CS 186 Discussion 2

Advanced SQL

Logistics

- Discussion / Vitamin Schedule
- Homework 2 due Thursday 2/11
  - Go to office hours early…

External Sorting…

- Q4: “How large of a file can we sort in k passes?”

External Hashing…

- Q6: Suppose you are hashing a file and one of the partitions is 36 MB after the first pass…

Vitamin SQL

- Q7: Number of songs released after 2010 with rating >= 9.0
  - SELECT COUNT(*) FROM Songs WHERE year_released > 2010 AND rating >= 9.0;
  - SELECT COUNT(*) FROM Songs GROUP BY year_released, rating HAVING year_released > 2010 AND rating >= 9.0;
  - SELECT COUNT(*) FROM Songs WHERE rating >= 9.0 GROUP BY year_released HAVING year_released > 2010;
  - SELECT COUNT(song_id) FROM Songs WHERE year_released > 2010 AND rating >= 9.0;

Vitamin SQL

- Q8: List of artists, without duplicates, with at least one song more than 5 minutes long?
  - SELECT DISTINCT artist_name FROM Songs WHERE length_seconds > 300;
  - SELECT artist_name FROM Songs WHERE length_seconds > 300 GROUP BY artist_name;
  - SELECT artist_name FROM Songs WHERE length_seconds > 300 GROUP BY artist_name, length_seconds HAVING COUNT(*) >= 1;
  - SELECT artist_name FROM Songs GROUP BY artist_name, length_seconds HAVING length_seconds > 300;

SQL Queries

SELECT [DISTINCT] <column list>
FROM <table1> AS A, <table2> AS B
WHERE <predicate>
GROUP BY <column list>
HAVING <predicate>
ORDER BY <column list>
LIMIT <amount>

Worksheet 1, 2, 3, 4

1. Total number of ‘Techno’ albums released each year:

Worksheet 1, 2, 3, 4

1. Total number of ‘Techno’ albums released each year:

```
SELECT    year_released, COUNT(*)
FROM      Albums
WHERE     genre = ‘Techno’
GROUP BY year_released;
```

Worksheet 1, 2, 3, 4

2. Number of albums per genre, ignoring genres with fewer than 10 albums:

Worksheet 1, 2, 3, 4

2. Number of albums per genre, ignoring genres with fewer than 10 albums:

```
SELECT    genre, COUNT(*)
FROM      Albums
GROUP BY genre
HAVING    COUNT(*) >= 10;
```

Worksheet 1, 2, 3, 4

3. For each song… song name, album name, and artist name:

Worksheet 1, 2, 3, 4

3. For each song… song name, album name, and artist name:

```
SELECT song_name, album_name, artist_name
```

Worksheet 1, 2, 3, 4

3. For each song… song name, album name, and artist name:

```
SELECT song_name, album_name, artist_name
FROM   Songs S, Albums A, Artists R
```

Worksheet 1, 2, 3, 4

3. For each song… song name, album name, and artist name:

```
SELECT song_name, album_name, artist_name
FROM   Songs S, Albums A, Artists R
WHERE  S.album_id = A.album_id
  AND  A.artist_id = R.artist_id;
```

Worksheet 1, 2, 3, 4

4. Singers with both ‘Pop’ and ‘Techno’:

Worksheet 1, 2, 3, 4

4. Singers with both ‘Pop’ and ‘Techno’:

```
SELECT   artist_name
FROM     Albums A1, Albums A2, Artists R
WHERE    R.artist_id = A1.artist_id
  AND    A1.artist_id = A2.artist_id
  AND    A1.genre = ‘Pop’
  AND    A2.genre = ‘Techno’;
```

Bonus

- Number of albums from each artist:

Bonus

- Number of albums from each artist:

```
SELECT    artist_name, COUNT(*)
FROM      Albums A, Artists R
WHERE     A.artist_id = R.artist_id
GROUP BY R.artist_id, artist_name;
```

Why do we need to group by artist_id?

SQL Joins

Inner/Natural Join

TABLE Students

SID	Name
23357852	Bob the Builder
24821529	Anonymous
27983421	Nicholas
27994495	Oskicat

TABLE Enrollment

CCN	SID
25227	23357852
26586	23357852
21545	21592421
23495	23374219
26586	27994495

SID	Name	CCN	*SID*
23357852	Bob the Builder	25227	*23357852*
23357852	Bob the Builder	26586	*23357852*
27994495	Oskicat	26586	*27994495*

- All ‘Country’ songs that have been in the top 40 for more than 2 weeks:

```
SELECT song_name
FROM   Songs S, Albums A
WHERE S.album_id = A.album_id
   AND  genre = ‘Country’
   AND  weeks_in_top_40 > 14;
```

Is this an inner join?

