Bilko ADR-017: RLS Multi-Tenancy

Author: ALAI, 2026

```
# ADR-017: RLS Multi-Tenancy Migration

**Status:** Accepted
**Date:** 2026-04-21
**Author:** ALAI, 2026
**Replaces:** ADR-005 (App-Layer Multi-Tenancy)
**Related:** ADR-015 (Four-Jurisdiction Plugin), ADR-018 (Market vs Locale)

---

## Context

Bilko's current multi-tenancy model (ADR-005) uses **application-layer scoping**: every
query manually filters `WHERE organization_id = :current_org`. This works but has scaling
and security issues:

**Problems with ADR-005:**

1. **Error-prone:** Forgetting `WHERE organization_id = ...` exposes all tenant data
2. **Performance:** Can't use Postgres partition pruning (requires RLS)
3. **Audit complexity:** Cross-tenant queries are syntactically identical to legitimate
queries
4. **No defense-in-depth:** A single SQL injection bypasses all scoping

**Requirement from master plan (Phase 2):**

- Migrate to **PostgreSQL Row-Level Security (RLS)** for tenant isolation
- Non-disruptive 3-phase migration (coexist ? validate ? retire ADR-005)
- Add **versioned Chart of Accounts** table (regulatory changes = data writes, not schema
migrations)
- Partition **audit log** (`logged_action`) by `country_code` (different retention periods
per jurisdiction)

---

## Decision

### 1. Add `country_code` Column to All Tenant Tables

**Phase 2A Task 2.1:**

```sql
-- Flyway migration V2_001__add_country_code.sql
ALTER TABLE organizations ADD COLUMN country_code CHAR(2) NOT NULL DEFAULT 'RS';
ALTER TABLE invoices ADD COLUMN country_code CHAR(2);
ALTER TABLE expenses ADD COLUMN country_code CHAR(2);
-- ... repeat for all 15 tenant tables

-- Backfill from organizations.taxJurisdiction ? country_code mapping
UPDATE invoices i
SET country_code = SUBSTRING(o.tax_jurisdiction FROM 1 FOR 2)
FROM organizations o
WHERE i.organization_id = o.id;

-- Enforce NOT NULL after backfill
ALTER TABLE invoices ALTER COLUMN country_code SET NOT NULL;
```

Validation:

```sql
-- Zero rows should be NULL after backfill
```
```

```
SELECT COUNT(*) FROM invoices WHERE country_code IS NULL;
-- Expect: 0
```

2. Enable RLS Policies in PERMISSIVE Mode (Coexistence)

Phase 2A Task 2.1 (continued):

```sql
-- Enable RLS on all tenant tables
ALTER TABLE invoices ENABLE ROW LEVEL SECURITY;
ALTER TABLE expenses ENABLE ROW LEVEL SECURITY;
-- ... repeat for all tables

-- Create PERMISSIVE policies (allow all during migration)
CREATE POLICY tenant_isolation ON invoices
FOR ALL
USING (country_code = current_setting('app.current_org_country', TRUE)::text);

-- Policy for auditors (cross-tenant READ-ONLY)
CREATE POLICY auditor_country ON invoices
FOR SELECT
USING (
  current_setting('app.user_role', TRUE) = 'auditor'
  AND country_code = ANY(current_setting('app.auditor_countries', TRUE)::text[])
);
```

Key decisions:

- Use `current_setting('app.current_org_country')` session variable (set at connection open)
- RLS policies are **PERMISSIVE** during migration — app middleware still enforces scoping
- RLS becomes **redundant safety layer**, not the primary gate

3. Versioned Chart of Accounts Table

Problem: Pravilnik (regulatory chart of accounts) changes yearly. Example: Serbia changed CoA in 2020 (Sl. glasnik RS br. 89/2020). Existing `chart_of_accounts` table has no versioning — schema migration required for updates.

Solution: Time-versioned CoA table per jurisdiction.

