AQL dialect + syntax checker

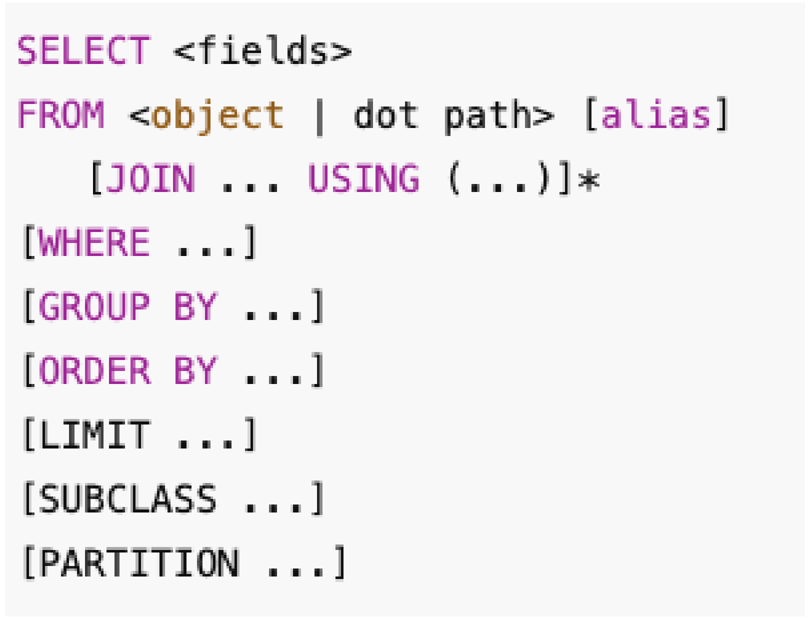
Code	Type	Description
AQL001	Syntax	Unexpected token (e.g., wrong keyword order, missing `FROM`).
AQL002	Syntax	Missing required clause (`SELECT`, `FROM`, etc.).
AQL005	Syntax	Malformed parameter (missing name after `:` or invalid characters).
AQL006	Syntax	Invalid `LIMIT` — must be a non-negative integer literal.
AQL011	Semantic	Unknown **USING** field — doesn't exist in both joined **classes**.
AQL012	Semantic	Ambiguous **field** reference — unqualified name appears in multiple joined **classes**.

Code	Type	Description
AQL013	Semantic	Duplicate **alias** — same alias defined for more than one **class**.
AQL014	Syntax	Empty `USING()` list — must specify at least one **field**.
AQL015	Syntax	Invalid `JOIN` form — only `JOIN … USING (…)` allowed in this dialect.
AQL020	Semantic	Dot-notation inside `OR` — forbidden because dot-notation implies inner auto-joins.
AQL021	Semantic	Type mismatch — incompatible operands (checked only if type info available).

Code	Type	Description
AQL022	Syntax	Reserved word used unquoted as a **class** or **field** name (e.g., `User`).
AQL023	Syntax	Invalid parameter token — malformed `:param` or unknown shorthand.
AQL024	Semantic	Unknown **field** — not resolvable within current **class** or its `SUBCLASS` scope.
AQL025	Semantic	Ambiguous **field** — same field name visible from multiple **classes** (not a USING key).

AQL001/AQL002 Core Query Shape

A valid query looks like:



```
SELECT <fields>
FROM <object | dot path> [alias]
   [JOIN ... USING (...)]*
[WHERE ...]
[GROUP BY ...]
[ORDER BY ...]
[LIMIT ...]
[SUBCLASS ...]
[PARTITION ...]
```

Checker rule:

- Must have SELECT and FROM.
- Clauses must appear in the correct order.
- No duplicates of the same clause type (e.g., two WHEREs).

AQL005

- AQL parameters always start with a colon (:) followed by a valid name (letters, digits, or underscore).

```
SELECT Name
FROM Supplier
WHERE Country = :countryParam
```

Incorrect or incomplete parameter examples (AQL005)

Example	What's wrong	Fix
`WHERE Country = :`	The colon has **no name** after it.	→ `WHERE Country = :countryParam`
`WHERE Price > :123abc`	Parameter names **cannot start with a number**.	→ `WHERE Price > :priceValue`
`WHERE Status = :@param`	Invalid character (@) in parameter name.	→ `WHERE Status = :statusParam`
`WHERE Region = ::region`	Double colons not allowed.	→ `WHERE Region = :region`
`WHERE Code = :region-name`	Hyphen not allowed in parameter names.	→ `WHERE Code = :regionName`

Incorrect usage (AQL006 errors)

Example	What’s wrong	Fix
`LIMIT -10`	Negative numbers are **not allowed**.	→ `LIMIT 10`
`LIMIT 10.5`	Decimal numbers **not supported** — must be an integer.	→ `LIMIT 10`
`LIMIT '100'`	The limit is a **string**, not a number.	→ `LIMIT 100`
`LIMIT :maxRows`	Parameters are **not allowed** here (AQL requires a literal).	→ `LIMIT 500`
`LIMIT` *(no number)*	Missing value — incomplete syntax.	→ `LIMIT 100`

Explicit Joins (JOIN … USING (…)) AQL011/ AQL015

Supported form

- Only **inner** joins in this tiny dialect:
- From <ClassName> [alias]
- JOIN <ClassName> [alias] USING (field1 [, field2, …])
- [JOIN …]*

Continue

- USING must list ≥1 **field (**AQL014**)**.
- With schema: each USING **field** must exist on **both classes**.
- After the join, each USING field appears **once** in the visible field set.
- Any other **field** name that exists in multiple joined **classes** is **ambiguous** if unqualified → require alias.Field (AQL012).
- **Aliases** must be unique per **class** reference (AQL013).

