

SEB

DB2 RUG 2007

DEADLOCK OCH TIMEOUT

Av Lars-Göran Karlsson



SEB GROUP IT



- SEB Group IT är en IT enhet inom bankkoncernen SEB
- DB-design är en del av en utvecklingsavdelning som centralt förser IT avdelningar med support för
 - Datamodellering
 - Logisk databasdesign
 - Fysisk databasdesign
 - Övriga frågeställning om databashanterare
 - Hanterar IMS/DB, DB2, Oracle och SQL Server



STATUS DB2 V8

- De svenska systemen har new function mode sedan november 2006
- De danska systemen har kompatibel mode januari 2007
- De tyska systemen har kompatibel mode januari 2007



AGENDA DEADLOCK OCH TIMEOUT

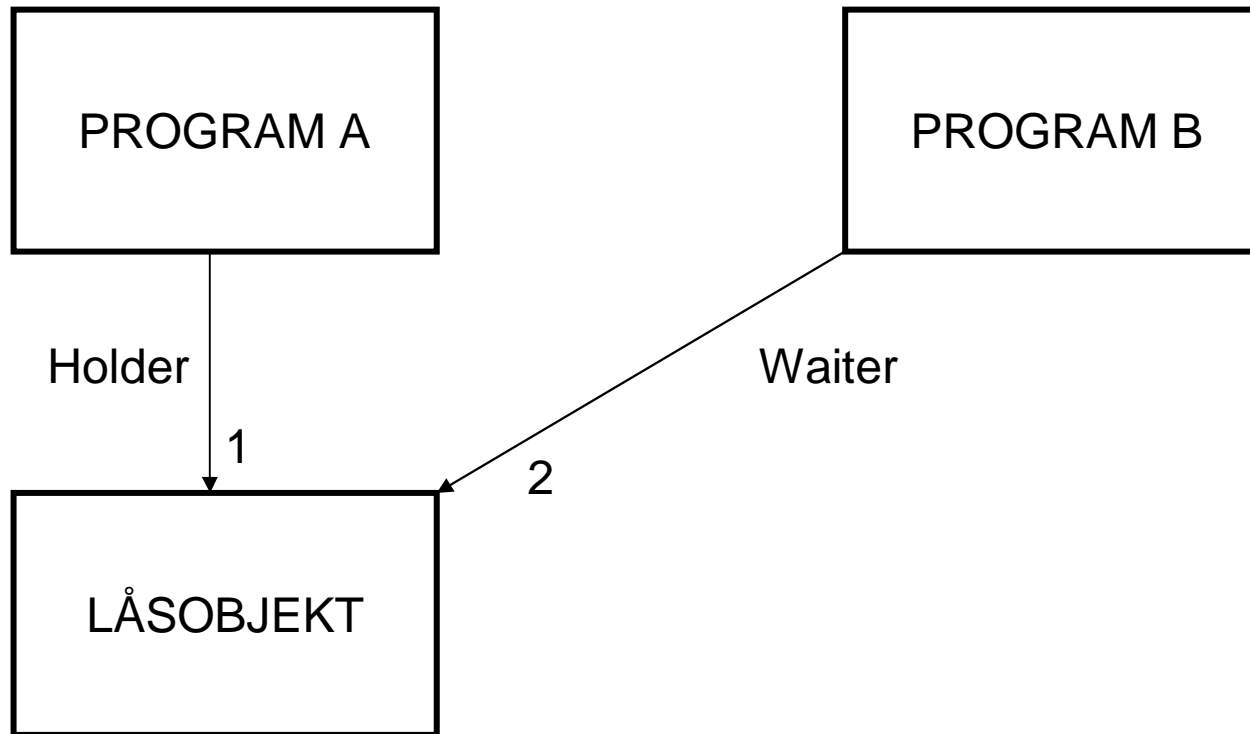
- Kort introduktion till deadlock och timeout
- Uppföljningsrutiner
- Erfarenheter för att undvika deadlock och timeout
- Några exempel på deadlock

VAD ÄR DEADLOCK OCH TIMEOUT ?

