Oracle ACE

Tech Superstars Unite

Get worldwide recognition as an Oracle ACE

Oracle.com profile page

Swag, certification exam credit & event passes

Exclusive content

Networking events

Your own Oracle cloud account

Travel support

@oracleace

Linkedin.com/groups/1796302

@oracleace.bsky.social

Learn more at:
ace.oracle.com

ORACLE

Let’s Talk APEXLang

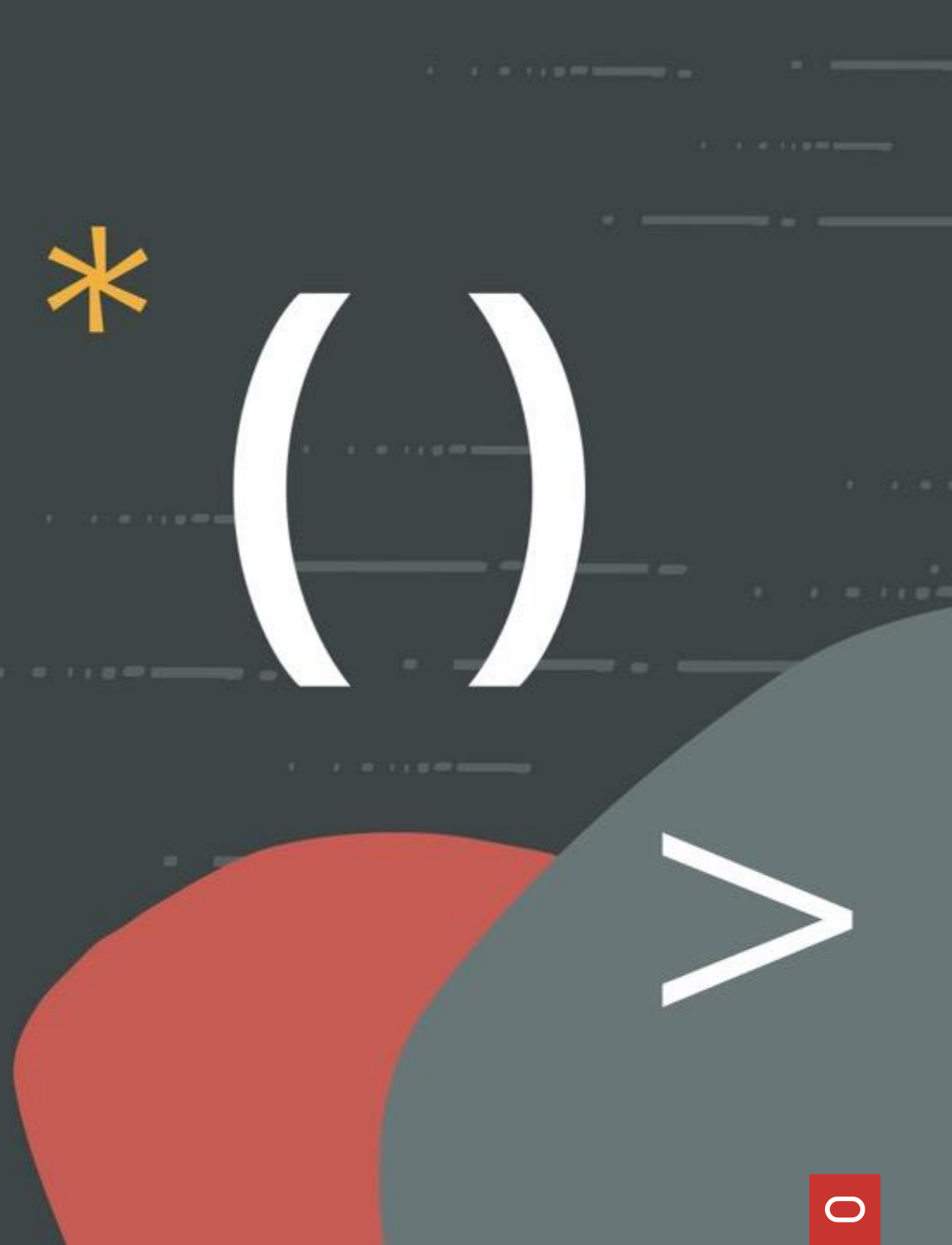
Details from the Inside

Menno Hoogendijk

Consulting Member of Technical Staff

Oracle APEX

April, 2026



Safe harbor statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, timing, and pricing of any features or functionality described for Oracle's products may change and remains at the sole discretion of Oracle Corporation.

