KONTROL KONKURENSI TERDISTRIBUSI (DCC)

Mekanisme DCC ini memastikan kekonsistensi-an database.

Jika transaksi konsisten secara internal, cara termudah adalah mengeksekusi satu transaksi satu per satu.

Level konkurensi menjadi parameter yang terpenting dalam sistem terdistribusi. [Balter et.al, 1992]

Mekanisme kontrol konkurensi berusaha menyeimbangkan antara konsistensi database dan level konkurensi yang tinggi.

TEORI SERIAL
Sebuah jadual S (disebut history) ditentukan sebagai sekumpulan transaksi T = {T_1, T_2, …,T_n} dan dispesifikasikan sebagai eksekusi operasi transaksi yang terurut.

2 buah operasi $O_{ij}(x)$ dan $O_{kl}(x)$ (i dan k tidak perlu berbeda) mengakses database entitas x yang sama, ini disebut dengan terjadinya konflik jika salah satu merupakan operasi tulis. Operasi baca tidak menjadikan konflik satu sama lain.

Ada 2 macam tipe konflik :
- Read-write
- Write-write

2 buah transaksi menjadi konflik jika 2 buah operasi tersebut mengindikasikan bahwa urutan eksekusi adalah penting.

Jadual yang lengkap
$S_T c$ didefinisikan sebagai sekumpulan transaksi T = { T_1, T_2, …,T_n } merupakan urutan parsial $S_T c = \{\Sigma_T, \prec_T\}$ dimana

1. $\Sigma_T = U_{i=1}^{n} \Sigma_i$
2. $\prec_T \supseteq U_{i=1}^{n} \prec_i$
3. untuk setiap terjadi operasi konflik O_{ij} dan $O_{kl} \in \Sigma_T$, $O_{ij} \prec_T O_{kl}$ atau $O_{kl} \prec_T O_{ij}$.

Diasumsikan ada 2 buah transaksi :

T1 : read (x)	T2 : read (x)
x ← x + 1	x ← x + 1
write (x)	write (x)
commit	commit

Jadual lengkap : $S_T c$ untuk $T = \{T_1, T_2\}$ ditulis dalam urutan parsial $S_T c = \{\Sigma_T, \prec_T\}$, dimana

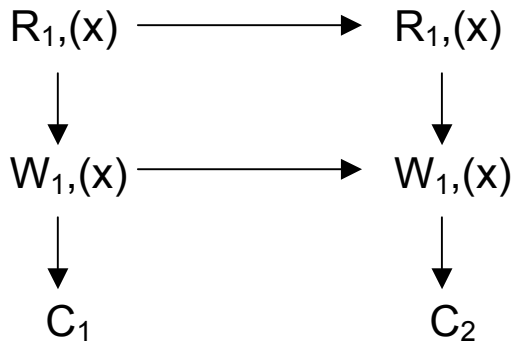
$\Sigma_1 = \{R_1(x), W_1(x), C_1\}$
$\Sigma_2 = \{R_2(x), W_2(x), C_2\}$

sehingga

$\Sigma_T = \Sigma_1 \cup \Sigma_2 = \{R_1(x), W_1(x), C_1, R_2(x), W_2(x), C_2\}$

dan

$\prec_T = \{(R_1,R_2),(R_1,W_1), (R_1,C_1), (R_1,W_2),(R_1,C_2), (R_2,W_1), (R_2,C_1),(R_2,W_2), (R_2,C_2), (W_1,C_1),(W_1,W_2), (W_1,C_2) (C_1,W_2),(C_1,C_2), (W_2,C_2)\}$



Secara sederhana :

$S_T c$ = = {$R_1(x)$, $R_2(x)$, $W_1(x)$,C_1, $W_2(x)$,C_2}

Jadual parsial : P = {Σ, $\prec$}, P' = {Σ', $\prec$'} merupakan awal / prefix dari P jika

1. $\Sigma' \subseteq \Sigma$;

