETTAMENTE DEVE PERMANERE
NEL SISTEMA A TEMPO INDEFINITO E NON DEVE
VENIR PERSO PER QUALSIASI MOTIVO**

CONTROLLO DELLA CONCORRENZA

- **L'ESECUZIONE CONCORRENTE DI TRANSAZIONI E' UNO DEGLI OBIETTIVI DEI DBMS**
 - IL *THROUGHPUT* PUO' VARIARE TRA QUALCHE DECINA E QUALCHE MIGLIAIO DI TRANSAZIONI AL SECONDO (*tps*)
 - ESEMPI TIPICI
 - SISTEMI BANCARI
 - PRENOTAZIONE E *CHECK-IN* DI VOLI
 -

© Fabio A. Schreiber

Tecnologia server 14

CONTROLLO DELLA CONCORRENZA

ARCHITETTURA

- IL GRANULO E' L'UNITA' DI DATI AD ACCESSO CONCORRENTE
- SI CONSIDERANO OPERAZIONI DI INPUT-OUTPUT SU OGGETTI ASTRATTI x, y, z, \dots
- LE OPERAZIONI DI INPUT-OUTPUT LEGGONO/SCRIVONO BLOCCHI DI DISCO IN PAGINE DI MEMORIA E VICEVERSA

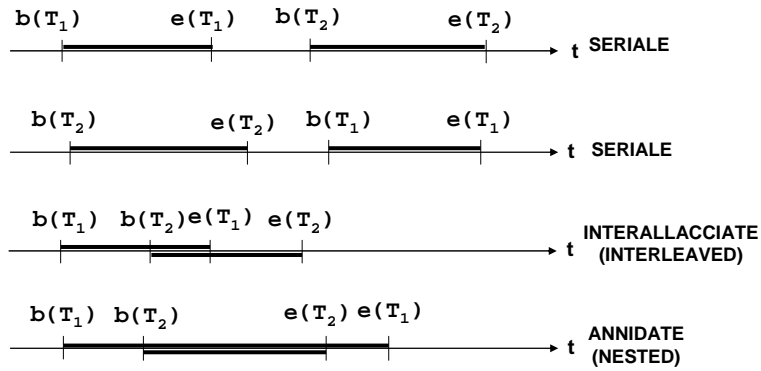
PROBLEMI

- LE ANOMALIE DOVUTE ALL'ESECUZIONE CONCORRENTE

© Fabio A. Schreiber

Tecnologia server 15

ESECUZIONI CONCORRENTI

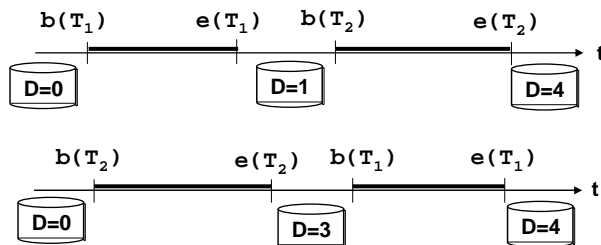


© Fabio A. Schreiber

Tecnologia server 16

ESECUZIONI CONCORRENTI CASI SERIALI

$bot(T_1)$	$bot(T_2)$	
read(D,x)	read(D,y)	
$x=x+1$	$y=y+3$	
write(D,x)	write(D,y)	$D(t_0)=0$
commit	commit	
$eot(T_1)$	$eot(T_2)$	