Today’s Topics

- Upgrade / Import to 26.1
- APEX Meta-Metadata
- How to use APEXLang
- The APEXLang Project
- What to expect in APEX 26.1

Who puts APEX App Exports under Version Control?

Demo: Upgrade / Import from Previous APEX Release

What's changed?

What Happened?

Export → Import → Export

1. Export from 24.2

- Readable Format (YAML)
- Split into Multiple Files
- Put under Version Control
- **Incorrect App Metadata**

2. Import into 26.1

- SQL imports are still supported
- App is functionally identical

3. Export from 26.1

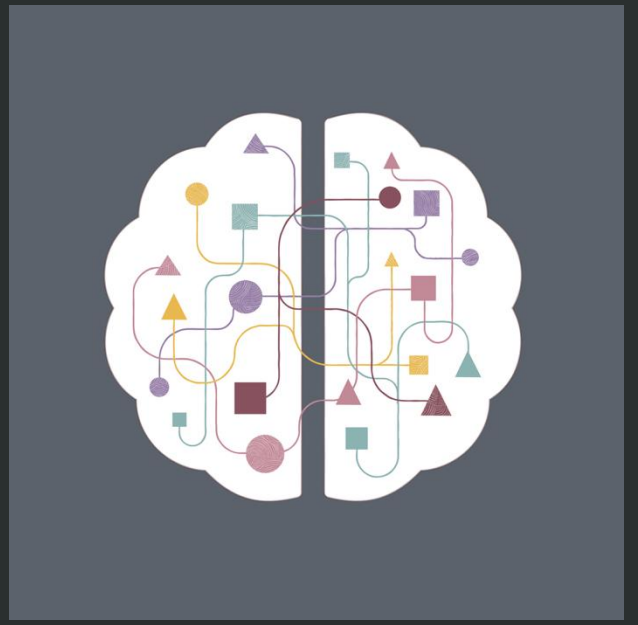
- Many file changes
- New Static IDs everywhere
- **Correct App Metadata**

Why Do We Need to Fix the Metadata?

Metadata vs. Meta-Metadata

A short explanation

What is Metadata?



Describes the Application

The App Functionality: Pages, Shared Components, Look & Feel, etc..

Provided by Developer

Using Page Designer and Application Builder, you provide values for the **components** and their **properties**

Required Data for App Exports

Based on the Metadata, APEX can import your Apps

What is Meta-Metadata?

{MMD}

Components

What types of components are available (e.g., page, region, button, page items, list of values, workflow, etc.)?

Relationships

How are these components related (e.g., parent/child relationships)?

Attributes

What attributes can be set for each component type?

Visibility Conditions

Under what conditions is an attribute visible?

What is Meta-Metadata?

{MMD}

Default Values

What is the default value of an attribute if it is not specified?

Data Types and Allowed Values

What is the data type of each attribute and what values are allowed?

Additional Rules

Are there any additional rules, such as whether an attribute is required, if its value must be uppercase, if it should allow only certain characters, or if it must be unique within its parent or page?

How much Meta-Metadata has APEX?

+230

Unique Components

+2500

Unique Attributes

+6500

Attributes used by Components

Meta-Metadata Validates Your Metadata

When is it Validated?

