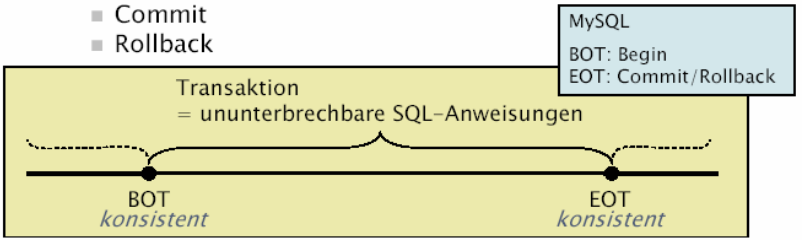
Atomarität einer Transaktion Konsistenz und Dauerhaftigkeit der DB

- *Wer soll sich um die Atomarität von Transaktionen sowie die Konsistenz und die Dauerhaftigkeit der Datenbank kümmern?*
  - ***Sie als Anwender*** eines DBS sollen beim Verwenden von SQL-Anweisungen darauf achten, dass eine Transaktion atomar bleibt, und die Markierungen (BOT/EOT) entsprechend setzen.
    - BOT: Begin of Transaction (bei Oracle nur implizit)
      - Erstellen der Verbindung zu einer Datenbank
      - Die erste ausführbare SQL-Anweisung nach EOT
    - EOT: End of Transaction (bei Oracle explizit und implizit)
      - Commit
      - Rollback



Transaktionsverwaltung

- *Transaktion*
  - Logische Verarbeitungseinheit von DB-Aktionen, d.h. eine Serie von SQL-Anweisungen, die als eine Einheit betrachtet werden.
- *Ziel der Transaktionsverwaltung*
  - Durch die von einer Transaktion ausgeführten Operationen wird die DB von einem konsistenten Zustand in einen anderen überführt.
- *Eigenschaften (Forderung) der Transaktion (ACID)*
  - **A**tomicity — Transaktion soll atomar (nicht zerlegbar) sein. (alles oder nichts).
  - **C**onsistency — Transaktion soll eine DB von einem konsistenten Zustand zum nächsten führen. (Während der Transaktion sind Inkonsistenzen erlaubt.)
  - **I**solation — Parallel ausgeführte Transaktionen dürfen sich nicht beeinflussen. (Jede hat die DB für sich allein.)
  - **D**urability — Änderungen erfolgreicher Transaktionen müssen äußere Einflüsse überleben.

Transaktionssteuerung in Oracle: Syntax von COMMIT und ROLLBACK

- **COMMIT**
  - bedeutet „Transaktion wird erfolgreich beendet."
  - Macht alle Änderungen innerhalb dieser Transaktion persistent.
  - gibt alle von der Transaktion gesperrten Ressourcen frei.
    - Achtung!
      Vor und nach jedem DDL-Satz (create, alter, drop) führt Oracle ein implizites Commit aus.
- **ROLLBACK**
  - bedeutet „Transaktion wird erfolglos beendet (abgebrochen)."
  - setzt alle Änderungen innerhalb dieser Transaktion zurück.
  - gibt alle von der Transaktion gesperrten Ressourcen frei.
    - Bei Abbruch der Verbindung, Netzstörung, Systemfehler bzw. Programmabsturz führt Oracle ein implizites Rollback aus.

Transaktionssteuerung in Oracle: Syntax der SAVEPOINT-Anweisung

- **SAVEPOINT** *savepoint_name*
  - setzt einen Punkt innerhalb einer Transaktion, bis zu dem ein Rollback durchgeführt werden kann.
- **ROLLBACK TO** *savepoint_name*
  - Setzt alle Änderungen, die seit dem Savepoint *savepoint_name* durchgeführt worden sind, zurück (Teilweises Zurücksetzen einer Transaktion)
    - Achtung
      - Das Rollback zu einem Savepoint (das teilweise Zurücksetzen) beinhaltet kein Commit der Transaktion.
      - Es können mehrere Savepoints innerhalb einer Transaktion gesetzt werden.

Transaktionssteuerung in Oracle (Forts.)