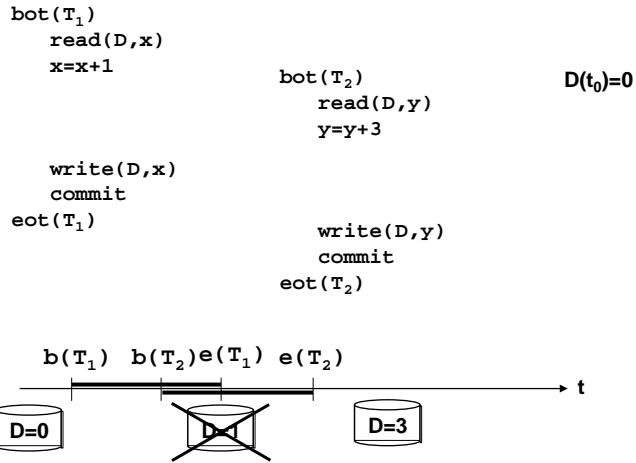


© Fabio A. Schreiber

Tecnologia server 17

ESECUZIONI CONCORRENTI

ANOMALIA 1 - PERDITA DI UN AGGIORNAMENTO

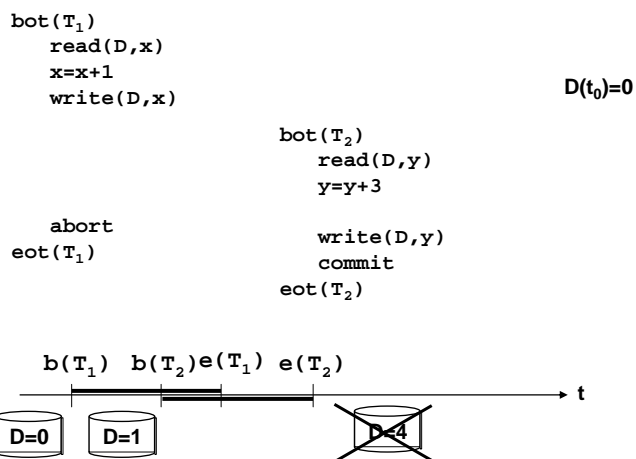


© Fabio A. Schreiber

Tecnologia server 18

ESECUZIONI CONCORRENTI

ANOMALIA 2 - LETTURA SPORCA



© Fabio A. Schreiber

Tecnologia server 19

ESECUZIONI CONCORRENTI

ANOMALIA 3 - LETTURA INCONSISTENTE

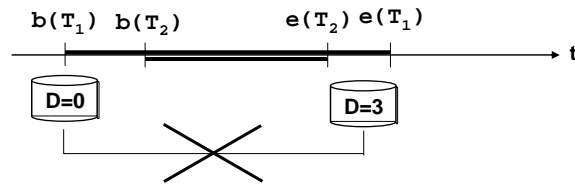
```

bot(T1)
  read(D,x)

bot(T2)
  read(D,y)
  y=y+3
  write(D,y)
  commit
eot(T2)

read(D,x)
commit
eot(T1)
    
```

$D(t_0)=0$



© Fabio A. Schreiber

Tecnologia server 20

ESECUZIONI CONCORRENTI

ANOMALIA 4 - AGGIORNAMENTO FANTASMA

```

bot(T1)
  read(X,x)

  read(Y,y)

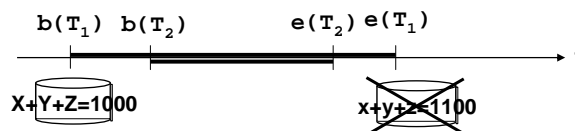
  read(Z,z)
  s=x+y+z
  commit
eot(T1)

bot(T2)
  read(Y,y)
  y=y-100
  read(Z,z)
  z=z+100
  write(Y,y)
  write(Z,z)
  commit
eot(T2)
    
```

INVARIANTE
 $X+Y+Z=1000$

$X(t_0)=300$
 $Y(t_0)=300$
 $Z(t_0)=400$

IL VINCOLO DI INTEGRITA' VIENE VIOLATO PERCHE' T₁ VEDE UN AGGIORNAMENTO FANTASMA



© Fabio A. Schreiber

Tecnologia server 21

CONCETTO DI SCHEDULAZIONE

RAPPRESENTA LA SEQUENZA DELLE OPERAZIONI DI INPUT/OUTPUT EFFETTUATE DALLE TRANSAZIONI CONCORRENTI

$$S_1: r_1(x) r_2(z) w_1(x) w_2(z)$$

$$r_1, w_1 \in T_1$$

$$r_2, w_2 \in T_2$$

© Fabio A. Schreiber

Tecnologia server 22

PRINCIPI FONDAMENTALI DEL CONTROLLO DELLA CONCORRENZA

- **OBIETTIVO**
 - RIFIUTARE LE SCHEDULAZIONI CHE PROVOCANO ANOMALIE
- **SCHEDULATORE**
 - COMPONENTE CHE ACCETTA O RIFIUTA LE OPERAZIONI RICHIESTE DALLE TRANSAZIONI
- **SCHEDULAZIONE SERIALE**
 - LE AZIONI DI OGNI TRANSAZIONE APPAIONO IN SEQUENZE CONTIGUE
 - $S_2: r_0(x) r_0(y) w_0(x) r_1(y) r_1(x) w_1(y) r_2(x) r_2(y) r_2(z) w_2(z)$

© Fabio A. Schreiber

Tecnologia server 23

PRINCIPI FONDAMENTALI DEL CONTROLLO DELLA CONCORRENZA

- **SCHEDULAZIONE SERIALIZZABILE**
PRODUCE GLI STESSI RISULTATI DI UNA QUALCHE SCHEDULAZIONE SERIALE SULLE STESSE TRANSAZIONI
 - RICHIEDE LA NOZIONE DI EQUIVALENZA TRA LE SCHEDULAZIONI
 - EQUIVALENZA DI VIEW
 - EQUIVALENZA DI CONFLITTI
 - LOCKING A DUE FASI (2PL)
 - BASATA SULLE MARCHE DI TEMPO (TIME STAMPING)
- **OSSERVAZIONE**
UNO SCHEDULATORE PERMETTE DI INDIVIDUARE CLASSI PIU' O MENO AMPIE DI SCHEDULAZIONI ACCETTABILI IN FUNZIONE DEL COSTO DELLA VERIFICA DELL'EQUIVALENZA