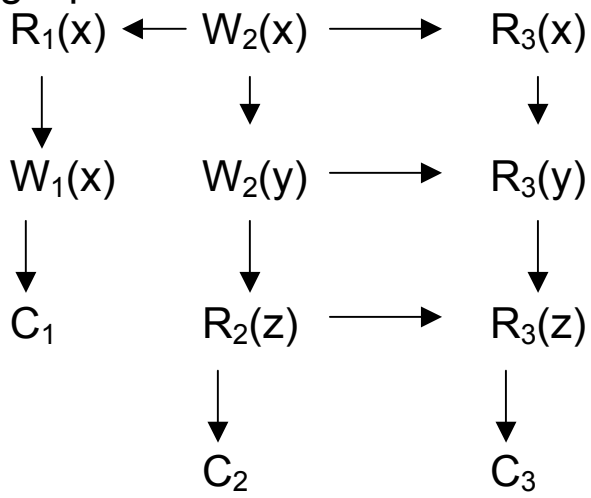
2. $\forall e_i \in \Sigma'$, $e_1 \prec' e_2$ jika dan hanya jika $e_1 \prec e_2$

3. $\forall e_i \in \Sigma'$, jika $\exists e_j \in \Sigma$ dan $e_j \prec e_i$, maka jika $e_j \in \Sigma'$

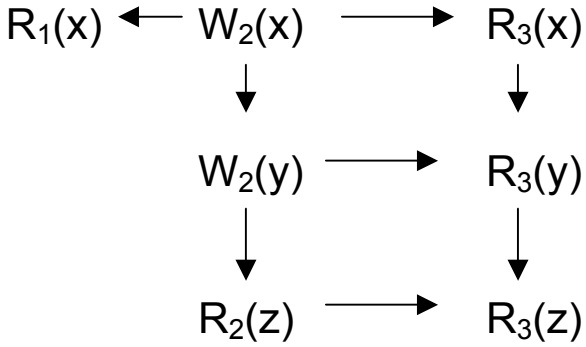
Diasumsikan ada 3 buah transaksi :

T1 : read (x)	T2 : write (x)	T3 : Read(x)
write (x)	write (y)	read(y)
commit	read(z)	read(z)
	commit	commit

jadual lengkap S^c



Prefix S^c



Jika dalam sebuah jadual S, operasi dari berbagai transaksi terjadi secara berurutan, berarti jadual dikatakan serial.

Contoh dari ketiga transaksi sebelumnya

S = { $W_2(x)$, $W_2(y)$,$R_2(z)$, C_2, $W_1(x)$, $R_1(x)$, C_1,
$R_3(x)$, $R_3(y)$, $R_3(z)$, C_3}

$T_2 \prec_S T_1 \prec_S T_3$ atau $T_2 \rightarrow T_1 \rightarrow T_3$

Konflik equivalence
2 buah jadual S_1 dan S_2 didefinisikan pada sekumpulan transaksi T, dikatakan equivalence jika untuk setiap pasangan operasi yang konflik O_{ij} dan O_{kl} ($i \neq k$) dimanapun $O_{ij} \prec_1 O_{kl}$ maka $O_{ij} \prec_2 O_{kl}$.

Sebuah jadual dikatakan dapat diserialkan (serializable) jika dan hanya jika dari konflik equivalence ke jadual berseri. Disebut juga dengan conflict-based serializability.

Jadual eksekusi transaksi di setiap site disebut dengan jadual local. Jika database tidak direplikasi dan setiap jadual local diserialkan, gabungannya (jadual global) juga akan diserialkan.

Contoh :

T1 : read (x)	T2 : read (x)
x ← x + 5	x ← x * 10
write (x)	write (x)
commit	commit

