Análise de Dados sobre uma Base de Dados SQL

As aplicações de análise de dados procuram padrões incomuns nos dados. Existem quatro passos distintos nesta análise:

• Formulação de uma consulta que extraia a informação relevante de uma grande base de dados.

• Extracção dos dados agregados da base de dados para um ficheiro ou tabela.

• Visualização dos dados de uma forma gráfica.

• Análise dos resultados e formulação de uma nova query.

Ferramentas de visualização:

• Mostram tendências nos dados, grupos e diferenças.

• Representam normalmente o conjunto de dados como um espaço N-dimensional.

Ferramentas de Visualização

• Uma aplicação de folha de cálculo como o Excel é um exemplo de uma ferramenta de visualização para análise de dados.

• Estas ferramentas tentam normalmente identifi car um sub-espaç o do espaç o N-dimensional que seja "interessante" em termos de análise ("dimensionality reduction").

• Esta "dimensionality reduction" é muitas vezes feita pela sumarização dos dados nas dimensões que não são consideradas. Por exemplo na análise de vendas de carros, podemos centrar-nos no papel do modelo, ano e cor dos carros vendidos e ignorar uma dimensão concessionário analisando apenas o total de vendas por modelo, ano e cor.

• Para além da sumarização e redução de dimensionalidade, as aplicações de análise de dados usam ferramentas como histogramas, "cross-tabulation", "roll-up" e "drill-down".

Extracção de Dados SQL e Relacional

• De que forma BD's relacionais se adequam a uma análise de dados multi-dimensional?

• De que forma os ficheiros bi-dimensionais (tabelas) modelam um problema N-dimensional?

• Como é que os sistemas relacionais suportam as operações sobre as representações N-dimensionais que são centrais para os programas de visualização e análise de dados?

Os sistemas relacionais modelam dados N-dimensionais como uma relação com N-atributos. Por exemplo, dados 4-dimensionais de medidas de temperatura da Terra são tipicamente representados pela seguinte tabela **Tempo**:

Tabela Tempo					
Data/Hora	Latitude	Longitude	Altitude (m)	Temp (c)	Pres (mb)
21/03/98:1500	37:58:33N	122:45:28W	102	21	1009
...muitos outros tuplos...					
21/03/04:1500	32:51:93N	112:41:20W	1234	12	801

Sumarização e redução de dimensões em SQL

Em SQL standard a agregação necessária para a implementação de sumarização e redução de dimensões é feita através das funções de agregação e do operador Group By.

• Cinco funções de agregação: `COUNT()`, `SUM()`, `MAX()`, `MIN()`, `AVG()`.

Exemplo: Média de todas as temperaturas medidas

```
SELECT AVG(Temp)
FROM Tempo;
```

• Agregação sobre valores distintos:

Exemplo: Contagem das diferentes horas de medições

```
SELECT COUNT(DISTINCT Data/Hora)
FROM Tempo;
```

Group By

As funções de agregação retornam um único valor. Usando o operador `GROUP BY` o SQL pode criar uma tabela com vários tuplos indexados por um conjunto de atributos.

Exemplo: Temperatura média para cada Data/Hora e altitude de medição

```
SELECT Data/Hora, Altitude, AVG(Temp)
FROM Tempo
GROUP BY Data/Hora, Altitude;
```

O `GROUP BY` particiona a relação em conjuntos disjuntos de tuplos e agrega os valores dentro de cada conjunto.

Muitos sistemas SQL adicionam outras funções de agregação para além das cinco funções standard: (mediana, desvio padrão, centro de massa, volatilidade, etc.).

Funções de agregação genéricas

Alguns sistemas SQL permitem a definição de funções de agregação genéricas através da definição de 3 funções que implementam o operador de agregação:

- `init(&handle)` - Reserva espaço para a estrutura `handle` e inicializa a computação agregada.
- `iter(&handle, value)` - Agrega o próximo valor no agregado corrente.
- `value = final(&handle)` - Calcula e retorna o resultado do agregado através dos dados guardados na estrutura `handle`. Liberta a memória reservada para a estrutura `handle`.

