```
diff --git a/src/include/storage/sql_table.h b/src/include/storage/sql_table.h
index d55e9a53b..1b5cb2507 100644
--- a/src/include/storage/sql_table.h
+++ b/src/include/storage/sql_table.h
@@ -75,12 +75,13 @@ class SqlTable {
    * Inserts a tuple, as given in the redo, and return the slot allocated for the tuple.
    *
    * @param txn the calling transaction
-   * @param redo after-image of the inserted tuple. The TupleSlot in this RedoRecord will be set to the inserted
-   * location.
+   * @param redo after-image of the inserted tuple.
+   * @return TupleSlot for the inserted tuple
    */
-  void Insert(transaction::TransactionContext *const txn, RedoRecord *const redo) const {
+  TupleSlot Insert(transaction::TransactionContext *const txn, RedoRecord *const redo) const {
     const auto slot = table_.data_table->Insert(txn, *(redo->Delta()));
     redo->SetTupleSlot(slot);
+    return slot;
   }

   /**
diff --git a/src/include/transaction/transaction_context.h b/src/include/transaction/transaction_context.h
index 910f44443..be1502209 100644
--- a/src/include/transaction/transaction_context.h
+++ b/src/include/transaction/transaction_context.h
@@ -96,11 +96,16 @@ class TransactionContext {

   /**
    * Expose a record that can hold a change, described by the initializer given, that will be logged out to disk.
-   * The change can either be copied into this space, or written in the space and then used to change the DataTable.
+   * The change should be written in the space and then used to change the SqlTable.
    * @param db_oid the database oid that this record changes
    * @param table_oid the table oid that this record changes
    * @param initializer the initializer to use for the underlying record
    * @return pointer to the initialized redo record.
+   * @warning RedoRecords returned by StageWrite are not guaranteed to remain valid forever. If you call StageWrite
+   * again, the previous RedoRecord's buffer may be swapped out, written to disk, and handed back out to another
```

```
+   * transaction.
+   * @warning If you call StageWrite, its contents WILL be logged to disk. If you StageWrite anything that you didn't
+   * succeed in writing into the table or decide you don't want to use, the transaction MUST abort.
    */
   storage::RedoRecord *StageWrite(const catalog::db_oid_t db_oid, const catalog::table_oid_t table_oid,
                                   const storage::ProjectedRowInitializer &initializer) {
diff --git a/test/include/util/tpcc/loader.h b/test/include/util/tpcc/loader.h
index 68f3f71a6..3a1062ba9 100644
--- a/test/include/util/tpcc/loader.h
+++ b/test/include/util/tpcc/loader.h
@@ -142,10 +142,9 @@ struct Loader {
         auto *const item_redo = txn->StageWrite(db->db_oid_, db->item_table_oid_, item_tuple_pr_initializer);
         BuildItemTuple(i_id + 1, item_original[i_id], item_redo->Delta(), item_tuple_pr_map, db->item_schema_,
                        generator);
-        db->item_table_->Insert(txn, item_redo);
+        const auto item_slot = db->item_table_->Insert(txn, item_redo);

         // insert in index
-        const auto item_slot = item_redo->GetTupleSlot();
         const auto *const item_key = BuildItemKey(i_id + 1, worker->item_key_buffer, item_key_pr_initializer,
                                                   item_key_pr_map, db->item_primary_index_schema_);
         bool UNUSED_ATTRIBUTE index_insert_result = db->item_primary_index_->InsertUnique(txn, *item_key, item_slot);
@@ -160,10 +159,9 @@ struct Loader {
         txn->StageWrite(db->db_oid_, db->warehouse_table_oid_, warehouse_tuple_pr_initializer);
       BuildWarehouseTuple(static_cast<int8_t>(w_id + 1), warehouse_redo->Delta(), warehouse_tuple_pr_map,
                           db->warehouse_schema_, generator);
-      db->warehouse_table_->Insert(txn, warehouse_redo);
+      const auto warehouse_slot = db->warehouse_table_->Insert(txn, warehouse_redo);

       // insert in index
-      const auto warehouse_slot = warehouse_redo->GetTupleSlot();
       const auto *const warehouse_key =