- *Beispiel*

```
ALTER TABLE prod ADD verkauf NUMBER(6);
COMMIT;
INSERT INTO auf (anr, knr, adat)
VALUES (780, 1, TO_DATE(2003-11-14, 'YYYY-MM-DD'));
INSERT INTO pos (anr, pnr, menge, vkp)
VALUES (780, 40, 20, 999);
UPDATE prod SET verkauf = verkauf + 20
WHERE pnr = 40;
COMMIT/ROLLBACK;
```

Systemabsturz

Im Fall Oracle nicht notwendig. (Weil sowieso ein implizites COMMIT durchgeführt wird.)

Transaktion

Im Fall eines Systemabsturzes wird der Datenbankzustand (vom DBS) zurückgesetzt.

Unabhängig davon, ob Sie interaktiv arbeiten oder eine Applikation SQL-Anweisungen schicken lassen, sollen Sie hier entweder „COMMIT“ oder „ROLLBACK“ setzen und damit die Transaktion markieren.

Transaktionssteuerung in Oracle (Forts.)

- *Beispiel*

```
COMMIT;
INSERT INTO auf (anr, knr, adat)
VALUES (101, 1, TO_DATE(2003-11-15, 'YYYY-MM-DD'));
(Fehlermeldung, dass die Auftragsnummer 101 bereits existiert und diese Eingabe die Integritätsbedingungen verletzt.)
INSERT INTO pos (anr, pnr, menge, vkp)
VALUES (101, 30, 10, 800);
UPDATE prod SET verkauf = verkauf + 10
WHERE pnr = 30;
COMMIT;
```

ROLLBACK;

1

2

Transaktion

Sie sollen dafür sorgen, dass die Applikation im Fall eines Fehlers die Fehlermeldung abfängt und an Position „1“ bis Position „2“ ein ROLLBACK setzt, damit die Transaktion zurückgesetzt wird.

Sperrverfahren in Oracle

- *Manuelle und Automatische Sperre*
  - Objekte können manuell gesperrt werden. (LOCK TABLE)
  - Bei einigen SQL-Anweisungen werden die betreffenden Objekte automatisch gesperrt.
    - DML-Sperren
    - DDL-Sperren
- *Gültigkeit einer Sperre*
  - Eine manuelle Sperre wird aufgehoben:
    - bei einem impliziten bzw. expliziten COMMIT
    - bei einem impliziten bzw. expliziten ROLLBACK
    - bei einem ROLLBACK TO *Savepoint* vor der Sperre-Anweisung.
  - Eine automatische Sperre wird mit dem Ausführung der betreffenden SQL-Anweisung aufgehoben.
- *Voraussetzung vor dem Sperren*
  - Ein Benutzer kann nur eigene Tabellen/Views oder solche, worauf er Privilegien hat, sperren.

Sperrverfahren: Syntax der LOCK TABLE Anweisung

- **LOCK TABLE** *[schema.]table_name|view_name*
  IN *lockmode* **MODE** [**NOWAIT**]
- lockmode
  - *Wann verwendet man den Modus?*
  - *Was dürfen andere während der Sperre machen?*
  - **SHARE** [Gut bei Queries. Lesekonsistenz garantiert.]
    - Andere Transaktionen können die Tabelle lesen.
    - Andere können sie im Share-Modus sperren. (share, row share)
    - Andere können sie im Exclusive-Modus nicht sperren.
    - Andere können sie nicht ändern.
  - **EXCLUSIVE** [Mächtig bei allen Arten Änderungen.]
    - Andere Transaktionen können die Tabelle lesen.
    - Andere können sie im Share-Modus nicht sperren.
    - Andere können sie im Exclusive-Modus nicht sperren.
    - Andere können sie nicht ändern.

Sperrverfahren: Syntax der LOCK TABLE Anweisung (Forts.)

- lockmode
  - `ROW SHARE` [Vorsorglich vor Zeilenänderung.]
    - Andere Transaktionen können die Tabelle lesen.
    - Andere können sie im Share-Modus sperren.
    - Andere können sie im Exclusive-Modus nicht sperren.
    - Andere können einzelne Zeilen im Exclusive-Modus sperren und ändern.
  - `ROW EXCLUSIVE` [Bei einer Zeilenänderung]
    - Andere Transaktionen können die Tabelle lesen.
    - Andere können sie im Share-Modus nicht sperren.
    - Andere können sie im Exclusive-Modus nicht sperren.
    - Andere können einzelne Zeile im Exclusive-Modus sperren und ändern.
      - Beim UPDATE, INSERT, DELETE wird die Tabelle automatisch im Row-Exclusive-Modus gesperrt.
      - Row Exclusive kann parallel gewährt werden.