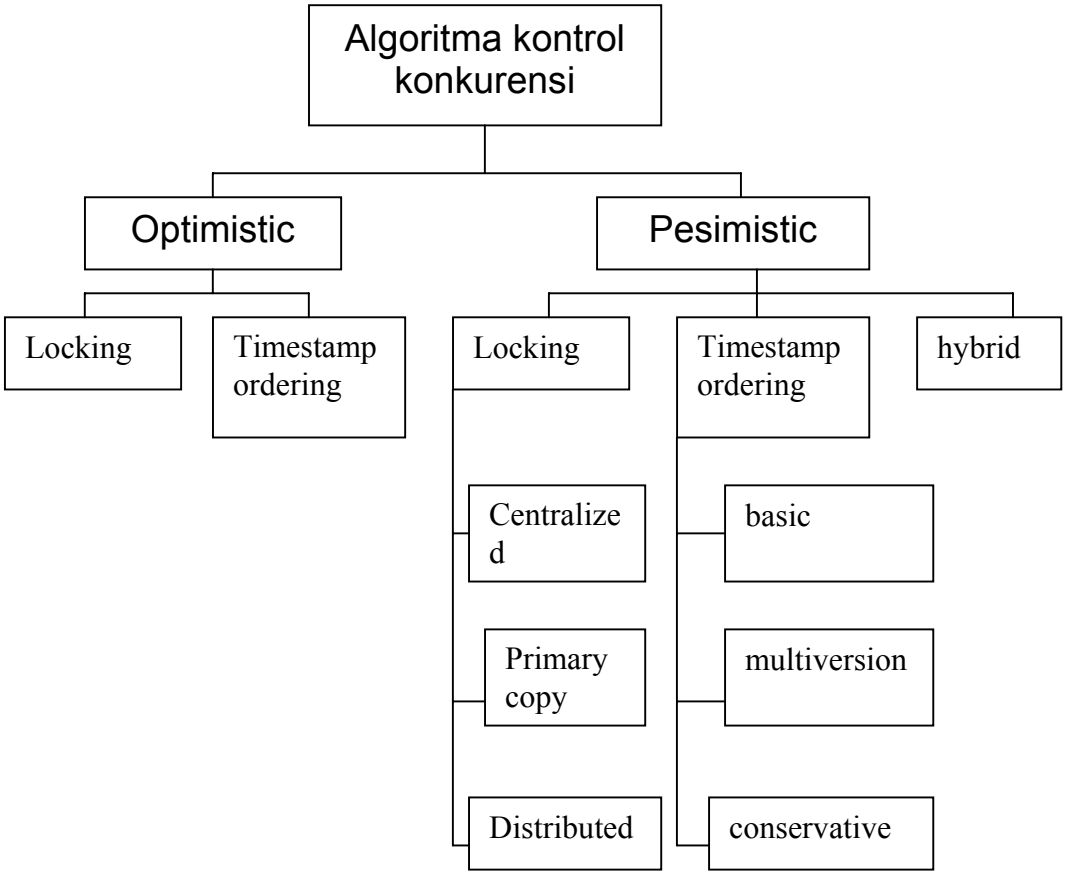
Seharusnya kedua transaksi ini dilaksanakan di kedua site. misalnya ada 2 jadual yang dilaksanakan secara local di kedua site :

$S_1 = \{ R_1(x), W_1(x), C_1, R_2(x), W_2(x), C_2\}$
$S_2 = \{ R_2(x), W_2(x), C_2, R_1(x), W_1(x), C_1\}$

Ini yang disebut mutually consistency di kedua database local. Konsistensi ini membutuhkan nilai yang sama/ identik untuk semua item data yang direplikasi. Jadualnya disebut dengan one-copy serializable. Ada kondisi2, yaitu :

1. Setiap jadual local diserialkan
2. 2 operasi konflik dibuat dalam urutan yang sama untuk semua jadual local dimana berada.

TAKSONOMI MEKANISME KONTROL KONKURENSI



Metode pesimistik
Mengsinkronkan eksekusi transaksi yang konkuren di awal siklus hidup eksekusi.

Metode optimistic
Menunda sinkronisasi dari transaksi hingga selesai.

Pendekatan locking
Sinkronisasi dari transaksi dicapai dengan lock fisik atau logika pada beberapa bagian dari database. Ukuran bagian ini (disebut locking granularity) menjadi hal yang penting.

- Centralized locking

  Satu site dibuat menjadi site utama dimana tabel yang terkunci/ lock untuk semua database disimpan dan diberikan tanggungjawab untuk membuka locking.

- Primary copy locking

Satu kopi untuk setiap unit lock dibuat sebagai kopi utama dan kopian ini harus dilock untuk tujuan pengaksesan unit tertentu. Misalnya jika unit x direplikasi ke site 1, 2 dan 3, salah satu site dipilih sebagai site utama untuk x. Semua transaksi yang mengakses x terkunci di site 1 sebelum dapat mengakses kopi x. jika database tidak terreplikasi mekanisme locking kopi utama mendistribusikan tanggungjawab manajemen lock diantara sejumlah site.

