xapian-core Reference Manual

1.1.1

Generated by Doxygen 1.5.2

Wed Jun 10 02:28:30 2009

Contents

Chapter 1

xapian-core Namespace Index

1.1 xapian-core Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 2

xapian-core Hierarchical Index

2.1 xapian-core Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

xapian-core Class Index

3.1 xapian-core Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

xapian-core File Index

4.1 xapian-core File List

Here is a list of all documented files with brief descriptions:

Chapter 5

xapian-core Namespace Documentation

5.1 Xapian Namespace Reference

The Xapian library lives in the Xapian namespace.

Classes

- class Database
- class WritableDatabase
- class **DerefStringWrapper_**
- class Document

  A document in the database - holds data, values, terms, and postings.

- class MSet
- class MSetIterator
- class ESet
- class ESetIterator
- class RSet
- class MatchDecider
- class Enquire
- class ErrorHandler
- class ExpandDecider
- class ExpandDeciderAnd
- class ExpandDeciderFilterTerms
- class **TermPosWrapper**
- class PositionIterator
- class **DocIDWrapper**
- class PostingIterator
- class PostingSource

- class ValuePostingSource
- class ValueWeightPostingSource
- class ValueMapPostingSource
- class FixedWeightPostingSource
- class Query
- class Stopper

  Base class for stop-word decision functor.

- class SimpleStopper

  Simple implementation of Stopper class - this will suit most users.

- struct ValueRangeProcessor

  Base class for value range processors.

- class StringValueRangeProcessor
- class DateValueRangeProcessor
- class NumberValueRangeProcessor
- class QueryParser

  Build a Xapian::Query object from a user query string.

- struct ReplicationInfo
- class DatabaseMaster
- class DatabaseReplica
- class SerialisationContext
- class Sorter
- class MultiValueSorter
- class Stem

  Class representing a stemming algorithm.

- class TermGenerator
- class TermIterator
- class Utf8Iterator
- class **ValueIteratorEnd_**
- class ValueIterator

  Class for iterating over document values.

- class ValueSetMatchDecider
- class Weight
- class BoolWeight
- class BM25Weight

  Xapian::Weight subclass implementing the BM25 probabilistic formula.

- class TradWeight

Typedefs

- typedef unsigned doccount
- typedef int doccount_diff
- typedef unsigned docid
- typedef double doclength
- typedef int percent
- typedef unsigned termcount
- typedef int termcount_diff
- typedef unsigned termpos
- typedef int termpos_diff
- typedef unsigned timeout
- typedef unsigned valueno
- typedef int valueno_diff
- typedef double weight

Functions

- bool **operator==** (const MSetIterator &a, const MSetIterator &b)
- bool **operator!=** (const MSetIterator &a, const MSetIterator &b)
- bool **operator==** (const ESetIterator &a, const ESetIterator &b)
- bool **operator!=** (const ESetIterator &a, const ESetIterator &b)
- bool operator== (const PositionIterator &a, const PositionIterator &b)

  Test equality of two PositionIterators.

- bool operator!= (const PositionIterator &a, const PositionIterator &b)

  Test inequality of two PositionIterators.

- bool operator== (const PostingIterator &a, const PostingIterator &b)

  Test equality of two PostingIterators.

- bool operator!= (const PostingIterator &a, const PostingIterator &b)

  Test inequality of two PostingIterators.

- XAPIAN_VISIBILITY_DEFAULT std::string sortable_serialise (double value)
- XAPIAN_VISIBILITY_DEFAULT double sortable_unserialise (const std::string &value)
- bool **operator==** (const TermIterator &a, const TermIterator &b)
- bool **operator!=** (const TermIterator &a, const TermIterator &b)
- bool **operator==** (const ValueIterator &a, const ValueIterator &b)
- bool **operator==** (const ValueIterator &a, const ValueIteratorEnd_ &)
- bool **operator==** (const ValueIteratorEnd_ &a, const ValueIterator &b)
- bool **operator==** (const ValueIteratorEnd_ &, const ValueIteratorEnd_ &)
- bool **operator!=** (const ValueIterator &a, const ValueIterator &b)
- bool **operator!=** (const ValueIterator &a, const ValueIteratorEnd_ &b)
- bool **operator!=** (const ValueIteratorEnd_ &a, const ValueIterator &b)

- bool **operator!=** (const ValueIteratorEnd_ &a, const ValueIteratorEnd_ &b)
- XAPIAN_VISIBILITY_DEFAULT const char * version_string ()
- XAPIAN_VISIBILITY_DEFAULT int major_version ()
- XAPIAN_VISIBILITY_DEFAULT int minor_version ()
- XAPIAN_VISIBILITY_DEFAULT int revision ()

Variables

- const int DB_CREATE_OR_OPEN = 1
- const int DB_CREATE = 2
- const int DB_CREATE_OR_OVERWRITE = 3
- const int DB_OPEN = 4
- const valueno BAD_VALUENO = static_cast<valueno>(-1)

5.1.1 Detailed Description

The Xapian library lives in the Xapian namespace.

5.1.2 Typedef Documentation

5.1.2.1 typedef unsigned Xapian::doccount

A count of documents.

This is used to hold values such as the number of documents in a database and the frequency of a term in the database.

5.1.2.2 typedef int Xapian::doccount_diff

A signed difference between two counts of documents.

This is used by the Xapian classes which are STL containers of documents for "difference_type".

5.1.2.3 typedef unsigned Xapian::docid

A unique identifier for a document.

Docid 0 is invalid, providing an "out of range" value which can be used to mean "not a valid document".

5.1.2.4 typedef double Xapian::doclength

A normalised document length.

The normalised document length is the document length divided by the average document length in the database.

5.1.2.5 typedef int Xapian::percent

The percentage score for a document in an MSet.

5.1.2.6 typedef unsigned Xapian::termcount

A counts of terms.

This is used to hold values such as the Within Document Frequency (wdf).

5.1.2.7 typedef int Xapian::termcount_diff

A signed difference between two counts of terms.

This is used by the Xapian classes which are STL containers of terms for "difference_type".

5.1.2.8 typedef unsigned Xapian::termpos

A term position within a document or query.

5.1.2.9 typedef int Xapian::termpos_diff

A signed difference between two term positions.

This is used by the Xapian classes which are STL containers of positions for "difference_type".

5.1.2.10 typedef unsigned Xapian::timeout

A timeout value in microseconds.

There are 1 million microseconds in a second, so for example, to set a timeout of 5 seconds use 5000000.

5.1.2.11 typedef unsigned Xapian::valueno

The number for a value slot in a document.

Any value slot number except Xapian::BAD_VALUENO is valid.

5.1.2.12 typedef int Xapian::valueno_diff

A signed difference between two value slot numbers.

This is used by the Xapian classes which are STL containers of values for "difference_type".

5.1.2.13 typedef double Xapian::weight

The weight of a document or term.

5.1.3 Function Documentation

5.1.3.1 XAPIAN_VISIBILITY_DEFAULT int Xapian::major_version ()

Report the major version of the library which the program is linked to.

This may be different to the version compiled against (given by XAPIAN_MAJOR_-VERSION) if shared libraries are being used.

5.1.3.2 XAPIAN_VISIBILITY_DEFAULT int Xapian::minor_version ()

Report the minor version of the library which the program is linked to.

This may be different to the version compiled against (given by XAPIAN_MINOR_-VERSION) if shared libraries are being used.

5.1.3.3 bool Xapian::operator!= (const PostingIterator & *a*, const PostingIterator & *b*) [inline]

Test inequality of two PostingIterators.

5.1.3.4 bool Xapian::operator!= (const PositionIterator & *a*, const PositionIterator & *b*) [inline]

Test inequality of two PositionIterators.

5.1.3.5 bool Xapian::operator== (const PostingIterator & *a*, const PostingIterator & *b*) [inline]

Test equality of two PostingIterators.

5.1.3.6 bool Xapian::operator== (const PositionIterator & *a*, const PositionIterator & *b*) [inline]

Test equality of two PositionIterators.

5.1.3.7 XAPIAN_VISIBILITY_DEFAULT int Xapian::revision ()

Report the revision of the library which the program is linked to.

This may be different to the version compiled against (given by XAPIAN_REVISION) if shared libraries are being used.

5.1.3.8 XAPIAN_VISIBILITY_DEFAULT std::string Xapian::sortable_serialise (double *value*)

Convert a floating point number to a string, preserving sort order.

This method converts a floating point number to a string, suitable for using as a value for numeric range restriction, or for use as a sort key.

The conversion is platform independent.

The conversion attempts to ensure that, for any pair of values supplied to the conversion algorithm, the result of comparing the original values (with a numeric comparison operator) will be the same as the result of comparing the resulting values (with a string comparison operator). On platforms which represent doubles with the precisions specified by IEEE_754, this will be the case: if the representation of doubles is more precise, it is possible that two very close doubles will be mapped to the same string, so will compare equal.

Note also that both zero and -zero will be converted to the same representation: since these compare equal, this satisfies the comparison constraint, but it's worth knowing this if you wish to use the encoding in some situation where this distinction matters.

Handling of NaN isn't (currently) guaranteed to be sensible.

5.1.3.9 XAPIAN_VISIBILITY_DEFAULT double Xapian::sortable_unserialise (const std::string & *value*)

Convert a string encoded using *sortable_serialise* back to a floating point number.

This expects the input to be a string produced by *sortable_serialise()*. If the input is not such a string, the value returned is undefined (but no error will be thrown).

The result of the conversion will be exactly the value which was supplied to *sortable_serialise()* when making the string on platforms which represent doubles with the precisions specified by IEEE_754, but may be a different (nearby) value on other platforms.

5.1.3.10 XAPIAN_VISIBILITY_DEFAULT const char* Xapian::version_string ()

Report the version string of the library which the program is linked with.

This may be different to the version compiled against (given by XAPIAN_VERSION) if shared libraries are being used.

5.1.4 Variable Documentation

5.1.4.1 const valueno Xapian::BAD_VALUENO = static_cast<valueno>(-1)

Reserved value to indicate "no valueno".

5.1.4.2 const int Xapian::DB_CREATE = 2

Create a new database; fail if db exists.

5.1.4.3 const int Xapian::DB_CREATE_OR_OPEN = 1

Open for read/write; create if no db exists.

5.1.4.4 const int Xapian::DB_CREATE_OR_OVERWRITE = 3

Overwrite existing db; create if none exists.

5.1.4.5 const int Xapian::DB_OPEN = 4

Open for read/write; fail if no db exists.

Chapter 6

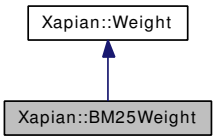
xapian-core Class Documentation

6.1 Xapian::BM25Weight Class Reference

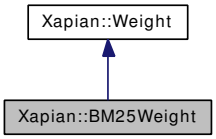
Xapian::Weight subclass implementing the BM25 probabilistic formula.

`#include <weight.h>`

Inheritance diagram for Xapian::BM25Weight:



Collaboration diagram for Xapian::BM25Weight:



Public Member Functions

- BM25Weight (double k1, double k2, double k3, double b, double min_normlen)
- std::string name () const
- std::string serialise () const
- BM25Weight * unserialise (const std::string &s) const

- Xapian::weight get_sumpart (Xapian::termcount wdf, Xapian::termcount doclen) const
- Xapian::weight get_maxpart () const
- Xapian::weight get_sumextra (Xapian::termcount doclen) const
- Xapian::weight get_maxextra () const

6.1.1 Detailed Description

Xapian::Weight subclass implementing the BM25 probabilistic formula.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 Xapian::BM25Weight::BM25Weight (double *k1*, double *k2*, double *k3*, double *b*, double *min_normlen*) [inline]

Construct a BM25Weight.

Parameters:

k1 A non-negative parameter controlling how influential within-document-frequency (wdf) is. k1=0 means that wdf doesn't affect the weights. The larger k1 is, the more wdf influences the weights. (default 1)

k2 A non-negative parameter which controls the strength of a correction factor which depends upon query length and normalised document length. k2=0 disable this factor; larger k2 makes it stronger. (default 0)

k3 A non-negative parameter controlling how influential within-query-frequency (wqf) is. k3=0 means that wqf doesn't affect the weights. The larger k3 is, the more wqf influences the weights. (default 1)

b A parameter between 0 and 1, controlling how strong the document length normalisation of wdf is. 0 means no normalisation; 1 means full normalisation. (default 0.5)

min_normlen A parameter specifying a minimum value for normalised document length. Normalised document length values less than this will be clamped to this value, helping to prevent very short documents getting large weights. (default 0.5)

6.1.3 Member Function Documentation

6.1.3.1 std::string Xapian::BM25Weight::name () const [virtual]

Return the name of this weighting scheme.

This name is used by the remote backend. It is passed with the serialised parameters to the remote server so that it knows which class to create.

Return the full namespace-qualified name of your class here - if your class is called FooWeight, return "FooWeight" from this method (Xapian::BM25Weight returns "Xapian::BM25Weight" here).

If you don't want to support the remote backend in your weighting scheme, you can just implement this to throw Xapian::UnimplementedError.

Implements Xapian::Weight.

6.1.3.2 std::string Xapian::BM25Weight::serialise () const [virtual]

Return this object's parameters serialised as a single string.

If you don't want to support the remote backend in your weighting scheme, you can just implement this to throw Xapian::UnimplementedError.

Implements Xapian::Weight.

6.1.3.3 BM25Weight* Xapian::BM25Weight::unserialise (const std::string & *s*) const [virtual]

Unserialise parameters.

This method unserialises parameters serialised by the *serialise()* method and allocates and returns a new object initialised with them.

If you don't want to support the remote backend in your weighting scheme, you can just implement this to throw Xapian::UnimplementedError.

Implements Xapian::Weight.

6.1.3.4 Xapian::weight Xapian::BM25Weight::get_sumpart (Xapian::termcount *wdf*, Xapian::termcount *doclen*) const [virtual]

Calculate the weight contribution for this object's term to a document.

The parameters give information about the document which may be used in the calculations:

Parameters:

wdf The within document frequency of the term in the document.

doclen The document's length (unnormalised).

Implements Xapian::Weight.

6.1.3.5 Xapian::weight Xapian::BM25Weight::get_maxpart () const [virtual]

Return an upper bound on what get_sumpart() can return for any document.

This information is used by the matcher to perform various optimisations, so strive to make the bound as tight as possible.

Implements Xapian::Weight.

6.1.3.6 Xapian::weight Xapian::BM25Weight::get_sumextra (Xapian::termcount *doclen*) const [virtual]

Calculate the term-independent weight component for a document.

The parameter gives information about the document which may be used in the calculations:

Parameters:

doclen The document's length (unnormalised).

Implements Xapian::Weight.

6.1.3.7 Xapian::weight Xapian::BM25Weight::get_maxextra () const [virtual]

Return an upper bound on what get_sumextra() can return for any document.

This information is used by the matcher to perform various optimisations, so strive to make the bound as tight as possible.

Implements Xapian::Weight.

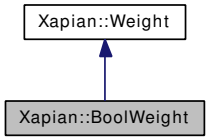
The documentation for this class was generated from the following file:

- include/xapian/weight.h

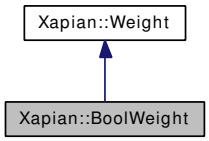
6.2 Xapian::BoolWeight Class Reference

`#include <weight.h>`

Inheritance diagram for Xapian::BoolWeight:



Collaboration diagram for Xapian::BoolWeight:



Public Member Functions

- BoolWeight ()
- std::string name () const
- std::string serialise () const
- BoolWeight * unserialise (const std::string &s) const
- Xapian::weight get_sumpart (Xapian::termcount wdf, Xapian::termcount doclen) const
- Xapian::weight get_maxpart () const
- Xapian::weight get_sumextra (Xapian::termcount doclen) const
- Xapian::weight get_maxextra () const

6.2.1 Detailed Description

Class implementing a "boolean" weighting scheme.

This weighting scheme gives all documents zero weight.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 Xapian::BoolWeight::BoolWeight () `[inline]`

Construct a BoolWeight.

6.2.3 Member Function Documentation

6.2.3.1 std::string Xapian::BoolWeight::name () const [virtual]

Return the name of this weighting scheme.

This name is used by the remote backend. It is passed with the serialised parameters to the remote server so that it knows which class to create.

Return the full namespace-qualified name of your class here - if your class is called FooWeight, return "FooWeight" from this method (Xapian::BM25Weight returns "Xapian::BM25Weight" here).

If you don't want to support the remote backend in your weighting scheme, you can just implement this to throw Xapian::UnimplementedError.

Implements Xapian::Weight.

6.2.3.2 std::string Xapian::BoolWeight::serialise () const [virtual]

Return this object's parameters serialised as a single string.

If you don't want to support the remote backend in your weighting scheme, you can just implement this to throw Xapian::UnimplementedError.

Implements Xapian::Weight.

6.2.3.3 BoolWeight* Xapian::BoolWeight::unserialise (const std::string & *s*) const [virtual]

Unserialise parameters.

This method unserialises parameters serialised by the *serialise()* method and allocates and returns a new object initialised with them.

If you don't want to support the remote backend in your weighting scheme, you can just implement this to throw Xapian::UnimplementedError.

Implements Xapian::Weight.

6.2.3.4 Xapian::weight Xapian::BoolWeight::get_sumpart (Xapian::termcount *wdf*, Xapian::termcount *doclen*) const [virtual]

Calculate the weight contribution for this object's term to a document.

The parameters give information about the document which may be used in the calculations:

Parameters:

wdf The within document frequency of the term in the document.

doclen The document's length (unnormalised).

Implements Xapian::Weight.

6.2.3.5 Xapian::weight Xapian::BoolWeight::get_maxpart () const [virtual]

Return an upper bound on what get_sumpart() can return for any document.

This information is used by the matcher to perform various optimisations, so strive to make the bound as tight as possible.

Implements Xapian::Weight.

6.2.3.6 Xapian::weight Xapian::BoolWeight::get_sumextra (Xapian::termcount *doclen*) const [virtual]

Calculate the term-independent weight component for a document.

The parameter gives information about the document which may be used in the calculations:

Parameters:

doclen The document's length (unnormalised).

Implements Xapian::Weight.

6.2.3.7 Xapian::weight Xapian::BoolWeight::get_maxextra () const [virtual]

Return an upper bound on what get_sumextra() can return for any document.

This information is used by the matcher to perform various optimisations, so strive to make the bound as tight as possible.

Implements Xapian::Weight.

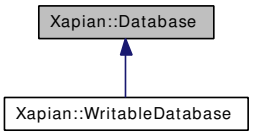
The documentation for this class was generated from the following file:

- include/xapian/weight.h

6.3 Xapian::Database Class Reference

`#include <database.h>`

Inheritance diagram for Xapian::Database:



Public Member Functions

- void add_database (const Database &database)
- Database ()
- Database (const std::string &path)
- virtual ~Database ()
- Database (const Database &other)
- void operator= (const Database &other)
- void reopen ()
- virtual void close ()
- virtual std::string get_description () const

  Return a string describing this object.

- PostingIterator postlist_begin (const std::string &tname) const
- PostingIterator postlist_end (const std::string &) const
- TermIterator termlist_begin (Xapian::docid did) const
- TermIterator termlist_end (Xapian::docid) const
- bool has_positions () const
- PositionIterator positionlist_begin (Xapian::docid did, const std::string &tname) const
- PositionIterator positionlist_end (Xapian::docid, const std::string &) const
- TermIterator allterms_begin () const
- TermIterator allterms_end () const
- TermIterator allterms_begin (const std::string &prefix) const
- TermIterator allterms_end (const std::string &) const
- Xapian::doccount get_doccount () const

  Get the number of documents in the database.

- Xapian::docid get_lastdocid () const

  Get the highest document id which has been used in the database.

- Xapian::doclength get_avlength () const

  Get the average length of the documents in the database.

- Xapian::doccount get_termfreq (const std::string &tname) const

  Get the number of documents in the database indexed by a given term.

- bool term_exists (const std::string &tname) const
- Xapian::termcount get_collection_freq (const std::string &tname) const
- Xapian::doccount get_value_freq (Xapian::valueno valno) const
- std::string get_value_lower_bound (Xapian::valueno valno) const
- std::string get_value_upper_bound (Xapian::valueno valno) const
- Xapian::termcount get_doclength_lower_bound () const
- Xapian::termcount get_doclength_upper_bound () const

  Get an upper bound on the length of a document in this DB.

- Xapian::termcount get_wdf_upper_bound (const std::string &term) const

  Get an upper bound on the wdf of term term.

- ValueIterator valuestream_begin (Xapian::valueno slot) const

  Return an iterator over the value in slot slot *for each document.*

- ValueIteratorEnd_ valuestream_end (Xapian::valueno) const

  Return end iterator corresponding to valuestream_begin().

- Xapian::termcount get_doclength (Xapian::docid did) const

  Get the length of a document.

- void keep_alive ()
- Xapian::Document get_document (Xapian::docid did) const
- std::string get_spelling_suggestion (const std::string &word, unsigned max_edit_distance=2) const
- Xapian::TermIterator spellings_begin () const
- Xapian::TermIterator spellings_end () const

  Corresponding end iterator to spellings_begin().

- Xapian::TermIterator synonyms_begin (const std::string &term) const
- Xapian::TermIterator synonyms_end (const std::string &) const

  Corresponding end iterator to synonyms_begin(term).

- Xapian::TermIterator synonym_keys_begin (const std::string &prefix=std::string()) const
- Xapian::TermIterator synonym_keys_end (const std::string &=std::string()) const

  Corresponding end iterator to synonym_keys_begin(prefix).

- std::string get_metadata (const std::string &key) const
- Xapian::TermIterator metadata_keys_begin (const std::string &prefix=std::string()) const
- Xapian::TermIterator metadata_keys_end (const std::string &=std::string()) const

Corresponding end iterator to metadata_keys_begin().

- std::string get_uuid () const

6.3.1 Detailed Description

This class is used to access a database, or a group of databases.

For searching, this class is used in conjunction with an Enquire object.

Exceptions:

InvalidArgumentError will be thrown if an invalid argument is supplied, for example, an unknown database type.

DatabaseOpeningError may be thrown if the database cannot be opened (for example, a required file cannot be found).

DatabaseVersionError may be thrown if the database is in an unsupported format (for example, created by a newer version of Xapian which uses an incompatible format).

6.3.2 Constructor & Destructor Documentation

6.3.2.1 Xapian::Database::Database ()

Create a Database with no databases in.

6.3.2.2 Xapian::Database::Database (const std::string & *path*) [explicit]

Open a Database, automatically determining the database backend to use.

Parameters:

path directory that the database is stored in.

6.3.2.3 virtual Xapian::Database::~Database () [virtual]

Destroy this handle on the database.

If there are no copies of this object remaining, the database(s) will be closed.

6.3.2.4 Xapian::Database::Database (const Database & *other*)

Copying is allowed. The internals are reference counted, so copying is cheap.

6.3.3 Member Function Documentation

6.3.3.1 void Xapian::Database::add_database (const Database & *database*)

Add an existing database (or group of databases) to those accessed by this object.

Parameters:

database the database(s) to add.

6.3.3.2 void Xapian::Database::operator= (const Database & *other*)

Assignment is allowed. The internals are reference counted, so assignment is cheap.

6.3.3.3 void Xapian::Database::reopen ()

Re-open the database.

This re-opens the database(s) to the latest available version(s). It can be used either to make sure the latest results are returned, or to recover from a Xapian::DatabaseModifiedError.

Calling reopen() on a database which has been closed (with *close()*) will always raise a Xapian::DatabaseError.

6.3.3.4 virtual void Xapian::Database::close () [virtual]

Close the database.

This closes the database and releases all file handles held by the database.

This is a permanent close of the database: calling reopen() after closing a database will not reopen it, and will raise an exception.

Calling close() on a database which is already closed has no effect (and doesn't raise an exception).

After this call, calls made to methods of the database (other than close() or the destructor), or to objects associated with the database will behave in one of the following ways (but which behaviour happens may vary between releases, and between database backends):

- raise a Xapian::DatabaseError indicating that the database is closed.
- behave exactly as they would have done if the database had not been closed (by using cached data).

To summarise - you should not rely on the exception being raised, or the normal result being available, but if you do get a result, it will be correct.

6.3.3.5 virtual std::string Xapian::Database::get_description () const [virtual]

Return a string describing this object.

Reimplemented in Xapian::WritableDatabase.

6.3.3.6 PostingIterator Xapian::Database::postlist_begin (const std::string & *tname*) const

An iterator pointing to the start of the postlist for a given term.

If the term name is the empty string, the iterator returned will list all the documents in the database. Such an iterator will always return a WDF value of 1, since there is no obvious meaning for this quantity in this case.

6.3.3.7 PostingIterator Xapian::Database::postlist_end (const std::string &) const [inline]

Corresponding end iterator to postlist_begin().

6.3.3.8 TermIterator Xapian::Database::termlist_begin (Xapian::docid *did*) const

An iterator pointing to the start of the termlist for a given document.

6.3.3.9 TermIterator Xapian::Database::termlist_end (Xapian::docid) const [inline]

Corresponding end iterator to termlist_begin().

6.3.3.10 bool Xapian::Database::has_positions () const

Does this database have any positional information?

6.3.3.11 PositionIterator Xapian::Database::positionlist_begin (Xapian::docid *did*, const std::string & *tname*) const

An iterator pointing to the start of the position list for a given term in a given document.

6.3.3.12 PositionIterator Xapian::Database::positionlist_end (Xapian::docid, const std::string &) const [inline]

Corresponding end iterator to positionlist_begin().

6.3.3.13 TermIterator Xapian::Database::allterms_begin () const

An iterator which runs across all terms in the database.

6.3.3.14 TermIterator Xapian::Database::allterms_end () const [inline]

Corresponding end iterator to allterms_begin().

6.3.3.15 TermIterator Xapian::Database::allterms_begin (const std::string & *prefix*) const

An iterator which runs across all terms with a given prefix.

This is functionally similar to getting an iterator with allterms_begin() and then calling skip_to(prefix) on that iterator to move to the start of the prefix, but is more convenient (because it detects the end of the prefixed terms), and may be more efficient than simply calling skip_to() after opening the iterator, particularly for network databases.

Parameters:

prefix The prefix to restrict the returned terms to.

6.3.3.16 TermIterator Xapian::Database::allterms_end (const std::string &) const [inline]

Corresponding end iterator to allterms_begin(prefix).

6.3.3.17 Xapian::doccount Xapian::Database::get_doccount () const

Get the number of documents in the database.

6.3.3.18 Xapian::docid Xapian::Database::get_lastdocid () const

Get the highest document id which has been used in the database.

6.3.3.19 Xapian::doclength Xapian::Database::get_avlength () const

Get the average length of the documents in the database.

6.3.3.20 Xapian::doccount Xapian::Database::get_termfreq (const std::string & *tname*) const

Get the number of documents in the database indexed by a given term.

6.3.3.21 bool Xapian::Database::term_exists (const std::string & *tname*) const

Check if a given term exists in the database.

Return true if and only if the term exists in the database. This is the same as (get_-termfreq(tname) != 0), but will often be more efficient.

6.3.3.22 Xapian::termcount Xapian::Database::get_collection_freq (const std::string & *tname*) const

Return the total number of occurrences of the given term.

This is the sum of the number of occurrences of the term in each document it indexes: i.e., the sum of the within document frequencies of the term.

Parameters:

tname The term whose collection frequency is being requested.

6.3.3.23 Xapian::doccount Xapian::Database::get_value_freq (Xapian::valueno *valno*) const

Return the frequency of a given value slot.

This is the number of documents which have a (non-empty) value stored in the slot.

Parameters:

valno The value slot to examine.

Exceptions:

UnimplementedError The frequency of the value isn't available for this database type.

6.3.3.24 std::string Xapian::Database::get_value_lower_bound (Xapian::valueno *valno*) const

Get a lower bound on the values stored in the given value slot.

If there are no values stored in the given value slot, this will return an empty string.

If the lower bound isn't available for the given database type, this will return the lowest possible bound - the empty string.

Parameters:

valno The value slot to examine.

6.3.3.25 std::string Xapian::Database::get_value_upper_bound (Xapian::valueno *valno*) const

Get an upper bound on the values stored in the given value slot.

If there are no values stored in the given value slot, this will return an empty string.

Parameters:

valno The value slot to examine.

Exceptions:

UnimplementedError The upper bound of the values isn't available for this database type.

6.3.3.26 Xapian::termcount Xapian::Database::get_doclength_lower_bound () const

Get a lower bound on the length of a document in this DB.

This bound does not include any zero-length documents.

6.3.3.27 Xapian::termcount Xapian::Database::get_doclength_upper_bound () const

Get an upper bound on the length of a document in this DB.

6.3.3.28 Xapian::termcount Xapian::Database::get_wdf_upper_bound (const std::string & *term*) const

Get an upper bound on the wdf of term *term*.

6.3.3.29 ValueIterator Xapian::Database::valuestream_begin (Xapian::valueno *slot*) const

Return an iterator over the value in slot *slot* for each document.

6.3.3.30 ValueIteratorEnd_ Xapian::Database::valuestream_end (Xapian::valueno) const [inline]

Return end iterator corresponding to valuestream_begin().

6.3.3.31 Xapian::termcount Xapian::Database::get_doclength (Xapian::docid *did*) const

Get the length of a document.

6.3.3.32 void Xapian::Database::keep_alive ()

Send a "keep-alive" to remote databases to stop them timing out.

Has no effect on non-remote databases.

6.3.3.33 Xapian::Document Xapian::Database::get_document (Xapian::docid *did*) const

Get a document from the database, given its document id.

This method returns a Xapian::Document object which provides the information about a document.

Parameters:

did The document id of the document to retrieve.

Returns:

A Xapian::Document object containing the document data

Exceptions:

Xapian::DocNotFoundError The document specified could not be found in the database.

6.3.3.34 std::string Xapian::Database::get_spelling_suggestion (const std::string & *word*, unsigned *max_edit_distance* = 2) const

Suggest a spelling correction.

Parameters:

word The potentially misspelled word.

max_edit_distance Only consider words which are at most *max_edit_distance* edits from *word*. An edit is a character insertion, deletion, or the transposition of two adjacent characters (default is 2).

