Locking

KdG Karel de Grote Hogeschool

- cursus ‘Databanken 1’ blz. 66
- Deze PowerPoint
- Aanvullend: Oracle Database 11g: SQL Fundamentals I Exam Guide Blz. 436-437

KdG Karel de Grote Hogeschool

Locking

Een databank wordt meestal door meerdere gebruikers tegelijk gebruikt. Onder deze gebruikers zijn er

- Lezers: halen informatie uit de databank (SELECT)
- Schrijvers: veranderen de databank inhoudelijk (DML)

Lezers kunnen simultaan dezelfde informatie opvragen.

- Schrijvers kunnen NIET tegelijkertijd dezelfde informatie wijzigen.

Locking

Dit wordt geregeld door het locking mechanisme op rij niveau.

- De databank blokkeert de rijen die de schrijver aanpast:
- alleen hij ziet de **nieuwe** situatie
- alleen hij kan de betrokken rijen wijzigen
- andere gebruikers zien de **oude** situatie (=read consistent view) **tot** de schrijver COMMIT geeft
- Na COMMIT wordt de lock op de rijen vrijgegeven.

Read consistentie

Sessie user 1 = Schrijver

```
SELECT salaris
FROM medewerkers
WHERE sofi_nr='999666666';
```

```
SALARIS
55000
```

```
UPDATE medewerkers
SET salaris=salaris*1.1
WHERE sofi_nr='999666666';
```

```
SELECT salaris
FROM medewerkers
WHERE sofi_nr='999666666';
```

```
SALARIS
60500
```

```
COMMIT;
```

De rij wordt voor user 1 gelockt
Alleen hij ziet de wijziging

Sessie user 2 = Lezer

```
SELECT salaris
FROM medewerkers
WHERE sofi_nr='999666666';
```

```
SALARIS
55000
```

```
SELECT salaris
FROM medewerkers
WHERE sofi_nr='999666666';
```

```
SALARIS
60500
```

Na commit door user 1 wordt de rij weer vrijgegeven => andere users zien de wijziging ook

Concurrency control/locking

Probleemstelling:

Twee users zijn bevoegd om salarissen aan te passen in de ONDERNEMING database.

Het jaarloon van medewerker 999666666 bedraagt momenteel 55000 en moet met 10% verhoogd worden.

Door een communicatiestoornis willen beide users de loonsverhoging doorvoeren zonder het van mekaar te weten.

Concurrency control/locking

Sessie user 1

```
SELECT salaris
FROM medewerkers
WHERE sofi_nr='999666666';
```

```
SALARIS
55000
```

```
UPDATE medewerkers
SET salaris=salaris*1.1
WHERE sofi_nr='999666666';
```

Rij is gelockt door user 1

User 2 ziet een

read consistent view

```
COMMIT;
```

Lock komt vrij

Sessie user 2

Update user 2 wordt niet doorgevoerd tgv lock update user

```
SELECT salaris
FROM medewerkers
WHERE sofi_nr='999666666';
```

```
SALARIS
55000
```

```
UPDATE medewerkers
SET salaris=salaris*1.1
WHERE sofi_nr='999666666';
```

Update wordt nu (2e X) uitgevoerd, dus

```
ROLLBACK;
```

Concurrency control/locking

Het probleem is opgelost, maar er werd niet efficiënt gewerkt.

Er bestaat een grote kans op inconsistenties

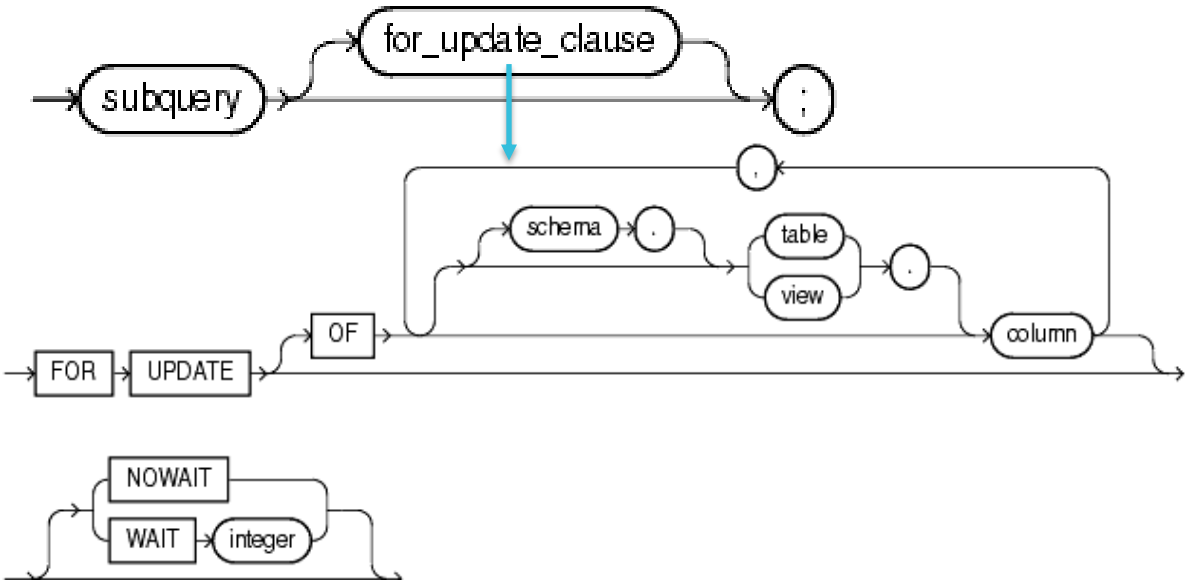
Hoe kan user 2 weten of de gegevens die hij selecteert niet gelockt zijn?

