Unidade 3

Bancos de Dados Relacionais e Linguagem SQL

3.4 Constraints e Views

Constraints

- Sem regras, não se pode confiar na integridade do banco de dados.
- *Constraints* são usadas para impedir a entrada de dados inválidos em tabelas de acordo com as regras de existência e relacionamento definidas nos modelos de dados.
- Salários negativos, registros de chave estrangeira sem a referência da chave primária, gravação de dados nulos em colunas que participam destas regras são alguns exemplos.

Constraints

- *Constraints* são regras de banco de dados que estão armazenadas no dicionário de dados e impõem regras para os dados sempre que ocorre modificação em coluna (s) de uma linha.
- Elas podem também impedir a exclusão de uma tabela se houver dependências de outras tabelas.
- Na instrução CREATE TABLE pode-se definir dois níveis de constraints: nível da coluna e nível da tabela.

Constraints – Nível da Coluna

- ***Constraint no nível da coluna:*** referencia uma única coluna e deve ser definida na instrução CREATE TABLE como parte da definição de coluna.

```
CREATE TABLE city (
  city_id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
  city VARCHAR(50) NOT NULL,
  country_id SMALLINT UNSIGNED NOT NULL,
  last_update TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY  (city_id),
  KEY idx_fk_country_id (country_id),
  CONSTRAINT `fk_city_country` FOREIGN KEY (country_id) REFERENCES country (country_id) ON DELETE RESTRICT ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Constraints – Nível da Coluna

- ***Constraint no nível da tabela:*** são listadas separadamente das definições de coluna na instrução CREATE TABLE após a definição de todas as colunas da tabela.

```
CREATE TABLE t1 (
  c1 int(11) DEFAULT NULL,
  c2 int(11) DEFAULT NULL,
  c3 int(11) DEFAULT NULL,
  CONSTRAINT c1_nonzero CHECK ((c1 <> 0)),
  CONSTRAINT c2_positive CHECK ((c2 > 0)),
  CONSTRAINT t1_chk_1 CHECK ((c1 <> c2)),
  CONSTRAINT t1_chk_2 CHECK ((c1 > 10)),
  CONSTRAINT t1_chk_3 CHECK ((c3 < 100)),
  CONSTRAINT t1_chk_4 CHECK ((c1 > c3))
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Constraints – Nível da Coluna

- *Constraints* podem se referir a mais de uma coluna (uma chave composta) e são definidas no nível da tabela.
- *NOT NULL* pode ser especificada apenas no nível da coluna, e não no nível da tabela enquanto *UNIQUE*, *PRIMARY KEY, FOREIGN KEY* e *CHECK* podem ser definidas no nível da coluna ou da tabela.

Constraints – Tipos de Constraints

- *NOT NULL:* exige que, para cada linha incluída na tabela, com esta descrição, um valor deve existir para essa coluna. É costume usar o sufixo_nn no nome desta *constraint*.
- *UNIQUE:* exige que todo valor em uma coluna ou conjunto de colunas(uma chave composta) seja exclusivo, ou seja, não pode haver valores duplicados nas linhas da tabela. Se a combinação de duas ou mais colunas deve ser exclusiva a cada entrada, a *constraint* é considerada uma chave exclusiva composta. É costume usar o sufixo_uk no nome desta *constraint*.

Constraints – Tipos de Constraints

- *PRIMARY KEY:* é uma regra que determina que os valores em uma coluna ou combinação de colunas devem identificar exclusivamente cada linha de uma tabela. É costume usar o sufixo_pk no nome desta *constraint*.
- *FOREIGN KEY:* também chamadas de *constraints* de **integridade referencial.** Designa uma coluna ou combinação de colunas como uma chave estrangeira vinculada à chave primária (ou a uma chave exclusiva) de outra tabela, e esse elo é a base do relacionamento entre tabelas.

Constraints – *FOREIGN KEY*

- Um valor de chave primária pode existir sem um valor de chave estrangeira correspondente. No entanto, uma chave estrangeira precisa ter uma chave primária correspondente.
- Desta forma, é preciso definir uma chave primária pai antes de criar uma chave estrangeira em uma tabela filho. É costume usar o sufixo_fk no nome desta *constraint*.

