# Partition Migration – Detailed Implementation Plan

## a. Create Partitioned Table

Goal: Prepare the parent table for partitioning.

CREATE SCHEMA IF NOT EXISTS vit\_trans;  
  
CREATE TABLE vit\_trans.wellness\_attributes\_part (  
 entity\_no BIGINT NOT NULL,  
 attribute\_id BIGINT NOT NULL,  
 eff\_from TIMESTAMP NOT NULL,  
 eff\_to TIMESTAMP NULL,  
 attribute\_value VARCHAR(200) NOT NULL,  
 origin\_type BIGINT NOT NULL,  
 origin\_id BIGINT NOT NULL,  
 os\_user\_last\_modified VARCHAR(15),  
 sess\_user\_last\_modified VARCHAR(15),  
 date\_time\_last\_modified TIMESTAMP,  
 archive\_status SMALLINT  
) PARTITION BY RANGE (eff\_from);

## b. Load Initial Data (2023-01-01 → Now)

Goal: Move historical data from the old table to the new partitioned table.

INSERT INTO vit\_trans.wellness\_attributes\_part  
SELECT \*  
FROM vit\_trans.wellness\_attributes  
WHERE eff\_from >= '2023-01-01';

## c. Create Quarterly Partitions (DO $$)

Goal: Create partitions for 2023, 2024 by quarter, and one partition for 2025 onwards.

DO $$  
DECLARE  
 y INT;  
 q INT;  
 start\_date DATE;  
 end\_date DATE;  
 tbl\_name TEXT;  
 constraint\_name TEXT;  
BEGIN  
 FOR y IN 2023..2024 LOOP  
 FOR q IN 1..4 LOOP  
 start\_date := make\_date(y, (q - 1) \* 3 + 1, 1);  
 end\_date := start\_date + interval '3 months';  
  
 tbl\_name := format('vit\_trans.wellness\_attributes\_%s\_q%s', y, q);  
 constraint\_name := format('wellness\_attributes\_%s\_q%s\_pk', y, q);  
  
 EXECUTE format($sql$  
 CREATE TABLE IF NOT EXISTS %I  
 PARTITION OF vit\_trans.wellness\_attributes\_part  
 FOR VALUES FROM (%L) TO (%L);  
 $sql$, tbl\_name, start\_date, end\_date);  
  
 BEGIN  
 EXECUTE format($sql$  
 ALTER TABLE %I  
 ADD CONSTRAINT %I PRIMARY KEY (entity\_no, attribute\_id, eff\_from);  
 $sql$, tbl\_name, constraint\_name);  
 EXCEPTION  
 WHEN duplicate\_object THEN  
 RAISE NOTICE 'Primary key already exists for %', tbl\_name;  
 END;  
 END LOOP;  
 END LOOP;  
  
 tbl\_name := 'vit\_trans.wellness\_attributes\_2025\_plus';  
 constraint\_name := 'wellness\_attributes\_2025\_plus\_pk';  
  
 EXECUTE format($sql$  
 CREATE TABLE IF NOT EXISTS %I  
 PARTITION OF vit\_trans.wellness\_attributes\_part  
 FOR VALUES FROM ('2025-01-01') TO (MAXVALUE);  
 $sql$, tbl\_name);  
  
 BEGIN  
 EXECUTE format($sql$  
 ALTER TABLE %I  
 ADD CONSTRAINT %I PRIMARY KEY (entity\_no, attribute\_id, eff\_from);  
 $sql$, tbl\_name, constraint\_name);  
 EXCEPTION  
 WHEN duplicate\_object THEN  
 RAISE NOTICE 'Primary key already exists for %', tbl\_name;  
 END;  
END$$;

## d. Create Trigger for Sync-Up

Goal: Keep old and new tables synchronized during migration.

CREATE OR REPLACE FUNCTION vit\_trans.sync\_to\_partition()  
RETURNS TRIGGER AS $$  
BEGIN  
 IF (TG\_OP = 'INSERT') THEN  
 INSERT INTO vit\_trans.wellness\_attributes\_part VALUES (NEW.\*);  
 ELSIF (TG\_OP = 'UPDATE') THEN  
 UPDATE vit\_trans.wellness\_attributes\_part  
 SET attribute\_value = NEW.attribute\_value,  
 eff\_to = NEW.eff\_to,  
 date\_time\_last\_modified = NEW.date\_time\_last\_modified  
 WHERE entity\_no = NEW.entity\_no  
 AND attribute\_id = NEW.attribute\_id  
 AND eff\_from = NEW.eff\_from;  
 ELSIF (TG\_OP = 'DELETE') THEN  
 DELETE FROM vit\_trans.wellness\_attributes\_part  
 WHERE entity\_no = OLD.entity\_no  
 AND attribute\_id = OLD.attribute\_id  
 AND eff\_from = OLD.eff\_from;  
 END IF;  
 RETURN NULL;  
END;  
$$ LANGUAGE plpgsql;  
  
CREATE TRIGGER sync\_partition\_trigger  
AFTER INSERT OR UPDATE OR DELETE ON vit\_trans.wellness\_attributes  
FOR EACH ROW EXECUTE FUNCTION vit\_trans.sync\_to\_partition();

## e. Monitoring – Data Quality Checks

Goal: Verify data consistency between old and partitioned tables.

SELECT COUNT(\*) FROM vit\_trans.wellness\_attributes;  
SELECT COUNT(\*) FROM vit\_trans.wellness\_attributes\_part;  
  
SELECT tableoid::regclass AS partition, COUNT(\*)  
FROM vit\_trans.wellness\_attributes\_part  
GROUP BY tableoid  
ORDER BY partition;

## f. Cut-over (Swap Tables)

Goal: Rename the new partitioned table to replace the old table.

LOCK TABLE vit\_trans.wellness\_attributes IN ACCESS EXCLUSIVE MODE;  
  
ALTER TABLE vit\_trans.wellness\_attributes RENAME TO wellness\_attributes\_backup;  
ALTER TABLE vit\_trans.wellness\_attributes\_part RENAME TO wellness\_attributes;  
  
GRANT SELECT, INSERT, UPDATE, DELETE ON vit\_trans.wellness\_attributes TO your\_app\_user;

## g. Re-enable Trigger (if needed)

Goal: If synchronization is needed with other systems, re-create the trigger on the new table.