PGConf.BE 2026

contentgrid BY AMEXIO

Scaling Semantic Models

Bespoke PostgreSQL Schemas at SaaS Scale

Thijs Lemmens · May 2026

About Xenit & Amexio Group

Alfresco ECM Service Provider

Long-time Alfresco implementation partner — now building and offering ContentGrid as our own cloud-native ECM product.

European Digital Solutions Group

Pan-European group specialising in ECM, digital transformation, and intelligent information management.

Xenit was acquired by Amexio Group in April 2025 — bringing ContentGrid into a broader portfolio of ECM expertise across Europe.

Agenda

1. ECM in 60 seconds
2. The problem with EAV / generic data models
3. Intro to ContentGrid
4. Attribute based access control
5. Search
6. Conclusions

contentgrid
BY AMEXIO

ECM in 60 seconds

Documents, metadata, and the customers who keep changing both

contentgrid
BY AMEXIO

What ECM Systems Actually Do

WHERE YOU FIND ECM

- **Insurance** — claim files: correspondence, expert reports, photos, payouts
- **Banking** — contracts, statements
- **HR** — employee files, contracts, reviews
- **Legal · engineering · government** — case files, drawings, citizen records

WHAT THE SYSTEM HAS TO DO

- Store documents alongside their structured metadata
- **Search** — by metadata filters and full-text content (the primary user action)
- Access control per document
- Retention, versioning, audit — data outlives the apps
- Volume: millions to billions of objects

The metadata is the database problem. Blob storage is the easy part.

contentgrid
BY AMEXIO

The Flexibility Trade-off

Each customer defines their own document types — `claim`, `policy_endorsement`, `expert_report` — and they keep changing. A product serving thousands of clients has to absorb that variability somewhere. There are two honest places to put it.

OPTION 1

One generic schema for every customer

+ One model fits any customer's types
+ No DDL when types change
− Database can't see types or constraints
− Indexes and statistics fight every query

OPTION 2

A bespoke schema per customer

+ Native columns, types, indexes
+ Constraints enforced by the database
− Thousands of schemas to provision and migrate
− DDL has to be automated, versioned, and safe

Both are legitimate engineering choices. The rest of the talk is about making the second one work.

How Other ECMs Store Metadata

The market splits across five patterns

Approach	Systems	Per-document cost
EAV / property rows	Alfresco · OnBase (one table per keyword type)	**N joins for N attributes**
Wide allocated table	SharePoint (`AllUserData` sparse) · FileNet (`DocVersion`, columns added via `ALTER`)	1 row, but rigid slots / NULL noise
Table-per-type	Documentum — `<type>_s` / `<type>_r` joined up the inheritance chain	Joins per inheritance level
Schema mapping	Nuxeo VCS — XSD schema → tables, complex types → child tables via FK	Native columns, DDL frozen at type design
Document store	Nuxeo DBS — MongoDB / MarkLogic	No SQL · no constraints · no joins

The Problem

Why generic EAV tables fall short

contentgrid
BY AMEXIO

The EAV Trap

How Alfresco — and many ECM systems — store content

Generic EAV schema

```
CREATE TABLE alf_node (
  id      bigint PRIMARY KEY,
  type_qname_id bigint REFERENCES alf_qname
);
CREATE TABLE alf_qname (
  id        bigint PRIMARY KEY,
  local_name varchar(200)
);
CREATE TABLE alf_node_properties (
  node_id       bigint REFERENCES alf_node,
  qname_id      bigint REFERENCES alf_qname,
  string_value  varchar(1024),
  boolean_value bit,
  int_value     int8,
  double_value  float8,
  ...
);
```

What you actually want

```
CREATE TABLE articles (
  id           uuid PRIMARY KEY,
  title        text        NOT NULL,
  published_at timestamptz,
  word_count   integer,
  author_id    uuid REFERENCES authors
);
```

contentgrid BY AMEXIO

One Document in EAV

Storing a single article takes rows across 3 tables

alf_node — 1 row

ID	TYPE_QNAME_ID
42	7

alf_qname — excerpt

ID	LOCAL_NAME
1	title
2	published_at
3	word_count
4	author_id
7	article

alf_node_properties — 4 rows (the actual values)

