SQL Faker Stored Procedures Documentation

Overview

This schema implements a set of stored procedures that generate realistic fake user data directly in PostgreSQL. The procedures use weighted randomness, locale-aware tables, and statistical distributions to produce names, addresses, emails, phone numbers, and physical attributes.

Core Random Generators

`prng_int(seed BIGINT, n INT, min INT, max INT)`

- **Purpose**: Generate reproducible pseudo-random integers.
- **Arguments**:
  - `seed`: deterministic seed for reproducibility.
  - `n`: sequence index (ensures different values per call).
  - `min`, `max`: bounds for integer output.
- **Algorithm**: Linear congruential generator (LCG) seeded with `seed + n`. Ensures reproducible outputs across runs.

`prng_float(seed BIGINT, n INT)`

- **Purpose**: Generate reproducible pseudo-random floating-point numbers in `[0,1)`.
- **Arguments**:
  - `seed`: deterministic seed.
  - `n`: sequence index.
- **Algorithm**: Same LCG as `prng_int`, normalized to `[0,1)`.

`prng_normal(seed BIGINT, n INT, mean DOUBLE, stddev DOUBLE)`

- **Purpose**: Generate normally distributed values.

- **Arguments**:
  - `seed`: deterministic seed.
  - `n`: sequence index.
  - `mean`: target mean.
  - `stddev`: target standard deviation.
- **Algorithm**: Box–Muller transform using two uniform randoms from `prng_float`. Produces Gaussian distribution with constant PDF.

`prng_sphere_coords(seed BIGINT, n INT)`

- **Purpose**: Generate latitude/longitude pairs uniformly distributed on Earth's sphere.
- **Arguments**:
  - `seed`: deterministic seed.
  - `n`: sequence index.
- **Algorithm**:
  - Latitude: `asin(2u - 1)` where `u` is uniform random.
  - Longitude: `2πv - π` where `v` is uniform random.
    Ensures constant probability density across the sphere (no clustering at poles).

Weighted Selection

`select_weighted_item(seed BIGINT, n INT, table_name TEXT, locale TEXT, gender CHAR)`

- **Purpose**: Select a row from a seed table based on frequency weights.
- **Arguments**:
  - `seed`: deterministic seed.
  - `n`: sequence index.
  - `table_name`: which table to query (`first_names`, `last_names`, `titles`, etc.).
  - `locale`: locale code (`en_US`, `de_DE`).
  - `gender`: optional filter (`M`, `F`, `U`).
- **Algorithm**:
  - Sum frequencies for the filtered rows.
  - Generate random integer in `[1, total]`.
  - Select the row whose cumulative frequency range contains the random number.

User Attribute Generators

`generate_name(seed BIGINT, n INT, locale TEXT)`

- **Purpose**: Generate a full name with title, first name, last name, and optional suffix.
- **Arguments**:
  - `seed`: deterministic seed.
  - `n`: sequence index.
  - `locale`: locale code.
- **Algorithm**:
  - Title: weighted from `titles`.
  - First name: weighted from `first_names`.
  - Last name: weighted from `last_names`.
  - Optional suffix (`Jr.`, `Sr.`, `III`) added with small probability.
- **Output**: `"Dr. Maria Müller Jr."`

`generate_address(seed BIGINT, n INT, locale TEXT)`

- **Purpose**: Generate realistic street addresses.
- **Arguments**:
  - `seed`: deterministic seed.
  - `n`: sequence index.
  - `locale`: locale code.
- **Algorithm**:
  - Street name: from `street_names`.
  - Street type: weighted from `street_types`.
  - Building number: random integer.
  - City: from `cities`.
  - Postal code: generated from city's `postal_code_pattern`.
- **Output**: `"123 Hauptstraße, Berlin 10115"`

`generate_phone(seed BIGINT, n INT, locale TEXT)`

- **Purpose**: Generate phone numbers in locale-specific formats.
- **Arguments**:
  - `seed`: deterministic seed.
  - `n`: sequence index.
  - `locale`: locale code.

- **Algorithm**:
  - Select format from `phone_formats` (weighted).
  - Replace `N` with digits.
- **Output**: `"(212) 555-1234"` or `"030/12345"`

`generate_email(seed BIGINT, n INT, locale TEXT, first TEXT, last TEXT)`

- **Purpose**: Generate realistic email addresses.
- **Arguments**:
  - `seed`: deterministic seed.
  - `n`: sequence index.
  - `locale`: locale code.
  - `first`, `last`: name components.
- **Algorithm**:
  - Combine first/last with separators (`.`, `_`, no separator).
  - Append random number with small probability.
  - Select domain from `email_domains`.
- **Output**: `"maria.mueller@gmx.de"`

`generate_physical_attributes(seed BIGINT, n INT, locale TEXT, gender CHAR)`

- **Purpose**: Generate eye color, height, weight.
- **Arguments**:
  - `seed`: deterministic seed.
  - `n`: sequence index.
  - `locale`: locale code.
  - `gender`: optional filter.
- **Algorithm**:
  - Eye color: weighted from `eye_colors`.
  - Height: normal distribution (mean/stddev by gender/locale).
  - Weight: normal distribution (mean/stddev by gender/locale).
- **Output**: `"Eye: Blau, Height: 175 cm, Weight: 72 kg"`

Full User Generator

`generate_fake_users(locale TEXT, seed BIGINT, count INT, batch_size INT)`

- **Purpose**: Generate a batch of complete fake users.
- **Arguments**:
  - `locale`: locale code.
  - `seed`: deterministic seed.
  - `count`: number of users to generate.
  - `batch_size`: number of rows per insert batch.
- **Algorithm**:
  - Iteratively call `generate_name`, `generate_address`, `generate_phone`, `generate_email`, `generate_physical_attributes`.
  - Insert results into `fake_users` table.
- **Output**: Table of users with all attributes.

📌 Usage Examples

```
-- Generate one German name
SELECT generate_name(42, 1, 'de_DE');

-- Generate 5 US addresses
SELECT generate_address(42, g, 'en_US')
FROM generate_series(1,5) g;

-- Generate 100 fake users in German locale
SELECT generate_fake_users('de_DE', 12345, 100, 20);
```

✨ Key Features

- **Deterministic**: Same seed → same output.
- **Locale-aware**: Names, addresses, phones, emails vary by locale.
- **Weighted randomness**: Common values appear more often.
- **Statistical realism**: Heights/weights use normal distribution; lat/long use constant PDF.
