Banco de Dados e SQL

MAC5865 - Tópicos em Ciência e Engenharia de Dados

Direcionem as perguntas aqui:

`https://app.sli.do/event/nc8cxa3q`

R. Hirata Jr. e Mateus Espadoto

Agradecemos Rodrigo Assirati Dias por parte do material.

2020/2

Instituto de Matemática e Estatística - IME USP

Conteúdo

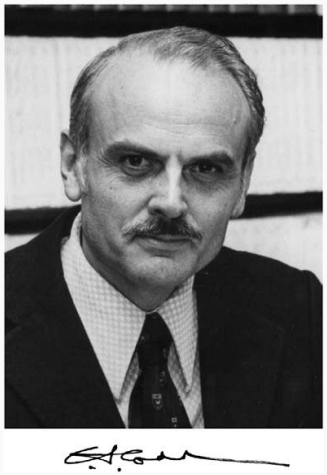
1. Modelo Relacional
2. SQL - Structured Query Language
3. Comandos de Manipulação de Dados (DML)

Modelo Relacional

Modelo Relacional

- Proposto por E. F. Codd em 1970, *A Relational Model of Data for Large Shared Data Banks*
  - Dados são apresentados ao usuário na forma de *relações* (tabelas, entidades), que possuem *atributos* (colunas, campos) e *registros* (linhas, tuplas)
  - Tabelas podem ter *relacionamentos* entre si, que possibilitam a criação de estruturas de dados sofisticadas
  - Modelo unificado para armazenamento e recuperação de dados
    - (lembre-se que em 1970 a norma era usar COBOL com arquivos indexados, em que cada programador podia definir sua estrutura de dados)
  - Formas Normais: regras de modelagem de dados
  - Linguagem para acesso a dados

Edgar Frank "Ted" Codd (1923 - 2003)



- Matemático e Cientista da Computação Inglês
- PhD pela U. de Michigan, estendeu o trabalho de von Neumann: *Self Reproducing Cellular Automata*
- Modelo Relacional (*breakthrough* na área de banco de dados)
- IBM Fellow em 1976, Prêmio Turing em 1981, ACM Fellow em 1994
- Trabalhou com Chris Date na IBM (*designer* do DB/2)
- `www.ibm.com/ibm/history/exhibits/builders/builders_codd.html`
- (e serviu na RAF durante a Segunda Guerra como piloto de patrulha)

SQL - Structured Query Language

SQL - Structured Query Language

- O modelo de Codd previa uma linguagem para acesso aos dados baseada em:
  - Operadores relacionais (união, intersecção, diferença, produto cartesiano)
  - Operador restrição (subconjunto das linhas)
  - Operador projeção (subconjunto das colunas)
  - Operador junção (junção de duas tabelas com base nas suas colunas em comum)
- SQL implementa essas operações e se tornou padrão nos sistemas de bancos de dados
- Primeiro padrão ANSI SQL criado em 1986, revisado em 1989, 1992, 1999, 2003, 2006, 2008, 2011 e 2016
- Fabricantes aderem aos padrões muito lentamente, e para complicar ainda criam suas próprias extensões (T-SQL, PL/SQL, PL/pgSQL, etc)

SQL - Structured Query Language

- Sistemas de banco de dados modernos tipicamente implementam três conjuntos de comandos:
  - DML: Data Manipulation Language (manipulação de dados, foco deste curso)
  - DDL: Data Definition Language (manipulação de objetos, como tabelas e colunas)
  - DCL: Data Control Language (controle de acesso a objetos)
- Cada produto possui suas variações, mas há grande similaridade, i.e., é simples transferir conhecimento de um produto para outro