Before APEX 26.1

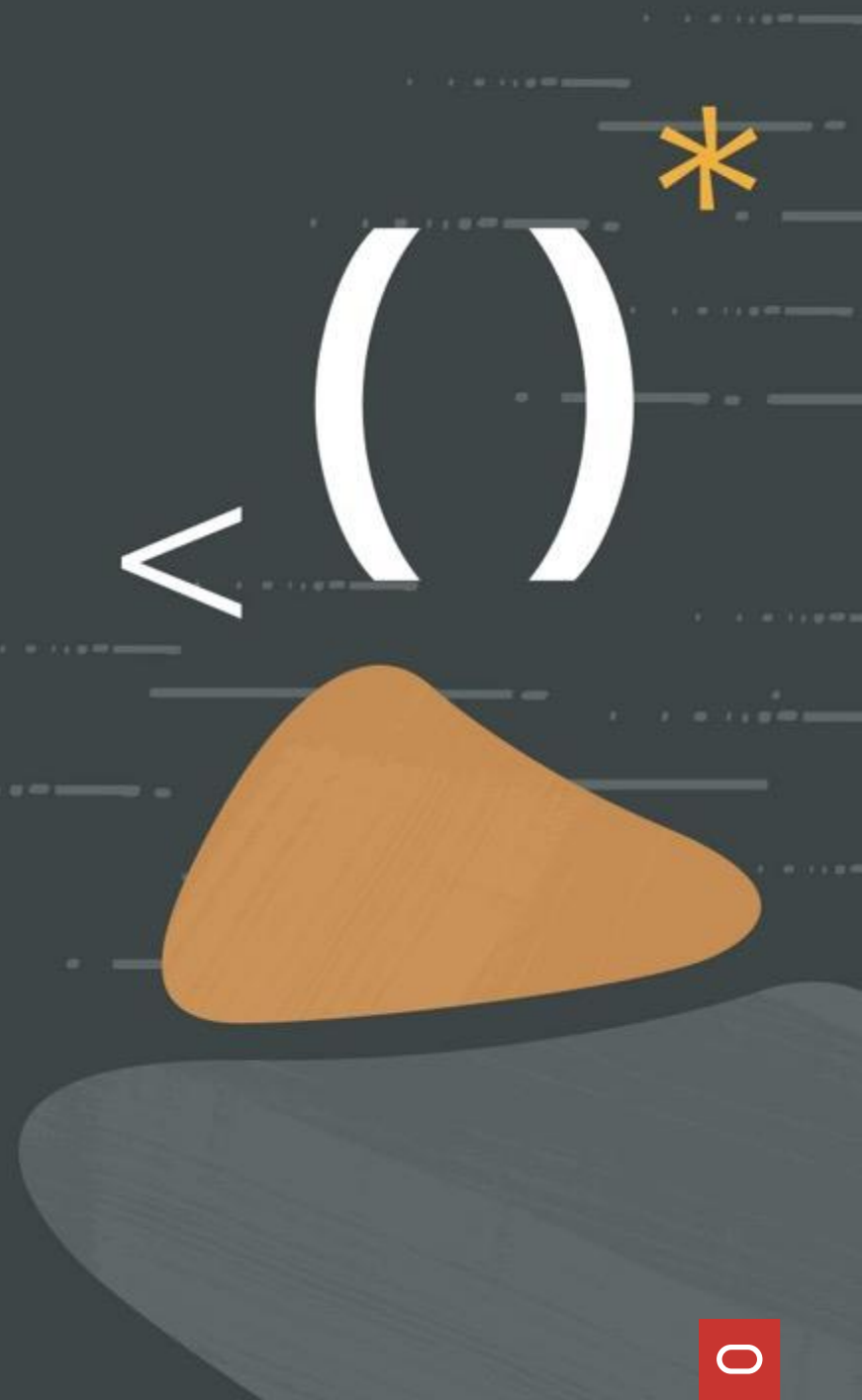
- Making Changes in the Application Builder

From APEX 26.1

- Making Changes in the Application Builder
- While Editing APEXLang in VS Code with the Oracle SQL Developer Extension
- Before Importing APEXLang exports

The Meta-Metadata is now needed outside APEX

Demo: APEXLang



SQLcl Commands

Command	Description
`help apex`	Shows help about all commands and options
`apex list`	List all APEX apps in your schema
`apex export`	Exports an APEX app to SQL or APEXLang
`apex import`	Imports APEXLang or SQL input
`apex validate (new)`	Validates that APEXLang input can be compiled
`apex generate (new)`	Generates an empty starter app

From APEXLang to Database

SQLcl now includes an APEXLang Compiler

APEX does not import APEXLang directly.

Compiles APEXLang to an Artifact (SQL)

The compiler uses the same Meta-Metadata to generate valid and importable artifacts.

