TripBook Database Schema Documentation

📊 Overview

This document provides comprehensive documentation for the TripBook SQLite database schema implemented using Room ORM. The database is designed to support the complete reservation system with proper relationships, indexing, and data integrity.

🏗️ Database Architecture

Technology Stack

- **Database**: SQLite 3
- **ORM**: Android Room
- **Language**: Kotlin
- **Type Converters**: LocalDate, LocalDateTime support

Key Design Principles

- **Normalization**: Proper 3NF normalization to reduce redundancy
- **Referential Integrity**: Foreign key constraints ensure data consistency
- **Performance**: Strategic indexing for common queries
- **Scalability**: Designed to handle growing data efficiently
- **Flexibility**: Extensible schema for future features

🏷️ Entity Relationships

```
Users (1) ⟷ (M) Reservations (M) ⟷ (1) Trips
                      ↓
                (M) ⟷ (M) Activities (via ReservationActivities)
                      ↓
                (M) ⟷ (1) TransportOptions
                      ↓
                (M) ⟷ (1) Hotels

Users (1) ⟷ (M) Notifications
Users (1) ⟷ (M) UserFavorites
```

🗃️ Core Entities

1. TripEntity (`trips`)

Purpose: Stores core trip information displayed in DashboardScreen

Column	Type	Description	Index
id	String (PK)	Unique trip identifier	✓
title	String	Trip name	
from_location	String	Departure location	✓
to_location	String	Destination	✓
departure_date	LocalDate	Trip start date	✓
return_date	LocalDate	Trip end date	
image_url	String	Trip image URL	
base_price	Double	Starting price	✓
description	String	Trip description	
duration	String	Trip duration	
category	String	Trip category (enum)	✓
is_active	Boolean	Availability status	
created_at	Long	Creation timestamp	
updated_at	Long	Last update timestamp	

Used by: DashboardScreen, TripManager, ReservationFlow

---

2. UserEntity (`users`)

Purpose: User profiles and authentication

Column	Type	Description	Index
id	String (PK)	Unique user identifier	✓
username	String	Username (unique)	✓
email	String	Email address (unique)	✓
first_name	String	User's first name	
last_name	String	User's last name	
phone	String?	Phone number	
profile_image_url	String?	Profile picture URL	
preferred_currency	String	Currency preference	
notification_enabled	Boolean	Notification settings	
is_active	Boolean	Account status	✓

PROF

Column	Type	Description	Index
is_verified	Boolean	Verification status	
created_at	Long	Registration timestamp	

Used by: ProfileScreen, Authentication, UserManager

---

3. ReservationEntity (`reservations`)

Purpose: Central booking records linking all reservation components

Column	Type	Description	Index
id	String (PK)	Unique reservation ID	✓
user_id	String (FK)	Reference to user	✓
trip_id	String (FK)	Reference to trip	✓
transport_id	String? (FK)	Selected transport	✓
hotel_id	String? (FK)	Selected hotel	✓
hotel_nights	Int	Number of nights	
total_cost	Double	Total reservation cost	
status	String	Reservation status (enum)	✓
payment_status	String	Payment status (enum)	✓
booking_date	LocalDateTime	Booking timestamp	✓
confirmation_number	String?	Booking confirmation	

Used by: ReservationListScreen, ReservationFlow, PaymentScreen

PROF

---

4. TransportOptionEntity (`transport_options`)

Purpose: Available transport for trips

Column	Type	Description	Index
id	String (PK)	Unique transport ID	✓
trip_id	String (FK)	Associated trip	✓
type	String	Transport type (enum)	✓
name	String	Transport service name	
departure_time	LocalDateTime	Departure time	✓

Column	Type	Description	Index
arrival_time	LocalDateTime	Arrival time	
price	Double	Transport cost	✓
capacity	Int?	Maximum capacity	
booked_seats	Int	Current bookings	
is_available	Boolean	Availability status	

Used by: TransportSelectionStep, TransportOptionsStep

---

5. HotelEntity (`hotels`)

Purpose: Hotel accommodation options

Column	Type	Description	Index
id	String (PK)	Unique hotel ID	✓
name	String	Hotel name	
rating	Int	Star rating (1-5)	✓
room_type	String	Room category	
price_per_night	Double	Nightly rate	✓
location	String	Hotel location	✓
amenities	String	Comma-separated amenities	
available_rooms	Int?	Room availability	
is_available	Boolean	Hotel status	✓