NODE_ID	QNAME_ID	STRING_VALUE	BOOLEAN_VALUE	INT_VALUE	DOUBLE_VALUE
42	1	**Getting Started with PostgreSQL**	*NULL*	*NULL*	*NULL*
42	2	**2024-03-15T09:00:00Z**	*NULL*	*NULL*	*NULL*
42	3	*NULL*	*NULL*	**1842**	*NULL*
42	4	**usr_abc123**	*NULL*	*NULL*	*NULL*

Same Document, Bespoke Table

One row. Every column typed and named.

articles — 1 row

id uuid	title text	published_at timestamptz	word_count integer	author_id uuid
42	Getting Started with PostgreSQL	2024-03-15 09:00:00+00	1842	usr_abc123

1 row · 0 NULLs · every value in the right type

contentgrid BY AMEXIO

EAV: The Hidden Costs (1/2)

Data integrity

Concern	EAV	Native schema
Type safety	One column per data type	Enforced by the column type
Constraints	`NOT NULL`, `UNIQUE` cannot be expressed	Declarative, DB-enforced
Referential integrity	Application-enforced	Foreign keys

EAV: The Hidden Costs (2/2)

Performance & tooling

Concern	EAV	Native schema
Query complexity	Reconstruct one node via dozens of `JOIN`s	Plain `SELECT`
Query planning	Value-column statistics mix all property types — per-attribute selectivity is invisible to the planner	Accurate per-column statistics
Index efficiency	Index on `(qname_id, string_value)` covers all attributes of type string	Targeted per-column indexes
Tooling compatibility	Breaks ORMs, analytics tools	Works out of the box
Search performance	Sync to Solr/Elasticsearch — operational heavy, denormalizes relations, eventual consistency	Native full-text search, joins, transactional consistency

> *EAV trades correctness and performance for schema flexibility — but you can have both.*

How Bad Is It? A Real Migration

The scenario

- **250 million** documents
- **60 attributes** each
- EAV model on Oracle

How large is the Oracle database?

A	< 500 GB
B	500 GB – 2 TB
C	2 TB – 5 TB
D	> 5 TB

contentgrid BY AMEXIO

The EAV Oracle database: 6 TB

- 15 billion property rows, each with 6+ typed value columns
- Indexes on `(qname_id, *_value)` for every attribute type

> *Now — same data, native PostgreSQL schema. Your guess?*

contentgrid
BY AMEXIO

Same Data, Native PostgreSQL Schema

What changes

- One row per document
- Typed columns — no attribute explosion

How large is the PostgreSQL database?

