NAME

indexscan – scanning Shore indexes

SYNOPSIS

```
enum CompareOp {
    badOp=0x0, eqOp=0x1, gtOp=0x2, geOp=0x3, ltOp=0x4, leOp=0x5
};
struct IndexId {
      lrid_t            obj;
      int               i;
};
typedef IndexId IndexId;

VASResult       shore_vas::openIndexScan(
    const IndexId       &iid,
    CompareOp           lc,
    const vec_t         &lbound,
    CompareOp           uc,
    const vec_t         &ubound,
    Cookie              *cookie
);
VASResult       shore_vas::nextIndexScan(
    Cookie              *cookie,
    const vec_t         &key,
    ObjectSize          *keylen,
    const vec_t         &value,
    ObjectSize          *vallen,
    bool                *eof=0
);
VASResult       shore_vas::closeIndexScan(
    const Cookie        &cookie
);
```

DESCRIPTION

Manual indexes in Shore are attributes of objects. The language binding layer creates, destroys, and updates indexes with the methods described in **manualindex(svas).**

When keys are known, the key-value pairs can be located with **findIndexElem,** which is also described in **manualindex(svas).**

The methods described here are used to locate ranges of key-value pairs in an index. The range of an index scan is limited by upper- and lower-bounds that are given when the scan is *opened.* Opening a scan establishes context for the scan. Each key-value pair withing the given range is returned by a distinct invocation of the method **nextIndexScan.** The scan must be closed with **closeIndexScan.**

Only one scan can be open in any transaction at any time. An index cannot be updated when a scan is open on the index.

ARGUMENTS

The argument *iid* identifies the manual index.

Opening a scan creates a context in which the method **nextIndexScan** works. **OpenIndexScan** returns a *cookie*, which is a value that is meaningful to the Shore Value-Added Server, but is opaque to the caller. The cookie identifies the context of the scan to the SVAS. The value written into **cookie* by the **openIndexScan** must be left undisturbed between calls to any of these methods.

The range of the scan are determined by the argument pairs *lc, lbound* and *uc, ubound.* A lower- or upper-bound is a vector with one element, which identifies a byte string that is compared with the keys in the index.

The lowest and highest possible values for bounds are *negative infinity* , and *positive infinity* , which are static members of the class *vec_t.* The following example always scans an entire index called "index":

```
class shore_vas *v;
Cookie       context;
  IndexId            iid

  iid.oid = ...
  iid.i = 1;
...
v->openIndexScan(iid, geOp, vec_t::neg_inf,
                 leOp, vec_t::pos_inf, &context);
```

The method **nextIndexScan** returns key-value pairs for which the key falls within the bounds of the scan. Each time it is called, the next key-value pair is returned. The key is placed in caller's address space in the place or places identified by the argument *key.* The caller is responsible for allocating this space, and it must be large enough to accommodate any of the keys in the range of the scan. The SVAS writes the length of the key returned into *keylen.* The value is treated the same way. Both vectors, *key* and *value* are passed as *const references* because the vectors are not modified, but *the memory areas that are identified by these vectors are updated.* The values **keylen* and **vallen* are ignored by **nextIndexScan.**

ENVIRONMENT

and can be invoked by clients and by value-added servers.

TODO: need a note about what kinds of locks are acquired.

BUGS

NB: A transaction can have no more than one pool scan open at any time. (TODO: extend SSM)

NB: index keys are not typed. (Typed keys are forthcoming. (TODO))

ERRORS

If a key or value is larger than the memory given in an output vector argument, the Shore Storage Manager returns an error (eRECWONTFIT). See **errors(ssm)** for more information about these errors.

Deadlocks can occur while locks are being acquired. See **transaction(svas)** for information about deadlocks.

A complete list of errors is in **errors(svas).**

VERSION

This manual page applies to Version 1.1 of the Shore software.

SPONSORSHIP

The Shore project is sponsored by the Advanced Research Project Agency, ARPA order number 018 (formerly 8230), monitored by the U.S. Army Research Laboratory under contract DAAB07-91-C-Q518.

SEE ALSO

registered(svas), transaction(svas), errors(svas), errors(ssm) and **manualindex(svas).**