© Fabio A. Schreiber

Tecnologia server 24

PRINCIPI FONDAMENTALI DEL CONTROLLO DELLA CONCORRENZA

- **VIEW-SERIALIZZABILITA' (VSR)**
 - SI BASA SUL CONFRONTO DELLE SEQUENZE DI LETTURA/SCRITTURA SULLO STESSO OGGETTO
 - IN GENERALE E' UN PROBLEMA NP-hard
- **CONFLITTO-SERIALIZZABILITA' (CSR)**
 - SI BASA SUL CONFRONTO DELLA SEQUENZA DELLE OPERAZIONI IN CONFLITTO
 - AGISCONO SULLO STESSO OGGETTO E ALMENO UNA DI ESSE E' UNA SCRITTURA
 - RICHIEDE LA VERIFICA DELL'ACICLICITA' DI UN GRAFO
 - NONOSTANTE LA COMPLESSITA' DEL PROBLEMA SIA LINEARE E' ANCORA INACCETTABILE SOPRATTUTTO IN CONTESTI DISTRIBUITI

© Fabio A. Schreiber

Tecnologia server 25

LOCKING

E' IL METODO PIU' USATO NEI SISTEMI COMMERCIALI

- UNA TRANSAZIONE E' BEN FORMATA RISPETTO AL LOCKING SE
 - LE OPERAZIONI DI LETTURA SONO PRECEDUTE DA `r_lock` (LOCK CONDIVISO) E SEGUITE DA `unlock`
 - LE OPERAZIONI DI SCRITTURA SONO PRECEDUTE DA `w_lock` (LOCK ESCLUSIVO) E SEGUITE DA `unlock`
- QUANDO UNA TRANSAZIONE PRIMA LEGGE E POI SCRIVE UN OGGETTO PUO'
 - USARE UN `w_lock`
 - MODIFICARE UN `r_lock` IN UN `w_lock` (INNALZAMENTO DEL LOCK)

© Fabio A. Schreiber

Tecnologia server 26

COMPORTAMENTO DEL LOCK MANAGER

- IL LOCK MANAGER RICEVE LE PRIMITIVE DALLE TRANSAZIONI E CONCEDE LE RISORSE SECONDO LA TABELLA DEI CONFLITTI
 - QUANDO VIENE CONCESSO IL `LOCK` LA RISORSA VIENE ACQUISITA
 - QUANDO VIENE ESEGUITO `UNLOCK` LA RISORSA VIENE RILASCIATA

RICHIESTA	STATO DELLA RISORSA		
	LIBERA	R_LOCKED	W_LOCKED
<code>r_lock</code>	OK R_LOCKED	OK R_LOCKED	NO W_LOCKED
<code>w_lock</code>	OK W_LOCKED	NO R_LOCKED	NO W_LOCKED
<code>unlock</code>	ERRORE	OK DIPENDE	OK LIBERA

© Fabio A. Schreiber

Tecnologia server 27

LOCKING A DUE FASI (2PL)

- **PER LA SERIALIZZABILITA' E' NECESSARIO CHE IL LOCKING SIA A DUE FASI**
 - UNA TRANSAZIONE NON PUO' ACQUISIRE ALTRI LOCK DOPO AVERNE RILASCIATO UNO
 - C'E' UNA FASE DI CRESCITA E UNA DI RESTRINGIMENTO
 - SE UNO SCHEDULATORE
 - USA TRANSAZIONI BEN FORMATE
 - CONCEDE I LOCK IN BASE AI CONFLITTI
 - E' A DUE FASI
- ALLORA PRODUCE LA CLASSE DELLE SCHEDULAZIONI 2PL

© Fabio A. Schreiber

Tecnologia server 28

LOCKING A DUE FASI (2PL)

- **LE SCHEDULAZIONI 2PL SONO SERIALIZZABILI E STRETTAMENTE INCLUSE IN CSR**
- **ESEMPIO DI SCHEDULAZIONE CSR, MA NON 2PL**
$$S_{12}: r_1(x) w_1(x) r_2(x) w_2(x) r_3(y) w_1(y)$$
- **QUANTO DETTO FINORA VALE NELL'IPOTESI CHE NESSUNA TRANSAZIONE ABORTISCA**
- **TALE IPOTESI NON SI VERIFICA IN PRATICA**
- **OCCORRE AGGIUNGERE UN VINCOLO**

© Fabio A. Schreiber

Tecnologia server 29

LOCKING A DUE FASI STRETTO

- I LOCK SU UNA TRANSAZIONE POSSONO ESSERE RILASCIATI SOLO SUCCESSIVAMENTE ALLE OPERAZIONI DI COMMIT/ABORT
- QUESTA VERSIONE DEL 2PL E' USATA NEI DBMS COMMERCIALI E ELIMINA L'ANOMALIA DELLE LETTURE SPORCHE

