Migrating RDML & Rdb SQL to PostgreSQL on OpenVMS/x86

VX/RDML modernizes the source for Rdb access on OpenVMS/Itanium.
VX/SQL finishes the job on OpenVMS/x86 — pre-processor path or library-call path.

Sector7 Migration Tools

Followed by a live demo

Get off RDML without leaving Rdb.

01

Stay on the database

The executable still attaches to AVERTZ_DB. No data move, no schema change, and the fallback is the original RDML image.

02

Stay on OpenVMS

Screens, COPY records, and the Q / L / U / X dialogue do not move. Only the database statements are rewritten.

03

Leave a SQL dialect

Embedded SQL is what SQLPRE, Pro*COBOL, and ECPG already understand. RDML is not.

VX/RDML processes multiple languages

COBOL, Fortran, Pascal, and C

Running VX/RDML

```
rdml_sql -- RDML to SQL rewriter for COBOL, Fortran, Pascal, and C source

Usage:
 rdml_sql [options] INPUT

Host language:
 --lang=cobol|fortran|pascal|c  pick the host front end
                        (default: auto-detect from INPUT's extension)

    .cob  .cbl  .cobol  .rco  .sco   -> cobol
    .f    .f77  .f90    .f95  .for   -> fortran
    .pas  .pp   .rpa                 -> pascal
    .rc   .pc   .sc     .ec          -> c
```

CUSTINQ; one program, two sources

Customer inquiry against AVERTZ.CUSTOMERS. The SQL source is the VX/RDML output, not a hand rewrite.

Action	What the user does
Q	Query one customer by CUSTOMER_ID
L	List every customer with a given LAST_NAME
U	Query one customer and replace the audit memo
X	Commit and exit

FILES

CUSTINQ.RCO
RDBPRE source, unchanged.

CUSTINQ.SCO
VX/RDML output. Renamed CUSTSQL.SCO on VMS.

AVERTZ.SQL
RMU/EXTRACT metadata. VX/RDML does not read the live database.

The COBOL around the database stays COBOL.

Screens	Same prompts, same Q / L / U / X loop.
Host variables	Bridge names such as RDML-O1-V0 are kept, with indicator variables.
CDD copy	The workspace COPY is left in place. --no-cdd does not invent a record.
Database	FILENAME stays AVERTZ_DB. No new schema, no data conversion.
Error blocks	&RDB& ON ERROR / END_ERROR is reported, not silently rewritten.

A stream becomes a cursor

RDML · stream

```
&RDB& DECLARE_STREAM RDML_C1_C
   USING C IN CUSTOMERS
   WITH C.CUSTOMER_ID =
        CUSTOMER_ID_IN

&RDB& OPEN RDML_C1_C

&RDB& FETCH RDML_C1_C

&RDB& END_STREAM RDML_C1_C
```

EMBEDDED SQL · cursor

```
EXEC SQL DECLARE RDML_C1_C
  CURSOR FOR
  SELECT C.CUSTOMER_ID, ...
END-EXEC.

EXEC SQL OPEN RDML_C1_C END-EXEC.
EXEC SQL FETCH RDML_C1_C INTO
  :RDML-O1-V0:RDML-O1-N0, ...
END-EXEC.
EXEC SQL CLOSE RDML_C1_C END-EXEC.
```

The Takeaway

VX/RDML

On the existing OpenVMS system

Removes RDML and replaces it with ordinary embedded SQL.

You stay on the same platform and can test against the existing Rdb database.

Hands the application to people who already know embedded SQL — far more common than RDML expertise.

VX/SQL

On OpenVMS/x86

Takes either the embedded SQL or the original RDML and reaches a modern database.

- Pre-processor path: standard EXEC SQL → sql$pre → PostgreSQL runtime.

- Library-call path: plain COBOL with S7 calls → no pre-processor → link Postgres, Oracle, or SQL Server.

VX/RDML · Remove RDML, keep the platform

File names from the example

CUSTINQ_T86.RCO — original RDML source
CUSTINQ_RDML.SCO — embedded SQL produced by the rewrite
CUSTSQL_T86.SCO — pure embedded SQL (already SQL, no RDML)

Why this step matters

- RDML is rare; embedded SQL is widely known.
- You can compile and test on the existing OpenVMS system against the live Rdb database.
- The resulting .SCO is ordinary embedded SQL — the same form that VX/sql$pre will later process on x86.
- No database change yet. This is a source modernization step.

VX/SQL on OpenVMS/x86 · Two paths

Path A · Pre-processor

Standard EXEC SQL that needs sql$pre

1. Source is .SCO (embedded SQL) or .RCO rewritten to .SCO.

2. VX/ql$pre / scob parses the SQL and generates COBOL plus calls into the VX/SQL runtime.

3. COBOL compile + link VXSQL_PGSQL_RUNTIME_SHR.

4. Executable reaches PostgreSQL through the runtime (libpq).

Path B · Library calls

S7 runtime calls — no pre-processor

1. rdml_sql --mode=libcall rewrites RDML into plain COBOL with S7ESQL… calls.