           BuildWarehouseKey(static_cast<int8_t>(w_id + 1), worker->warehouse_key_buffer, warehouse_key_pr_initializer,
```

```
                    warehouse_key_pr_map, db->warehouse_primary_index_schema_);
@@ -188,10 +186,9 @@ struct Loader {
         auto *const stock_redo = txn->StageWrite(db->db_oid_, db->stock_table_oid_,
stock_tuple_pr_initializer);
         BuildStockTuple(s_i_id + 1, static_cast<int8_t>(w_id + 1), stock_original[s_i_id],
stock_redo->Delta(),
                         stock_tuple_pr_map, db->stock_schema_, generator);
-        db->stock_table_->Insert(txn, stock_redo);
+        const auto stock_slot = db->stock_table_->Insert(txn, stock_redo);

         // insert in index
-        const auto stock_slot = stock_redo->GetTupleSlot();
         const auto *const stock_key =
             BuildStockKey(s_i_id + 1, static_cast<int8_t>(w_id + 1), worker->stock_key_buffer,
                           stock_key_pr_initializer, stock_key_pr_map,
db->stock_primary_index_schema_);
@@ -209,10 +206,9 @@ struct Loader {
           txn->StageWrite(db->db_oid_, db->district_table_oid_, district_tuple_pr_initializer);
       BuildDistrictTuple(static_cast<int8_t>(d_id + 1), static_cast<int8_t>(w_id + 1),
district_redo->Delta(),
                          district_tuple_pr_map, db->district_schema_, generator);
-      db->district_table_->Insert(txn, district_redo);
+      const auto district_slot = db->district_table_->Insert(txn, district_redo);

       // insert in index
-      const auto district_slot = district_redo->GetTupleSlot();
       const auto *const district_key =
           BuildDistrictKey(static_cast<int8_t>(d_id + 1), static_cast<int8_t>(w_id + 1),
worker->district_key_buffer,
                            district_key_pr_initializer, district_key_pr_map,
db->district_primary_index_schema_);
@@ -241,10 +237,9 @@ struct Loader {
           txn->StageWrite(db->db_oid_, db->customer_table_oid_,
customer_tuple_pr_initializer);
         BuildCustomerTuple(c_id + 1, static_cast<int8_t>(d_id + 1), static_cast<int8_t>(w_id +
1), c_credit[c_id],
                            customer_redo->Delta(), customer_tuple_pr_map, db->customer_schema_,
generator);
-        db->customer_table_->Insert(txn, customer_redo);
+        const auto customer_slot = db->customer_table_->Insert(txn, customer_redo);

         // insert in index
-        const auto customer_slot = customer_redo->GetTupleSlot();
```

```
        const auto *const customer_key = BuildCustomerKey(
            c_id + 1, static_cast<int8_t>(d_id + 1), static_cast<int8_t>(w_id + 1),
worker->customer_key_buffer,
            customer_key_pr_initializer, customer_key_pr_map,
db->customer_primary_index_schema_);
@@ -292,10 +287,9 @@ struct Loader {
        const auto order_results =
            BuildOrderTuple(o_id + 1, o_c_ids[c_id], static_cast<int8_t>(d_id + 1),
static_cast<int8_t>(w_id + 1),
                            order_redo->Delta(), order_tuple_pr_map, db->order_schema_, generator);
-       db->order_table_->Insert(txn, order_redo);
+        const auto order_slot = db->order_table_->Insert(txn, order_redo);

        // insert in index
-       const auto order_slot = order_redo->GetTupleSlot();
        const auto *const order_key = BuildOrderKey(
            o_id + 1, static_cast<int8_t>(d_id + 1), static_cast<int8_t>(w_id + 1),
worker->order_key_buffer,
            order_key_pr_initializer, order_key_pr_map, db->order_primary_index_schema_);
@@ -320,10 +314,9 @@ struct Loader {
            BuildOrderLineTuple(o_id + 1, static_cast<int8_t>(d_id + 1), static_cast<int8_t>(w_id +
1),
                                static_cast<int8_t>(ol_number + 1), order_results.o_entry_d,
order_line_redo->Delta(),
                                order_line_tuple_pr_map, db->order_line_schema_, generator);
-           db->order_line_table_->Insert(txn, order_line_redo);
+           const auto order_line_slot = db->order_line_table_->Insert(txn, order_line_redo);

            // insert in index
-           const auto order_line_slot = order_line_redo->GetTupleSlot();
