



**REPEATABLE READ:** Einzelne Zugriffe ROWS sind synchronisiert, Read and Write Lock für die gesamte T

**SERIALIZABLE:** Vollständige Isolation nach ACID

	Read Un-committed	Read Committed	Repeatability Read	Serializable
Dirty Write	*	*	*	X
Dirty Read	✓	X	X	X
Lost Update	✓	✓	X	X
Fuzzy Read	✓	✓	X	X
Phantom Read	✓	✓	✓	X
Read Skew	✓	✓	X	X
Write Skew	✓	✓	✓	*

\* Nur in SQL92 möglich, PSQL >= 9.1 verhindert dies

**Dirty Read:** Lese Daten von nicht committed T's

**Fuzzy Read:** Versch. Werte beim mehrmaligen Lesen gleicher Daten (da durch andere T geändert)

**Phantom Read:** Neue/Gelöschte Rows einer anderen T

**Read Skew:** Daten lesen, die sich während der T ändern

**Write Skew:** Mehrere T lesen Daten und Ändern sie

**Deadlock:** Mehrere T blockieren sich, da sie auf die gleiche Ressource warten

**Cascading Rollback:** T schlägt fehl und alle davon abhängigen T müssen ebenfalls zurückgerollt werden

	Serialisierbar	Deadlocks	Cas-cading RollB.	Konflikt-RollB.	Hohe Par-allelität	Realisti-sch
Two-Phase Locking	✓	✓	✓	X	X	X
Strict 2PL	✓	✓	X	X	X	✓
Precalim-ing 2PL	✓	X	X	X	X	X
Validation-based	✓	X	✓	✓	✓	✓
Timestamp-based	X	X	✓	✓	✓	✓
Snapshot Isolation	X	*	X	✓	✓	✓
SSI	✓	*	X	✓	✓	✓

\* Deadlock in PSQL mit Snapshot Isolation

### SQL Beispiele

```
BEGIN;
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
UPDATE accounts SET balance = balance - 100.00
    WHERE name = 'Alice';
SAVEPOINT my_savepoint;
UPDATE accounts SET balance = balance + 100.00
    WHERE name = 'Bob';
ROLLBACK TO my_savepoint;
UPDATE accounts SET balance = balance + 100.00
    WHERE name = 'Wally';
COMMIT;
```

### Relationale Algebra

$\pi_{R1,R4}(R)$  SELECT R1,R4 FROM R; (Projektion)

$\sigma_{R1>30}(R)$  SELECT \* FROM R WHERE R1 > 30; (Selektion)

$\rho_{a \leftarrow R}$  SELECT \* FROM R AS a; (Umbenennung/Alias)

$R \times S$  SELECT \* FROM R,S; (Kartesisches Produkt)

$R \bowtie_{A=B} S$  SELECT \* FROM R JOIN S ON R.A=S.B; (Verbund)

### Serialisierbarkeit

**Shared Lock:** Schreib- & Lesezugriffe (eine Transaktion)

**Exclusive Lock:** Lesezugriffe (mehrere Transaktionen)

**Serieller Schedule:** Führt Transaktionen am Stück aus

**Nicht serialisierbar:**

$S1=R1(x)R2(x)W1(x)R1(y)W2(x)W1(y)$



-- Welche Züchter haben in ihren Ställen mindestens 1 Kind von dem Vater mit Namen "Hermes"

### Konfliktpaare:

$R1(x) < W2(x)$        $R2(x) < W1(x)$

### Konflikt-Serialisierbar:

$r1(b)r2(b)w2(b)r2(c)r2(d)w3(a)r4(d)r3(b)w4(d)r5(c)r5(a)w4(c)$

### Konflikt-Äquivalenter serieller Schedule:

$r1(b)r2(b)w2(b)r2(c)r2(d)w3(a)r3(b)r5(c)r5(a)r4(d)w4(d)w4(c)$



### Vollständiges Backup

Exakte Kopie der ganzen DB

### Inkrementelles Backup

Sichert nur die seit dem letzten Backup geänderten Daten.

### Logisches Backup (SQL Dump)

Blockiert keine T. Für mittelgroße Datenmengen, interkompatibel mit neuen PG-Versionen und anderen Maschinen.

### Physisches Backup (File System)

Datenbank muss gestoppt werden, schneller als logisches Backup, passt nur zu derselben «Major Version» von PG.

### Multi-Version Concurrency Control (MVCC)

Ermöglicht es, mehreren T gleichzeitig zu laufen. Bei jeder Änderung wird eine neue Version der Daten erstellt. Leser sehen die älteren Versionen, während Schreiber die neuesten Versionen sehen.

### Two-Phase Locking (2PL)

**TODO: example** Stellt Isolation der T sicher

- 1) Growing Phase: Die T. kann neue Locks erwerben, jedoch keine freigeben
- 2) Shrinking Phase: Locks können freigegeben werden, aber keine neuen mehr erworben werden

### Write-Ahead Log (WAL)

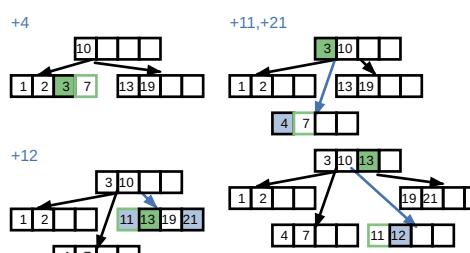
Schreibt Änderungen der T in Log, dann Commit loggen, dann Updates in DB. Kann bei Absturz replayed werden

### LSN, Taid, PageID, Redo, Undo, PrevLSN

### Dreiwertige Logik (cursed)

```
SELECT NULL IS NULL; -- true
SELECT NULL = NULL; -- [null]
```

### B-Baum



### SQL Beispiele

```
CREATE TABLE pferde (
    pnr SERIAL PRIMARY KEY,
```

```
    name TEXT,
```

```
    alter INT,
```

```
    zuechternr INT REFERENCES stall.pk,
```

```
    vaternr INT REFERENCES pferd.pk
);
```

```
CREATE TABLE stall (
    zuechternr SERIAL PRIMARY KEY,
```

```
    name TEXT,
```

```
    plz INT,
```

```
    ort TEXT,
```

```
    strasse TEXT
);
```