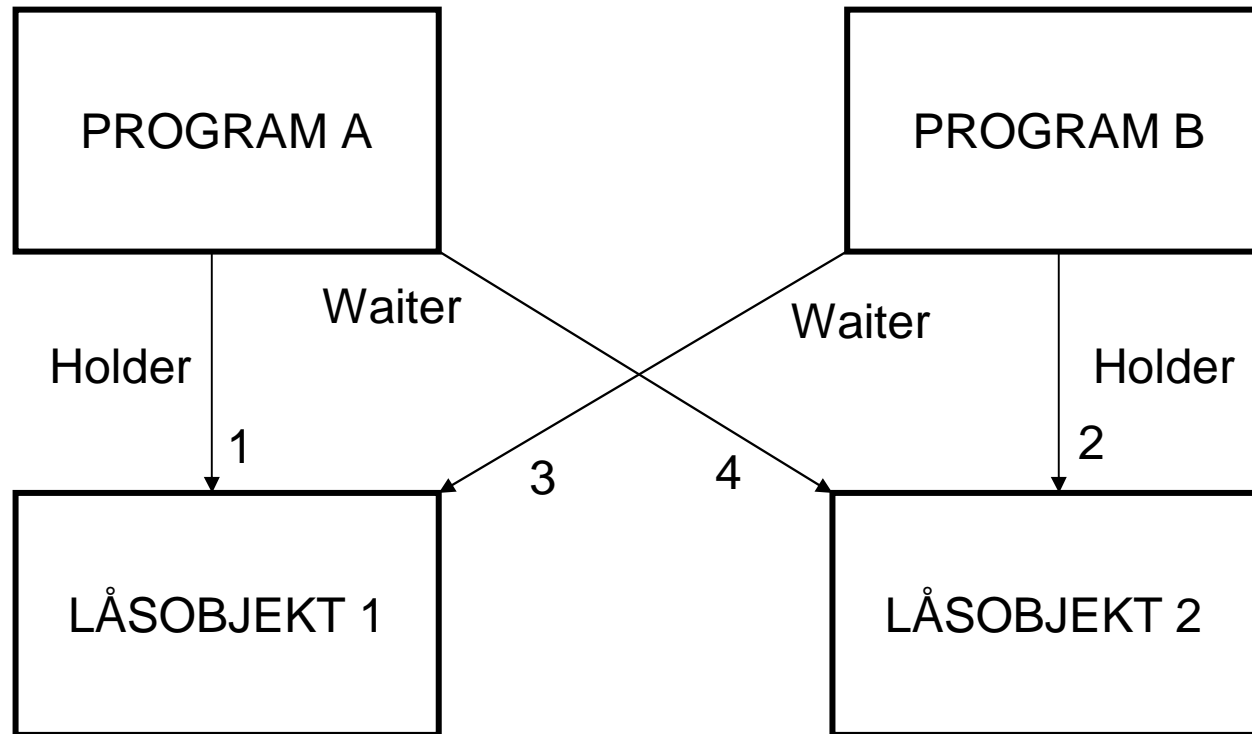
- Deadlock uppstår när två program som exekverar samtidigt och försöker läsa eller updatera en eller flera rader som det andra programmet redan håller ett lås på
- Timeout uppstår när ett program har väntat för länge på ett annat program som håller lås på en eller flera rader som det väntande program vill komma åt



GRAFISK BILD AV TIMEOUT



GRAFISK BILD AV DEADLOCK



DB2 SYSTEMPARAMETRAR

- DSNZPARM för timeout
 - Är satt till 1 minut
- DSNZPARM för deadlock
 - Är satt till 1 sekund



INSAMLING AV INFORMATION

- Vi samlar in uppgifter om deadlock och timeout som uppstår både i IMS/DB och DB2 på Z/os i produktionsmiljön
- Uppsamlingen sker varje natt och skriver de senaste dygnets inträffade händelser i två DB2 tabeller, en med uppgifter för DB2 och en med uppgifter för IMS/DB

