OCPQ Cheat Sheet

Object-Centric Process Querying.
A declarative query language for OCEL 2.0.

WHAT IS OCPQ?
OCPQ lets you query **object-centric event logs** (OCELs) for structural and temporal patterns, like *orders that received ≥ 3 reminders before being paid*. You sketch the pattern as a nested tree of typed variables and predicates; OCPQ returns every matching binding, and any added **constraints** additionally mark each binding as **satisfied** or **violated**.

INPUT

OCEL 2.0

OCPQ queries **Object-Centric Event Logs**.

Events	id · type · timestamp · attributes
Objects	id · type · attributes⁺
	⁺ *object attributes can change over time.*
Relations	Link events to objects (**E2O**) and objects to other objects (**O2O**).
E2O	event → object (qualified)
O2O	object → object (qualified)

No traces: One event may involve many objects.

THE UNIT OF QUERY

Anatomy of a query node

Five sections per node. The + button beside each section header adds a new variable, filter, label, or constraint.

Object Variables + #7659
o1: orders
o2: items

Event Variables +
e1: place order
e2: package delivered

Filters +
e1 🔗 o1
o1 🔗 o2
e2 🔗 o2

Labels +
price `o1.attr('price')`

Constraints +
e1 → e2 `0 - 2w` ⚠ 12.64% 968

Obj. Variables	typed *object* placeholders
Evt. Variables	typed *event* placeholders
Filters	specify which variable combinations to consider
Labels	values (e.g., KPIs) computed for each binding
Constraints	mark each binding satisfied / violated

A **binding** assigns each variable to a concrete event/object. Filters and Constraints are both **predicates** that evaluate to `true` or `false` for a given binding: **filters** determine which bindings are returned at all, while **constraints** mark bindings as *satisfied* or *violated* without excluding them.

After evaluating with ▷, the node shows #7659 (number of bindings, top-right) and, if constraints are present, ⚠ % of violated bindings.

DECLARATION

Variables & bindings

Variables are **typed placeholders**. They consist of a name (e.g., o1) and allowed types (e.g., orders).

Binding

A map from each variable to a concrete OCEL value, respecting types.

$$b = \{o1 \mapsto \text{Max}, o2 \mapsto \text{order132}, e1 \mapsto \text{po164}\}$$

Bindings table

Each node's output = set of bindings = table with one column per variable.

o1	o2	e1
Max	order132	po164
Max	order643	po712
Sophia	order728	po754

CONNECT VIA OCEL

Relationship predicates

Holds exactly when the **concrete events/objects** bound to the variables are in the asserted **E2O / O2O relationship** in the OCEL. Tool shows link icons; formal form uses predicate notation.

E2O: Event-to-Object

TOOL e1 🔗 o1 | e1 🔗 o1 @q
FORMAL `E2O(e1, o1)` | `E2O(e1, o1, q)`

O2O: Object-to-Object

Directed: holds for o1 → o2, not the reverse.

TOOL o1 🔗 o2 | o1 🔗 o2 @q
FORMAL `O2O(o1, o2)` | `O2O(o1, o2, q)`

Qualifiers

Disambiguate roles, e.g. `q = "primary"` vs `q = "secondary"` sales rep.

TIME, RANGES, EQUALITY

Time & attribute predicates

TBE: Time between events

`TBE(e, e', min, max)` holds if the duration from e to e' lies in $[\text{min}, \text{max}]$.

min > 0 ⇒ durations e' strictly after e
1h, 2d, -30min, ∞

OAR / EAR: Object/Event Attributes

`OAR(o, attr, cond)` · object attribute, e.g. o1.price between 100 and 200.

`EAR(e, attr, cond)` · event attribute, e.g. e1.mode = "online".

