Database Entity Relationship Diagram (ERD)

SmartBudget Canada Database Schema

This document provides a visual representation of the database schema for the SmartBudget Canada application.

```
erDiagram
    User {
        String id PK
        String name
        String email UK
        DateTime emailVerified
        String image
        String password
        DateTime createdAt
        DateTime updatedAt
    }

    Budget {
        String id PK
        String userId FK
        String guestId
        String guestSessionId
        String name
        String province
        String lifeSituation
        String ageRange
        String workStatus
        String housingSituation
        String primaryGoal
        Float grossIncome
        Float netIncome
        Float totalExpenses
        Float disposableIncome
        Float federalTax
        Float provincialTax
        Float totalTax
        DateTime createdAt
        DateTime updatedAt
    }

    BudgetItem {
        String id PK
        String budgetId FK
        String type
        String category
        String subcategory
        String name
        Float amount
        String frequency
        Float monthlyAmount
        String owner
        DateTime createdAt
        DateTime updatedAt
    }

    Asset {
        String id PK
        String userId FK
        String type
        String name
        Float value
        String description
        Json details
        DateTime createdAt
        DateTime updatedAt
    }
```

```
    Liability {
        String id PK
        String userId FK
        String type
        String name
        Float balance
        Float interestRate
        Float minimumPayment
        Float creditLimit
        String description
        Json details
        DateTime createdAt
        DateTime updatedAt
    }

    FinancialAudit {
        String id PK
        String userId FK
        Json auditData
        Json recommendations
        Float hammerScore
        String severity
        DateTime followUpDate
        Boolean completed
        DateTime createdAt
        DateTime updatedAt
    }

    CalculatorResult {
        String id PK
        String userId FK
        String calculatorType
        String title
        Json inputs
        Json results
        String notes
        Boolean isPublic
        DateTime createdAt
        DateTime updatedAt
    }

    Account {
        String id PK
        String userId FK
        String type
        String provider
        String providerAccountId
        String refresh_token
        String access_token
        Int expires_at
        String token_type
        String scope
        String id_token
        String session_state
    }

    Session {
        String id PK
        String sessionToken UK
        String userId FK
        DateTime expires
    }
```

```
    VerificationToken {
        String identifier
        String token UK
        DateTime expires
    }

    %% Relationships
    User ||--o{ Budget : creates
    User ||--o{ Asset : owns
    User ||--o{ Liability : has
    User ||--o{ FinancialAudit : receives
    User ||--o{ CalculatorResult : generates
    User ||--o{ Account : authenticates
    User ||--o{ Session : maintains

    Budget ||--o{ BudgetItem : contains
```

Key Relationships

Primary Relationships

- **User → Budget**: One-to-Many (A user can have multiple budgets)
- **Budget → BudgetItem**: One-to-Many (A budget contains multiple income/expense items)
- **User → Asset**: One-to-Many (A user can have multiple assets)
- **User → Liability**: One-to-Many (A user can have multiple liabilities)

Guest System Relationships

- **Budget.guestId**: Legacy guest tracking system
- **Budget.guestSessionId**: New guest session tracking for migration
- **Budget.userId**: Links to authenticated user (null for guest budgets)

Authentication Relationships

- **User → Account**: One-to-Many (OAuth providers)
- **User → Session**: One-to-Many (Active sessions)
- **VerificationToken**: Standalone for email verification

Migration System Architecture

The guest-to-user migration system uses the following fields:

```sql
-- Legacy guest budgets
SELECT * FROM Budget WHERE guestId IS NOT NULL AND userId IS NULL;

-- New guest session budgets
SELECT * FROM Budget WHERE guestSessionId IS NOT NULL AND userId IS NULL;

-- Migration query (simplified)
UPDATE Budget
SET userId = ?, guestId = NULL, guestSessionId = NULL
WHERE (guestId = ? OR guestSessionId = ?) AND userId IS NULL;
```

Indexes

The following indexes are applied for optimal performance:

- `Budget.guestId` - Index for guest budget queries
- `Budget.guestSessionId` - Index for session-based guest queries
- `Budget.userId` - Foreign key index for user budgets
- `User.email` - Unique index for authentication
- `Session.sessionToken` - Unique index for session lookup

Data Types

Enums

```
enum ItemType {
  INCOME
  EXPENSE
}

enum Frequency {
  WEEKLY
  BIWEEKLY
  MONTHLY
  QUARTERLY
  YEARLY
}
```

JSON Fields

- **FinancialAudit.auditData**: Structured audit information
- **FinancialAudit.recommendations**: AI-generated recommendations
- **Asset.details**: Asset-specific metadata
- **Liability.details**: Liability-specific metadata
- **CalculatorResult.inputs**: Calculator input parameters
- **CalculatorResult.results**: Calculated results and charts

Security Considerations

- **guestId/guestSessionId**: Used for anonymous budget tracking
- **Password**: Hashed using bcryptjs
- **Session tokens**: Managed by NextAuth.js
- **HTTP-only cookies**: Secure guest session management

Generating PNG ERD

To generate a PNG version of this ERD from the Prisma schema:

```
# Install prisma-erd-generator
npm install -g prisma-erd-generator

# Generate ERD
npx prisma generate
prisma-erd-generator --schema=./prisma/schema.prisma --output=./docs/database-erd.png
```

Alternative tools:
- Prisma ERD Generator (https://github.com/keonik/prisma-erd-generator)
- DBDiagram.io (https://dbdiagram.io/) with Prisma schema import
- DrawSQL (https://drawsql.app/) with PostgreSQL import