Databases
Lectures 9 and 10

Timothy G. Griffin

Computer Laboratory
University of Cambridge, UK

Databases, Lent 2009
Lecture 09 and 10

Two Themes ...

- Redundancy can be a GOOD thing!
- Duplicates, aggregates, and group by in SQL, and evolution to “Data Cube”

.... come together in OLAP

- OLTP : Online Transaction Processing (traditional databases)
  - Data is normalized for the sake of updates.

- OLAP : Online Analytic Processing
  - These are (almost) read-only databases.
  - Data is de-normalized for the sake of queries!
  - Multi-dimensional data cube emerging as common data model.
  - This can be seen as a generalization of SQL’s group by
Materialized Views

Suppose $Q$ is a very expensive, and very frequent query.
Why not de-normalize some data to speed up the evaluation of $Q$?

- This might be a reasonable thing to do, or ...
- ... it might be the first step to destroying the integrity of your data design.

Why not store the value of $Q$ in a table?
- This is called a materialized view.
- But now there is a problem: How often should this view be refreshed?
FIDO = Fetch Intensive Data Organization

fast updates

Extract

Normalized Database

fast reads

Read–optimized (NOT Normalized)
Example: Embedded databases

- Fast updates
- Table-driven applications
- Read-optimized
- Embedded Database
- Device

Normalized Database

Extract

T. Griffin (cl.cam.ac.uk)
Databases Lectures 9 and 10
DB 2009
Example: Hinxton Bioinformatics

Database system design from the European Bioinformatics Institute (Hinxton UK)

Other archives

Normalized Tables

Development DB

Production DB

Service DB

De-normalized Derived Tables

--- for fast access

Service Tools

End Users

Submitters

Q/C etc

Data exchange

Add value (computation)

Add value (review etc.)

Releases & Updates

Distrib
Example: Data Warehouse (Decision support)

Operational Database

fast updates

Extract

business analysis queries

Data Warehouse
## OLAP vs. OLTP

**OLTP**  
Online Transaction Processing

**OLAP**  
Online Analytical Processing

- Commonly associated with terms like Decision Support, Data Warehousing, etc.

<table>
<thead>
<tr>
<th></th>
<th>OLAP</th>
<th>OLTP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supports</td>
<td>analysis</td>
<td>day-to-day operations</td>
</tr>
<tr>
<td>Data is</td>
<td>historical</td>
<td>current</td>
</tr>
<tr>
<td>Transactions mostly</td>
<td>reads</td>
<td>updates</td>
</tr>
<tr>
<td>optimized for</td>
<td>query processing</td>
<td>updates</td>
</tr>
<tr>
<td>Normal Forms</td>
<td>not important</td>
<td>important</td>
</tr>
</tbody>
</table>
The big question

Is the relational model and its associated query language (SQL) well suited for OLAP databases?

- Aggregation (sums, averages, totals, ...) are very common in OLAP queries
  - Problem: SQL aggregation quickly runs out of steam.
  - Solution: Data Cube and associated operations (spreadsheets on steroids)

- Relational design is obsessed with normalization
  - Problem: Need to organize data well since all analysis queries cannot be anticipated in advance.
  - Solution: Multi-dimensional fact tables, with hierarchy in dimensions, star-schema design.

Let's start by looking at aggregate queries in SQL ...
An Example ...

```sql
mysql> select * from marks;

<table>
<thead>
<tr>
<th>sid</th>
<th>course</th>
<th>mark</th>
</tr>
</thead>
<tbody>
<tr>
<td>ev77</td>
<td>databases</td>
<td>92</td>
</tr>
<tr>
<td>ev77</td>
<td>spelling</td>
<td>99</td>
</tr>
<tr>
<td>tgg22</td>
<td>spelling</td>
<td>3</td>
</tr>
<tr>
<td>tgg22</td>
<td>databases</td>
<td>100</td>
</tr>
<tr>
<td>fm21</td>
<td>databases</td>
<td>92</td>
</tr>
<tr>
<td>fm21</td>
<td>spelling</td>
<td>100</td>
</tr>
<tr>
<td>jj25</td>
<td>databases</td>
<td>88</td>
</tr>
<tr>
<td>jj25</td>
<td>spelling</td>
<td>92</td>
</tr>
</tbody>
</table>
```

T. Griffin (cl.cam.ac.uk)
Databases Lectures 9 and 10
DB 2009
... of duplicates

```sql
mysql> select mark from marks;
+------+
| mark |
+------+
| 92   |
| 99   |
| 3    |
| 100  |
| 92   |
| 100  |
| 88   |
| 92   |
+------+
```
Why Multisets?