DEADLOCK

SI VERIFICA PER IL FATTO CHE TRANSAZIONI CONCORRENTI TRATTENGONO E A LORO VOLTA RICHIEDONO RISORSE OCCUPATE DALLE ALTRE

– $T_1: r_1(x) w_1(y)$

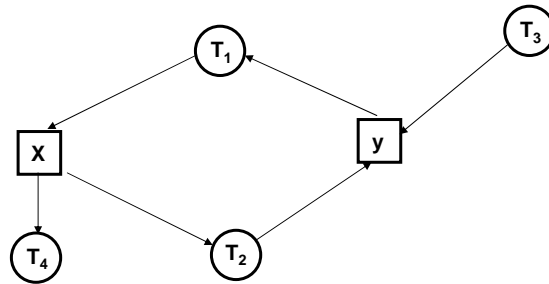
– $T_2: r_2(y) w_2(x)$

☞ $S: r_lock_1(x) r_lock_2(y) r_1(x) r_2(y) w_lock_1(y) w_lock_2(x)$

LA PROBABILITA' DI UN DEADLOCK CRESCE LINEARMENTE CON IL NUMERO DELLE TRANSAZIONI E CON IL QUADRATO DEL NUMERO DI LOCK DI CIASCUNA TRANSAZIONE

DEADLOCK

- UN DEADLOCK E' RAPPRESENTATO DA UN CICLO NEL GRAFO DI ATTESA (*WAIT-FOR GRAPH*) PER LE RISORSE



© Fabio A. Schreiber

Tecnologia server 32

TECNICHE DI RISOLUZIONE DEI DEADLOCK

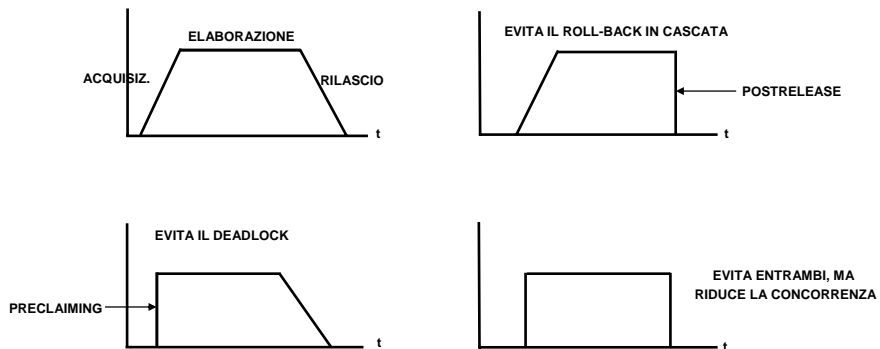
- **TIMEOUT**
 - IL PROBLEMA E' LA SCELTA DEL VALORE ADATTO
- **INDIVIDUAZIONE DEL DEADLOCK**
 - CERCA I CICLI NEL GRAFO *WAIT-FOR*
- **PREVENZIONE DEI DEADLOCK**
 - UCCIDE LE TRANSAZIONI CHE POTREBBERO CAUSARE DEI CICLI
 - OPZIONI PER INDIVIDUARE LA TRANSAZIONE DA UCCIDERE
 - PREEMPTIVE
 - NON PREEMPTIVE

© Fabio A. Schreiber

Tecnologia server 33

LOCKING A DUE FASI

TECNICHE DI ACQUISIZIONE E RILASCIO DELLE RISORSE



© Fabio A. Schreiber

Tecnologia server 34

CONTROLLO DELLA CONCORRENZA CON MARCATURE TEMPORALI (TIME STAMP)

- IL **TIME STAMP (TS)** DEFINISCE UN ORDINAMENTO TOTALE DEGLI EVENTI NEL SISTEMA
- AD OGNI TRANSAZIONE VIENE ASSEGNATO UN **TS** CORRISPONDENTE AL SUO MOMENTO DI INIZIO
- UNA SCHEDULAZIONE VIENE ACCETTATA SOLAMENTE SE RIFLETTE L'ORDINAMENTO SERIALE DELLE TRANSAZIONI BASATO SUL **TS** DI CIASCUNA DI ESSE