- Decentralized locking

Tugas manajemen locking dibagi ke seluruh site dalam jaringan. Eksekusi sebuah transaksi melibatkan partisipasi dan koordinasi dari para penjadual di lebih dari satu site.

Timestamp ordering (TO) melibatkan organisasi urutan eksekusi dari transaksi sehingga dapat dimaintain secara mutual dan interconsistency. Urutan ini dimaintain dengan menugaskan timestamp ke transaksi dan item data yang disimpan di database. Algoritma ini disebut TO dasar, TO multiversion, atau TO conservative.

Hybrid

Beberapa locking-based dan timestamp yang digunakan bersama.

LOCKING-BASED CC

Ide utamanya adalah untuk memastikan data dapat dibagi dengan operasi konflik yang diakses oleh satu operasi pada satu saat.

"Lock" untuk setiap unit lock.

Locking terjadi di transaksi sebelum diakses dan direset di akhir penggunaan. Unit lock tidak dapat diakses dengan sebuah operasi jika sudah dilock sebelumnya oleh yang lain.
Permintaan lock oleh sebuah transaksi diberikan jika lock yang terhubung tidak ditahan oleh transaksi lainnya.

Ada 2 tipe lock (disebut mode lock) sehubungan dengan setiap unit lock :

1. Read lock (rl)
2. Write lock (wl)

Tablenya :

	$rl_i(x)$	$wl_i(x)$
$rl_i(x)$	Compatible	Not compatible
$wl_i(x)$	Not compatible	Not compatible

Lihat potongan algoritma hal. 287 Ozsu.

Contoh
Ada 2 transaksi

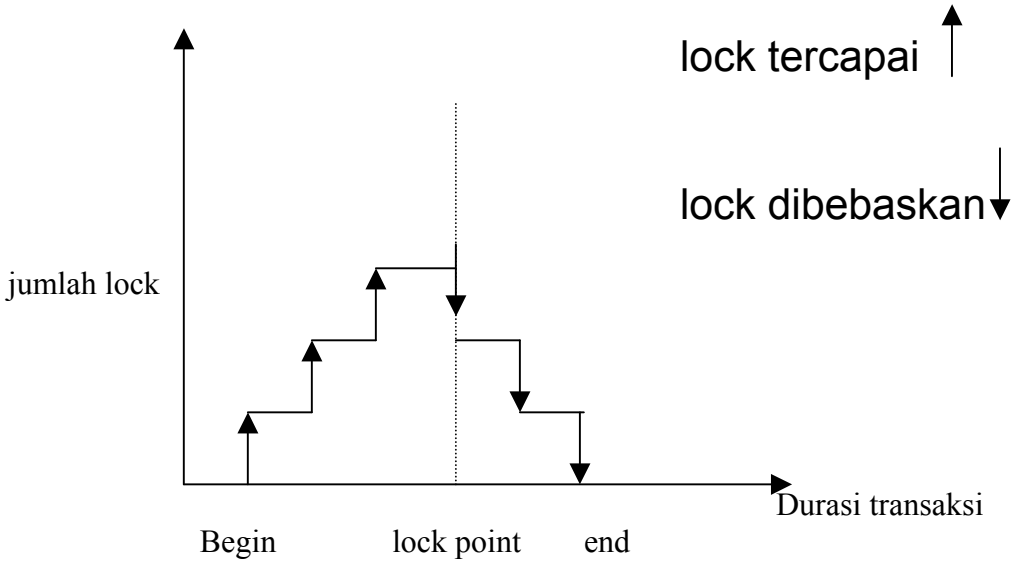
T1 : read (x)	T2 : read (x)
x = x + 1	read (y)
write (x)	y = y * 2
read(y)	write(y)
y = y – 1	commit
write(y)	
commit	