Phase 2A Task 2.1 DDL:

```sql
CREATE TABLE chart_of_accounts (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  country_code CHAR(2) NOT NULL,
  account_code VARCHAR(10) NOT NULL,
  account_name VARCHAR(255) NOT NULL,
  account_type VARCHAR(50) NOT NULL,  -- ASSET, LIABILITY, EQUITY, REVENUE, EXPENSE
  parent_code VARCHAR(10),  -- For hierarchical CoA
  valid_from DATE NOT NULL,
  valid_to DATE,  -- NULL = currently valid
  version VARCHAR(20) NOT NULL,  -- e.g., "RS_2020", "HR_2022"
  created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
  UNIQUE(country_code, account_code, valid_from)
);

CREATE INDEX idx_coa_country_valid ON chart_of_accounts(country_code, valid_from, valid_to);

-- Seed Serbia 2020 Pravilnik
INSERT INTO chart_of_accounts (country_code, account_code, account_name, account_type,
valid_from, version)
VALUES
  ('RS', '100', '????????????? ???????', 'ASSET', '2020-01-01', 'RS_2020'),
  ('RS', '200', '??????????', 'ASSET', '2020-01-01', 'RS_2020'),
  -- ... full CoA insert
;
```

Query pattern (get current CoA for organization):

```kotlin
// In application code:
val coa = db.query("""
  SELECT * FROM chart_of_accounts
  WHERE country_code = :country
  AND valid_from <= CURRENT_DATE
  AND (valid_to IS NULL OR valid_to > CURRENT_DATE)
```

```
  ORDER BY account_code
""", mapOf("country" to org.countryCode()))
```

Regulatory update workflow:

1. Government publishes new Pravilnik (e.g., Serbia 2027)
2. Admin inserts new rows with `valid_from = '2027-01-01'`, `version = 'RS_2027'`
3. Update old rows: `SET valid_to = '2026-12-31' WHERE version = 'RS_2020'`
4. **No schema migration required** — regulatory change is data, not code

4. Partition Audit Log by `country_code`

Problem: Different retention laws per jurisdiction:

- RS: 10 years
- HR: 11 years
- BA-FED: 10 years
- BA-RS: 11 years

Current `logged_action` table is unpartitioned — retention policy requires full-table scan.

Solution: Declarative partitioning by `country_code`.

Phase 2B Task 2.2 DDL:

```sql
-- Create partitioned table
CREATE TABLE logged_action_partitioned (
  id BIGSERIAL,
  schema_name TEXT NOT NULL,
  table_name TEXT NOT NULL,
  user_name TEXT,
  action TEXT NOT NULL,  -- INSERT, UPDATE, DELETE
  original_data JSONB,
  new_data JSONB,
  query TEXT,
  country_code CHAR(2) NOT NULL,
  created_at TIMESTAMPTZ NOT NULL DEFAULT NOW()
) PARTITION BY LIST (country_code);

-- Create partitions
CREATE TABLE logged_action_rs PARTITION OF logged_action_partitioned FOR VALUES IN ('RS');
CREATE TABLE logged_action_hr PARTITION OF logged_action_partitioned FOR VALUES IN ('HR');
CREATE TABLE logged_action_ba PARTITION OF logged_action_partitioned FOR VALUES IN ('BA');

-- Indexes per partition
CREATE INDEX idx_logged_action_rs_created ON logged_action_rs(created_at);
CREATE INDEX idx_logged_action_hr_created ON logged_action_hr(created_at);
CREATE INDEX idx_logged_action_ba_created ON logged_action_ba(created_at);

-- Retention policy (cron job)
-- RS: DELETE WHERE created_at < NOW() - INTERVAL '10 years'
-- HR: DELETE WHERE created_at < NOW() - INTERVAL '11 years'
```

Migration from old `logged_action` table:

1. Create `logged_action_partitioned` (new table)
2. Backfill from `logged_action` WHERE `created_at > NOW() - INTERVAL '1 year'` (recent data only)
3. Rename `logged_action` ? `logged_action_legacy` (keep for 90 days)
4. Rename `logged_action_partitioned` ? `logged_action`
5. Update audit trigger to insert into new partitioned table

Zero downtime: Both tables coexist during backfill.

5. Retire ADR-005 App-Layer Scoping

Phase 2C Task 2.3 (only after validation):

Validation gate (Task 4.1):

```sql
-- Test cross-tenant isolation with rogue role
SET app.current_org_country = 'RS';
SET ROLE rogue_user;
SELECT COUNT(*) FROM invoices WHERE country_code = 'HR';
-- Expect: 0 (RLS blocks cross-tenant read)
```

Only proceed if validation passes.