            const auto *const order_line_key = BuildOrderLineKey(
                o_id + 1, static_cast<int8_t>(d_id + 1), static_cast<int8_t>(w_id + 1),
                static_cast<int8_t>(ol_number + 1), worker->order_line_key_buffer,
order_line_key_pr_initializer,
@@ -341,10 +334,9 @@ struct Loader {
                txn->StageWrite(db->db_oid_, db->new_order_table_oid_,
new_order_tuple_pr_initializer);
            BuildNewOrderTuple(o_id + 1, static_cast<int8_t>(d_id + 1), static_cast<int8_t>(w_id +
1),
                               new_order_redo->Delta(), new_order_tuple_pr_map,
db->new_order_schema_);
-           db->new_order_table_->Insert(txn, new_order_redo);
+           const auto new_order_slot = db->new_order_table_->Insert(txn, new_order_redo);
```

```
            // insert in index
-           const auto new_order_slot = new_order_redo->GetTupleSlot();
            const auto *const new_order_key = BuildNewOrderKey(
                o_id + 1, static_cast<int8_t>(d_id + 1), static_cast<int8_t>(w_id + 1),
worker->new_order_key_buffer,
                new_order_key_pr_initializer, new_order_key_pr_map,
db->new_order_primary_index_schema_);
diff --git a/test/storage/bwtree_index_test.cpp b/test/storage/bwtree_index_test.cpp
index 498d57283..dca7ce6c9 100644
--- a/test/storage/bwtree_index_test.cpp
+++ b/test/storage/bwtree_index_test.cpp
@@ -107,8 +107,7 @@ TEST_F(BwTreeIndexTests, UniqueInsert) {
            insert_txn->StageWrite(CatalogTestUtil::test_db_oid, CatalogTestUtil::test_table_oid,
tuple_initializer_);
        auto *const insert_tuple = insert_redo->Delta();
        *reinterpret_cast<int32_t *>(insert_tuple->AccessForceNotNull(0)) = i;
-       sql_table_->Insert(insert_txn, insert_redo);
-       const auto tuple_slot = insert_redo->GetTupleSlot();
+       const auto tuple_slot = sql_table_->Insert(insert_txn, insert_redo);

        *reinterpret_cast<int32_t *>(insert_key->AccessForceNotNull(0)) = i;
        if (unique_index_->InsertUnique(insert_txn, *insert_key, tuple_slot)) {
@@ -125,8 +124,7 @@ TEST_F(BwTreeIndexTests, UniqueInsert) {
            insert_txn->StageWrite(CatalogTestUtil::test_db_oid, CatalogTestUtil::test_table_oid,
tuple_initializer_);
        auto *const insert_tuple = insert_redo->Delta();
        *reinterpret_cast<int32_t *>(insert_tuple->AccessForceNotNull(0)) = i;
-       sql_table_->Insert(insert_txn, insert_redo);
-       const auto tuple_slot = insert_redo->GetTupleSlot();
+       const auto tuple_slot = sql_table_->Insert(insert_txn, insert_redo);

        *reinterpret_cast<int32_t *>(insert_key->AccessForceNotNull(0)) = i;
        if (unique_index_->InsertUnique(insert_txn, *insert_key, tuple_slot)) {
@@ -183,8 +181,7 @@ TEST_F(BwTreeIndexTests, DefaultInsert) {
            insert_txn->StageWrite(CatalogTestUtil::test_db_oid, CatalogTestUtil::test_table_oid,
tuple_initializer_);
        auto *const insert_tuple = insert_redo->Delta();
        *reinterpret_cast<int32_t *>(insert_tuple->AccessForceNotNull(0)) = i;
-       sql_table_->Insert(insert_txn, insert_redo);
-       const auto tuple_slot = insert_redo->GetTupleSlot();
+       const auto tuple_slot = sql_table_->Insert(insert_txn, insert_redo);
```

```
        *reinterpret_cast<int32_t *>(insert_key->AccessForceNotNull(0)) = i;
        EXPECT_TRUE(default_index_->Insert(insert_txn, *insert_key, tuple_slot));
@@ -197,8 +194,7 @@ TEST_F(BwTreeIndexTests, DefaultInsert) {
           insert_txn->StageWrite(CatalogTestUtil::test_db_oid, CatalogTestUtil::test_table_oid,
tuple_initializer_);
        auto *const insert_tuple = insert_redo->Delta();
        *reinterpret_cast<int32_t *>(insert_tuple->AccessForceNotNull(0)) = i;
-       sql_table_->Insert(insert_txn, insert_redo);
-       const auto tuple_slot = insert_redo->GetTupleSlot();
+       const auto tuple_slot = sql_table_->Insert(insert_txn, insert_redo);

        *reinterpret_cast<int32_t *>(insert_key->AccessForceNotNull(0)) = i;
        EXPECT_TRUE(default_index_->Insert(insert_txn, *insert_key, tuple_slot));
@@ -246,8 +242,7 @@ TEST_F(BwTreeIndexTests, ScanAscending) {
