What is Data Retrieval?

SDSC SAN DIEGO SUPERCOMPUTER CENTER

What is Data Retrieval?

- **Data retrieval**
  - The way in which the desired data is specified and retrieved from a data store
- **Our focus**
  - How to specify a data request
    - For static and streaming data
  - The internal mechanism of data retrieval
    - For large and streaming data

What is a Query Language?

- A language to specify the data items you need

- A query language is declarative
  - Specify what you need rather than how to obtain it
  - SQL (Structured Query Language)
- Database programming language
  - Procedural programming language
  - Embeds query operations

SQL

- **The standard for structured data**
  - Oracle's SQL to Spark SQL
- **Example Database Schema**
  - **Bars(name, addr, license)**
  - **Beers(name, manf)**
  - **Sells(bar, beer, price)**
  - **Drinkers(name, addr, phone)**
  - **Frequents(drinker, bar)**
  - **Likes(drinker, beer)**

name	addr	license
Great American Bar	363 Main St., SD, CA 92390	41-437844098
Beer Paradise	6450 Mango Drive, SD, CA 92130	41-973428319
Have a Good Time	8236 Adams Avenue, SD, CA 92116	32-032263401

SELECT-FROM-WHERE

- **Which beers are made by Heineken?**

SELECT *name*
FROM *Beers*
WHERE *manf = 'Heineken'*

Output attribute(s)

Table(s) to use

The condition(s) to satisfy

Strings like 'Heineken' are case-sensitive and are put in quotes

$Select_{manf='Heineken'} (Beers)$

↓

$Project(name)$

name
Heineken Lager Beer
Amstel Lager
Amstel Light
...

More Example Queries

- **Find expensive beer**
  - SELECT DISTINCT beer, price
  - FROM Sells
  - WHERE price > 15

- **Which businesses have a Temporary License (starts with 32) in San Diego?**
  - SELECT name
  - FROM Bars
  - WHERE addr LIKE '%SD%' **AND** license LIKE '32%' LIMIT 5

name	addr	license
Great American Bar	363 Main St., SD, CA 92390	41-437844098
Beer Paradise	6450 Mango Drive, SD, CA 92130	41-973428319
Have a Good Time	8236 Adams Avenue, SD, CA 92116	32-032263401

Select-Project Queries in the Large

- **Large Tables can be partitioned**
  - Many partitioning schemes
    - Range partitioning on primary key

name	manf
A...	Gambrinus
A...	Heineken
...	
B...	Anheuser-Busch

Machine 1

name	manf
C...	MillerCoors
C...	MillerCoors
...	
D...	Duvel Moortgat

Machine 2

...

name	manf
H...	Heineken
H...	Pabst
...	
H...	Anheuser-Busch

Machine 5

...

Select-Project Queries in the Large

name	manf
A...	Gambrinus
A...	Heineken
...	
B...	Anheuser-Busch

Machine 1

name	manf
C...	MillerCoors
C...	MillerCoors
...	
D...	Duvel Moortgat

Machine 2

...

name	manf
H...	Heineken
H...	Pabst
...	
H...	Anheuser-Busch

Machine 5

...

SELECT *
FROM Beers
WHERE name like 'Am%'

pattern

SELECT name
FROM Beers
WHERE manf = 'Heineken'

- **Two queries**
  - Find records for beers whose name starts with 'Am'
  - Which beers are made by Heineken?

Evaluating SP Queries for Large Data

name	manf
A...	Gambrinus
A...	Heineken
...	
B...	Anheuser-Busch

Machine 1

name	manf
C...	MillerCoors
C...	MillerCoors
...	
D...	Duvel Moortgat

Machine 2

...

name	manf
H...	Heineken
H...	Pabst
...	
H...	Anheuser-Busch

Machine 5

...

*SELECT ***
*FROM *Beers
*WHERE *name like 'Am%'

- **A query processing trick**
  - Use the partitioning information
    - Just use partition 1!!

Evaluating SP Queries for Large Data

name	manf
A...	Gambrinus
A...	Heineken
...	
B...	Anheuser-Busch

Machine 1

name	manf
C...	MillerCoors
C...	MillerCoors
...	
D...	Duvel Moortgat

Machine 2

...

name	manf
H...	Heineken
H...	Pabst
...	
H...	Anheuser-Busch

Machine 5

...