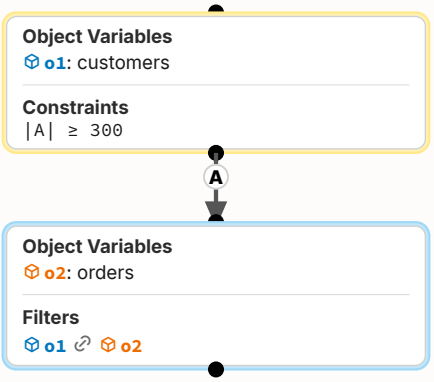
OCEL 2.0: object attributes change over time.
Set the temporal scope on each OAR:

`sometime`	holds at ≥ 1 value
`always`	holds at every value
`at event x`	at the time of a given event

STRUCTURE

Nested tree & child bindings

Queries are **trees**. Each node can have multiple child nodes, by default labeled as A, B, C, etc. Variables and predicates are propagated down the tree (e.g., o1 of parent can be used in child).



Child binding set A_b

For each parent binding b, A_b = child bindings of **A** that extend b.

o1	o2
Max	order132
Sophia	order421
Sophia	order312

The parent binding $b = \{o1 \mapsto \text{Sophia}\}$ has 2 child bindings under A, so $|A_b| = 2$.

The set of child bindings under edge A per parent binding is denoted by A and used for advanced predicates that use subqueries (e.g., $|A| \geq 3$).

ADVANCED PREDICATES

Predicate quick reference

Child set predicates

CBS: constrain the **number of bindings** under a child edge to lie in `[min, max]`. Either bound may be omitted.

TOOL `|A| ≥ 3`
FORMAL `CBS(A, min, max)`

CBPS: like CBS, but counts the **distinct values of variable v** across the A-bindings (a projection) instead of all bindings.

TOOL `|A[o2]| ≥ 3`
FORMAL `CBPS(A, v, min, max)`

CBE: require two or more child binding sets to be **identical as sets**.

TOOL `A = B`
FORMAL `CBE(A, B, …)`

CBPE: like CBE, but compare only on chosen variables (equal **projections**, not necessarily equal full bindings).

TOOL `A[o2] = B[o3]`
FORMAL `CBPE(A[v], B[v'], …)`

Common Expression Language

CEL: evaluate a boolean `expr` on the current variables. Use for any condition that doesn't fit the dedicated predicates.

TOOL `e1.attr('mode') == 'online'`
FORMAL `CEL(expr)`

CEL+: same as CEL, but the child binding sets `A,B,…` and node labels are also in scope. Enables aggregations across children.

TOOL `size(A) >= 2`
FORMAL `CEL+(expr)`

COMPUTED PER BINDING

Labels (CEL)

Attach named values to each binding. Labels are written in **Common Expression Language** (CEL).

Three canonical labels

Count a child set
num_orders `size(A)`
Aggregate child attributes
volume `A.map(b, b['o2'].attr('price')).sum()`
Read an attribute
p `o1.attr('price')`

Labels become extra columns in the output table and may be referenced by constraints (e.g. `volume ≥ 200000`).

See: `ocpq.aarkue.eu/docs/labeling-guide` · CEL reference `cel.dev`

AGGREGATING CHILDREN

Logic gates

Used in a parent's **Constraints** section to combine the satisfied / violated status of its child bindings into the parent's verdict.

SAT(A) A ✓ ✓ ✓ ⇒ **satisfied** *all child bindings for **A** are satisfied*
ANY(A) A ✗ ✓ ✗ ⇒ **satisfied** *at least one child binding for **A** is satisfied*
NOT(A) A ✗ ✗ ✗ ⇒ **satisfied** *no child binding for **A** is satisfied*
AND(A,B,…) A ✓ ✓ B ✓ ✓ ⇒ **satisfied** *all child bindings under **A** and **B** hold*
OR(A,B,…) A ✓ ✗ B ✓ ✓ ⇒ **satisfied** *all child bindings under **A** hold, or all under **B** hold*

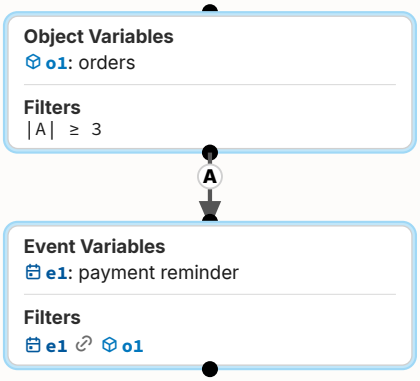
AND and OR accept any number of child edges.