        insert_txn->StageWrite(CatalogTestUtil::test_db_oid, CatalogTestUtil::test_table_oid,
tuple_initializer_);
    auto *const insert_tuple = insert_redo->Delta();
    *reinterpret_cast<int32_t *>(insert_tuple->AccessForceNotNull(0)) = i;
-   sql_table_->Insert(insert_txn, insert_redo);
-   const auto tuple_slot = insert_redo->GetTupleSlot();
+   const auto tuple_slot = sql_table_->Insert(insert_txn, insert_redo);

    auto *const insert_key =
default_index_->GetProjectedRowInitializer().InitializeRow(key_buffer_1_);
    *reinterpret_cast<int32_t *>(insert_key->AccessForceNotNull(0)) = i;
@@ -321,8 +316,7 @@ TEST_F(BwTreeIndexTests, ScanDescending) {
        insert_txn->StageWrite(CatalogTestUtil::test_db_oid, CatalogTestUtil::test_table_oid,
tuple_initializer_);
    auto *const insert_tuple = insert_redo->Delta();
    *reinterpret_cast<int32_t *>(insert_tuple->AccessForceNotNull(0)) = i;
-   sql_table_->Insert(insert_txn, insert_redo);
-   const auto tuple_slot = insert_redo->GetTupleSlot();
+   const auto tuple_slot = sql_table_->Insert(insert_txn, insert_redo);

    auto *const insert_key =
default_index_->GetProjectedRowInitializer().InitializeRow(key_buffer_1_);
    *reinterpret_cast<int32_t *>(insert_key->AccessForceNotNull(0)) = i;
@@ -395,8 +389,7 @@ TEST_F(BwTreeIndexTests, ScanLimitAscending) {
        insert_txn->StageWrite(CatalogTestUtil::test_db_oid, CatalogTestUtil::test_table_oid,
tuple_initializer_);
    auto *const insert_tuple = insert_redo->Delta();
    *reinterpret_cast<int32_t *>(insert_tuple->AccessForceNotNull(0)) = i;
-   sql_table_->Insert(insert_txn, insert_redo);
```

```
-    const auto tuple_slot = insert_redo->GetTupleSlot();
+    const auto tuple_slot = sql_table_->Insert(insert_txn, insert_redo);

     auto *const insert_key =
default_index_->GetProjectedRowInitializer().InitializeRow(key_buffer_1_);
     *reinterpret_cast<int32_t *>(insert_key->AccessForceNotNull(0)) = i;
@@ -465,8 +458,7 @@ TEST_F(BwTreeIndexTests, ScanLimitDescending) {
         insert_txn->StageWrite(CatalogTestUtil::test_db_oid, CatalogTestUtil::test_table_oid,
tuple_initializer_);
     auto *const insert_tuple = insert_redo->Delta();
     *reinterpret_cast<int32_t *>(insert_tuple->AccessForceNotNull(0)) = i;
-    sql_table_->Insert(insert_txn, insert_redo);
-    const auto tuple_slot = insert_redo->GetTupleSlot();
+    const auto tuple_slot = sql_table_->Insert(insert_txn, insert_redo);

     auto *const insert_key =
default_index_->GetProjectedRowInitializer().InitializeRow(key_buffer_1_);
     *reinterpret_cast<int32_t *>(insert_key->AccessForceNotNull(0)) = i;
@@ -531,8 +523,7 @@ TEST_F(BwTreeIndexTests, UniqueKey1) {
       txn0->StageWrite(CatalogTestUtil::test_db_oid, CatalogTestUtil::test_table_oid,
tuple_initializer_);
   auto *insert_tuple = insert_redo->Delta();
   *reinterpret_cast<int32_t *>(insert_tuple->AccessForceNotNull(0)) = 15721;
-  sql_table_->Insert(txn0, insert_redo);
-  const auto tuple_slot = insert_redo->GetTupleSlot();
+  const auto tuple_slot = sql_table_->Insert(txn0, insert_redo);

   // txn 0 inserts into index
   auto *insert_key = unique_index_->GetProjectedRowInitializer().InitializeRow(key_buffer_1_);
@@ -561,8 +552,7 @@ TEST_F(BwTreeIndexTests, UniqueKey1) {
   insert_redo = txn1->StageWrite(CatalogTestUtil::test_db_oid, CatalogTestUtil::test_table_oid,
tuple_initializer_);
   insert_tuple = insert_redo->Delta();
   *reinterpret_cast<int32_t *>(insert_tuple->AccessForceNotNull(0)) = 15721;
-  sql_table_->Insert(txn1, insert_redo);
-  const auto new_tuple_slot = insert_redo->GetTupleSlot();
+  const auto new_tuple_slot = sql_table_->Insert(txn1, insert_redo);

   // txn 1 inserts into index and fails due to write-write conflict with txn 0
   insert_key = unique_index_->GetProjectedRowInitializer().InitializeRow(key_buffer_1_);
@@ -594,8 +584,7 @@ TEST_F(BwTreeIndexTests, UniqueKey2) {
       txn0->StageWrite(CatalogTestUtil::test_db_oid, CatalogTestUtil::test_table_oid,
tuple_initializer_);
```

```
   auto *insert_tuple = insert_redo->Delta();
   *reinterpret_cast<int32_t *>(insert_tuple->AccessForceNotNull(0)) = 15721;