© Fabio A. Schreiber

Tecnologia server 35

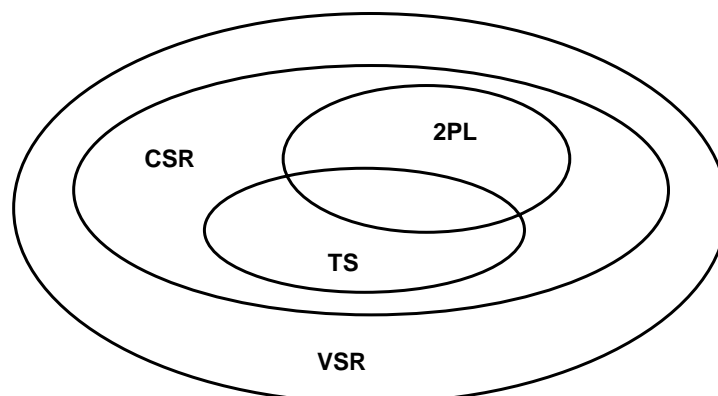
CONFRONTO TRA 2PL E TS

- CON 2PL LE TRANSAZIONI SONO POSTE IN ATTESA
- CON TS VENGONO UCCISE E FATTE RIPARTIRE
- L'ORDINE DI SERIALIZZAZIONE CON 2PL E' IMPOSTO DAI CONFLITTI
- CON TS E' IMPOSTO DALLA MARCATURA
- LA NECESSITA' DI ATTENDERE IL COMMIT DELLE TRANSAZIONI RICHIEDE IL 2PL STRETTO E LA MEMORIZZAZIONE TEMPORANEA DELLE OPERAZIONI DI SCRITTURA CON TS
- 2PL PUO' DAR LUOGO A DEADLOCK
- FAR RIPARTIRE UNA TRANSAZIONE COSTA PIU' CHE ASPETTARE: 2PL VINCE!

© Fabio A. Schreiber

Tecnologia server 36

PRINCIPI FONDAMENTALI DEL CONTROLLO DELLA CONCORRENZA



© Fabio A. Schreiber

Tecnologia server 37

GESTIONE DELLA MEMORIA

IL BUFFER E' UN'AREA DELLA MEMORIA CENTRALE ALLOCATA AL DBMS E CONDIVISA DALLE TRANSAZIONI

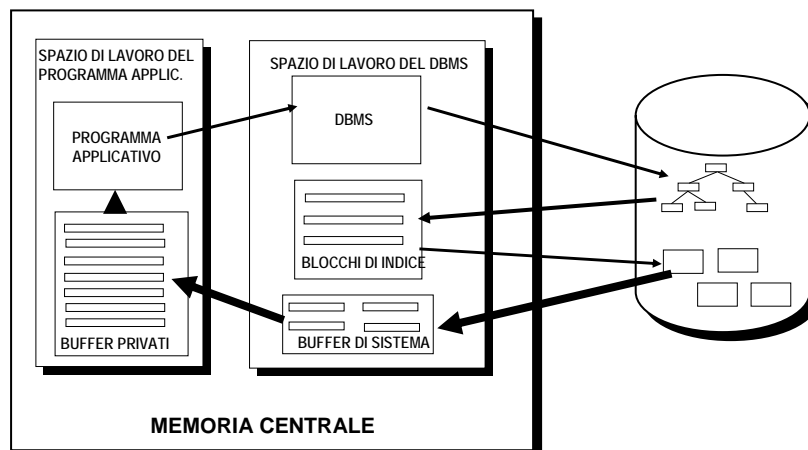
- E' ORGANIZZATA IN PAGINE DI DIMENSIONI UGUALI O MULTIPLE DI QUELLE DEL BLOCCO DI I/O USATO DAL SISTEMA OPERATIVO
 - 1 |-----| 100 KBYTE
 - ACCESSO 6 ORDINI DI GRANDEZZA PIU' VELOCE CHE VERSO IL DISCO

PUO' ASSUMERE DIMENSIONI MOLTO RILEVANTI

© Fabio A. Schreiber

Tecnologia server 38

INTERFACCIA CON I PROGRAMMI APPLICATIVI



© Fabio A. Schreiber

Tecnologia server 39

GESTORE DEL *BUFFER*

- **FORNISCE LE PRIMITIVE**
 - *fix, use, unfix, flush, force*
 - GESTISCE LE OPERAZIONI DI I/O IN RISPOSTA A QUESTE PRIMITIVE
- **LE POLITICHE DI GESTIONE DEL *BUFFER* SONO SIMILI A QUELLE CHE IL SISTEMA OPERATIVO USA PER GESTIRE LA MEMORIA CON LE CONDIZIONI**
 - PRINCIPIO DELLA LOCALITA' DEI DATI
 - I DATI CUI SI ACCEDE CORRENTEMENTE HANNO MAGGIOR PROBABILITA' DI ESSERE USATI ANCHE IN FUTURO
 - LEGGE EMPIRICA
 - L' 80% DELLE TRANSAZIONI ACCEDE SOLAMENTE AL 20% DEI DATI

© Fabio A. Schreiber

Tecnologia server 40

PRIMITIVE DEL GESTORE DEL *BUFFER*

- ***fix***
 - E' USATA PER CARICARE UNA PAGINA NEL *BUFFER*
 - RICHIEDE UNA LETTURA DA DISCO SOLO QUANDO LA PAGINA NON E' GIA' IN MEMORIA
 - LA PAGINA CARICATA E' VALIDA, CIOE' ALLOCATA AD UNA TRANSAZIONE
 - ALLA TRANSAZIONE VIENE FORNITO UN PUNTATORE ALLA PAGINA
- ***use***
 - E' USATA DALLA TRANSAZIONE PER ACCEDERE ALLA PAGINA CARICATA PRECEDENTEMENTE
 - CONFERMA DELLA SUA ALLOCAZIONE NEL *BUFFER*
 - CONFERMA DELLO STATO DI VALIDITA'

