Exercises for
Database Implementation
TUM

Viktor Leis (leis@in.tum.de)

Assignment 3

Info

- Send an email with information about your git repository or send your submission as a `zip` or `tar.gz` file to `leis@in.tum.de` by 13 May 2013, 10:00am.

Excercise 1

Create a metadata/schema segment for your database system. For each relation store its name, segment id, size in pages, and all its attributes and types. You should be able to serialize and deserialize the meta data segment to/from disk. Always store it in segment 0.

If you want to able to read SQL schemas, you may find the parser (and the example schema file) useful that you can download from the website. Alternatively, you can simply create the schema programmatically.

Excercise 2

Implement *slotted pages* for your database system.

1. Define a segment type `SPSegment` that operates on slotted pages. A slotted page consists of three parts: A header, the slots and the (variable-length) records. Records are addressed by TIDs (tuple identifier), consisting of a page ID and a slot ID.

2. Provide an interface to `insert`, `remove`, `update` and `lookup` "records". Records consist of a size and the data (you could also think of them as strings: they contain a length indicator `len` followed by a pointer to `len` characters).

   `TID SPSegment::insert(const Record& r)` Searches through the segment's pages looking for a page with enough space to store `r`. Returns the TID identifying the location where `r` was stored. Note: This can be implemented much more efficiently with a free space bitmap as described in chapter 3, slide 3, but you are not required to do this.

   `bool SPSegment::remove(TID tid)` Deletes the record pointed to by `tid` and updates the page header accordingly.

   `Record SPSegment::lookup(TID tid)` Returns the read-only record associated with TID `tid`.

   `bool SPSegment::update(TID tid, const Record& r)` Updates the record pointed to by `tid` with the content of record `r`.