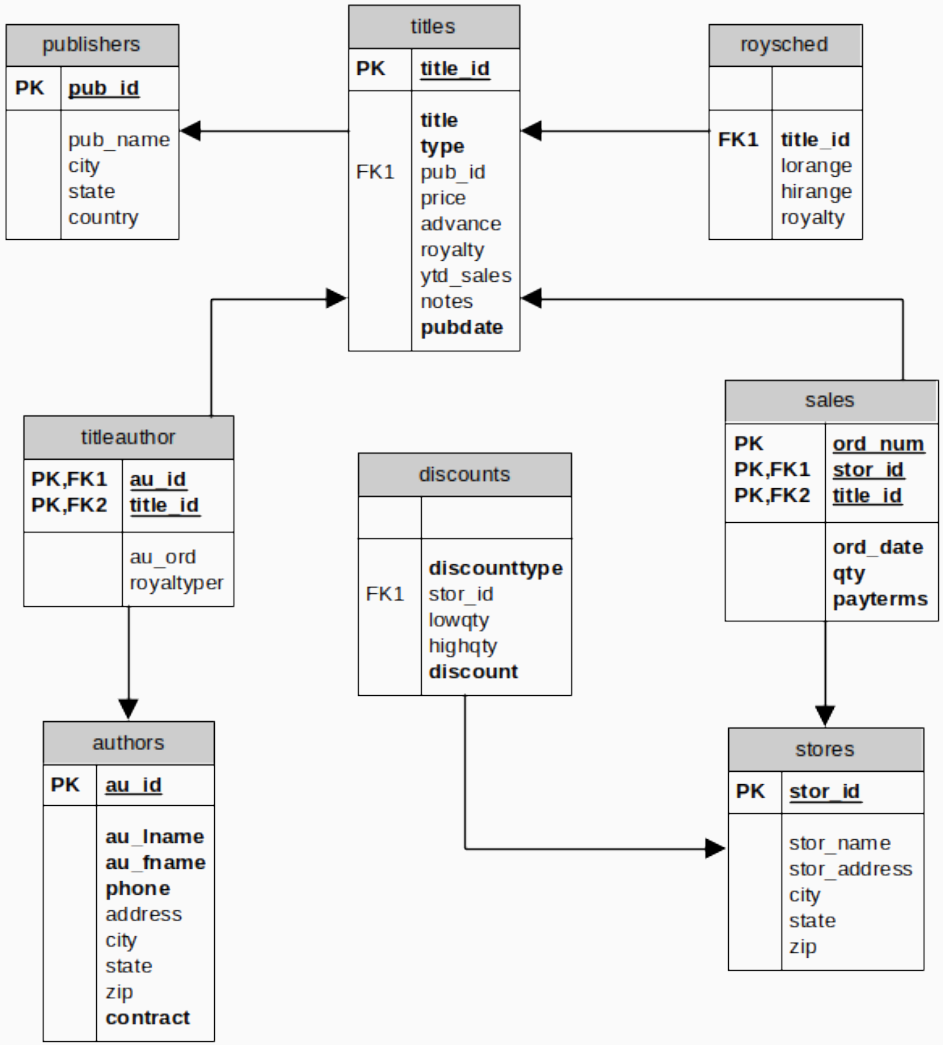
Comandos de Manipulação de Dados (DML)

Banco de Dados de Exemplo: pubs

- Banco de dados de exemplo que acompanhava o Microsoft SQL Server em versões passadas
- Simples, mas cobre diversos casos típicos de acesso a dados
- O banco contém dados editoras, títulos dos livros editados, autores, livrarias, vendas, descontos etc.

Banco de Dados de Exemplo: pubs

- Diagrama Entidade-Relacionamento do banco de dados pubs:



Comando SELECT

- De longe o comando mais importante, e também o mais complexo
- Forma básica:

```
SELECT <colunas>
FROM   <tabelas>
WHERE  <restricoes>
```

- Chamamos a cada uma das partes de um comando SQL (SELECT, FROM, WHERE nesse exemplo) de *cláusulas*.
- Algumas cláusulas são opcionais. O exemplo a seguir não possui a cláusula WHERE:

```
SELECT * FROM publishers
```

- Esse comando retorna todas as colunas e linhas da tabela *publishers*, sem restrições

Comando SELECT: Exemplos

- Mais exemplos:

```
-- Todas as colunas
SELECT pub_id, pub_name, city, state, country
FROM publishers

-- Apenas colunas desejadas (projecao)
SELECT pub_name, state
FROM publishers

-- Outra ordem de colunas
SELECT state, pub_name
FROM publishers

-- Usando um alias para uma coluna
SELECT state AS Estado, pub_name AS Editora
FROM publishers
```

Comando SELECT: Exemplos

- Mais exemplos (SQL é case-insensitive):

```
-- Dados apenas das editoras da
-- California (restricao)
select pub_name, city, state
from publishers
where state = 'CA'

-- Apenas editoras de Washington
```

Comando SELECT: Exemplos

- Mais exemplos (SQL é case-insensitive):

```
-- Dados apenas das editoras da
-- California (restricao)
select pub_name, city, state
from publishers
where state = 'CA'

-- Apenas editoras de Washington
select pub_name, city, state
from publishers
where state = 'WA'

-- Coluna de restricao nao presente na select list
```