-  sql_table_->Insert(txn0, insert_redo);
-  const auto tuple_slot = insert_redo->GetTupleSlot();
+  const auto tuple_slot = sql_table_->Insert(txn0, insert_redo);

   // txn 0 inserts into index
   auto *insert_key = unique_index_->GetProjectedRowInitializer().InitializeRow(key_buffer_1_);
@@ -621,8 +610,7 @@ TEST_F(BwTreeIndexTests, UniqueKey2) {
   insert_redo = txn1->StageWrite(CatalogTestUtil::test_db_oid, CatalogTestUtil::test_table_oid,
tuple_initializer_);
   insert_tuple = insert_redo->Delta();
   *reinterpret_cast<int32_t *>(insert_tuple->AccessForceNotNull(0)) = 15721;
-  sql_table_->Insert(txn1, insert_redo);
-  const auto new_tuple_slot = insert_redo->GetTupleSlot();
+  const auto new_tuple_slot = sql_table_->Insert(txn1, insert_redo);

   // txn 1 inserts into index and fails due to visible key conflict with txn 0
   insert_key = unique_index_->GetProjectedRowInitializer().InitializeRow(key_buffer_1_);
@@ -652,8 +640,7 @@ TEST_F(BwTreeIndexTests, UniqueKey3) {
       txn0->StageWrite(CatalogTestUtil::test_db_oid, CatalogTestUtil::test_table_oid,
tuple_initializer_);
   auto *insert_tuple = insert_redo->Delta();
   *reinterpret_cast<int32_t *>(insert_tuple->AccessForceNotNull(0)) = 15721;
-  sql_table_->Insert(txn0, insert_redo);
-  const auto tuple_slot = insert_redo->GetTupleSlot();
+  const auto tuple_slot = sql_table_->Insert(txn0, insert_redo);

   // txn 0 inserts into index
   auto *insert_key = unique_index_->GetProjectedRowInitializer().InitializeRow(key_buffer_1_);
@@ -675,8 +662,7 @@ TEST_F(BwTreeIndexTests, UniqueKey3) {
   insert_redo = txn0->StageWrite(CatalogTestUtil::test_db_oid, CatalogTestUtil::test_table_oid,
tuple_initializer_);
   insert_tuple = insert_redo->Delta();
   *reinterpret_cast<int32_t *>(insert_tuple->AccessForceNotNull(0)) = 15721;
-  sql_table_->Insert(txn0, insert_redo);
-  const auto new_tuple_slot = insert_redo->GetTupleSlot();
+  const auto new_tuple_slot = sql_table_->Insert(txn0, insert_redo);

   // txn 0 inserts into index and fails due to visible key conflict with txn 0
   insert_key = unique_index_->GetProjectedRowInitializer().InitializeRow(key_buffer_1_);
@@ -705,8 +691,7 @@ TEST_F(BwTreeIndexTests, UniqueKey4) {
       txn0->StageWrite(CatalogTestUtil::test_db_oid, CatalogTestUtil::test_table_oid,
```

```
tuple_initializer_);
  auto *insert_tuple = insert_redo->Delta();
  *reinterpret_cast<int32_t *>(insert_tuple->AccessForceNotNull(0)) = 15721;
-  sql_table_->Insert(txn0, insert_redo);
-  const auto tuple_slot = insert_redo->GetTupleSlot();
+  const auto tuple_slot = sql_table_->Insert(txn0, insert_redo);

  // txn 0 inserts into index
  auto *insert_key = unique_index_->GetProjectedRowInitializer().InitializeRow(key_buffer_1_);
@@ -739,8 +724,7 @@ TEST_F(BwTreeIndexTests, UniqueKey4) {
  insert_redo = txn1->StageWrite(CatalogTestUtil::test_db_oid, CatalogTestUtil::test_table_oid,
tuple_initializer_);
  insert_tuple = insert_redo->Delta();
  *reinterpret_cast<int32_t *>(insert_tuple->AccessForceNotNull(0)) = 15721;
-  sql_table_->Insert(txn1, insert_redo);
-  const auto new_tuple_slot = insert_redo->GetTupleSlot();
+  const auto new_tuple_slot = sql_table_->Insert(txn1, insert_redo);

  // txn 1 inserts into index and fails due to write-write conflict with txn 0
  insert_key = unique_index_->GetProjectedRowInitializer().InitializeRow(key_buffer_1_);
@@ -789,8 +773,7 @@ TEST_F(BwTreeIndexTests, CommitInsert1) {
      txn0->StageWrite(CatalogTestUtil::test_db_oid, CatalogTestUtil::test_table_oid,
tuple_initializer_);
  auto *insert_tuple = insert_redo->Delta();
  *reinterpret_cast<int32_t *>(insert_tuple->AccessForceNotNull(0)) = 15721;
-  sql_table_->Insert(txn0, insert_redo);
-  const auto tuple_slot = insert_redo->GetTupleSlot();
+  const auto tuple_slot = sql_table_->Insert(txn0, insert_redo);

  // txn 0 inserts into index
  auto *const insert_key =
default_index_->GetProjectedRowInitializer().InitializeRow(key_buffer_1_);
@@ -864,8 +847,7 @@ TEST_F(BwTreeIndexTests, CommitInsert2) {
      txn1->StageWrite(CatalogTestUtil::test_db_oid, CatalogTestUtil::test_table_oid,