6.3.3.35 Xapian::TermIterator Xapian::Database::spellings_begin () const

An iterator which returns all the spelling correction targets.

This returns all the words which are considered as targets for the spelling correction algorithm. The frequency of each word is available as the term frequency of each entry in the returned iterator.

6.3.3.36 Xapian::TermIterator Xapian::Database::spellings_end () const [inline]

Corresponding end iterator to spellings_begin().

6.3.3.37 Xapian::TermIterator Xapian::Database::synonyms_begin (const std::string & *term*) const

An iterator which returns all the synonyms for a given term.

Parameters:

term The term to return synonyms for.

6.3.3.38 Xapian::TermIterator Xapian::Database::synonyms_end (const std::string &) const [inline]

Corresponding end iterator to synonyms_begin(term).

6.3.3.39 Xapian::TermIterator Xapian::Database::synonym_keys_begin (const std::string & *prefix* = std::string()) const

An iterator which returns all terms which have synonyms.

Parameters:

prefix If non-empty, only terms with this prefix are returned.

6.3.3.40 Xapian::TermIterator Xapian::Database::synonym_keys_end (const std::string & = std::string()) const [inline]

Corresponding end iterator to synonym_keys_begin(prefix).

6.3.3.41 std::string Xapian::Database::get_metadata (const std::string & *key*) const

Get the user-specified metadata associated with a given key.

User-specified metadata allows you to store arbitrary information in the form of (key,tag) pairs. See *WritableDatabase::set_metadata()* for more information.

When invoked on a Xapian::Database object representing multiple databases, currently only the metadata for the first is considered but this behaviour may change in the future.

If there is no piece of metadata associated with the specified key, an empty string is returned (this applies even for backends which don't support metadata).

Empty keys are not valid, and specifying one will cause an exception.

Parameters:

key The key of the metadata item to access.

Returns:

The retrieved metadata item's value.

Exceptions:

Xapian::InvalidArgumentError will be thrown if the key supplied is empty.

6.3.3.42 Xapian::TermIterator Xapian::Database::metadata_keys_begin (const std::string & *prefix* = `std::string()`) const

An iterator which returns all user-specified metadata keys.

When invoked on a Xapian::Database object representing multiple databases, currently only the metadata for the first is considered but this behaviour may change in the future.

If the backend doesn't support metadata, then this method returns an iterator which compares equal to that returned by metadata_keys_end().

Parameters:

prefix If non-empty, only keys with this prefix are returned.

Exceptions:

Xapian::UnimplementedError will be thrown if the backend implements user-specified metadata, but doesn't implement iterating its keys (currently this happens for the InMemory backend).

6.3.3.43 Xapian::TermIterator Xapian::Database::metadata_keys_end (const std::string & = `std::string()`) const `[inline]`

Corresponding end iterator to metadata_keys_begin().

6.3.3.44 std::string Xapian::Database::get_uuid () const

Get a UUID for the database.

The UUID will persist for the lifetime of the database.

Replicas (eg, made with the replication protocol, or by copying all the database files) will have the same UUID. However, copies (made with copydatabase, or xapian-compact) will have different UUIDs.

If the backend does not support UUIDs or this database has no subdatabases, the UUID will be empty.

If this database has multiple sub-databases, the UUID string will contain the UUIDs of all the sub-databases.

The documentation for this class was generated from the following file:

- include/xapian/database.h

6.4 Xapian::DatabaseMaster Class Reference

`#include <replication.h>`

Public Member Functions

- DatabaseMaster (const std::string &path_)
- void write_changesets_to_fd (int fd, const std::string &start_revision, ReplicationInfo *info) const
- std::string get_description () const

  Return a string describing this object.

6.4.1 Detailed Description

Access to a master database for replication.

Warning: the replication interface is currently experimental, and is liable to change between releases without warning.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 Xapian::DatabaseMaster::DatabaseMaster (const std::string & *path_*) [inline]

Create a new DatabaseMaster for the database at the specified path.

The database isn't actually opened until a set of changesets is requested.

6.4.3 Member Function Documentation

6.4.3.1 void Xapian::DatabaseMaster::write_changesets_to_fd (int *fd*, const std::string & *start_revision*, ReplicationInfo * *info*) const

Write a set of changesets for upgrading the database to a file.

The changesets will be such that, if they are applied in order to a copy of the database at the start revision, a copy of the database at the current revision (i.e. the revision which the database object is currently open at) will be produced.

If suitable changesets have been stored in the database, this will write the appropriate changesets, in order. If suitable changesets are not available, this will write a copy of sufficient blocks of the database to reconstruct the current revision.

This will therefore potentially write a very large amount of data to the file descriptor.

Parameters:

fd An open file descriptor to write the changes to.

start_revision The starting revision of the database that the changesets are to be applied to. Specify an empty string to get a "creation" changeset, which includes the creation of the database. The revision will include the unique identifier for the database, if one is available.

info If non-NULL, the supplied structure will be updated to reflect the changes written to the file descriptor.

6.4.3.2 std::string Xapian::DatabaseMaster::get_description () const

Return a string describing this object.

The documentation for this class was generated from the following file:

- include/xapian/replication.h

6.5 Xapian::DatabaseReplica Class Reference

`#include <replication.h>`

Public Member Functions

- DatabaseReplica (const DatabaseReplica &other)

  Copying is allowed (and is cheap).

- void operator= (const DatabaseReplica &other)

  Assignment is allowed (and is cheap).

- DatabaseReplica ()

  Default constructor - for declaring an uninitialised replica.

- ~DatabaseReplica ()

  Destructor.

- DatabaseReplica (const std::string &path)
- std::string get_revision_info () const
- void set_read_fd (int fd)
- bool apply_next_changeset (ReplicationInfo *info)
- void close ()
- std::string get_description () const

  Return a string describing this object.

6.5.1 Detailed Description

Access to a database replica, for applying replication to it.

Warning: the replication interface is currently experimental, and is liable to change between releases without warning.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 Xapian::DatabaseReplica::DatabaseReplica (const DatabaseReplica & *other*)

Copying is allowed (and is cheap).

6.5.2.2 Xapian::DatabaseReplica::DatabaseReplica ()

Default constructor - for declaring an uninitialised replica.

6.5.2.3 Xapian::DatabaseReplica::~DatabaseReplica ()

Destructor.

6.5.2.4 Xapian::DatabaseReplica::DatabaseReplica (const std::string & *path*)

Open a DatabaseReplica for the database at the specified path.

The path should either point to a database previously created by a DatabaseReplica, or to a path which doesn't yet exist.

The path should always be in a directory which exists.

If the specified path does not contain a database, a database will be created when an appropriate changeset is supplied to the replica.

Parameters:

path The path to make the replica at.

6.5.3 Member Function Documentation

6.5.3.1 void Xapian::DatabaseReplica::operator= (const DatabaseReplica & *other*)

Assignment is allowed (and is cheap).

6.5.3.2 std::string Xapian::DatabaseReplica::get_revision_info () const

Get a string describing the current revision of the replica.

The revision information includes a unique identifier for the master database that the replica is of, as well as information about the exact revision of the master database that the replica represents. This information allows the master database to send the appropriate changeset to mirror whatever changes have been made on the master.

6.5.3.3 void Xapian::DatabaseReplica::set_read_fd (int *fd*)

Set the file descriptor to read changesets from.

This will be remembered in the DatabaseReplica, but the caller is still responsible for closing it after it is finished with.

Parameters:

fd The file descriptor to read the changeset from.

6.5.3.4 bool Xapian::DatabaseReplica::apply_next_changeset (ReplicationInfo * *info*)

Read and apply the next changeset.

If no changesets are found on the file descriptor, returns false immediately.

If any changesets are found on the file descriptor, exactly one of them is applied.

A common way to use this method is to call it repeatedly until it returns false, with an appropriate gap between each call.

Information beyond the end of the next changeset may be read from the file descriptor and cached in the DatabaseReplica object. Therefore, the file descriptor shouldn't be accessed by any other external code, since it will be in an indeterminate state.

Note that if this raises an exception (other than DatabaseCorruptError) the database will be left in a valid and consistent state. It may or may not be changed from its initial state, and may or may not be fully synchronised with the master database.

Parameters:

info If non-NULL, the supplied structure will be updated to reflect the changes read from the file descriptor.

Returns:

true if there are more changesets to apply on the file descriptor, false otherwise.

6.5.3.5 void Xapian::DatabaseReplica::close ()

Close the DatabaseReplica.

After this has been called, there will no longer be a write lock on the database created by the DatabaseReplica, and if any of the methods of this object which access the database are called, they will throw an InvalidOperationError.

6.5.3.6 std::string Xapian::DatabaseReplica::get_description () const

Return a string describing this object.

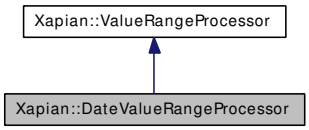
The documentation for this class was generated from the following file:

- include/xapian/replication.h

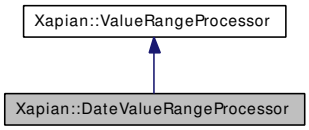
6.6 Xapian::DateValueRangeProcessor Class Reference

`#include <queryparser.h>`

Inheritance diagram for Xapian::DateValueRangeProcessor:



Collaboration diagram for Xapian::DateValueRangeProcessor:



Public Member Functions

- DateValueRangeProcessor (Xapian::valueno valno_, bool prefer_mdy_=false, int epoch_year_=1970)
- Xapian::valueno operator() (std::string &begin, std::string &end)

6.6.1 Detailed Description

Handle a date range.

Begin and end must be dates in a recognised format.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 Xapian::DateValueRangeProcessor::DateValueRangeProcessor (Xapian::valueno *valno_*, bool *prefer_mdy_* = `false`, int *epoch_year_* = `1970`) `[inline]`

Constructor.

Parameters:

valno_ The value number to return from operator().

prefer_mdy_ Should ambiguous dates be interpreted as month/day/year rather than day/month/year? (default: false)

epoch_year_ Year to use as the epoch for dates with 2 digit years (default: 1970, so 1/1/69 is 2069 while 1/1/70 is 1970).

6.6.3 Member Function Documentation

6.6.3.1 Xapian::valueno Xapian::DateValueRangeProcessor::operator() (std::string & *begin*, std::string & *end*) [virtual]

See if <begin>..<end> is a valid date value range.

If <begin>..<end> is a sensible date range, this method returns the value number of range filter on. Otherwise it returns Xapian::BAD_VALUENO.

Implements Xapian::ValueRangeProcessor.

The documentation for this class was generated from the following file:

- include/xapian/queryparser.h

6.7 Xapian::Document Class Reference

A document in the database - holds data, values, terms, and postings.

`#include <document.h>`

Public Member Functions

- Document (const Document &other)
- void operator= (const Document &other)
- Document ()

  Make a new empty Document.

- ~Document ()

  Destructor.

- std::string get_value (Xapian::valueno valueno) const
- void add_value (Xapian::valueno valueno, const std::string &value)
- void remove_value (Xapian::valueno valueno)

  Remove any value with the given number.

- void clear_values ()

  Remove all values associated with the document.

- std::string get_data () const
- void set_data (const std::string &data)

  Set data stored in the document.

- void add_posting (const std::string &tname, Xapian::termpos tpos, Xapian::termcount wdfinc=1)
- void add_term (const std::string &tname, Xapian::termcount wdfinc=1)
- void remove_posting (const std::string &tname, Xapian::termpos tpos, Xapian::termcount wdfdec=1)
- void remove_term (const std::string &tname)
- void clear_terms ()

  Remove all terms (and postings) from the document.

- Xapian::termcount termlist_count () const
- TermIterator termlist_begin () const

  Iterator for the terms in this document.

- TermIterator termlist_end () const

  Equivalent end iterator for termlist_begin().

- Xapian::termcount values_count () const

  Count the values in this document.

- ValueIterator values_begin () const

  Iterator for the values in this document.

- ValueIteratorEnd_ values_end () const

  Equivalent end iterator for values_begin().

- docid get_docid () const
- std::string serialise () const
- std::string get_description () const

  Return a string describing this object.

Static Public Member Functions

- static Document unserialise (const std::string &s)

6.7.1 Detailed Description

A document in the database - holds data, values, terms, and postings.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 Xapian::Document::Document (const Document & *other*)

Copying is allowed. The internals are reference counted, so copying is cheap.

6.7.2.2 Xapian::Document::Document ()

Make a new empty Document.

6.7.2.3 Xapian::Document::~Document ()

Destructor.

6.7.3 Member Function Documentation

6.7.3.1 void Xapian::Document::operator= (const Document & *other*)

Assignment is allowed. The internals are reference counted, so assignment is cheap.

6.7.3.2 std::string Xapian::Document::get_value (Xapian::valueno *valueno*) const

Get value by number.

Returns an empty string if no value with the given number is present in the document.

Parameters:

valueno The number of the value.

6.7.3.3 void Xapian::Document::add_value (Xapian::valueno *valueno*, const std::string & *value*)

Add a new value.

The new value will replace any existing value with the same number (or if the new value is empty, it will remove any existing value with the same number).

6.7.3.4 void Xapian::Document::remove_value (Xapian::valueno *valueno*)

Remove any value with the given number.

6.7.3.5 void Xapian::Document::clear_values ()

Remove all values associated with the document.

6.7.3.6 std::string Xapian::Document::get_data () const

Get data stored in the document. This is a potentially expensive operation, and shouldn't normally be used in a match decider functor. Put data for use by match deciders in a value instead.

6.7.3.7 void Xapian::Document::set_data (const std::string & *data*)

Set data stored in the document.

6.7.3.8 void Xapian::Document::add_posting (const std::string & *tname*, Xapian::termpos *tpos*, Xapian::termcount *wdfinc* = 1)

Add an occurrence of a term at a particular position.

Multiple occurrences of the term at the same position are represented only once in the positional information, but do increase the wdf.

If the term is not already in the document, it will be added to it.

Parameters:

tname The name of the term.

tpos The position of the term.

wdfinc The increment that will be applied to the wdf for this term.

6.7.3.9 void Xapian::Document::add_term (const std::string & *tname*, Xapian::termcount *wdfinc* = 1)

Add a term to the document, without positional information.

Any existing positional information for the term will be left unmodified.

Parameters:

tname The name of the term.

wdfinc The increment that will be applied to the wdf for this term.

6.7.3.10 void Xapian::Document::remove_posting (const std::string & *tname*, Xapian::termpos *tpos*, Xapian::termcount *wdfdec* = 1)

Remove a posting of a term from the document.

Note that the term will still index the document even if all occurrences are removed. To remove a term from a document completely, use remove_term().

Parameters:

tname The name of the term.

tpos The position of the term.

wdfdec The decrement that will be applied to the wdf when removing this posting. The wdf will not go below the value of 0.

Exceptions:

Xapian::InvalidArgumentError will be thrown if the term is not at the position specified in the position list for this term in this document.

Xapian::InvalidArgumentError will be thrown if the term is not in the document

6.7.3.11 void Xapian::Document::remove_term (const std::string & *tname*)

Remove a term and all postings associated with it.

Parameters:

tname The name of the term.

Exceptions:

Xapian::InvalidArgumentError will be thrown if the term is not in the document

6.7.3.12 void Xapian::Document::clear_terms ()

Remove all terms (and postings) from the document.

6.7.3.13 Xapian::termcount Xapian::Document::termlist_count () const

The length of the termlist - i.e. the number of different terms which index this document.

6.7.3.14 TermIterator Xapian::Document::termlist_begin () const

Iterator for the terms in this document.

6.7.3.15 TermIterator Xapian::Document::termlist_end () const [inline]

Equivalent end iterator for termlist_begin().

6.7.3.16 Xapian::termcount Xapian::Document::values_count () const

Count the values in this document.

6.7.3.17 ValueIterator Xapian::Document::values_begin () const

Iterator for the values in this document.

6.7.3.18 ValueIteratorEnd_ Xapian::Document::values_end () const [inline]

Equivalent end iterator for values_begin().

6.7.3.19 docid Xapian::Document::get_docid () const

Get the document id which is associated with this document (if any).

NB If multiple databases are being searched together, then this will be the document id in the individual database, not the merged database!

Returns:

If this document came from a database, return the document id in that database. Otherwise, return 0.

6.7.3.20 std::string Xapian::Document::serialise () const

Serialise document into a string.

The document representation may change between Xapian releases: even between minor versions. However, it is guaranteed not to change if the remote database protocol has not changed between releases.

6.7.3.21 static Document Xapian::Document::unserialise (const std::string & *s*) [static]

Unserialise a document from a string produced by serialise().

6.7.3.22 std::string Xapian::Document::get_description () const

Return a string describing this object.

The documentation for this class was generated from the following file:

- include/xapian/document.h

6.8 Xapian::Enquire Class Reference

`#include <enquire.h>`

Public Types

- enum **docid_order** { **ASCENDING** = 1, **DESCENDING** = 0, **DONT_CARE** = 2 }

Public Member Functions

- Enquire (const Enquire &other)

  Copying is allowed (and is cheap).

- void operator= (const Enquire &other)

  Assignment is allowed (and is cheap).

- Enquire (const Database &database, ErrorHandler *errorhandler_=0)
- ~Enquire ()
- void set_query (const Xapian::Query &query, Xapian::termcount qlen=0)
- const Xapian::Query & get_query () const
- void set_weighting_scheme (const Weight &weight_)
- void set_collapse_key (Xapian::valueno collapse_key, Xapian::doccount collapse_max=1)
- void set_docid_order (docid_order order)
- void set_cutoff (Xapian::percent percent_cutoff, Xapian::weight weight_cutoff=0)
- void set_sort_by_relevance ()
- void set_sort_by_value (Xapian::valueno sort_key, bool reverse)
- **XAPIAN_DEPRECATED** (void set_sort_by_value(Xapian::valueno sort_key))
- void set_sort_by_key (Xapian::Sorter *sorter, bool reverse)
- **XAPIAN_DEPRECATED** (void set_sort_by_key(Xapian::Sorter *sorter))
- void set_sort_by_value_then_relevance (Xapian::valueno sort_key, bool reverse)
- **XAPIAN_DEPRECATED** (void set_sort_by_value_then_relevance(Xapian::valueno sort_key))
- void set_sort_by_key_then_relevance (Xapian::Sorter *sorter, bool reverse)
- **XAPIAN_DEPRECATED** (void set_sort_by_key_then_relevance(Xapian::Sorter *sorter))
- void set_sort_by_relevance_then_value (Xapian::valueno sort_key, bool reverse)
- **XAPIAN_DEPRECATED** (void set_sort_by_relevance_then_value(Xapian::valueno sort_key))
- void set_sort_by_relevance_then_key (Xapian::Sorter *sorter, bool reverse)
- **XAPIAN_DEPRECATED** (void set_sort_by_relevance_then_key(Xapian::Sorter *sorter))

- MSet get_mset (Xapian::doccount first, Xapian::doccount maxitems, Xapian::doccount checkatleast=0, const RSet *omrset=0, const MatchDecider *mdecider=0, const MatchDecider *matchspy=0) const
- MSet **get_mset** (Xapian::doccount first, Xapian::doccount maxitems, const RSet *omrset, const MatchDecider *mdecider=0) const
- ESet get_eset (Xapian::termcount maxitems, const RSet &omrset, int flags=0, double k=1.0, const Xapian::ExpandDecider *edecider=0) const
- ESet get_eset (Xapian::termcount maxitems, const RSet &omrset, const Xapian::ExpandDecider *edecider) const
- TermIterator get_matching_terms_begin (Xapian::docid did) const
- TermIterator get_matching_terms_end (Xapian::docid) const
- TermIterator get_matching_terms_begin (const MSetIterator &it) const
- TermIterator get_matching_terms_end (const MSetIterator &) const
- std::string get_description () const

  Return a string describing this object.

Public Attributes

- Xapian::Internal::RefCntPtr< Internal > **internal**

Static Public Attributes

- static const int **INCLUDE_QUERY_TERMS** = 1
- static const int **USE_EXACT_TERMFREQ** = 2

6.8.1 Detailed Description

This class provides an interface to the information retrieval system for the purpose of searching.

Databases are usually opened lazily, so exceptions may not be thrown where you would expect them to be. You should catch Xapian::Error exceptions when calling any method in Xapian::Enquire.

Exceptions:

Xapian::InvalidArgumentError will be thrown if an invalid argument is supplied, for example, an unknown database type.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 Xapian::Enquire::Enquire (const Enquire & *other*)

Copying is allowed (and is cheap).

6.8.2.2 Xapian::Enquire::Enquire (const Database & *database*, ErrorHandler * *errorhandler_* = 0) [explicit]

Create a Xapian::Enquire object.

This specification cannot be changed once the Xapian::Enquire is opened: you must create a new Xapian::Enquire object to access a different database, or set of databases.

The database supplied must have been initialised (ie, must not be the result of calling the Database::Database() constructor). If you need to handle a situation where you have no index gracefully, a database created with InMemory::open() can be passed here, which represents a completely empty database.

Parameters:

database Specification of the database or databases to use.

errorhandler_ A pointer to the error handler to use. Ownership of the object pointed to is not assumed by the Xapian::Enquire object - the user should delete the Xapian::ErrorHandler object after the Xapian::Enquire object is deleted. To use no error handler, this parameter should be 0.

Exceptions:

Xapian::InvalidArgumentError will be thrown if an initialised Database object is supplied.

6.8.2.3 Xapian::Enquire::~Enquire ()

Close the Xapian::Enquire object.

6.8.3 Member Function Documentation

6.8.3.1 void Xapian::Enquire::operator= (const Enquire & *other*)

Assignment is allowed (and is cheap).

6.8.3.2 void Xapian::Enquire::set_query (const Xapian::Query & *query*, Xapian::termcount *qlen* = 0)

Set the query to run.

Parameters:

query the new query to run.

qlen the query length to use in weight calculations - by default the sum of the wqf of all terms is used.

6.8.3.3 const Xapian::Query& Xapian::Enquire::get_query () const

Get the query which has been set. This is only valid after set_query() has been called.

Exceptions:

Xapian::InvalidArgumentError will be thrown if query has not yet been set.

6.8.3.4 void Xapian::Enquire::set_weighting_scheme (const Weight & *weight_*)

Set the weighting scheme to use for queries.

Parameters:

weight_ the new weighting scheme. If no weighting scheme is specified, the default is BM25 with the default parameters.

6.8.3.5 void Xapian::Enquire::set_collapse_key (Xapian::valueno *collapse_key*, Xapian::doccount *collapse_max* = 1)

Set the collapse key to use for queries.

Parameters:

collapse_key value number to collapse on - at most one MSet entry with each particular value will be returned (default is Xapian::BAD_VALUENO which means no collapsing).

collapse_max Max number of items with the same key to leave after collapsing (default 1).

The MSet returned by get_mset() will have only the "best" (at most) *collapse_max* entries with each particular value of *collapse_key* ("best" being highest ranked - i.e. highest weight or highest sorting key).

An example use might be to create a value for each document containing an MD5 hash of the document contents. Then duplicate documents from different sources can be eliminated at search time by collapsing with *collapse_max* = 1 (it's better to eliminate duplicates at index time, but this may not be always be possible - for example the search may be over more than one Xapian database).

Another use is to group matches in a particular category (e.g. you might collapse a mailing list search on the Subject: so that there's only one result per discussion thread). In this case you can use get_collapse_count() to give the user some idea how many other results there are. And if you index the Subject: as a boolean term as well as putting it in a value, you can offer a link to a non-collapsed search restricted to that thread using a boolean filter.

6.8.3.6 void Xapian::Enquire::set_docid_order (docid_order *order*)

Set the direction in which documents are ordered by document id in the returned MSet.

This order only has an effect on documents which would otherwise have equal rank. For a weighted probabilistic match with no sort value, this means documents with equal weight. For a boolean match, with no sort value, this means all documents. And if a sort value is used, this means documents with equal sort value (and also equal weight if ordering on relevance after the sort).

Parameters:

order This can be:

- Xapian::Enquire::ASCENDING docids sort in ascending order (default)
- Xapian::Enquire::DESCENDING docids sort in descending order
- Xapian::Enquire::DONT_CARE docids sort in whatever order is most efficient for the backend

Note: If you add documents in strict date order, then a boolean search - i.e. set_weighting_scheme(Xapian::BoolWeight()) - with set_docid_order(Xapian::Enquire::DESCENDING) is a very efficient way to perform "sort by date, newest first".

6.8.3.7 void Xapian::Enquire::set_cutoff (Xapian::percent *percent_cutoff*, Xapian::weight *weight_cutoff* = 0)

Set the percentage and/or weight cutoffs.

Parameters:

percent_cutoff Minimum percentage score for returned documents. If a document has a lower percentage score than this, it will not appear in the MSet. If your intention is to return only matches which contain all the terms in the query, then it's more efficient to use Xapian::Query::OP_AND instead of Xapian::Query::OP_OR in the query than to use set_cutoff(100). (default 0 => no percentage cut-off).

weight_cutoff Minimum weight for a document to be returned. If a document has a lower score that this, it will not appear in the MSet. It is usually only possible to choose an appropriate weight for cutoff based on the results of a previous run of the same query; this is thus mainly useful for alerting operations. The other potential use is with a user specified weighting scheme. (default 0 => no weight cut-off).

6.8.3.8 void Xapian::Enquire::set_sort_by_relevance ()

Set the sorting to be by relevance only.

This is the default.

6.8.3.9 void Xapian::Enquire::set_sort_by_value (Xapian::valueno *sort_key*, bool *reverse*)

Set the sorting to be by value only.

NB sorting of values uses a string comparison, so you'll need to store numbers padded with leading zeros or spaces, or with the number of digits prepended.

Parameters:

sort_key value number to sort on.

reverse If true, reverses the sort order.

6.8.3.10 void Xapian::Enquire::set_sort_by_key (Xapian::Sorter * *sorter*, bool *reverse*)

Set the sorting to be by key generated from values only.

Parameters:

sorter The functor to use for generating keys.

reverse If true, reverses the sort order.

6.8.3.11 void Xapian::Enquire::set_sort_by_value_then_relevance (Xapian::valueno *sort_key*, bool *reverse*)

Set the sorting to be by value, then by relevance for documents with the same value.

NB sorting of values uses a string comparison, so you'll need to store numbers padded with leading zeros or spaces, or with the number of digits prepended.

Parameters:

sort_key value number to sort on.

reverse If true, reverses the sort order.

6.8.3.12 void Xapian::Enquire::set_sort_by_key_then_relevance (Xapian::Sorter * *sorter*, bool *reverse*)

Set the sorting to be by keys generated from values, then by relevance for documents with identical keys.

Parameters:

sorter The functor to use for generating keys.

reverse If true, reverses the sort order.

6.8.3.13 void Xapian::Enquire::set_sort_by_relevance_then_value (Xapian::valueno *sort_key*, bool *reverse*)

Set the sorting to be by relevance then value.

NB sorting of values uses a string comparison, so you'll need to store numbers padded with leading zeros or spaces, or with the number of digits prepended.

Note that with the default BM25 weighting scheme parameters, non-identical documents will rarely have the same weight, so this setting will give very similar results to set_sort_by_relevance(). It becomes more useful with particular BM25 parameter settings (e.g. BM25Weight(1,0,1,0,0)) or custom weighting schemes.

Parameters:

sort_key value number to sort on.

reverse If true, reverses the sort order.

6.8.3.14 void Xapian::Enquire::set_sort_by_relevance_then_key (Xapian::Sorter * *sorter*, bool *reverse*)

Set the sorting to be by relevance, then by keys generated from values.

Note that with the default BM25 weighting scheme parameters, non-identical documents will rarely have the same weight, so this setting will give very similar results to set_sort_by_relevance(). It becomes more useful with particular BM25 parameter settings (e.g. BM25Weight(1,0,1,0,0)) or custom weighting schemes.

Parameters:

sorter The functor to use for generating keys.

reverse If true, reverses the sort order.

6.8.3.15 MSet Xapian::Enquire::get_mset (Xapian::doccount *first*, Xapian::doccount *maxitems*, Xapian::doccount *checkatleast* = 0, const RSet * *omrset* = 0, const MatchDecider * *mdecider* = 0, const MatchDecider * *matchspy* = 0) const

Get (a portion of) the match set for the current query.

Parameters:

first the first item in the result set to return. A value of zero corresponds to the first item returned being that with the highest score. A value of 10 corresponds to the first 10 items being ignored, and the returned items starting at the eleventh.

maxitems the maximum number of items to return. If you want all matches, then you can pass the result of calling get_doccount() on the Database object

(though if you are doing this so you can filter results, you are likely to get much better performance by using Xapian's match-time filtering features instead). You can pass 0 for maxitems which will give you an empty MSet with valid statistics (such as get_matches_estimated()) calculated without looking at any postings, which is very quick, but means the estimates may be more approximate and the bounds may be much looser.

checkatleast the minimum number of items to check. Because the matcher optimises, it won't consider every document which might match, so the total number of matches is estimated. Setting checkatleast forces it to consider at least this many matches and so allows for reliable paging links.

omrset the relevance set to use when performing the query.

mdecider a decision functor to use to decide whether a given document should be put in the MSet.

matchspy a decision functor to use to decide whether a given document should be put in the MSet. The matchspy is applied to every document which is a potential candidate for the MSet, so if there are checkatleast or more such documents, the matchspy will see at least checkatleast. The mdecider is assumed to be a relatively expensive test so may be applied in a lazier fashion.

Returns:

A Xapian::MSet object containing the results of the query.

Exceptions:

Xapian::InvalidArgumentError See class documentation.

6.8.3.16 ESet Xapian::Enquire::get_eset (Xapian::termcount *maxitems*, const RSet & *omrset*, int *flags* = 0, double *k* = 1.0, const Xapian::ExpandDecider * *edecider* = 0) const

Get the expand set for the given rset.

Parameters:

maxitems the maximum number of items to return.

omrset the relevance set to use when performing the expand operation.

flags zero or more of these values |-ed together:

- Xapian::Enquire::INCLUDE_QUERY_TERMS query terms may be returned from expand
- Xapian::Enquire::USE_EXACT_TERMFREQ for multi dbs, calculate the exact termfreq; otherwise an approximation is used which can greatly improve efficiency, but still returns good results.

k the parameter k in the query expansion algorithm (default is 1.0)

edecider a decision functor to use to decide whether a given term should be put in the ESet

Returns:

An ESet object containing the results of the expand.