Inner/Natural Join

```
SELECT <column list>
FROM <table1> A, <table2> B
WHERE A.<column> = B.<column>
```

```
SELECT <column list>
FROM <table1> A INNER JOIN <table2> B
ON A.<column> = B.<column>
```

```
SELECT <column list>
FROM <table1> A NATURAL JOIN <table2> B
```

Outer Joins

SELECT <column list>

FROM A {LEFT, RIGHT, FULL} OUTER JOIN B

ON A.<column> = B.<column>

Left Outer Join

TABLE Students

SID	Name
23357852	Bob the Builder
24821529	Anonymous
27983421	Nicholas
27994495	Oskicat

TABLE Enrollment

CCN	SID
25227	23357852
26586	23357852
21545	21592421
23495	23374219
26586	27994495

SID	Name	CCN	SID
23357852	Bob the Builder	25227	23357852
23357852	Bob the Builder	26586	23357852
24821529	Anonymous		
27983421	Nicholas		
27994495	Oskicat	26586	27994495

Right Outer Join

TABLE Students

SID	Name
23357852	Bob the Builder
24821529	Anonymous
27983421	Nicholas
27994495	Oskicat

TABLE Enrollment

CCN	SID
25227	23357852
26586	23357852
21545	21592421
23495	23374219
26586	27994495

SID	Name	CCN	SID
23357852	Bob the Builder	25227	23357852
23357852	Bob the Builder	26586	23357852
		21545	21592421
		23495	23374219
27994495	Oskicat	26586	27994495

Full Outer Join

SID	Name	CCN	SID
23357852	Bob the Builder	25227	23357852
23357852	Bob the Builder	26586	23357852
24821529	Anonymous		
27983421	Nicholas		
		21545	21592421
		23495	23374219
27994495	Oskicat	26586	27994495

Worksheet 5

5. Artist name and album id for artists active since 2000 or later.
   - Include artists who do not have albums

Worksheet 5

5. Artist name and album id for artists active since 2000 or later.
   - Include artists who do not have albums.

```
SELECT artist_name, album_id
```

Worksheet 5

5. Artist name and album id for artists active since 2000 or later.
   - Include artists who do not have albums.

```
SELECT artist_name, album_id
FROM   Artists Ar
LEFT OUTER JOIN  Albums Ab
```

Worksheet 5

5. Artist name and album id for artists active since 2000 or later.
   - Include artists who do not have albums.

```
SELECT artist_name, album_id
FROM   Artists Ar
LEFT OUTER JOIN  Albums Ab
    ON   Ar.artist_id = Ab.artist_id
```

Worksheet 5

5. Artist name and album id for artists active since 2000 or later.
   - Include artists who do not have albums.

```
SELECT artist_name, album_id
FROM   Artists Ar
LEFT OUTER JOIN  Albums Ab
    ON   Ar.artist_id = Ab.artist_id
WHERE first_year_active >= 2000;
```

Six-Degrees of Separation

```
CREATE TABLE Friends(
  fromID integer,
  toID integer,
  since date,
PRIMARY KEY (fromID, toID),
  FOREIGN KEY (fromID)
    REFERENCES Users,
  FOREIGN KEY (toID)
    REFERENCES Users,
  CHECK (fromID < toID));
```

Six-Degrees of Separation

```
CREATE TABLE Friends(
  fromID integer,
  toID integer,
  since date,
PRIMARY KEY (fromID, toID),
  FOREIGN KEY (fromID)
     REFERENCES Users,
  FOREIGN KEY (toID)
     REFERENCES Users,
  CHECK (fromID < toID));
```

```
CREATE VIEW Edges AS
  …
```

Six-Degrees of Separation

```
CREATE TABLE Friends(
  fromID integer,
  toID integer,
  since date,
PRIMARY KEY (fromID, toID),
FOREIGN KEY (fromID)
   REFERENCES Users,
FOREIGN KEY (toID)
   REFERENCES Users,
CHECK (fromID < toID));
```

```
CREATE VIEW Edges AS
 (SELECT * FROM Friends
  UNION ALL
  SELECT F.toID AS fromID,
  F.fromID AS toID, F.since
  FROM Friends F);
```

Six-Degrees of Separation

```
CREATE TABLE Friends(
  fromID integer,
  toID integer,
  since date,
PRIMARY KEY (fromID, toID),
FOREIGN KEY (fromID)
   REFERENCES Users,
FOREIGN KEY (toID)
   REFERENCES Users,
CHECK (fromID < toID));
```

```
CREATE VIEW Edges AS
 (SELECT * FROM Friends
  UNION ALL
  SELECT F.toID AS fromID,
  F.fromID AS toID, F.since
  FROM Friends F);
```

```
SELECT F1.fromID, F5.toID
FROM Edges F1, Edges F2,
Edges F3, Edges F4, Edges F5
WHERE F1.toID = F2.fromID
    AND F2.toID = F3.fromID
    AND F3.toID = F4.fromID
    AND F4.toID = F5.fromID;
```

Median

T (c integer)

Median

T (c integer)

```
SELECT c AS median FROM T
WHERE
 (SELECT COUNT(*) FROM T AS T1
  WHERE T1.c <= T.c)
  =
(SELECT COUNT(*) FROM T AS T2
  WHERE T2.c >= T.c);
```

Median

T (c integer)

```
SELECT x.c AS median
FROM T x, T y
GROUP BY x.c
 HAVING
  SUM(CASE WHEN y.c <= x.c THEN 1 ELSE 0 END)
   >= (COUNT(*) + 1)/2
  AND
  SUM(CASE WHEN y.c >= x.c THEN 1 ELSE 0 END)
   >= COUNT(*)/2 + 1;
```

Median

T (c integer)

```
CREATE VIEW twocounters AS
 (SELECT c,
  ROW_NUMBER() OVER (ORDER BY c ASC) AS RowAsc
  ROW_NUMBER() OVER (ORDER BY c DESC) AS RowDesc
  FROM T);

SELECT MIN(c) AS median FROM twocounters
WHERE RowAsc IN (RowDesc, RowDesc - 1, RowDesc + 1);
```