Database - Support #11863

review and correct Built-In Functions in WHERE Clauses support levels

09/10/2026 07:08 PM - Greg Shah

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**reviewer:**	
vendor_id:	GCD	**production:**	No
case_num:		**env_name:**	
version_reported:		**topics:**	
version_resolved:			

Description

History

#1 - 09/10/2026 07:10 PM - Greg Shah

Proposed Gap Marking Support Levels: Built-In Functions in WHERE Clauses

Purpose

The **Built-In Functions Called in WHERE Clauses** report now carries support level columns. Because a built-in behaves differently inside a WHERE clause than it does in ordinary code, the levels come from a WHERE-specific map (addWhereClauseFuncs() in rules/gaps/database.rules) rather than from the general built-in function marking in rules/gaps/expressions.rules.

This document records how every level was derived so the team can correct it. See Gap Analysis for the meaning of each support level.

The changes are in 11747a revision 16740. Please review and correct.

How the levels were derived

The conversion level is always carried over from the general built-in marking. The runtime level reflects how far FWD can push the call into the database:

1. **Server-side translation.** annotations/where_clause.rules maps a fixed set of built-ins onto FQL functions (entryIn(, substringOf(, trimws(, toString(and so on) under the server_op flag. These are pushed into the generated SQL, so the general level is kept unchanged.
2. **Hoisted to a substitution parameter.** A function which cannot reference a buffer field is evaluated once when the query is built and passed as a parameter, so there is no per-row cost and the general level is kept unchanged.
3. **Client-side evaluation.** Anything else leaves the enclosing expression unconvertible to FQL, so FWD evaluates it on the client, row by row. Results are correct but server-side filtering is permanently given up, so a general runtime level of Full is reduced to **Full (R)** here.
4. **Already limited.** Where the general runtime level is already below Full, it is carried over unchanged rather than being reduced further.

Group 1 - server-side translation, general level kept (37)

These have an explicit FQL translation, so using them in a WHERE costs nothing. All are Full / Full today except CAN-FIND(), which is Partial / Full from the general marking.

add_invl, can_do, can_find, caps, chr, date, date_tz, datetime, day, dec, entry, fill, if, index, int, int64, interval, l_trim, length, logical, lookup, max, min, month, mtime, num_ent, recid, replace, round, rowid, string, substr, timezone, to_rowid, trim, week, year

Group 2 - hoisted to a substitution parameter, general level kept (5)

etime, now, time, today, userid

Group 3 - client-side evaluation, runtime reduced to Full (R) (113)

This is the group that most needs review. The reduction is applied uniformly, on the basis that the enclosing expression cannot be pushed to the

database. Two things to check:

- Is Full (R) the right level, or should client-side evaluation be Basic or Partial instead?
- Several of these can never sensibly appear in a WHERE clause at all (ACCUMULATE, ALIAS, DYNAMIC-FUNCTION and similar). They are harmless because the report only ever shows functions a project actually uses, but the list can be pruned if you would rather it only covered realistic cases.

abs, accum, alias, all, ambig, asc, avail, base64_d, base64_e, can_qry, can_set, cast, cbit, conn_ed, count_of, cur_chg, cur_res, cur_val, data_sm, dbtaskid, dbtype, dbvers, dyn_cast, dyn_curv, dyn_enum, dyn_func, dyn_invk, dyn_new, dyn_nexv, encode, entered, exp, extent, first, first_of, fr_down, fr_line, fr_row, gen_pbek, get_b_or, get_bits, get_byte, get_byts, get_clnt, get_dbl, get_flt, get_i64, get_long, get_ptr, get_shrt, get_str, get_sz, get_ul, get_usht, getclass, guid, handle, hashcode, hex_decd, hex_encd, hwnd, input, iso_date, kblabel, keycode, keyfunc, keylab, kw, last, last_of, lc, ldbname, line_cnt, load_pic, locked, log, md5_dig, member, new, next_val, num_res, os_g_env, p2j_rc, page_num, page_sz, pdbname, progname, prover, qry_off, quoter, r_index, r_trim, random, rec_len, retry, rgb_val, row_stat, search, seek, set_ptr, sqrt, substit, super, ten_id, tenname, trans, trunc, type_of, u_msg, val_evt, val_hnd, val_obj, wid_hand

Group 4 - general level already below Full, carried over unchanged (72)

aud_enab, avg, avl_msgs, box, check_am, compare, count, cp_cvt, cvt_dt, db_rem_h, dbcoll, dbcp, dbparam, dbrest, decrypt, del_cook, dyn_prop, encrypt, fmt_dt, fr_col, get_cfg, get_cgi, get_cgl, get_cll, get_codp, get_coll, get_cook, get_dbcl, get_fld, get_l_v, get_lic, get_msgg, get_msgs, get_u_f, get_val, hid_fld, hid_fldl, html_enc, is_colcp, is_cp_fx, is_db_mt, is_lead, lib, lst_evnt, lst_qry, lst_set, lst_wid, msg_dig, normalze, out_hh, out_msgs, outputct, que_msg, raw, rejected, sdbname, set_cook, set_dbcl, set_lstk, set_u_f, set_w_s, setuser, sha1_dig, ssl_srvn, sum, tn_toid, unbox, url_deco, url_enco, url_fld, url_fldl, url_fmt

Known side effect

Support level is a single annotation per AST node, and a built-in function call inside a WHERE clause is also reported by the general **Builtin Function Usage** report. A call which is reduced to Full (R) here therefore carries that reduced level in the general report too, for that occurrence. That is arguably accurate, since that particular usage really does give up server-side filtering, but it means a single WHERE clause usage can lower the level shown for a function which is otherwise fully supported. Confirm this is acceptable before the marking is relied on.