Exercício: Implementar usando a API C do MySQL um protótipo genérico de função de agregação baseado em chamadas às funções `init()`, `iter()` e `final()`.

Exercício: Implementar a função de agregação `AVG()` usando o protótipo anterior.

Esta definição de protótipos de funções de agregação genéricas foi adicionada ao Draft Proposed Standard para o SQL.

Estruturas e funções em C para implementação de operadores de agregação

- Estrutura genérica de operador de agregação:

```
typedef struct aggr_prot {
  char *name;
  void (*init)(void **handle);
  void (*iter)(void *handle, char *value);
  void (*final)(void *handle, void *result);
  void *result;
} aggr_prot_type;
```

- Estrutura do `handle` específico do operador de agregação (`AVG`):

```
struct avg {
  int count;
  double sum;
};
```

Estruturas e funções em C para implementação de operadores de agregação

- Protótipos das funções `aggr_function()`, `init()`, `iter()` e `final()` (AVG):

```
void aggr_function(MYSQL_RES *res_set, aggr_prot_type *a);
void avg_init(void **handle);
void avg_iter(void *handle, char *value);
void avg_final(void *handle, void *result);
```

- Função `aggr_function()`:

```
void aggr_function(MYSQL_RES *res_set, aggr_prot_type *a)
{ MYSQL_ROW row;
  void *handle;
  (*a->init)(&handle);
  while ((row = mysql_fetch_row(res_set))!=NULL)
     (*a->iter)(handle,row[0]);
  (*a->final)(handle,a->result);
}
```

Estruturas e funções em C para implementação de operadores de agregação

- Funções `init()` e `iter()` (AVG):

```
void avg_init(void **handle)
{ struct avg *local_handle = (struct avg *)
     malloc(sizeof(struct avg));
  local_handle->count=0;
  local_handle->sum=0;
  *handle = (void *) local_handle;
}

void avg_iter(void *handle, char *value)
{ struct avg *local_handle = (struct avg *) handle;
  local_handle->count++;
  local_handle->sum+=atof(value);
}
```

Estruturas e funções em C para implementação de operadores de agregação

- Função `final()` (AVG):

```
void avg_final(void *handle,void *result)
{ double *local_result;
  struct avg *local_handle = (struct avg *) handle;
  local_result = (double *) result;
  *local_result = local_handle->sum/local_handle->count;
  free(handle);
}
```

Estruturas e funções em C para implementação de operadores de agregação

- Função `main()` (AVG):

```
main()
{ aggr_prot_type a;
  a.name = ‘‘AVG’’;
  a.init = &avg_init;
  a.iter = &avg_iter;
  a.final = &avg_final;
  a.result= malloc(sizeof(double));
  MYSQL mysql;
  MYSQL_RES *res_set;
  mysql_init(&mysql);
  mysql_real_connect(&mysql,"host","user","pass","db",0,NULL,0);
  mysql_query(&mysql,"SELECT origem FROM ramos");
  res_set = mysql_store_result(&mysql);
  aggr_function(res_set,&a);
  printf("%s: %f\n",a.name,*((double *) a.result));
  free(a.result);
  mysql_free_result(res_set);
}
```

Outros operadores de agregação

• `RANK(expressão)`: retorna o lugar da `expressão` no conjunto de todos os valores no domínio em causa da tabela. Se existem `N` valores na coluna e o valor é o mais alto, o seu `RANK` é `N`, se é o mais baixo o `RANK` é 1.

• `N_tile(expressão, n)`: o intervalo da `expressão` é calculado e dividido em `n` intervalos de valores com população aproximadamente igual. O número do intervalo para a `expressão` é retornado.

