Transações, Concorrência e Recuperação em SGBDs parte 2

1) TRANSFERÊNCIA ATÔMICA COM COMMIT/ROLLBACK (MySQL/InnoDB)
Crie a tabela contas(id INT PK, saldo DECIMAL(10,2)) e insira duas linhas. Faça START TRANSACTION; UPDATE para debitar R$200 de A e creditar em B; verifique com SELECT dentro e fora da transação (em outra sessão). Finalize com COMMIT e repita depois fazendo ROLLBACK. Observe os saldos.

2) DEADLOCK NA PRÁTICA COM FOR UPDATE
Sessão 1: START TRANSACTION; SELECT * FROM contas WHERE id=1 FOR UPDATE; Sessão 2: START TRANSACTION; SELECT * FROM contas WHERE id=2 FOR UPDATE; Sessão 1: SELECT * FROM contas WHERE id=2 FOR UPDATE; Sessão 2: SELECT * FROM contas WHERE id=1 FOR UPDATE; Explique o deadlock detectado e o rollback automático de uma das sessões.

3) ISOLATION LEVEL – NON-REPEATABLE READ (READ COMMITTED)
Sessão 1: SET TRANSACTION ISOLATION LEVEL READ COMMITTED; START TRANSACTION; SELECT saldo FROM contas WHERE id=1; Sessão 2: UPDATE contas SET saldo = saldo + 10 WHERE id=1; COMMIT; Sessão 1: SELECT saldo FROM contas WHERE id=1; Mostre que o valor mudou dentro da mesma transação (não repetível).

4) PHANTOM READ COM READ COMMITTED
Crie tabela pedidos(id INT PK, valor DECIMAL(10,2), status ENUM('P','F')). Sessão 1: SET TRANSACTION ISOLATION LEVEL READ COMMITTED; START TRANSACTION; SELECT COUNT(*) FROM pedidos WHERE status='P'; Sessão 2: INSERT INTO pedidos VALUES(...,'P'); COMMIT; Sessão 1: repita o COUNT; explique o 'fantasma'.

5) SAVEPOINT E ROLLBACK PARCIAL
START TRANSACTION; UPDATE contas SET saldo=saldo-50 WHERE id=1; SAVEPOINT s1; UPDATE contas SET saldo=saldo-100 WHERE id=1; ROLLBACK TO s1; Verifique que apenas o primeiro UPDATE permaneceu. COMMIT.

6) FOR UPDATE vs FOR SHARE
Mostre, em duas sessões, a diferença entre SELECT ... FOR UPDATE (bloqueio exclusivo) e SELECT ... FOR SHARE (bloqueio compartilhado). Demonstre que FOR SHARE permite múltiplas leituras concorrentes, mas impede updates.

7) FILA COM SKIP LOCKED / NOWAIT (MySQL 8)
Crie tabela fila(id INT PK, status ENUM('N','P','C'), payload JSON). Sessão 1: START TRANSACTION; SELECT id FROM fila WHERE status='N' ORDER BY id LIMIT 1 FOR UPDATE SKIP LOCKED; Atualize para status='P'; Sessão 2: tente pegar o próximo item com SKIP LOCKED (não bloqueia). Repita com NOWAIT e explique o erro imediato de lock.

8) DIAGNÓSTICO DE DEADLOCK (SHOW ENGINE INNODB STATUS)
Provoque um deadlock (exercício 2) e, em seguida, execute: SHOW ENGINE INNODB STATUS\G Copie o trecho 'LATEST DETECTED DEADLOCK' e identifique as transações e locks.

9) AUTOCOMMIT, TRANSAÇÕES IMPLÍCITAS E DURABILIDADE
SET autocommit=0; START TRANSACTION; UPDATE contas SET saldo=saldo+1 WHERE id=2; Em outra sessão leia o saldo (não deve ver a mudança). Faça COMMIT e confirme a visibilidade. Explique o comportamento se ocorrer ROLLBACK em vez de COMMIT.

10) CONSISTENT SNAPSHOT (MVCC) E GAP LOCKS
Crie tabela produtos(id INT PK, preco DECIMAL(10,2)); insira várias linhas. Sessão 1: START TRANSACTION WITH CONSISTENT SNAPSHOT; SELECT AVG(preco) FROM produtos WHERE id BETWEEN 10 AND 100; Sessão 2: insira/atualize linhas nesse intervalo e faça COMMIT. Sessão 1: repita a mesma SELECT (sem locking read) e comente a leitura consistente. Depois repita usando SELECT ... FOR UPDATE para observar locks (next-key/gap).