A	Still ~6 TB
B	1 TB – 3 TB
C	250 GB – 1 TB
D	< 250 GB

contentgrid
BY AMEXIO

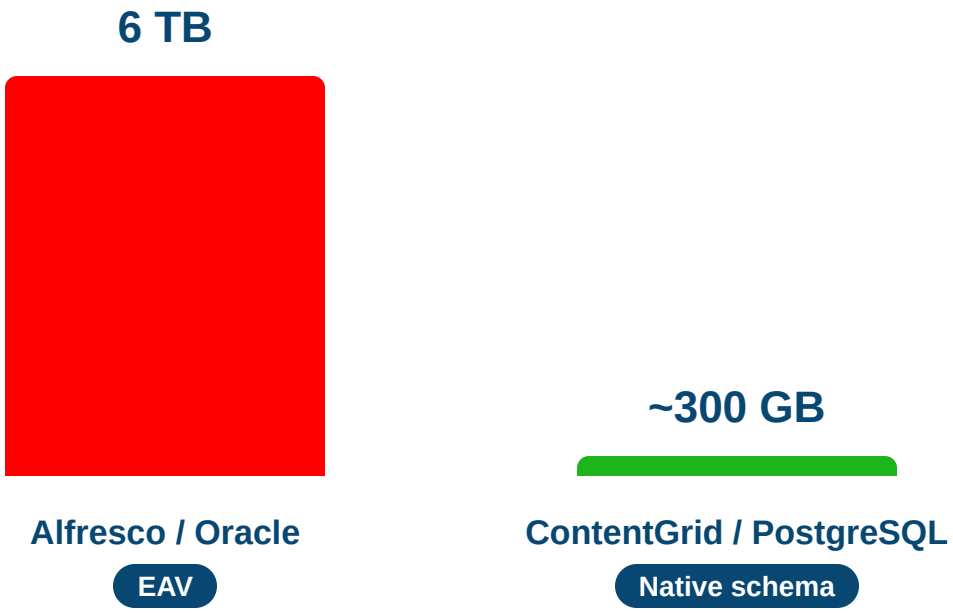
The ContentGrid database: ~300 GB

- 250M rows of typed data
- Targeted indexes
- No attribute row explosion

20× smaller — by fixing the data model

contentgrid
BY AMEXIO

Storage at a Glance



Our Approach

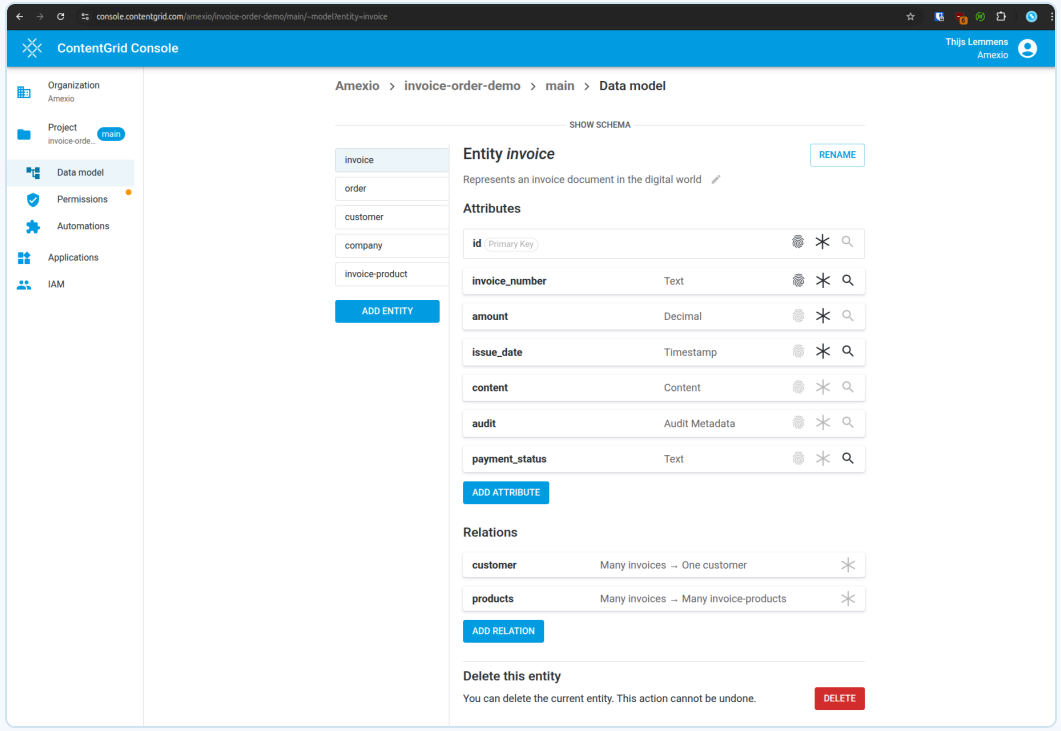
Generating native schemas from semantic models

contentgrid
BY AMEXIO

Two Platforms

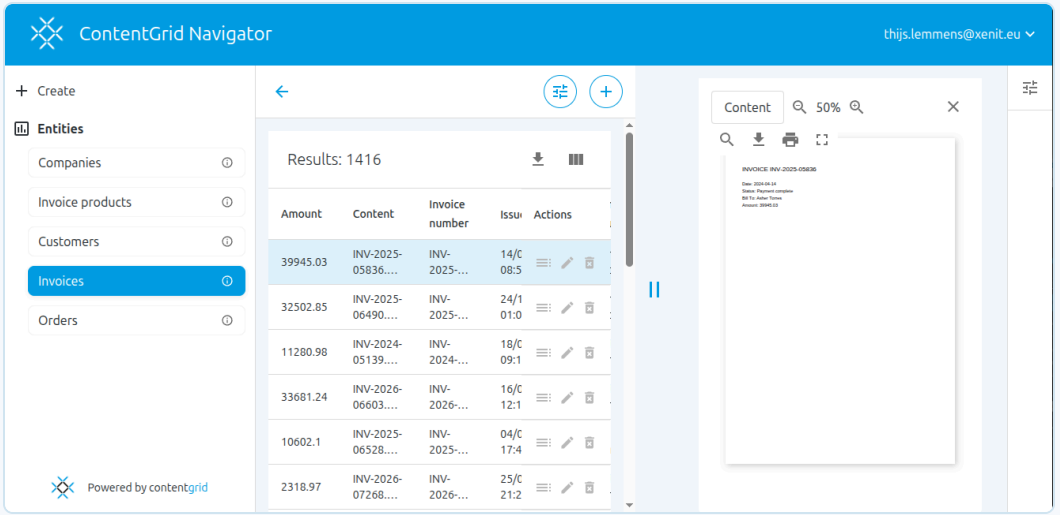
One to define the model. One to run it.

