Rusu Wu
1164764
Assignment 1

1. Big data application: Amazon

Five big V's:

Volume: The amount of data of Amazon is horrendously large. The number of products, the information size of sellers or buyer and et.al are larger beyond our imagination.

Velocity: In order to provider good experiences for sellers and purchase, the speed of data entering a solution should be as fast as possible.

Variety: Amazon contain many different data source and different types, such as the information of products, which includes text, pictures, video and so on. More over, the information of buyers also vary form sellers

Veracity: The quality and accuracy of data should be guaranteed in Amazon, and Amazon should avoid generate dirty data for the well run of business.

Value: The collected data of buyers' behavior can generate grate value. Amazon's collected data not only can help buyers making decisions, but also it is benefit to the business.

I would use relational model to design its data base because the data in Amazon are well relational, such as "Customer-buy-product-seller".

2. **(a)**

Relation schema: Airport(Airport ID, Name, City, country....,Source)

Attribute: The column headers are the attributes. In this case, the table has 14 attributes including Airport ID, Name, City, country and so on.

Attribute domain: Each Attribute has domain which defines its logical definition and data-type or format. For example, attribute ICAO is defined as a 4-latter code.

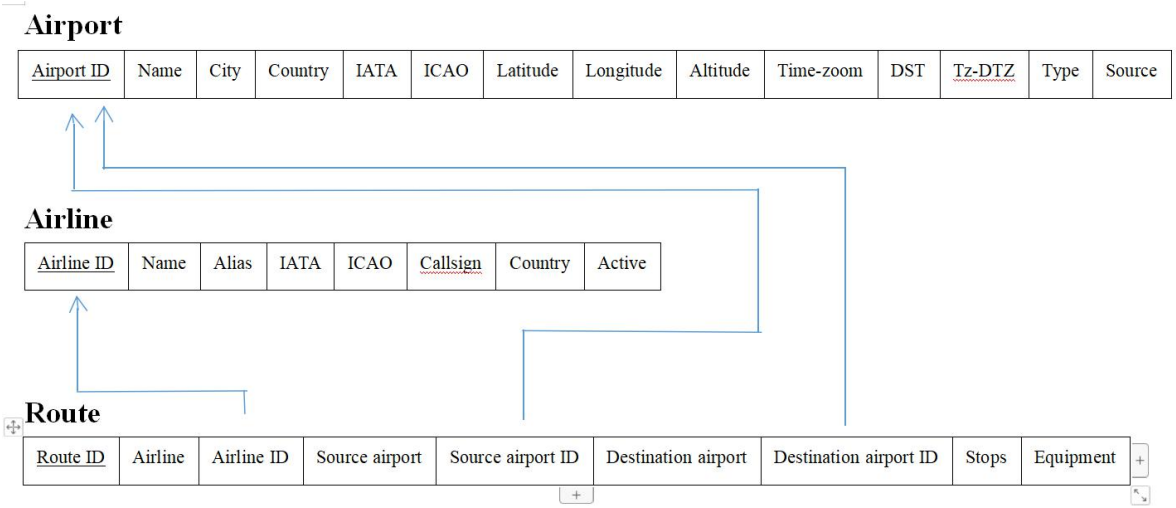
Relation instance: A relation instance is a tuple in a relation. In this case, a particular combination of airport's attribute value is one relation instance.

Airport

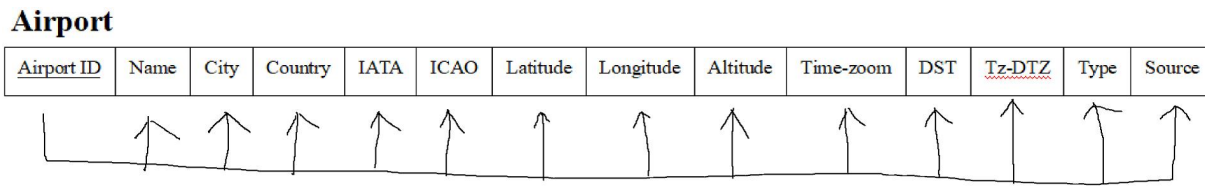
Airport ID	Name	City	Coun-try	IATA	ICAO	Latitud e	Longitude	Alti-t ude	Time zoom	DST	Tz-DTZ	Type	Source
3467	Spokane International Airport	Spoka ne	US	GEG	KGE G	47.619 899749 75586	-117.5339965 8203125	2376	-8	US/Can nada	Unknow	Airport	OpenFlights
3370	Guangzhou Baiyun International Airport	Guang zhou	CHN	ZGG G	ZGG G	23.392 400741 57715	113.29900360 107422	50	8	Unkno w	Unknow	Airport	OpenFlights
3577	Seattle Tacoma International Airport	Seattle	US	SEA	KSE A	47.449 001	-122.308998	433	-8	US/Can nada	Unknow	Airport	OpenFlights

3484	Los Angeles International Airport	Los Angeles	US	LAX	KLAX	33.94250107	-118.4079971	125	-8	US/Canada	Unknow	Airport	OpenFlights

(b)



Each row of attributes can be seen as a schema.
Primary key of each schema is underlined.
Foreign key constrain is shown as directed arc from Foreign key to referenced table.
FDs:



The airport schema has 13 FDs because there will be no unique attribute value if we remove the airport ID. In the similar way we can detect the FDs of Airline schema and Route schema .