Exceptions:

Xapian::InvalidArgumentError See class documentation.

6.8.3.17 ESet Xapian::Enquire::get_eset (Xapian::termcount *maxitems*, const RSet & *omrset*, const Xapian::ExpandDecider * *edecider*) const [inline]

Get the expand set for the given rset.

Parameters:

maxitems the maximum number of items to return.

omrset the relevance set to use when performing the expand operation.

edecider a decision functor to use to decide whether a given term should be put in the ESet

Returns:

An ESet object containing the results of the expand.

Exceptions:

Xapian::InvalidArgumentError See class documentation.

6.8.3.18 TermIterator Xapian::Enquire::get_matching_terms_begin (Xapian::docid *did*) const

Get terms which match a given document, by document id.

This method returns the terms in the current query which match the given document.

It is possible for the document to have been removed from the database between the time it is returned in an MSet, and the time that this call is made. If possible, you should specify an MSetIterator instead of a Xapian::docid, since this will enable database backends with suitable support to prevent this occurring.

Note that a query does not need to have been run in order to make this call.

Parameters:

did The document id for which to retrieve the matching terms.

Returns:

An iterator returning the terms which match the document. The terms will be returned (as far as this makes any sense) in the same order as the terms in the query. Terms will not occur more than once, even if they do in the query.

Exceptions:

Xapian::InvalidArgumentError See class documentation.

Xapian::DocNotFoundError The document specified could not be found in the database.

6.8.3.19 TermIterator Xapian::Enquire::get_matching_terms_end (Xapian::docid) const [inline]

End iterator corresponding to get_matching_terms_begin()

6.8.3.20 TermIterator Xapian::Enquire::get_matching_terms_begin (const MSetIterator & *it*) const

Get terms which match a given document, by match set item.

This method returns the terms in the current query which match the given document.

If the underlying database has suitable support, using this call (rather than passing a Xapian::docid) will enable the system to ensure that the correct data is returned, and that the document has not been deleted or changed since the query was performed.

Parameters:

it The iterator for which to retrieve the matching terms.

Returns:

An iterator returning the terms which match the document. The terms will be returned (as far as this makes any sense) in the same order as the terms in the query. Terms will not occur more than once, even if they do in the query.

Exceptions:

Xapian::InvalidArgumentError See class documentation.

Xapian::DocNotFoundError The document specified could not be found in the database.

6.8.3.21 TermIterator Xapian::Enquire::get_matching_terms_end (const MSetIterator &) const [inline]

End iterator corresponding to get_matching_terms_begin()

6.8.3.22 std::string Xapian::Enquire::get_description () const

Return a string describing this object.

The documentation for this class was generated from the following file:

- include/xapian/enquire.h

6.9 Xapian::ErrorHandler Class Reference

`#include <errorhandler.h>`

Public Member Functions

- ErrorHandler ()

  Default constructor.

- virtual ~ErrorHandler ()

  We require a virtual destructor because we have virtual methods.

- void operator() (Xapian::Error &error)

6.9.1 Detailed Description

Decide if a Xapian::Error exception should be ignored.

You can create your own subclass of this class and pass in an instance of it when you construct a Xapian::Enquire object. Xapian::Error exceptions which happen during the match process are passed to this object and it can decide whether they should propagate or whether Enquire should attempt to continue.

The motivation is to allow searching over remote databases to handle a remote server which has died (both to allow results to be returned, and also so that such errors can be logged and dead servers temporarily removed from use).

6.9.2 Constructor & Destructor Documentation

6.9.2.1 Xapian::ErrorHandler::ErrorHandler () `[inline]`

Default constructor.

6.9.2.2 virtual Xapian::ErrorHandler::~ErrorHandler () `[virtual]`

We require a virtual destructor because we have virtual methods.

6.9.3 Member Function Documentation

6.9.3.1 void Xapian::ErrorHandler::operator() (Xapian::Error & *error*)

Handle a Xapian::Error object.

This method is called when a Xapian::Error object is thrown and caught inside Enquire. If this is the first ErrorHandler that the Error has been passed to, then the handle_error() virtual method is called, which allows the API user to decide how to handle the error.

Parameters:

error The Xapian::Error object under consideration.

The documentation for this class was generated from the following file:

- include/xapian/errorhandler.h

6.10 Xapian::ESet Class Reference

`#include <enquire.h>`

Public Member Functions

- ESet ()

  Construct an empty ESet.

- ~ESet ()

  Destructor.

- ESet (const ESet &other)

  Copying is allowed (and is cheap).

- void operator= (const ESet &other)

  Assignment is allowed (and is cheap).

- Xapian::termcount get_ebound () const
- Xapian::termcount size () const
- Xapian::termcount max_size () const
- bool empty () const
- void swap (ESet &other)
- ESetIterator begin () const
- ESetIterator end () const
- ESetIterator back () const
- ESetIterator operator[ ] (Xapian::termcount i) const
- std::string get_description () const

  Return a string describing this object.

Public Attributes

- Xapian::Internal::RefCntPtr< Internal > **internal**

6.10.1 Detailed Description

Class representing an ordered set of expand terms (an ESet). This set represents the results of an expand operation, which is performed by Xapian::Enquire::get_eset().

6.10.2 Constructor & Destructor Documentation

6.10.2.1 Xapian::ESet::ESet ()

Construct an empty ESet.

6.10.2.2 Xapian::ESet::~ESet ()

Destructor.

6.10.2.3 Xapian::ESet::ESet (const ESet & *other*)

Copying is allowed (and is cheap).

6.10.3 Member Function Documentation

6.10.3.1 void Xapian::ESet::operator= (const ESet & *other*)

Assignment is allowed (and is cheap).

6.10.3.2 Xapian::termcount Xapian::ESet::get_ebound () const

A lower bound on the number of terms which are in the full set of results of the expand. This will be greater than or equal to size()

6.10.3.3 Xapian::termcount Xapian::ESet::size () const

The number of terms in this E-Set

6.10.3.4 Xapian::termcount Xapian::ESet::max_size () const [inline]

Required to allow use as an STL container.

6.10.3.5 bool Xapian::ESet::empty () const

Test if this E-Set is empty

6.10.3.6 void Xapian::ESet::swap (ESet & *other*)

Swap the E-Set we point to with another

6.10.3.7 ESetIterator Xapian::ESet::begin () const

Iterator for the terms in this E-Set

6.10.3.8 ESetIterator Xapian::ESet::end () const

End iterator corresponding to begin()

6.10.3.9 ESetIterator Xapian::ESet::back () const

Iterator pointing to the last element of this E-Set

6.10.3.10 ESetIterator Xapian::ESet::operator[ ] (Xapian::termcount *i*) const

This returns the term at position i in this E-Set.

6.10.3.11 std::string Xapian::ESet::get_description () const

Return a string describing this object.

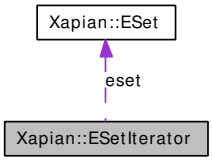
The documentation for this class was generated from the following file:

- include/xapian/enquire.h

6.11 Xapian::ESetIterator Class Reference

`#include <enquire.h>`

Collaboration diagram for Xapian::ESetIterator:



Public Types

- typedef std::bidirectional_iterator_tag iterator_category
  Allow use as an STL iterator.
- typedef std::string **value_type**
- typedef Xapian::termcount_diff **difference_type**
- typedef std::string * **pointer**
- typedef std::string & **reference**

Public Member Functions

- ESetIterator ()
- ESetIterator (const ESetIterator &other)
  Copying is allowed (and is cheap).
- void operator= (const ESetIterator &other)
  Assignment is allowed (and is cheap).
- ESetIterator & operator++ ()
  Advance the iterator.
- ESetIterator operator++ (int)
  Advance the iterator (postfix variant).
- ESetIterator & operator– ()
  Decrement the iterator.
- ESetIterator operator– (int)
  Decrement the iterator (postfix variant).
- const std::string & operator * () const
  Get the term for the current position.

- Xapian::weight get_weight () const

  Get the weight of the term at the current position.

- std::string get_description () const

  Return a string describing this object.

Friends

- class **ESet**
- bool **operator==** (const ESetIterator &a, const ESetIterator &b)
- bool **operator!=** (const ESetIterator &a, const ESetIterator &b)

6.11.1 Detailed Description

Iterate through terms in the ESet

6.11.2 Member Typedef Documentation

6.11.2.1 typedef std::bidirectional_iterator_tag Xapian::ESetIterator::iterator_category

Allow use as an STL iterator.

6.11.3 Constructor & Destructor Documentation

6.11.3.1 Xapian::ESetIterator::ESetIterator () [inline]

Create an uninitialised iterator; this cannot be used, but is convenient syntactically.

6.11.3.2 Xapian::ESetIterator::ESetIterator (const ESetIterator & *other*) [inline]

Copying is allowed (and is cheap).

6.11.4 Member Function Documentation

6.11.4.1 void Xapian::ESetIterator::operator= (const ESetIterator & *other*) [inline]

Assignment is allowed (and is cheap).

6.11.4.2 ESetIterator& Xapian::ESetIterator::operator++ () [inline]

Advance the iterator.

6.11.4.3 ESetIterator Xapian::ESetIterator::operator++ (int) [inline]

Advance the iterator (postfix variant).

6.11.4.4 ESetIterator& Xapian::ESetIterator::operator– () [inline]

Decrement the iterator.

6.11.4.5 ESetIterator Xapian::ESetIterator::operator– (int) [inline]

Decrement the iterator (postfix variant).

6.11.4.6 const std::string& Xapian::ESetIterator::operator ∗ () const

Get the term for the current position.

6.11.4.7 Xapian::weight Xapian::ESetIterator::get_weight () const

Get the weight of the term at the current position.

6.11.4.8 std::string Xapian::ESetIterator::get_description () const

Return a string describing this object.

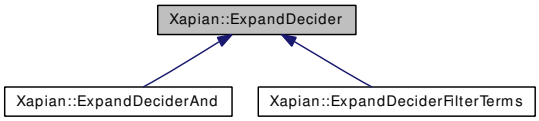
The documentation for this class was generated from the following file:

- include/xapian/enquire.h

6.12 Xapian::ExpandDecider Class Reference

`#include <expanddecider.h>`

Inheritance diagram for Xapian::ExpandDecider:



Public Member Functions

- virtual bool operator() (const std::string &term) const =0
- virtual ~ExpandDecider ()

6.12.1 Detailed Description

Virtual base class for expand decider functor.

6.12.2 Constructor & Destructor Documentation

6.12.2.1 virtual Xapian::ExpandDecider::~ExpandDecider () [virtual]

Virtual destructor, because we have virtual methods.

6.12.3 Member Function Documentation

6.12.3.1 virtual bool Xapian::ExpandDecider::operator() (const std::string & *term*) const [pure virtual]

Do we want this term in the ESet?

Implemented in Xapian::ExpandDeciderAnd, and Xapian::ExpandDeciderFilterTerms.

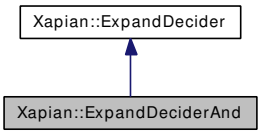
The documentation for this class was generated from the following file:

- include/xapian/expanddecider.h

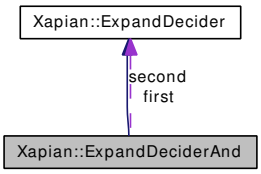
6.13 Xapian::ExpandDeciderAnd Class Reference

`#include <expanddecider.h>`

Inheritance diagram for Xapian::ExpandDeciderAnd:



Collaboration diagram for Xapian::ExpandDeciderAnd:



Public Member Functions

- ExpandDeciderAnd (const ExpandDecider &first_, const ExpandDecider &second_)
- ExpandDeciderAnd (const ExpandDecider *first_, const ExpandDecider *second_)
- virtual bool operator() (const std::string &term) const

6.13.1 Detailed Description

ExpandDecider subclass which rejects terms using two ExpandDeciders.

Terms are only accepted if they are accepted by both of the specified ExpandDecider objects.

6.13.2 Constructor & Destructor Documentation

6.13.2.1 Xapian::ExpandDeciderAnd::ExpandDeciderAnd (const ExpandDecider & *first_*, const ExpandDecider & *second_*) [inline]

Terms will be checked with *first*, and if accepted, then checked with *second*.

6.13.2.2 Xapian::ExpandDeciderAnd::ExpandDeciderAnd (const ExpandDecider * *first_*, const ExpandDecider * *second_*) [inline]

Compatibility method.

6.13.3 Member Function Documentation

6.13.3.1 virtual bool Xapian::ExpandDeciderAnd::operator() (const std::string & *term*) const [virtual]

Do we want this term in the ESet?

Implements Xapian::ExpandDecider.

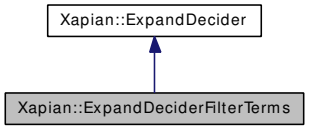
The documentation for this class was generated from the following file:

- include/xapian/expanddecider.h

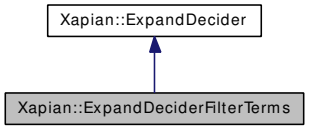
6.14 Xapian::ExpandDeciderFilterTerms Class Reference

`#include <expanddecider.h>`

Inheritance diagram for Xapian::ExpandDeciderFilterTerms:



Collaboration diagram for Xapian::ExpandDeciderFilterTerms:



Public Member Functions

- template<class Iterator>
  ExpandDeciderFilterTerms (Iterator reject_begin, Iterator reject_end)
- virtual bool operator() (const std::string &term) const

6.14.1 Detailed Description

ExpandDecider subclass which rejects terms in a specified list.

ExpandDeciderFilterTerms provides an easy way to filter out terms from a fixed list when generating an ESet.

6.14.2 Constructor & Destructor Documentation

6.14.2.1 template<class Iterator> Xapian::ExpandDeciderFilterTerms::ExpandDeciderFilterTerms (Iterator *reject_begin*, Iterator *reject_end*) [inline]

The two iterators specify a list of terms to be rejected.

reject_begin and *reject_end* can be any input_iterator type which returns std::string or char * (e.g. TermIterator or char **).

6.14.3 Member Function Documentation

6.14.3.1 virtual bool Xapian::ExpandDeciderFilterTerms::operator() (const std::string & *term*) const [virtual]

Do we want this term in the ESet?

Implements Xapian::ExpandDecider.

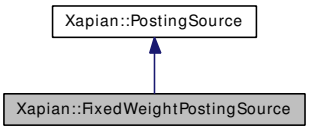
The documentation for this class was generated from the following file:

- include/xapian/expanddecider.h

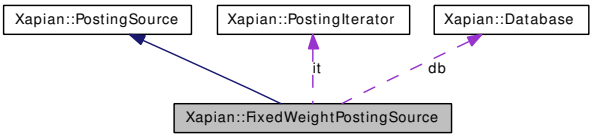
6.15 Xapian::FixedWeightPostingSource Class Reference

`#include <postingsource.h>`

Inheritance diagram for Xapian::FixedWeightPostingSource:



Collaboration diagram for Xapian::FixedWeightPostingSource:



Public Member Functions

- FixedWeightPostingSource (Xapian::weight wt)
- Xapian::doccount get_termfreq_min () const
- Xapian::doccount get_termfreq_est () const
- Xapian::doccount get_termfreq_max () const
- Xapian::weight get_weight () const
- void next (Xapian::weight min_wt)
- void skip_to (Xapian::docid min_docid, Xapian::weight min_wt)
- bool check (Xapian::docid min_docid, Xapian::weight min_wt)
- bool at_end () const
- Xapian::docid get_docid () const
- FixedWeightPostingSource * clone () const
- std::string name () const
- std::string serialise () const
- FixedWeightPostingSource * unserialise (const std::string &s) const
- void init (const Database &db_)
- std::string get_description () const

6.15.1 Detailed Description

A posting source which returns a fixed weight for all documents.

This returns entries for all documents in the given database, with a fixed weight (specified by a parameter to the constructor).

6.15.2 Constructor & Destructor Documentation

6.15.2.1 Xapian::FixedWeightPostingSource::FixedWeightPostingSource (Xapian::weight *wt*)

Construct a FixedWeightPostingSource.

Parameters:

wt The fixed weight to return.

6.15.3 Member Function Documentation

6.15.3.1 Xapian::doccount Xapian::FixedWeightPostingSource::get_termfreq_min () const [virtual]

A lower bound on the number of documents this object can return.

Xapian will always call init() on a PostingSource before calling this for the first time.

Implements Xapian::PostingSource.

6.15.3.2 Xapian::doccount Xapian::FixedWeightPostingSource::get_termfreq_est () const [virtual]

An estimate of the number of documents this object can return.

It must always be true that:

get_termfreq_min() <= get_termfreq_est() <= get_termfreq_max()

Xapian will always call init() on a PostingSource before calling this for the first time.

Implements Xapian::PostingSource.

6.15.3.3 Xapian::doccount Xapian::FixedWeightPostingSource::get_termfreq_max () const [virtual]

An upper bound on the number of documents this object can return.

Xapian will always call init() on a PostingSource before calling this for the first time.

Implements Xapian::PostingSource.

6.15.3.4 Xapian::weight Xapian::FixedWeightPostingSource::get_weight () const [virtual]

Return the weight contribution for the current document.

This default implementation always returns 0, for convenience when implementing "weight-less" PostingSource subclasses.

This method may assume that it will only be called when there is a "current document". In detail: Xapian will always call init() on a PostingSource before calling this for the first time. It will also only call this if the PostingSource reports that it is pointing to a valid document (ie, it will not call it before calling at least one of next(), skip_to() or check(), and will ensure that the PostingSource is not at the end by calling at_end()).

Reimplemented from Xapian::PostingSource.

6.15.3.5 void Xapian::FixedWeightPostingSource::next (Xapian::weight *min_wt*) [virtual]

Advance the current position to the next matching document.

The PostingSource starts before the first entry in the list, so next() must be called before any methods which need the context of the current position.

Xapian will always call init() on a PostingSource before calling this for the first time.

Parameters:

min_wt The minimum weight contribution that is needed (this is just a hint which subclasses may ignore).

Implements Xapian::PostingSource.

6.15.3.6 void Xapian::FixedWeightPostingSource::skip_to (Xapian::docid *did*, Xapian::weight *min_wt*) [virtual]

Skip forward to the specified docid.

If the specified docid isn't in the list, position ourselves on the first document after it (or at_end() if no greater docids are present).

If the current position is already the specified docid, this method will leave the position unmodified.

If the specified docid is earlier than the current position, the behaviour is unspecified. A sensible behaviour would be to leave the current position unmodified, but it is also reasonable to move to the specified docid.

The default implementation calls next() repeatedly, which works but skip_to() can often be implemented much more efficiently.

Xapian will always call init() on a PostingSource before calling this for the first time.

Note: in the case of a multi-database search, the docid specified is the docid in the single subdatabase relevant to this posting source. See the *init()* method for details.

Parameters:

min_wt The minimum weight contribution that is needed (this is just a hint which subclasses may ignore).

Reimplemented from Xapian::PostingSource.

6.15.3.7 bool Xapian::FixedWeightPostingSource::check (Xapian::docid *did*, Xapian::weight *min_wt*) [virtual]

Check if the specified docid occurs.

The caller is required to ensure that the specified document id *did* actually exists in the database. If it does, it must move to that document id, and return true. If it does not, it may either:

- return true, having moved to a definite position (including "at_end"), which must be the same position as skip_to() would have moved to.

or

- return false, having moved to an "indeterminate" position, such that a subsequent call to next() or skip_to() will move to the next matching position after *did*.

Generally, this method should act like skip_to() and return true if that can be done at little extra cost.

Otherwise it should simply check if a particular docid is present, returning true if it is, and false if it isn't.

The default implementation calls skip_to() and always returns true.

Xapian will always call init() on a PostingSource before calling this for the first time.

Note: in the case of a multi-database search, the docid specified is the docid in the single subdatabase relevant to this posting source. See the *init()* method for details.

Reimplemented from Xapian::PostingSource.

6.15.3.8 bool Xapian::FixedWeightPostingSource::at_end () const [virtual]

Return true if the current position is past the last entry in this list.

At least one of *next()*, *skip_to()* or *check()* will be called before this method is first called.

Implements Xapian::PostingSource.

6.15.3.9 Xapian::docid Xapian::FixedWeightPostingSource::get_docid () const [virtual]

Return the current docid.

This method may assume that it will only be called when there is a "current document". See *get_weight()* for details.

Note: in the case of a multi-database search, the returned docid should be in the single subdatabase relevant to this posting source. See the *init()* method for details.

Implements Xapian::PostingSource.

6.15.3.10 FixedWeightPostingSource* Xapian::FixedWeightPostingSource::clone () const [virtual]

Clone the posting source.

The clone should inherit the configuration of the parent, but need not inherit the state. ie, the clone does not need to be in the same iteration position as the original: the matcher will always call init() on the clone before attempting to move the iterator, or read the information about the current position of the iterator.

This may return NULL to indicate that cloning is not supported. In this case, the PostingSource may only be used with a single-database search.

The default implementation returns NULL.

Note that the returned object will be deallocated by Xapian after use with "delete". It must therefore have been allocated with "new".

Reimplemented from Xapian::PostingSource.

6.15.3.11 std::string Xapian::FixedWeightPostingSource::name () const [virtual]

Name of the posting source class.

This is used when serialising and unserialising posting sources; for example, for performing remote searches.

If the subclass is in a C++ namespace, the namespace should be included in the name, using "::" as a separator. For example, for a PostingSource subclass called "FooPostingSource" in the "Xapian" namespace the result of this call should be "Xapian::FooPostingSource".

This should only be implemented if serialise() and unserialise() are also implemented. The default implementation returns an empty string.

If this returns an empty string, Xapian will assume that serialise() and unserialise() are not implemented.

Reimplemented from Xapian::PostingSource.

6.15.3.12 std::string Xapian::FixedWeightPostingSource::serialise () const [virtual]

Serialise object parameters into a string.

The serialised parameters should represent the configuration of the posting source, but need not (indeed, should not) represent the current iteration state.

Reimplemented from Xapian::PostingSource.

6.15.3.13 FixedWeightPostingSource* Xapian::FixedWeightPostingSource::unserialise (const std::string & *s*) const [virtual]

Create object given string serialisation returned by serialise().

Parameters:

s A serialised instance of this PostingSource subclass.

Reimplemented from Xapian::PostingSource.

6.15.3.14 void Xapian::FixedWeightPostingSource::init (const Database & *db*) [virtual]

Set this PostingSource to the start of the list of postings.

This is called automatically by the matcher prior to each query being processed.

If a PostingSource is used for multiple searches, *init()* will therefore be called multiple times, and must handle this by using the database passed in the most recent call.

Parameters:

db The database which the PostingSource should iterate through.

Note: the database supplied to this method must not be modified: in particular, the reopen() method should not be called on it.

Note: in the case of a multi-database search, a separate PostingSource will be used for each database (the separate PostingSources will be obtained using *clone()*), and each PostingSource will be passed one of the sub-databases as the *db* parameter here. The *db* parameter will therefore always refer to a single database. All docids passed to, or returned from, the PostingSource refer to docids in that single database, rather than in the multi-database.

Implements Xapian::PostingSource.

6.15.3.15 std::string Xapian::FixedWeightPostingSource::get_description () const [virtual]

Return a string describing this object.

This default implementation returns a generic answer. This default it provided to avoid forcing those deriving their own PostingSource subclass from having to implement this (they may not care what get_description() gives for their subclass).

Reimplemented from Xapian::PostingSource.

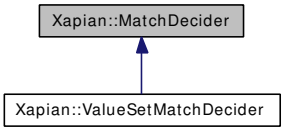
The documentation for this class was generated from the following file:

- include/xapian/postingsource.h

6.16 Xapian::MatchDecider Class Reference

`#include <enquire.h>`

Inheritance diagram for Xapian::MatchDecider:



Public Member Functions

- virtual bool operator() (const Xapian::Document &doc) const=0
- virtual ~MatchDecider ()

  Destructor.

6.16.1 Detailed Description

Base class for matcher decision functor.

6.16.2 Constructor & Destructor Documentation

6.16.2.1 virtual Xapian::MatchDecider::~MatchDecider () [virtual]

Destructor.

6.16.3 Member Function Documentation

6.16.3.1 virtual bool Xapian::MatchDecider::operator() (const Xapian::Document & *doc*) const [pure virtual]

Decide whether we want this document to be in the MSet.

Return true if the document is acceptable, or false if the document should be excluded from the MSet.

Implemented in Xapian::ValueSetMatchDecider.

The documentation for this class was generated from the following file:

- include/xapian/enquire.h

6.17 Xapian::MSet Class Reference

`#include <enquire.h>`

Public Types

- typedef MSetIterator value_type
  Allow use as an STL container.

- typedef MSetIterator **iterator**
- typedef MSetIterator **const_iterator**
- typedef MSetIterator & **reference**
- typedef MSetIterator & **const_reference**
- typedef MSetIterator * **pointer**
- typedef Xapian::doccount_diff **difference_type**
- typedef Xapian::doccount **size_type**

Public Member Functions

- **MSet** (MSet::Internal *internal_)
- MSet ()
  Create an empty Xapian::MSet.

- ~MSet ()
  Destroy a Xapian::MSet.

- MSet (const MSet &other)
  Copying is allowed (and is cheap).

- void operator= (const MSet &other)
  Assignment is allowed (and is cheap).

- void fetch (const MSetIterator &begin, const MSetIterator &end) const
- void fetch (const MSetIterator &item) const
- void fetch () const
- Xapian::percent convert_to_percent (Xapian::weight wt) const
- Xapian::percent convert_to_percent (const MSetIterator &it) const
  Return the percentage score for a particular item.

- Xapian::doccount get_termfreq (const std::string &tname) const
- Xapian::weight get_termweight (const std::string &tname) const
- Xapian::doccount get_firstitem () const
- Xapian::doccount get_matches_lower_bound () const
- Xapian::doccount get_matches_estimated () const
- Xapian::doccount get_matches_upper_bound () const
- Xapian::doccount get_uncollapsed_matches_lower_bound () const

- Xapian::doccount get_uncollapsed_matches_estimated () const
- Xapian::doccount get_uncollapsed_matches_upper_bound () const
- Xapian::weight get_max_possible () const
- Xapian::weight get_max_attained () const
- Xapian::doccount size () const
- Xapian::doccount max_size () const
- bool empty () const
- void swap (MSet &other)
- MSetIterator begin () const
- MSetIterator end () const
- MSetIterator back () const
- MSetIterator operator[ ] (Xapian::doccount i) const
- std::string get_description () const

  Return a string describing this object.

Public Attributes

- Xapian::Internal::RefCntPtr< Internal > **internal**

6.17.1 Detailed Description

A match set (MSet). This class represents (a portion of) the results of a query.

6.17.2 Member Typedef Documentation

6.17.2.1 typedef MSetIterator Xapian::MSet::value_type

Allow use as an STL container.

6.17.3 Constructor & Destructor Documentation

6.17.3.1 Xapian::MSet::MSet ()

Create an empty Xapian::MSet.

6.17.3.2 Xapian::MSet::~MSet ()

Destroy a Xapian::MSet.

6.17.3.3 Xapian::MSet::MSet (const MSet & *other*)

Copying is allowed (and is cheap).

6.17.4 Member Function Documentation

6.17.4.1 void Xapian::MSet::operator= (const MSet & *other*)

Assignment is allowed (and is cheap).

6.17.4.2 void Xapian::MSet::fetch (const MSetIterator & *begin*, const MSetIterator & *end*) const

Fetch the document info for a set of items in the MSet.

This method causes the documents in the range specified by the iterators to be fetched from the database, and cached in the Xapian::MSet object. This has little effect when performing a search across a local database, but will greatly speed up subsequent access to the document contents when the documents are stored in a remote database.

The iterators must be over this Xapian::MSet - undefined behaviour will result otherwise.

Parameters:

begin MSetIterator for first item to fetch.

end MSetIterator for item after last item to fetch.

6.17.4.3 void Xapian::MSet::fetch (const MSetIterator & *item*) const

Fetch the single item specified.

6.17.4.4 void Xapian::MSet::fetch () const

Fetch all the items in the MSet.

6.17.4.5 Xapian::percent Xapian::MSet::convert_to_percent (Xapian::weight *wt*) const

This converts the weight supplied to a percentage score. The return value will be in the range 0 to 100, and will be 0 if and only if the item did not match the query at all.

6.17.4.6 Xapian::percent Xapian::MSet::convert_to_percent (const MSetIterator & *it*) const

Return the percentage score for a particular item.

6.17.4.7 Xapian::doccount Xapian::MSet::get_termfreq (const std::string & *tname*) const

Return the term frequency of the given query term.

Parameters:

tname The term to look for.

This is sometimes more efficient than asking the database directly for the term frequency - in particular, if the term was in the query, its frequency will usually be cached in the MSet.

6.17.4.8 Xapian::weight Xapian::MSet::get_termweight (const std::string & *tname*) const

Return the term weight of the given query term.

Parameters:

tname The term to look for.

Exceptions:

Xapian::InvalidArgumentError is thrown if the term was not in the query.

6.17.4.9 Xapian::doccount Xapian::MSet::get_firstitem () const

The index of the first item in the result which was put into the MSet.

This corresponds to the parameter "first" specified in Xapian::Enquire::get_mset(). A value of 0 corresponds to the highest result being the first item in the MSet.

6.17.4.10 Xapian::doccount Xapian::MSet::get_matches_lower_bound () const

A lower bound on the number of documents in the database which match the query.

This figure takes into account collapsing of duplicates, and weighting cutoff values.

This number is usually considerably less than the actual number of documents which match the query.

6.17.4.11 Xapian::doccount Xapian::MSet::get_matches_estimated () const

An estimate for the number of documents in the database which match the query.

This figure takes into account collapsing of duplicates, and weighting cutoff values.

This value is returned because there is sometimes a request to display such information. However, our experience is that presenting this value to users causes them to worry about the large number of results, rather than how useful those at the top of the result set are, and is thus undesirable.

6.17.4.12 Xapian::doccount Xapian::MSet::get_matches_upper_bound () const

An upper bound on the number of documents in the database which match the query.

This figure takes into account collapsing of duplicates, and weighting cutoff values.