Sperrverfahren: Syntax der LOCK TABLE Anweisung (Forts.)

- lockmode
  - `SHARE ROW EXCLUSIVE` [Bei einer Zeilenänderung]
    - Andere Transaktionen können die Tabelle lesen.
    - Andere können sie im Share-Modus nicht sperren.
    - Andere können sie im Exclusive-Modus nicht sperren.
    - Andere können einzelne Zeilen im Exclusive-Modus nicht sperren und ändern.
      - Share Row Exclusive kann nicht parallel gewährt werden.
- NOWAIT – Option
  - Ohne NOWAIT-Option wartet Oracle auf die Freigabe des spezifizierten Objektes.
  - Mit der NOWAIT-Option wartet Oracle nicht auf die Freigabe von gesperrten Objekten, sondern gibt sofort mit einer Fehlermeldung auf, die LOCK TABLE-Anweisung durchzuführen.

Verträglichkeit verschiedener Sperrenmodi

T1 \ T2	ROW SHARE	ROW EXCLUSIVE	SHARE	SHARE ROW EXCLUSIVE	EXCLUSIVE
ROW SHARE	zulässig	zulässig	zulässig	zulässig	–
ROW EXCLUSIVE	zulässig	zulässig	–	–	–
SHARE	zulässig	–	zulässig	–	–
SHARE ROW EXCLUSIVE	zulässig	–	–	–	–
EXCLUSIVE	–	–	–	–	–

Automatische Sperre

- *DML-Sperren*
  - Bei einer DML-Anweisung (insert, update, delete, select ... for update, lock table) wird die Tabelle so gesperrt, dass keine DDL-Anweisung (alter, drop ...), die zu Konflikten mit der Transaktion führen würde, möglich ist.
  - Bei einer DML-Anweisung (insert, update, delete, select ... for update) werden die betreffenden Zeilen so gesperrt, dass keiner sie modifizieren kann. (Row Exclusive Sperre)
- *DDL-Sperren*
  - Während ein Schemaobjekt von einer DDL-Anweisung bearbeitet bzw. referenziert wird, wird es so gesperrt, dass keine andere DDL-Anweisung es modifizieren bzw. referenzieren kann.
- *SELECT ... FOR UPDATE*
  - SELECT-Anweisung mit FOR UPDATE-Option sperrt die ausgewählten Zeilen exklusiv (wie eine UPDATE-Anweisung aber ohne die Daten zu ändern). Änderungen der Zeilen können danach erfolgen.

Beispiel LOCK TABLE

- *Lesekonsistenz auf Transaktionsebene (transaction-level read consistency) kann folgendermaßen erreicht werden.*

T1	T2
LOCK TABLE konten IN SHARE MODE;	INSERT impliziert Anforderung zu Row-Exclusive-Sperre.
SELECT * FROM konten;	
	INSERT INTO konten VALUES (...);
SELECT * FROM konten;	
COMMIT; Sperrfreigabe	Warten auf Sperrfreigabe
	Ausführung von INSERT.
	COMMIT;

Beispiel LOCK TABLE (Forts)

- *Eine Exclusive-Sperre wartet auf ihre Durchführung bis keine andere Sperre mehr vorhanden ist.*

Weil es eine Share-Sperre gibt, kann eine Exclusive-Sperre nicht durchgeführt werden.

T1	T2
LOCK TABLE konten IN SHARE MODE;	LOCK TABLE konten IN SHARE MODE;
	LOCK TABLE konten IN EXCLUSIVE MODE; Warten auf Sperrfreigabe
COMMIT; Sperrfreigabe	
	Sperre wird hier gewährt.

Deadlock

- *Wenn zwei oder mehr Benutzer auf Daten warten, die sie gegenseitig gesperrt haben, müssen sie ewig warten. (Dead Lock)*
- *Oracle erkennt den Deadlock automatisch und löst ihn durch Zurückrollen der Anweisung.*