2. COBOL compiler only — no sql$pre.

3. Link the runtime for the target database.

4. Same source reaches PostgreSQL, Oracle, or SQL Server.

Example · CUSTSQL_T86 (pure embedded SQL)

Build steps

```
CUSTSQL_T86.SCO
    →  sql$pre / scob
    →  CUSTSQL_T86.COB
    →  COBOL compile
    →  link VXSQL_PGSQL_RUNTIME_SHR
    →  CUSTSQL_T86.EXE
```

What the source looks like

```
EXEC SQL CONNECT TO :dbname;
EXEC SQL SELECT name, balance
    INTO :ws-name, :ws-balance
    FROM customer
    WHERE id_number = :ws-id;
EXEC SQL DISCONNECT;
```

VX/sql$pre turns these statements into calls into the VX/SQL runtime. The program never calls libpq directly.

Example · CUSTINQ_RDML (RDML → embedded SQL)

Build steps

```
CUSTINQ_T86.RCO          (original RDML)
    →  rdml_sql --mode=execsql
    →  CUSTINQ_RDML.SCO
    →  sql$pre / scob
    →  CUSTINQ_RDML.COB
    →  link VXSQL_PGSQL_RUNTIME_SHR
    →  CUSTINQ_RDML.EXE
```

Same end result as pure SQL

The RDML is rewritten once into ordinary embedded SQL.

From that point the path is identical to CUSTSQL_T86:
VX/sql$pre, COBOL compile, link the PostgreSQL runtime.

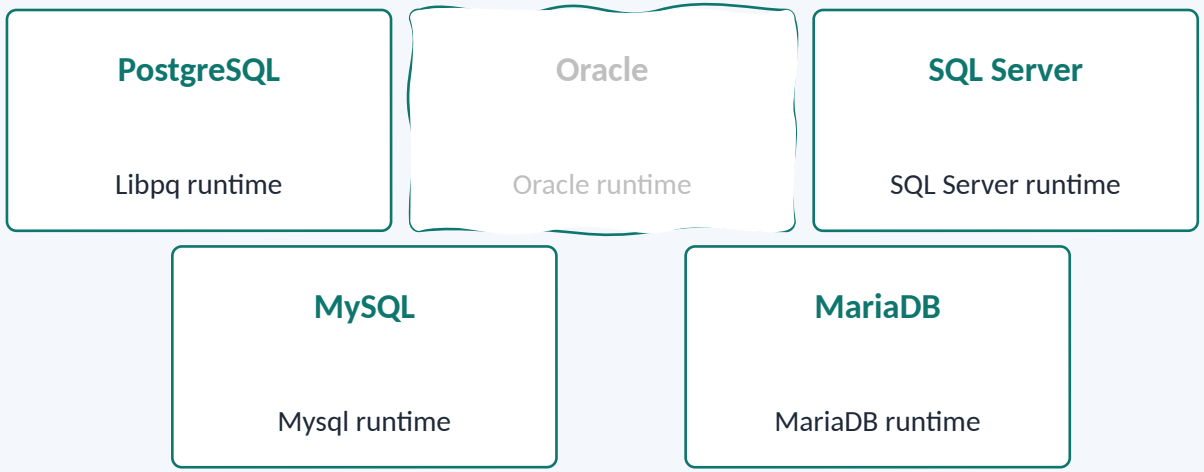
No Rdb is required on x86.
The executable reaches PostgreSQL through the VX/SQL runtime.

Example · CUSTINQ_LIBCALL (no pre-processor)

Build steps

```
CUSTINQ_T86.RCO
    →  rdml_sql --mode=libcall
    →  CUSTINQ_LIBCALL.COB   (plain COBOL)
    →  COBOL compiler only
    →  link chosen runtime
    →  CUSTINQ_LIBCALL.EXE
```

Same source, Multiple targets



What the generated COBOL looks like

```
CALL "S7ESQL_CONNECT" USING …
CALL "S7ESQL_EXEC" USING …
CALL "S7ESQL_FETCH" USING …
CALL "S7ESQL_DISCONNECT" USING …
```

No EXEC SQL remains. The COBOL compiler sees ordinary CALL statements. The database is chosen only by which runtime is linked.

Next · Live demo

We will show the three executables running on OpenVMS/x86 against PostgreSQL:

CUSTSQL_T86 — pure embedded SQL through sql$pre
CUSTINQ_RDML — RDML rewritten to embedded SQL, then vx/sql$pre
CUSTINQ_LIBCALL — RDML rewritten to library calls, no pre-processor

What you will see

- All three programs connect to the same PostgreSQL database.
- The first two required the pre-compiler; the third did not.
- Switching the libcall executable to Oracle or SQL Server other requires only a different link — no source change.