Comando SELECT: Exemplos

- Mais exemplos (SQL é case-insensitive):

```
-- Dados apenas das editoras da
-- California (restricao)
select pub_name, city, state
from publishers
where state = 'CA'

-- Apenas editoras de Washington
select pub_name, city, state
from publishers
where state = 'WA'

-- Coluna de restricao nao presente na select list
select pub_name, city
from publishers
where state = 'TX'
```

Comando SELECT: Operadores

- Comparação: $=$, $<>$, $<$, $>$, $<=$, $>=$
- Lógicos: not, and, or
- Range da valores: between
- Conjuntos: in, exists
- Match parcial de texto: like
- Nulabilidade: is null, is not null

Comando SELECT: Exemplos de uso de operadores

- Operadores de comparação:

```
-- Livros com preco inferior a $7
select title, price
from titles
where price < 7

-- Livros com preco de $7 ou menos
```

Comando SELECT: Exemplos de uso de operadores

- Operadores de comparação:

```
-- Livros com preco inferior a $7
select title, price
from titles
where price < 7

-- Livros com preco de $7 ou menos
select title, price
from titles
where price <= 7

-- Livros que nao sejam de negocios
```

Comando SELECT: Exemplos de uso de operadores

- Operadores de comparação:

```
-- Livros com preco inferior a $7
select title, price
from titles
where price < 7

-- Livros com preco de $7 ou menos
select title, price
from titles
where price <= 7

-- Livros que nao sejam de negocios
select title, type, price
from titles
where type <> 'business'
```

Comando SELECT: Exemplos de uso de operadores

- Operadores lógicos:

```
-- Livros de negocios que custam mais que $15
```

Comando SELECT: Exemplos de uso de operadores

- Operadores lógicos:

```
  -- Livros de negocios que custam mais que $15
select title, price, type
from titles
where type = 'business' and price > 15

-- Livros de negocios ou que custam menos que $15
```

Comando SELECT: Exemplos de uso de operadores

- Operadores lógicos:

```
  -- Livros de negocios que custam mais que $15
select title, price, type
from titles
where type = 'business' and price > 15

-- Livros de negocios ou que custam menos que $15
select title, price, type
from titles
where type = 'business' or price < 15

-- Livros que no sao de psicologia
```

Comando SELECT: Exemplos de uso de operadores

- Operadores lógicos:

```
  -- Livros de negocios que custam mais que $15
select title, price, type
from titles
where type = 'business' and price > 15

-- Livros de negocios ou que custam menos que $15
select title, price, type
from titles
where type = 'business' or price < 15

-- Livros que no sao de psicologia
select title, price, type
from titles
where not type = 'psychology'
```

Comando SELECT: Exemplos de uso de operadores

- Operadores de range:

```
-- Livros com preco entre $15 e $25 inclusive
select title, price, type
from titles
where price between 15 and 25
```

Comando SELECT: Exemplos de uso de operadores

- Operadores de range:

```
-- Livros com preco entre $15 e $25 inclusive
select title, price, type
from titles
where price between 15 and 25

-- Equivalente a escrever:
select title, price, type
from titles
where price >=15 and price <= 25
```

Comando SELECT: Exemplos de uso de operadores

- Operadores de conjuntos:

```
-- Livros de negocios ou psicologia
select title, price, type
from titles
where type in ( 'business', 'psychology' )
```

Comando SELECT: Exemplos de uso de operadores

- Operadores de conjuntos:

```
-- Livros de negocios ou psicologia
select title, price, type
from titles
where type in ( 'business', 'psychology' )

-- Equivalente a escrever:
select title, price, type
from titles
where type = 'business' or type = 'psychology'

-- Autores que nao vivem em Berkeley
-- nem em Oakland
```

Comando SELECT: Exemplos de uso de operadores

- Operadores de conjuntos:

```
-- Livros de negocios ou psicologia
select title, price, type
from titles
where type in ( 'business', 'psychology' )

-- Equivalente a escrever:
select title, price, type
from titles
where type = 'business' or type = 'psychology'

-- Autores que nao vivem em Berkeley
-- nem em Oakland
select au_id, au_lname, city
from authors
where city not in ( 'Berkeley', 'Oakland' )
```

Comando SELECT: Exemplos de uso de operadores

- Operadores de match parcial de texto:

```
-- Todos os livros cujo titulo inicia por 'S'
```