Duplicates are important for **aggregate functions**.

```sql
mysql> select min(mark),
       max(mark),
       sum(mark),
       avg(mark)
from marks;
```

<table>
<thead>
<tr>
<th>min(mark)</th>
<th>max(mark)</th>
<th>sum(mark)</th>
<th>avg(mark)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>100</td>
<td>666</td>
<td>83.2500</td>
</tr>
</tbody>
</table>

T. Griffin (cl.cam.ac.uk)
Databases Lectures 9 and 10
The \textbf{group by clause}

```sql
mysql> select course,
              min(mark),
              max(mark),
              avg(mark)
from marks
group by course;
+-----------+-----------+-----------+-----------+
| course    | min(mark) | max(mark) | avg(mark) |
| databases | 88        | 100       | 93.0000   |
| spelling  | 3         | 100       | 73.5000   |
+-----------+-----------+-----------+-----------+
```
Visualizing group by

<table>
<thead>
<tr>
<th>sid</th>
<th>course</th>
<th>mark</th>
</tr>
</thead>
<tbody>
<tr>
<td>ev77</td>
<td>databases</td>
<td>92</td>
</tr>
<tr>
<td>ev77</td>
<td>spelling</td>
<td>99</td>
</tr>
<tr>
<td>tgg22</td>
<td>spelling</td>
<td>3</td>
</tr>
<tr>
<td>tgg22</td>
<td>databases</td>
<td>100</td>
</tr>
<tr>
<td>fm21</td>
<td>databases</td>
<td>92</td>
</tr>
<tr>
<td>fm21</td>
<td>spelling</td>
<td>100</td>
</tr>
<tr>
<td>jj25</td>
<td>databases</td>
<td>88</td>
</tr>
<tr>
<td>jj25</td>
<td>spelling</td>
<td>92</td>
</tr>
</tbody>
</table>

group by course

<table>
<thead>
<tr>
<th>course</th>
<th>mark</th>
</tr>
</thead>
<tbody>
<tr>
<td>spelling</td>
<td>99</td>
</tr>
<tr>
<td>spelling</td>
<td>3</td>
</tr>
<tr>
<td>spelling</td>
<td>100</td>
</tr>
<tr>
<td>spelling</td>
<td>92</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>course</th>
<th>mark</th>
</tr>
</thead>
<tbody>
<tr>
<td>databases</td>
<td>92</td>
</tr>
<tr>
<td>databases</td>
<td>100</td>
</tr>
<tr>
<td>databases</td>
<td>92</td>
</tr>
<tr>
<td>databases</td>
<td>88</td>
</tr>
</tbody>
</table>
Visualizing group by

<table>
<thead>
<tr>
<th>course</th>
<th>mark</th>
</tr>
</thead>
<tbody>
<tr>
<td>spelling</td>
<td>99</td>
</tr>
<tr>
<td>spelling</td>
<td>3</td>
</tr>
<tr>
<td>spelling</td>
<td>100</td>
</tr>
<tr>
<td>spelling</td>
<td>92</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>course</th>
<th>min(mark)</th>
</tr>
</thead>
<tbody>
<tr>
<td>databases</td>
<td>92</td>
</tr>
<tr>
<td>databases</td>
<td>100</td>
</tr>
<tr>
<td>databases</td>
<td>92</td>
</tr>
<tr>
<td>databases</td>
<td>88</td>
</tr>
</tbody>
</table>

\[
\text{min}(\text{mark}) \Rightarrow
\]

<table>
<thead>
<tr>
<th>course</th>
<th>min(\text{mark})</th>
</tr>
</thead>
<tbody>
<tr>
<td>spelling</td>
<td>3</td>
</tr>
<tr>
<td>databases</td>
<td>88</td>
</tr>
</tbody>
</table>
The **having** clause

