3NF Proof

Jingkai Zheng

April 2023

Park

Consider the relation of schema Park is R(park_code, park_name, state, acres, latitude, longitude), where park_code is the primary key. Consider there is no dependencies between non-key attributes, The minimum cover of functional dependencies are the following:

$$F = \{ \begin{array}{l} park\_code \rightarrow park\_name, \\ park\_code \rightarrow state, \\ park\_code \rightarrow acres, \\ park\_code \rightarrow latitude, \\ park\_code \rightarrow longitude \end{array} \}$$

The key is park_code. It is in both BCNF and 3NF because park_code is a superkey for R.

Species

Consider the relation of schema Species is R(species_id, park_name, category, order_name, family, scientific_name , common_names, record_status, occurrence, nativeness, abundance, seasonality , conservation_status), where species_id is the primary key. The minimum cover of functional dependencies are the following:

$$F = \{ \begin{array}{l} species\_id \rightarrow park\_name, \\ species\_id \rightarrow scientific\_name, \\ scientific\_name \rightarrow common\_names, \\ scientific\_name \rightarrow family, \\ family \rightarrow order\_name, \\ order\_name \rightarrow category, \\ species\_id \rightarrow record\_status, \\ species\_id \rightarrow occurrence, \\ species\_id \rightarrow nativeness, \\ species\_id \rightarrow abundance, \\ species\_id \rightarrow seasonality, \\ species\_id \rightarrow conservation\_status \end{array} \}$$

Since there are dependencies between non-key attributes like $\{scientific\_name \rightarrow common\_names, scientific\_name \rightarrow family, family \rightarrow order\_name, order\_name \rightarrow category\}$, we can decompose these transitive dependencies into smaller schema in order to achieve 3NF such as the following:

$$\begin{aligned}&R(\underline{scientific\_name}, common\_names)\\&R(\underline{scientific\_name}, family)\\&R(\underline{family}, order\_name)\\&R(\underline{order\_name}, category)\end{aligned}$$

However, for all our species-related queries, we want to retrieve all kinds of information regarding a specific species, including its scientific name, common names, family, order name, and category. Therefore, if we separate these non-key attributes into separate schema in order to satisfy 3NF, our query runtime will unnecessarily increase because we would need to join all these smaller tables together for each query. Therefore, we decided to follow our original schema for species in this case to improve query performance, even though the schema does not satisfy 3NF.

Trail

Consider the relation of schema Trail is R(trail_id, park_code, name, park_name, city_name, state_name, country_name, popularity, length, elevation_gain, difficulty_rating, route_type, avg_rating, num_reviews, features, activities, latitude, longitude), where trail_id is the primary key. The minimum cover of functional dependencies are the following:

$$F = \left\{ \begin{array}{l} trail\_id \rightarrow park\_code,\\ trail\_id \rightarrow name,\\ park\_code \rightarrow park\_name,\\ park\_name \rightarrow city\_name,\\ city\_name \rightarrow state\_name,\\ state\_name \rightarrow country\_name,\\ trail\_id \rightarrow popularity,\\ trail\_id \rightarrow length,\\ trail\_id \rightarrow elevation\_gain,\\ trail\_id \rightarrow difficulty\_rating,\\ trail\_id \rightarrow route\_type,\\ trail\_id \rightarrow avg\_rating,\\ trail\_id \rightarrow num\_reviews,\\ trail\_id \rightarrow features,\\ trail\_id \rightarrow activities,\\ trail\_id \rightarrow latitude,\\ trail\_id \rightarrow longitude \end{array} \right\}$$

Since there are dependencies between non-key attributes like $\{park\_code \rightarrow park\_name, park\_name \rightarrow city\_name, city\_name \rightarrow state\_name, state\_name \rightarrow$

$country\_name\}$, we can decompose these transitive dependencies into smaller schema in order to achieve 3NF such as the following:

$$\begin{array}{l} R(\underline{park\_code}, park\_name) \\ R(\underline{park\_name}, city\_name) \\ R(\underline{city\_name}, state\_name) \\ R(\underline{state\_name}, country\_name) \end{array}$$

However, for all our trail-related queries, we want to retrieve all kinds of information regarding a trail, including its park name, city name, state name and country name. Therefore, if we separate these non-key attributes into separate schema in order to satisfy 3NF, our query runtime will unnecessarily increase because we would need to join all these smaller tables together for each query. Therefore, we decided to follow our original schema for Trail in this case to improve query performance, even though the schema does not satisfy 3NF.

Airbnb

Consider the relation of schema Airbnb is R(id, name, host_id, host_name, neighbourhood_group, neighbourhood, latitude, longitude, room_type, price, minimum_nights, number_of_reviews, last_review, reviews_per_month, calculated_host_listings_count, availability_365, number_of_reviews_ltm, license, state, city), where id is the primary key. The minimum cover of functional dependencies are the following:

$$F = \{ \begin{array}{l} id \rightarrow name, \\ id \rightarrow host\_id, \\ host\_id \rightarrow host\_name, \\ latitude, longitude \rightarrow neighbourhood\_group, \\ latitude, longitude \rightarrow neighbourhood, \\ id \rightarrow latitude, \\ id \rightarrow longitude, \\ id \rightarrow room\_type, \\ id \rightarrow minimum\_nights, \\ id \rightarrow number\_of\_reviews, \\ id \rightarrow last\_review, \\ id \rightarrow reviews\_per\_month, \\ id \rightarrow calculated\_host\_listings\_count, \\ id \rightarrow availability\_365, \\ id \rightarrow number\_of\_reviews\_ltm, \\ id \rightarrow license, \\ id \rightarrow state, \\ latitude, longitude \rightarrow city \end{array} \}$$

Since there are dependencies between non-key attributes like $\{latitude, longitude \rightarrow neighbourhood\_group; latitude, longitude \rightarrow neighbourhood; latitude, longitude \rightarrow city\}$, we can decompose these transitive dependencies into smaller schema in order to achieve 3NF such as the following:

$$R(\underline{latitude}, \underline{longitude}, neighbourhood\_group)$$
$$R(\underline{latitude}, \underline{longitude}, neighbourhood)$$
$$R(\underline{latitude}, \underline{longitude}, city)$$

However, for all our Airbnb-related queries, we want to retrieve all kinds of information regarding an Airbnb, including its neighbourhood, neighbourhood group, and city. Therefore, if we separate these non-key attributes into separate schema in order to satisfy 3NF, our query runtime will unnecessarily increase because we would need to join all these smaller tables together for each query. Therefore, we decided to follow our original schema for Airbnb in this case to improve query performance, even though the schema does not satisfy 3NF.