(c)

If $Y \subseteq X$, then $X \rightarrow Y$

Airport(Name+ City +Country...) → City

If X→Y, then XZ→YZ

Airport ID→Airport Name

Airport ID+City=Airport Name+City

IF X→Y,and Y→Z, then X→Z

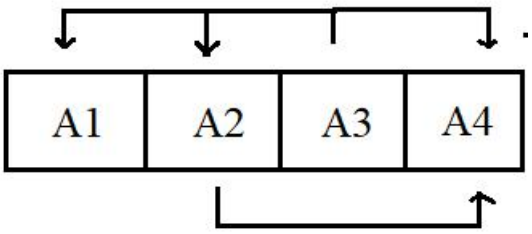
Given a specific Route ID

Route ID→Airline

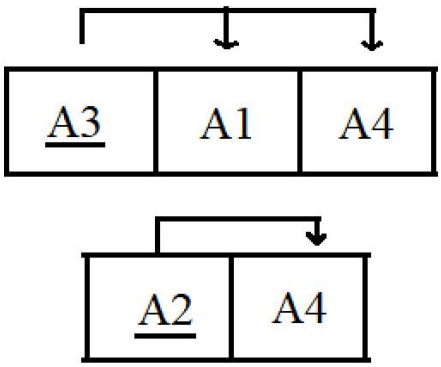
Airline→Country

We can say Route ID→Country

(d)



A table is in third normal form (3NF) when It contains no transitive dependencies. So, since A3 is the primary key, and there is one transitive dependence form A2 to A4, we can convert it to follow one.



3.

Q1: $\pi_{Theater}(\sigma_{Title=Zootopia}(Schedule))$

Q2:

$\pi_{Title, Address}(\gamma_{Director=Steven\ Spielberg}(Movie \bowtie_{title} Schedule \bowtie_{title} Location))$

Q3: $\pi_{Theater, Address, Phonenumber}(\sigma_{Theater=LeChampo}(Location))$

Q4:

$\rho(Movie', Movie)$

$\rho(Movie'(3 \rightarrow Actor2))$

$\pi_{Actor, Actor2}(\gamma_{Actor!=Actor2}(Movie' \bowtie_{title} Movie))$

4.(a)

for each block B_R of R

for each block B_s of S

for each tuple t_R in B_R

for each tuple t_S in B_s

test if pair (t_R ,t_S) satisfy the join condition R.A=S.B

end for

end for

end for

end for

R=100,000, B_R =10,000

S=20000, B_s 2000

Worst case, B_R * B_s + B_R =20,010,000 transfer, 2 B_R seeks= 20000

Best case, B_s + B_R =12000 transfer, 2seeks

Not best not worst: If the memory can hold 52 blocks

Transfer= B_R + B_R *(B_s -51)+51+ B_R =10,000 × (2,000 − 51) + 51 + 10,000

= 19,500,051

Seeks=2 B_R =20,000

(b)

1. Use N blocks of memory to buffer input runs, and 1 block to buffer output. Read the first block of each of the M runs into its buffer page (N block in total)
2. repeat
   Select the first record (in sort order) in each block (N in total) among all buffer pages
   Write the record to the output buffer. If the output buffer is full write it to disk.
   Delete the record from its input buffer page.
   If the buffer page becomes empty then read the next block (if any) of the run into the buffer.
3. until all input buffer pages are empty

M=52

Transfers= B_R (2 $\log_{M-1}$ (B_R /M) + 2)=10,000*(2log(51)(10,000/52)+2)≈46751

Seeks=2⌈ B_R /M⌉ +⌈ B_S / B_b ⌉ (2 $\log_{M-1}$ (B_R /M) − 1)≈453

(c)

Partition the relation s and using hashing function *h*

For each partition in both r and s

Load s_i into memory and build an in-memory hash index on it using the join attribute. This hash index using a different hash function *h'* than the earlier *h*.

Sequentially read the tuples in r_i from the disk one by one, for each tuple locate each matching tuple in s_i using the in-memory hash index, output the concatenation of their attributes.

One block of memory is reserved for loading r_i .

If recursive partitioning is not required (s_i <= mem buffer), cost of hash join is:

Transfers=3($B_R + B_s$)=3*12000=36000

Seeks=2(B_R / B_b + B_s / B_b)=2(10,000 / 50 + 2,000/50) = 480