Grouping and aggregate functions

Jaroslav Porubän, Miroslav Biňas, Milan Nosáľ (c) 2011 - 2016

Introduction

- Aggregate functions return a value computed from values in a column
  - They aggregate collection of values to a single value
  - `NULL` value is ignored (`NVL` function)
- You can use aggregate functions to
  - Count tuples/values
  - Minimum/maximum of values
  - Average value
  - Variance
  - Standard deviation
  - Sum

COUNT()

- Number (count) of tuples
- Usable on: tuple, text, number and date
- Examples:

```
SELECT COUNT(*) FROM fbuser;
SELECT COUNT(type)
    FROM post;
SELECT
    COUNT(DISTINCT type)
    FROM post;
```

AVG()

- Returns average value of values in a numeric column
- Usable on: numbers
- Examples:

```
SELECT AVG(floor((current_date -
birthday)/365))
FROM fbuser;
```

MIN() and MAX()

- Returns the biggest(maximum)/smallest(minimum) value in the given column
- Usable on: text, numbers and date
- Examples:

```
SELECT MAX(username) FROM
fbuser;
SELECT MIN(username) FROM fbuser
    WHERE sex='M';
```

SUM()

- Returns total sum of values in a column
- Usable on: numbers
- Examples:

```
SELECT SUM(floor((current_date -
birthday)/365)) FROM fbuser
    WHERE sex='M';

SELECT SUM(floor((current_date -
birthday)/365))
    FROM fbuser
    WHERE birthday < '01.01.90';
```

STDDEV() and VARIANCE()

- Statistical functions
  - Standard deviation - `STDDEV()`
  - Variance - `VARIANCE()`
- Usable on: numbers
- Examples:

```
SELECT
STDDEV(floor((current_date -
birthday)/365))
FROM fbuser WHERE sex='F';
SELECT VARIANCE(floor((current_d
ate - birthday)/365))
FROM fbuser;
```

Grouping

- Usually used with aggregate functions (aggregating values for groups)
- GROUP BY groups tuples according to the same value in the given column(s)
- syntax:

```
SELECT column_name,
aggregate_function(column_name)
FROM table_name
WHERE expr
GROUP BY column_name;
```

GROUP BY - example

- SELECT may contain only columns present in `GROUP BY`, or aggregate functions
- Examples:

```
SELECT type, count(*)
    FROM post
    GROUP BY type;
SELECT type, count(*),
    min(dateOfPost),
    max(dateOfPost)
    FROM post
    GROUP BY type;
```

HAVING

- Used in combination with `GROUP BY` to filter out tuples representing groups
  - `WHERE` filters tuples **before** they are grouped (can contain columns not present in `GROUP BY`)
  - `HAVING` filters tuples **after** grouping
- Examples:

```
SELECT type, count(*)
    FROM post
    GROUP BY type
    HAVING count(*) > 1;
```

ROLLUP

- Used with `GROUP BY` to produce group subtotals from right to left and a grand total
- Example (count of users with the same sex and age, subtotal for the same sex, grandtotal for all users):

```
SELECT sex,
   floor((current_date - birthday)/365),
   count(*)
FROM fbuser
GROUP BY ROLLUP (sex,
  floor((current_date - birthday)/365));
```

CUBE

- Used with `GROUP BY` to produce all subtotals and grandtotalfor grouping columns
- Example (count of users with the same sex and age, subtotal for the same sex, subtotal for the same age, grandtotal for all users):

```
SELECT sex,
   floor((current_date - birthday)/365),
   count(*)
FROM fbuser
GROUP BY CUBE (sex,
  floor((current_date - birthday)/365));
```

CUBE and ROLLUP

- Substitutable using `GROUP BY` and `UNION`
- `ROLLUP` simulation:

```
SELECT sex,
   floor((current_date - birthday)/365),
   count(*)
FROM fbuser
GROUP BY sex,
   floor((current_date - birthday)/365)
UNION
SELECT sex, null, count(*) FROM fbuser
GROUP BY sex
UNION
SELECT null, null, count(*) FROM fbuser;
```

Questions?