Exemplo:

```
SELECT Percentil, MIN(Temp), MAX(Temp)
FROM Tempo
GROUP BY N_tile(Temp,10) as Percentil
HAVING Percentil = 5;
```

Problemas com Group By

- Histogramas
- Totais para “Roll-ups” e sub-totais para “Drill-downs”
- “Cross-tabulations”

O problema na construção de histogramas resulta da incapacidade do SQL standard fazer agregações sobre categorias calculadas.

Exemplo:

```
SELECT dia, país, MAX(Temp)
FROM Tempo
GROUP BY Day(Data/Hora) as dia,
         Nation(Latitude, Longitude) as país;
```

Histogramas

- Solução para histogramas em SQL92:

```
SELECT dia, país, MAX(Temp)
FROM (SELECT Day(Data/Hora) as dia,
         Nation(Latitude, Longitude) as país, Temp
      FROM Tempo
     ) AS aux
GROUP BY dia, país;
```

Relatórios com totais para “Roll-ups” e sub-totais para “Drill-downs”

Exemplo:

Roll-up de Vendas por Modelo por Ano e por Cor					
Modelo	Ano	Cor	Vendas por Modelo por Ano por Cor	Vendas por Modelo por Ano	Vendas por Modelo
Laguna	2002	preto	50		
		branco	40		
				90	
	2003	preto	85		
		branco	115		
				200	
					290

• Esta representação sugere a criação de 2^N colunas de agregação para um roll-up de N elementos.

• Esta representação não é relacional por causa dos valores vazios (NULL) nas colunas que não são permitidos na chave da relação. Alternativa relacional:

Roll-up de Vendas por Modelo por Ano e por Cor (Relacional)					
Modelo	Ano	Cor	Vendas	Vendas por Modelo por Ano	Vendas por Modelo
Laguna	2002	preto	50	90	290
Laguna	2002	branco	40	90	290
Laguna	2003	preto	85	200	290
Laguna	2003	branco	115	200	290

Tabelas Pivot

• A representação anterior implica que o número de colunas cresç a exponentialmente no número de atributos agregados. Para um roll-up a 6 dimensões seria necessário adicionar 64 colunas à relação, criando difi culdades de nomeação dos atributos.

• Uma alternativa é a criação de tabelas Pivot:

Tabela Pivot para representação de Vendas de carros							
Somat´orio de Vendas	Ano	Cor					
	2002		Total 2002	2003		Total 2003	Total
Modelo	preto	branco		preto	branco		
Laguna	50	40	90	85	115	200	290
Megane	50	10	60	85	75	160	220
Total	100	50	150	170	190	360	510

• A ideia das tabelas pivot é basear a criação de colunas nos *valores* de um sub-conjunto de colunas, ao invés de num sub-conjunto de nomes de colunas.

• Pode levar à criação de muitas colunas se os atributos pivotados tiverem muitos valores.

O valor especial 'ALL'

Podemos evitar a criação exponencial de colunas para valores agregados através da *sobrecarga* de valores de colunas com um valor especial 'ALL'.

Exemplo:

Sumário de Vendas			
Modelo	Ano	Cor	Unidades
Laguna	2002	preto	50
Laguna	2002	branco	40
Laguna	2002	'ALL'	90
Laguna	2003	preto	85
Laguna	2003	branco	115
Laguna	2003	'ALL'	200
Laguna	'ALL'	'ALL'	290

- Esta representação é relacional.
- O valor especial 'ALL' representa todos os valores do atributo em causa.

