Oracle OUTER JOIN syntax (+) Translation

Some of the procedures that Bull translated for a South American customer included queries that used the popular Oracle OUTER JOIN syntax to specify LEFT or RIGHT OUTER JOINs. With this syntax, the SQL developer appends the character sequence (+) to a term in the restriction to specify that the term is optional and that the term opposite from it (the one it's joined with) is required. Postgres does not accept the Oracle syntax. Fortunately, it does accept the standardized LEFT OUTER JOIN and RIGHT OUTER JOIN syntax. So Bull incorporated this into our translation tool. It's not as simple as just replacing the Oracle syntax with LEFT or RIGHT. Much more is involved in the process -- we trust that you already appreciate from first hand experience or will after reading this paper.

To illustrate join translation, let's first examine an Oracle based OUTER JOIN query and its execution results then let's look at the Postgres equivalent.

The test data we will use consists of three tables. The first contains recording artists. The second table contains albums produced by those artists. The third connects the one to many relationship between artists and albums and it includes an entry for albums under contract but not published yet. Only albums actually published are listed in the albums table.

```
create table artists (
  artistId int primary key,
  lastName char(20),
  firstName char(20)
);
insert into artists (artistId, lastName, firstName) values (400, 'Urban', 'Keith');
insert into artists (artistId, lastName, firstName) values (401, 'Swift', 'Taylor');

create table albums (
  albumId int primary key,
  title varchar(20)
);
insert into albums (albumId, title) values (1, 'Be Here');
insert into albums (albumId, title) values (2, 'Defying Gravity');
insert into albums (albumId, title) values (3, 'Golden Road');
insert into albums (albumId, title) values (100, 'Taylor Swift');
insert into albums (albumId, title) values (101, 'Fearless');

create table artists_albums (
  artistId int not null,
  albumId  int
);
insert into artists_albums (artistId, albumId) values (400, 1);
insert into artists_albums (artistId, albumId) values (400, 2);
insert into artists_albums (artistId, albumId) values (400, 3);
insert into artists_albums (artistId, albumId) values (401, 100);
insert into artists_albums (artistId, albumId) values (401, 101);
insert into artists_albums (artistId, albumId) values (401, 102);
```

To view the all albums known for Taylor Swift and Keith Urban plus to determine if there are any under contract we use an OUTER JOIN query. Here is the Oracle query:

```
SELECT lastName, firstName, title
  FROM artists tar, albums tal, artists_albums tli
 WHERE tar.artistId = tli.artistId
   and tli.albumId =  tal.albumId(+)
  ORDER BY lastName;
```

And its results:

```
LASTNAME             FIRSTNAME            TITLE
-------------------- -------------------- --------------------
Swift                Taylor               Fearless
Swift                Taylor               Taylor Swift
Swift                Taylor
Urban                Keith                Defying Gravity
Urban                Keith                Be Here
Urban                Keith                Golden Road
```

From this we can ascertain that Taylor Swift is under contract currently to produce a new album. Without the (+) directive, the third row for Taylor Swift would not exist because the join condition `tli.albumId = tal.albumId` would fail for albumId 102.

The translation to Postgres involves:

identifying the required table(s) and noting them with the LEFT clause of the OUTER JOIN directive.
listing the conditions involved in the OUTER JOIN in a new clause involving the keyword ON.
bridging "dangling" OUTER JOINs with INNER JOINs (example to follow).
grouping the remaining conditions in the WHERE clause together following the ON list.
dropping the Oracle (+) notation.

Using our translation tool, the prior Oracle query translated to:

```
  SELECT lastName, firstName, title
    FROM
      artists tar,
      artists_albums tli LEFT OUTER JOIN albums tal
        ON tli.albumId = tal.albumId
  WHERE
      tar.artistId = tli.artistId
  ORDER BY lastName;
```

And generates the same results as its Oracle equivalent:

```
       lastname        |      firstname        |      title
-----------------------+-----------------------+------------------
 Swift                 | Taylor                | Fearless
 Swift                 | Taylor                | Taylor Swift
 Swift                 | Taylor                |
 Urban                 | Keith                 | Defying Gravity
 Urban                 | Keith                 | Be Here
 Urban                 | Keith                 | Golden Road
```

In this query the LEFT OUTER JOIN clause indicates that the artists_albums table is required while the albums table is optional.

As the complexity of the SQL statement to translate increases, the need for automation increases also. Consider the following more typical case that is only slightly more difficult (3 outer joins instead of just 1 and 5 tables instead of 3):

```
 SELECT t1.col1, t1.col2, t2.col3, t3.col4, t5.c2 as col5
   from t1, t2, t3, t4, t5
   where t1.col2 = t2.col2(+)
         and   t2.col3 = t3.col3(+)
         and   t1.col1 = t4.c1
         and   t4.c1 = t5.c1(+)
   order by 1;
```

The Postgres equivalent (as generated by our translation tool) is:

```
SELECT t1.col1, t1.col2, t2.col3, t3.col4, t5.c2 as col5
      FROM
          t1 LEFT OUTER JOIN t2
            ON t1.col2 = t2.col2
              INNER JOIN t4
                ON t1.col1 = t4.c1
              LEFT OUTER JOIN t5
                ON t4.c1 = t5.c1
              LEFT OUTER JOIN t3
                ON t2.col3 = t3.col3
      order by 1;
```

Its worth noting that the explicit INNER JOIN was required to associate t4 with t1 before the OUTER JOIN between t4 and t5 could be specified. To say the least, translating this by hand would be a very time consuming process fraught with the possibility of error.