© Fabio A. Schreiber

Tecnologia server 41

PRIMITIVE DEL GESTORE DEL *BUFFER*

- **unfix**
 - INDICA CHE LA TRANSAZIONE HA TERMINATO L'USO DELLA PAGINA
 - LA PAGINA NON E' PIU' VALIDA
- **force**
 - TRASFERISCE IN MODO SINCRONO UNA PAGINA DAL *BUFFER* AL DISCO
- **flush**
 - TRASFERISCE IN MODO ASINCRONO LE PAGINE NON PIU' VALIDE INDIPENDENTEMENTE DALLE TRANSAZIONI ATTIVE

© Fabio A. Schreiber

Tecnologia server 42

POLITICHE DEL GESTORE DEL *BUFFER*

SOSTITUZIONE DELLE PAGINE (REPLACEMENT)

- QUANDO IL *BUFFER* E' PIENO, UN'OPERAZIONE DI **fix** COMPORTA LA SOSTITUZIONE DI UNA (O PIU') PAGINA
 - PREFERENZA ALLE PAGINE NON VALIDE (*NO-STEAL*)
 - SOSTITUZIONE DI PAGINE VALIDE (*STEAL*)
 - SPESSO CON POLITICA *MRU* (Most Recently Used)
- LA SCRITTURA SU DISCO DELLE PAGINE VALIDE DI UNA TRANSAZIONE PUO' ESSERE
 - FORZATA PRIMA DEL COMMIT (*FORCE*)
 - LASCIATA ALL'INIZIATIVA DEL GESTORE DEL *BUFFER* (*NO-FORCE*)
- NEI DBMS VIENE GENERALMENTE ATTUATA LA POLITICA *NO-STEAL/NO-FORCE*

© Fabio A. Schreiber

Tecnologia server 43

GESTORE DEL *BUFFER* E FILE SYSTEM

- **IL FILE SYSTEM CONOSCE LA STRUTTURA A DIRETTORI DELLA MEMORIA SECONDARIA E IL SUO STATO DI USO**
 - TIENE CONTO DEI BLOCCHI LIBERI E DI QUELLI ALLOCATI A FILE
- **IL DBMS USA IL FILE SYSTEM PER**
 - CREARE (*create*) ED ELIMINARE (*delete*) UN FILE
 - APRIRE (*open*) E CHIUDERE (*close*) UN FILE
 - LEGGERE CON ACCESSO DIRETTO UN BLOCCO IN UNA PAGINA DEL BUFFER *read* (*fileid*, *block*, *buffer*)
 - LEGGERE SEQUENZIALMENTE UN NUMERO DI BLOCCHI DEL FILE NEL BUFFER *read_seq* (*fileid*, *f-block*, *count*, *f-buffer*)
 - LE OPERAZIONI DUALI IN SCRITTURA *write* E *write_seq*

© Fabio A. Schreiber

Tecnologia server 44

STABILITA' DELLA MEMORIA

- **VOLATILE ($\lambda = 10^3$ ORE): MEMORIA CENTRALE E REGISTRI**
 - L'INFORMAZIONE VIENE DISTRUTTA DA QUALSIASI GUASTO DEL SISTEMA
- **NON VOLATILE ($\lambda =$ QUALCHE ANNO): DISCHI E NASTRI**
 - L'INFORMAZIONE SOPRAVVIVE AI GUASTI DEL SISTEMA, MA NON A QUELLI DELL'UNITA' DI MEMORIZZAZIONE
- **STABILE ($\lambda = 7 \cdot 10^2$ ANNI)**
 - L'INFORMAZIONE NON VIENE MAI DISTRUTTA
 - E' UN'ASTRAZIONE REALIZZATA FISICAMENTE CON COMPONENTI RIDONDANTI

© Fabio A. Schreiber

Tecnologia server 45

STATI DELLA BASE DI DATI

- **CORRETTO**
 - ULTIMA VERSIONE COMPLETA E AGGIORNATA
- **VALIDO**
 - PARTE DI UNO STATO CORRETTO (PARTE DELL'INFORMAZIONE E' ANDATA PERSA)
- **CONSISTENTE**
 - STATO VALIDO NEL QUALE SONO RISPETTATI I VINCOLI DI CONSISTENZA

© Fabio A. Schreiber

Tecnologia server 46

TIPI DI ERRORI O GUASTI

- **LOGICO**
 - IL PROGRAMMA CHE ESEGUE LA TRANSAZIONE NON PUO' PROSEGUIRE A CAUSA DI UN ERRORE
 - INPUT ERRATO
 - ECCEZIONE
 - DATO INESISTENTE
 - DIVISIONE PER ZERO
 - OVERFLOW

BISOGNA VERIFICARE IL PROGRAMMA
- **DI SISTEMA**
 - L' Istanza di esecuzione del programma non puo' proseguire per problemi di sistema
 - CONFLITTO DI TRANSAZIONI CONCORRENTI
 - DEADLOCK