alvorens DML instructie te geven,

gebruik maken van

```
SELECT … FOR UPDATE  [NOWAIT]
```

SELECT FOR UPDATE



Definitie
S Q L
Reference
p. 163

Met SELECT FOR UPDATE selecteert de gebruiker een aantal rijen, die *bovendien* voor hem gelockt worden.

Bedoeling is de SELECT FOR UPDATE te laten volgen door een DML instructie op de geselecteerde (gelockte) rijen.

De lock wordt opnieuw vrijgegeven door COMMIT/ROLLBACK.

SELECT FOR UPDATE

Voorbeeld

```
SELECT salaris
FROM medewerkers
WHERE sofi_nr='999666666'
FOR UPDATE;
```

Bij een SELECT FOR UPDATE heb je 2 mogelijke scenario's:

1. De instructie geeft een resultatentabel terug. Dat wil zeggen dat er op de geselecteerde rijen geen lock stond en dat er nu één op geplaatst wordt.
2. De instructie geeft geen rijen terug (blokkeert). Dat wil zeggen dat op (een aantal van) de betrokken rijen een lock staat. De gebruiker kan pas verder werken *nadat* de gebruiker die de rijen lockte, deze terug heeft vrijgegeven.

SELECT FOR UPDATE NOWAIT

Voorbeeld

```
SELECT salaris
FROM medewerkers
WHERE sofi_nr='999666666'
FOR UPDATE NOWAIT;
```

Bij een `SELECT FOR UPDATE NOWAIT` zal, wanneer de rijen gelockt zijn, de gebruiker die de bovenstaande SELECT onmiddellijk een foutboodschap krijgen. Hij kan voorlopig de rij betreffende medewerker '999666666' niet wijzigen, maar hij kan ondertussen op andere rijen werken (DML op andere niet gelockte rijen, SELECTs op alle rijen)

SELECT FOR UPDATE NOWAIT

Sessie user 1

```
SELECT salaris
FROM medewerkers
WHERE sofi_nr='999666666'
FOR UPDATE NOWAIT;
```

```
SALARIS
55000
```

```
UPDATE medewerkers
SET salaris=salaris*1.1
WHERE sofi_nr='999666666';

COMMIT;
```

De door SELECT FOR UPDATE gelockte rijen komen nu terug vrij.

Sessie user 2

```
SELECT salaris
FROM medewerkers
WHERE sofi_nr='999666666'
FOR UPDATE NOWAIT;
```

```
ORA-00054: resource busy and
acquire with NOWAIT specified or
timeout expired
```

user 2 weet dat de rijen gelockt zijn en dat hij later opnieuw moet proberen.

```
SELECT salaris
FROM medewerkers
WHERE sofi_nr='999666666'
FOR UPDATE NOWAIT;
```

```
SALARIS
60500
```

user 2 ziet de gewijzigde situatie en weet dat de update overbodig is.
ROLLBACK of COMMIT om de lock terug vrij te geven!!

SELECT FOR UPDATE WAIT

Middenweg tussen niet en eeuwig wachten

```
SELECT salaris
FROM medewerkers
WHERE sofi_nr='999666666'
FOR UPDATE WAIT 5;
```

Als de actie na maximaal 5 seconden nog niet kan doorgaan, geeft het systeem een foutmelding en wordt de actie geannuleerd.

SELECT FOR UPDATE OF

Bemerking (hoewel join pas in periode 2 behandeld wordt…)

```
SELECT sofi_nr,achternaam,salaris,afd_naam
FROM medewerkers
JOIN afdelingen USING (afd_nr)
WHERE upper(geslacht)='M' AND
UPPER(afd_naam)='PRODUCTIE'
FOR UPDATE;
```

➔ beide tabellen in de FROM worden gelockt

Stel dat we in de daarop volgende DML instructie enkel salaris willen aanpassen, dan is de volgende instructie beter:

```
SELECT sofi_nr,achternaam,salaris,afd_naam
FROM medewerkers
JOIN afdelingen USING (afd_nr)
WHERE upper(geslacht)='M' AND UPPER(afd_naam)='PRODUCTIE'
FOR UPDATE OF salaris;
```

➔ enkel de tabel MEDEWERKERS wordt gelockt.