TECH STYLE LTD

TECH-STYLE.CO

PRODUCT BROCHURE

pgrecon — Product Brochure

Free, open-source Oracle > PostgreSQL migration reconnaissance: 76 rules, effort estimates and offline analysis.

WHAT IS INSIDE

Document	pgrecon brochure
Generated	2 September 2026
Online	https://www.tech-style.co/docs/pgrecon-brochure

Free, open-source migration reconnaissance. It answers the one question that decides whether an Oracle-to-PostgreSQL migration succeeds or overruns: what is actually in there, and what will it cost to move? It parses your PL/SQL with a real grammar, reports every incompatibility with file-and-line evidence, estimates the effort in person-days, and converts the schema it can prove into PostgreSQL DDL — all without ever connecting to your database.

```
$ pip install pgrecon
```

The tables are the easy part

Oracle licensing pushes organisations towards PostgreSQL, and the schema itself rarely stands in the way — tables, indexes and data move with well-trodden tooling. What sinks the timeline is twenty years of business logic: PL/SQL packages full of Oracle-only constructs that have no direct equivalent on the other side.

Migrations routinely run twelve to eighteen months and overrun their budgets for a single reason — nobody measured the code before committing to the date. The scary parts are discovered in month nine, by which point the plan is already written.

pgrecon is the measurement step. It reads the code you already have, tells you exactly which constructs will not survive the move, shows you where each one lives, and turns that into a defensible number. Run it before anyone signs anything.

We built it because we have done these migrations professionally, including for heavy-industry estates still running very old Oracle versions. The rules are the list of things that hurt us.

Status	v0.6.0 on PyPI · alpha
Licence	Apache 2.0 — free commercially
Rules	76, each with fixture tests
Analysis	Fully offline — no DB connection

It never touches your database.
pgrecon generates a SQL*Plus script of plain SELECTs that your own DBA reads and runs. You send back the output files; analysis happens on a laptop, against a local SQLite inventory. There is no listener to open, no account to create and nothing for a security team to sign off.

Parsed, not grepped.
The engine builds a real parse tree of your DDL and PL/SQL. It knows the difference between a dangerous construct in live code and the same word inside a comment or a string literal — which is where keyword-matching tools generate the noise that gets them ignored.

HOW IT WORKS

Four moves: generate, extract, analyse, convert

The only thing that runs near production is a read-only script your own DBA inspects first.

STEP 1

Generate the extraction script

pgrecon script --source-version 19 writes a SQL*Plus script tailored to your Oracle release. For 9.2 to 11.1 estates, --legacy produces a variant that runs on the older dictionary views.

STEP 2

Your DBA reads it, then runs it

It is plain SELECT statements against the data dictionary — nothing to approve in a change board meeting. sqlplus readonly_user@service @pgrecon_extract.sql SCHEMA_NAME produces a directory of dump files. SQL*Plus is the only requirement on the database server; no Python, no agent, no network access.

STEP 3

Load the dump into a local inventory

pgrecon load dump_dir --db inventory.db parses the extract into a local database — every table, index, package, trigger, view, job and dependency, with the source text of each program unit.

STEP 4

Run the rules

pgrecon report --db inventory.db walks the parse tree with all 76 rules and prints every finding: rule ID, severity from info to blocker, the object it lives in, and the evidence. Add --remedies for what to do about each one, or --format json to feed a tracker.

STEP 5

Price the work

pgrecon estimate --db inventory.db converts the inventory and the findings into a low / expected / high range in person-days, and shows the arithmetic that produced it.

STEP 6

Convert what can be proved

pgrecon convert --db inventory.db emits PostgreSQL DDL for everything it can carry faithfully, and a residue file naming every object it declined and why. Still offline, still from the same inventory.

STEP 7

Decide with evidence

You now hold a defensible scope before a single line has been ported — and the same numbers can be re-run against a later extract to show progress.