REGISTRERADE UPPGIFTER OM DEADLOCK OCH TIMEOUT

- För varje händelse registreras
 - Datum och klockslag
 - Namn på planer och correlation-id
 - Databaser och tabeller
 - Typ av låssituation, deadlock eller timeout
 - Holder och waiter per program
 - Sid- och ev. radnummer
 - Typ av låsnivå, X-, U- eller S-lock
 - Typ av lås, row, page eller table
 - Typ av program, IMS, batch, Utility, m.m.
- Fyra rader skrivs i DB2 tabellen för en deadlock
- Två rader skrivs i DB2 tabellen för en timeout

UPPFÖLJNINGSRUTIN

- Varje vecka ställs en värstinglista per plan samman utifrån insamlat material
- Värstinglistan omfattar de senaste 30 dagarnas händelser
- Förvaltande DBA meddelas om sina system och som då undersöker de händelser som finns
- Kontakt tas med systemförvaltare för att beskriva situationen och gör upp en åtgärdsplan
- Notering om åtgärd och resultat görs



HUR UNDVIKA TIMEOUT OCH DEADLOCK

- Låsningstiden
- Bearbetningssekvens
- Låsnivå
- Cursorlås
- Hot spots



LÅSNINGSTID

- Reducera tiden för att ett lås hålls
 - INSERT, UPDATE och DELETE i slutet av programmens exekvering
 - Väl trimmade SQL anrop
 - Väl trimmade tabeller
- COMMIT görs med ett passande tidsintervall
- Alltid COMMIT



SEKVENSIELL BEARBETNING

- Använd sekvensiell uppdatering
 - Alla program använder samma bearbetningssekvens på tabeller
 - Försiktighet när både DB2 och IMS/DB används
 - Försiktighet när join används
- Håll hög cluster ratio på index

LÅSNIVÅ OCH CURSORS

- Överväg radlås, men är dyrbarare än sidlås
- Koda med `with for fetch only` på cursor när det är passande
- Koda med `with for update of` på cursor när det är passande
- `CLOSE CURSOR` så snart som möjligt om `WITH HOLD` är använts


HOT SPOTS

- Hot spots är när ett fåtal rader upprepningsvis och ofta uppdateras
- Exempelvis
 - Räknare
 - Antals- eller beloppssummor
- Sprid updateringar på fler rader eller sidor
- Räknare ska uppdateras före dom läses
- Använd radlås

EXEMPEL PÅ FEL ANROPSSEKVENENS

- En läsning av en rad gjordes innehållande en räknare
- Därefter uppdaterades raden med +1
- Resultatet blev deadlocks vid intensivt nyttjande av tabellen för program som kördes parallellt.
- Detta ändrades så att en update gjordes före läsning, vilket resulterade i enbart en wait

EXEMPEL PÅ PROBLEM MED HOT SPOT

- En stor tabell där vissa rader är hot spot, samtidigt är tabellen av varierande antal rader och som resulterade i timeout.
 - Tabell med ytterst få rader resulterade i deadlock
 - Bägge fallen skapade tablespace scan
 - Åtgärdades med tabell parametern VOLATILE
 - Före v8 manipulerades statistiken så att det alltid såg ut som att det fanns många rader i tabellen
- 

EXEMPEL PÅ PROBLEM MED BEARBETNINGSEKVENSER

- Fem intensivt använda program gjorde ungefär samma uppdaterande anrop mot samma tabeller
- Fyra av dem gjorde anropen i samma sekvens
- Men det femte gjorde anropen i annan sekvens
- Resultatet blev att vid peak-tid blev det mycket deadlock
- Detta åtgärdades genom att det femte programmet ändrades så att bearbetningssekvensen blev den samma som för de övriga

EXEMPEL MED PROBLEM MED CLUSTER RATIO

- Mycket insert och delete orsakade låg cluster ratio som i sin tur orsakade deadlock mellan läsande och uppdaterande program
- Select, Insert och Update sker online dagligen av ett antal program
- Delete skedde tidigare med ett batch som kördes varannan vecka.
- Delete sker nu två gånger per vecka
- Free space % per page ändrades till ett högre värde
- Resultat blev en jämnare balans mellan insert och delete som gör att det inte längre blir deadlock