1. What does ACID stand for, and what does each part mean? (2 pt)

ACID is an database-related acronym standing for Atomicity, Consistency, Isolation, and Durability. *Atomicity* is the property of a database transaction relating to its ability to abort and retry a transaction in the case of failure on writing. *Consistency* (in this context) is essentially constraining which writes are valid, though this is generally handled at an application level when designing business rules instead of at the database level. *Isolation* is the database property pertaining to how concurrent writes are handled, though the commonality between all isolation implementations is that writes cannot interfere with one another. *Durability* is the property of a database to store its contents such that a crash or hardware issue will not erase the data. This could take the form of writing to disk, replication, cloud storage, or a mixture of all three.

1. What are dirty reads and dirty writes? (2 pt)

Dirty reads are database read operations performed on uncommitted transactions. Most databases disallow this, which is endemic to a read committed isolation level. Dirty writes are write operations performed on a database that disallow overwriting a value that is currently beholden to another write transaction.

1. What is read skew and write skew? (2 pt)

Read skew is a type of nonrepeatable read that arises from latency in what a user might read from a database soon after a write (whatever soon means in terms of a write being committed and a read being served to a user). Snapshot isolation is the a common solution to read skew. Write skew is an anomaly that arises from two concurrent writes on different objects violating a given constraint (generally a business rule that isn’t coded into the database). Write skew can come up when two updates are attempted and each validates the business rule before committing a write that would otherwise invalidate the attempted write.

1. What is serializable isolation? (2 pt)

Serializable isolation is a strong transactional isolation level that has a number of different implementations. The common thread among them is that even if concurrent transactions may be processed in a parallel fashion, that they are treated as though they were done consecutively (*serially*). The three techniques for providing serialiability that Kleppman mentions are *Actual Serial* *Execution, Two-Phase Locking,* and *Serializable Snapshot Isolation*.

1. What is Two-Phase Locking and how is it used? (2 pt)

Two phase locking is a prevalent method of providing database serializability that uses a two-way lock on objects in a database. A write operation blocks all reads until the transaction is either committed or aborted for any reason, and any read blocks all write operations until the read operation is either committed or aborted for any reason. This is contrast to snapshot isolation, where write transactions never block read transactions, and reads never block writes.

Two-phase locking is accomplished by read transactions acquiring a shared lock, which blocks write transactions, though write transactions require an exclusive lock (to not only block reads, but other writes as well).