WHAT IT FINDS

76 rules across nine domains

Every rule ships with fixture tests, a severity, a remediation note and a documented explanation you can read with pgrecon explain RULE_ID.

pgrecon rule categories, counts and representative examples

DOMAIN	RULES	REPRESENTATIVE FINDINGS
PL/SQL code	18	Autonomous transactions, dynamic SQL, FORALL, collection types, the empty-string NULL trap
Schema objects	13	Database links, scheduler jobs, materialised view logs, queues, evolved types, unparseable DDL
Storage	12	Interval partitioning, global temporary tables, IOTs, read-only tables, bitmap and function-based indexes
SQL constructs	12	CONNECT BY, Oracle outer-join syntax, ROWNUM, MERGE, the MODEL clause, PIVOT, flashback queries
Data types	7	LONG, XMLTYPE, ROWID, BFILE, TIMESTAMP WITH LOCAL TIME ZONE
System packages	5	UTL_FILE, UTL_HTTP / SMTP / TCP, DBMS_SQL, DBMS_LOB, DBMS_OUTPUT
Performance	5	Optimiser hints, global indexes on partitioned tables, plan baselines, query-rewrite MVs
Environment	2	Character-set encoding decision, object grants to migrate
Packages	2	Package-level state, initialisation blocks

Evidence, not adjectives

Each finding names the object and the location inside it, so an engineer can open the code and see the problem rather than take the tool's word for it.

Severity that means something

Findings run from info to blocker. A blocker is something with no PostgreSQL equivalent that will require a design decision — not merely a syntax difference.

A remedy per rule

--remedies prints what to do about each finding: the PostgreSQL equivalent, the extension that covers it, or the redesign it forces.

Parse failures are findings

If the parser cannot read a program unit, that is reported as a finding — never silently skipped. Unknown code is a risk, and it is accounted for as one.

Deterministic by design

Same dump in, same findings out. Two people running the same extract get the same report, which is what makes it usable in a commercial conversation.

JSON for the rest of your stack

--format json exports the full finding set for a backlog, a spreadsheet or a dashboard, so the assessment survives the meeting it was made for.

A number that shows its own arithmetic

An estimate nobody can interrogate is worth nothing in a budget meeting. pgrecon estimate builds its range from five components — baseline and environment setup, schema conversion, remediating the findings it just reported, porting PL/SQL by volume, and moving the data — and prints the contribution of each.

The output is a low, expected and high figure in person-days, with the person-month equivalent. Because the workings are visible, your team can argue with any line of it and re-run with their own assumptions instead of discarding the whole thing.

Calibration is the honest caveat: the model is built from real project experience, but it does not know your team, your test estate or your change-control overhead. That judgement is what our assessment report adds.

```
# estimate from the bundled sample inventory pgrecon estimate --db sample.db Components
Baseline & environment Schema conversion Finding remediation PL/SQL porting (by volume)
Data movement Low 105 person-days Expected 129 person-days High 178 person-days (5.0 to
8.5 person-months)
```

It converts what it can prove, and refuses the rest by name

Since v0.2 the same inventory drives a converter. pgrecon convert writes two files, offline like everything else. The first is PostgreSQL DDL for what it can carry faithfully: tables under a documented type mapping (NUMBER stays exact, never a float), keys, checks, foreign keys, secondary indexes, native partition children for range, list, hash and composite layouts, views transpiled with (+) joins folded to ANSI, sequences restarted at their extracted position, synonyms as views, database links scaffolded as oracle_fdw servers, generated columns, and standalone functions and procedures whose every construct has a provably equivalent PL/pgSQL form.

The second file is the residue: one line per declined object, naming the construct and the line number. Packages, triggers, CONNECT BY, autonomous transactions, REF CURSOR interfaces, BULK COLLECT — the work that needs a person is refused by name rather than guessed at, and a routine that calls a refused routine is refused with it.

```
# convert from the same inventory the report came from pgrecon convert --db inventory.db
schema.sql what converts, provably residue.txt what does not, named and located
```

Nothing invalid ships, and nothing is lost silently.
CI applies the bundled sample's conversion to a live PostgreSQL 16 on every commit, with check_function_bodies on. Whatever the converter cannot carry faithfully becomes a named residue line instead of quietly wrong output — so the gap between "converted" and "done" is a file you can count, not a surprise in testing.