tuple_initializer_);
  auto *insert_tuple = insert_redo->Delta();
  *reinterpret_cast<int32_t *>(insert_tuple->AccessForceNotNull(0)) = 15721;
-  sql_table_->Insert(txn1, insert_redo);
-  const auto tuple_slot = insert_redo->GetTupleSlot();
+  const auto tuple_slot = sql_table_->Insert(txn1, insert_redo);

  // txn 1 inserts into index
  auto *const insert_key =
```

```
default_index_->GetProjectedRowInitializer().InitializeRow(key_buffer_1_);
@@ -936,8 +918,7 @@ TEST_F(BwTreeIndexTests, AbortInsert1) {
      txn0->StageWrite(CatalogTestUtil::test_db_oid, CatalogTestUtil::test_table_oid,
tuple_initializer_);
   auto *insert_tuple = insert_redo->Delta();
   *reinterpret_cast<int32_t *>(insert_tuple->AccessForceNotNull(0)) = 15721;
-  sql_table_->Insert(txn0, insert_redo);
-  const auto tuple_slot = insert_redo->GetTupleSlot();
+  const auto tuple_slot = sql_table_->Insert(txn0, insert_redo);

   // txn 0 inserts into index
   auto *const insert_key =
default_index_->GetProjectedRowInitializer().InitializeRow(key_buffer_1_);
@@ -1010,8 +991,7 @@ TEST_F(BwTreeIndexTests, AbortInsert2) {
      txn1->StageWrite(CatalogTestUtil::test_db_oid, CatalogTestUtil::test_table_oid,
tuple_initializer_);
   auto *insert_tuple = insert_redo->Delta();
   *reinterpret_cast<int32_t *>(insert_tuple->AccessForceNotNull(0)) = 15721;
-  sql_table_->Insert(txn1, insert_redo);
-  const auto tuple_slot = insert_redo->GetTupleSlot();
+  const auto tuple_slot = sql_table_->Insert(txn1, insert_redo);

   // txn 1 inserts into index
   auto *const insert_key =
default_index_->GetProjectedRowInitializer().InitializeRow(key_buffer_1_);
@@ -1081,8 +1061,7 @@ TEST_F(BwTreeIndexTests, CommitUpdate1) {
      insert_txn->StageWrite(CatalogTestUtil::test_db_oid, CatalogTestUtil::test_table_oid,
tuple_initializer_);
   auto *insert_tuple = insert_redo->Delta();
   *reinterpret_cast<int32_t *>(insert_tuple->AccessForceNotNull(0)) = 15721;
-  sql_table_->Insert(insert_txn, insert_redo);
-  const auto tuple_slot = insert_redo->GetTupleSlot();
+  const auto tuple_slot = sql_table_->Insert(insert_txn, insert_redo);

   // insert_txn inserts into index
   auto *insert_key = default_index_->GetProjectedRowInitializer().InitializeRow(key_buffer_1_);
@@ -1112,8 +1091,8 @@ TEST_F(BwTreeIndexTests, CommitUpdate1) {
   insert_redo = txn0->StageWrite(CatalogTestUtil::test_db_oid, CatalogTestUtil::test_table_oid,
tuple_initializer_);
   insert_tuple = insert_redo->Delta();
   *reinterpret_cast<int32_t *>(insert_tuple->AccessForceNotNull(0)) = 15445;
-  sql_table_->Insert(txn0, insert_redo);
-  const auto new_tuple_slot = insert_redo->GetTupleSlot();
+
```

```
+  const auto new_tuple_slot = sql_table_->Insert(txn0, insert_redo);
+
   insert_key = default_index_->GetProjectedRowInitializer().InitializeRow(key_buffer_1_);
   *reinterpret_cast<int32_t *>(insert_key->AccessForceNotNull(0)) = 15445;
   EXPECT_TRUE(default_index_->Insert(txn0, *insert_key, new_tuple_slot));
@@ -1209,8 +1188,7 @@ TEST_F(BwTreeIndexTests, CommitUpdate2) {
       insert_txn->StageWrite(CatalogTestUtil::test_db_oid, CatalogTestUtil::test_table_oid,
tuple_initializer_);
   auto *insert_tuple = insert_redo->Delta();
   *reinterpret_cast<int32_t *>(insert_tuple->AccessForceNotNull(0)) = 15721;
-  sql_table_->Insert(insert_txn, insert_redo);
-  const auto tuple_slot = insert_redo->GetTupleSlot();
+  const auto tuple_slot = sql_table_->Insert(insert_txn, insert_redo);

   // insert_txn inserts into index
   auto *insert_key = default_index_->GetProjectedRowInitializer().InitializeRow(key_buffer_1_);
@@ -1240,8 +1218,8 @@ TEST_F(BwTreeIndexTests, CommitUpdate2) {
   insert_redo = txn1->StageWrite(CatalogTestUtil::test_db_oid, CatalogTestUtil::test_table_oid,
tuple_initializer_);
   insert_tuple = insert_redo->Delta();
   *reinterpret_cast<int32_t *>(insert_tuple->AccessForceNotNull(0)) = 15445;
-  sql_table_->Insert(txn1, insert_redo);
-  const auto new_tuple_slot = insert_redo->GetTupleSlot();
+  const auto new_tuple_slot = sql_table_->Insert(txn1, insert_redo);
+