This number is usually considerably greater than the actual number of documents which match the query.

6.17.4.13 Xapian::doccount Xapian::MSet::get_uncollapsed_matches_lower_bound () const

A lower bound on the number of documents in the database which would match the query if collapsing wasn't used.

6.17.4.14 Xapian::doccount Xapian::MSet::get_uncollapsed_matches_estimated () const

A estimate of the number of documents in the database which would match the query if collapsing wasn't used.

6.17.4.15 Xapian::doccount Xapian::MSet::get_uncollapsed_matches_upper_bound () const

A upper bound on the number of documents in the database which would match the query if collapsing wasn't used.

6.17.4.16 Xapian::weight Xapian::MSet::get_max_possible () const

The maximum possible weight in the MSet.

This weight is likely not to be attained in the set of results, but represents an upper bound on the weight which a document could attain for the given query.

6.17.4.17 Xapian::weight Xapian::MSet::get_max_attained () const

The greatest weight which is attained by any document in the database.

If firstitem == 0 and the primary ordering is by relevance, this is the weight of the first entry in the MSet.

If no documents are found by the query, this will be 0.

Note that calculation of max_attained requires calculation of at least one result item - therefore, if no items were requested when the query was performed (by specifying maxitems = 0 in Xapian::Enquire::get_mset()), this value will be 0.

6.17.4.18 Xapian::doccount Xapian::MSet::size () const

The number of items in this MSet

6.17.4.19 Xapian::doccount Xapian::MSet::max_size () const [inline]

Required to allow use as an STL container.

6.17.4.20 bool Xapian::MSet::empty () const

Test if this MSet is empty

6.17.4.21 void Xapian::MSet::swap (MSet & *other*)

Swap the MSet we point to with another

6.17.4.22 MSetIterator Xapian::MSet::begin () const

Iterator for the items in this MSet

6.17.4.23 MSetIterator Xapian::MSet::end () const

End iterator corresponding to begin()

6.17.4.24 MSetIterator Xapian::MSet::back () const

Iterator pointing to the last element of this MSet

6.17.4.25 MSetIterator Xapian::MSet::operator[ ] (Xapian::doccount *i*) const

This returns the document at position i in this MSet object.

Note that this is not the same as the document at rank i in the query, unless the "first" parameter to Xapian::Enquire::get_mset was 0. Rather, it is the document at rank i + first.

In other words, the offset is into the documents represented by this object, not into the set of documents matching the query.

6.17.4.26 std::string Xapian::MSet::get_description () const

Return a string describing this object.

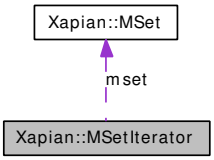
The documentation for this class was generated from the following file:

- include/xapian/enquire.h

6.18 Xapian::MSetIterator Class Reference

`#include <enquire.h>`

Collaboration diagram for Xapian::MSetIterator:



Public Types

- typedef std::bidirectional_iterator_tag iterator_category
  Allow use as an STL iterator.

- typedef Xapian::docid **value_type**
- typedef Xapian::doccount_diff **difference_type**
- typedef Xapian::docid * **pointer**
- typedef Xapian::docid & **reference**

Public Member Functions

- MSetIterator ()
- MSetIterator (const MSetIterator &other)
  Copying is allowed (and is cheap).

- void operator= (const MSetIterator &other)
  Assignment is allowed (and is cheap).

- MSetIterator & operator++ ()
  Advance the iterator.

- MSetIterator operator++ (int)
  Advance the iterator (postfix variant).

- MSetIterator & operator– ()
  Decrement the iterator.

- MSetIterator operator– (int)
  Decrement the iterator (postfix variant).

- Xapian::docid operator * () const
  Get the document ID for the current position.

- Xapian::Document get_document () const
- Xapian::doccount get_rank () const
- Xapian::weight get_weight () const

  Get the weight of the document at the current position.

- std::string get_collapse_key () const
- Xapian::doccount get_collapse_count () const
- Xapian::percent get_percent () const
- std::string get_description () const

  Return a string describing this object.

Friends

- class **MSet**
- bool **operator==** (const MSetIterator &a, const MSetIterator &b)
- bool **operator!=** (const MSetIterator &a, const MSetIterator &b)

6.18.1 Detailed Description

An iterator pointing to items in an MSet. This is used for access to individual results of a match.

6.18.2 Member Typedef Documentation

6.18.2.1 typedef std::bidirectional_iterator_tag Xapian::MSetIterator::iterator_category

Allow use as an STL iterator.

6.18.3 Constructor & Destructor Documentation

6.18.3.1 Xapian::MSetIterator::MSetIterator () [inline]

Create an uninitialised iterator; this cannot be used, but is convenient syntactically.

6.18.3.2 Xapian::MSetIterator::MSetIterator (const MSetIterator & *other*) [inline]

Copying is allowed (and is cheap).

6.18.4 Member Function Documentation

6.18.4.1 void Xapian::MSetIterator::operator= (const MSetIterator & *other*) [inline]

Assignment is allowed (and is cheap).

6.18.4.2 MSetIterator& Xapian::MSetIterator::operator++ () [inline]

Advance the iterator.

6.18.4.3 MSetIterator Xapian::MSetIterator::operator++ (int) [inline]

Advance the iterator (postfix variant).

6.18.4.4 MSetIterator& Xapian::MSetIterator::operator– () [inline]

Decrement the iterator.

6.18.4.5 MSetIterator Xapian::MSetIterator::operator– (int) [inline]

Decrement the iterator (postfix variant).

6.18.4.6 Xapian::docid Xapian::MSetIterator::operator * () const

Get the document ID for the current position.

6.18.4.7 Xapian::Document Xapian::MSetIterator::get_document () const

Get a Xapian::Document object for the current position.

This method returns a Xapian::Document object which provides the information about the document pointed to by the MSetIterator.

If the underlying database has suitable support, using this call (rather than asking the database for a document based on its document ID) will enable the system to ensure that the correct data is returned, and that the document has not been deleted or changed since the query was performed.

Returns:

A Xapian::Document object containing the document data.

Exceptions:

Xapian::DocNotFoundError The document specified could not be found in the database.

6.18.4.8 Xapian::doccount Xapian::MSetIterator::get_rank () const [inline]

Get the rank of the document at the current position.

The rank is the position that this document is at in the ordered list of results of the query. The result is 0-based - i.e. the top-ranked document has a rank of 0.

6.18.4.9 Xapian::weight Xapian::MSetIterator::get_weight () const

Get the weight of the document at the current position.

6.18.4.10 std::string Xapian::MSetIterator::get_collapse_key () const

Get the collapse key for this document.

6.18.4.11 Xapian::doccount Xapian::MSetIterator::get_collapse_count () const

Get an estimate of the number of documents that have been collapsed into this one.

The estimate will always be less than or equal to the actual number of other documents satisfying the match criteria with the same collapse key as this document.

This method may return 0 even though there are other documents with the same collapse key which satisfying the match criteria. However if this method returns non-zero, there definitely are other such documents. So this method may be used to inform the user that there are "at least N other matches in this group", or to control whether to offer a "show other documents in this group" feature (but note that it may not offer it in every case where it would show other documents).

6.18.4.12 Xapian::percent Xapian::MSetIterator::get_percent () const

This returns the weight of the document as a percentage score.

The return value will be an integer in the range 0 to 100: 0 meaning that the item did not match the query at all.

The intention is that the highest weighted document will get 100 if it matches all the weight-contributing terms in the query. However, currently it may get a lower percentage score if you use a MatchDecider and the sorting is primarily by value. In this case, the percentage for a particular document may vary depending on the first, max_size, and checkatleast parameters passed to Enquire::get_mset() (this bug is hard to fix without having to apply the MatchDecider to potentially many more documents, which is potentially costly).

6.18.4.13 std::string Xapian::MSetIterator::get_description () const

Return a string describing this object.

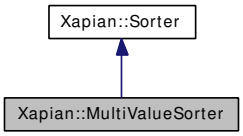
The documentation for this class was generated from the following file:

- include/xapian/enquire.h

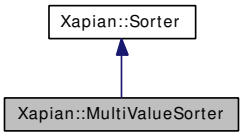
6.19 Xapian::MultiValueSorter Class Reference

`#include <sorter.h>`

Inheritance diagram for Xapian::MultiValueSorter:



Collaboration diagram for Xapian::MultiValueSorter:



Public Member Functions

- template<class Iterator>
  MultiValueSorter (Iterator begin, Iterator end)
- virtual std::string operator() (const Xapian::Document &doc) const
- void **add** (Xapian::valueno valno, bool forward=true)

6.19.1 Detailed Description

Sorter subclass which sorts by a several values.

Results are ordered by the first value. In the event of a tie, the second is used. If this is the same for both, the third is used, and so on.

6.19.2 Member Function Documentation

6.19.2.1 virtual std::string Xapian::MultiValueSorter::operator() (const Xapian::Document & *doc*) const [virtual]

This method takes a Document object and builds a sort key from it.

Documents are then ordered by a string compare on the sort keys.

Implements Xapian::Sorter.

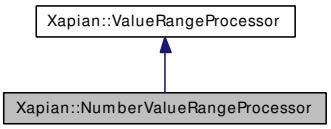
The documentation for this class was generated from the following file:

- include/xapian/sorter.h

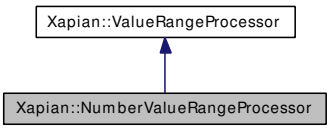
6.20 Xapian::NumberValueRangeProcessor Class Reference

`#include <queryparser.h>`

Inheritance diagram for Xapian::NumberValueRangeProcessor:



Collaboration diagram for Xapian::NumberValueRangeProcessor:



Public Member Functions

- NumberValueRangeProcessor (Xapian::valueno valno_)
- NumberValueRangeProcessor (Xapian::valueno valno_, const std::string &str_, bool prefix_=true)
- Xapian::valueno operator() (std::string &begin, std::string &end)

6.20.1 Detailed Description

Handle a number range.

This class must be used on values which have been encoded using Xapian::sortable_serialise() which turns numbers into strings which will sort in the same order as the numbers (the same values can be used to implement a numeric sort).

6.20.2 Constructor & Destructor Documentation

6.20.2.1 Xapian::NumberValueRangeProcessor::NumberValueRangeProcessor (Xapian::valueno *valno_*) [inline]

Constructor.

Parameters:

valno_ The value number to return from operator().

6.20.2.2 Xapian::NumberValueRangeProcessor::NumberValueRangeProcessor (Xapian::valueno *valno_*, const std::string & *str_*, bool *prefix_* = `true`) `[inline]`

Constructor.

Parameters:

valno_ The value number to return from operator().

str_ A string to look for to recognise values as belonging to this numeric range.

prefix_ Whether to look for the string at the start or end of the values. If true, the string is a prefix; if false, the string is a suffix (default: true).

The string supplied in str_ is used by *operator()* to decide whether the pair of strings supplied to it constitute a valid range. If prefix_ is true, the first value in a range must begin with str_ (and the second value may optionally begin with str_); if prefix_ is false, the second value in a range must end with str_ (and the first value may optionally end with str_).

If str_ is empty, the setting of prefix_ is irrelevant, and no special strings are required at the start or end of the strings defining the range.

The remainder of both strings defining the endpoints must be valid floating point numbers. (FIXME: define format recognised).

For example, if str_ is "$" and prefix_ is true, and the range processor has been added to the queryparser, the queryparser will accept "$10..50" or "$10..$50", but not "10..50" or "10..$50" as valid ranges. If str_ is "kg" and prefix_ is false, the queryparser will accept "10..50kg" or "10kg..50kg", but not "10..50" or "10kg..50" as valid ranges.

6.20.3 Member Function Documentation

6.20.3.1 Xapian::valueno Xapian::NumberValueRangeProcessor::operator() (std::string & *begin*, std::string & *end*) `[virtual]`

See if <begin>..<end> is a valid numeric value range.

If <begin>..<end> is a valid numeric value range, and has the appropriate prefix or suffix (if specified) required for this NumberValueRangeProcessor, this method returns the value number of range filter on, and sets begin and end to the appropriate serialised values needed to delimit the range. Otherwise it returns Xapian::BAD_VALUENO.

Implements Xapian::ValueRangeProcessor.

The documentation for this class was generated from the following file:

- include/xapian/queryparser.h

6.21 Xapian::PositionIterator Class Reference

`#include <positioniterator.h>`

Public Types

- typedef std::input_iterator_tag **iterator_category**
- typedef Xapian::termpos **value_type**
- typedef Xapian::termpos_diff **difference_type**
- typedef Xapian::termpos * **pointer**
- typedef Xapian::termpos & **reference**

Public Member Functions

- **PositionIterator** (Internal *internal_)
- PositionIterator ()

  Default constructor - for declaring an uninitialised iterator.

- ~PositionIterator ()

  Destructor.

- PositionIterator (const PositionIterator &o)
- void operator= (const PositionIterator &o)
- Xapian::termpos **operator** * () const
- PositionIterator & **operator++** ()
- TermPosWrapper **operator++** (int)
- void **skip_to** (Xapian::termpos pos)
- std::string get_description () const

  Return a string describing this object.

Friends

- class **PostingIterator**
- class **TermIterator**
- class **Database**
- bool operator== (const PositionIterator &a, const PositionIterator &b)

  Test equality of two PositionIterators.

6.21.1 Detailed Description

An iterator pointing to items in a list of positions.

6.21.2 Constructor & Destructor Documentation

6.21.2.1 Xapian::PositionIterator::PositionIterator ()

Default constructor - for declaring an uninitialised iterator.

6.21.2.2 Xapian::PositionIterator::~PositionIterator ()

Destructor.

6.21.2.3 Xapian::PositionIterator::PositionIterator (const PositionIterator & *o*)

Copying is allowed. The internals are reference counted, so copying is also cheap.

6.21.3 Member Function Documentation

6.21.3.1 void Xapian::PositionIterator::operator= (const PositionIterator & *o*)

Assignment is allowed. The internals are reference counted, so assignment is also cheap.

6.21.3.2 std::string Xapian::PositionIterator::get_description () const

Return a string describing this object.

6.21.4 Friends And Related Function Documentation

6.21.4.1 bool operator== (const PositionIterator & *a*, const PositionIterator & *b*) [friend]

Test equality of two PositionIterators.

The documentation for this class was generated from the following file:

- include/xapian/positioniterator.h

6.22 Xapian::PostingIterator Class Reference

`#include <postingiterator.h>`

Public Types

- typedef std::input_iterator_tag iterator_category

  Allow use as an STL iterator.

- typedef Xapian::docid **value_type**
- typedef Xapian::doccount_diff **difference_type**
- typedef Xapian::docid * **pointer**
- typedef Xapian::docid & **reference**

Public Member Functions

- PostingIterator ()

  Default constructor - for declaring an uninitialised iterator.

- ~PostingIterator ()

  Destructor.

- PostingIterator (const PostingIterator &other)
- void operator= (const PostingIterator &other)
- PostingIterator & **operator++** ()
- DocIDWrapper **operator++** (int)
- void skip_to (Xapian::docid did)
- Xapian::docid operator * () const

  Get the document id at the current position in the postlist.

- Xapian::termcount get_doclength () const
- Xapian::termcount get_wdf () const
- PositionIterator positionlist_begin () const
- PositionIterator positionlist_end () const
- std::string get_description () const

  Return a string describing this object.

Friends

- class **Database**
- bool operator== (const PostingIterator &a, const PostingIterator &b)

  Test equality of two PostingIterators.

6.22.1 Detailed Description

An iterator pointing to items in a list of postings.

6.22.2 Member Typedef Documentation

6.22.2.1 typedef std::input_iterator_tag Xapian::PostingIterator::iterator_category

Allow use as an STL iterator.

6.22.3 Constructor & Destructor Documentation

6.22.3.1 Xapian::PostingIterator::PostingIterator ()

Default constructor - for declaring an uninitialised iterator.

6.22.3.2 Xapian::PostingIterator::~PostingIterator ()

Destructor.

6.22.3.3 Xapian::PostingIterator::PostingIterator (const PostingIterator & *other*)

Copying is allowed. The internals are reference counted, so copying is also cheap.

6.22.4 Member Function Documentation

6.22.4.1 void Xapian::PostingIterator::operator= (const PostingIterator & *other*)

Assignment is allowed. The internals are reference counted, so assignment is also cheap.

6.22.4.2 void Xapian::PostingIterator::skip_to (Xapian::docid *did*)

Skip the iterator to document did, or the first document after did if did isn't in the list of documents being iterated.

6.22.4.3 Xapian::docid Xapian::PostingIterator::operator * () const

Get the document id at the current position in the postlist.

6.22.4.4 Xapian::termcount Xapian::PostingIterator::get_doclength () const

Get the length of the document at the current position in the postlist.

This information may be stored in the postlist, in which case this lookup should be extremely fast (indeed, not require further disk access). If the information is not present in the postlist, it will be retrieved from the database, at a greater performance cost.

6.22.4.5 Xapian::termcount Xapian::PostingIterator::get_wdf () const

Get the within document frequency of the document at the current position in the postlist.

6.22.4.6 PositionIterator Xapian::PostingIterator::positionlist_begin () const

Return PositionIterator pointing to start of positionlist for current document.

6.22.4.7 PositionIterator Xapian::PostingIterator::positionlist_end () const [inline]

Return PositionIterator pointing to end of positionlist for current document.

6.22.4.8 std::string Xapian::PostingIterator::get_description () const

Return a string describing this object.

6.22.5 Friends And Related Function Documentation

6.22.5.1 bool operator== (const PostingIterator & *a*, const PostingIterator & *b*) [friend]

Test equality of two PostingIterators.

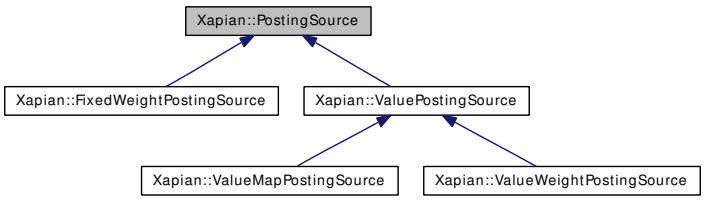
The documentation for this class was generated from the following file:

- include/xapian/postingiterator.h

6.23 Xapian::PostingSource Class Reference

`#include <postingsource.h>`

Inheritance diagram for Xapian::PostingSource:



Public Member Functions

- virtual Xapian::doccount get_termfreq_min () const=0
- virtual Xapian::doccount get_termfreq_est () const=0
- virtual Xapian::doccount get_termfreq_max () const=0
- Xapian::weight get_maxweight () const

  Return the currently set upper bound on what get_weight() can return.

- virtual Xapian::weight get_weight () const
- virtual Xapian::docid get_docid () const=0
- virtual void next (Xapian::weight min_wt)=0
- virtual void skip_to (Xapian::docid did, Xapian::weight min_wt)
- virtual bool check (Xapian::docid did, Xapian::weight min_wt)
- virtual bool at_end () const=0
- virtual PostingSource * clone () const
- virtual std::string name () const
- virtual std::string serialise () const
- virtual PostingSource * unserialise (const std::string &s) const
- virtual void init (const Database &db)=0
- virtual std::string get_description () const

Protected Member Functions

- PostingSource ()

  Allow subclasses to be instantiated.

- void set_maxweight (Xapian::weight max_weight)

6.23.1 Detailed Description

Base class which provides an "external" source of postings.

Warning: the PostingSource interface is currently experimental, and is liable to change between releases without warning.

6.23.2 Constructor & Destructor Documentation

6.23.2.1 Xapian::PostingSource::PostingSource () [inline, protected]

Allow subclasses to be instantiated.

6.23.3 Member Function Documentation

6.23.3.1 void Xapian::PostingSource::set_maxweight (Xapian::weight *max_weight*) [protected]

Set an upper bound on what get_weight() can return from now on.

This upper bound is used by the matcher to perform various optimisations, so if you can return a good bound, then matches will generally run faster.

This method should be called after calling init(), and may be called during iteration if the upper bound drops.

It is valid for the posting source to have returned a higher value from get_weight() earlier in the iteration, but the posting source must not return a higher value from get_weight() than the currently set upper bound, and the upper bound must not be increased (until init() has been called).

If you don't call this method, the upper bound will default to 0, for convenience when implementing "weight-less" PostingSource subclasses.

6.23.3.2 virtual Xapian::doccount Xapian::PostingSource::get_termfreq_min () const [pure virtual]

A lower bound on the number of documents this object can return.

Xapian will always call init() on a PostingSource before calling this for the first time.

Implemented in Xapian::ValuePostingSource, and Xapian::FixedWeightPostingSource.

6.23.3.3 virtual Xapian::doccount Xapian::PostingSource::get_termfreq_est () const [pure virtual]

An estimate of the number of documents this object can return.

It must always be true that:

get_termfreq_min() <= get_termfreq_est() <= get_termfreq_max()

Xapian will always call init() on a PostingSource before calling this for the first time.

Implemented in Xapian::ValuePostingSource, and Xapian::FixedWeightPostingSource.

6.23.3.4 virtual Xapian::doccount Xapian::PostingSource::get_termfreq_max () const [pure virtual]

An upper bound on the number of documents this object can return.

Xapian will always call init() on a PostingSource before calling this for the first time.

Implemented in Xapian::ValuePostingSource, and Xapian::FixedWeightPostingSource.

6.23.3.5 Xapian::weight Xapian::PostingSource::get_maxweight () const [inline]

Return the currently set upper bound on what get_weight() can return.

6.23.3.6 virtual Xapian::weight Xapian::PostingSource::get_weight () const [virtual]

Return the weight contribution for the current document.

This default implementation always returns 0, for convenience when implementing "weight-less" PostingSource subclasses.

This method may assume that it will only be called when there is a "current document". In detail: Xapian will always call init() on a PostingSource before calling this for the first time. It will also only call this if the PostingSource reports that it is pointing to a valid document (ie, it will not call it before calling at least one of next(), skip_to() or check(), and will ensure that the PostingSource is not at the end by calling at_end()).

Reimplemented in Xapian::ValueWeightPostingSource, Xapian::ValueMapPostingSource, and Xapian::FixedWeightPostingSource.

6.23.3.7 virtual Xapian::docid Xapian::PostingSource::get_docid () const [pure virtual]

Return the current docid.

This method may assume that it will only be called when there is a "current document". See *get_weight()* for details.

Note: in the case of a multi-database search, the returned docid should be in the single subdatabase relevant to this posting source. See the *init()* method for details.

Implemented in Xapian::ValuePostingSource, and Xapian::FixedWeightPostingSource.

6.23.3.8 virtual void Xapian::PostingSource::next (Xapian::weight *min_wt*) [pure virtual]

Advance the current position to the next matching document.

The PostingSource starts before the first entry in the list, so next() must be called before any methods which need the context of the current position.

Xapian will always call init() on a PostingSource before calling this for the first time.

Parameters:

min_wt The minimum weight contribution that is needed (this is just a hint which subclasses may ignore).

Implemented in Xapian::ValuePostingSource, and Xapian::FixedWeightPostingSource.

6.23.3.9 virtual void Xapian::PostingSource::skip_to (Xapian::docid *did*, Xapian::weight *min_wt*) [virtual]

Skip forward to the specified docid.

If the specified docid isn't in the list, position ourselves on the first document after it (or at_end() if no greater docids are present).

If the current position is already the specified docid, this method will leave the position unmodified.

If the specified docid is earlier than the current position, the behaviour is unspecified. A sensible behaviour would be to leave the current position unmodified, but it is also reasonable to move to the specified docid.

The default implementation calls next() repeatedly, which works but skip_to() can often be implemented much more efficiently.

Xapian will always call init() on a PostingSource before calling this for the first time.

Note: in the case of a multi-database search, the docid specified is the docid in the single subdatabase relevant to this posting source. See the *init()* method for details.

Parameters:

min_wt The minimum weight contribution that is needed (this is just a hint which subclasses may ignore).

Reimplemented in Xapian::ValuePostingSource, and Xapian::FixedWeightPostingSource.

6.23.3.10 virtual bool Xapian::PostingSource::check (Xapian::docid *did*, Xapian::weight *min_wt*) [virtual]

Check if the specified docid occurs.

The caller is required to ensure that the specified document id *did* actually exists in the database. If it does, it must move to that document id, and return true. If it does not, it may either:

- return true, having moved to a definite position (including "at_end"), which must be the same position as skip_to() would have moved to.

or

- return false, having moved to an "indeterminate" position, such that a subsequent call to next() or skip_to() will move to the next matching position after *did*.

Generally, this method should act like skip_to() and return true if that can be done at little extra cost.

Otherwise it should simply check if a particular docid is present, returning true if it is, and false if it isn't.

The default implementation calls skip_to() and always returns true.

Xapian will always call init() on a PostingSource before calling this for the first time.

Note: in the case of a multi-database search, the docid specified is the docid in the single subdatabase relevant to this posting source. See the *init()* method for details.

Reimplemented in Xapian::ValuePostingSource, and Xapian::FixedWeightPostingSource.

6.23.3.11 virtual bool Xapian::PostingSource::at_end () const [pure virtual]

Return true if the current position is past the last entry in this list.

At least one of *next()*, *skip_to()* or *check()* will be called before this method is first called.

Implemented in Xapian::ValuePostingSource, and Xapian::FixedWeightPostingSource.

6.23.3.12 virtual PostingSource* Xapian::PostingSource::clone () const [virtual]

Clone the posting source.

The clone should inherit the configuration of the parent, but need not inherit the state. ie, the clone does not need to be in the same iteration position as the original: the matcher will always call init() on the clone before attempting to move the iterator, or read the information about the current position of the iterator.

This may return NULL to indicate that cloning is not supported. In this case, the PostingSource may only be used with a single-database search.

The default implementation returns NULL.

Note that the returned object will be deallocated by Xapian after use with "delete". It must therefore have been allocated with "new".

Reimplemented in Xapian::ValueWeightPostingSource, Xapian::ValueMapPostingSource, and Xapian::FixedWeightPostingSource.

6.23.3.13 virtual std::string Xapian::PostingSource::name () const [virtual]

Name of the posting source class.

This is used when serialising and unserialising posting sources; for example, for performing remote searches.

If the subclass is in a C++ namespace, the namespace should be included in the name, using "::" as a separator. For example, for a PostingSource subclass called "FooPostingSource" in the "Xapian" namespace the result of this call should be "Xapian::FooPostingSource".

This should only be implemented if serialise() and unserialise() are also implemented. The default implementation returns an empty string.

If this returns an empty string, Xapian will assume that serialise() and unserialise() are not implemented.

Reimplemented in Xapian::ValueWeightPostingSource, Xapian::ValueMapPostingSource, and Xapian::FixedWeightPostingSource.

6.23.3.14 virtual std::string Xapian::PostingSource::serialise () const [virtual]

Serialise object parameters into a string.

The serialised parameters should represent the configuration of the posting source, but need not (indeed, should not) represent the current iteration state.

Reimplemented in Xapian::ValueWeightPostingSource, Xapian::ValueMapPostingSource, and Xapian::FixedWeightPostingSource.

6.23.3.15 virtual PostingSource* Xapian::PostingSource::unserialise (const std::string & *s*) const [virtual]

Create object given string serialisation returned by serialise().

Parameters:

s A serialised instance of this PostingSource subclass.

Reimplemented in Xapian::ValueWeightPostingSource, Xapian::ValueMapPostingSource, and Xapian::FixedWeightPostingSource.

6.23.3.16 virtual void Xapian::PostingSource::init (const Database & *db*) [pure virtual]

Set this PostingSource to the start of the list of postings.

This is called automatically by the matcher prior to each query being processed.

If a PostingSource is used for multiple searches, *init()* will therefore be called multiple times, and must handle this by using the database passed in the most recent call.

Parameters:

db The database which the PostingSource should iterate through.

Note: the database supplied to this method must not be modified: in particular, the reopen() method should not be called on it.

Note: in the case of a multi-database search, a separate PostingSource will be used for each database (the separate PostingSources will be obtained using *clone()*), and each PostingSource will be passed one of the sub-databases as the *db* parameter here. The *db* parameter will therefore always refer to a single database. All docids passed to, or returned from, the PostingSource refer to docids in that single database, rather than in the multi-database.

Implemented in Xapian::ValuePostingSource, Xapian::ValueWeightPostingSource, Xapian::ValueMapPostingSource, and Xapian::FixedWeightPostingSource.

6.23.3.17 virtual std::string Xapian::PostingSource::get_description () const [virtual]

Return a string describing this object.

This default implementation returns a generic answer. This default it provided to avoid forcing those deriving their own PostingSource subclass from having to implement this (they may not care what get_description() gives for their subclass).

Reimplemented in Xapian::ValueWeightPostingSource, Xapian::ValueMapPostingSource, and Xapian::FixedWeightPostingSource.

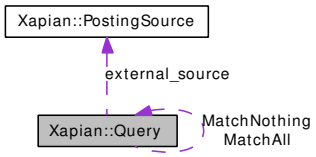
The documentation for this class was generated from the following file:

- include/xapian/postingsource.h

6.24 Xapian::Query Class Reference

`#include <query.h>`

Collaboration diagram for Xapian::Query:



Public Types

- enum op {
  OP_AND, OP_OR, OP_AND_NOT, OP_XOR,
  OP_AND_MAYBE, OP_FILTER, OP_NEAR, OP_PHRASE,
  OP_VALUE_RANGE, OP_SCALE_WEIGHT, OP_ELITE_SET, OP_VALUE_GE,
  OP_VALUE_LE, OP_SYNONYM }
  Enum of possible query operations.
- typedef std::vector< Internal * > subquery_list
  The container type for storing pointers to subqueries.
- typedef int op_t
  Type storing the operation.

Public Member Functions

- Query (const Query &copyme)
- Query & operator= (const Query &copyme)
- Query ()
- ~Query ()
- Query (const std::string &tname_, Xapian::termcount wqf_=1, Xapian::termpos pos_=0)
- Query (Query::op op_, const Query &left, const Query &right)
- Query (Query::op op_, const std::string &left, const std::string &right)
- template<class Iterator>
  Query (Query::op op_, Iterator qbegin, Iterator qend, Xapian::termcount parameter=0)
- Query (Query::op op_, Xapian::Query q, double parameter)
- Query (Query::op op_, Xapian::valueno valno, const std::string &begin, const std::string &end)

- Query (Query::op op_, Xapian::valueno valno, const std::string &value)
- Query (Xapian::PostingSource *external_source)
- Xapian::termcount get_length () const
- TermIterator get_terms_begin () const
- TermIterator get_terms_end () const
- bool empty () const
- std::string serialise () const
- std::string get_description () const

  Return a string describing this object.