BISOGNA RILANCIARE LA TRANSAZIONE

© Fabio A. Schreiber

Tecnologia server 47

TIPI DI ERRORI O GUASTI

- **CRASH DI SISTEMA**
 - GUASTO HW O SW A LIVELLO DEL SISTEMA DI ELABORAZIONE
 - GUASTO AD UNA SCHEDA
 - MANCANZA DI ALIMENTAZIONE
 - PROBLEMI DEL SISTEMA OPERATIVO
 - PUO' ANDAR PERSO IL CONTENUTO DEL *BUFFER*
- **GUASTO DELLA MEMORIA SECONDARIA**
 - PUO' INTERESSARE
 - QUALCHE BLOCCO
 - LA "ARATURA" DI GRAN PARTE DEL DISCO
 - VIENE PERSO IL CONTENUTO DELLA BASE DI DATI
- **CATASTROFE FISICA**
 - INCENDIO, ALLAGAMENTO, TERREMOTO, ...
 - SI SALVI CHI PUO'!

© Fabio A. Schreiber

Tecnologia server 48

STRUMENTI PER IL RECUPERO DAI GUASTI

- **DUMP**
 - SCARICAMENTO DEL CONTENUTO DELL'INTERA MEMORIA SU ALTRO SUPPORTO
- **LOG (GIORNALE)**
 - REGISTRAZIONE SEQUENZIALE SU ALTRO SUPPORTO DI TUTTE LE OPERAZIONI SVOLTE SULLA MEMORIA (A PARTIRE DALL'ULTIMO DUMP)
- **CHECKPOINT (MARCA)**
 - SPECIALE SEGNALAZIONE PERIODICA DELLO STATO DELLE TRANSAZIONI IN QUELL'ISTANTE MEMORIZZATA SUL LOG

© Fabio A. Schreiber

Tecnologia server 49

STRUTTURA DEL LOG

- **IL LOG E' UN FILE SEQUENZIALE CREATO E GESTITO DAL SOTTOSISTEMA DI AFFIDABILITA'**
- **IL FORMATO PIU' GENERALE DI UN RECORD DI LOG CONTIENE**
 - IDENTIFICATORE DELLA TRANSAZIONE `t_id`
 - IDENTIFICAZIONE DEL RECORD FISICO DEL DB `r_id`
 - IL VALORE PRECEDENTE LA MODIFICA (BEFORE IMAGE) `bi`
 - IL VALORE CONSEGUENTE ALLA MODIFICA (AFTER IMAGE) `ai`

© Fabio A. Schreiber

Tecnologia server 50

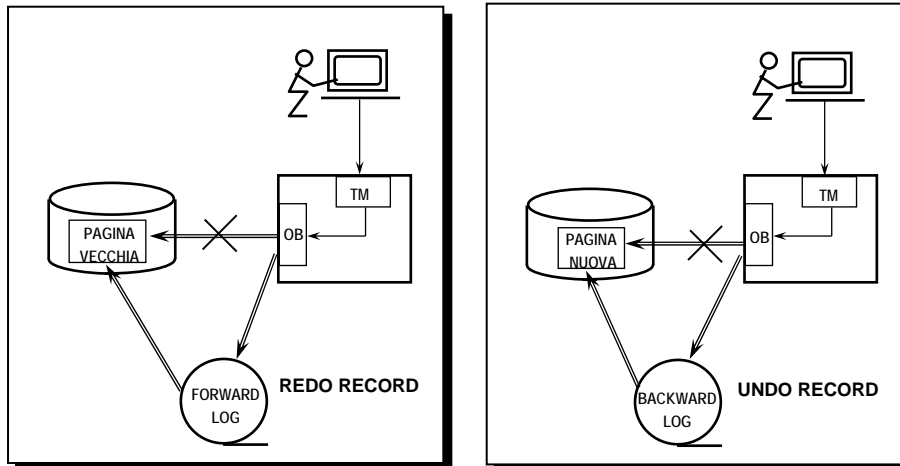
TIPI DI RECORD DI LOG

- **PER LE TRANSAZIONI**
 - `begin (t_id)`
 - `insert (t_id, r_id, ai)`
 - `delete (t_id, r_id, bi)`
 - `update (t_id, r_id, bi, ai)`
 - `commit (t_id)`
 - `abort (t_id)`
 - `read (t_id, r_id)` **IN GENERE NON E' NECESSARIO**
- **PER IL SISTEMA**
 - `dump`
 - `checkpoint (t_id1, t_id2, ..., t_idn)`

© Fabio A. Schreiber

Tecnologia server 51

TECNICHE DI LOGGING



© Fabio A. Schreiber

Tecnologia server 52

OPERAZIONI PER IL RIPRISTINO

- **UNDO**
 - PER update E delete
COPIA NEL RECORD r_{id} IL VALORE b_i
 - PER insert
CANCELLA IL RECORD r_{id}
- **REDO**
 - PER insert E update
COPIA NEL RECORD r_{id} IL VALORE a_i
 - PER delete
RISCRIVE IL RECORD r_{id}