   insert_key = default_index_->GetProjectedRowInitializer().InitializeRow(key_buffer_1_);
   *reinterpret_cast<int32_t *>(insert_key->AccessForceNotNull(0)) = 15445;
   EXPECT_TRUE(default_index_->Insert(txn1, *insert_key, new_tuple_slot));
@@ -1337,8 +1315,7 @@ TEST_F(BwTreeIndexTests, AbortUpdate1) {
       insert_txn->StageWrite(CatalogTestUtil::test_db_oid, CatalogTestUtil::test_table_oid,
tuple_initializer_);
   auto *insert_tuple = insert_redo->Delta();
   *reinterpret_cast<int32_t *>(insert_tuple->AccessForceNotNull(0)) = 15721;
-  sql_table_->Insert(insert_txn, insert_redo);
-  const auto tuple_slot = insert_redo->GetTupleSlot();
+  const auto tuple_slot = sql_table_->Insert(insert_txn, insert_redo);

   // insert_txn inserts into index
   auto *insert_key = default_index_->GetProjectedRowInitializer().InitializeRow(key_buffer_1_);
@@ -1368,8 +1345,8 @@ TEST_F(BwTreeIndexTests, AbortUpdate1) {
   insert_redo = txn0->StageWrite(CatalogTestUtil::test_db_oid, CatalogTestUtil::test_table_oid,
tuple_initializer_);
   insert_tuple = insert_redo->Delta();
```

```
   *reinterpret_cast<int32_t *>(insert_tuple->AccessForceNotNull(0)) = 15445;
-  sql_table_->Insert(txn0, insert_redo);
-  const auto new_tuple_slot = insert_redo->GetTupleSlot();
+  const auto new_tuple_slot = sql_table_->Insert(txn0, insert_redo);
+
   insert_key = default_index_->GetProjectedRowInitializer().InitializeRow(key_buffer_1_);
   *reinterpret_cast<int32_t *>(insert_key->AccessForceNotNull(0)) = 15445;
   EXPECT_TRUE(default_index_->Insert(txn0, *insert_key, new_tuple_slot));
@@ -1465,8 +1442,7 @@ TEST_F(BwTreeIndexTests, AbortUpdate2) {
       insert_txn->StageWrite(CatalogTestUtil::test_db_oid, CatalogTestUtil::test_table_oid,
tuple_initializer_);
   auto *insert_tuple = insert_redo->Delta();
   *reinterpret_cast<int32_t *>(insert_tuple->AccessForceNotNull(0)) = 15721;
-  sql_table_->Insert(insert_txn, insert_redo);
-  const auto tuple_slot = insert_redo->GetTupleSlot();
+  const auto tuple_slot = sql_table_->Insert(insert_txn, insert_redo);

   // insert_txn inserts into index
   auto *insert_key = default_index_->GetProjectedRowInitializer().InitializeRow(key_buffer_1_);
@@ -1496,8 +1472,8 @@ TEST_F(BwTreeIndexTests, AbortUpdate2) {
   insert_redo = txn1->StageWrite(CatalogTestUtil::test_db_oid, CatalogTestUtil::test_table_oid,
tuple_initializer_);
   insert_tuple = insert_redo->Delta();
   *reinterpret_cast<int32_t *>(insert_tuple->AccessForceNotNull(0)) = 15445;
-  sql_table_->Insert(txn1, insert_redo);
-  const auto new_tuple_slot = insert_redo->GetTupleSlot();
+  const auto new_tuple_slot = sql_table_->Insert(txn1, insert_redo);
+
   insert_key = default_index_->GetProjectedRowInitializer().InitializeRow(key_buffer_1_);
   *reinterpret_cast<int32_t *>(insert_key->AccessForceNotNull(0)) = 15445;
   EXPECT_TRUE(default_index_->Insert(txn1, *insert_key, new_tuple_slot));
@@ -1593,8 +1569,7 @@ TEST_F(BwTreeIndexTests, CommitDelete1) {
       insert_txn->StageWrite(CatalogTestUtil::test_db_oid, CatalogTestUtil::test_table_oid,
tuple_initializer_);
   auto *insert_tuple = insert_redo->Delta();
   *reinterpret_cast<int32_t *>(insert_tuple->AccessForceNotNull(0)) = 15721;
-  sql_table_->Insert(insert_txn, insert_redo);
-  const auto tuple_slot = insert_redo->GetTupleSlot();
+  const auto tuple_slot = sql_table_->Insert(insert_txn, insert_redo);

   // insert_txn inserts into index
   auto *insert_key = default_index_->GetProjectedRowInitializer().InitializeRow(key_buffer_1_);
@@ -1686,8 +1661,7 @@ TEST_F(BwTreeIndexTests, CommitDelete2) {
```

```
      insert_txn->StageWrite(CatalogTestUtil::test_db_oid, CatalogTestUtil::test_table_oid,
tuple_initializer_);
   auto *insert_tuple = insert_redo->Delta();
   *reinterpret_cast<int32_t *>(insert_tuple->AccessForceNotNull(0)) = 15721;
-  sql_table_->Insert(insert_txn, insert_redo);
-  const auto tuple_slot = insert_redo->GetTupleSlot();
+  const auto tuple_slot = sql_table_->Insert(insert_txn, insert_redo);

   // insert_txn inserts into index
   auto *insert_key = default_index_->GetProjectedRowInitializer().InitializeRow(key_buffer_1_);
