Official PostgreSQL 8.3 TSearch Documentation URL:
http://www.postgresql.org/docs/8.3/static/textsearch.html
http://www.postgresonline.com
We cover only a subset of what we feel are the most useful constructs that we could squash in a single cheatsheet page
commonly used

pg_catalog Tables

pg_ts_config
pg_ts_config_map
pg_ts_dict
pg_ts_parser
pg_ts_template

Data Types

regconfig
regdictionary
tsquery
tsvector

Query Condition Operators

|
&
!

Query Operators

!!
&&
||

Vector Operators

||
@@
@@@

DDL

CREATE TEXT SEARCH DICTIONARY
ALTER TEXT SEARCH CONFIGURATION

Trigger Functions

tsvector_update_trigger()
tsvector_update_trigger_column()

Functions

numnode(tsquery)
plainto_tsquery(text)
plainto_tsquery(regconfig, text)
querytree(tsquery)
setweight(tsvector, "char")
strip(tsvector)
to_tsquery(text)
to_tsquery(regconfig, text)
to_tsvector(text)
to_tsvector(regconfig, text)
ts_debug(*text*)
ts_headline(*text*, *tsquery*)
ts_headline(*regconfig*, *text*, *tsquery*, *text*)
ts_headline(regconfig, text, tsquery)
ts_headline(text, tsquery, text)
ts_lexize(regdictionary, text)
ts_match_qv(tsquery, tsvector)
ts_match_tq(text, tsquery)
ts_match_tt(text, text)
ts_match_vq(tsvector, tsquery)
ts_parse(oid, text)
ts_parse(text, text)
ts_rank(real[], tsvector, tsquery)
ts_rank(tsvector, tsquery)
ts_rank(tsvector, tsquery, integer)
ts_rank_cd(tsvector, tsquery)
ts_rewrite(tsquery, tsquery, tsquery)
ts_rewrite(tsquery, text)
ts_stat(text)
ts_stat(text, text)

DDL AND DATA LOAD EXAMPLES

```
CREATE TABLE sometable
( myid serial PRIMARY KEY, title varchar(255), description text,
  mytsfield tsvector, myconfig regconfig);

CREATE INDEX idx_sometable_somefield
 ON sometable
  USING gin(to_tsvector('pg_catalog.english', mytsfield));

--This is if you don't want to store ts vector and you always want to recalc from fields as needed
CREATE INDEX idx_sometable_ts
 ON sometable
  USING gin(to_tsvector(myconfig, COALESCE(title,'') || ' ' || COALESCE(description)));

INSERT INTO sometable(title, description,myconfig, mytsfield)
        VALUES('John Doe', 'a story about a man name John', 'pg_catalog.english',
        to_tsvector('pg_catalog.english', 'John Doe' || ' ' || 'a story about a man name John'));
```

```
CREATE TEXT SEARCH DICTIONARY thesaurus_simple (
 TEMPLATE = thesaurus,
 DictFile = mythesaurus,
 Dictionary = pg_catalog.english_stem
);

ALTER TEXT SEARCH CONFIGURATION russian
 ALTER MAPPING FOR asciiword, asciihword, hword_asciipart
  WITH thesaurus_simple;
```

TRIGGER EXAMPLES

--This trigger updates the field tsvector type field called
mytsvfield in mytable with combined tsvector of field1, field2,..fieldn fields in the table.
Note -- number of fields is not limited. All fields are equally weighted.

```
CREATE TRIGGER mytable_mytsvfield_trigger
  BEFORE INSERT OR UPDATE
  ON mytable
  FOR EACH ROW
  EXECUTE PROCEDURE
   tsvector_update_trigger('mytsvfield', 'pg_catalog.english', 'field1', 'field2');
```

--tsvector_update_trigger_column is similar to the
tsvector_update_trigger except
instead of taking a constant configuration,you specify a column
in the table which defines the configuration.
This allows for multi-lingual records in the same table
with different full text search rules.
Note that myconfig_column must be name of a table column of type *regconfig*

```
CREATE TRIGGER mytable_mytsvfield_trigger
 BEFORE INSERT OR UPDATE ON mytable
 FOR EACH ROW EXECUTE PROCEDURE
   tsvector_update_trigger_column('mytsvfield', 'myconfig_column', 'field1', 'field2');
```

--An example of using your own custom trigger instead of using built-in ones

```
CREATE FUNCTION mytable_ft_trigger() RETURNS trigger AS $$
begin
  new.tsv :=
     setweight(to_tsvector('pg_catalog.english',
                coalesce(new.field1,'')), 'A') ||
     setweight(to_tsvector('pg_catalog.english',
                coalesce(new.field2,'')), 'B');
  return new;
end
$$ LANGUAGE plpgsql;
CREATE TRIGGER mytable_trigiu BEFORE INSERT OR UPDATE
ON mytable FOR EACH ROW EXECUTE PROCEDURE mytable_ft_trigger();
```