Constraints – *FOREIGN KEY*

- O uso da opção *ON DELETE CASCADE* quando se define uma chave estrangeira permite às linhas dependentes na tabela filho serem excluídas quando uma linha na tabela pai é excluída.
- Se a chave estrangeira não tiver uma opção *ON DELETE CASCADE*, as linhas referenciadas na tabela pai não poderão ser excluídas.
- Em outras palavras, a *constraint FOREIGN KEY* da tabela filho inclui a permissão ON DELETE CASCADE, possibilitando que sua tabela pai exclua as linhas às quais se refere.

Constraints – *FOREIGN KEY*

- O uso da opção *ON DELETE SET NULL* as linhas na tabela filho possam ser preenchidas com valores nulos quando uma linha na tabela pai é excluída.
- Pode ser útil quando o valor da tabela pai está sendo alterado para um novo número, um exemplo é converter números de inventário para códigos de barra.
- Não excluo os registros e quando os novos códigos de barra forem incluídos na tabela pai, posso inseri-los na tabela filho sem precisar recriar totalmente cada linha da tabela filho.

Constraints – Tipos de Constraints

- *CHECK:* define explicitamente uma condição que deve ser atendida para satisfazer a *constraint*, cada linha na tabela deve tornar a condição Verdadeira ou desconhecida (devido a um valor nulo).
- A condição de uma *constraint CHECK* pode se referir a qualquer coluna na tabela especificada, mas não a colunas de outras tabelas.
- Não há limite para o número de *constraints CHECK* que você pode definir em uma coluna.

Constraints – Gerenciamento

- *ALTER TABLE* é usada para fazer mudanças nas constraints em tabelas existentes que incluem o acréscimo, a eliminação, a ativação ou a desativação de *constraints* e assim como a inclusão de uma *constraint* NOT NULL a uma coluna.
- Para que as regras definidas pelas *constraints* de integridade, funcionem é preciso ativá-las, porém , há situações onde é preferível desativar temporariamente as *constraints* de integridade por questões de desempenho como quando estamos carregando grandes quantidades de dados em uma tabela ou realizando operações em lote.

Views

- Assim como uma tabela, uma ***view*** é um objeto de banco de dados, porém, ***views*** não são tabelas físicas ou seja, não possuem arquivos de dados alocados para elas.
- Podemos entende-las como representações lógicas de tabelas existentes ou de outra **view** assim elas não possuem dados próprios.
- Elas funcionam como um mecanismo através do qual se pode ver ou alterar os dados das tabelas.

Views

- As tabelas nas quais uma *view* se baseia são chamadas de tabelas básicas e sua criação é baseada em uma consulta armazenada como uma instrução SELECT no dicionário de dados.
- As *views* restringem o acesso aos dados da tabela básica, pois podem exibir colunas selecionadas e podem ser usadas para reduzir a complexidade da execução de consultas com base em instruções SELECT mais complicadas.
- O usuário da *view* não vê o código subjacente nem o modo como ele é criado.

Views

```
CREATE VIEW Resultado_Mensal AS
SELECT d.dept_name, t.title as "Cargo", count(de.emp_no) as "Qtde de funcionários com este cargo",
       ROUND((AVG(salary)/12),2) as "Média de salário mensal",
       ROUND((MAX(salary)/12),2) as "Maior salário mensal do departamento",
       ROUND((MIN(salary)/12),2) as "Menor salário mensal do departamento"
FROM dept_emp de INNER JOIN employees e   ON (de.emp_no=e.emp_no)
                 INNER JOIN departments d ON (de.dept_no=d.dept_no)
                 INNER JOIN salaries    s ON (s.emp_no=e.emp_no)
                 INNER JOIN titles      t ON (e.emp_no=t.emp_no)
WHERE de.to_date = '9999-01-01'
AND    s.to_date = '9999-01-01'
GROUP BY d.dept_name, t.title  WITH ROLLUP;
```

```
SELECT * FROM Resultado_Mensal;
```

Views

- Por razões de desempenho, a subconsulta que define a *view* não deve conter uma cláusula *ORDER BY*. Essa cláusula é melhor especificada quando você recupera dados da *view.*
- Utiliza-se a opção OR REPLACE para mudar a definição da *view* sem precisar eliminá-la ou conceder novamente os privilégios de objeto que já pertenceram a ela.
-  Sempre utilize esta opção para não ter que refazer tudo e correr o risco de alterações indevidas com manutenção de versões anteriores à VIEW.