How can we select on the aggregated columns?

```mysql
mysql> select course,
min(mark),
max(mark),
avg(mark)
from marks
group by course
having min(mark) > 60;
```

<table>
<thead>
<tr>
<th>course</th>
<th>min(mark)</th>
<th>max(mark)</th>
<th>avg(mark)</th>
</tr>
</thead>
<tbody>
<tr>
<td>databases</td>
<td>88</td>
<td>100</td>
<td>93.0000</td>
</tr>
</tbody>
</table>
mysql> select course,
       min(mark) as minimum,
       max(mark) as maximum,
       avg(mark) as average
       from marks
       group by course
       having minimum > 60;

+-----------+---------+---------+---------+
| course    | minimum | maximum | average |
+-----------+---------+---------+---------+
| databases | 88      | 100     | 93.0000 |
+-----------+---------+---------+---------+
Limits of SQL aggregation

- Flat tables are great for processing, but hard for people to read and understand.
- Pivot tables and cross tabulations (spreadsheet terminology) are very useful for presenting data in ways that people can understand.
- SQL does not handle pivot tables and cross tabulations well.
Data Cube: A Relational Aggregation Operator
Generalizing Group-By, Cross-Tab, and Sub-Totals*

JIM GRAY
SURAJIT CHAUDHURI
ADAM BOSWORTH
ANDREW LAYMAN
DON REICHAхT
MURALI VENKATRAO
Microsoft Research, Advanced Technology Division, Microsoft Corporation, One Microsoft Way, Redmond, WA 98052

FRANK PELLLOW
HAMID PIRAHESH
IBM Research, 500 Harry Road, San Jose, CA 95120

Gray@Microsoft.com
SurajitC@Microsoft.com
AdamB@Microsoft.com
AndrewL@Microsoft.com
DonRei@Microsoft.com
MuraliV@Microsoft.com
Pellow@vnet.IBM.com
Pirahesh@Almaden.IBM.com
From aggregates to data cubes

[Diagram showing the transition from aggregates to data cubes, including concepts like Group By (with total), Cross Tab, and The Data Cube and The Sub-Space Aggregates.]
The Data Cube

- Data modeled as an $n$-dimensional (hyper-) cube
- Each dimension is associated with a hierarchy
- Each “point” records facts
- Aggregation and cross-tabulation possible along all dimensions
Hierarchy for **Location** Dimension

```
   all
  /   \\       \       \\
 region /  Europe   North_America
        /  /  ...   ... \\\  \\
 country /  Germany  Spain  Canada  ...  Mexico
        /  /  ...   ... \\
       /  /  Frankfurt  ...  Vancouver  ...  Toronto
      /  /  /  ...   ...  L. Chan  ...  M. Wind
      /  /  /  \\
    city  office
```
Cube Operations

Example: computing sums

```
<table>
<thead>
<tr>
<th>day 1</th>
<th>c1</th>
<th>c2</th>
<th>c3</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>12</td>
<td>11</td>
<td>8</td>
</tr>
<tr>
<td>p2</td>
<td>56</td>
<td>11</td>
<td>8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>day 2</th>
<th>c1</th>
<th>c2</th>
<th>c3</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>44</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>p2</td>
<td>1</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>day 1</th>
<th>c1</th>
<th>c2</th>
<th>c3</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>56</td>
<td>4</td>
<td>50</td>
</tr>
<tr>
<td>p2</td>
<td>11</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sum</th>
<th>c1</th>
<th>c2</th>
<th>c3</th>
</tr>
</thead>
<tbody>
<tr>
<td>sum</td>
<td>67</td>
<td>12</td>
<td>50</td>
</tr>
</tbody>
</table>

```

rollup

drill-down

129
The Star Schema as a design tool

- **time**
  - time_key
  - day
  - day_of_the_week
  - month
  - quarter
  - year

- **branch**
  - branch_key
  - branch_name
  - branch_type

- **item**
  - item_key
  - item_name
  - brand
  - type
  - supplier_type

- **location**
  - location_key
  - street
  - city
  - province_or_street
  - country

- **Sales Fact Table**
  - time_key
  - item_key
  - branch_key
  - location_key
  - units_sold
  - dollars_sold
  - avg_sales