COMMANDS

The whole surface area

One command per step, and nothing hidden behind a service.

pgrecon command reference

pgrecon script	Generate the SQL*Plus extraction script. --source-version targets your Oracle release; --legacy produces the 9.2–11.1 variant.
pgrecon load	Parse a dump directory into a local inventory database. --encoding handles dumps that are not UTF-8.
pgrecon report	Run all rules and print the findings. --remedies adds remediation guidance; --format json exports the full set.
pgrecon explain	Print the full documentation for a single rule ID — what it detects, why it matters and how to resolve it.
pgrecon estimate	Produce the low / expected / high effort range in person-days, with the component breakdown.
pgrecon convert	Emit PostgreSQL DDL for everything provably convertible, plus a residue file naming every object it declined and where it lives.

```
# two minutes, no Oracle required — a sample dump ships with the package pip install pgrecon # or run
it against your own estate pgrecon script --source-version 19 # ... your DBA runs the generated script
and returns dump_dir/ pgrecon load dump_dir --db inventory.db pgrecon report --db inventory.db --
remedies pgrecon estimate --db inventory.db pgrecon convert --db inventory.db
```

MIGRATION ASSESSMENT REPORT

The document that gets the budget approved

The free tool answers the engineer's question — what is in there and what breaks. The assessment report answers the board's: what does this cost, how long does it take, in what order, and what could go wrong.

What you receive

- An executive summary a non-technical sponsor can act on
- Every finding prioritised and rewritten in plain English, with business impact
- A phased migration roadmap — what moves first, what moves last, and why
- A testing and cutover plan, including how you prove equivalence
- A cost and effort estimate calibrated against real migration projects, not just the tool's model
- The risk register: what we would worry about in your specific estate
- A walkthrough call with the engineer who wrote it

How an assessment runs

Typical turnaround is two weeks from receiving the dump. The fee is credited against a subsequent migration engagement if you choose to go ahead with us.

Nobody is locked in.
The detection engine is Apache 2.0 and stays that way — you can run it forever, fork it, or hand it to another supplier. What we sell is the judgement layered on top and the hands that do the migration.

STACK

What it runs on

pgrecon technical specification

Runtime	Python 3.11 or later, installed from PyPI with pip install pgrecon.
Source Oracle	11.2 and later with the standard script — tested against Oracle XE 11g and 21c. Oracle 9.2 to 11.1 via pgrecon script --legacy.
Target	PostgreSQL. Rules are written against stock PostgreSQL, noting where an extension covers the gap.

On the database server	SQL*Plus only. No Python, no agent, no outbound connection.
Analysis store	A local single-file inventory database built from the dump.
Analysis method	Grammar-based parsing of DDL and PL/SQL into a parse tree — not keyword matching.
Output	Terminal report with severities and remedies, JSON export, per-rule explanations, effort estimate.
Testing	Every rule carries fixture tests; the suite runs against real Oracle extracts.
Licence	Apache License 2.0. Issues and pull requests welcome.

RELATED

Where this fits

SERVICES

Oracle > PostgreSQL migration

Assessment, schema and PL/SQL conversion, data movement, cutover and support.

SERVICES

Data engineering & BI

What happens to the reporting layer once the database has moved.

OPEN SOURCE

Our other packages

Everything else we publish on PyPI, npm and GitHub.

CONTACT

Talk to us

Tech Style Ltd is a UK technology company building enterprise software, procurement intelligence and AI platforms. Tell us what you are trying to do and we will tell you honestly whether we are the right people for it.

This document online	www.tech-style.co/product-pgrecon
Website	www.tech-style.co
Enquiries	hello@tech-style.co

Products	www.tech-style.co/products
Partner programme	www.tech-style.co/partners

This document was generated on request.
It is built from the live content of tech-style.co at the moment you downloaded it, so it can never quote a price, a version or a status the site has since moved on from. Request a fresh copy any time at www.tech-style.co/downloads.