SELECT name
FROM Beers
WHERE manf = 'Heineken'

Broadcast query

In each machine in parallel:
Select $_{manf='Heineken'}$ (Beers)
Project(name)

Gather Partial Results
Union
Return

Local and Global Indexing

- **What if a machine does not have any data for the query attributes?**

- **Index structures**
  - Given value, return records
  - Several solutions
    - Use local index on each machine
    - Use a machine index for each value
    - Use a combined index in a global index server

manf	RecordIDs
...	...
MillerCoors	34, 35, 87, 129, ...
Duvel Moortgat	5, 298, 943, 994,...
Heineken	631, 683, 882,...
...	...

manf	machineIDs
...	...
MillerCoors	10
Duvel Moortgat	3, 4
Heineken	1, 3, 5
...	...

Pause

Querying Two Relations

- **Often we need to combine two relations for queries**
  - Find the beers liked by drinkers who frequent The Great American Bar

- **In SQL**

- **SELECT DISTINCT beer**
- **FROM Likes L, Frequents F**
- **WHERE bar = The Great American Bar' AND**
- **F.drinker = L.drinker**

Frequents(drinker, bar)
Likes(drinker, beer)

SPJ Queries

SELECT DISTINCT beer
FROM Likes L, Frequents F
WHERE bar = 'The Great American Bar'
    AND F.drinker = L.drinker

- **Steps**

    Selection $_{bar = \text{'The Great American Bar'}}$ (Frequents)

    ⬇ ← *No intermediate storage*

    Join $_{F.drinker = L.drinker}$ (_, Likes)

    ⬇ ← *R(drinker, beer)*

    Project $_{beer}$(_)

    ⬇

    Deduplicate(_)

