Proposed PostgreSQL Schema for Hotel Review JSON

Understanding the JSON structure

The JSON sample sent by the user represents one **review record** for a single hotel on the Agoda platform. At the top level the JSON contains the hotel identifier, hotel name and platform name. The `comment` object contains detailed information about a single reviewer's experience, including the review text, rating and response information as well as a nested `reviewerInfo` sub-object that captures the reviewer's metadata (country, group type, room type etc.). The `overallByProviders` array summarises aggregated scores for the hotel on the same platform (overall score, number of reviews and "grades" for different aspects like cleanliness and location).

Storing the entire JSON blob in one column would lose the benefits of type-checking and efficient querying. The design below **normalises** the JSON into a set of related tables, allowing efficient queries (for example, "find all reviews for hotel X", "list hotels with an overall score above 8" or "count reviews by country"). A few fields (e.g., formatted date strings) are excluded because they duplicate other fields; these exclusions are noted with explanations.

Proposed relational tables

1. `hotels`

Stores basic hotel information.

```
CREATE TABLE hotels (
    hotel_id           INTEGER PRIMARY KEY,
    hotel_name         TEXT NOT NULL
);
```

2. `providers`

Represents review platforms/providers (Agoda in this case).

```
CREATE TABLE providers (
    provider_id        INTEGER PRIMARY KEY,
    provider_name      TEXT NOT NULL
);
```

3. `reviewers`

Stores information about reviewers.

```
CREATE TABLE reviewers (
    reviewer_id            SERIAL PRIMARY KEY,
    display_member_name    TEXT,
```

```
    country_id              INTEGER,
    country_name            TEXT,
    flag_name               TEXT,
    review_group_id         INTEGER,
    review_group_name       TEXT,
    room_type_id            INTEGER,
    room_type_name          TEXT,
    length_of_stay          INTEGER,
    reviewer_reviewed_count INTEGER,
    is_expert_reviewer      BOOLEAN,
    is_show_global_icon     BOOLEAN,
    is_show_reviewed_count BOOLEAN
);
```

4. reviews

Captures individual review comments. It references the `hotels`, `providers` and `reviewers` tables. The `hotel_review_id` from the JSON becomes the primary key.

```
CREATE TABLE reviews (
    hotel_review_id           INTEGER PRIMARY KEY,
    hotel_id                  INTEGER NOT NULL REFERENCES hotels(hotel_id),
    provider_id               INTEGER NOT NULL REFERENCES
providers(provider_id),
    reviewer_id               INTEGER NOT NULL REFERENCES
reviewers(reviewer_id),
    rating                    NUMERIC(3,1),
    rating_text               TEXT,
    check_in_month_year       TEXT,
    review_date               TIMESTAMP WITH TIME ZONE,
    review_title              TEXT,
    review_comments           TEXT,
    review_negatives          TEXT,
    review_positives          TEXT,
    encrypted_review_data     TEXT,
    responder_name            TEXT,
    response_date_text        TEXT,
    response_translate_source TEXT,
    response_text             TEXT,
    translate_source          TEXT,
    translate_target          TEXT,
    is_show_review_response   BOOLEAN,
    original_title            TEXT,
    original_comment          TEXT,
    formatted_response_date   TEXT,
    review_provider_text      TEXT
);
```

> **Note:** The `response_text` column is included to store any textual response from the hotel; in the sample this field is empty but the schema is prepared for it.

5. `hotel_provider_overall`

Contains aggregated scores and counts of reviews by provider for each hotel. Each row identifies a specific hotel and provider combination.

```
CREATE TABLE hotel_provider_overall (
    id              SERIAL PRIMARY KEY,
    hotel_id        INTEGER NOT NULL REFERENCES hotels(hotel_id),
    provider_id     INTEGER NOT NULL REFERENCES providers(provider_id),
    overall_score   NUMERIC(3,1),
    review_count    INTEGER
);
```

6. `hotel_provider_grades`

Stores per-category grades (Cleanliness, Facilities, etc.) for each hotel/provider combination.

```
CREATE TABLE hotel_provider_grades (
    id              SERIAL PRIMARY KEY,
    hotel_id        INTEGER NOT NULL REFERENCES hotels(hotel_id),
    provider_id     INTEGER NOT NULL REFERENCES providers(provider_id),
    category_name   TEXT NOT NULL,
    score           NUMERIC(3,1)
);
```

These tables normalise the JSON structure while preserving the relationships between hotels, providers, reviews, and reviewers.

Mapping of JSON fields to database columns

The table below shows how each JSON key maps to the proposed tables and columns. Data types are chosen to accommodate the values seen in the sample JSON (for instance, `NUMERIC(3,1)` for one-decimal-place ratings). Fields not mapped are listed at the bottom with reasons.

JSON field	Target table.column	Data type / notes	Reasoning
hotelId	`hotels.hotel_id`	integer	Primary key for hotels.
hotelName	`hotels.hotel_name`	text	Name of the hotel.
platform	`providers.provider_name`	text	Platform name (Agoda). `provider_id` in the JSON

JSON field	Target table.column	Data type / notes	Reasoning
			appears elsewhere; mapping platform string into `providers` allows joining.
comment.providerId	`providers.provider_id`	integer	Primary key for the providers table.
comment.hotelReviewId	`reviews.hotel_review_id`	integer	Unique review identifier and primary key for `reviews`.
comment.rating	`reviews.rating`	numeric(3,1)	Numeric rating.
comment.ratingText	`reviews.rating_text`	text	Human-friendly rating (e.g., “Good”).
comment.checkInDateMonthAndYear	`reviews.check_in_month_year`	text	Retains month and year of check-in; kept as string because day is not included.
comment.encryptedReviewData	`reviews.encrypted_review_data`	text	Encrypted data preserved.
comment.formattedRating	*excluded*	*n/a*	Duplicates numeric `rating` and is not needed for queries; can be recreated in the application.
comment.formattedReviewDate	*excluded*	*n/a*	Duplicate of `review_date` formatted as a string; omitted to avoid redundancy.