Using mostly the existing Import PL/SQL APIs

The import logic in APEX did not change that much.

Upgrade of APEX = Upgrade of SQLcl and ORDS

No need to keep older SQLcl and ORDS versions for different APEX versions. Use the latest and greatest.

APEXLang – Syntax

Looks like YAML but isn’t

Component Syntax

```
<component_type> <staticID> (
    <properties>
    <groups>
    <child_components>
)
```

```
column ENAME (
    reportColumnQueryId: 2
    derivedColumn: N
    heading {
        heading: Employee Name
    }
    layout {
        sequence: 2
    }
    sorting {
        defaultSequence: 1
    }
)
```

Group Syntax

```
<group name> {
    <properties>
}
```

```
column ENAME (
    reportColumnQueryId: 2
    derivedColumn: N
    heading {
        heading: Employee Name
    }
    layout {
        sequence: 2
    }
    sorting {
        defaultSequence: 1
    }
)
```

Multi-value Property Syntax

```
<multivalue_property_type>: [
    <value_1>
    <value_n>
]
```

```
region employees (
    name: Employees
    type: classicReport
    source {
        location: localDatabase
        tableName: EMP
    }
    componentAppearance {
        template: @/standard
        templateOptions: [
            #DEFAULT#
            t-Report--stretch
            t-Report--staticRowColors
            t-Report--rowHighlight
            t-Report--inline
            t-Report--hideNoPagination
        ]
    }
)
```

Complex Property Syntax

```
<complex_property_type>: {
    <property_type>: <value>
}
```

```
link {
    target: f?p=&APP_ID.:1:&APP_SESSION.:MYREQUEST:LEVEL9:RR::
    linkText: #DUMMY#
}
```

Vs.

```
link {
    target: {
        page: 1
        action: resetRegions
        request: MYREQUEST
        debug: LEVEL9
    }
    linkText: #DUMMY#
}
```

Other Language Rules

- New lines are required
- Indentation is not required (unlike YAML)
- Blank lines are allowed
- Shared Components are referenced via @
- Supports Comments via // prefix

Demo: APEXLang Syntax

APEXLang – A Huge Cross Team Effort

What made it so complex?

Phase 1 – The Architecture

Clarify the goal

We want to be able to export apps into a readable file format and allow developers to edit the files and import it.

Identify Components

Export: APEX, SQLcl, and SQL Developer
Import: APEX, ORDS, SQLcl, and SQL Developer, Compiler
Editing: Grammar checks with SQL Developer

