

Ausarbeitung UE10

(Ausarbeitung zu 1 ganz unten aufgrund von Ausarbeitungsmethode!)

2. Vergabe und Entzug von Rechten

-- 2.1 mit Johann Hoffmann

```
GRANT SELECT ON departments TO s1710307108;
COMMIT;
```

```
SELECT *
FROM s1710307108.departments;
```

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	9999	Dept 9999	<null>	<null>
2	10	Administration	200	1700
3	20	Marketing	201	1800
4	50	Shipping	124	1500
5	60	IT	103	1400
6	80	Sales	149	2500
7	90	Executive	100	1700
8	110	Accounting	205	1700
9	190	Contracting	<null>	1700

-- 2.2

```
SELECT table_name
FROM user_tables;
```

	TABLE_NAME
1	COUNTRIES
2	DEPARTMENTS
3	DEPT
4	EMP
5	EMPLOYEES
6	JOBS
7	JOB_GRADES
8	JOB_HISTORY
9	LOCATIONS
10	MY_EMPLOYEE
11	REGIONS
12	TEST

```
SELECT table_name, owner, privilege
FROM all_tables
INNER JOIN all_tab_privs USING (table_name)
WHERE owner <> 'S1710307099'
AND privilege = 'SELECT';
```

	TABLE_NAME	OWNER	PRIVILEGE
110	WWW_TLOW_DUAL100	AFEX_040200	SELECT
117	TEST	S1710307108	SELECT
118	TEST	S1710307108	SELECT
119	DEPARTMENTS	S1710307108	SELECT
120	DEPARTMENTS	S1710307108	SELECT
121	MAP_OBJECT	SYS	SELECT
122	PSTUBTBL	SYS	SELECT
123	WRI\$_ADV_ASA_RE...	SYS	SELECT
124	WRI\$_HEATMAP_TO...	SYS	SELECT

3. COMMIT, SAVEPOINT und ROLLBACK

-- 3.1

```
CREATE TABLE my_employee
AS
SELECT *
FROM employees;
```

-- 3.2

-- pre update

```
SELECT *
FROM my_employee
WHERE job_id LIKE '%MGR'
OR job_id LIKE '%MAN';
```

	employee_id	first_name	last_name	email	phone_number	hire_date	job_id	salary
1	124	Kevin	Mourgos	KMOURGOS	650.123.5234	1999-11-16 00:00:00	ST_MAN	5800.00
2	149	Eleni	Zlotkey	EZLOTKEY	011.44.1344.429018	2000-01-29 00:00:00	SA_MAN	10500.00
3	201	Michael	Hartstein	MHARTSTE	515.123.5555	1996-02-17 00:00:00	MK_MAN	13000.00
4	205	Shelley	Higgins	SHIGGINS	515.123.8080	1994-06-07 00:00:00	AC_MGR	12000.00

```
UPDATE my_employee
SET salary = salary * 1.05
WHERE job_id LIKE '%MGR'
OR job_id LIKE '%MAN';
```

-- post update

```
SELECT *
FROM my_employee
WHERE job_id LIKE '%MGR'
OR job_id LIKE '%MAN';
```

	employee_id	first_name	last_name	email	phone_number	hire_date	job_id	salary
1	124	Kevin	Mourgos	KMOURGOS	650.123.5234	1999-11-16 00:00:00	ST_MAN	6090.00
2	149	Eleni	Zlotkey	EZLOTKEY	011.44.1344.429018	2000-01-29 00:00:00	SA_MAN	11025.00
3	201	Michael	Hartstein	MHARTSTE	515.123.5555	1996-02-17 00:00:00	MK_MAN	13650.00
4	205	Shelley	Higgins	SHIGGINS	515.123.8080	1994-06-07 00:00:00	AC_MGR	12600.00

COMMIT;

-- 3.3

```
UPDATE my_employee
SET salary = 6100
WHERE salary <= 5000;
```

-- should yield empty table

```
SELECT *
FROM my_employee
WHERE salary <= 5000;
```

employee_id	first_name	last_name	email	phone_
-------------	------------	-----------	-------	--------

SAVEPOINT after_salary_raise;

-- 3.4

```
DELETE
FROM my_employee;
```

```
SELECT *
FROM my_employee;
```

employee_id	first_name	last_name	email	phone_
-------------	------------	-----------	-------	--------

```
-- 3.5
```

```
ROLLBACK TO after_salary_raise;
```

```
SELECT *
```

```
FROM my_employee;
```

	employee_id	first_name	last_name	email	phone_n
1	100	Steven	King	SKING	515.123.456
2	101	Neena	Kochhar	NKOCHHAR	515.123.456
3	102	Lex	De Haan	LDEHAAN	515.123.456
4	103	Alexander	Hunold	AHUNOLD	590.423.456
5	104	Bruce	Ernst	BERNST	590.423.456
6	107	Diana	Lorentz	DLORENTZ	590.423.556
7	124	Kevin	Mourgos	KMOURGOS	650.123.523
8	141	Trenna	Rajs	TRAJS	650.121.800

```
COMMIT; -- updates only
```

4. Isolationsstufen in Oracle I (Read Committed)

4.1.