- Internal (const Query::Internal &copyme)
- void operator= (const Query::Internal &copyme)
- Internal (const std::string &tname_, Xapian::termcount wqf_=1, Xapian::termpos term_pos_=0)
- Internal (op_t op_, Xapian::termcount parameter)
- Internal (op_t op_, Xapian::valueno valno, const std::string &begin, const std::string &end)
- Internal (op_t op_, Xapian::valueno valno, const std::string &value)
- Internal (Xapian::PostingSource *external_source_, bool owned)

  Construct an external source query.

- ~Internal ()
- void add_subquery (const Query::Internal *subq)
- void add_subquery_nocopy (Query::Internal *subq)
- void **set_dbl_parameter** (double dbl_parameter_)
- double **get_dbl_parameter** () const
- Query::Internal * end_construction ()
- std::string serialise () const
- std::string get_description () const

  Return a string describing this object.

- Xapian::termcount get_parameter () const
- Xapian::termcount get_length () const
- TermIterator get_terms () const

Static Public Member Functions

- static Query unserialise (const std::string &s)
- static Query unserialise (const std::string &s, const SerialisationContext &context)
- static Xapian::Query::Internal * **unserialise** (const std::string &s, const SerialisationContext &context)

Static Public Attributes

- static const Xapian::Query MatchAll
- static const Xapian::Query MatchNothing
- static const int **OP_LEAF** = -1
- static const int **OP_EXTERNAL_SOURCE** = -2

Friends

- class **::LocalSubMatch**
- class **::MultiMatch**
- class **::QueryOptimiser**
- struct **::SortPosName**
- class **Query**

6.24.1 Detailed Description

Class representing a query.

Queries are represented as a tree of objects.

6.24.2 Member Typedef Documentation

6.24.2.1 typedef std::vector<Internal *> Xapian::Query::subquery_list

The container type for storing pointers to subqueries.

6.24.2.2 typedef int Xapian::Query::op_t

Type storing the operation.

6.24.3 Member Enumeration Documentation

6.24.3.1 enum Xapian::Query::op

Enum of possible query operations.

Enumerator:

OP_AND Return iff both subqueries are satisfied.

OP_OR Return if either subquery is satisfied.

OP_AND_NOT Return if left but not right satisfied.

OP_XOR Return if one query satisfied, but not both.

OP_AND_MAYBE Return iff left satisfied, but use weights from both.

OP_FILTER As AND, but use only weights from left subquery.

OP_NEAR Find occurrences of a list of terms with all the terms occurring within a specified window of positions. Each occurrence of a term must be at a different position, but the order they appear in is irrelevant.

The window parameter should be specified for this operation, but will default to the number of terms in the list.

OP_PHRASE Find occurrences of a list of terms with all the terms occurring within a specified window of positions, and all the terms appearing in the order specified. Each occurrence of a term must be at a different position.

The window parameter should be specified for this operation, but will default to the number of terms in the list.

OP_VALUE_RANGE Filter by a range test on a document value.

OP_SCALE_WEIGHT Scale the weight of a subquery by the specified factor.

A factor of 0 means this subquery will contribute no weight to the query - it will act as a purely boolean subquery.

If the factor is negative, Xapian::InvalidArgumentError will be thrown.

OP_ELITE_SET Select an elite set from the subqueries, and perform a query with these combined as an OR query.

OP_VALUE_GE Filter by a greater-than-or-equal test on a document value.

OP_VALUE_LE Filter by a less-than-or-equal test on a document value.

OP_SYNONYM Treat a set of queries as synonyms.

This returns all results which match at least one of the queries, but weighting as if all the sub-queries are instances of the same term: so multiple matching terms for a document increase the wdf value used, and the term frequency is based on the number of documents which would match an OR of all the subqueries.

The term frequency used will usually be an approximation, because calculating the precise combined term frequency would be overly expensive.

Identical to OP_OR, except for the weightings returned.

6.24.4 Constructor & Destructor Documentation

6.24.4.1 Xapian::Query::Query (const Query & *copyme*)

Copy constructor.

6.24.4.2 Xapian::Query::Query ()

Default constructor: makes an empty query which matches no documents.

Also useful for defining a Query object to be assigned to later.

An exception will be thrown if an attempt is made to use an undefined query when building up a composite query.

6.24.4.3 Xapian::Query::~Query ()

Destructor.

6.24.4.4 Xapian::Query::Query (const std::string & *tname_*, Xapian::termcount *wqf_* = 1, Xapian::termpos *pos_* = 0)

A query consisting of a single term.

6.24.4.5 Xapian::Query::Query (Query::op *op_*, const Query & *left*, const Query & *right*)

A query consisting of two subqueries, opp-ed together.

6.24.4.6 Xapian::Query::Query (Query::op *op_*, const std::string & *left*, const std::string & *right*)

A query consisting of two termnames opp-ed together.

6.24.4.7 template<class Iterator> Xapian::Query::Query (Query::op *op_*, Iterator *qbegin*, Iterator *qend*, Xapian::termcount *parameter* = 0) [inline]

Combine a number of Xapian::Query-s with the specified operator.

The Xapian::Query objects are specified with begin and end iterators.

AND, OR, NEAR and PHRASE can take any number of subqueries. Other operators take exactly two subqueries.

The iterators may be to Xapian::Query objects, pointers to Xapian::Query objects, or termnames (std::string-s).

For NEAR and PHRASE, a window size can be specified in parameter.

For ELITE_SET, the elite set size can be specified in parameter.

6.24.4.8 Xapian::Query::Query (Query::op *op_*, Xapian::Query *q*, double *parameter*)

Apply the specified operator to a single Xapian::Query object, with a double parameter.

6.24.4.9 Xapian::Query::Query (Query::op *op_*, Xapian::valueno *valno*, const std::string & *begin*, const std::string & *end*)

Construct a value range query on a document value.

A value range query matches those documents which have a value stored in the slot given by *valno* which is in the range specified by *begin* and *end* (in lexicographical order), including the endpoints.

Parameters:

op_ The operator to use for the query. Currently, must be OP_VALUE_RANGE.

valno The slot number to get the value from.

begin The start of the range.

end The end of the range.

6.24.4.10 Xapian::Query::Query (Query::op *op_*, Xapian::valueno *valno*, const std::string & *value*)

Construct a value comparison query on a document value.

This query matches those documents which have a value stored in the slot given by *valno* which compares, as specified by the operator, to *value*.

Parameters:

op_ The operator to use for the query. Currently, must be OP_VALUE_GE or OP_VALUE_LE.

valno The slot number to get the value from.

value The value to compare.

6.24.4.11 Xapian::Query::Query (Xapian::PostingSource * *external_source*) [explicit]

Construct an external source query.

An attempt to clone the posting source will be made immediately, so if the posting source supports clone(), the source supplied may be safely deallocated after this call. If the source does not support clone(), the caller must ensure that the posting source remains valid until the Query is deallocated.

Parameters:

external_source The source to use in the query.

6.24.4.12 Xapian::Query::~Internal ()

Destructor.

6.24.5 Member Function Documentation

6.24.5.1 Query& Xapian::Query::operator= (const Query & *copyme*)

Assignment.

6.24.5.2 Xapian::termcount Xapian::Query::get_length () const

Get the length of the query, used by some ranking formulae. This value is calculated automatically - if you want to override it you can pass a different value to Enquire::set_query().

6.24.5.3 TermIterator Xapian::Query::get_terms_begin () const

Return a Xapian::TermIterator returning all the terms in the query, in order of termpos. If multiple terms have the same term position, their order is unspecified. Duplicates (same term and termpos) will be removed.

6.24.5.4 TermIterator Xapian::Query::get_terms_end () const [inline]

Return a Xapian::TermIterator to the end of the list of terms in the query.

6.24.5.5 bool Xapian::Query::empty () const

Test if the query is empty (i.e. was constructed using the default ctor or with an empty iterator ctor).

6.24.5.6 std::string Xapian::Query::serialise () const

Serialise query into a string.

The query representation may change between Xapian releases: even between minor versions. However, it is guaranteed not to change unless the remote database protocol has also changed between releases.

6.24.5.7 static Query Xapian::Query::unserialise (const std::string & *s*) [static]

Unserialise a query from a string produced by serialise().

This method will fail if the query contains any external PostingSource leaf nodes.

Parameters:

s The string representing the serialised query.

6.24.5.8 static Query Xapian::Query::unserialise (const std::string & *s*, const SerialisationContext & *context*) [static]

Unserialise a query from a string produced by serialise().

The supplied context will be used to attempt to unserialise any external PostingSource leaf nodes. This method will fail if the query contains any external PostingSource leaf nodes which are not registered in the context.

Parameters:

s The string representing the serialised query.

context A context to use when unserialising the query.

6.24.5.9 std::string Xapian::Query::get_description () const

Return a string describing this object.

6.24.5.10 Xapian::Query::Internal (const Query::Internal & *copyme*)

Copy constructor.

6.24.5.11 void Xapian::Query::operator= (const Query::Internal & *copyme*)

Assignment.

6.24.5.12 Xapian::Query::Internal (const std::string & *tname_*, Xapian::termcount *wqf_* = 1, Xapian::termpos *term_pos_* = 0) [explicit]

A query consisting of a single term.

6.24.5.13 Xapian::Query::Internal (op_t *op_*, Xapian::termcount *parameter*)

Create internals given only the operator and a parameter.

6.24.5.14 Xapian::Query::Internal (op_t *op_*, Xapian::valueno *valno*, const std::string & *begin*, const std::string & *end*)

Construct a range query on a document value.

6.24.5.15 Xapian::Query::Internal (op_t *op_*, Xapian::valueno *valno*, const std::string & *value*)

Construct a value greater-than-or-equal query on a document value.

6.24.5.16 Xapian::Query::Internal (Xapian::PostingSource * *external_source_*, bool *owned*) [explicit]

Construct an external source query.

6.24.5.17 void Xapian::Query::add_subquery (const Query::Internal * *subq*)

Add a subquery.

6.24.5.18 void Xapian::Query::add_subquery_nocopy (Query::Internal * *subq*)

Add a subquery without copying it.

subq is owned by the object this is called on after the call.

6.24.5.19 Query::Internal* Xapian::Query::end_construction ()

Finish off the construction.

6.24.5.20 std::string Xapian::Query::serialise () const [inline]

Return a string in an easily parsed form which contains all the information in a query.

6.24.5.21 std::string Xapian::Query::get_description () const

Return a string describing this object.

6.24.5.22 Xapian::termcount Xapian::Query::get_parameter () const [inline]

Get the numeric parameter used in this query.

This is used by the QueryParser to get the value number for VALUE_RANGE queries. It should be replaced by a public method on the Query class at some point, but the API which should be used for that is unclear, so this is a temporary workaround.

6.24.5.23 Xapian::termcount Xapian::Query::get_length () const

Get the length of the query, used by some ranking formulae. This value is calculated automatically - if you want to override it you can pass a different value to Enquire::set_query().

6.24.5.24 TermIterator Xapian::Query::get_terms () const

Return an iterator over all the terms in the query, in order of termpos. If multiple terms have the same term position, their order is unspecified. Duplicates (same term and termpos) will be removed.

6.24.6 Member Data Documentation

6.24.6.1 const Xapian::Query Xapian::Query::MatchAll [static]

A query which matches all documents in the database.

6.24.6.2 const Xapian::Query Xapian::Query::MatchNothing [static]

A query which matches no documents.

The documentation for this class was generated from the following file:

- include/xapian/query.h

6.25 Xapian::QueryParser Class Reference

Build a Xapian::Query object from a user query string.

`#include <queryparser.h>`

Public Types

- enum feature_flag {

  FLAG_BOOLEAN = 1, FLAG_PHRASE = 2, FLAG_LOVEHATE = 4, FLAG_BOOLEAN_ANY_CASE = 8,

  FLAG_WILDCARD = 16, FLAG_PURE_NOT = 32, FLAG_PARTIAL = 64, FLAG_SPELLING_CORRECTION = 128,

  FLAG_SYNONYM = 256, FLAG_AUTO_SYNONYMS = 512, FLAG_AUTO_MULTIWORD_SYNONYMS = 1024 | FLAG_AUTO_SYNONYMS, FLAG_DEFAULT = FLAG_PHRASE|FLAG_BOOLEAN|FLAG_LOVEHATE }

  Enum of feature flags.

- enum **stem_strategy** { **STEM_NONE**, **STEM_SOME**, **STEM_ALL** }

Public Member Functions

- QueryParser (const QueryParser &o)

  Copy constructor.

- QueryParser & operator= (const QueryParser &o)

  Assignment.

- QueryParser ()

  Default constructor.

- ~QueryParser ()

  Destructor.

- void set_stemmer (const Xapian::Stem &stemmer)
- void set_stemming_strategy (stem_strategy strategy)
- void set_stopper (const Stopper *stop=NULL)

  Set the stopper.

- void set_default_op (Query::op default_op)
- Query::op get_default_op () const
- void set_database (const Database &db)

  Specify the database being searched.

- Query parse_query (const std::string &query_string, unsigned flags=FLAG_DEFAULT, const std::string &default_prefix=std::string())
- void add_prefix (const std::string &field, const std::string &prefix)
- void add_boolean_prefix (const std::string &field, const std::string &prefix)
- TermIterator stoplist_begin () const

  Iterate over terms omitted from the query as stopwords.

- TermIterator **stoplist_end** () const
- TermIterator unstem_begin (const std::string &term) const

  Iterate over unstemmed forms of the given (stemmed) term used in the query.

- TermIterator **unstem_end** (const std::string &) const
- void add_valuerangeprocessor (Xapian::ValueRangeProcessor *vrproc)

  Register a ValueRangeProcessor.

- std::string get_corrected_query_string () const
- std::string get_description () const

  Return a string describing this object.

6.25.1 Detailed Description

Build a Xapian::Query object from a user query string.

6.25.2 Member Enumeration Documentation

6.25.2.1 enum Xapian::QueryParser::feature_flag

Enum of feature flags.

Enumerator:

FLAG_BOOLEAN Support AND, OR, etc and bracketed subexpressions.

FLAG_PHRASE Support quoted phrases.

FLAG_LOVEHATE Support + and -.

FLAG_BOOLEAN_ANY_CASE Support AND, OR, etc even if they aren't in ALLCAPS.

FLAG_WILDCARD Support right truncation (e.g. Xap*).

NB: You need to tell the QueryParser object which database to expand wildcards from by calling set_database.

FLAG_PURE_NOT Allow queries such as 'NOT apples'.

These require the use of a list of all documents in the database which is potentially expensive, so this feature isn't enabled by default.

FLAG_PARTIAL Enable partial matching.

Partial matching causes the parser to treat the query as a "partially entered" search. This will automatically treat the final word as a wildcarded match, unless it is followed by whitespace, to produce more stable results from interactive searches.

NB: You need to tell the QueryParser object which database to expand wildcards from by calling set_database.

FLAG_SPELLING_CORRECTION Enable spelling correction.

For each word in the query which doesn't exist as a term in the database, Database::get_spelling_suggestion() will be called and if a suggestion is returned, a corrected version of the query string will be built up which can be read using QueryParser::get_corrected_query_string(). The query returned is based on the uncorrected query string however - if you want a parsed query based on the corrected query string, you must call QueryParser::parse_query() again.

NB: You must also call set_database() for this to work.

FLAG_SYNONYM Enable synonym operator '~'.

NB: You must also call set_database() for this to work.

FLAG_AUTO_SYNONYMS Enable automatic use of synonyms for single terms.

NB: You must also call set_database() for this to work.

FLAG_AUTO_MULTIWORD_SYNONYMS Enable automatic use of synonyms for single terms and groups of terms.

NB: You must also call set_database() for this to work.

FLAG_DEFAULT The default flags.

Used if you don't explicitly pass any to *parse_query()*.

Added in Xapian 1.0.11.

6.25.3 Constructor & Destructor Documentation

6.25.3.1 Xapian::QueryParser::QueryParser (const QueryParser & *o*)

Copy constructor.

6.25.3.2 Xapian::QueryParser::QueryParser ()

Default constructor.

6.25.3.3 Xapian::QueryParser::~QueryParser ()

Destructor.

6.25.4 Member Function Documentation

6.25.4.1 QueryParser& Xapian::QueryParser::operator= (const QueryParser & *o*)

Assignment.

6.25.4.2 void Xapian::QueryParser::set_stemmer (const Xapian::Stem & *stemmer*)

Set the stemmer.

This sets the stemming algorithm which will be used by the query parser. Note that the stemming algorithm will only be used according to the stemming strategy set by set_stemming_strategy(), which defaults to STEM_NONE. Therefore, to use a stemming algorithm, you will also need to call set_stemming_strategy() with a value other than STEM_NONE.

6.25.4.3 void Xapian::QueryParser::set_stemming_strategy (stem_strategy *strategy*)

Set the stemming strategy.

This controls how the query parser will apply the stemming algorithm. The default value is STEM_NONE. The possible values are:

- STEM_NONE: Don't perform any stemming.
- STEM_SOME: Search for stemmed forms of terms except for those which start with a capital letter, or are followed by certain characters (currently: (/@<>=*[{"), or are used with operators which need positional information. Stemmed terms are prefixed with 'Z'.
- STEM_ALL: Search for stemmed forms of all words (note: no 'Z' prefix is added).

Note that the stemming algorithm is only applied to words in probabilistic fields - boolean filter terms are never stemmed.

6.25.4.4 void Xapian::QueryParser::set_stopper (const Stopper * *stop* = NULL)

Set the stopper.

6.25.4.5 void Xapian::QueryParser::set_default_op (Query::op *default_op*)

Set the default boolean operator.

6.25.4.6 Query::op Xapian::QueryParser::get_default_op () const

Get the default boolean operator.

6.25.4.7 void Xapian::QueryParser::set_database (const Database & *db*)

Specify the database being searched.

6.25.4.8 Query Xapian::QueryParser::parse_query (const std::string & *query_string*, unsigned *flags* = FLAG_DEFAULT, const std::string & *default_prefix* = std::string())

Parse a query.

Parameters:

query_string A free-text query as entered by a user

flags Zero or more Query::feature_flag specifying what features the QueryParser should support. Combine multiple values with bitwise-or (|) (default FLAG_DEFAULT).

default_prefix The default term prefix to use (default none). For example, you can pass "A" when parsing an "Author" field.

6.25.4.9 void Xapian::QueryParser::add_prefix (const std::string & *field*, const std::string & *prefix*)

Add a probabilistic term prefix.

For example:

```
qp.add_prefix("author", "A");
```

This allows the user to search for author:Orwell which will be converted to a search for the term "Aorwell".

Multiple fields can be mapped to the same prefix. For example, you can make title: and subject: aliases for each other.

As of 1.0.4, you can call this method multiple times with the same value of field to allow a single field to be mapped to multiple prefixes. Multiple terms being generated for such a field, and combined with Xapian::Query::OP_OR.

If any prefixes are specified for the empty field name (i.e. you call this method with an empty string as the first parameter) these prefixes will be used as the default prefix. If you do this and also specify the default_prefix parameter to parse_query(), then the default_prefix parameter will override.

If you call add_prefix() and add_boolean_prefix() for the same value of *field*, a Xapian::InvalidOperationError exception will be thrown.

In 1.0.3 and earlier, subsequent calls to this method with the same value of *field* had no effect.

Parameters:

field The user visible field name

prefix The term prefix to map this to

6.25.4.10 void Xapian::QueryParser::add_boolean_prefix (const std::string & *field*, const std::string & *prefix*)

Add a boolean term prefix allowing the user to restrict a search with a boolean filter specified in the free text query.

For example:

```
qp.add_boolean_prefix("site", "H");
```

This allows the user to restrict a search with site:xapian.org which will be converted to Hxapian.org combined with any probabilistic query with `Xapian::Query::OP_FILTER`.

If multiple boolean filters are specified in a query for the same prefix, they will be combined with the `Xapian::Query::OP_OR` operator. Then, if there are boolean filters for different prefixes, they will be combined with the `Xapian::Query::OP_AND` operator.

Multiple fields can be mapped to the same prefix (so for example you can make site: and domain: aliases for each other). Instances of fields with different aliases but the same prefix will still be combined with the OR operator.

For example, if "site" and "domain" map to "H", but author maps to "A", a search for "site:foo domain:bar author:Fred" will map to "(Hfoo OR Hbar) AND Afred".

As of 1.0.4, you can call this method multiple times with the same value of field to allow a single field to be mapped to multiple prefixes. Multiple terms being generated for such a field, and combined with `Xapian::Query::OP_OR`.

Calling this method with an empty string for *field* will cause a `Xapian::InvalidArgumentError`.

If you call `add_prefix()` and `add_boolean_prefix()` for the same value of *field*, a `Xapian::InvalidOperationError` exception will be thrown.

In 1.0.3 and earlier, subsequent calls to this method with the same value of *field* had no effect.

Parameters:

field The user visible field name

prefix The term prefix to map this to

6.25.4.11 TermIterator Xapian::QueryParser::stoplist_begin () const

Iterate over terms omitted from the query as stopwords.

6.25.4.12 TermIterator Xapian::QueryParser::unstem_begin (const std::string & *term*) const

Iterate over unstemmed forms of the given (stemmed) term used in the query.

6.25.4.13 void Xapian::QueryParser::add_valuerangeprocessor (Xapian::ValueRangeProcessor * *vrproc*)

Register a ValueRangeProcessor.

6.25.4.14 std::string Xapian::QueryParser::get_corrected_query_string () const

Get the spelling-corrected query string.

This will only be set if FLAG_SPELLING_CORRECTION is specified when QueryParser::parse_query() was last called.

If there were no corrections, an empty string is returned.

6.25.4.15 std::string Xapian::QueryParser::get_description () const

Return a string describing this object.

The documentation for this class was generated from the following file:

- include/xapian/queryparser.h

6.26 Xapian::ReplicationInfo Struct Reference

`#include <replication.h>`

Public Member Functions

- void **clear** ()

Public Attributes

- int changeset_count

  Number of changesets applied.

- int fullcopy_count

  Number of times a full database copy was performed.

- bool changed

6.26.1 Detailed Description

Information about the steps involved in performing a replication.

Warning: the replication interface is currently experimental, and is liable to change between releases without warning.

6.26.2 Member Data Documentation

6.26.2.1 int Xapian::ReplicationInfo::changeset_count

Number of changesets applied.

6.26.2.2 int Xapian::ReplicationInfo::fullcopy_count

Number of times a full database copy was performed.

6.26.2.3 bool Xapian::ReplicationInfo::changed

True if and only if the replication corresponds to a change in the live version of the database. Note that this may be false even if changeset_count and fullcopy_count are non-zero, since the changes may have been made to a non-live copy of the database.

The documentation for this struct was generated from the following file:

- include/xapian/replication.h

6.27 Xapian::RSet Class Reference

`#include <enquire.h>`

Public Member Functions

- RSet (const RSet &rset)

  Copy constructor.

- void operator= (const RSet &rset)

  Assignment operator.

- RSet ()

  Default constructor.

- ~RSet ()

  Destructor.

- Xapian::doccount size () const
- bool empty () const
- void add_document (Xapian::docid did)

  Add a document to the relevance set.

- void add_document (const Xapian::MSetIterator &i)

  Add a document to the relevance set.

- void remove_document (Xapian::docid did)

  Remove a document from the relevance set.

- void remove_document (const Xapian::MSetIterator &i)

  Remove a document from the relevance set.

- bool contains (Xapian::docid did) const

  Test if a given document in the relevance set.

- bool contains (const Xapian::MSetIterator &i) const

  Test if a given document in the relevance set.

- std::string get_description () const

  Return a string describing this object.

Public Attributes

- Xapian::Internal::RefCntPtr< Internal > **internal**

6.27.1 Detailed Description

A relevance set (R-Set). This is the set of documents which are marked as relevant, for use in modifying the term weights, and in performing query expansion.

6.27.2 Constructor & Destructor Documentation

6.27.2.1 Xapian::RSet::RSet (const RSet & *rset*)

Copy constructor.

6.27.2.2 Xapian::RSet::RSet ()

Default constructor.

6.27.2.3 Xapian::RSet::~RSet ()

Destructor.

6.27.3 Member Function Documentation

6.27.3.1 void Xapian::RSet::operator= (const RSet & *rset*)

Assignment operator.

6.27.3.2 Xapian::doccount Xapian::RSet::size () const

The number of documents in this R-Set

6.27.3.3 bool Xapian::RSet::empty () const

Test if this R-Set is empty

6.27.3.4 void Xapian::RSet::add_document (Xapian::docid *did*)

Add a document to the relevance set.

6.27.3.5 void Xapian::RSet::add_document (const Xapian::MSetIterator & *i*) [inline]

Add a document to the relevance set.

6.27.3.6 void Xapian::RSet::remove_document (Xapian::docid *did*)

Remove a document from the relevance set.

6.27.3.7 void Xapian::RSet::remove_document (const Xapian::MSetIterator & *i*) [inline]

Remove a document from the relevance set.

6.27.3.8 bool Xapian::RSet::contains (Xapian::docid *did*) const

Test if a given document in the relevance set.

6.27.3.9 bool Xapian::RSet::contains (const Xapian::MSetIterator & *i*) const [inline]

Test if a given document in the relevance set.

6.27.3.10 std::string Xapian::RSet::get_description () const

Return a string describing this object.

The documentation for this class was generated from the following file:

- include/xapian/enquire.h

6.28 Xapian::SerialisationContext Class Reference

`#include <serialisationcontext.h>`

Public Member Functions

- SerialisationContext (const SerialisationContext &other)
- SerialisationContext & operator= (const SerialisationContext &other)
- SerialisationContext ()
- void register_weighting_scheme (const Xapian::Weight &wt)

  Register a weighting scheme with the context.

- const Xapian::Weight * get_weighting_scheme (const std::string &name) const
- void register_posting_source (const Xapian::PostingSource &source)

  Register a user-defined posting source class.

- const Xapian::PostingSource * get_posting_source (const std::string &name) const

6.28.1 Detailed Description

A context for serialisation.

This context is used to look up weighting schemes and posting sources when unserialising.

6.28.2 Constructor & Destructor Documentation

6.28.2.1 Xapian::SerialisationContext::SerialisationContext (const SerialisationContext & *other*)

Copy the context.

The internals are reference counted, so copying is cheap.

6.28.2.2 Xapian::SerialisationContext::SerialisationContext ()

Default constructor: makes a context with default settings.

The context will contain all standard weighting schemes and posting sources.

6.28.3 Member Function Documentation

6.28.3.1 SerialisationContext& Xapian::SerialisationContext::operator= (const SerialisationContext & *other*)

Assign to the context - the copy is shallow.

The internals are reference counted, so assignment is cheap.

6.28.3.2 void Xapian::SerialisationContext::register_weighting_scheme (const Xapian::Weight & *wt*)

Register a weighting scheme with the context.

6.28.3.3 const Xapian::Weight* Xapian::SerialisationContext::get_weighting_scheme (const std::string & *name*) const

Get a weighting scheme given a name.

The returned weighting scheme is owned by the context object.

Returns NULL if the weighting scheme could not be found.

6.28.3.4 void Xapian::SerialisationContext::register_posting_source (const Xapian::PostingSource & *source*)

Register a user-defined posting source class.

6.28.3.5 const Xapian::PostingSource* Xapian::SerialisationContext::get_posting_source (const std::string & *name*) const

Get a posting source given a name.

The returned posting source is owned by the context object.

Returns NULL if the posting source could not be found.

The documentation for this class was generated from the following file:

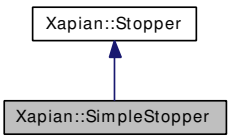
- include/xapian/serialisationcontext.h

6.29 Xapian::SimpleStopper Class Reference

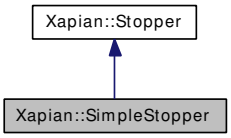
Simple implementation of Stopper class - this will suit most users.

`#include <queryparser.h>`

Inheritance diagram for Xapian::SimpleStopper:



Collaboration diagram for Xapian::SimpleStopper:



Public Member Functions

- SimpleStopper ()

  Default constructor.

- template<class Iterator>
  SimpleStopper (Iterator begin, Iterator end)

  Initialise from a pair of iterators.

- void add (const std::string &word)

  Add a single stop word.

- virtual bool operator() (const std::string &term) const

  Is term a stop-word?

- virtual std::string get_description () const

  Return a string describing this object.

6.29.1 Detailed Description

Simple implementation of Stopper class - this will suit most users.

6.29.2 Constructor & Destructor Documentation

6.29.2.1 Xapian::SimpleStopper::SimpleStopper () [inline]

Default constructor.

6.29.2.2 template<class Iterator> Xapian::SimpleStopper::SimpleStopper (Iterator *begin*, Iterator *end*) [inline]

Initialise from a pair of iterators.

6.29.3 Member Function Documentation

6.29.3.1 void Xapian::SimpleStopper::add (const std::string & *word*) [inline]

Add a single stop word.

6.29.3.2 virtual bool Xapian::SimpleStopper::operator() (const std::string & *term*) const [inline, virtual]

Is term a stop-word?

Implements Xapian::Stopper.

6.29.3.3 virtual std::string Xapian::SimpleStopper::get_description () const [virtual]

Return a string describing this object.

Reimplemented from Xapian::Stopper.

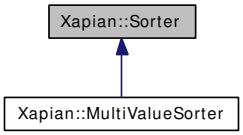
The documentation for this class was generated from the following file:

- include/xapian/queryparser.h

6.30 Xapian::Sorter Class Reference

`#include <sorter.h>`

Inheritance diagram for Xapian::Sorter:



Public Member Functions

- virtual std::string operator() (const Xapian::Document &doc) const=0
- virtual ~Sorter ()

6.30.1 Detailed Description

Virtual base class for sorter functor.