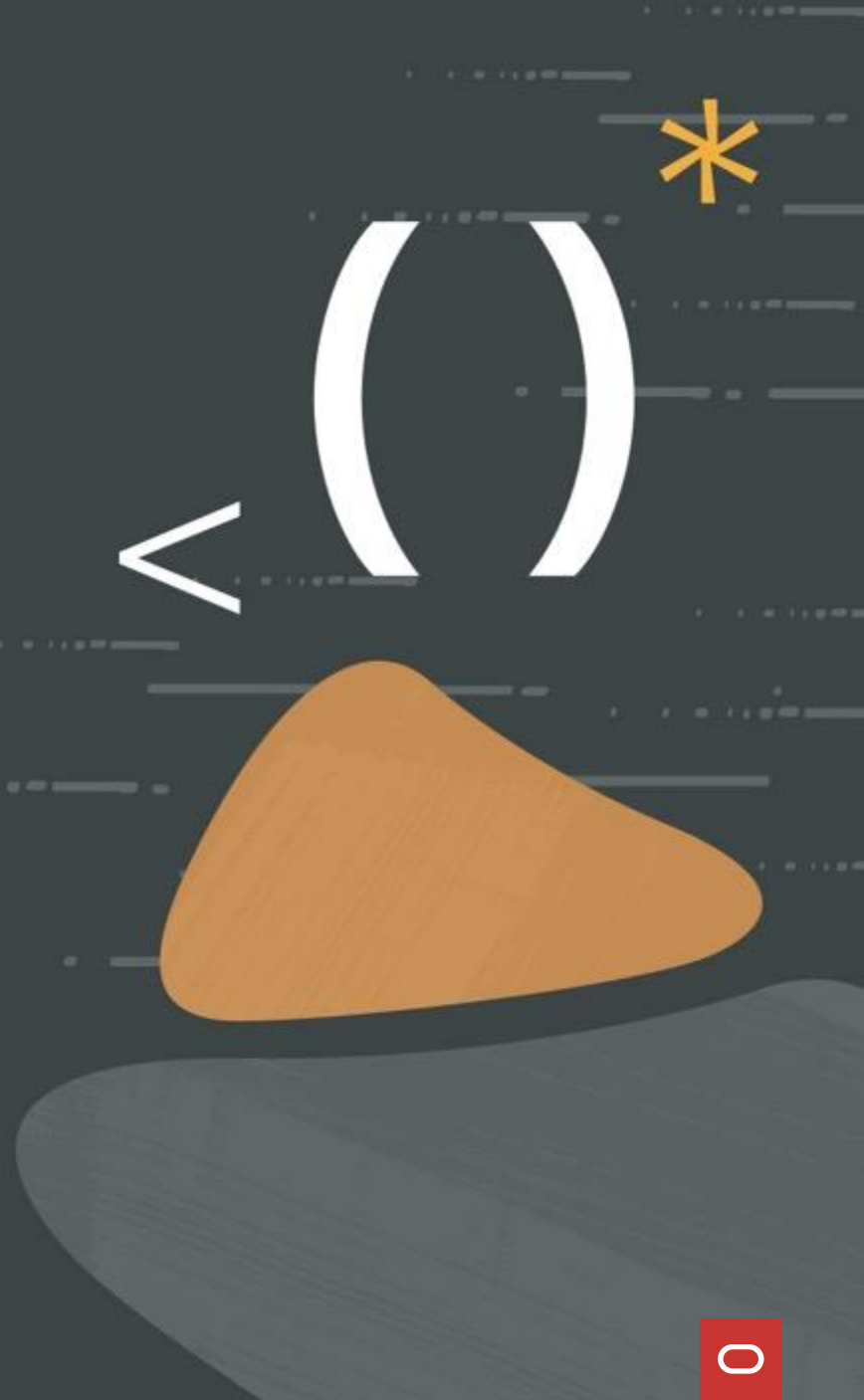
Identify required Changes in APEX

Many projects for many APEX Team Members

Phase 2 - Many APEX Projects by Many Team Members

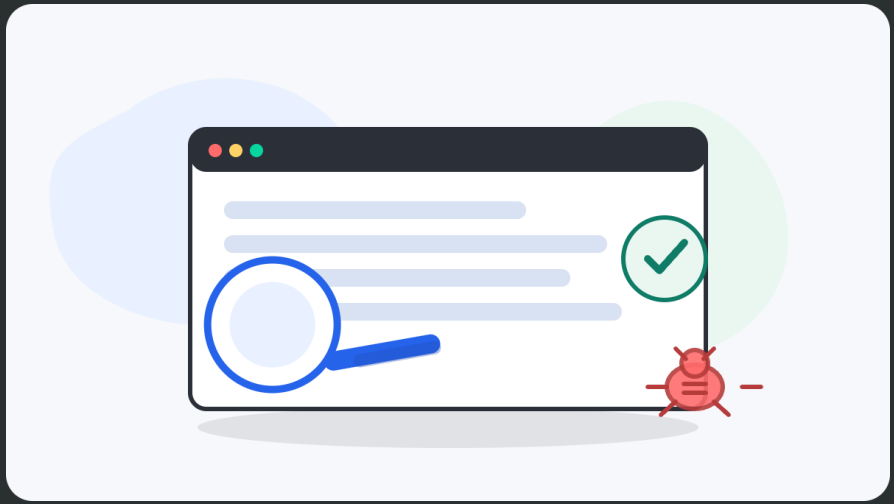
- Unlimited attributes for all Plug-In Types
- Readable Static IDs for all our Component Types
- Cleanup of Metadata during pre 26.1 App Imports
- Support APEXLang Exports
- APEXLang View in Page Designer
- Many, many more...

