DEFINIÇÃO DE COLUNAS EM SQL

Sérgio Mergen

Definição de colunas em SQL

- Ao definir uma coluna de tabela, muitos aspectos devem ser considerados
  - Uso do tipos de dados adequados
  - Uso de colunas opcionais
  - Uso de restrições de integridade
  - ...
- Na aula de hoje analisaremos os dois primeiros aspectos

USO DE TIPOS DE DADOS ADEQUADOS

Tipos de dados em SQL

- “Alguns” tipos disponíveis
  - INT, INTEGER, SHORT, LONG, NUMBER, NUMERIC, SMALLINT, INTEGER2, INTEGER4
  - CHAR, VARCHAR, ALPHANUMERIC, CHARACTER, STRING
  - DATE, TIME, DATETIME, TIMESTAMP
  - DECIMAL, REAL, DOUBLE, FLOAT, FLOAT4, FLOAT8
  - BINARY, BIT, BYTE, YESNO, BOOLEAN
  - CURRENCY, MONEY
  - TEXT, OLEOBJECT, LONGTEXT, MEMO, NOTE, GENERAL

Tipos de dados em SQL

- Com tantos tipos, como escolher o tipo mais adequado?
- O que significa ser adequado?

- A análise pode levar em consideração diversos fatores
  - Codificação ampla x codificação restrita
  - Tamanho variável x tamanho fixo
  - Capacidade maior x capacidade menor

Codificação ampla x codificação restrita

- Um alfabeto indica o universo de símbolos permitidos por um tipo de dados
  - Alfabeto restrito
    - Possibilita a representação de uma quantidade menor de símbolos
  - Alfabeto amplo
    - Possibilita a representação de uma quantidade maior de símbolos

- Um alfabeto mais amplo
  - gasta mais bits para representar cada símbolo.
- Ex.
  - o tipo INT (caracteres numéricos) é mais restrito que o tipo CHAR (caracteres alfanuméricos)

Codificação ampla x codificação restrita

- Ex. Uma tabela de monitoramento guarda dados de altitude de aviões durante o voo.

- Que tipo de dados você usaria?
  - Numérico
    - ex. INT
  - Textual
    - Ex. CHAR

- Usualmente se escolhe um tipo que melhor represente a informação que se quer armazenar
  - Mas outras análises também são possíveis

Codificação ampla x codificação restrita

- Ex. Uma tabela de monitoramento guarda dados de altitude de aviões durante o voo.

- Vantagens de usar o tipo numérico
  - Ocupa menos espaço que o tipo textual
  - Filtros por altitude são mais eficientes
  - Restringe o domínio de valores permitidos
    - Apenas dígitos são aceitos

- Vantagens de usar o tipo textual
  - Não restringe o domínio de valores permitidos
    - Ex. Permite valores como “10.000 pés”

Codificação ampla x codificação restrita

- Textos também podem ser representados em alfabetos mais amplos ou mais restritos
- SGBDS disponibilizam diversos alfabetos (ou encodings)
- Dois das mais conhecidos:
  - ISO-8859-1 (Latin1)
  - UTF8

Codificação ampla x codificação restrita

- ISO-8859-1
  - Cada caractere ocupa 1 byte
  - Apenas 256 caracteres estão disponíveis
  - Possui diversas variações (Latin1, Latin2, ...)
    - Latin1 é a que mapeia o alfabeto ocidental
- É uma extensão da tabela ASCII
  - Os 7 bits da tabela ASCII original mapeiam 128 símbolos
  - O oitavo bit adiciona outros 128 símbolos
  - Para os caracteres ocidentais, o código ASCII e o código Latin1 são equivalentes
- Exemplos de símbolos suportados
  - ‘a’, ‘A’, ‘ç’, ‘é’, ‘â’

Codificação ampla x codificação restrita

- UTF8
  - Cada símbolo pode ocupar de 1 a 4 bytes
  - Cada caractere latino não diacrítico ocupa 1 byte
  - Cada caractere latino diacrítico ocupa 2 bytes
  - Para alguns alfabetos são necessários de 3 a 4 bytes

- Exemplos de símbolos que usam 1 byte
  - A, a, -, +, 0, 4
- Exemplos de símbolos que usam 2 bytes
  - ã, ú, î, é, ç
- Exemplos de símbolos que usam 3 bytes
  - Aqueles dos alfabetos orientais