6.30.2 Constructor & Destructor Documentation

6.30.2.1 virtual Xapian::Sorter::~Sorter () [virtual]

Virtual destructor, because we have virtual methods.

6.30.3 Member Function Documentation

6.30.3.1 virtual std::string Xapian::Sorter::operator() (const Xapian::Document & *doc*) const [pure virtual]

This method takes a Document object and builds a sort key from it.

Documents are then ordered by a string compare on the sort keys.

Implemented in Xapian::MultiValueSorter.

The documentation for this class was generated from the following file:

- include/xapian/sorter.h

6.31 Xapian::Stem Class Reference

Class representing a stemming algorithm.

`#include <stem.h>`

Public Member Functions

- Stem (const Stem &o)

  Copy constructor.

- void operator= (const Stem &o)

  Assignment.

- Stem ()
- Stem (const std::string &language)
- ~Stem ()

  Destructor.

- std::string operator() (const std::string &word) const
- std::string get_description () const

  Return a string describing this object.

Static Public Member Functions

- static std::string get_available_languages ()

6.31.1 Detailed Description

Class representing a stemming algorithm.

6.31.2 Constructor & Destructor Documentation

6.31.2.1 Xapian::Stem::Stem (const Stem & *o*)

Copy constructor.

6.31.2.2 Xapian::Stem::Stem ()

Construct a Xapian::Stem object which doesn't change terms.

Equivalent to Stem("none").

6.31.2.3 Xapian::Stem::Stem (const std::string & *language*) [explicit]

Construct a Xapian::Stem object for a particular language.

Parameters:

language Either the English name for the language or the two letter ISO639 code.

The following language names are understood (aliases follow the name):

- none - don't stem terms
- danish (da)
- dutch (nl)
- english (en) - Martin Porter's 2002 revision of his stemmer
- english_lovins (lovins) - Lovin's stemmer
- english_porter (porter) - Porter's stemmer as described in his 1980 paper
- finnish (fi)
- french (fr)
- german (de)
- italian (it)
- norwegian (no)
- portuguese (pt)
- russian (ru)
- spanish (es)
- swedish (sv)

Exceptions:

Xapian::InvalidArgumentError is thrown if language isn't recognised.

6.31.2.4 Xapian::Stem::~Stem ()

Destructor.

6.31.3 Member Function Documentation

6.31.3.1 void Xapian::Stem::operator= (const Stem & *o*)

Assignment.

6.31.3.2 std::string Xapian::Stem::operator() (const std::string & *word*) const

Stem a word.

Parameters:

word a word to stem.

Returns:

the stem

6.31.3.3 std::string Xapian::Stem::get_description () const

Return a string describing this object.

6.31.3.4 static std::string Xapian::Stem::get_available_languages () [static]

Return a list of available languages.

Each stemmer is only included once in the list (not once for each alias). The name included is the English name of the language.

The list is returned as a string, with language names separated by spaces. This is a static method, so a Xapian::Stem object is not required for this operation.

The documentation for this class was generated from the following file:

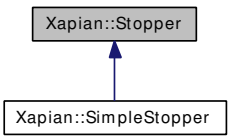
- include/xapian/stem.h

6.32 Xapian::Stopper Class Reference

Base class for stop-word decision functor.

`#include <queryparser.h>`

Inheritance diagram for Xapian::Stopper:



Public Member Functions

- virtual bool operator() (const std::string &term) const =0

  Is term a stop-word?

- virtual ~Stopper ()

  Class has virtual methods, so provide a virtual destructor.

- virtual std::string get_description () const

  Return a string describing this object.

6.32.1 Detailed Description

Base class for stop-word decision functor.

6.32.2 Constructor & Destructor Documentation

6.32.2.1 virtual Xapian::Stopper::~Stopper () `[inline, virtual]`

Class has virtual methods, so provide a virtual destructor.

6.32.3 Member Function Documentation

6.32.3.1 virtual bool Xapian::Stopper::operator() (const std::string & *term*) const `[pure virtual]`

Is term a stop-word?

Implemented in Xapian::SimpleStopper.

6.32.3.2 virtual std::string Xapian::Stopper::get_description () const [virtual]

Return a string describing this object.

Reimplemented in Xapian::SimpleStopper.

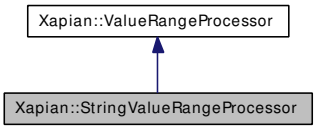
The documentation for this class was generated from the following file:

- include/xapian/queryparser.h

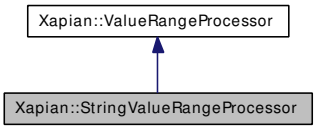
6.33 Xapian::StringValueRangeProcessor Class Reference

`#include <queryparser.h>`

Inheritance diagram for Xapian::StringValueRangeProcessor:



Collaboration diagram for Xapian::StringValueRangeProcessor:



Public Member Functions

- StringValueRangeProcessor (Xapian::valueno valno_)
- Xapian::valueno operator() (std::string &, std::string &)

  Any strings are valid as begin and end.

6.33.1 Detailed Description

Handle a string range.

The end points can be any strings.

6.33.2 Constructor & Destructor Documentation

6.33.2.1 Xapian::StringValueRangeProcessor::StringValueRangeProcessor (Xapian::valueno *valno_*) `[inline]`

Constructor.

Parameters:

valno_ The value number to return from operator().

6.33.3 Member Function Documentation

6.33.3.1 Xapian::valueno Xapian::StringValueRangeProcessor::operator() (std::string &, std::string &) [inline, virtual]

Any strings are valid as begin and end.

Implements Xapian::ValueRangeProcessor.

The documentation for this class was generated from the following file:

- include/xapian/queryparser.h

6.34 Xapian::TermGenerator Class Reference

`#include <termgenerator.h>`

Public Types

- enum flags { FLAG_SPELLING = 128 }

  Flags to OR together and pass to TermGenerator::set_flags().

Public Member Functions

- TermGenerator (const TermGenerator &o)

  Copy constructor.

- TermGenerator & operator= (const TermGenerator &o)

  Assignment.

- TermGenerator ()

  Default constructor.

- ~TermGenerator ()

  Destructor.

- void set_stemmer (const Xapian::Stem &stemmer)

  Set the Xapian::Stem object to be used for generating stemmed terms.

- void set_stopper (const Xapian::Stopper *stop=NULL)

  Set the Xapian::Stopper object to be used for identifying stopwords.

- void set_document (const Xapian::Document &doc)

  Set the current document.

- const Xapian::Document & get_document () const

  Get the current document.

- void set_database (const Xapian::WritableDatabase &db)

  Set the database to index spelling data to.

- flags set_flags (flags toggle, flags mask=flags(0))
- void index_text (const Xapian::Utf8Iterator &itor, Xapian::termcount weight=1, const std::string &prefix=std::string())
- void index_text (const std::string &text, Xapian::termcount weight=1, const std::string &prefix=std::string())
- void index_text_without_positions (const Xapian::Utf8Iterator &itor, Xapian::termcount weight=1, const std::string &prefix=std::string())

- void index_text_without_positions (const std::string &text, Xapian::termcount weight=1, const std::string &prefix=std::string())
- void increase_termpos (Xapian::termcount delta=100)
- Xapian::termcount get_termpos () const

  Get the current term position.

- void set_termpos (Xapian::termcount termpos)

  Set the current term position.

- std::string get_description () const

  Return a string describing this object.

6.34.1 Detailed Description

Parses a piece of text and generate terms.

This module takes a piece of text and parses it to produce words which are then used to generate suitable terms for indexing. The terms generated are suitable for use with Query objects produced by the QueryParser class.

6.34.2 Member Enumeration Documentation

6.34.2.1 enum Xapian::TermGenerator::flags

Flags to OR together and pass to TermGenerator::set_flags().

Enumerator:

FLAG_SPELLING Index data required for spelling correction.

6.34.3 Constructor & Destructor Documentation

6.34.3.1 Xapian::TermGenerator::TermGenerator (const TermGenerator & *o*)

Copy constructor.

6.34.3.2 Xapian::TermGenerator::TermGenerator ()

Default constructor.

6.34.3.3 Xapian::TermGenerator::~TermGenerator ()

Destructor.

6.34.4 Member Function Documentation

6.34.4.1 TermGenerator& Xapian::TermGenerator::operator= (const TermGenerator & *o*)

Assignment.

6.34.4.2 void Xapian::TermGenerator::set_stemmer (const Xapian::Stem & *stemmer*)

Set the Xapian::Stem object to be used for generating stemmed terms.

6.34.4.3 void Xapian::TermGenerator::set_stopper (const Xapian::Stopper * *stop* = NULL)

Set the Xapian::Stopper object to be used for identifying stopwords.

6.34.4.4 void Xapian::TermGenerator::set_document (const Xapian::Document & *doc*)

Set the current document.

6.34.4.5 const Xapian::Document& Xapian::TermGenerator::get_document () const

Get the current document.

6.34.4.6 void Xapian::TermGenerator::set_database (const Xapian::WritableDatabase & *db*)

Set the database to index spelling data to.

6.34.4.7 flags Xapian::TermGenerator::set_flags (flags *toggle*, flags *mask* = flags(0))

Set flags.

The new value of flags is: (flags & mask) ^ toggle

To just set the flags, pass the new flags in toggle and the default value for mask.

Parameters:

toggle Flags to XOR.

mask Flags to AND with first.

Returns:

The old flags setting.

6.34.4.8 void Xapian::TermGenerator::index_text (const Xapian::Utf8Iterator & *itor*, Xapian::termcount *weight* = 1, const std::string & *prefix* = std::string())

Index some text.

Parameters:

weight The wdf increment (default 1).

prefix The term prefix to use (default is no prefix).

6.34.4.9 void Xapian::TermGenerator::index_text (const std::string & *text*, Xapian::termcount *weight* = 1, const std::string & *prefix* = std::string()) [inline]

Index some text in a std::string.

Parameters:

weight The wdf increment (default 1).

prefix The term prefix to use (default is no prefix).

6.34.4.10 void Xapian::TermGenerator::index_text_without_positions (const Xapian::Utf8Iterator & *itor*, Xapian::termcount *weight* = 1, const std::string & *prefix* = std::string())

Index some text without positional information.

Just like index_text, but no positional information is generated. This means that the database will be significantly smaller, but that phrase searching and NEAR won't be supported.

6.34.4.11 void Xapian::TermGenerator::index_text_without_positions (const std::string & *text*, Xapian::termcount *weight* = 1, const std::string & *prefix* = std::string()) [inline]

Index some text in a std::string without positional information.

Just like index_text, but no positional information is generated. This means that the database will be significantly smaller, but that phrase searching and NEAR won't be supported.

6.34.4.12 void Xapian::TermGenerator::increase_termpos (Xapian::termcount *delta* = 100)

Increase the termpos used by index_text by *delta*.

This can be used to prevent phrase searches from spanning two unconnected blocks of text (e.g. the title and body text).

6.34.4.13 Xapian::termcount Xapian::TermGenerator::get_termpos () const

Get the current term position.

6.34.4.14 void Xapian::TermGenerator::set_termpos (Xapian::termcount *termpos*)

Set the current term position.

6.34.4.15 std::string Xapian::TermGenerator::get_description () const

Return a string describing this object.

The documentation for this class was generated from the following file:

- include/xapian/termgenerator.h

6.35 Xapian::TermIterator Class Reference

`#include <termiterator.h>`

Public Types

- typedef std::input_iterator_tag iterator_category

  Allow use as an STL iterator.

- typedef std::string **value_type**
- typedef Xapian::termcount_diff **difference_type**
- typedef std::string * **pointer**
- typedef std::string & **reference**

Public Member Functions

- **TermIterator** (Internal *internal_)
- TermIterator ()

  Default constructor - for declaring an uninitialised iterator.

- ~TermIterator ()

  Destructor.

- TermIterator (const TermIterator &other)
- void operator= (const TermIterator &other)
- std::string operator * () const

  Return the current term.

- TermIterator & **operator++** ()
- DerefStringWrapper_ **operator++** (int)
- void skip_to (const std::string &tname)
- Xapian::termcount get_wdf () const
- Xapian::doccount get_termfreq () const
- Xapian::termcount positionlist_count () const
- PositionIterator positionlist_begin () const
- PositionIterator positionlist_end () const
- std::string get_description () const

  Return a string describing this object.

Public Attributes

- Xapian::Internal::RefCntPtr< Internal > **internal**

6.35.1 Detailed Description

An iterator pointing to items in a list of terms.

6.35.2 Member Typedef Documentation

6.35.2.1 typedef std::input_iterator_tag Xapian::TermIterator::iterator_category

Allow use as an STL iterator.

6.35.3 Constructor & Destructor Documentation

6.35.3.1 Xapian::TermIterator::TermIterator ()

Default constructor - for declaring an uninitialised iterator.

6.35.3.2 Xapian::TermIterator::~TermIterator ()

Destructor.

6.35.3.3 Xapian::TermIterator::TermIterator (const TermIterator & *other*)

Copying is allowed. The internals are reference counted, so copying is also cheap.

6.35.4 Member Function Documentation

6.35.4.1 void Xapian::TermIterator::operator= (const TermIterator & *other*)

Assignment is allowed. The internals are reference counted, so assignment is also cheap.

6.35.4.2 std::string Xapian::TermIterator::operator * () const

Return the current term.

6.35.4.3 void Xapian::TermIterator::skip_to (const std::string & *tname*)

Skip the iterator to term tname, or the first term after tname if tname isn't in the list of terms being iterated.

6.35.4.4 Xapian::termcount Xapian::TermIterator::get_wdf () const

Return the wdf of the current term (if meaningful).

The wdf (within document frequency) is the number of occurences of a term in a particular document.

6.35.4.5 Xapian::doccount Xapian::TermIterator::get_termfreq () const

Return the term frequency of the current term (if meaningful).

The term frequency is the number of documents which a term indexes.

6.35.4.6 Xapian::termcount Xapian::TermIterator::positionlist_count () const

Return length of positionlist for current term.

6.35.4.7 PositionIterator Xapian::TermIterator::positionlist_begin () const

Return PositionIterator pointing to start of positionlist for current term.

6.35.4.8 PositionIterator Xapian::TermIterator::positionlist_end () const [inline]

Return PositionIterator pointing to end of positionlist for current term.

6.35.4.9 std::string Xapian::TermIterator::get_description () const

Return a string describing this object.

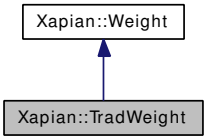
The documentation for this class was generated from the following file:

- include/xapian/termiterator.h

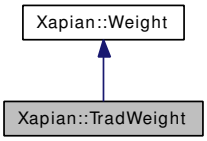
6.36 Xapian::TradWeight Class Reference

`#include <weight.h>`

Inheritance diagram for Xapian::TradWeight:



Collaboration diagram for Xapian::TradWeight:



Public Member Functions

- TradWeight (double k=1.0)
- std::string name () const
- std::string serialise () const
- TradWeight * unserialise (const std::string &s) const
- Xapian::weight get_sumpart (Xapian::termcount wdf, Xapian::termcount doclen) const
- Xapian::weight get_maxpart () const
- Xapian::weight get_sumextra (Xapian::termcount doclen) const
- Xapian::weight get_maxextra () const

6.36.1 Detailed Description

Xapian::Weight subclass implementing the traditional probabilistic formula.

This class implements the "traditional" Probabilistic Weighting scheme, as described by the early papers on Probabilistic Retrieval. BM25 generally gives better results.

TradWeight(k) is equivalent to BM25Weight(k, 0, 0, 1, 0), except that the latter returns weights (k+1) times larger.

6.36.2 Constructor & Destructor Documentation

6.36.2.1 Xapian::TradWeight::TradWeight (double *k* = 1.0) [inline, explicit]

Construct a TradWeight.

Parameters:

k A non-negative parameter controlling how influential within-document-frequency (wdf) and document length are. k=0 means that wdf and document length don't affect the weights. The larger k1 is, the more they do. (default 1)

6.36.3 Member Function Documentation

6.36.3.1 std::string Xapian::TradWeight::name () const [virtual]

Return the name of this weighting scheme.

This name is used by the remote backend. It is passed with the serialised parameters to the remote server so that it knows which class to create.

Return the full namespace-qualified name of your class here - if your class is called FooWeight, return "FooWeight" from this method (Xapian::BM25Weight returns "Xapian::BM25Weight" here).

If you don't want to support the remote backend in your weighting scheme, you can just implement this to throw Xapian::UnimplementedError.

Implements Xapian::Weight.

6.36.3.2 std::string Xapian::TradWeight::serialise () const [virtual]

Return this object's parameters serialised as a single string.

If you don't want to support the remote backend in your weighting scheme, you can just implement this to throw Xapian::UnimplementedError.

Implements Xapian::Weight.

6.36.3.3 TradWeight* Xapian::TradWeight::unserialise (const std::string & *s*) const [virtual]

Unserialise parameters.

This method unserialises parameters serialised by the *serialise()* method and allocates and returns a new object initialised with them.

If you don't want to support the remote backend in your weighting scheme, you can just implement this to throw Xapian::UnimplementedError.

Implements Xapian::Weight.

6.36.3.4 Xapian::weight Xapian::TradWeight::get_sumpart (Xapian::termcount *wdf*, Xapian::termcount *doclen*) const [virtual]

Calculate the weight contribution for this object's term to a document.

The parameters give information about the document which may be used in the calculations:

Parameters:

wdf The within document frequency of the term in the document.

doclen The document's length (unnormalised).

Implements Xapian::Weight.

6.36.3.5 Xapian::weight Xapian::TradWeight::get_maxpart () const [virtual]

Return an upper bound on what get_sumpart() can return for any document.

This information is used by the matcher to perform various optimisations, so strive to make the bound as tight as possible.

Implements Xapian::Weight.

6.36.3.6 Xapian::weight Xapian::TradWeight::get_sumextra (Xapian::termcount *doclen*) const [virtual]

Calculate the term-independent weight component for a document.

The parameter gives information about the document which may be used in the calculations:

Parameters:

doclen The document's length (unnormalised).

Implements Xapian::Weight.

6.36.3.7 Xapian::weight Xapian::TradWeight::get_maxextra () const [virtual]

Return an upper bound on what get_sumextra() can return for any document.

This information is used by the matcher to perform various optimisations, so strive to make the bound as tight as possible.

Implements Xapian::Weight.

The documentation for this class was generated from the following file:

- include/xapian/weight.h

6.37 Xapian::Utf8Iterator Class Reference

`#include <unicode.h>`

Public Types

- typedef std::input_iterator_tag iterator_category
  We implement the semantics of an STL input_iterator.
- typedef unsigned **value_type**
- typedef size_t **difference_type**
- typedef const unsigned * **pointer**
- typedef const unsigned & **reference**

Public Member Functions

- const char * raw () const
- size_t left () const
- void assign (const char *p_, size_t len)
- void assign (const std::string &s)
- Utf8Iterator (const char *p_)
- Utf8Iterator (const char *p_, size_t len)
- Utf8Iterator (const std::string &s)
- Utf8Iterator ()
- unsigned operator * () const
- Utf8Iterator operator++ (int)
- Utf8Iterator & operator++ ()
- bool operator== (const Utf8Iterator &other) const
- bool operator!= (const Utf8Iterator &other) const

6.37.1 Detailed Description

An iterator which returns Unicode character values from a UTF-8 encoded string.

6.37.2 Member Typedef Documentation

6.37.2.1 typedef std::input_iterator_tag Xapian::Utf8Iterator::iterator_category

We implement the semantics of an STL input_iterator.

6.37.3 Constructor & Destructor Documentation

6.37.3.1 Xapian::Utf8Iterator::Utf8Iterator (const char * *p_*) [explicit]

Create an iterator given a pointer to a null terminated string.

The iterator will return characters from the start of the string when next called. The string is not copied into the iterator, so it must remain valid while the iteration is in progress.

Parameters:

p A pointer to the start of the null terminated string to read.

6.37.3.2 Xapian::Utf8Iterator::Utf8Iterator (const char * *p_*, size_t *len*) [inline]

Create an iterator given a pointer and a length.

The iterator will return characters from the start of the string when next called. The string is not copied into the iterator, so it must remain valid while the iteration is in progress.

Parameters:

p A pointer to the start of the string to read.

len The length of the string to read.

6.37.3.3 Xapian::Utf8Iterator::Utf8Iterator (const std::string & *s*) [inline]

Create an iterator given a string.

The iterator will return characters from the start of the string when next called. The string is not copied into the iterator, so it must remain valid while the iteration is in progress.

Parameters:

s The string to read. Must not be modified while the iteration is in progress.

6.37.3.4 Xapian::Utf8Iterator::Utf8Iterator () [inline]

Create an iterator which is at the end of its iteration.

This can be compared to another iterator to check if the other iterator has reached its end.

6.37.4 Member Function Documentation

6.37.4.1 const char* Xapian::Utf8Iterator::raw () const [inline]

Return the raw const char * pointer for the current position.

6.37.4.2 size_t Xapian::Utf8Iterator::left () const [inline]

Return the number of bytes left in the iterator's buffer.

6.37.4.3 void Xapian::Utf8Iterator::assign (const char * *p_*, size_t *len*) [inline]

Assign a new string to the iterator.

The iterator will forget the string it was iterating through, and return characters from the start of the new string when next called. The string is not copied into the iterator, so it must remain valid while the iteration is in progress.

Parameters:

p A pointer to the start of the string to read.

len The length of the string to read.

6.37.4.4 void Xapian::Utf8Iterator::assign (const std::string & *s*) [inline]

Assign a new string to the iterator.

The iterator will forget the string it was iterating through, and return characters from the start of the new string when next called. The string is not copied into the iterator, so it must remain valid while the iteration is in progress.

Parameters:

s The string to read. Must not be modified while the iteration is in progress.

6.37.4.5 unsigned Xapian::Utf8Iterator::operator * () const

Get the current Unicode character value pointed to by the iterator.

Returns unsigned(-1) if the iterator has reached the end of its buffer.

6.37.4.6 Utf8Iterator Xapian::Utf8Iterator::operator++ (int) [inline]

Move forward to the next Unicode character.

Returns:

An iterator pointing to the position before the move.

6.37.4.7 Utf8Iterator& Xapian::Utf8Iterator::operator++ () [inline]

Move forward to the next Unicode character.

Returns:

A reference to this object.

6.37.4.8 bool Xapian::Utf8Iterator::operator== (const Utf8Iterator & *other*) const [inline]

Test two Utf8Iterators for equality.

Returns:

true iff the iterators point to the same position.

6.37.4.9 bool Xapian::Utf8Iterator::operator!= (const Utf8Iterator & *other*) const [inline]

Test two Utf8Iterators for inequality.

Returns:

true iff the iterators do not point to the same position.

The documentation for this class was generated from the following file:

- include/xapian/unicode.h

6.38 Xapian::ValueIterator Class Reference

Class for iterating over document values.

`#include <valueiterator.h>`

Public Member Functions

- ValueIterator (const ValueIterator &o)

  Copy constructor.

- **ValueIterator** (const ValueIteratorEnd_ &)
- ValueIterator & operator= (const ValueIterator &o)

  Assignment.

- ValueIterator & **operator=** (const ValueIteratorEnd_ &)
- ValueIterator ()
- ~ValueIterator ()

  Destructor.

- std::string operator * () const

  Return the value at the current position.

- ValueIterator & operator++ ()

  Advance the iterator to the next position.

- DerefStringWrapper_ operator++ (int)

  Advance the iterator to the next position (postfix version).

- Xapian::docid get_docid () const
- Xapian::valueno get_valueno () const
- void skip_to (Xapian::docid docid_or_slot)
- bool check (Xapian::docid docid)
- std::string get_description () const

  Return a string describing this object.

6.38.1 Detailed Description

Class for iterating over document values.

6.38.2 Constructor & Destructor Documentation

6.38.2.1 Xapian::ValueIterator::ValueIterator (const ValueIterator & *o*)

Copy constructor.

6.38.2.2 Xapian::ValueIterator::ValueIterator ()

Default constructor.

Creates an uninitialised iterator, which can't be used before being assigned to, but is sometimes syntactically convenient.

6.38.2.3 Xapian::ValueIterator::~ValueIterator ()

Destructor.

6.38.3 Member Function Documentation

6.38.3.1 ValueIterator& Xapian::ValueIterator::operator= (const ValueIterator & *o*)

Assignment.

6.38.3.2 std::string Xapian::ValueIterator::operator * () const

Return the value at the current position.

6.38.3.3 ValueIterator& Xapian::ValueIterator::operator++ ()

Advance the iterator to the next position.

6.38.3.4 DerefStringWrapper_ Xapian::ValueIterator::operator++ (int) [inline]

Advance the iterator to the next position (postfix version).

6.38.3.5 Xapian::docid Xapian::ValueIterator::get_docid () const

Return the docid at the current position.

If we're iterating over values of a document, this method will throw Xapian::InvalidOperationError.

6.38.3.6 Xapian::valueno Xapian::ValueIterator::get_valueno () const

Return the value slot number for the current position.

If the iterator is over all values in a slot, this returns that slot's number. If the iterator is over the values in a particular document, it returns the number of each slot in turn.

6.38.3.7 void Xapian::ValueIterator::skip_to (Xapian::docid *docid_or_slot*)

Advance the iterator to document id or value slot *docid_or_slot*.

If this iterator is over values in a document, then this method advances the iterator to value slot *docid_or_slot*, or the first slot after it if there is no value in slot *slot*.

If this iterator is over values in a particular slot, then this method advances the iterator to document id *docid_or_slot*, or the first document id after it if there is no value in the slot we're iterating over for document *docid_or_slot*.

Note: The "two-faced" nature of this method is due to how C++ overloading works. Xapian::docid and Xapian::valueno are both typedefs for the same unsigned integer type, so overloading can't distinguish them.

6.38.3.8 bool Xapian::ValueIterator::check (Xapian::docid *docid*)

Check if the specified docid occurs.

The caller is required to ensure that the specified document id *did* actually exists in the database.

This method acts like skip_to() if that can be done at little extra cost, in which case it then returns true. This is how chert behaves because it stores values in streams which allow for an efficient implementation of skip_to().

Otherwise it simply checks if a particular docid is present. If it is, it returns true. If it isn't, it returns false, and leaves the position unspecified (and hence the result of calling methods which depends on the current position, such as get_docid(), are also unspecified). In this state, next() will advance to the first matching position after document *did*, and skip_to() will act as it would if the position was the first matching position after document *did*.

Currently the inmemory, flint, and remote backends behave in the latter way because they don't support streamed values and so skip_to() must check each document it skips over which is significantly slower.

6.38.3.9 std::string Xapian::ValueIterator::get_description () const

Return a string describing this object.

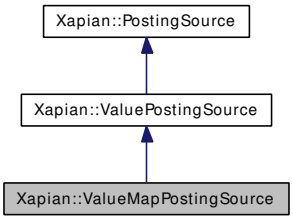
The documentation for this class was generated from the following file:

- include/xapian/valueiterator.h

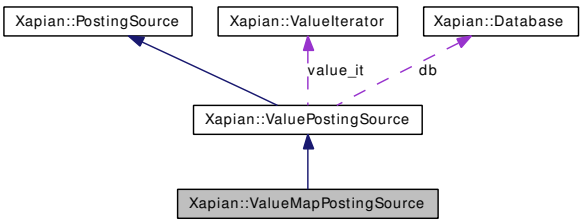
6.39 Xapian::ValueMapPostingSource Class Reference

`#include <postingsource.h>`

Inheritance diagram for Xapian::ValueMapPostingSource:



Collaboration diagram for Xapian::ValueMapPostingSource:



Public Member Functions

- ValueMapPostingSource (Xapian::valueno slot_)
- void add_mapping (const std::string &key, double weight)
- void clear_mappings ()
- void set_default_weight (double wt)
- Xapian::weight get_weight () const
- ValueMapPostingSource * clone () const
- std::string name () const
- std::string serialise () const
- ValueMapPostingSource * unserialise (const std::string &s) const
- void init (const Database &db_)
- std::string get_description () const

6.39.1 Detailed Description

A posting source which looks up weights in a map using values as the key.

This allows will return entries for all documents in the given database which have a value in the slot specified. The values will be mapped to the corresponding weight in the weight map. If there is no mapping for a particular value, the default weight will be returned (which itself defaults to 0.0).

6.39.2 Constructor & Destructor Documentation

6.39.2.1 Xapian::ValueMapPostingSource::ValueMapPostingSource (Xapian::valueno *slot_*)

Construct a ValueWeightPostingSource.

Parameters:

slot_ The value slot to read values from.

6.39.3 Member Function Documentation

6.39.3.1 void Xapian::ValueMapPostingSource::add_mapping (const std::string & *key*, double *weight*)

Add a mapping.

Parameters:

key The key looked up from the value slot.

weight The weight to give this key.

6.39.3.2 void Xapian::ValueMapPostingSource::clear_mappings ()

Clear all mappings.

6.39.3.3 void Xapian::ValueMapPostingSource::set_default_weight (double *wt*)

Set a default weight for document values not in the map.

6.39.3.4 Xapian::weight Xapian::ValueMapPostingSource::get_weight () const [virtual]

Return the weight contribution for the current document.

This default implementation always returns 0, for convenience when implementing "weight-less" PostingSource subclasses.

This method may assume that it will only be called when there is a "current document". In detail: Xapian will always call init() on a PostingSource before calling this for the first time. It will also only call this if the PostingSource reports that it is pointing to a valid document (ie, it will not call it before calling at least one of next(), skip_to() or check(), and will ensure that the PostingSource is not at the end by calling at_end()).

Reimplemented from Xapian::PostingSource.

6.39.3.5 ValueMapPostingSource* Xapian::ValueMapPostingSource::clone () const [virtual]

Clone the posting source.

The clone should inherit the configuration of the parent, but need not inherit the state. ie, the clone does not need to be in the same iteration position as the original: the matcher will always call init() on the clone before attempting to move the iterator, or read the information about the current position of the iterator.

This may return NULL to indicate that cloning is not supported. In this case, the PostingSource may only be used with a single-database search.

The default implementation returns NULL.

Note that the returned object will be deallocated by Xapian after use with "delete". It must therefore have been allocated with "new".

Reimplemented from Xapian::PostingSource.

6.39.3.6 std::string Xapian::ValueMapPostingSource::name () const [virtual]

Name of the posting source class.

