Cheat sheet

Wednesday, October 27, 2021 1:27 AM

Functional Dependency

Given
A → B, C
D → C
E → D
B, E → C

Closure
$\{A, E\}^+ = \{A, B, C, D, E\}$ ← Superkey (because the closure contains all keys)
Key is when none of the subsets are superkey

Armstrong's Axiom: 1
Reflexivity

Axiom: 2
Augmentation
- For any attribute sets X, Y, Z
  - If X -> Y, then X, Z -> Y, Z

Axiom: 3
Transitivity
- For any attribute sets X, Y, Z
  - If X -> Y, and Y -> Z, then X -> Z

DCNF: For all non-trivial *X* → *B*, X must be a superkey

BCNF Example

Books(author, gender, title, genera, price)
- author -> gender
- title -> genre, price

Key?
{author, title}

Is BCNF?
No, bc the left hand side of both FDs are not superkey

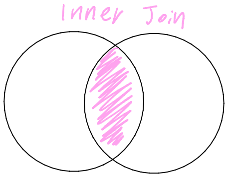
Split!

Split Books using author ->gender
- **Author**(author, gender)
  - FD: author -> gender in BCNF
- **Books2**(author, title, genre, price)
  - FD: title -> genre, price not in BCNF

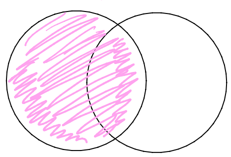
Split again!
- BookInfo(title, genre, price)
  - FD: title -> genre, price in BCNF
- BookAuthor(author, title) in BCNF

SQL

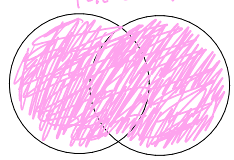
Inner Join



Left Outer



full outer join



MovieID	MovieName
1	Star Wars
2	Sholay
3	The Italian Job

MovieID	CinemaName	RunForDays
2	Naz Cinema	101
5	Apollo Theater	45

Cross Join

MovieID	MovieName	MovieID	CinemaName	RunForDays
1	Star Wars	2	Naz Cinema	101
1	Star Wars	5	Apollo Theater	45
2	Sholay	2	Naz Cinema	101
2	Sholay	5	Apollo Theater	45
3	The Italian Job	2	Naz Cinema	101
3	The Italian Job	5	Apollo Theater	45

Inner

MovieID	MovieName	MovieID	CinemaName	RunForDays
2	Sholay	2	Naz Cinema	101

Left Inner

MovieID	MovieName	MovieID	CinemaName	RunForDays
1	Star Wars	NULL	NULL	NULL
2	Sholay	2	Naz Cinema	101
3	The Italian Job	NULL	NULL	NULL

Full Outer

MovieID	MovieName	MovieID	CinemaName	RunForDays
1	Star Wars	NULL	NULL	NULL
2	Sholay	2	Naz Cinema	101
3	The Italian Job	NULL	NULL	NULL
NULL	NULL	5	Apollo Theater	45

Sample queries

for each country the population of the most populated city, only for countries with at least 10 cities

```
SELECT C.Name, MAX(T.Population)
FROM City as T, Country as C
WHERE T.CountryCode = C.Code
GROUP BY C.Name
HAVING COUNT(T.ID) >= 10
```

Get all countries in Europe that have all cities with <1m population

```
SELECT C.Name
FROM Country C
WHERE C.Continent = 'Europe'
AND NOT EXISTS (SELECT *
                FROM City T
                WHERE T.Population > 1000000
                AND T.CountryCode = C.Code);
```

Inserting into Table

```
INSERT INTO Users (email, bio, country)
VALUES (
  "email@gmail.com",
  "Student",
  "US"
)
```

← OR →

```
INSERT INTO Users (email, bio, country)
VALUES
  ("email@gmail.com", "Student", "US")
  ("email@hotmail.com", "Teacher", "CA")
  ("email@apple.com", "a long string…", "CN")
```

Row Number, Find longest sequence

```
WITH num AS
   (SELECT A, (A-row_number() OVER (ORDER BY A ASC)) AS B FROM R)
SELECT COUNT(*) AS len
FROM num
GROUP BY B
ORDER BY len DESC
```

View

```
CREATE VIEW SomeViewName AS
   -- Standard query
```

Buffer Manager

Clock

- When a frame is considered:
  - If pin count > 0, increment current
  - If referenced = 1, set to 0 and increment
  - If referenced = 0 and pin count = 0, choose the page
  - Each frame has a referenced bit that is set to 1 when pin count becomes 0
  - A current variable points to a frame

LRU

- Uses a queue of pointers to frames that have pin count = 0
- A page request uses frames only from the head of the queue
- When the pin count of a frame goes to 0, it is added to the end of the queue

access time = rotational delay + seek time + transfer time

7200 RPM = 4.17 ms Rotational delay
4 KB / 210 MB/s = 0.014 ms Transfer time
4.17 + 0.014 + 9 = 13.1 ms ← Total time
4 KB / 13.1 ms = 0.3 MB/s I/O Rate

Relational Algebra

SELECTION
$\sigma_C(R)$

PROJECTION
$\pi_{A1,A2,\ldots,An}(R)$

UNION
$R_1 \cup R_2$

- Outputs all tuples in R_1 or R_2
- Both relations mush have the same schema
- Output schema: same as input

DIFFERENCE
$R_1 - R_2$

- Outputs all tuples in R_1 and not in R_2
- Both relations mush have the same schema
- Output schema: same as input

CROSS-PRODUCT
$R_1 \times R_2$

- Matches each tuples in R_1 with each tuple in R_2
- Input schema: $R_1(A_1,\ldots,A_n)$, $R_2(B_1,\ldots,B_m)$
- Output schema: $R(A_1,\ldots,A_n,B_1,\ldots,B_m)$
- CANNOT have the same named columns

DIVISION
R_1 / R_2

- Suppose $R_1(A, B)$ and $R_2(B)$
- The output contains all values **a** such that for every tuple **(b)** in R_2, tuple **(a, b)** is in R_1
- Output schema: R(A)

How to derive

$\pi_A(R) - \pi_A((\pi_A(R) \times S) - R)$

A	B
a_1	b_1
a_1	b_2
a_1	b_3
a_2	b_1

B_2
b_1

A/B_2 =

A
a_1
a_2

NATURAL JOIN
$R_1 \bowtie R_2$

- Equi-join on all the common fields
- The output schema has on copy of each common attribute

JOIN (THETA JOIN)
$R_1 \bowtie_\theta R_2 = \sigma_\theta(R_1 \times R_2)$

- Cross-product followed by a selection
- Θ can be any boolean-valued condition
- Might have less tuples than the cross-product

- Article(artid, title, confid, numpages)
- Conference(confid, name, year, location)
- Author(artid, pid)
- Person(pid, name, affiliation)

Name of people affiliated with UW who submitted an article in 2019

$\pi_{name}(\sigma_{affiliation="UW-Madison"}(Person) \bowtie Author \bowtie Article \bowtie \pi_{confid}(\sigma_{year=2019}(Conference)))$

Output the names of the people who coauthored an article with `John Doe`. Be careful: a person cannot be coauthor with herself!

$\pi_{oname}(\sigma_{name="JohnDoe"}(Person) \bowtie Author \bowtie \rho_{pid \to oid}(Author) \bowtie \rho_{pid \to oid, name \to oname} \sigma_{name \neq "JohnDoe"}(Author))$