jadualnya adalah

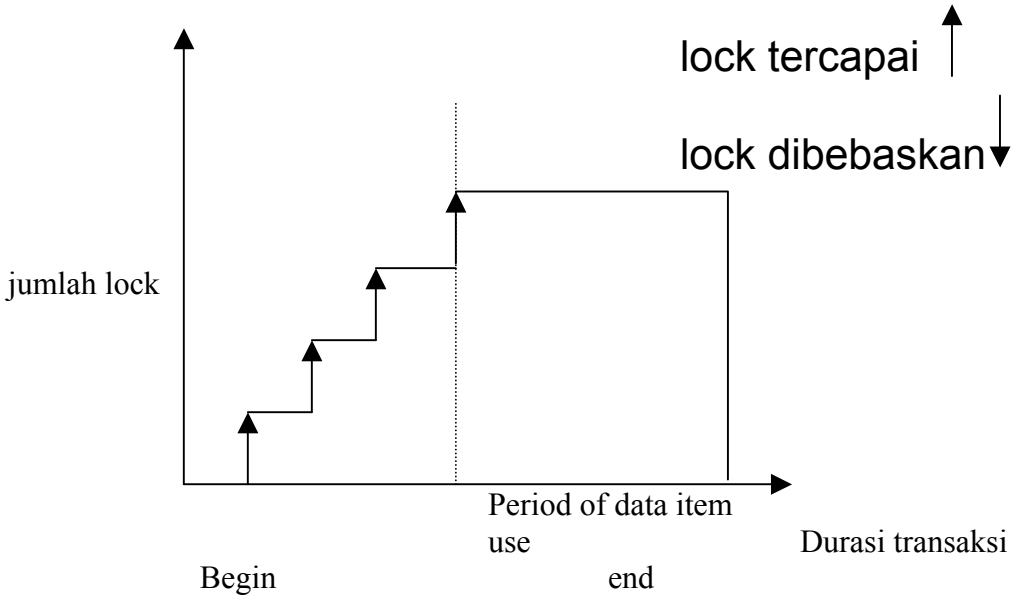
S = { $wl_1(x)$, $R_1(x)$, $W_1(x)$, $lr_1(x)$, $rl_2(x)$, $R_2(x)$, $lr_2(x)$, $wl_2(y)$, $R_r(y)$, $W_2(y)$, $lr_2(y)$, C_2, $wl_1(y)$, $R_1(y)$, $W_1(y)$, $lr_1(y)$, C_1}

$lr_i(z)$, membebaskan lock dari z yang menyebabkan transaksi T_1 tertahan.