This is used when serialising and unserialising posting sources; for example, for performing remote searches.

If the subclass is in a C++ namespace, the namespace should be included in the name, using "::" as a separator. For example, for a PostingSource subclass called "FooPostingSource" in the "Xapian" namespace the result of this call should be "Xapian::FooPostingSource".

This should only be implemented if serialise() and unserialise() are also implemented. The default implementation returns an empty string.

If this returns an empty string, Xapian will assume that serialise() and unserialise() are not implemented.

Reimplemented from Xapian::PostingSource.

6.39.3.7 std::string Xapian::ValueMapPostingSource::serialise () const [virtual]

Serialise object parameters into a string.

The serialised parameters should represent the configuration of the posting source, but need not (indeed, should not) represent the current iteration state.

Reimplemented from Xapian::PostingSource.

6.39.3.8 ValueMapPostingSource* Xapian::ValueMapPostingSource::unserialise (const std::string & *s*) const [virtual]

Create object given string serialisation returned by serialise().

Parameters:

s A serialised instance of this PostingSource subclass.

Reimplemented from Xapian::PostingSource.

6.39.3.9 void Xapian::ValueMapPostingSource::init (const Database & *db*) [virtual]

Set this PostingSource to the start of the list of postings.

This is called automatically by the matcher prior to each query being processed.

If a PostingSource is used for multiple searches, *init()* will therefore be called multiple times, and must handle this by using the database passed in the most recent call.

Parameters:

db The database which the PostingSource should iterate through.

Note: the database supplied to this method must not be modified: in particular, the reopen() method should not be called on it.

Note: in the case of a multi-database search, a separate PostingSource will be used for each database (the separate PostingSources will be obtained using *clone()*), and each PostingSource will be passed one of the sub-databases as the *db* parameter here. The *db* parameter will therefore always refer to a single database. All docids passed to, or returned from, the PostingSource refer to docids in that single database, rather than in the multi-database.

Reimplemented from Xapian::ValuePostingSource.

6.39.3.10 std::string Xapian::ValueMapPostingSource::get_description () const [virtual]

Return a string describing this object.

This default implementation returns a generic answer. This default it provided to avoid forcing those deriving their own PostingSource subclass from having to implement this (they may not care what get_description() gives for their subclass).

Reimplemented from Xapian::PostingSource.

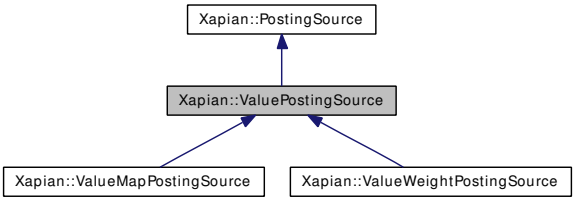
The documentation for this class was generated from the following file:

- include/xapian/postingsource.h

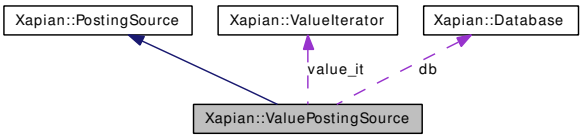
6.40 Xapian::ValuePostingSource Class Reference

`#include <postingsource.h>`

Inheritance diagram for Xapian::ValuePostingSource:



Collaboration diagram for Xapian::ValuePostingSource:



Public Member Functions

- ValuePostingSource (Xapian::valueno slot_)
- Xapian::doccount get_termfreq_min () const
- Xapian::doccount get_termfreq_est () const
- Xapian::doccount get_termfreq_max () const
- void next (Xapian::weight min_wt)
- void skip_to (Xapian::docid min_docid, Xapian::weight min_wt)
- bool check (Xapian::docid min_docid, Xapian::weight min_wt)
- bool at_end () const
- Xapian::docid get_docid () const
- void init (const Database &db_)

Protected Attributes

- Xapian::Database db

  The database we're reading values from.

- Xapian::valueno slot

  The slot we're reading values from.

- Xapian::ValueIterator value_it

  Value stream iterator.

- bool started

  Flag indicating if we've started (true if we have).

- Xapian::doccount termfreq_min
- Xapian::doccount termfreq_est
- Xapian::doccount termfreq_max

6.40.1 Detailed Description

A posting source which generates weights from a value slot.

This is a base class for classes which generate weights using values stored in the specified slot. For example, ValueWeightPostingSource uses sortable_unserialise to convert values directly to weights.

The upper bound on the weight returned is set to DBL_MAX. Subclasses should call set_maxweight() in their init() methods after calling ValuePostingSource::init() if they know a tighter bound on the weight.

6.40.2 Constructor & Destructor Documentation

6.40.2.1 Xapian::ValuePostingSource::ValuePostingSource (Xapian::valueno *slot_*)

Construct a ValuePostingSource.

Parameters:

slot_ The value slot to read values from.

6.40.3 Member Function Documentation

6.40.3.1 Xapian::doccount Xapian::ValuePostingSource::get_termfreq_min () const [virtual]

A lower bound on the number of documents this object can return.

Xapian will always call init() on a PostingSource before calling this for the first time.

Implements Xapian::PostingSource.

6.40.3.2 Xapian::doccount Xapian::ValuePostingSource::get_termfreq_est () const [virtual]

An estimate of the number of documents this object can return.

It must always be true that:

get_termfreq_min() <= get_termfreq_est() <= get_termfreq_max()

Xapian will always call init() on a PostingSource before calling this for the first time.

Implements Xapian::PostingSource.

6.40.3.3 Xapian::doccount Xapian::ValuePostingSource::get_termfreq_max () const [virtual]

An upper bound on the number of documents this object can return.

Xapian will always call init() on a PostingSource before calling this for the first time.

Implements Xapian::PostingSource.

6.40.3.4 void Xapian::ValuePostingSource::next (Xapian::weight *min_wt*) [virtual]

Advance the current position to the next matching document.

The PostingSource starts before the first entry in the list, so next() must be called before any methods which need the context of the current position.

Xapian will always call init() on a PostingSource before calling this for the first time.

Parameters:

min_wt The minimum weight contribution that is needed (this is just a hint which subclasses may ignore).

Implements Xapian::PostingSource.

6.40.3.5 void Xapian::ValuePostingSource::skip_to (Xapian::docid *did*, Xapian::weight *min_wt*) [virtual]

Skip forward to the specified docid.

If the specified docid isn't in the list, position ourselves on the first document after it (or at_end() if no greater docids are present).

If the current position is already the specified docid, this method will leave the position unmodified.

If the specified docid is earlier than the current position, the behaviour is unspecified. A sensible behaviour would be to leave the current position unmodified, but it is also reasonable to move to the specified docid.

The default implementation calls next() repeatedly, which works but skip_to() can often be implemented much more efficiently.

Xapian will always call init() on a PostingSource before calling this for the first time.

Note: in the case of a multi-database search, the docid specified is the docid in the single subdatabase relevant to this posting source. See the *init()* method for details.

Parameters:

min_wt The minimum weight contribution that is needed (this is just a hint which subclasses may ignore).

Reimplemented from Xapian::PostingSource.

6.40.3.6 bool Xapian::ValuePostingSource::check (Xapian::docid *did*, Xapian::weight *min_wt*) [virtual]

Check if the specified docid occurs.

The caller is required to ensure that the specified document id *did* actually exists in the database. If it does, it must move to that document id, and return true. If it does not, it may either:

- return true, having moved to a definite position (including "at_end"), which must be the same position as skip_to() would have moved to.

or

- return false, having moved to an "indeterminate" position, such that a subsequent call to next() or skip_to() will move to the next matching position after *did*.

Generally, this method should act like skip_to() and return true if that can be done at little extra cost.

Otherwise it should simply check if a particular docid is present, returning true if it is, and false if it isn't.

The default implementation calls skip_to() and always returns true.

Xapian will always call init() on a PostingSource before calling this for the first time.

Note: in the case of a multi-database search, the docid specified is the docid in the single subdatabase relevant to this posting source. See the *init()* method for details.

Reimplemented from Xapian::PostingSource.

6.40.3.7 bool Xapian::ValuePostingSource::at_end () const [virtual]

Return true if the current position is past the last entry in this list.

At least one of *next()*, *skip_to()* or *check()* will be called before this method is first called.

Implements Xapian::PostingSource.

6.40.3.8 Xapian::docid Xapian::ValuePostingSource::get_docid () const [virtual]

Return the current docid.

This method may assume that it will only be called when there is a "current document". See *get_weight()* for details.

Note: in the case of a multi-database search, the returned docid should be in the single subdatabase relevant to this posting source. See the *init()* method for details.

Implements Xapian::PostingSource.

6.40.3.9 void Xapian::ValuePostingSource::init (const Database & *db*) [virtual]

Set this PostingSource to the start of the list of postings.

This is called automatically by the matcher prior to each query being processed.

If a PostingSource is used for multiple searches, *init()* will therefore be called multiple times, and must handle this by using the database passed in the most recent call.

Parameters:

db The database which the PostingSource should iterate through.

Note: the database supplied to this method must not be modified: in particular, the reopen() method should not be called on it.

Note: in the case of a multi-database search, a separate PostingSource will be used for each database (the separate PostingSources will be obtained using *clone()*), and each PostingSource will be passed one of the sub-databases as the *db* parameter here. The *db* parameter will therefore always refer to a single database. All docids passed to, or returned from, the PostingSource refer to docids in that single database, rather than in the multi-database.

Implements Xapian::PostingSource.

Reimplemented in Xapian::ValueWeightPostingSource, and Xapian::ValueMapPostingSource.

6.40.4 Member Data Documentation

6.40.4.1 Xapian::Database Xapian::ValuePostingSource::db [protected]

The database we're reading values from.

6.40.4.2 Xapian::valueno Xapian::ValuePostingSource::slot [protected]

The slot we're reading values from.

6.40.4.3 Xapian::ValueIterator Xapian::ValuePostingSource::value_it [protected]

Value stream iterator.

6.40.4.4 bool Xapian::ValuePostingSource::started [protected]

Flag indicating if we've started (true if we have).

6.40.4.5 Xapian::doccount Xapian::ValuePostingSource::termfreq_min [protected]

A lower bound on the term frequency.

Subclasses should set this if they are overriding the next(), skip_to() or check() methods to return fewer documents.

6.40.4.6 Xapian::doccount Xapian::ValuePostingSource::termfreq_est [protected]

An estimate of the term frequency.

Subclasses should set this if they are overriding the next(), skip_to() or check() methods.

6.40.4.7 Xapian::doccount Xapian::ValuePostingSource::termfreq_max [protected]

An upper bound on the term frequency.

Subclasses should set this if they are overriding the next(), skip_to() or check() methods.

The documentation for this class was generated from the following file:

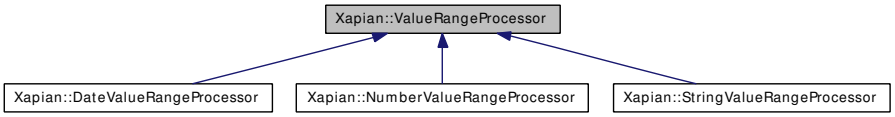
- include/xapian/postingsource.h

6.41 Xapian::ValueRangeProcessor Struct Reference

Base class for value range processors.

`#include <queryparser.h>`

Inheritance diagram for Xapian::ValueRangeProcessor:



Public Member Functions

- virtual ~ValueRangeProcessor ()

  Destructor.

- virtual Xapian::valueno operator() (std::string &begin, std::string &end)=0

6.41.1 Detailed Description

Base class for value range processors.

6.41.2 Constructor & Destructor Documentation

6.41.2.1 virtual Xapian::ValueRangeProcessor::~ValueRangeProcessor () [virtual]

Destructor.

6.41.3 Member Function Documentation

6.41.3.1 virtual Xapian::valueno Xapian::ValueRangeProcessor::operator() (std::string & *begin*, std::string & *end*) [pure virtual]

See if <begin>..<end> is a valid value range.

If this ValueRangeProcessor recognises <begin>..<end> it returns the value number of range filter on. Otherwise it returns Xapian::BAD_VALUENO.

Implemented in Xapian::StringValueRangeProcessor, Xapian::DateValueRangeProcessor, and Xapian::NumberValueRangeProcessor.

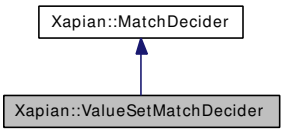
The documentation for this struct was generated from the following file:

- include/xapian/queryparser.h

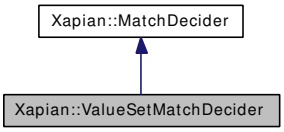
6.42 Xapian::ValueSetMatchDecider Class Reference

`#include <valuesetmatchdecider.h>`

Inheritance diagram for Xapian::ValueSetMatchDecider:



Collaboration diagram for Xapian::ValueSetMatchDecider:



Public Member Functions

- ValueSetMatchDecider (Xapian::valueno valuenum, bool inclusive)
- void add_value (const std::string &value)
- void remove_value (const std::string &value)
- bool operator() (const Xapian::Document &doc) const

6.42.1 Detailed Description

MatchDecider filtering results based on whether document values are in a user-defined set.

6.42.2 Constructor & Destructor Documentation

6.42.2.1 Xapian::ValueSetMatchDecider::ValueSetMatchDecider (Xapian::valueno *valuenum*, bool *inclusive*)

Construct a ValueSetMatchDecider.

Parameters:

valuenum The value slot number to look in.

inclusive If true, match decider accepts documents which have a value in the specified slot which is a member of the test set; if false, match decider accepts documents which do not have a value in the specified slot.

6.42.3 Member Function Documentation

6.42.3.1 void Xapian::ValueSetMatchDecider::add_value (const std::string & *value*) [inline]

Add a value to the test set.

Parameters:

value The value to add to the test set.

6.42.3.2 void Xapian::ValueSetMatchDecider::remove_value (const std::string & *value*) [inline]

Remove a value from the test set.

Parameters:

value The value to remove from the test set.

6.42.3.3 bool Xapian::ValueSetMatchDecider::operator() (const Xapian::Document & *doc*) const [virtual]

Decide whether we want this document to be in the MSet.

Return true if the document is acceptable, or false if the document should be excluded from the MSet.

Implements Xapian::MatchDecider.

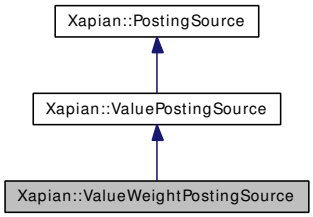
The documentation for this class was generated from the following file:

- include/xapian/valuesetmatchdecider.h

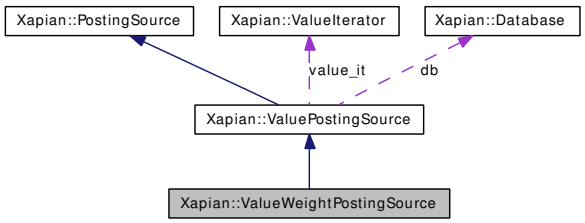
6.43 Xapian::ValueWeightPostingSource Class Reference

`#include <postingsource.h>`

Inheritance diagram for Xapian::ValueWeightPostingSource:



Collaboration diagram for Xapian::ValueWeightPostingSource:



Public Member Functions

- ValueWeightPostingSource (Xapian::valueno slot_)
- Xapian::weight get_weight () const
- ValueWeightPostingSource * clone () const
- std::string name () const
- std::string serialise () const
- ValueWeightPostingSource * unserialise (const std::string &s) const
- void init (const Database &db_)
- std::string get_description () const

6.43.1 Detailed Description

A posting source which reads weights from a value slot.

This returns entries for all documents in the given database which have a non empty values in the specified slot. It returns a weight calculated by applying sortable_unserialise to the value stored in the slot (so the values stored should probably have been calculated by applying sortable_serialise to a floating point number at index time).

The upper bound on the weight returned is set using the upper bound on the values in the specified slot, or DBL_MAX if value bounds aren't supported by the current backend.

For efficiency, this posting source doesn't check that the stored values are valid in any way, so it will never raise an exception due to invalid stored values. In particular, it doesn't ensure that the unserialised values are positive, which is a requirement for weights. The behaviour if the slot contains values which unserialise to negative values is undefined.

6.43.2 Constructor & Destructor Documentation

6.43.2.1 Xapian::ValueWeightPostingSource::ValueWeightPostingSource (Xapian::valueno *slot_*)

Construct a ValueWeightPostingSource.

Parameters:

slot_ The value slot to read values from.

6.43.3 Member Function Documentation

6.43.3.1 Xapian::weight Xapian::ValueWeightPostingSource::get_weight () const [virtual]

Return the weight contribution for the current document.

This default implementation always returns 0, for convenience when implementing "weight-less" PostingSource subclasses.

This method may assume that it will only be called when there is a "current document". In detail: Xapian will always call init() on a PostingSource before calling this for the first time. It will also only call this if the PostingSource reports that it is pointing to a valid document (ie, it will not call it before calling at least one of next(), skip_to() or check(), and will ensure that the PostingSource is not at the end by calling at_end()).

Reimplemented from Xapian::PostingSource.

6.43.3.2 ValueWeightPostingSource* Xapian::ValueWeightPostingSource::clone () const [virtual]

Clone the posting source.

The clone should inherit the configuration of the parent, but need not inherit the state. ie, the clone does not need to be in the same iteration position as the original: the matcher will always call init() on the clone before attempting to move the iterator, or read the information about the current position of the iterator.

This may return NULL to indicate that cloning is not supported. In this case, the PostingSource may only be used with a single-database search.

The default implementation returns NULL.

Note that the returned object will be deallocated by Xapian after use with "delete". It must therefore have been allocated with "new".

Reimplemented from Xapian::PostingSource.

6.43.3.3 std::string Xapian::ValueWeightPostingSource::name () const [virtual]

Name of the posting source class.

This is used when serialising and unserialising posting sources; for example, for performing remote searches.

If the subclass is in a C++ namespace, the namespace should be included in the name, using "::" as a separator. For example, for a PostingSource subclass called "FooPostingSource" in the "Xapian" namespace the result of this call should be "Xapian::FooPostingSource".

This should only be implemented if serialise() and unserialise() are also implemented. The default implementation returns an empty string.

If this returns an empty string, Xapian will assume that serialise() and unserialise() are not implemented.

Reimplemented from Xapian::PostingSource.

6.43.3.4 std::string Xapian::ValueWeightPostingSource::serialise () const [virtual]

Serialise object parameters into a string.

The serialised parameters should represent the configuration of the posting source, but need not (indeed, should not) represent the current iteration state.

Reimplemented from Xapian::PostingSource.

6.43.3.5 ValueWeightPostingSource* Xapian::ValueWeightPostingSource::unserialise (const std::string & *s*) const [virtual]

Create object given string serialisation returned by serialise().

Parameters:

s A serialised instance of this PostingSource subclass.

Reimplemented from Xapian::PostingSource.

6.43.3.6 void Xapian::ValueWeightPostingSource::init (const Database & *db*) [virtual]

Set this PostingSource to the start of the list of postings.

This is called automatically by the matcher prior to each query being processed.

If a PostingSource is used for multiple searches, *init()* will therefore be called multiple times, and must handle this by using the database passed in the most recent call.

Parameters:

db The database which the PostingSource should iterate through.

Note: the database supplied to this method must not be modified: in particular, the reopen() method should not be called on it.

Note: in the case of a multi-database search, a separate PostingSource will be used for each database (the separate PostingSources will be obtained using *clone()*), and each PostingSource will be passed one of the sub-databases as the *db* parameter here. The *db* parameter will therefore always refer to a single database. All docids passed to, or returned from, the PostingSource refer to docids in that single database, rather than in the multi-database.

Reimplemented from Xapian::ValuePostingSource.

6.43.3.7 std::string Xapian::ValueWeightPostingSource::get_description () const [virtual]

Return a string describing this object.

This default implementation returns a generic answer. This default it provided to avoid forcing those deriving their own PostingSource subclass from having to implement this (they may not care what get_description() gives for their subclass).

Reimplemented from Xapian::PostingSource.

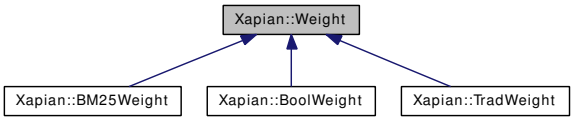
The documentation for this class was generated from the following file:

- include/xapian/postingsource.h

6.44 Xapian::Weight Class Reference

`#include <weight.h>`

Inheritance diagram for Xapian::Weight:



Public Member Functions

- virtual ~Weight ()
- virtual std::string name () const=0
- virtual std::string serialise () const=0
- virtual Weight * unserialise (const std::string &s) const=0
- virtual Xapian::weight get_sumpart (Xapian::termcount wdf, Xapian::termcount doclen) const =0
- virtual Xapian::weight get_maxpart () const=0
- virtual Xapian::weight get_sumextra (Xapian::termcount doclen) const =0
- virtual Xapian::weight get_maxextra () const=0

Protected Types

- enum stat_flags {

  COLLECTION_SIZE = 1, **RSET_SIZE** = 2, **AVERAGE_LENGTH** = 4, **TERMFREQ** = 8,

  RELTERMFREQ = 16, **QUERY_LENGTH** = 32, **WQF** = 64, **WDF** = 128,

  DOC_LENGTH = 256, **DOC_LENGTH_MIN** = 512, **DOC_LENGTH_MAX** = 1024, **WDF_MAX** = 2048 }

  Stats which the weighting scheme can use (see need_stat()).

Protected Member Functions

- void need_stat (stat_flags flag)
- virtual void init (double factor)=0
- Weight (const Weight &)

  Only allow subclasses to copy us.

- Weight ()

  Default constructor, needed by subclass constructors.

- Xapian::doccount get_collection_size () const
  The number of documents in the collection.
- Xapian::doccount get_rset_size () const
  The number of documents marked as relevant.
- Xapian::doclength get_average_length () const
  The average length of a document in the collection.
- Xapian::doccount get_termfreq () const
  The number of documents which this term indexes.
- Xapian::doccount get_reltermfreq () const
  The number of relevant documents which this term indexes.
- Xapian::termcount get_query_length () const
  The length of the query.
- Xapian::termcount get_wqf () const
  The within-query-frequency of this term.
- Xapian::termcount get_doclength_upper_bound () const
- Xapian::termcount get_doclength_lower_bound () const
- Xapian::termcount get_wdf_upper_bound () const

6.44.1 Detailed Description

Abstract base class for weighting schemes.

6.44.2 Member Enumeration Documentation

6.44.2.1 enum Xapian::Weight::stat_flags [protected]

Stats which the weighting scheme can use (see *need_stat()*).

6.44.3 Constructor & Destructor Documentation

6.44.3.1 virtual Xapian::Weight::~Weight () [virtual]

Virtual destructor, because we have virtual methods.

6.44.3.2 Xapian::Weight::Weight (const Weight &) [protected]

Only allow subclasses to copy us.

6.44.3.3 Xapian::Weight::Weight () [inline, protected]

Default constructor, needed by subclass constructors.

6.44.4 Member Function Documentation

6.44.4.1 void Xapian::Weight::need_stat (stat_flags *flag*) [inline, protected]

Tell Xapian that your subclass will want a particular statistic.

Some of the statistics can be costly to fetch or calculate, so Xapian needs to know which are actually going to be used. You should call need_stat() from your constructor for each such statistic.

Parameters:

flag The stat_flags value for a required statistic.

6.44.4.2 virtual void Xapian::Weight::init (double *factor*) [protected, pure virtual]

Allow the subclass to perform any initialisation it needs to.

Parameters:

factor Any scaling factor (e.g. from OP_SCALE_WEIGHT).

6.44.4.3 virtual std::string Xapian::Weight::name () const [pure virtual]

Return the name of this weighting scheme.

This name is used by the remote backend. It is passed with the serialised parameters to the remote server so that it knows which class to create.

Return the full namespace-qualified name of your class here - if your class is called FooWeight, return "FooWeight" from this method (Xapian::BM25Weight returns "Xapian::BM25Weight" here).

If you don't want to support the remote backend in your weighting scheme, you can just implement this to throw Xapian::UnimplementedError.

Implemented in Xapian::BoolWeight, Xapian::BM25Weight, and Xapian::TradWeight.

6.44.4.4 virtual std::string Xapian::Weight::serialise () const [pure virtual]

Return this object's parameters serialised as a single string.

If you don't want to support the remote backend in your weighting scheme, you can just implement this to throw Xapian::UnimplementedError.

Implemented in Xapian::BoolWeight, Xapian::BM25Weight, and Xapian::TradWeight.

6.44.4.5 virtual Weight* Xapian::Weight::unserialise (const std::string & *s*) const [pure virtual]

Unserialise parameters.

This method unserialises parameters serialised by the *serialise()* method and allocates and returns a new object initialised with them.

If you don't want to support the remote backend in your weighting scheme, you can just implement this to throw Xapian::UnimplementedError.

Implemented in Xapian::BoolWeight, Xapian::BM25Weight, and Xapian::TradWeight.

6.44.4.6 virtual Xapian::weight Xapian::Weight::get_sumpart (Xapian::termcount *wdf*, Xapian::termcount *doclen*) const [pure virtual]

Calculate the weight contribution for this object's term to a document.

The parameters give information about the document which may be used in the calculations:

Parameters:

wdf The within document frequency of the term in the document.

doclen The document's length (unnormalised).

Implemented in Xapian::BoolWeight, Xapian::BM25Weight, and Xapian::TradWeight.

6.44.4.7 virtual Xapian::weight Xapian::Weight::get_maxpart () const [pure virtual]

Return an upper bound on what get_sumpart() can return for any document.

This information is used by the matcher to perform various optimisations, so strive to make the bound as tight as possible.

Implemented in Xapian::BoolWeight, Xapian::BM25Weight, and Xapian::TradWeight.

6.44.4.8 virtual Xapian::weight Xapian::Weight::get_sumextra (Xapian::termcount *doclen*) const [pure virtual]

Calculate the term-independent weight component for a document.

The parameter gives information about the document which may be used in the calculations:

Parameters:

doclen The document's length (unnormalised).

Implemented in Xapian::BoolWeight, Xapian::BM25Weight, and Xapian::TradWeight.

6.44.4.9 virtual Xapian::weight Xapian::Weight::get_maxextra () const [pure virtual]

Return an upper bound on what get_sumextra() can return for any document.

This information is used by the matcher to perform various optimisations, so strive to make the bound as tight as possible.

Implemented in Xapian::BoolWeight, Xapian::BM25Weight, and Xapian::TradWeight.

6.44.4.10 Xapian::doccount Xapian::Weight::get_collection_size () const [inline, protected]

The number of documents in the collection.

6.44.4.11 Xapian::doccount Xapian::Weight::get_rset_size () const [inline, protected]

The number of documents marked as relevant.

6.44.4.12 Xapian::doclength Xapian::Weight::get_average_length () const [inline, protected]

The average length of a document in the collection.

6.44.4.13 Xapian::doccount Xapian::Weight::get_termfreq () const [inline, protected]

The number of documents which this term indexes.

6.44.4.14 Xapian::doccount Xapian::Weight::get_reltermfreq () const [inline, protected]

The number of relevant documents which this term indexes.

6.44.4.15 Xapian::termcount Xapian::Weight::get_query_length () const [inline, protected]

The length of the query.

6.44.4.16 Xapian::termcount Xapian::Weight::get_wqf () const [inline, protected]

The within-query-frequency of this term.

6.44.4.17 Xapian::termcount Xapian::Weight::get_doclength_upper_bound () const [inline, protected]

An lower bound on the maximum length of any document in the database.

This should only be used by get_maxpart() and get_maxextra().

6.44.4.18 Xapian::termcount Xapian::Weight::get_doclength_lower_bound () const [inline, protected]

An upper bound on the maximum length of any document in the database.

This should only be used by get_maxpart() and get_maxextra().

6.44.4.19 Xapian::termcount Xapian::Weight::get_wdf_upper_bound () const [inline, protected]

An upper bound on the wdf of this term.

This should only be used by get_maxpart() and get_maxextra().

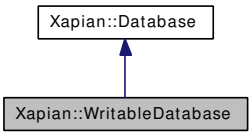
The documentation for this class was generated from the following file:

- include/xapian/weight.h

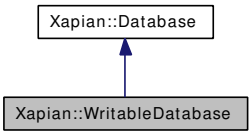
6.45 Xapian::WritableDatabase Class Reference

`#include <database.h>`

Inheritance diagram for Xapian::WritableDatabase:



Collaboration diagram for Xapian::WritableDatabase:



Public Member Functions

- virtual ~WritableDatabase ()
- WritableDatabase ()
- WritableDatabase (const std::string &path, int action)
- WritableDatabase (const WritableDatabase &other)
- void operator= (const WritableDatabase &other)
- void commit ()
- void flush ()
- void begin_transaction (bool flushed=true)
- void commit_transaction ()
- void cancel_transaction ()
- Xapian::docid add_document (const Xapian::Document &document)
- void delete_document (Xapian::docid did)
- void delete_document (const std::string &unique_term)
- void replace_document (Xapian::docid did, const Xapian::Document &document)
- Xapian::docid replace_document (const std::string &unique_term, const Xapian::Document &document)
- void add_spelling (const std::string &word, Xapian::termcount freqinc=1) const
- void remove_spelling (const std::string &word, Xapian::termcount freqdec=1) const
- void add_synonym (const std::string &term, const std::string &synonym) const
- void remove_synonym (const std::string &term, const std::string &synonym) const
- void clear_synonyms (const std::string &term) const

- void set_metadata (const std::string &key, const std::string &value)
- std::string get_description () const

  Return a string describing this object.

6.45.1 Detailed Description

This class provides read/write access to a database.

6.45.2 Constructor & Destructor Documentation

6.45.2.1 virtual Xapian::WritableDatabase::~WritableDatabase () [virtual]

Destroy this handle on the database.

If there are no copies of this object remaining, the database will be closed. If there are any transactions in progress these will be aborted as if cancel_transaction had been called.

6.45.2.2 Xapian::WritableDatabase::WritableDatabase ()

Create an empty WritableDatabase.

6.45.2.3 Xapian::WritableDatabase::WritableDatabase (const std::string & *path*, int *action*)

Open a database for update, automatically determining the database backend to use.