SIMPLE QUERY EXAMPLES

```
--Snippet two  - examples using TQuery
--We want to check if the provided phrase contains the words dog and sick.  This returns true
SELECT to_tsvector('english', 'My dog is sick')
    @@ to_tsquery('english', 'dog & SICK');

--This one uses a plain text string and convertes to a valid tsquery
NOTE: plain to_tsquery will try to convert a plain text statement to valid ts query this returns
ts_query - "'sick' & 'dog' & 'manhattan'"
SELECT plainto_tsquery('english','sick dogs in manhattan');

--This one is false because doggy is not a word boundary for dog
SELECT to_tsvector('english', 'My doggy is sick')
    @@ to_tsquery('english', 'dog & SICK');

--However dogs and dog are lexically equivalent so this is true
SELECT to_tsvector('english', 'I want a dog')
    @@@ to_tsquery('english', 'want & dogs');
```

```
See list of configurations
SELECT cfgname FROM pg_catalog.pg_ts_config;
--This one is also true because ski and skiing
--are derived from same word (lexeme)
SELECT to_tsvector('english', 'I like to ski')
    @@ to_tsquery('english', 'like & skiing');
--This uses the default locale
SELECT to_tsvector('My dog is sick')
    @@ to_tsquery('dog & SICK');
```

```
--Search all views that have SUM or FILM
SELECT * FROM information_schema.views
WHERE to_tsvector(view_definition)
 @@ to_tsquery('sum | film');

--Search all records in sometable use myconfig column to determine configuration to use
SELECT *
FROM sometable
WHERE mytsfield
 @@ to_tsquery(myconfig, '(sum & store) & !film ');
```

RANKING EXAMPLES

```
--Weight positions are demarcated by the letters A, B, C, D. Create a fulltext field where the title is
--marked as weight position A and description is weight position B
ALTER TABLE film ADD COLUMN ftext_weighted tsvector;
UPDATE film SET ftext_weighted = (setweight(to_tsvector(title), 'A')
        || setweight(to_tsvector(description), 'B'));
CREATE INDEX idx_books_ftext_weighted ON film
    USING gin(ftext_weighted);

--List top 3 films about Mysql that are epic, documentary or chocolate
--NOTE: the {0,0,0.10,0.90} corresponds to weight positions
--D,C, B, A and the sum of the array should be 1 which means
--weight the title higher than the summary
--NOTE: we are doing a subselect here because if we don't the expensive
--highlight function gets called all the results that match the WHERE, not just the highest 3
SELECT title, description, therank, ts_headline(title || ' ' || description, q,
 'StartSel = , StopSel = , HighlightAll=TRUE') as htmlmarked_summary
FROM (SELECT title, description,  ts_rank('{0,0,0.10,0.90}', ftext_weighted, q) as therank, q
  FROM film, to_tsquery('(epic | documentary | chocolate) & mysql') as q
  WHERE ftext_weighted @@ q
  ORDER BY therank DESC
   LIMIT 3) As results;
```

```
--List top 3 films with (chocolate, secretary , or mad) and (mysql or boring)
in the title or description
--the {0,0,0.90,0.10} corresponds to weight positions
--D,C, B, A which means based on how we weighted our index weight the title higher than the summary.
--This time we are using ts_rank_cd which will penalize
--query words that are further apart
--For highlighting this uses the default ts_headline which is to make terms bold
SELECT title, description,  therank,
  ts_headline(title || ' ' || description, q) as htmlmarked_summary
FROM (SELECT title, description,
    ts_rank_cd('{0,0,0.9,0.10}', ftext_weighted, q) as therank, q
    FROM film,
        to_tsquery('(chocolate | secretary | mad) & (mysql | boring)') as
    WHERE ftext_weighted @@ q
    ORDER BY therank DESC
    LIMIT 3) As results;

--We only want to count secretary and mad (:A) if it appears in the title of the document
--NOTE: Since we are using a GIN index, we need to use the slower @@@
SELECT title, description,  therank,
 ts_headline(title || ' ' || description, q) as htmlmarked_summary
FROM (SELECT title, description,
 ts_rank_cd('{0,0,0.9,0.10}', ftext_weighted, q) as therank, q
         FROM film, to_tsquery('(chocolate | secretary:A | mad:A)
                        & (mysql | boring)') as q
     WHERE ftext_weighted @@@ q
     ORDER BY therank DESC
    LIMIT 3) As results;
```

http://www.postgresonline.com