LEARNING BY EXAMPLE

Example Queries

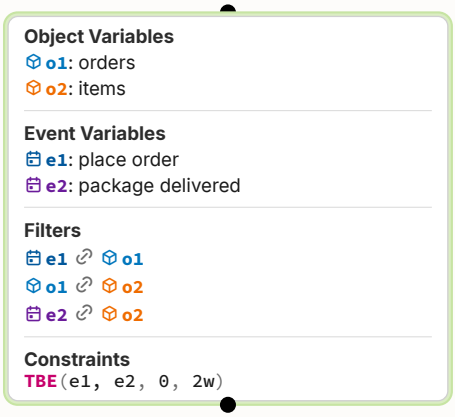
№ 1 Orders with ≥ 3 payment reminders

Child **A** gathers reminder events per order; filter the parent down to orders with at least three. The classic count-and-threshold pattern.



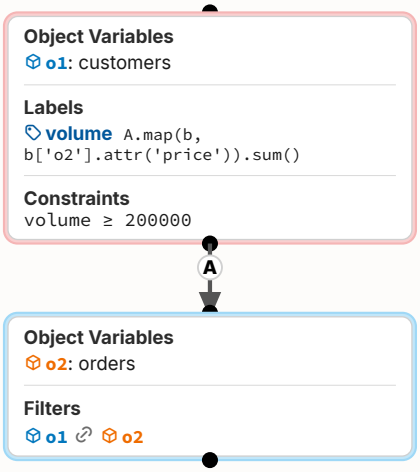
№ 2 Items delivered within 2 weeks

Link place-order to the order and package-delivered to the item via E2O / O2O. The TBE constraint flags bindings where delivery is later than 2 weeks after placement.



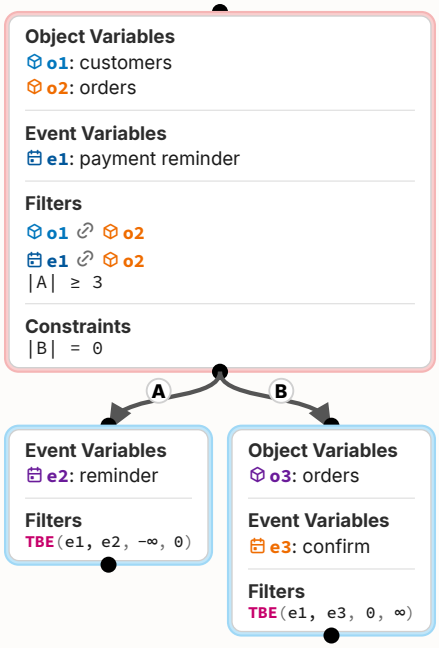
№ 3 Customers with volume ≥ 200k

Child **A** collects each customer's orders; a CEL label sums the prices; a CEL+ constraint flags customers below the threshold. Pattern for any KPI aggregation.



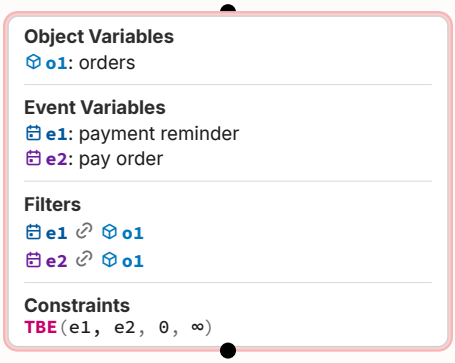
№ 4 After 3rd reminder: no later confirm

Parent fixes a reminder `e1`. Child **A** holds earlier reminders on the same order (A includes `e1` itself, since TBE upper bound is `0`); child **B** later confirms on a sibling order. Filter `|A| ≥ 3` keeps reminders that are the 3rd or later; constraint `|B| = 0` flags any later confirm.



