WSU-CPTS-415

Charles Nguyen – 011606177

Homework 1

1. Volume, Velocity, Variety, Veracity, Value

Example: Atmospheric Radiation Measurement (ARM) user facility at the Pacific Northwest National Lab (PNNL). I worked with a team of scientists there during my Capstone to investigate the possibility of writing API for querying on a data collection of 1000TB.

* Volume: 1000TB
* Velocity: Near real-time; a time series is so large that they must be computed on cloud server and only results are sent back to user.
* Variety: Data is mostly structured, but some part was unstructured due to both old and new measurement devices.
* Veracity: Data is trusted but the degrees of accuracy and precision are completely dependent on measurement devices.
* Value: Highly meaningful meteoric information that helps shape socio-economic policies for the Pacific Northwest and the nation at large.
  1. Terms:
* **Relation schema**: informally a data table; formally a set of attributes each with domain
  + `airports` is a *relation schema*, wherein its attributes are `AirportID`, `Name` and so on.
* **Relational database schema**: the layout that reflects the relations among multiple *relation schemas*; transcendent over *relation schema*
  + `airports`, `airline ` and `route ` compose a *relational database schema*.
* **Domain**: a range of all possible values, which itself must be defined with a specific data type and format with the possibility for NULL value.
  + `Latitude` and `Longitude` both belong to one domain and are both a double type and cannot be NULL.
* **Attribute**: informally a column in the table; formally a particular feature in the set of attributes
  + `Name`, `City`, `Altitude` and so on are all *attributes* of the *relation schema* `airports`.
* **Attribute domain**: defines the role of a domain in a *relation schema*.
  + `Latitude` and `Longitude` are independent attributes with *attribute domains* orthogonal to each other.
* **Relation instance**: informally a set of rows of selected attributes in the table; formally an ordered -tuple of values, each derived from a proper domain.
  + `FROM airports SELECT AirportID, Name, City, Country INTO location OFFSET 5;`

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | Goroka Airport | Goroka | Papua New Guinea |
| 2 | Madang Airport | Madang | Papua New Guinea |
| 3 | Mount Hagen Kagamuga Airport | Mount Hagen | Papua New Guinea |
| 4 | Nadzab Airport | Nadzab | Papua New Guinea |
| 5 | Port Moresby Jacksons International Airport | Port Moresby | Papua New Guinea |

* 1. **Relational database schema** and **Relation schemas** for `airports`, `airline` and `route`:
  + Airport:
    - AirportID (PK)
    - Name
    - City
    - Country
    - IATA
    - ICAO
    - Latitude
    - Longitude
    - Altitude
    - Timezone
    - DST
    - Tz database time zone
    - Type
    - Source
  + Airline:
    - AirlineID (PK)
    - Name
    - Alias
    - IATA
    - ICAO
    - Callsign
    - Country
    - Active
  + Route:
    - Airline
    - AirlineID (FK=Airline.AirportID, PK1)
    - SourceAirport
    - SourceAirportID (FK=Airport.AirportID, PK2)
    - DestinationAirport
    - DestinationAirportID (FK=Airport.AirportID, PK3)
    - Codeshare
    - Stops
    - Equipment

Functional dependencies: all primary keys should have full functional dependencies over the remaining attributes in the table. In other words, given a foreign key (which is also a primary key pointing to another table), a query should be able to retrieve at most all information from the second table. For example, a query in Route using the foreign key AirlineID should be able to extract any and all information in the table Airline.

1. Functional Dependencies:
   1. Armstrong’s:
      1. Reflexive: Given the primary key AirlineID and we select some columns into a new table, all instances in the new table should have the same values as all instances in the table Airline.
      2. Augmentation: In the table Route, using AirlineID to query only Name and Alias into a new table NameAlias. Using AirlineID again to query Callsign, Country and Active into a new table CCA. Every single attribute in either is fully dependent on AirlineID. Therefore, joining the two table makes a new table that is a proper subset of Airline.
   2. Proofs:
      1. Decomposition**: if then: and** .

From *a.ii.* above, we can see that the table can be decomposed back into and by making appropriate queries from .

* + - 1. (Reflexive on )
      2. (Transitive on 1 and 2)
    1. Pseudo transitivity: **if and then:** .
       1. (Augmenting W to )
       2. (Applying transitivity on 3 and 2)

1. Normalization

Given:

3NF and BCNF:

* Using augmentation on :
  + (decomposition)
  + (3NF)
  + Given , we have: in BCNF
* Using augmentation on :
  + (decomposition)
  + (3NF)
  + Given , we have: in BCNF
* Using augmentation on :
  + Already in so we discard.