Codificação ampla x codificação restrita

- Suponha que a coluna char(6) tenha como conteúdo o texto ‘versão’. Quantos bytes ocupa esse valor?
  - Depende do encoding
- Em utf8
  - 7 bytes
- Em latin1
  - 6 bytes

Codificação ampla x codificação restrita

- Se bastam os 256 caracteres do latin1
  - Faz sentido usá-lo

- Motivos
  - Economia de espaço
  - Eficiência em comparações
    - É mais rápido comparar strings onde cada símbolo ocupa um byte do que strings em que a quantidade de bytes de cada símbolo é variável

Codificação ampla x codificação restrita

- O encoding é definido para todas colunas da tabela

- Ex. na criação da tabela (mySQL)

```
CREATE TABLE  tabela (...) CHARSET = latin1;
```

- Ex. na alteração da tabela (mySQL)

```
ALTER TABLE tabela
        CONVERT TO CHARACTER SET latin1;
```

Uso de Collation

- Além do encoding, é possível definir a colação (**collation**)
  - A colação de uma coluna textual diz respeito à forma como os símbolos que compõe o valor da coluna são comparados em clausulas WHERE e ORDER BY

- Ex. na criação da tabela (mySQL)

CREATE TABLE **tabela** (...) CHARSET = **latin1**
COLATTE **latin1_swedish_ci**;

CREATE TABLE **tabela** (...) CHARSET = **latin1**
COLATTE **latin1_german2_ci**;

Uso de Collation

- SELECT nome FROM tabela ORDER BY nome;

usando
latin1_swedish_ci;

nome
bar
bär
ber
dar

usando
latin1_german2_ci;

nome
bär
bar
ber
dar

Uso de Collation

- SELECT nome FROM tabela where nome like ‘bä%’;

usando
latin1_swedish_ci;

nome
bär

usando
latin1_german2_ci;

nome
bär
bar

Tipos de dados em SQL

- Com tantos tipos, como escolher o tipo mais adequado?
- O que significa ser adequado?

- A análise pode levar em consideração diversos fatores
  - Tipo abrangente x tipo restrito
  - Tamanho variável x tamanho fixo
  - Capacidade maior x capacidade menor

Tamanho variável x tamanho fixo

- Tamanho fixo (CHAR)
  - Todos os valores da coluna ocuparão o mesmo espaço

- Tamanho variável (VARCHAR)
  - O espaço destinado à coluna equivale ao comprimento da string

  - Precisa usar bytes de controle para determinar o tamanho da string
    - 1 byte se o tamanho máximo é de 255 caracteres
      - Ex. VARCHAR (10)
    - 2 bytes se o tamanho máximo é maior do que 255 caracteres
      - Ex. VARCHAR (300)
      - Ex. VARCHAR (1000)

Tamanho variável x tamanho fixo

- Ex. quantos bytes ocupa o seguinte texto? ‘projeto ACME’
  - Usando CHAR(50) = 50 bytes
  - Usando VARCHAR (50) = 13 bytes
    - 12 bytes de caracteres
    - 1 byte de controle

Tamanho variável x tamanho fixo

- Ex. quantos bytes ocupa o seguinte texto? ‘projeto ACME’
  - Usando CHAR(50) = 50 bytes
  - Usando VARCHAR (50) = 13 bytes
    - 12 bytes de caracteres
    - 1 byte de controle

- Dica: ao usar CHAR, escolha um tamanho
  - grande o suficiente para guardar todos os possíveis valores
  - Pequeno o suficiente para não desperdiçar bytes que nunca serão usados

Exercício

- É muito comum encontrar esquemas de bancos de dados cujos campos de texto de tamanho variável são definidos como
  - VARCHAR (255)

- Você saberia explicar o porquê?

Tamanho variável x tamanho fixo

- Ex. Como modelar o nome de uma pessoa?

- Tamanho fixo?
  - De que tamanho?

- Tamanho variável?
  - De que tamanho?

Tamanho variável x tamanho fixo

- Ex. Como modelar o nome de uma pessoa?

- Tamanho fixo?
  - De que tamanho?

- Tamanho variável?
  - De que tamanho?