UNDO E REDO SONO OPERAZIONI IDEMPOTENTI

- $\text{undo}(\text{undo}(\text{undo}(A))) = \text{undo}(A)$
- $\text{redo}(\text{redo}(\text{redo}(A))) = \text{redo}(A)$

© Fabio A. Schreiber

Tecnologia server 53

OPERAZIONI PER IL RIPRISTINO

- **CHECKPOINT**
 - **REGISTRA PERIODICAMENTE LE TRANSAZIONI ATTIVE**
 - TEMPO
 - NUMERO DI TRANSAZIONI COMMITTED
 - **AGGIORNA LA MEMORIA SECONDARIA PER QUEL CHE RIGUARDA LE TRANSAZIONI COMPLETATE**
 - DOPO L'INIZIO DI UN CHECKPOINT NON VENGONO PIU' ACCETTATE OPERAZIONI DI `commit` DA PARTE DELLE TRANSAZIONI ATTIVE
 - IL CHECKPOINT FINISCE CON UN'OPERAZIONE DI **force** DI UN RECORD `checkpoint(t_id1, t_id2, ..., t_idn)` CONTENENTE GLI IDENTIFICATORI DELLE TRANSAZIONI ATTIVE

© Fabio A. Schreiber

Tecnologia server 54

OPERAZIONI PER IL RIPRISTINO

- **DUMP**
 - **COPIA COMPLETA DELLA BASE DI DATI (BACKUP) CREATA QUANDO IL SISTEMA E' A RIPOSO**
 - **VIENE MEMORIZZATA SU SUPPORTI DIVERSI**
 - NASTRI
 - CD JUKE-BOX
 - **IL RECORD DI `dump` NEL LOG IDENTIFICA LA COPIA E IL SUPPORTO**

© Fabio A. Schreiber

Tecnologia server 55

PROCEDURE DI RIPRISTINO

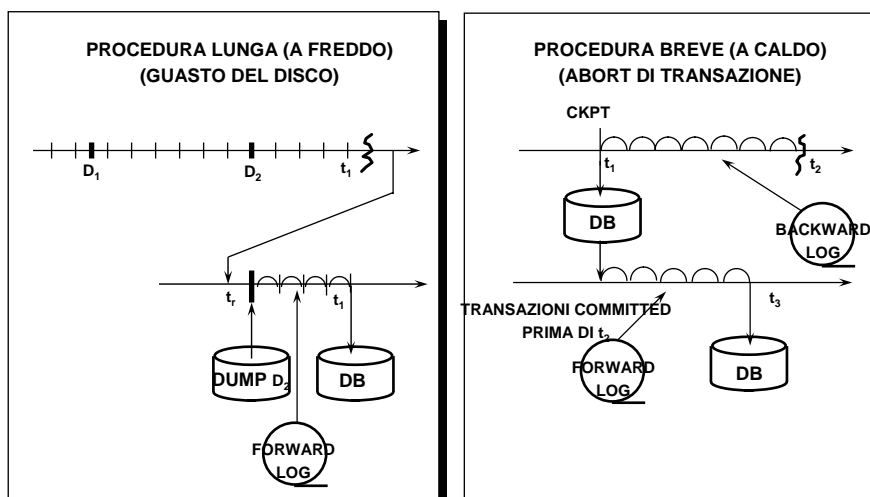
LE TRANSAZIONI VENGONO CLASSIFICATE COME

- **COMPLETATE**
I RISULTATI SONO MEMORIZZATI IN MEMORIA STABILE
- **COMMITTED MA NON COMPLETATE**
I RISULTATI DEVONO ESSERE RIFATTI (REDO PROCESS)
- **NON COMMITTED (O ATTIVE)**
LE AZIONI DEVONO ESSERE DISFATTE (UNDO PROCESS)
- **RIPARTENZA A CALDO (WARM RESTART)**
 - VIENE USATA PER GUASTI O ERRORI CHE NON RICHIEDONO L'ARRESTO DEL SISTEMA
- **RIPARTENZA A FREDDO (COLD RESTART)**
 - VIENE USATA PER I GUASTI PIU' GRAVI CHE RICHIEDONO L'ARRESTO DEL SISTEMA E IL CARICAMENTO DEL DUMP
 - VIENE SEGUITA DA UNA RIPARTENZA A CALDO

© Fabio A. Schreiber

Tecnologia server 56

PROCEDURE DI RIPRISTINO



© Fabio A. Schreiber

Tecnologia server 57

IL SEGUITO ... ALLA PROSSIMA PUNTATA

- **ARCHITETTURE DI SERVER PER BASI DI DATI**
 - BASI DI DATI 2 (L. S.)
- **GESTORE DELLA SICUREZZA**
 - TECNOLOGIE DEI SISTEMI INFORMATIVI (L. S.)
 - SISTEMI INFORMATIVI AVANZATI (L. S.)