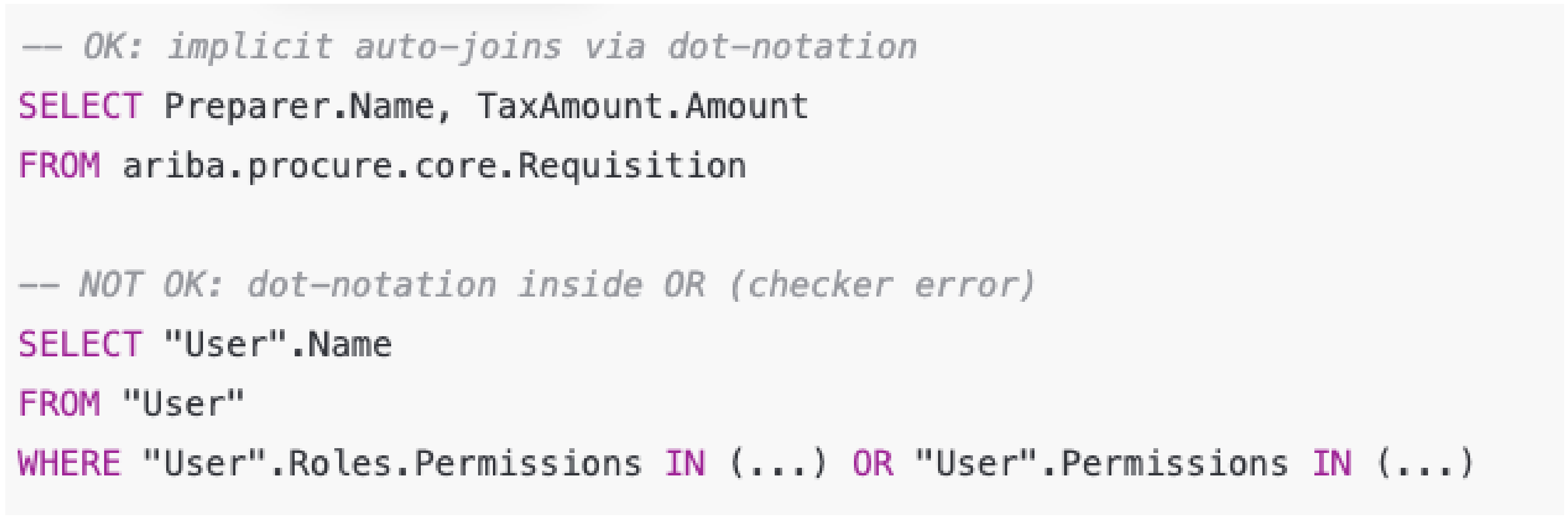
Dot-Notation Logic (Auto Joins) (AQL020)

- Writing a dotted path like Preparer.Name or "User".Roles.Permissions tells AQL to **follow relationships** between objects.
- AQL automatically brings in the related objects behind the scenes—this is an **implicit inner auto-join** (always inner when created by dot-notation).
- **Where we can use it:** SELECT, WHERE, and ORDER BY may reference dotted paths. It's just normal AQL—no explicit join keywords needed for these paths.
- **Hard rule with OR:** if a WHERE expression uses OR, **We must not use dot-notation** anywhere inside that OR tree. Because dot-notation implies inner presence of those related objects, combining them with OR will not behave like a true "either/or."

- Reserved words: AQL reserved identifiers must be quoted. For example, always write "User" (not User). Dotted segments that are reserved should also be quoted as needed: "User".Roles.Permissions.

What the checker does (AQL logic only):

- Detects dotted paths and records an **implicit inner auto-join** for each path.
- Blocks any **dot-notation within OR** conditions.
- Ensures reserved words are **properly quoted** in any segment.



```
-- OK: implicit auto-joins via dot-notation
SELECT Preparer.Name, TaxAmount.Amount
FROM ariba.procure.core.Requisition

-- NOT OK: dot-notation inside OR (checker error)
SELECT "User".Name
FROM "User"
WHERE "User".Roles.Permissions IN (...) OR "User".Permissions IN (...)
```

WHERE Clause Logic

Expression forms

- Comparisons: =, !=, <, <=, >, >=
- Null checks: IS NULL, IS NOT NULL
- Set/range: IN (...), BETWEEN ... AND ...
- Pattern: LIKE 'pattern' (with escapes)
- Boolean: AND, OR, NOT, parentheses (...)
- Operands: FieldRef (can be dotted), literals, parameters (:name, :NUM)

Reserved Words & Identifiers