- A decisão geralmente depende do contexto
  - Todas as colunas devem ser analisadas
  - Por quê? Por causa da organização física de um SGBD

Tamanho variável x tamanho fixo

- Organização de arquivos no SGBD
  - Registros de tamanho fixo
    - Indexação mais eficiente
    - As buscas são mais rápidas
  - Registros de tamanho variável
    - Melhor ocupação de espaço

- O registro terá tamanho variável quando qualquer coluna
  - tiver tamanho variável
  - ou permitir nulo
    - Mais adiante falaremos sobre colunas nulas

Tamanho variável x tamanho fixo

- Se otimização de espaço for mais importante
  - Usar tamanho variável
- Se desempenho no acesso é mais importante
  - Usar tamanho fixo
- Se o tamanho for sempre o mesmo (ou quase)
  - Usar tamanho fixo

- Obs. Se alguma outra coluna já tiver tamanho variável
  - Perde-se o desempenho no acesso
  - Dica: em alguns casos vale a pena mover colunas nulas ou de tamanho variável para outra tabela
    - Como se chama esse processo de refatoração?

Exercício

- O que você faria para melhorar o desempenho no acesso ao nome e ao salário de um funcionário?
  - Considerando a questão dos registros de tamanho variável
  - Sabendo que dados de email, twitter e facebook são pouco acessados

Func	
*idFunc	int
Nome	varchar (255)
Facebook	varchar (255)
Salario	decimal (8)
email	varchar (255)
twitter	varchar (255)

Tipos de dados em SQL

- Com tantos tipos, como escolher o tipo mais adequado?
- O que significa ser adequado?

- A análise pode levar em consideração diversos fatores
  - Tipo abrangente x tipo restrito
  - Tamanho variável x tamanho fixo
  - Capacidade maior x capacidade menor

Capacidade maior x capacidade menor

- Cada tipo de dados é representado de forma diferente
  - Através de uma sequência de bytes
- Cada tipo de dados tem
  - Um tamanho em bytes
  - Que leva a uma **capacidade** de armazenamento (o **intervalo** de valores suportado)

- Ex.
  - Um tipo INT
    - ocupa mais bytes do que um tipo SMALLINT
    - Tem capacidade de armazenamento maior do que SMALLINT

Capacidade maior x capacidade menor

- Que tipo de dado usar para o salário de um funcionário?
  - Int
  - Smallint

- E se analisássemos também outros formatos de representação além do inteiro?
  - Int
  - Smallint
  - Float
  - Decimal

Capacidade maior x capacidade menor

tipo	bytes	Capacidade			
		Signed (com sinal)		Unsigned (sem sinal)	
		mínimo	máximo	mínimo	máximo
tinyint	1	-128	127	0	255
Smallint	2	-32.768	32.767	0	65.535
Int	4	-2.147.483.648	2.147.483.647	0	4.294.967.295

Capacidade maior x capacidade menor

- Decimal (x,y)
  - x-y = número de dígitos antes da vírgula
  - y = número de dígitos depois da vírgula

- Exemplos
  - Decimal (8,2)
    - 6 dígitos antes da virgula
    - 2 dígitos depois da vírgula

  - Decimal (6,0)
    - 6 dígitos antes da virgula
    - nenhum dígito depois da vírgula

Capacidade maior x capacidade menor

- Para calcular o tamanho, deve-se somar a quantidade de bytes necessária para os dígitos
  - Antes da vírgula
  - Depois da vírgula

- A quantidade é calculada usando a seguinte tabela

# digitos	bytes
1-2	1
3-4	2
5-6	3
7-9	4

Capacidade maior x capacidade menor

tipo	Bytes antes	Bytes depois	Total de bytes	Capacidade	
				mínimo	máximo
Decimal (5,0)	3	0	3	-99.999	99.999
Decimal (6,0)	3	0	3	-999.999	999.999
Decimal (7,2)	3	1	4	-99.999,99	99.999,99
Decimal (8,2)	3	1	4	-999.999,99	999.999,99

Capacidade maior x capacidade menor

- Dicas na hora da escolha
- Usar o tipo de dado mais específico, quando se aplicar
  - DATE em vez de DATETIME
    - Quando a hora não importa
  - SMALLINT em vez de INT
    - Quando o valor máximo nem chega próximo ao limite superior do SMALLINT
  - DECIMAL (8,2) em vez de DECIMAL (7,2)
    - O tamanho é o mesmo
    - Mas a capacidade de armazenamento é maior