Retirement steps:

1. Remove `org-scope.ts` middleware from Ktor request chain
2. Wire `SET app.current_org_country = :country` at Prisma connection open:
   ```kotlin
   val conn = dataSource.connection
   conn.prepareStatement("SET app.current_org_country = ?")
     .apply { setString(1, org.countryCode()) }
     .execute()
   ```
3. Switch RLS policies from PERMISSIVE ? RESTRICTIVE:
   ```sql
   DROP POLICY tenant_isolation ON invoices;
   CREATE POLICY tenant_isolation ON invoices
   FOR ALL
   USING (country_code = current_setting('app.current_org_country')::text)
   WITH CHECK (country_code = current_setting('app.current_org_country')::text);
   ```
4. Mark ADR-005 as "Superseded by ADR-017"

Consequences

Positive

1. **Defense-in-depth:** Even if app logic forgets `WHERE organization_id`, RLS blocks cross-tenant queries
2. **Partition pruning:** Queries scoped to one country ? Postgres prunes other partitions ? faster
3. **Compliance clarity:** Retention policy is per-jurisdiction, encoded in partition retention jobs
4. **Auditability:** RLS policy violations logged to `pg_stat_activity` — security team can detect anomalies

Negative

1. **Migration complexity:** 3-phase migration requires careful sequencing (coexist ? validate ? retire)
2. **Session state:** Every connection must `SET app.current_org_country` — connection pooling requires session reset
3. **Performance overhead:** RLS policies add ~2-5% query overhead (measured in Postgres 16 benchmarks)

Risks

1. **Connection pooling bugs:** If `app.current_org_country` not reset between connections ? cross-tenant leak. **Mitigation:** Wrap all queries in session initializer; unit test connection pool state.
2. **Partition key mismatch:** If `country_code` and `tax_jurisdiction` diverge (e.g., BA split) ? partition logic breaks. **Mitigation:** `country_code` derived from `tax_jurisdiction` via DB trigger; enforce consistency at write time.
3. **Backfill failure:** If backfill leaves NULLs in `country_code` ? RLS blocks all queries. **Mitigation:** NOT NULL constraint only applied after zero-NULL validation query passes.

Implementation Notes

Exchange Rate Precision (Corrected per Petter G review)

Problem: Current schema uses `NUMERIC(19,4)` for exchange rates. Crypto conversions (BTC/RSD) require 10 decimal places.

Fix (Phase 2A Task 2.1):

```sql
ALTER TABLE exchange_rates ALTER COLUMN rate TYPE NUMERIC(20,10);
```

RLS Policy Testing

Test suite (Proveo E2E):

1. Create org in RS jurisdiction
2. Insert invoice with `country_code = 'RS'`

````
3. Set session `app.current_org_country = 'HR'`
4. Query invoices ? expect empty result set
5. Set session `app.current_org_country = 'RS'`
6. Query invoices ? expect invoice returned

**Evidence:** `~/system/evidence/bilko-rls-isolation-YYYYMMDD.json`

### Data Quality Audit (Task 2.4)

**Nightly cron:**

```sql
-- Alert if any tenant table has NULL country_code
SELECT table_name, COUNT(*) FROM (
  SELECT 'invoices' AS table_name, COUNT(*) AS ct FROM invoices WHERE country_code IS NULL
  UNION ALL
  SELECT 'expenses', COUNT(*) FROM expenses WHERE country_code IS NULL
) WHERE ct > 0;
```

Slack alert if any row returns count > 0.

---

## References

- **Postgres RLS docs:** https://www.postgresql.org/docs/16/ddl-rowsecurity.html
- **Declarative partitioning:** https://www.postgresql.org/docs/16/ddl-partitioning.html
- **Master plan:** `~/system/specs/bilko-multi-market-architecture-plan.md` (Phase 2 Tasks 2.1–2.3)
- **Related ADRs:**
  - ADR-005: App-Layer Multi-Tenancy (superseded by this ADR)
  - ADR-015: Four-Jurisdiction Plugin Architecture
  - ADR-018: Market vs Locale Separation

---

## Approval

**Approved:** 2026-04-21 by CEO Alem Basic
**Execution:** Phase 2 Tasks 2.1–2.3 (not yet started — blocked on Phase 1 completion)
````

Revision #2
Created 2026-04-22 10:36:58 UTC by John
Updated 2026-05-31 20:06:26 UTC by John