$o : object	$n: number
$f: field name (string)	$oc: object or classname
$fo: field object	$db: y_database object
(string or y_op_field)	$yop: y_op object
$q: y_query object	$oa: object array
$v: value of field for comparison	$c: class name

```
$db = new y_db($config);
  $config['y_db_name'] = 'thing_-_thing.com';
  $config['y_db_host'] = 'thing.com'; // opt
  $config['y_db_user'] ='whoever';
  $config['y_db_password'] = 'whatever';
  $config['y_db_log'] = 'folder'; // opt
  $config['y_db_prefix'] = 'whatever_'; // opt
  $config['y_db_no_locks'] = TRUE; // opt
  $config['y_db_debug'] = TRUE; // opt

$db->lock();

$db->select($o, ...);
// $o can be array of same

$n = $db->info($oc, $yop);

$n = $db->delete($o, $f,... );
// $f can be array of same

$n = $db->update($o, $f,....);
// ditto

$n = $db->insert($o);
```

```
$q = $db->query();  or $q = $db->query($owrt);

$q->where($yop);

$q->ascending($fo, ...); // $fo can be array of same
$q->descending($fo, ...); // ditto

$q->limit($count [, $start]);
$q->distinct([$fo]); // $fo can be array of same

$info = $q->info($oc, $yop);

$q->groupby($fop, ...);
/* $fop is a mixture of field names and operators,
or array of same; $groupby needs a subsequent select */

$remaining = $q->select($o, ...);
/* or array of same; $o can also be result of
y_op::<whatever>join */

$n = $q->delete($oc, [$f, ...]); // or array of $f
$n = $q->update($o, $key [, ...]);
// or array of $f, at least one $f mandatory
$n = $q->insert($o, ...);
// or array of objects;
// common classes and fields optimized
```

```
$yop = y_op::eq($fo, $v, ...); // or array of values
$yop = y_op::ne($fo, $v, ...); // ditto
$yop = y_op::lt($fo, $v);
$yop = y_op::le($fo, $v);
$yop = y_op::gt($fo, $v);
$yop = y_op::ge($fo, $v);
$yop = y_op::isnull($fo);
$yop = y_op::isntnull($fo);
$yop = y_op::between($fo, $v1, $v2);

$yop = y_op:aand($yop1, $yop2, ...);
// or array of same
$yop = y_op:oor($yop1, $yop2, ...);
// or array of same
$yop = y_op::not($yop);

$yop = y_op::max($fo);
$yop = y_op::min($fo);
$yop = y_op::average($fo);
$yop = y_op::standard_deviation($fo);
$yop = y_op::sum($fo);
$yop = y_op::count_distinct($fo);
$yop = y_op::count([$fo]);

$yop = y_op::feq($fo1, $fo2); // comparison of fields
$yop = y_op::fne($fo1, $fo2);
$yop = y_op::flt($fo1, $fo2);
$yop = y_op::fle($fo1, $fo2);
$yop = y_op::fgt($fo1, $fo2);
$yop = y_op::fge($fo1, $fo2);

$yop = y_op_field::field($f [, $n]);
// use $n to disambiguate fields in a join

$yop = y_op::leftjoin($yop);
$yop = y_op::rightjoin($yop);
$yop = y_op::innerjoin($yop);
// for more complex conditions than field equality
// applies to preceding and subsequent objects
```

```
class ... {
  function y_table() {
    return 'tablename'; /* including any prefix */ }
    /* default same name excluding any prefix */

  function y_fields() {
    return array($f=>$c, ...); } }
    /* default same names in class and table */

  function y_xrefs($c) {
    return array($fme=>$fit, ...); } }
    /* explicit mapping of field fme etc of class
    call in to fit etc of class classname,
    default foo -> classname_foo */
}
```