USO DE COLUNAS OPCIONAIS

Colunas opcionais

- Colunas opcionais são usadas quando o valor nem sempre será preenchido

- Ex.
  - **nome** CHAR (50) NOT NULL
    - A coluna nome é obrigatória (necessariamente terá um valor)
    - O não preenchimento do valor leva a um erro

  - **bairro** CHAR (50) NULL
    - A coluna bairro é opcional (pode permanecer sem valor algum)
    - O não preenchimento deixa o campo com o valor NULL
    - O conteúdo NULL significa que o valor é desconhecido

Colunas opcionais

- Se a tabela possuir colunas opcionais
  - Cada registro terá um vetor de bits que indicará quais colunas opcionais possuem valor
- O tamanho do vetor de bits é múltiplo de 1 byte
  - Caso haja de 1 a 8 colunas opcionais, é gasto 1 byte
  - Caso haja de 9 a 16 colunas opcionais, são gastos 2 bytes
  - E assim por diante

Colunas opcionais

- Ex.

  nome CHAR (50) NOT NULL = 50 bytes

  nome CHAR (50) NULL =

  50 bytes + 1 byte (se valor **não** nulo)

  1 byte (se valor nulo)

  altitude INT NOT NULL = 4 bytes

  altitude INT NULL =

  4 bytes + 1 byte (se valor **não** nulo)

  1 byte (se valor nulo)

- O exemplo acima considera que haja apenas uma coluna opcional na tabela

Colunas opcionais

- Como vimos, a escolha entre colunas opcionais e obrigatórias afeta o tamanho dos registros
  - O tamanho do overhead depende do tamanho do registro
    - Em registros grandes, esse bitmap de 1 ou 2 bytes é insignificante
- No entanto, esse não é o único critério a utilizar
  - O uso de colunas nulas faz com que o registro tenha tamanho variável
  - Isso pode trazer um impacto em consultas
    - Como veremos na próxima aula

Atividade Individual

- Foi disponibilizado um banco de dados contendo uma tabela chamada pessoa
- O objetivo do trabalho é fazer ajustes no esquema para reduzir ao máximo a quantidade de bytes usada por registro
- Os ajustes devem ser feitos em uma tabela com outro nome. Os dados da tabela original devem ser migrados para ela
- Analise os registros para decidir que tipo de ajuste pode ser feito

pessoa
Id
Nome
Idade
IP
Nascimento
Sexo
email
CPF

Atividade Individual

- Os dois comandos abaixo são formas alternativas que permitem ver o tamanho dos dados de cada tabela

```
SELECT table_name, table_rows, data_length
FROM  information_schema.TABLES
WHERE  table_schema = 'NOME_DO_SEU_BANCO';
```

```
SHOW TABLE STATUS FROM `NOME_DO_SEU_BANCO`
LIKE 'NOME_DA_SUA_TABELA';
```

- O objetivo é fazer com que a tabela ajustada ocupe menos de 135000 bytes

Comandos úteis

- Função que mantém apenas os caracteres a partir de um caractere específico

```
REPLACE(col, ‘x', '')
```

- Função que extrai de **col** todo o conteúdo anterior a uma ocorrência **pos** de um delimitador **del**. Se pos = -1, é extraído o conteúdo posterior a todas as ocorrências

```
SUBSTRING_INDEX(col, del, pos)
```

- Função que mantém apenas os caracteres a partir de um caractere específico

```
SUBSTRING(col, 2)
```

Atividade Individual 2

- Suponha que o projeto tem um status que pode ser
  - ‘A’ de ativo
  - ‘I’ de inativo
  - ‘S’ de suspenso.
- Alguns projetos não têm valor definido para status

- Como modelar status?
  - Obs. Se fossem apenas duas possibilidades de valor (ex. ‘ativo’ ou ‘inativo’), daria para usar um tipo booleano
    - BOOLEAN NULL
  - No entanto, essa opção já é inviabilizada porque são três possiblidades de valor

Atividade Individual 2

- Como modelar status?
  - Como opcional de tamanho fixo
    - CHAR (1) NULL
  - Como obrigatório de tamanho fixo e status = ‘?’ para indefinido
    - CHAR (1) NOT NULL
  - Como obrigatório de tamanho variável
    - VARCHAR (1) NOT NULL

- Justifique sua resposta apresentando como argumento uma análise sobre os possíveis cenários
  - muitos registros com status
  - poucos registros com status