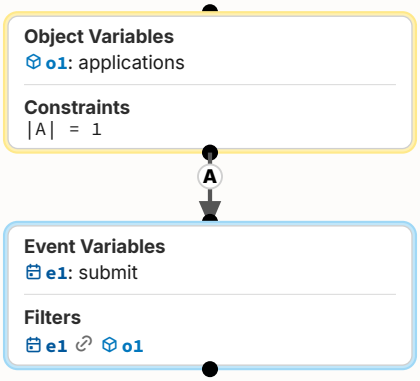
№ 5 No reminder after payment

For every (order, reminder, pay) triple, TBE forces a non-negative duration from reminder to pay. The "X must precede Y" pattern in a single node.



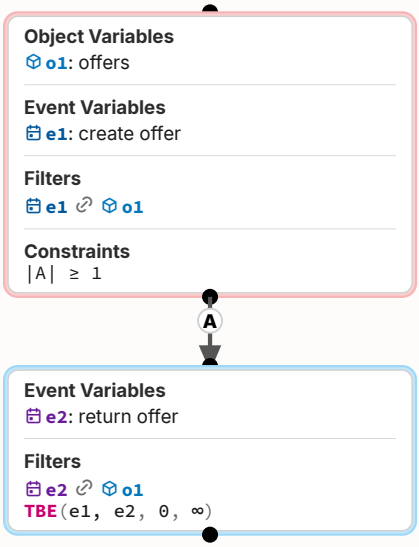
№ 6 Each application submitted exactly once

Child **A** collects every submit event for the application. `|A| = 1` flags applications never submitted or submitted more than once. The exact-cardinality variant.



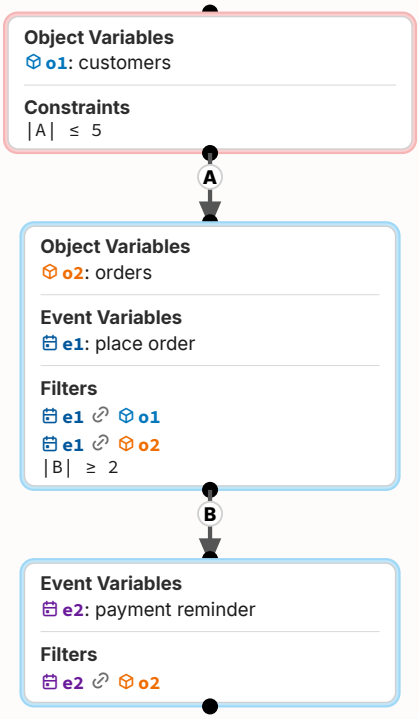
№ 7 Offers must eventually be returned

Parent fixes a creation event; child **A** gathers later returns on the same offer (TBE `[0, ∞]` enforces "at or after"). `|A| ≥ 1` flags offers never returned. The "X must eventually happen" pattern.



№ 8 Problematic customers

Three-level nesting. The middle node's size-filter `|B| ≥ 2` keeps only orders with 2+ reminders, so child **A** counts "problematic orders". Top constraint `|A| ≤ 5` flags customers exceeding that number: counts cascade through nesting levels.



Learn more · Download · Get in touch

Docs & guides: `ocpq.aarkue.eu/docs` ·
Pattern library (copy + paste ready): `ocpq.aarkue.eu/docs/examples`
Download (Windows · macOS · Linux): `ocpq.aarkue.eu/download`
Collaboration, feedback, feature requests: write to `kuesters@pads.rwth-aachen.de` or open an issue at `github.com/aarkue/OCPQ/issues`

RESOURCES

Docs: `ocpq.aarkue.eu/docs` · **Download:** `ocpq.aarkue.eu/download` · **Source Code:** `github.com/aarkue/OCPQ`

Paper: *OCPQ: Object-Centric Process Querying and Constraints.* Küsters & van der Aalst, RCIS 2025. `doi.org/10.1007/978-3-031-92474-3_23`

Demo: *Easy and Efficient Object-Centric Process Querying with the OCPQ Tool.* Küsters & van der Aalst, BPM 2025 (Demos).

by **Aaron Küsters**

v1.0
Latest: `ocpq.aarkue.eu/cheat-sheet.pdf`