    ⬇

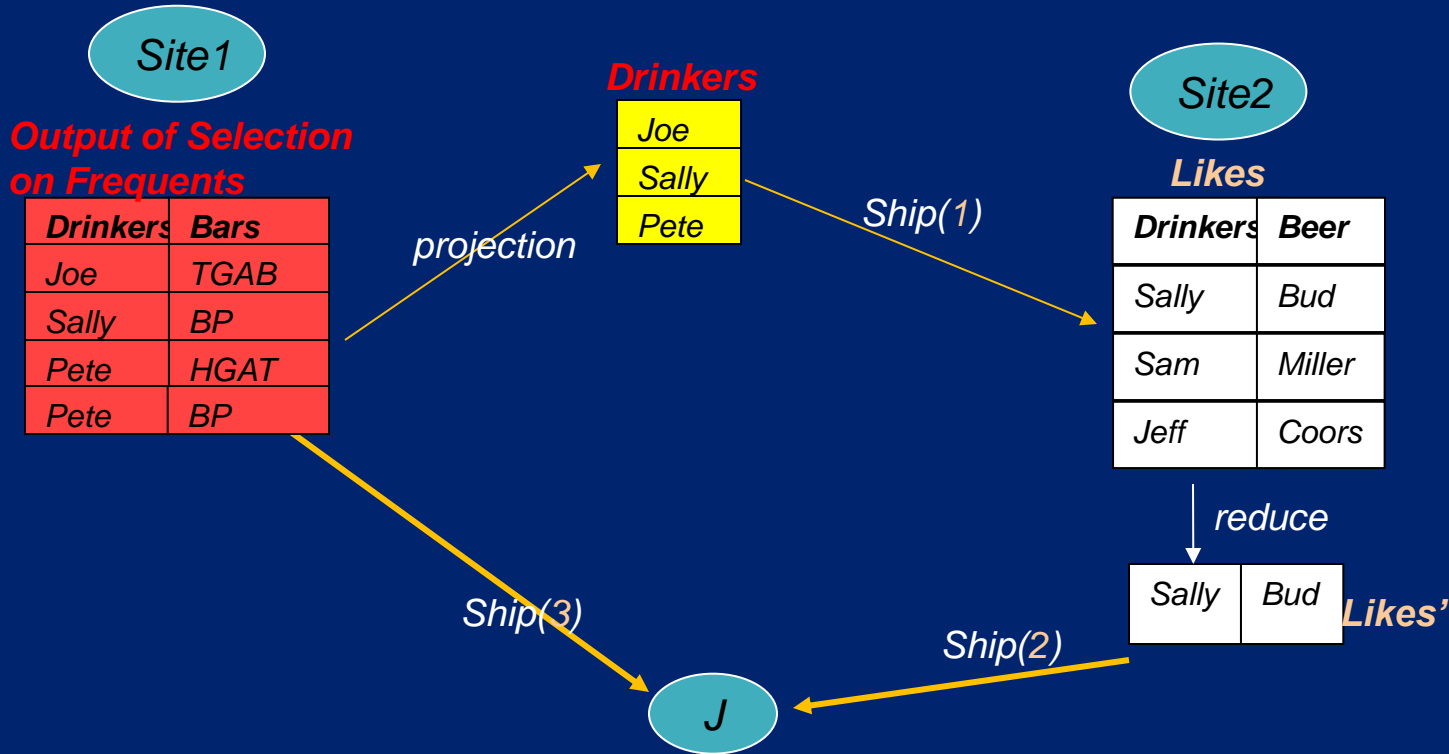   Output

Join in a Distributed Setting

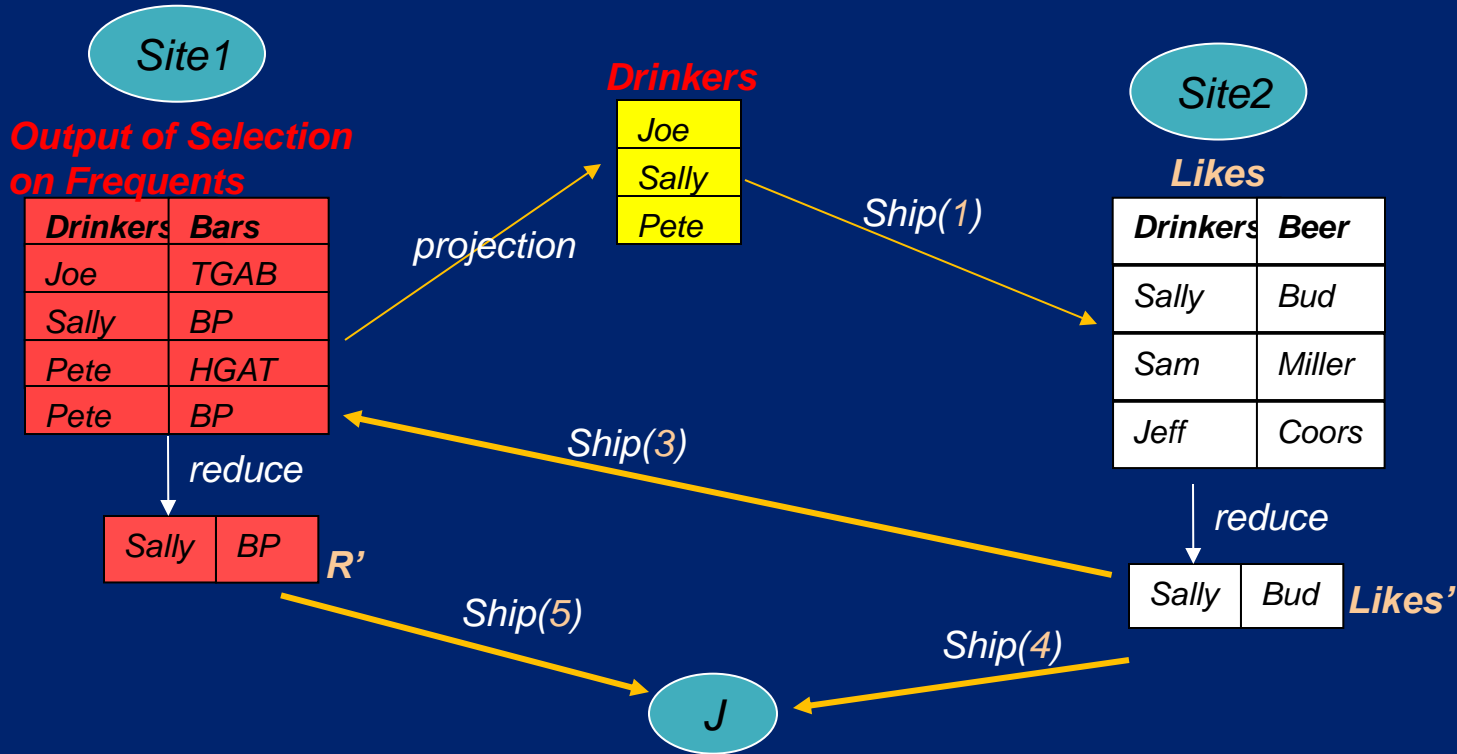Frequents(_drinker_, _bar_)
Likes(_drinker_, _beer_)

- ## Semijoin
  - A semijoin from R to S on attribute is used to reduce the data transmission cost
  - Computing steps:
    - **Project** R on attribute A and call it (R[A]) – the Drinkers column
    - **Ship** this projection (a semijoin projection) from the site of R to the site of S
    - **Reduce** S to S' by eliminating tuples where attribute A are not matching any value in R[A]

Semijoin s: Frequents—Drinkers →Likes

Site1

Output of Selection on Frequents

Drinkers	Bars
Joe	TGAB
Sally	BP
Pete	HGAT
Pete	BP

projection

Drinkers

Joe
Sally
Pete

Ship(1)