JSON field	Target table.column	Data type / notes	Reasoning
comment.ratingText	mapped above		
comment.responderName	`reviews.responder_name`	text	Hotel representative responding to the review.
comment.responseDateText	`reviews.response_date_text`	text	Free-form date string for response (if any).
comment.responseTranslateSource	`reviews.response_translate_source`	text	Language code for the response translation.
comment.reviewComments	`reviews.review_comments`	text	Main body of the review.
comment.reviewNegatives	`reviews.review_negatives`	text	Negative feedback, if provided.
comment.reviewPositives	`reviews.review_positives`	text	Positive feedback, if provided.
comment.reviewProviderLogo	*excluded*	*n/a*	Image/logo URL is not needed for analytical queries; could be added later if display is required.
comment.reviewProviderText	`reviews.review_provider_text`	text	Name of the review provider; stored for redundancy.
comment.reviewTitle	`reviews.review_title`	text	Title of the review.
comment.translateSource	`reviews.translate_source`	text	Language code of the original comment.
comment.translateTarget	`reviews.translate_target`	text	Language code the comment was translated

JSON field	Target table.column	Data type / notes	Reasoning
			into.
comment.reviewDate	`reviews.review_date`	timestamp with time zone	Stored as a timestamp for accurate date/time queries.
comment.reviewerInfo.countryName	`reviewers.country_name`	text	Reviewer's country.
comment.reviewerInfo.displayMemberName	`reviewers.display_member_name`	text	Masked display name.
comment.reviewerInfo.flagName	`reviewers.flag_name`	text	Country code for flag display.
comment.reviewerInfo.reviewGroupName	`reviewers.review_group_name`	text	Group category (e.g., solo, family).
comment.reviewerInfo.roomTypeName	`reviewers.room_type_name`	text	Type of room booked.
comment.reviewerInfo.countryId	`reviewers.country_id`	integer	Numeric country identifier.
comment.reviewerInfo.lengthOfStay	`reviewers.length_of_stay`	integer	Number of nights stayed.
comment.reviewerInfo.reviewGroupId	`reviewers.review_group_id`	integer	Identifier for group category.
comment.reviewerInfo.roomTypeId	`reviewers.room_type_id`	integer	Identifier for room type.
comment.reviewerInfo.reviewerReviewedCount	`reviewers.reviewer_reviewed_count`	integer	Count of previous reviews by the reviewer.
comment.reviewerInfo.isExpertReviewer	`reviewers.is_expert_reviewer`	boolean	Flag indicating expert status.
comment.reviewerInfo.isShowGlobalIcon	`reviewers.is_show_global_icon`	boolean	Controls display of a global icon on UI.
comment.reviewerInfo.isShowReviewedCount	`reviewers.is_show_reviewed_count`	boolean	Indicates whether the review count is shown.

JSON field	Target table.column	Data type / notes	Reasoning
comment.originalTitle	`reviews.original_title`	text	Title before translation (empty in sample).
comment.originalComment	`reviews.original_comment`	text	Original comment before translation (empty in sample).
comment.formattedResponseDate	`reviews.formatted_response_date`	text	Free-form formatted response date.
overallByProviders[].providerId	`hotel_provider_overall.provider_id` & `hotel_provider_grades.provider_id`	integer	Links aggregated scores to the provider.
overallByProviders[].provider	*redundant*	*n/a*	The provider name is already stored in `providers.provider_name`.
overallByProviders[].overallScore	`hotel_provider_overall.overall_score`	numeric(3,1)	Average score for the hotel on this provider.
overallByProviders[].reviewCount	`hotel_provider_overall.review_count`	integer	Number of reviews aggregated.
overallByProviders[].grades.Cleanliness	`hotel_provider_grades.score` with `category_name='Cleanliness'`	numeric(3,1)	Score for cleanliness.
overallByProviders[].grades.Facilities	`hotel_provider_grades.score` with `category_name='Facilities'`	numeric(3,1)	Score for facilities.
overallByProviders[].grades.Location	`hotel_provider_grades.score` with `category_name='Location'`	numeric(3,1)	Score for location.

JSON field	Target table.column	Data type / notes	Reasoning
overallByProviders[].grades.Room comfort and quality	`hotel_provider_grades.score` with `category_name='Room comfort and quality'`	numeric(3,1)	Score for comfort/quality.
overallByProviders[].grades.Service	`hotel_provider_grades.score` with `category_name='Service'`	numeric(3,1)	Service score.
overallByProviders[].grades.Value for money	`hotel_provider_grades.score` with `category_name='Value for money'`	numeric(3,1)	Value-for-money score.

Fields not mapped

JSON field	Reason for exclusion
`comment.formattedRating`, `comment.formattedReviewDate`	These are human-readable string versions of numeric/date fields (`rating` and `reviewDate`). Storing them separately would duplicate information. Applications can format numeric/date fields when displaying data.
`comment.reviewProviderLogo`	The URL of a logo image is not required for analysis or reporting. It could be stored in a separate assets table if needed later.
`overallByProviders[].provider`	The provider name is already stored in the `providers` table, avoiding redundancy.

Summary

The schema decomposes the JSON into six tables (`hotels`, `providers`, `reviewers`, `reviews`, `hotel_provider_overall`, and `hotel_provider_grades`). By normalising the data, we retain data integrity and enable efficient querying while still being able to reconstruct the original structure when needed. The design chooses not to store some duplicated fields (formatted strings and image logos) because they are either derived from existing data or not essential for analytical queries.