MANAGEMENT PLATFORM



ContentGrid Console — define entities, attributes, permissions

RUNTIME PLATFORM



ContentGrid Navigator — end-user app, driven by the model

contentgrid BY AMEXIO

The Management Platform

Architect

Source of truth for the domain model — entities, attributes, relations

→

Scribe

Compiles model changes into:

- JSON model
- Flyway SQL migrations
- OPA policies

→

Captain

- Provisions database
- manages credentials
- deployment

Every model change → versioned, reviewed SQL migration

Schema always reflects the model — no manual DDL

contentgrid
BY AMEXIO

Model-Driven Schema Generation

User-defined semantic model

```
{
  "name": "article",
  "table": "article",
  "attributes": [
    {
      "type": "simple",
      "name": "title",
      "dataType": "text",
      "columnName": "title",
      "constraints": [{ "type": "required" }]
    },
    {
      "type": "simple",
      "name": "publishedAt",
      "dataType": "datetime",
      "columnName": "published_at"
    },
    {
      "type": "simple",
      "name": "wordCount",
      "dataType": "long",
      "columnName": "word_count"
    }
  ],
  ...
}
```

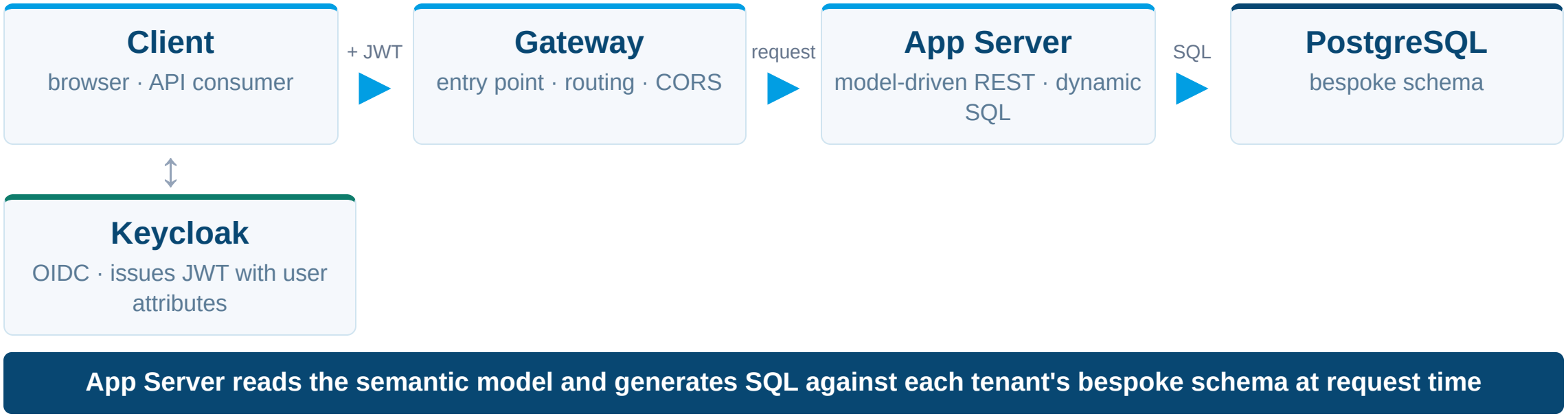
Generated PostgreSQL DDL

```
CREATE TABLE article (
  id           uuid PRIMARY KEY
                 DEFAULT gen_random_uuid(),
  title        text NOT NULL,
  published_at timestamptz,
  word_count   bigint
);
```