Beispiel mit zwei Transaktionen:

```
-- Session 1
```

```
UPDATE emp
```

```
SET sal = 6000
```

```
WHERE empno = 7839;
```

```
-- Session 2 -- muss commit von S1 abwarten
```

```
UPDATE emp
```

```
SET sal = 7000
```

```
WHERE empno = 7839;
```

```
-- Session 1
```

```
SELECT *
```

```
FROM emp
```

```
WHERE empno = 7839; -- liest sal = 6000 (Wert von S1)
```

```
COMMIT; -- erst jetzt darf S2 schreiben
```

Oracle verhindert Lost-Update Fehler durch die Read Committed Isolationsstufe! Wenn zwei Transaktionen versuchen, auf dasselbe Datenobjekt zu schreiben, wird eine der zwei Transaktionen warten müssen bis die andere den Datensatz (mittels Beendung d. Transakt.) freigibt.

Anmerkung: Ich habe mir von Kollegen sagen lassen, dass es hier zu lost updates kommen kann, konnte aber keinen Ausführungsplan erzeugen, der diesen Fehler verursacht. Ich verwende Datenbank-Tools von JetBrains und habe sichergestellt, dass ich in zwei verschiedenen Sessions unterwegs bin und mich im Read Committed Isolationsmodus befinde. Ich habe auch im Internet recherchiert und dort ergab sich ebenfalls, dass Oracle ohne Locks etc. solche Fehler nicht verhindert. Dennoch konnte ich empirisch nie ein anderes Verhalten als das oben Beschriebene feststellen.

4.2.

Beim Ausführen der angeführten Operationen entsteht ein Deadlock durch die letzte Update-Anweisung der zweiten Transaktion:

```
sql> UPDATE emp
..... SET sal = sal * 1.05
..... WHERE sal > 2800
[2018-12-11 22:50:30] [61000][60] ORA-00060: deadlock detected while waiting for resource
```

FOR UPDATE würde schon vorab notwendige Locks beantragen, was ein gegenseitiges Blockieren verhindert.

4.3.

Non-Repeatable Reads können sogar in der Isolationsstufe SERIALIZABLE (/SNAPSHOT) auftreten, siehe 5.3. Um dies zu verhindern können z.B. explizite Sperren vom Benutzer gesetzt werden.

5. Isolationsstufen in Oracle II (Serializable)

1. Durch Änderung der Isolationsstufe bricht Oracle die zweite schreibende Transaktion ab:

```
sql> UPDATE emp
..... SET sal = 7000
..... WHERE empno = 7839
[2018-12-12 08:57:12] [72000][8177] ORA-08177: can't serialize access for this transaction
```

2. Lost-Update-Fehler treten bei Schreiboperationen auf das selbe Objekt auf, daher siehe 1.
3. Es können zum Beispiel dirty reads auftreten: Beide Transaktionen schreiben unterschiedliche Zeilen einer Tabelle, haben jeweils für sich konsistente Daten, können diese aber nach einem Commit nicht mehr mit gleichem Ergebnis abfragen weil die Änderungen an der Tabelle zusammengeführt wurden (Quelle: dbi-services.com):

```
23:18:40 SID=365> set transaction isolation level serializable;
Transaction set.

23:18:41 SID=365> select sum(sal) from EMP where deptno=10;

SUM(SAL)
-----
      8750

23:18:44 SID=365> update EMP set sal=sal+250 where ename='MILLER';
1 row updated.
```

```
23:18:30 SID=12> set transaction isolation level serializable;
Transaction set.

23:18:51 SID=12> select sum(sal) from EMP where deptno=10;

  SUM(SAL)
-----
      8750

23:18:53 SID=12> update EMP set sal=sal+250 where ename='CLARK';
1 row updated.
```

```
23:18:46 SID=365> select sum(sal) from EMP where deptno=10;

  SUM(SAL)
-----
      9000

23:19:04 SID=365> commit;
Commit complete.
```

```
23:18:55 SID=12> select sum(sal) from EMP where deptno=10;

  SUM(SAL)
-----
      9000

23:19:08 SID=12> commit;
Commit complete.
```

Zusammengeführte Daten nach dem COMMIT:

```
23:19:09 SID=12> select sum(sal) from EMP where deptno=10;

  SUM(SAL)
-----
      9250
```