Place of improvement identified: ora2pg Fails to Translate Explicit Cursor Attributes

Summary

The ora2pg PL/SQL translator does not correctly convert explicit cursor attributes like cursor_name%ISOPEN and cursor_name%ROWCOUNT, leading to invalid PostgreSQL code that fails at runtime.

Steps to Reproduce

1. Create an Oracle procedure that uses %ISOPEN and %ROWCOUNT on an explicit cursor.
2. Configure ora2pg.conf with PLSQL_PGSQL = 1.
3. Run ora2pg to translate the procedure.
4. Execute the generated SQL script in PostgreSQL and call the procedure.

Observed Behavior (The Bug)

The procedure call fails with the runtime error: ERROR: column "isopen" does not exist. The generated code contains the untranslated Oracle attributes c1%ISOPEN and c1%ROWCOUNT.

Expected Behavior (The Fix)

ora2pg should generate valid PL/pgSQL code where %ISOPEN is replaced with a boolean tracking variable and %ROWCOUNT is replaced with a GET DIAGNOSTICS call. The procedure should execute without errors.

Implemented Solution

The bug was fixed by creating a new, dedicated subroutine, handle_explicit_cursor_attributes, within lib/Ora2Pg/PLSQL.pm and integrating it into the main translation pipeline.

1. Integration Point

The new subroutine is called from plsql_to_plpgsql immediately after the existing cursor declaration is processed:

```
Perl


# In plsql_to_plpgsql:
$str = replace_cursor_def($str);

# Handle explicit cursor attributes (%ISOPEN, %ROWCOUNT)
$str = handle_explicit_cursor_attributes($class, $str);
```

2. Core Logic (The New Subroutine)

The new handle_explicit_cursor_attributes subroutine performs the required context-aware transformations:

Perl

```
sub handle_explicit_cursor_attributes
{
    my ($class, $str) = @_;

    # 1. Collect declared explicit cursor names
    my %cursor_names = ();
    if ($str =~ /(DECLARE.*?BEGIN)/is) {
        my $declare_block = $1;
        while ($declare_block =~ /\b([A-Z0-9_\$]+)\s+CURSOR\b/igs) {
            $cursor_names{$1} = 1;
        }
    }
    return $str if (!%cursor_names);

    # 2. Prepare and inject new variable declarations
    my @isopen_decls = map { "  ${_}_is_open BOOLEAN := FALSE;" } keys %cursor_names;
    my $rowcount_decl = '  v_ora2pg_row_count INTEGER;';
    $str =~ s/(\bDECLARE\b)/$1\n@isopen_decls\n$rowcount_decl\n/is;

    # 3. Handle %ISOPEN translation
    foreach my $cname (keys %cursor_names) {
        # Add state tracking after OPEN/CLOSE
        $str =~ s/(\bOPEN\s+$cname\s*;)/$1\n$cname\_is_open := TRUE;/igs;
        $str =~ s/(\bCLOSE\s+$cname\s*;)/$1\n$cname\_is_open := FALSE;/igs;
        # Replace the attribute check
        $str =~ s/\b$cname\s*\%\s*ISOPEN\b/${cname}_is_open/igs;
    }

    # 4. Handle %ROWCOUNT translation
    foreach my $cname (keys %cursor_names) {
        # Add GET DIAGNOSTICS after every FETCH
        $str =~ s/(\bFETCH\s+.*?$cname.*?;)/$1\nGET DIAGNOSTICS v_ora2pg_row_count =
ROW_COUNT;/igs;
        # Replace the attribute check
        $str =~ s/\b$cname\s*\%\s*ROWCOUNT\b/v_ora2pg_row_count/igs;
    }

    return $str;
}
```