Comando SELECT: Exemplos de uso de operadores

- Operadores de match parcial de texto:

```
  -- Todos os livros cujo titulo inicia por 'S'
select title, type
from titles
where title like 'S%'

-- Os livros que tem a palavra 'computer'
-- no titulo
```

Comando SELECT: Exemplos de uso de operadores

- Operadores de match parcial de texto:

```
  -- Todos os livros cujo titulo inicia por 'S'
select title, type
from titles
where title like 'S%'

-- Os livros que tem a palavra 'computer'
-- no titulo
select title, type
from titles
where title like '%computer%'
```

Comando SELECT: Exemplos de uso de operadores

- NULL significa “valor indefinido”
- Não é igual a ” (string vazia)
- Nada é igual a NULL, nem mesmo NULL
- Operadores de nulabilidade:

```
-- Apenas livros com preco indefinido
```

Comando SELECT: Exemplos de uso de operadores

- NULL significa “valor indefinido”
- Não é igual a ” (string vazia)
- Nada é igual a NULL, nem mesmo NULL
- Operadores de nulabilidade:

```
   -- Apenas livros com preco indefinido
select title, price, type
from titles
where price is null

-- Apenas livros com preco definido
```

Comando SELECT: Exemplos de uso de operadores

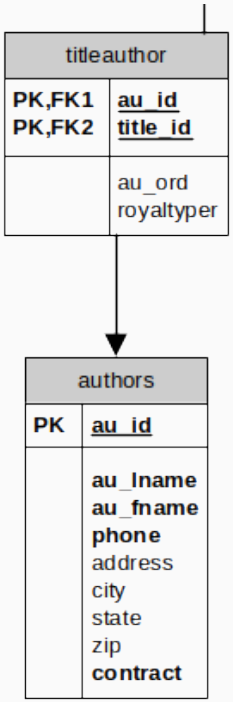
- NULL significa “valor indefinido”
- Não é igual a ” (string vazia)
- Nada é igual a NULL, nem mesmo NULL
- Operadores de nulabilidade:

```
  -- Apenas livros com preco indefinido
select title, price, type
from titles
where price is null

-- Apenas livros com preco definido
select title, price, type
from titles
where price is not null
```

Comando SELECT: Junções (JOIN)

- O relacionamento entre tabelas é feito por meio de chaves, que identificam unicamente cada linha
- Chaves podem ser (basicamente) de dois tipos:
  - Primária (PK): identifica linha em sua tabela de origem
  - Estrangeira (FK): identifica linha de uma tabela em outra
- Exemplo: a coluna *au_id* é PK da tabela *authors* e FK na tabela *titleauthor*



Comando SELECT: Junções (JOIN)

- Joins são utilizados para obter dados de mais de uma tabela em um comando
- Cada JOIN conecta tabelas aos pares, mas podem aparecer quantas cláusulas JOIN forem necessárias em um comando
- Podem ser de 3 tipos:
  - INNER: traz todos os dados em que há match das chaves nas duas tabelas envolvidas
  - OUTER: traz todos os dados de uma das tabelas independentemente de existirem na outra tabela
  - CROSS: produto cartesiano entre as tabelas, sem match de chaves
- Exemplo:

```
-- Titulo e o nome da editora dos livros
SELECT title, pub_name
FROM titles INNER JOIN publishers
ON pub_id = pub_id
```

Comando SELECT: Junções (JOIN)

- Problema com este exemplo: coluna *pub_id* tem o mesmo nome nas duas tabelas
- Prática comum de modelagem, facilita a identificação visual de quais tabelas se relacionam
- Gera ambiguidade na select list: a coluna *pub_id* de qual tabela deve ser retornada?
- (no caso do INNER JOIN são iguais por definição, mas no OUTER JOIN pode ser NULL)

```
-- Titulo e o nome da editora dos livros
SELECT title, pub_name, pub_id
FROM titles INNER JOIN publishers
ON pub_id = pub_id
```

Comando SELECT: Junções (JOIN)

- Problema com este exemplo: coluna *pub_id* tem o mesmo nome nas duas tabelas
- Prática comum de modelagem, facilita a identificação visual de quais tabelas se relacionam
- Gera ambiguidade na select list: a coluna *pub_id* de qual tabela deve ser retornada?
- (no caso do INNER JOIN são iguais por definição, mas no OUTER JOIN pode ser NULL)

```
-- Titulo e o nome da editora dos livros
SELECT title, pub_name, pub_id -- isso funciona?
FROM titles INNER JOIN publishers
ON pub_id = pub_id
```

