Sample scenario - store all field changes for an object

In this scenario, the customer wants, for whatever reason, to track changes of all the fields in a single record. Salesforce provides the default field tracking, but it is available for only twenty fields per object. If this object we are talking about has more, then it is impossible to solve it with the standard, declarative tools.

Big objects are the ideal candidate for this, because we are talking about data that users probably don't need to report (big objects do not support reporting), there's a change that it is a lot of data (if the record is changed frequently), and possibly there's a legal reason for keeping those changes stored (compliance).

So to do that we'll need a trigger on the object, running preferably on the `after update` trigger event. At this point the record is already saved but the transaction is not yet committed to the database, so the changes were made and we get the difference using `Trigger.oldMap` to get the old version of the changed records.

After iterating through all the fields on the object, we check for differences, and for each one we instantiate a new big object. When the iteration ends we insert them immediately (using `Database.insertImmediate()`).

In this configuration, the big object's index would be the related record's Id, the field that was modified and the date/time stamp of the change (depending on requirements, one might want to spend some time thinking if it is best to have the timestamp before the field name). This way, if we wanted to display the data in a Lightning Component, for example, we could query the specific record data synchronously in Apex because of the indices created:

```
SELECT
    RecordId__c,
    Field__c,
    Timestamp__c,
    OldValue__c,
    NewValue__c
FROM ObjectHistory__b
WHERE RecordId__c = :theRecordId
```

Revision #3
Created 13 November 2019 11:47:04 by Renato Oliveira
Updated 10 March 2020 10:47:48 by thejamesjames