If the database is to be created, Xapian will try to create the directory indicated by path if it doesn't already exist (but only the leaf directory, not recursively).

Parameters:

path directory that the database is stored in.

action one of:

- Xapian::DB_CREATE_OR_OPEN open for read/write; create if no db exists
- Xapian::DB_CREATE create new database; fail if db exists
- Xapian::DB_CREATE_OR_OVERWRITE overwrite existing db; create if none exists
- Xapian::DB_OPEN open for read/write; fail if no db exists

Exceptions:

Xapian::DatabaseCorruptError will be thrown if the database is in a corrupt state.

Xapian::DatabaseLockError will be thrown if a lock couldn't be acquired on the database.

6.45.2.4 Xapian::WritableDatabase::WritableDatabase (const WritableDatabase & *other*)

Copying is allowed. The internals are reference counted, so copying is cheap.

6.45.3 Member Function Documentation

6.45.3.1 void Xapian::WritableDatabase::operator= (const WritableDatabase & *other*)

Assignment is allowed. The internals are reference counted, so assignment is cheap.

Note that only an WritableDatabase may be assigned to an WritableDatabase: an attempt to assign a Database is caught at compile-time.

6.45.3.2 void Xapian::WritableDatabase::commit ()

Commit any pending modifications made to the database.

For efficiency reasons, when performing multiple updates to a database it is best (indeed, almost essential) to make as many modifications as memory will permit in a single pass through the database. To ensure this, Xapian batches up modifications.

This method may be called at any time to commit any pending modifications to the database.

If any of the modifications fail, an exception will be thrown and the database will be left in a state in which each separate addition, replacement or deletion operation has either been fully performed or not performed at all: it is then up to the application to work out which operations need to be repeated.

It's not valid to call commit() within a transaction.

Beware of calling commit() too frequently: this will make indexing take much longer.

Note that commit() need not be called explicitly: it will be called automatically when the database is closed, or when a sufficient number of modifications have been made. By default, this is every 10000 documents added, deleted, or modified. This value is rather conservative, and if you have a machine with plenty of memory, you can improve indexing throughput dramatically by setting XAPIAN_FLUSH_THRESHOLD in the environment to a larger value.

Exceptions:

Xapian::DatabaseError will be thrown if a problem occurs while modifying the database.

Xapian::DatabaseCorruptError will be thrown if the database is in a corrupt state.

6.45.3.3 void Xapian::WritableDatabase::flush () [inline]

Pre-1.1.0 name for commit().

Use commit() instead in new code. This alias may be deprecated in the future.

6.45.3.4 void Xapian::WritableDatabase::begin_transaction (bool *flushed* = true)

Begin a transaction.

In Xapian a transaction is a group of modifications to the database which are linked such that either all will be applied simultaneously or none will be applied at all. Even in the case of a power failure, this characteristic should be preserved (as long as the filesystem isn't corrupted, etc).

A transaction is started with begin_transaction() and can either be committed by calling commit_transaction() or aborted by calling cancel_transaction().

By default, a transaction implicitly calls commit() before and after so that the modifications stand and fall without affecting modifications before or after.

The downside of these calls to commit() is that small transactions can harm indexing performance in the same way that explicitly calling commit() frequently can.

If you're applying atomic groups of changes and only wish to ensure that each group is either applied or not applied, then you can prevent the automatic commit() before and after the transaction by starting the transaction with begin_transaction(false). However, if cancel_transaction is called (or if commit_transaction isn't called before the WritableDatabase object is destroyed) then any changes which were pending before the transaction began will also be discarded.

Transactions aren't currently supported by the InMemory backend.

Exceptions:

Xapian::UnimplementedError will be thrown if transactions are not available for this database type.

Xapian::InvalidOperationError will be thrown if this is called at an invalid time, such as when a transaction is already in progress.

6.45.3.5 void Xapian::WritableDatabase::commit_transaction ()

Complete the transaction currently in progress.

If this method completes successfully and this is a flushed transaction, all the database modifications made during the transaction will have been committed to the database.

If an error occurs, an exception will be thrown, and none of the modifications made to the database during the transaction will have been applied to the database.

In all cases the transaction will no longer be in progress.

Exceptions:

Xapian::DatabaseError will be thrown if a problem occurs while modifying the database.

Xapian::DatabaseCorruptError will be thrown if the database is in a corrupt state.

Xapian::InvalidOperationError will be thrown if a transaction is not currently in progress.

Xapian::UnimplementedError will be thrown if transactions are not available for this database type.

6.45.3.6 void Xapian::WritableDatabase::cancel_transaction ()

Abort the transaction currently in progress, discarding the pending modifications made to the database.

If an error occurs in this method, an exception will be thrown, but the transaction will be cancelled anyway.

Exceptions:

Xapian::DatabaseError will be thrown if a problem occurs while modifying the database.

Xapian::DatabaseCorruptError will be thrown if the database is in a corrupt state.

Xapian::InvalidOperationError will be thrown if a transaction is not currently in progress.

Xapian::UnimplementedError will be thrown if transactions are not available for this database type.

6.45.3.7 Xapian::docid Xapian::WritableDatabase::add_document (const Xapian::Document & *document*)

Add a new document to the database.

This method adds the specified document to the database, returning a newly allocated document ID. Automatically allocated document IDs come from a per-database monotonically increasing counter, so IDs from deleted documents won't be reused.

If you want to specify the document ID to be used, you should call replace_document() instead.

Note that changes to the database won't be immediately committed to disk; see commit() for more details.

As with all database modification operations, the effect is atomic: the document will either be fully added, or the document fails to be added and an exception is thrown (possibly at a later time when commit() is called or the database is closed).

Parameters:

document The new document to be added.

Returns:

The document ID of the newly added document.

Exceptions:

Xapian::DatabaseError will be thrown if a problem occurs while writing to the database.

Xapian::DatabaseCorruptError will be thrown if the database is in a corrupt state.

6.45.3.8 void Xapian::WritableDatabase::delete_document (Xapian::docid *did*)

Delete a document from the database.

This method removes the document with the specified document ID from the database.

Note that changes to the database won't be immediately committed to disk; see commit() for more details.

As with all database modification operations, the effect is atomic: the document will either be fully removed, or the document fails to be removed and an exception is thrown (possibly at a later time when commit() is called or the database is closed).

Parameters:

did The document ID of the document to be removed.

Exceptions:

Xapian::DatabaseError will be thrown if a problem occurs while writing to the database.

Xapian::DatabaseCorruptError will be thrown if the database is in a corrupt state.

6.45.3.9 void Xapian::WritableDatabase::delete_document (const std::string & *unique_term*)

Delete any documents indexed by a term from the database.

This method removes any documents indexed by the specified term from the database.

A major use is for convenience when UIDs from another system are mapped to terms in Xapian, although this method has other uses (for example, you could add a "deletion date" term to documents at index time and use this method to delete all documents due for deletion on a particular date).

Parameters:

unique_term The term to remove references to.

Exceptions:

Xapian::DatabaseError will be thrown if a problem occurs while writing to the database.

Xapian::DatabaseCorruptError will be thrown if the database is in a corrupt state.

6.45.3.10 void Xapian::WritableDatabase::replace_document (Xapian::docid *did*, const Xapian::Document & *document*)

Replace a given document in the database.

This method replaces the document with the specified document ID. If document ID *did* isn't currently used, the document will be added with document ID *did*.

The monotonic counter used for automatically allocating document IDs is increased so that the next automatically allocated document ID will be did + 1. Be aware that if you use this method to specify a high document ID for a new document, and also use WritableDatabase::add_document(), Xapian may get to a state where this counter wraps around and will be unable to automatically allocate document IDs!

Note that changes to the database won't be immediately committed to disk; see commit() for more details.

As with all database modification operations, the effect is atomic: the document will either be fully replaced, or the document fails to be replaced and an exception is thrown (possibly at a later time when commit() is called or the database is closed).

Parameters:

did The document ID of the document to be replaced.

document The new document.

Exceptions:

Xapian::DatabaseError will be thrown if a problem occurs while writing to the database.

Xapian::DatabaseCorruptError will be thrown if the database is in a corrupt state.

6.45.3.11 Xapian::docid Xapian::WritableDatabase::replace_document (const std::string & *unique_term*, const Xapian::Document & *document*)

Replace any documents matching a term.

This method replaces any documents indexed by the specified term with the specified document. If any documents are indexed by the term, the lowest document ID will be

used for the document, otherwise a new document ID will be generated as for add_document.

The intended use is to allow UIDs from another system to easily be mapped to terms in Xapian, although this method probably has other uses.

Note that changes to the database won't be immediately committed to disk; see commit() for more details.

As with all database modification operations, the effect is atomic: the document(s) will either be fully replaced, or the document(s) fail to be replaced and an exception is thrown (possibly at a later time when commit() is called or the database is closed).

Parameters:

unique_term The "unique" term.

document The new document.

Returns:

The document ID that document was given.

Exceptions:

Xapian::DatabaseError will be thrown if a problem occurs while writing to the database.

Xapian::DatabaseCorruptError will be thrown if the database is in a corrupt state.

6.45.3.12 void Xapian::WritableDatabase::add_spelling (const std::string & *word*, Xapian::termcount *freqinc* = 1) const

Add a word to the spelling dictionary.

If the word is already present, its frequency is increased.

Parameters:

word The word to add.

freqinc How much to increase its frequency by (default 1).

6.45.3.13 void Xapian::WritableDatabase::remove_spelling (const std::string & *word*, Xapian::termcount *freqdec* = 1) const

Remove a word from the spelling dictionary.

The word's frequency is decreased, and if would become zero or less then the word is removed completely.

Parameters:

word The word to remove.

freqdec How much to decrease its frequency by (default 1).

6.45.3.14 void Xapian::WritableDatabase::add_synonym (const std::string & *term*, const std::string & *synonym*) const

Add a synonym for a term.

If *synonym* is already a synonym for *term*, then no action is taken.

6.45.3.15 void Xapian::WritableDatabase::remove_synonym (const std::string & *term*, const std::string & *synonym*) const

Remove a synonym for a term.

If *synonym* isn't a synonym for *term*, then no action is taken.

6.45.3.16 void Xapian::WritableDatabase::clear_synonyms (const std::string & *term*) const

Remove all synonyms for a term.

If *term* has no synonyms, no action is taken.

6.45.3.17 void Xapian::WritableDatabase::set_metadata (const std::string & *key*, const std::string & *value*)

Set the user-specified metadata associated with a given key.

This method sets the metadata value associated with a given key. If there is already a metadata value stored in the database with the same key, the old value is replaced. If you want to delete an existing item of metadata, just set its value to the empty string.

User-specified metadata allows you to store arbitrary information in the form of (key,tag) pairs.

There's no hard limit on the number of metadata items, or the size of the metadata values. Metadata keys have a limited length, which depends on the backend. We recommend limiting them to 200 bytes. Empty keys are not valid, and specifying one will cause an exception.

Metadata modifications are committed to disk in the same way as modifications to the documents in the database are: i.e., modifications are atomic, and won't be committed to disk immediately (see commit() for more details). This allows metadata to be used to link databases with versioned external resources by storing the appropriate version number in a metadata item.

You can also use the metadata to store arbitrary extra information associated with terms, documents, or postings by encoding the termname and/or document id into the metadata key.

Parameters:

key The key of the metadata item to set.

value The value of the metadata item to set.

Exceptions:

Xapian::DatabaseError will be thrown if a problem occurs while writing to the database.

Xapian::DatabaseCorruptError will be thrown if the database is in a corrupt state.

Xapian::InvalidArgumentError will be thrown if the key supplied is empty.

Xapian::UnimplementedError will be thrown if the database backend in use doesn't support user-specified metadata.

6.45.3.18 std::string Xapian::WritableDatabase::get_description () const [virtual]

Return a string describing this object.

Reimplemented from Xapian::Database.

The documentation for this class was generated from the following file:

- include/xapian/database.h

Chapter 7

xapian-core File Documentation

7.1 include/xapian.h File Reference

Public interfaces for the Xapian library.

```
#include <xapian/version.h>
#include <xapian/types.h>
#include <xapian/error.h>
#include <xapian/errorhandler.h>
#include <xapian/database.h>
#include <xapian/dbfactory.h>
#include <xapian/document.h>
#include <xapian/positioniterator.h>
#include <xapian/postingiterator.h>
#include <xapian/termiterator.h>
#include <xapian/valueiterator.h>
#include <xapian/termgenerator.h>
#include <xapian/enquire.h>
#include <xapian/expanddecider.h>
#include <xapian/postingsource.h>
#include <xapian/query.h>
#include <xapian/queryparser.h>
#include <xapian/sorter.h>
#include <xapian/valuesetmatchdecider.h>
#include <xapian/weight.h>
```

```
#include <xapian/stem.h>
#include <xapian/serialisationcontext.h>
#include <xapian/unicode.h>
#include <xapian/visibility.h>
```

Namespaces

- namespace Xapian

Functions

- XAPIAN_VISIBILITY_DEFAULT const char * Xapian::version_string ()
- XAPIAN_VISIBILITY_DEFAULT int Xapian::major_version ()
- XAPIAN_VISIBILITY_DEFAULT int Xapian::minor_version ()
- XAPIAN_VISIBILITY_DEFAULT int Xapian::revision ()

7.1.1 Detailed Description

Public interfaces for the Xapian library.

7.2 include/xapian/database.h File Reference

API for working with Xapian databases.

```
#include <string>
#include <vector>
#include <xapian/base.h>
#include <xapian/document.h>
#include <xapian/types.h>
#include <xapian/positioniterator.h>
#include <xapian/postingiterator.h>
#include <xapian/termiterator.h>
#include <xapian/valueiterator.h>
#include <xapian/visibility.h>
```

Namespaces

- namespace Xapian

Classes

- class Xapian::Database
- class Xapian::WritableDatabase

Variables

- const int Xapian::DB_CREATE_OR_OPEN = 1
- const int Xapian::DB_CREATE = 2
- const int Xapian::DB_CREATE_OR_OVERWRITE = 3
- const int Xapian::DB_OPEN = 4

7.2.1 Detailed Description

API for working with Xapian databases.

7.3 include/xapian/dbfactory.h File Reference

Factory functions for constructing Database and WritableDatabase objects.

```
#include <string>
#include <xapian/types.h>
#include <xapian/version.h>
#include <xapian/visibility.h>
```

Namespaces

- namespace Xapian
- namespace **Xapian::Auto**
- namespace **Xapian::InMemory**
- namespace **Xapian::Chert**
- namespace **Xapian::Flint**
- namespace **Xapian::Remote**

Functions

- XAPIAN_VISIBILITY_DEFAULT Database Xapian::Auto::open_stub (const std::string &file)
- XAPIAN_VISIBILITY_DEFAULT WritableDatabase Xapian::Auto::open_stub (const std::string &file, int action)
- XAPIAN_VISIBILITY_DEFAULT WritableDatabase Xapian::InMemory::open ()
- XAPIAN_VISIBILITY_DEFAULT Database Xapian::Chert::open (const std::string &dir)
- XAPIAN_VISIBILITY_DEFAULT WritableDatabase Xapian::Chert::open (const std::string &dir, int action, int block_size=8192)
- XAPIAN_VISIBILITY_DEFAULT Database Xapian::Flint::open (const std::string &dir)
- XAPIAN_VISIBILITY_DEFAULT WritableDatabase Xapian::Flint::open (const std::string &dir, int action, int block_size=8192)
- XAPIAN_VISIBILITY_DEFAULT Database Xapian::Remote::open (const std::string &host, unsigned int port, Xapian::timeout timeout=10000, Xapian::timeout connect_timeout=10000)
- XAPIAN_VISIBILITY_DEFAULT WritableDatabase Xapian::Remote::open_writable (const std::string &host, unsigned int port, Xapian::timeout timeout=0, Xapian::timeout connect_timeout=10000)
- XAPIAN_VISIBILITY_DEFAULT Database Xapian::Remote::open (const std::string &program, const std::string &args, Xapian::timeout timeout=10000)
- XAPIAN_VISIBILITY_DEFAULT WritableDatabase Xapian::Remote::open_writable (const std::string &program, const std::string &args, Xapian::timeout timeout=0)

7.3.1 Detailed Description

Factory functions for constructing Database and WritableDatabase objects.

7.4 include/xapian/derefwrapper.h File Reference

Class for wrapping std::string returned by an input_iterator.

`#include <string>`

Namespaces

- namespace Xapian

Classes

- class **Xapian::DerefStringWrapper_**

7.4.1 Detailed Description

Class for wrapping std::string returned by an input_iterator.

7.5 include/xapian/document.h File Reference

API for working with documents.

```
#include <string>
#include <xapian/base.h>
#include <xapian/types.h>
#include <xapian/termiterator.h>
#include <xapian/valueiterator.h>
#include <xapian/visibility.h>
```

Namespaces

- namespace Xapian

Classes

- class Xapian::Document

  A document in the database - holds data, values, terms, and postings.

7.5.1 Detailed Description

API for working with documents.

7.6 include/xapian/enquire.h File Reference

API for running queries.

```
#include <string>
#include <xapian/base.h>
#include <xapian/deprecated.h>
#include <xapian/sorter.h>
#include <xapian/types.h>
#include <xapian/termiterator.h>
#include <xapian/visibility.h>
```

Namespaces

- namespace Xapian

Classes

- class Xapian::MSet
- class Xapian::MSetIterator
- class Xapian::ESet
- class Xapian::ESetIterator
- class Xapian::RSet
- class Xapian::MatchDecider
- class Xapian::Enquire

Functions

- bool **Xapian::operator==** (const MSetIterator &a, const MSetIterator &b)
- bool **Xapian::operator!=** (const MSetIterator &a, const MSetIterator &b)
- bool **Xapian::operator==** (const ESetIterator &a, const ESetIterator &b)
- bool **Xapian::operator!=** (const ESetIterator &a, const ESetIterator &b)

7.6.1 Detailed Description

API for running queries.

7.7 include/xapian/errorhandler.h File Reference

Decide if a Xapian::Error exception should be ignored.

```
#include <xapian/visibility.h>
```

Namespaces

- namespace Xapian

Classes

- class Xapian::ErrorHandler

7.7.1 Detailed Description

Decide if a Xapian::Error exception should be ignored.

7.8 include/xapian/expanddecider.h File Reference

Allow rejection of terms during ESet generation.

```
#include <set>
#include <string>
#include <xapian/visibility.h>
```

Namespaces

- namespace Xapian

Classes

- class Xapian::ExpandDecider
- class Xapian::ExpandDeciderAnd
- class Xapian::ExpandDeciderFilterTerms

7.8.1 Detailed Description

Allow rejection of terms during ESet generation.

7.9 include/xapian/positioniterator.h File Reference

Classes for iterating through position lists.

```
#include <iterator>
#include <string>
#include <xapian/base.h>
#include <xapian/types.h>
#include <xapian/visibility.h>
```

Namespaces

- namespace Xapian

Classes

- class **Xapian::TermPosWrapper**
- class Xapian::PositionIterator

Functions

- bool Xapian::operator== (const PositionIterator &a, const PositionIterator &b)

  Test equality of two PositionIterators.

- bool Xapian::operator!= (const PositionIterator &a, const PositionIterator &b)

  Test inequality of two PositionIterators.

7.9.1 Detailed Description

Classes for iterating through position lists.

7.10 include/xapian/postingiterator.h File Reference

Classes for iterating through posting lists.

```
#include <iterator>
#include <string>
#include <xapian/base.h>
#include <xapian/types.h>
#include <xapian/positioniterator.h>
#include <xapian/visibility.h>
```

Namespaces

- namespace Xapian

Classes

- class **Xapian::DocIDWrapper**
- class Xapian::PostingIterator

Functions

- bool Xapian::operator== (const PostingIterator &a, const PostingIterator &b)

  Test equality of two PostingIterators.

- bool Xapian::operator!= (const PostingIterator &a, const PostingIterator &b)

  Test inequality of two PostingIterators.

7.10.1 Detailed Description

Classes for iterating through posting lists.

7.11 include/xapian/postingsource.h File Reference

External sources of posting information.

```
#include <xapian/database.h>
#include <xapian/types.h>
#include <xapian/visibility.h>
#include <string>
#include <map>
```

Namespaces

- namespace Xapian

Classes

- class Xapian::PostingSource
- class Xapian::ValuePostingSource
- class Xapian::ValueWeightPostingSource
- class Xapian::ValueMapPostingSource
- class Xapian::FixedWeightPostingSource

7.11.1 Detailed Description

External sources of posting information.

7.12 include/xapian/query.h File Reference

Classes for representing a query.

```
#include <string>
#include <vector>
#include <xapian/base.h>
#include <xapian/types.h>
#include <xapian/termiterator.h>
#include <xapian/visibility.h>
```

Namespaces

- namespace Xapian

Classes

- class Xapian::Query
- class Xapian::Query

7.12.1 Detailed Description

Classes for representing a query.

7.13 include/xapian/queryparser.h File Reference

parsing a user query string to build a Xapian::Query object

```
#include <xapian/base.h>
#include <xapian/query.h>
#include <xapian/termiterator.h>
#include <xapian/visibility.h>
#include <set>
#include <string>
```

Namespaces

- namespace Xapian

Classes

- class Xapian::Stopper

  Base class for stop-word decision functor.

- class Xapian::SimpleStopper

  Simple implementation of Stopper class - this will suit most users.

- struct Xapian::ValueRangeProcessor

  Base class for value range processors.

- class Xapian::StringValueRangeProcessor
- class Xapian::DateValueRangeProcessor
- class Xapian::NumberValueRangeProcessor
- class Xapian::QueryParser

  Build a Xapian::Query object from a user query string.

Functions

- XAPIAN_VISIBILITY_DEFAULT std::string Xapian::sortable_serialise (double value)
- XAPIAN_VISIBILITY_DEFAULT double Xapian::sortable_unserialise (const std::string &value)

7.13.1 Detailed Description

parsing a user query string to build a Xapian::Query object

7.14 include/xapian/replication.h File Reference

Replication support for Xapian databases.

```
#include <xapian/base.h>
#include <xapian/visibility.h>
#include <string>
```

Namespaces

- namespace Xapian

Classes

- struct Xapian::ReplicationInfo
- class Xapian::DatabaseMaster
- class Xapian::DatabaseReplica

7.14.1 Detailed Description

Replication support for Xapian databases.

7.15 include/xapian/serialisationcontext.h File Reference

Context for looking up objects during unserialisation.

```
#include <xapian/base.h>
#include <xapian/visibility.h>
#include <string>
```

Namespaces

- namespace Xapian

Classes

- class Xapian::SerialisationContext

7.15.1 Detailed Description

Context for looking up objects during unserialisation.

7.16 include/xapian/sorter.h File Reference

Build sort keys for MSet ordering.

```
#include <string>
#include <vector>
#include <xapian/document.h>
#include <xapian/visibility.h>
```

Namespaces

- namespace Xapian

Classes

- class Xapian::Sorter
- class Xapian::MultiValueSorter

7.16.1 Detailed Description

Build sort keys for MSet ordering.

7.17 include/xapian/stem.h File Reference

stemming algorithms

```
#include <xapian/base.h>
#include <xapian/visibility.h>
#include <string>
```

Namespaces

- namespace Xapian

Classes

- class Xapian::Stem

  Class representing a stemming algorithm.

7.17.1 Detailed Description

stemming algorithms

7.18 include/xapian/termgenerator.h File Reference

parse free text and generate terms

```
#include <xapian/base.h>
#include <xapian/types.h>
#include <xapian/unicode.h>
#include <xapian/visibility.h>
#include <string>
```

Namespaces

- namespace Xapian

Classes

- class Xapian::TermGenerator

7.18.1 Detailed Description

parse free text and generate terms

7.19 include/xapian/termiterator.h File Reference

Classes for iterating through term lists.

```
#include <iterator>
#include <string>
#include <xapian/base.h>
#include <xapian/derefwrapper.h>
#include <xapian/types.h>
#include <xapian/positioniterator.h>
#include <xapian/visibility.h>
```

Namespaces

- namespace Xapian

Classes

- class Xapian::TermIterator

Functions

- bool **Xapian::operator==** (const TermIterator &a, const TermIterator &b)
- bool **Xapian::operator!=** (const TermIterator &a, const TermIterator &b)

7.19.1 Detailed Description

Classes for iterating through term lists.

7.20 include/xapian/types.h File Reference

typedefs for Xapian

Namespaces

- namespace Xapian

Typedefs

- typedef unsigned Xapian::doccount
- typedef int Xapian::doccount_diff
- typedef unsigned Xapian::docid
- typedef double Xapian::doclength
- typedef int Xapian::percent
- typedef unsigned Xapian::termcount
- typedef int Xapian::termcount_diff
- typedef unsigned Xapian::termpos
- typedef int Xapian::termpos_diff
- typedef unsigned Xapian::timeout
- typedef unsigned Xapian::valueno
- typedef int Xapian::valueno_diff
- typedef double Xapian::weight

Variables

- const valueno Xapian::BAD_VALUENO = static_cast<valueno>(-1)

7.20.1 Detailed Description

typedefs for Xapian

7.21 include/xapian/unicode.h File Reference

Unicode and UTF-8 related classes and functions.

```
#include <xapian/visibility.h>
#include <string>
```

Namespaces

- namespace Xapian
- namespace **Xapian::Unicode**
- namespace **Xapian::Unicode::Internal**

Classes

- class Xapian::Utf8Iterator

Enumerations

- enum **category** {

  UNASSIGNED, **UPPERCASE_LETTER**, **LOWERCASE_LETTER**, **TITLECASE_LETTER**,

  MODIFIER_LETTER, **OTHER_LETTER**, **NON_SPACING_MARK**, **ENCLOSING_MARK**,

  COMBINING_SPACING_MARK, **DECIMAL_DIGIT_NUMBER**, **LETTER_NUMBER**, **OTHER_NUMBER**,

  SPACE_SEPARATOR, **LINE_SEPARATOR**, **PARAGRAPH_SEPARATOR**, **CONTROL**,

  FORMAT, **PRIVATE_USE**, **SURROGATE**, **CONNECTOR_PUNCTUATION**,

  DASH_PUNCTUATION, **OPEN_PUNCTUATION**, **CLOSE_PUNCTUATION**, **INITIAL_QUOTE_PUNCTUATION**,

  FINAL_QUOTE_PUNCTUATION, **OTHER_PUNCTUATION**, **MATH_SYMBOL**, **CURRENCY_SYMBOL**,

  MODIFIER_SYMBOL, **OTHER_SYMBOL** }

Functions

- XAPIAN_VISIBILITY_DEFAULT int **Xapian::Unicode::Internal::get_character_info** (unsigned ch)
- int **Xapian::Unicode::Internal::get_case_type** (int info)
- category **Xapian::Unicode::Internal::get_category** (int info)
- int **Xapian::Unicode::Internal::get_delta** (int info)

- XAPIAN_VISIBILITY_DEFAULT unsigned Xapian::Unicode::nonascii_to_utf8 (unsigned ch, char *buf)
- unsigned Xapian::Unicode::to_utf8 (unsigned ch, char *buf)
- void Xapian::Unicode::append_utf8 (std::string &s, unsigned ch)
- category Xapian::Unicode::get_category (unsigned ch)

  Return the category which a given Unicode character falls into.

- bool Xapian::Unicode::is_wordchar (unsigned ch)

  Test if a given Unicode character is "word character".

- bool Xapian::Unicode::is_whitespace (unsigned ch)

  Test if a given Unicode character is a whitespace character.

- bool Xapian::Unicode::is_currency (unsigned ch)

  Test if a given Unicode character is a currency symbol.

- unsigned Xapian::Unicode::tolower (unsigned ch)

  Convert a Unicode character to lowercase.

- unsigned Xapian::Unicode::toupper (unsigned ch)

  Convert a Unicode character to uppercase.

- std::string Xapian::Unicode::tolower (const std::string &term)

  Convert a UTF-8 std::string to lowercase.

- std::string Xapian::Unicode::toupper (const std::string &term)

  Convert a UTF-8 std::string to uppercase.

7.21.1 Detailed Description

Unicode and UTF-8 related classes and functions.

7.22 include/xapian/valueiterator.h File Reference

Class for iterating over document values.

```
#include <iterator>
#include <string>
#include <xapian/base.h>
#include <xapian/derefwrapper.h>
#include <xapian/types.h>
#include <xapian/visibility.h>
```

Namespaces

- namespace Xapian

Classes

- class **Xapian::ValueIteratorEnd_**
- class Xapian::ValueIterator

  Class for iterating over document values.

Functions

- bool **Xapian::operator==** (const ValueIterator &a, const ValueIterator &b)
- bool **Xapian::operator==** (const ValueIterator &a, const ValueIteratorEnd_ &)
- bool **Xapian::operator==** (const ValueIteratorEnd_ &a, const ValueIterator &b)
- bool **Xapian::operator==** (const ValueIteratorEnd_ &, const ValueIteratorEnd_ &)
- bool **Xapian::operator!=** (const ValueIterator &a, const ValueIterator &b)
- bool **Xapian::operator!=** (const ValueIterator &a, const ValueIteratorEnd_ &b)
- bool **Xapian::operator!=** (const ValueIteratorEnd_ &a, const ValueIterator &b)
- bool **Xapian::operator!=** (const ValueIteratorEnd_ &a, const ValueIteratorEnd_ &b)

7.22.1 Detailed Description

Class for iterating over document values.

7.23 include/xapian/valuesetmatchdecider.h File Reference

MatchDecider subclass for filtering results by value.

```
#include <xapian/enquire.h>
#include <xapian/types.h>
#include <string>
#include <set>
```

Namespaces

- namespace Xapian

Classes

- class Xapian::ValueSetMatchDecider

7.23.1 Detailed Description

MatchDecider subclass for filtering results by value.

7.24 include/xapian/weight.h File Reference

Weighting scheme API.

```
#include <string>
#include <xapian/types.h>
#include <xapian/visibility.h>
```

Namespaces

- namespace Xapian

Classes

- class Xapian::Weight
- class Xapian::BoolWeight
- class Xapian::BM25Weight

  Xapian::Weight subclass implementing the BM25 probabilistic formula.

- class Xapian::TradWeight

7.24.1 Detailed Description

Weighting scheme API.

Index