@@ -1779,8 +1753,7 @@ TEST_F(BwTreeIndexTests, AbortDelete1) {
      insert_txn->StageWrite(CatalogTestUtil::test_db_oid, CatalogTestUtil::test_table_oid,
tuple_initializer_);
   auto *insert_tuple = insert_redo->Delta();
   *reinterpret_cast<int32_t *>(insert_tuple->AccessForceNotNull(0)) = 15721;
-  sql_table_->Insert(insert_txn, insert_redo);
-  const auto tuple_slot = insert_redo->GetTupleSlot();
+  const auto tuple_slot = sql_table_->Insert(insert_txn, insert_redo);

   // insert_txn inserts into index
   auto *insert_key = default_index_->GetProjectedRowInitializer().InitializeRow(key_buffer_1_);
@@ -1873,8 +1846,7 @@ TEST_F(BwTreeIndexTests, AbortDelete2) {
      insert_txn->StageWrite(CatalogTestUtil::test_db_oid, CatalogTestUtil::test_table_oid,
tuple_initializer_);
   auto *insert_tuple = insert_redo->Delta();
   *reinterpret_cast<int32_t *>(insert_tuple->AccessForceNotNull(0)) = 15721;
-  sql_table_->Insert(insert_txn, insert_redo);
-  const auto tuple_slot = insert_redo->GetTupleSlot();
+  const auto tuple_slot = sql_table_->Insert(insert_txn, insert_redo);

   // insert_txn inserts into index
   auto *insert_key = default_index_->GetProjectedRowInitializer().InitializeRow(key_buffer_1_);
diff --git a/test/storage/bwtree_key_test.cpp b/test/storage/bwtree_key_test.cpp
index 963afae9f..fec0717c6 100644
--- a/test/storage/bwtree_key_test.cpp
+++ b/test/storage/bwtree_key_test.cpp
@@ -244,8 +244,7 @@ class BwTreeKeyTests : public TerrierTest {
    index->ScanKey(*txn, *key, &results);
    EXPECT_TRUE(results.empty());

-   sql_table.Insert(txn, insert_redo);
-   const auto tuple_slot = insert_redo->GetTupleSlot();
+   const auto tuple_slot = sql_table.Insert(txn, insert_redo);
```

```
    EXPECT_TRUE(index->Insert(txn, *key, tuple_slot));

diff --git a/test/util/tpcc/new_order.cpp b/test/util/tpcc/new_order.cpp
index 7a380be3c..5a919b96f 100644
--- a/test/util/tpcc/new_order.cpp
+++ b/test/util/tpcc/new_order.cpp
@@ -90,10 +90,9 @@ bool NewOrder::Execute(transaction::TransactionManager *const
txn_manager, Datab
  *reinterpret_cast<int8_t
*>(new_order_insert_tuple->AccessForceNotNull(no_d_id_insert_pr_offset)) = args.d_id;
  *reinterpret_cast<int8_t
*>(new_order_insert_tuple->AccessForceNotNull(no_w_id_insert_pr_offset)) = args.w_id;

-  db->new_order_table_->Insert(txn, new_order_insert_redo);
+  const auto new_order_slot = db->new_order_table_->Insert(txn, new_order_insert_redo);

  // insert in New Order index
-  const auto new_order_slot = new_order_insert_redo->GetTupleSlot();
  const auto new_order_key_pr_initializer =
db->new_order_primary_index_->GetProjectedRowInitializer();
  auto *const new_order_key =
new_order_key_pr_initializer.InitializeRow(worker->new_order_key_buffer);

@@ -119,10 +118,9 @@ bool NewOrder::Execute(transaction::TransactionManager *const
txn_manager, Datab
  *reinterpret_cast<int8_t
*>(order_insert_tuple->AccessForceNotNull(o_all_local_insert_pr_offset)) =
      static_cast<int8_t>(args.o_all_local);

-  db->order_table_->Insert(txn, order_insert_redo);
+  const auto order_slot = db->order_table_->Insert(txn, order_insert_redo);

  // insert in Order index
-  const auto order_slot = order_insert_redo->GetTupleSlot();
  const auto order_key_pr_initializer =
db->order_primary_index_->GetProjectedRowInitializer();
  auto *const order_key = order_key_pr_initializer.InitializeRow(worker->order_key_buffer);

@@ -263,10 +261,9 @@ bool NewOrder::Execute(transaction::TransactionManager *const
txn_manager, Datab
        order_line_insert_tuple->AccessForceNotNull(ol_dist_info_insert_pr_offset)) =
varlen_entry;
```

```
    }

-   db->order_line_table_->Insert(txn, order_line_insert_redo);
+    const auto order_line_slot = db->order_line_table_->Insert(txn, order_line_insert_redo);

    // insert in Order Line index
-   const auto order_line_slot = order_line_insert_redo->GetTupleSlot();
    const auto order_line_key_pr_initializer =
db->order_line_primary_index_->GetProjectedRowInitializer();
    auto *const order_line_key =
order_line_key_pr_initializer.InitializeRow(worker->order_line_key_buffer);
```