****a. Joins and Denormalization:**** Performing joins on a database which is running on one server is straightforward, but once a database is partitioned and spread across multiple machines it is often not feasible to perform joins that span database shards. Such joins will not be performance efficient since data has to be compiled from multiple servers. A common workaround for this problem is to denormalize the database so that queries that previously required joins can be performed from a single table. Of course, the service now has to deal with all the perils of denormalization such as data inconsistency.

****b. Referential integrity:**** As we saw that performing a cross-shard query on a partitioned database is not feasible, similarly trying to enforce data integrity constraints such as foreign keys in a sharded database can be extremely difficult.

Most of RDBMS do not support foreign keys constraints across databases on different database servers. Which means that applications that require referential integrity on sharded databases often have to enforce it in application code. Often in such cases, applications have to run regular SQL jobs to clean up dangling references.

****c. Rebalancing:**** There could be many reasons we have to change our sharding scheme:

1. The data distribution is not uniform, e.g., there are a lot of places for a particular ZIP code, that cannot fit into one database partition.
2. There are a lot of load on a shard, e.g., there are too many requests being handled by the DB shard dedicated to user photos.

In such cases, either we have to create more DB shards or have to rebalance existing shards, which means the partitioning scheme changed and all existing data moved to new locations. Doing this without incurring downtime is extremely difficult. Using a scheme like directory based partitioning does make rebalancing a more palatable experience at the cost of increasing the complexity of the system and creating a new single point of failure (i.e. the lookup service/database).