Site2

Likes

Drinkers	Beer
Sally	Bud
Sam	Miller
Jeff	Coors

reduce

Sally	Bud

Likes'

Ship(2)

Ship(3)

J

Semijoin s: Frequents—Drinkers →Likes

Site1

Output of Selection on Frequents

Drinkers	Bars
Joe	TGAB
Sally	BP
Pete	HGAT
Pete	BP

Drinkers

Joe
Sally
Pete

projection

Ship(1)

Site2

Likes

Drinkers	Beer
Sally	Bud
Sam	Miller
Jeff	Coors

reduce

| Sally | BP | R'

Ship(3)

reduce

| Sally | Bud | Likes'

Ship(5)

Ship(4)

J

Pause

Subqueries

- A slightly complex query
- Find the bars that serve Miller for the same or less price than what TGAB charges for Bud
- We may break it into two queries:
  1. Find the price TGAB charges for Bud
  2. Find the bars that serve Miller at that price

Subqueries in SQL

SELECT bar

FROM Sells

WHERE beer = 'Miller' AND

price <= (SELECT price

FROM Sells

WHERE bar = 'TGAB'

AND beer = 'Bud');

The price at which TGAB sells Bud

Subqueries with IN

- Find the name and manufacturer of each beer that Fred does not like

- Query

  Beers(*name*, *manf*)
  Likes(*drinker*, *beer*)

  SELECT *
  FROM Beers
  WHERE name NOT IN
                  (SELECT beer
                          FROM Likes
                          WHERE drinker = 'Fred');

Correlated Subqueries

Bar	Beer	Price
HGAT	Bud	5
BP	Michelob	4
TGAB	Heineken	6
HGAT	Guinness	10

- Find the name and price of each beer that is more expensive than the average price of beers sold in the bar

SELECT beer, price

FROM Sells s1

WHERE price >

(SELECT AVG(price)
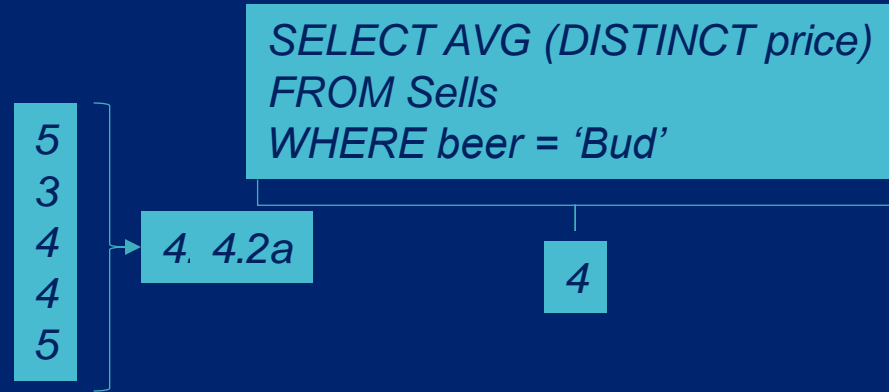
FROM Sells s2

WHERE s1.bar = s2.bar)

Aggregate Queries

- **Example**
  - Find the average price of Bud:
    - SELECT AVG(price)
    - FROM Sells
    - WHERE beer = 'Bud';

- **Other aggregate functions**
  - SUM, MIN, MAX, COUNT, ...

5
3
4
4
5

→ 4. 4.2a

SELECT AVG (DISTINCT price)
FROM Sells
WHERE beer = 'Bud'

4

GROUP BY Queries

- Find for each drinker the average price of Bud at the bars they frequent

SELECT drinker, AVG(price)

FROM Frequents, Sells

WHERE beer = 'Bud' AND

    Frequents.bar = Sells.bar

GROUP BY drinker;

Drinker	Bar	Price
Pete	HGAT	5
Pete	BP	4
Joe	TGAB	6
Joe	HGAT	5

Drinker	Price
Pete	4.5
Joe	5.5

Grouping Aggregates over Partitioned Data

Drinker	Bar	Price
Pete	HGAT	5
Pete	BP	4
Joe	TGAB	6
John	HGAT	5

Drinker	Bar	Price
Pete	HGAT	5
Pete	BP	4
Pete	BO	6
Joe	TGAB	6

Drinker	Price
Pete	5
Joe	6

Drinker	Bar	Price
Pete	BO	6
John	BP	4
Sally	TGAB	6
Sally	HGAT	5

Drinker	Bar	Price
John	HGAT	5
John	BP	4
Sally	TGAB	6
Sally	HGAT	5

Drinker	Price
John	4.5
Sally	5.5