Comando SELECT: Junções (JOIN)

- Solução 1: qualificar as colunas com o nome da tabela:
  - Em comandos grandes começa a ficar repetitivo

```
SELECT titles.title,
    publishers.pub_name,
    publishers.pub_id
FROM titles INNER JOIN publishers
ON publishers.pub_id = titles.pub_id
```

- Solução 2: atribuir aliases às tabelas:
  - Bem mais compacto

```
SELECT t.title, p.pub_name, p.pub_id
FROM titles t INNER JOIN publishers p
ON t.pub_id = p.pub_id
```

Comando SELECT: Ordenação

- As linhas inseridas em uma tabela não têm ordem pré-determinada
- A cláusula ORDER BY permite ordenar os resultados

```
-- Resultado ordenado pelo preco
select title, price, type
from titles
where type in ( 'business', 'psychology' )
order by price

-- Resultado em ordem descendente
-- pelo preco e ascendente por tipo
select title, price, type
from titles
where type in ( 'business', 'psychology' )
order by price desc, type
```

Comando SELECT: Agregação

- Funções de agregação calculam valores sobre o resultado de um comando
- Exemplo:

```
select count(*) as 'quantidade',
    min(price) as 'mais_barato',
    max(price) as 'mais_caro',
    sum(price) as 'soma_precos',
    avg(price) as 'media_precos'
from titles
```

Comando SELECT: Agregação + GROUP BY

- GROUP BY agrupa resultados e possibilita o uso de funções de agregação sobre cada grupo
- Exemplo:

```
-- media, mais caro e mais barato por tipo
select type, avg(price), min(price), max(price)
from titles
group by type

-- media de preco e vendas por tipo e editora
select type as 'tipo', pub_id as 'editora',
    avg(price), sum(ytd_sales) as 'vendas'
from titles
where type <> 'UNDECIDED'
group by type, pub_id
```

- Importante: No GROUP BY, a select list pode conter apenas as colunas de agrupamento e as agregações

Comando SELECT: Funções de apoio

- Colunas podem ser transformadas por meio de funções
- Existem funções de vários tipos:
  - Manipulação de texto
  - Aritméticas e de manipulação numérica
  - Manipulação de datas
  - Outros
- Nomes de funções variam bastante entre diferentes produtos, mas em geral, possuem funcionalidade similar

Comando SELECT: Funções de apoio

- Exemplos de funções de manipulação de texto

Uso	**Postgresql**	**MySQL**	**MS SQL**	**Oracle**
Tamanho	length	char_length	len	length
Concatenação	||	concat	+	||
Substring	substring	substr	substring	substr
Minúsculas	lower	lcase	lower	lower
Maiúsculas	upper	ucase	upper	upper
Posição de caracter	position	instr	charindex	instr

Comando SELECT: Funções de apoio

- Exemplos de funções aritméticas e de manipulação numérica

Uso	**Postgresql**	**MySQL**	**MS SQL**	**Oracle**
Operadores	+,-,/,*	+,-,/,*	+,-,/,*	+,-,/,*
Módulo	mod	mod	%	mod
Potênciação	exp	exp	exp	exp
Piso	floor	floor	floor	floor
Teto	ceiling	ceiling	ceiling	ceil
Raiz quadrada	sqrt	sqrt	sqrt	sqrt

Comando SELECT: Funções de apoio

- Funções de manipulação de datas

Uso	**Postgresql**	**MySQL**	**MS SQL**	**Oracle**
Data e hora	now	now	getdate	sysdate
Parte	date_part,extract	extract	datepart	extract
Diferença	-	datediff	datediff	-
Soma	+	date_add	dateadd	+

Comando SELECT: Funções de apoio

- Exemplos de outras funções importantes

Uso	**Postgresql**	**MySQL**	**MS SQL**	**Oracle**
Conversão	cast	cast	convert	cast
Valor se NULL	isnull	ifnull	isnull	nvl
NULL se valor	nullif	nullif	nullif	nullif
Primeiro valor não NULL	coalesce	coalesce	coalesce	coalesce

Comando INSERT

- Insere dados em uma tabela
- Duas formas básicas:
  - Insere uma linha por vez
  - Insere resultados de um comando SELECT

```
-- uma linha por vez
INSERT INTO <tabela> (col1, col2, ...)
VALUES (val1, val2, ...)
-- resultados do SELECT
INSERT INTO <tabela> (col1, col2, ...)
SELECT val1, val2, ... FROM ...
```