contentgrid BY AMEXIO

The Runtime Platform

How a request flows from client to database

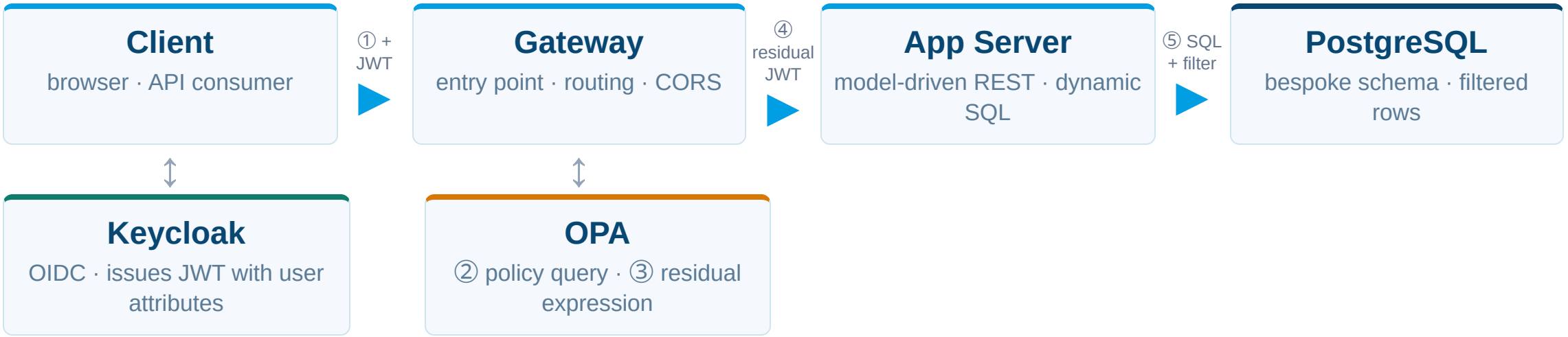


App Server reads the semantic model and generates SQL against each tenant's bespoke schema at request time

contentgrid BY AMEXIO

Adding Authorization

OPA filters rows before they leave the database



Gateway encodes OPA's residual as a JWT claim — App Server applies it as a SQL `WHERE` filter

contentgrid BY AMEXIO

Partial Evaluation: The Key Insight

Naïve approach

```
for each row in invoices:        ← full table scan
  if OPA.check(row, user):       ← N network calls
    return row
```

N rows fetched, N OPA calls, unauthorized data in memory

Partial evaluation

```
OPA.partial_eval(policy, user)  ← 1 call, no entity data
    → residual: dept='sales'
      OR status='published'

SELECT * FROM invoices
WHERE dept='sales'
  OR status='published'          ← filtered at the DB
```

1 OPA call, unauthorized rows never leave PostgreSQL

The security boundary is enforced inside the database — not in application memory

contentgrid BY AMEXIO

From Policy Rule to SQL Filter

① Policy condition

```
entity.department == user.department
OR entity.status == "published"
```

② OPA partial eval

```
OPA.partial_eval(policy, user)
  → deptartment == "sales"  OR  status == "published"
```

③ SQL WHERE (JOOQ)

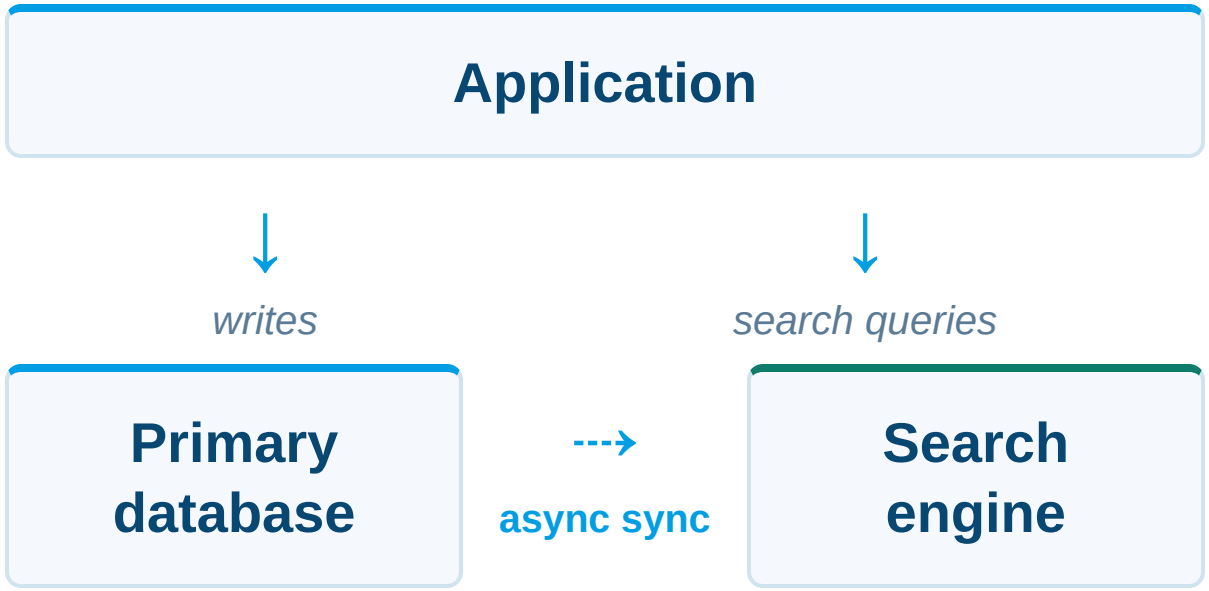
```
SELECT * FROM invoices
WHERE department = 'sales' OR status = 'published'
```