2-phase locking

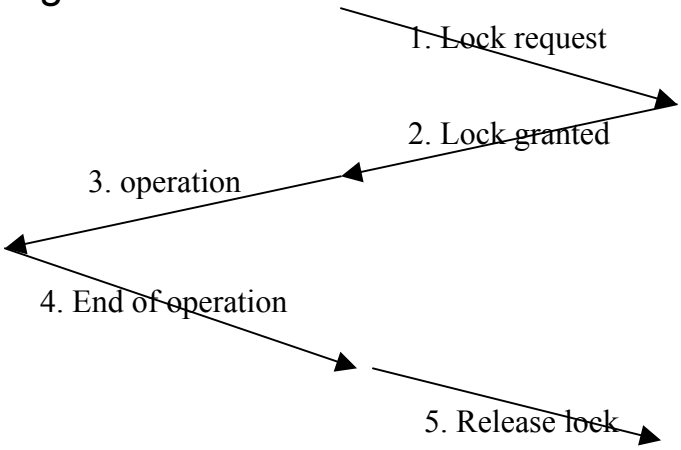


Strict 2-phase locking



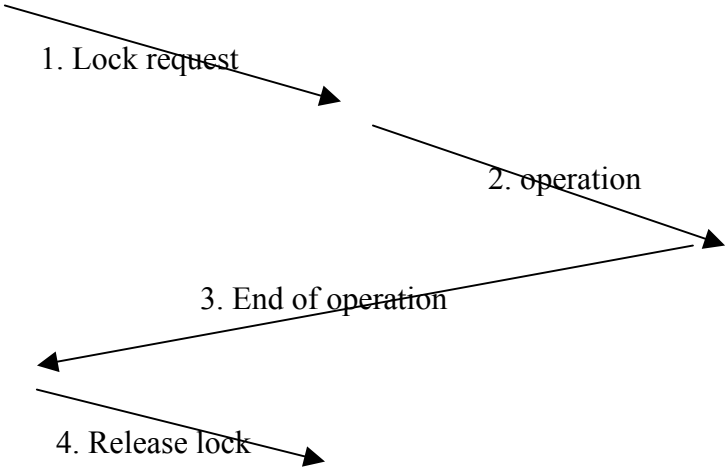
- Centralized 2PL

Data processor at coordinating TM central site LM
Participating sites



- Primary copy 2 PL
- Distributed 2 PL

coordinating TM Participating LMs Participating DPs

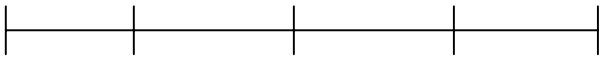


TIMESTAMP-BASED CC
TO rule (Timestamp Ordering)
Diberikan 2 operasi konflik O_{ij} dan O_{kl} terhadap transaksi T_i dan T_k dieksekusi sebelum O_{kl} jika dan hanya jika $ts(T_i) < ts(T_i)$. T_i dikatakan transaksi yang lebih tua dan T_k dikatakan transaksi yang lebih muda.

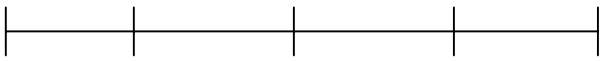
- Basic TO
- Conservation TO
- Multiversion TO

Lihat contoh algoritma hal. 300-301 untuk BTO

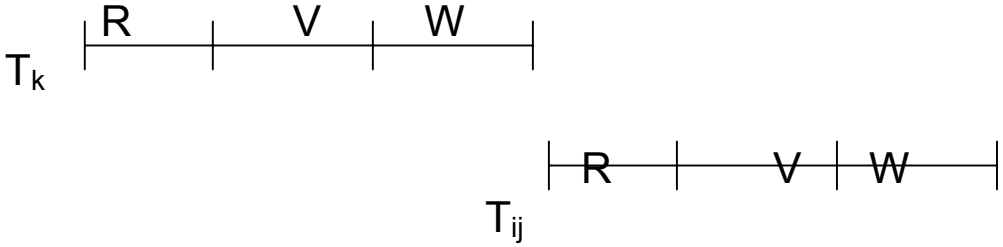
OPTIMISTIC CC
Fase dari eksekusi transaksi yang pesimistik



Fase dari eksekusi transaksi yang optimistic

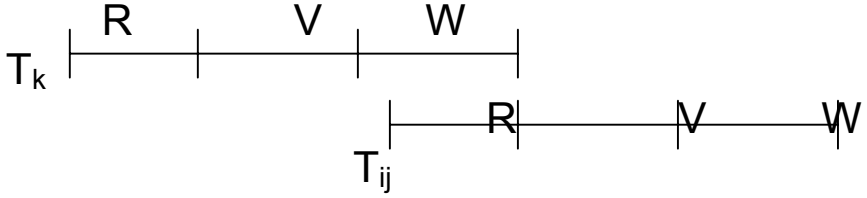


Skenario yang memungkinkan
Rule 1



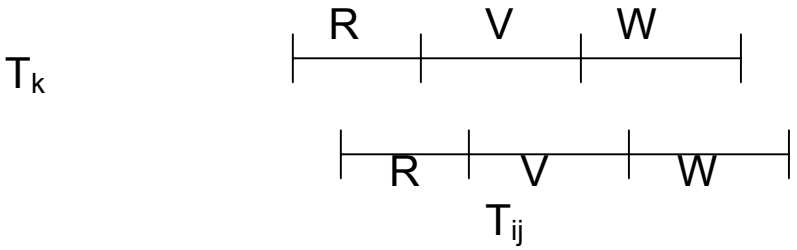
Jika semua transaksi T_k , dimana $ts(T_k) < ts(T_{ij})$telah menyelesaikan fase tulis sebelum T_{ij} memulai fase baca, validasi berhasil karena eksekusi transaksi secara terurut serial.

Rule 2



Jika ada transaksi T_k yang $ts(T_k) < ts(T_{ij})$ telah menyelesaikan fase tulis dimana T_{ij} berada pada fase baca, validasi berhasil jika $WS(T_k) \cap RS(T_{ij}) = \varnothing$

Rule 3



Jika ada transaksi T_k yang $ts(T_k) < ts(T_{ij})$ telah menyelesaikan fase baca sebelum T_{ij} berada pada fase baca, validasi berhasil jika $WS(T_k) \cap RS(T_{ij}) = \varnothing$ dan $WS(T_k) \cap WS(T_{ij}) = \varnothing$

.