PROF

Used by: HotelSelectionStep, SummaryStep

---

6. ActivityEntity (`activities`)

Purpose: Bookable activities and experiences

Column	Type	Description	Index
id	String (PK)	Unique activity ID	✓
name	String	Activity name	
description	String	Activity description	
price	Double	Activity cost	✓

Column	Type	Description	Index
duration	String	Activity duration	
category	String	Activity category (enum)	✓
location	String	Activity location	✓
min_participants	Int	Minimum participants	
max_participants	Int?	Maximum participants	
is_available	Boolean	Availability status	✓

Used by: SummaryStep, ActivitySelectionFragment

🔗 Junction Tables

7. ReservationActivityEntity (`reservation_activities`)

Purpose: Many-to-many relationship between reservations and activities

Column	Type	Description
id	String (PK)	Unique junction ID
reservation_id	String (FK)	Reference to reservation
activity_id	String (FK)	Reference to activity
quantity	Int	Number of bookings
unit_price	Double	Price per unit
total_price	Double	Total cost
participants	Int	Number of participants
is_confirmed	Boolean	Confirmation status

PROF

📱 Supporting Tables

8. NotificationEntity (`notifications`)

Purpose: User notifications and alerts

Column	Type	Description	Index
id	String (PK)	Unique notification ID	✓
user_id	String (FK)	Target user	✓
title	String	Notification title	

Column	Type	Description	Index
message	String	Notification content	
type	String	Notification type (enum)	✓
is_read	Boolean	Read status	✓
reservation_id	String? (FK)	Related reservation	

Used by: NotificationScreen, FakeNotificationDispatcher

---

9. UserFavoriteEntity (`user_favorites`)

Purpose: User's favorite trips, hotels, and activities

Column	Type	Description	Index
id	String (PK)	Unique favorite ID	✓
user_id	String (FK)	User reference	✓
favorite_id	String	ID of favorited item	✓
favorite_type	String	Type (TRIP/HOTEL/ACTIVITY)	✓
notes	String?	Personal notes	
priority	Int	Priority level	

Used by: Dashboard, ProfileScreen, Recommendations

🚀 Performance Optimizations

Strategic Indexing

PROF

- **Primary Keys**: All tables have indexed primary keys
- **Foreign Keys**: All foreign key columns are indexed
- **Search Fields**: Common search fields (location, category, price) are indexed
- **Composite Indexes**: Multi-column indexes for complex queries

Query Optimization

- **Reactive Queries**: Flow-based queries for real-time UI updates
- **Efficient Joins**: Optimized relationship queries
- **Pagination Support**: Ready for large dataset pagination
- **Caching Strategy**: Room's built-in caching mechanisms

🔧 Usage Examples

Getting Database Instance

```
val database = TripBookDatabase.getDatabase(context)
val tripDao = database.tripDao()
```

Reactive Data Queries

```
// Observe user reservations
tripDao.getUserReservations(userId).collect { reservations ->
    // Update UI
}
```

Complex Queries

```
// Search trips with filters
tripDao.searchTrips("safari")
    .combine(tripDao.getTripsByPriceRange(100.0, 2000.0)) { search,
price ->
        // Combine results
    }
```

📈 Future Enhancements

Planned Features

- **Full-text search** for better search capabilities
- **Geolocation support** for location-based queries
- **Audit trails** for data change tracking
- **Soft delete patterns** for data recovery
- **Database encryption** for sensitive data

PROF

Migration Strategy

- **Version control**: Schema versioning for updates
- **Migration scripts**: Automated database updates
- **Backward compatibility**: Support for older app versions

🛠️ Development Guidelines

For Developers

1. **Always use DAOs** - Never access entities directly

2. **Leverage Flow** - Use reactive queries for UI updates
3. **Handle nullability** - Properly handle optional relationships
4. **Use transactions** - For multi-table operations
5. **Test thoroughly** - Write unit tests for all DAO methods

Best Practices

- **Consistent naming** - Follow established naming conventions
- **Proper indexing** - Add indexes for new query patterns
- **Data validation** - Validate data before database operations
- **Error handling** - Implement proper exception handling
- **Documentation** - Document all custom queries

Created by: TripBook Development Team
Last Updated: December 2024
Version: 1.0
Database Version: 1