Defined in Architect UI → compiled to Rego by Scribe → evaluated by OPA (1 call, result in Gateway JWT) → App Server builds JOOQ predicate

Unauthorized rows never leave the database — the filter runs inside PostgreSQL

contentgrid BY AMEXIO

How most ECMs handle search



Two stores · one source of truth · eventually consistent

contentgrid BY AMEXIO

Costs of an external search index

Eventual consistency

Writes become searchable after a delay — seconds to hours. Users hit a "not found" window.

Operational overhead

A second cluster to size, monitor, upgrade, and secure. Schema changes trigger reindex.

Bugs cause divergence

Dropped events and schema drift make the index silently disagree with the DB. Recovery = full reindex.

Resource consumption

Searchable corpus stored twice — CPU, memory, storage per tenant per environment.

An external search tier is a second distributed system to design, operate, and reason about

contentgrid
BY AMEXIO

Searchable fields, declared in the model

Exact

```
title = $1
```

Standard btree index

Prefix

```
normalize(title)
  LIKE normalize($1) || '%'
```

Functional btree range scan

Full-text

```
to_tsvector('english',
    normalize(body))
  @@ websearch_to_tsquery(
    'english', normalize($1))
```

Per-property language, GIN index

Three index-friendly query shapes — no Solr, no Elasticsearch

contentgrid BY AMEXIO

Exact + prefix: encoding matters

NFKC

`cafe\u0301`

4 chars + combining accent

→

café

4 chars + precomposed é

Unicode normalization

lower()

`Café`

C a f é

→

café

c a f é

Case fold

unaccent()

`café`

c a f é

→

cafe

c a f e

Strip diacritics

Composed into one `IMMUTABLE PARALLEL SAFE` function — eligible for functional btree index

Full-text search

Per-property language

```
ENGLISH  → "english"
FRENCH   → "french"
Dutch    → "dutch"
```

Locale → PostgreSQL FTS dictionary

Generated SQL

```
to_tsvector('english',
    normalize(body))
  @@ websearch_to_tsquery(
    'english', normalize($1))
```

Same normalize() as prefix search

Per-tenant tables · per-property language · zero external search service

contentgrid BY AMEXIO

The Future of Search

ParadeDB — advanced search, still in PostgreSQL

Faceted search

Aggregate result counts per attribute value

Next-gen FTS + ranking

BM25 scoring, relevance-ordered results via `pg_search`

Hit highlighting

Return matching text fragments

Cross-entity search

Search across multiple tables

Suggestions

Autocomplete as-you-type

Semantic search

Vector embeddings via `pgvector`

More search features · still one database · still transactional

contentgrid BY AMEXIO

Conclusions

- EAV trades correctness & performance for flexibility — you can have both
- Native schemas: 20× smaller, full SQL tooling, real constraints
- ContentGrid generates & migrates schemas from a semantic model
- Security boundary enforced inside PostgreSQL via partial evaluation
- Advanced search — keeping everything in one database

contentgrid
BY AMEXIO

ContentGrid is Open Source

github.com/xenit-eu/contentgrid-appserver

contentgrid
BY AMEXIO

Questions?

contentgrid.com

contentgrid
BY AMEXIO

Thank You

Thijs Lemmens

thijs.lemmens@xenit.eu