Implementação em SQL

- Sendo a representação anterior relacional, pode ser implementada por consultas SQL sobre uma tabela Vendas:

```
SELECT 'ALL', 'ALL', 'ALL', SUM(Vendas)
FROM Vendas
WHERE Modelo = 'Laguna'
UNION
SELECT Modelo, 'ALL', 'ALL', SUM(Vendas)
FROM Vendas
WHERE Modelo = 'Laguna'
GROUP BY Modelo
UNION
SELECT Modelo, Ano, 'ALL', SUM(Vendas)
FROM Vendas
WHERE Modelo = 'Laguna'
GROUP BY Modelo, Ano
UNION
SELECT Modelo, Ano, Cor, SUM(Vendas)
FROM Vendas
WHERE Modelo = 'Laguna'
GROUP BY Modelo, Ano, Cor;
```

“Cross-tabulation”

• A tabela **Sumário de Vendas** constitui um simples “roll-up” 3-dimensional. A agregação de N dimensões requer N `UNION`’s idênticas às apresentadas.

• A operação de “roll-up” é assimétrica. De notar que a tabela anterior agrega vendas por ano mas não por cor. Os tuplos que faltam são os seguintes:

Sumário de Vendas			
Modelo	Ano	Cor	Unidades
Laguna	’ALL’	preto	135
Laguna	’ALL’	branco	155

e a consulta SQL que retorna estes tuplos é a seguinte:

```
SELECT Modelo, ’ALL’, Cor, SUM(Vendas)
FROM Vendas
WHERE Modelo = ’Laguna’
GROUP BY Modelo, Cor;
```

• Esta agregação simétrica é chamada “*cross-tabulation*”.

Representação de “cross-tabulations”

A representação de uma “cross-tabulation” não é usualmente feita através que uma tabela como a tabela **Sumário de Vendas**, mas sim da forma seguinte:

“Cross-tabulation” das Vendas de Lagunas			
Laguna	2002	2003	Total('ALL')
preto	50	85	135
branco	40	115	155
Total('ALL')	90	200	290

Esta “cross-tabulation” representa uma agregação bi-dimensional. Se adicionarmos outros modelos de carros torna-se uma agregação tri-dimensional. A representação é agora um cubo em que cada plano apresenta uma “cross-tabulation” bi-dimensional para cada modelo de carro:

“Cross-tabulation” das Vendas de M éganes			
M´egane	2002	2003	Total('ALL')
preto	50	85	135
branco	10	75	95
Total('ALL')	60	160	220

Estas “cross-tabulations” generalizam-se para N-dimensões, sendo representadas por hiper-cubos.

O operador de agregação CUBE

• O operador de agregação CUBE constrói uma tabela que contém as várias combinações do valor 'ALL' nos tuplos, calculando o valor agregado para a função de agregação em causa.

• A criação do CUBE requer a geração do conjunto de todos os subconjuntos dos atributos de agregação.

• Exemplo do operador CUBE:

```
SELECT dia, país, MAX(Temp)
FROM Tempo
GROUP BY CUBE
    Day(Data/Hora) as dia,
    Country(Latitude,Longitude) as país;
```

Semântica do operador CUBE

A semântica do operador CUBE é a seguinte:

• Calcula primeiro a agregação sobre os atributos na lista de atributos do SELECT, tal como no operador de GROUP BY;

• Em seguida executa a UNION com a super-agregação do cubo global, substituindo com o valor 'ALL' nas colunas agregadas.

• Se existirem N atributos na lista de SELECT existirão 2^{N-1} valores super-agregados. Se a cardinalidade dos N atributos for $C_1, C_2, ..., C_N$ então a cardinalidade da relação representando o cubo será $\Pi(C_i + 1)$.

O operador ROLLUP

• Por vezes o cálculo completo do cubo não é necessário, visto que alguns valores agregados podem ser pouco signifi cativos.

• Podemos acrescentar um operador de ROLLUP que calcula apenas os seguintes valores:

```
(v1, v2,..., vn,  f()),
(v1, v2,..., ALL, f()),
       ...
(v1 ,ALL,...,ALL, f()),
(ALL,ALL,...,ALL, f()).
```

A sintaxe proposta para utilização dos operadores CUBE e ROLLUP é pela generalização do operador de GROUP BY:

```
GROUP BY <select list>
  ROLLUP <select list>
    CUBE <select list>
```