Demo: Changes in APEX

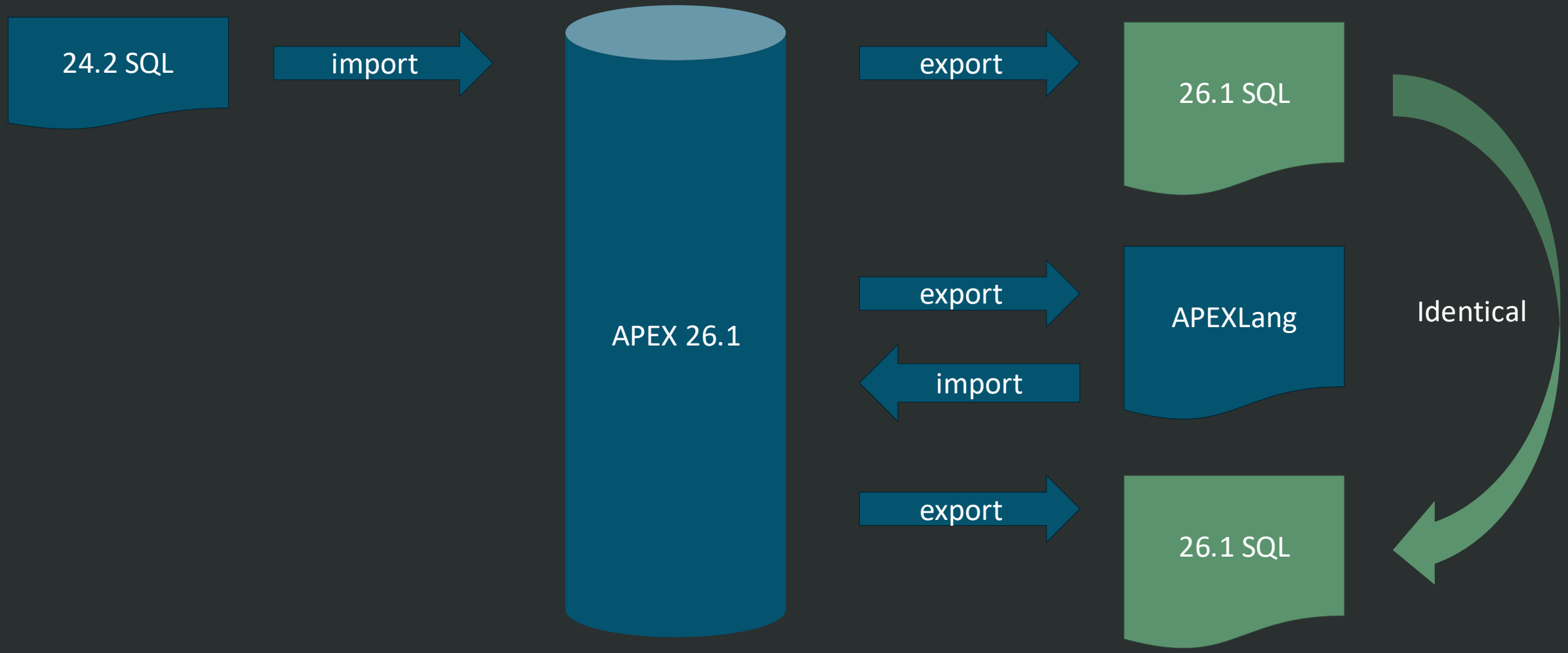


Phase 3 – Testing, Testing, Testing

- Unit Tests
- Roundtrip tests
  - We have access to ~50.000 apps for testing
  - Ensure all apps can be imported
  - Ensure no difference in Metadata between APEXLang import and SQL import
  - +25 Year Old Apps with corrupt Metadata
  - Many tickets filed to fix issues



Phase 3 – Roundtrip Testing



Phase 4 – The Release of APEX 26.1

Limitations

- No APEXLang import from the browser
- No APEXLang import of individual components / pages

The APEX Team will keep improving APEXLang.

Phase 4 – The Release of APEX 26.1

- ✓ Application Builder vs. File Based Development
- ✓ Version Control Friendly
- ✓ AI Friendly
- ✓ Search & Replace Friendly

Your feedback helps us grow!

Final Thoughts

- APEXLang is extremely Low-Code
- The Easier for You, the Harder for Us
- Project Requires 100% dedication for everyone involved

Thank you


ORACLE