- A palavra-chave INTO é opcional

Comando INSERT

- A lista de colunas pode ser omitida, desde que os valores estejam na ordem correta:

```
-- se as colunas da tabela esperam val1, val2,
-- o comando vai falhar
INSERT INTO <tabela>
VALUES (val2, val1, ...)
```

- Colunas que aceitam NULL podem ser omitidas, ou valores NULL podem ser passados explicitamente:

```
-- se col2 aceita NULL, isso funciona
INSERT INTO <tabela> (col1, col2, col3)
VALUES (val1, NULL, val3)
-- e isso tambem
INSERT INTO <tabela> (col1, col3)
VALUES (val1, val3)
```

Comando INSERT

- Se a coluna possui valor default definido na tabela, este pode ser usado no INSERT:

```
-- se col2 nao aceita NULL mas tem valor default,
--isso funciona
INSERT INTO <tabela> (col1, col2, col3)
VALUES (val1, default, val3)
-- e isso tambem
INSERT INTO <tabela> (col1, col3)
VALUES (val1, val3)
```

Comando UPDATE

- Altera valores em colunas de uma tabela
- Duas formas básicas:
  - Atualiza baseado em uma cláusula WHERE (ou nenhuma, o que atualiza todas as linhas)
  - Atualiza baseado nos resultados de outras tabelas (a.k.a *correlated update*)
- Exemplos:

```
-- Aumenta em 5% o preco dos livros de psicologia
UPDATE titles
SET price = price * 1.05
WHERE type = 'psychology'

-- Zera as vendas do ano corrente para TODOS os
-- livros da tabela titles
UPDATE titles
SET ytd_sales = 0
```

Comando UPDATE

- *Correlated update*
  - O comando SELECT calcula a soma das vendas de cada livro, para cada linha da tabela *titles*
  - Isto ocorre porque a coluna *titles.title_id* foi referenciada na cláusula WHERE do SELECT:

```
-- Atribui a coluna ytd_sales os totais
-- de vendas de cada livro no ano de 1999
UPDATE titles
SET ytd_sales = (
    SELECT sum(qty)
    FROM sales
    -- note a referencia para a tabela titles
    WHERE sales.title_id = titles.title_id
    AND ord_date
        BETWEEN '19990101' AND '19990131')
```

Comando MERGE

- Atualiza dados em uma tabela baseado em dados de outra e:
  - Se existe a linha, atualiza
  - Se não existe a linha, insere
- Também conhecido como *upsert* (update + insert):

```
MERGE <tabela destino>
USING <tabela origem ou query>
ON <condicao de merge>
-- se linha existe no destino
WHEN MATCHED
THEN <comando de update>
-- se linha nao existe no destino
WHEN NOT MATCHED
THEN <comando de insert>
```

- Cuidado: por ser um comando relativamente recente, a sintaxe pode variar bastante entre diferentes produtos

Comando DELETE

- Apaga linhas de uma tabela
- Aceita cláusula WHERE para selecionar linhas
- Exemplo:

```
-- apaga livros cujo preco exceda 1.000.000
DELETE titles
WHERE price > 1000000
```

- Pode apagar linhas de uma tabela baseado em dados de outras tabelas usando JOIN:

```
-- apaga livros da editora 'Imaginary Publisher'
DELETE titles
FROM titles t INNER JOIN publishers p
ON t.pub_id = p.pub_id
WHERE p.pub_name = 'Imaginary␣Publisher'
```

Comando TRUNCATE TABLE

- Apaga todas as linhas de uma tabela
- Mais rápido do que o comando DELETE porque não trabalha no nível lógico (linha) e sim no nível físico (alocação no disco)
- Tipicamente requer permissões especiais no banco de dados
- Use com cuidado: na maioria dos produtos não é reversível
- Exemplo:

```
TRUNCATE TABLE mytable
```

Comando SELECT..INTO

- Cria nova tabela com base no resultado de um comando SELECT
- Útil para materializar resultados intermediários em análise exploratória de dados
- Requer permissão para criar tabelas no banco de dados
- Se mal utilizado, faz balbúrdia no banco de dados, com a proliferação de tabelas estranhas ao modelo :-)
- Exemplo:

```
SELECT titles.title,
    publishers.pub_name,
    publishers.pub_id
INTO titles_with_publishers
FROM titles INNER JOIN publishers
ON publishers.pub_id = titles.pub_id
```