Views – Opções de criação

COMANDO	OPERAÇÃO
OR REPLACE	Recria a view, caso ela já exista.
FORCE	Cria a view, mesmo que as tabelas básicas existam ou não.
NOFORCE	Cria a view apenas se a tabela básica existir (padrão).
WITH CHECK OPTION	Especifica que as linhas permaneçam acessíveies à view após operações de inserção ou atualização.
CONSTRAINT *	É o nome atribuído à *constraint* CHECK OPTION.
WITH READ ONLY *	Garante que nenhuma operação DML possa ser executada na view.

(*) Disponível em apenas algumas distribuições

Views – Classificação das Views

- Por razões de desempenho, a subconsulta que define a *view* não deve conter uma cláusula *ORDER BY*. Essa cláusula é melhor especificada quando você recupera dados da *view.*
- Utiliza-se a opção OR REPLACE para mudar a definição da *view* sem precisar eliminá-la ou conceder novamente os privilégios de objeto que já pertenceram a ela.

Views - Classificação das Views

- **Simples:** a subconsulta deriva de dados originários de apenas uma tabela e não contém uma função de junção nem funções de grupo.
- **Como se trata de uma *view* simples, as operações *INSERT*, *UPDATE*, *DELETE* e *MERGE* que afetam a tabela básica podem ser executadas através da *view*.**
-  Para não correr este risco utiliza-se a opção WITH READ ONLY.
- **Complexas:** são aquelas que podem conter funções de grupo e junções.

Views - Classificação das Views

```
USE world;

CREATE OR REPLACE VIEW Paises_do_Caribe AS
SELECT Code,Name,Population,Capital
FROM country
WHERE region = "Caribbean"
WITH CASCADED CHECK OPTION;

 UPDATE Paises_do_Caribe
 SET Name = 'Aruba4'
 WHERE Code = 'ABW';
```

```
UPDATE Paises_do_Caribe
SET Name = 'Aruba4'
WHERE Code = 'ABW';

-- Observem como isso pode ser prigoso...

SELECT *
FROM country
WHERE Code= 'ABW';
```

(*) No Mysql os bloqueios são tratados com comandos DCL que veremos em breve.

Views – Classificação das Views

- A diferença entre as *views* simples e complexas está em sua capacidade para permitir operações DML através de uma *view* onde:
- Nas ***views* simples**, operações DML podem ser executadas através de uma view e para ***views* complexas**, operações DML nem sempre são permitidas.
- Não é possível remover uma linha de uma tabela básica subjacente se a *view* contiver um os elementos: funções de grupo, cláusula GROUP BY, palavra-chave DISTINCT, colunas definidas por expressões e não inclui colunas NOT NULL nas tabelas básicas.

Views – Gerenciando as *Views*

- *Views* são armazenadas como instruções *SELECT* no dicionário de dados e somente o criador ou os usuários com o privilégio *DROP VIEW* podem remover uma *view*.
- *Views* em linha também são chamadas de subconsultas na cláusula FROM podendo-se inserir uma subconsulta na cláusula FROM exatamente como se a subconsulta fosse um nome de tabela.
- As *views* em linha **são muito usadas para simplificar consultas complexas**, removendo as operações de junção e condensando várias consultas em uma.

Views - Gerenciando as Views

```
USE employees;

SELECT e.last_name, vil.dept_no, vil.maxsal
FROM employees e, dept_emp de,
                (SELECT dept_no, max(salary) maxsal
                 FROM salaries INNER JOIN dept_emp USING (emp_no)
                 GROUP BY dept_no) vil, salaries sal
WHERE e.emp_no = de.emp_no
AND vil.dept_no  = de.dept_no
AND sal.emp_no=e.emp_no
AND sal.salary = vil.maxsal;
```

Views – Análise TOP-N

- Análise top-n é uma operação SQL usada para organizar resultados e é útil usar a análise top-n quando você quiser recuperar os cinco registros superiores, ou n registros superiores, de um conjunto de resultados retornado por uma consulta.
- No MySQL é utilizada a palavra reservada *LIMIT* para esta aplicação.
- Em outros SGBDS utiliza-se o recurso *ROWNUM* que é uma coluna virtual que enumera os registros de acordo com a ordem de organização.

Referências bibliográficas

ELMASRI, R., NAVATHE, S. B., Sistemas de Banco de Dados: Fundamentos e Aplicações. 3ª Ed., Editora LTC, 2002.

MANNINO, Michael V. Projeto, Desenvolvimento de Aplicações e Administração de Banco de Dados. 3ª. Ed. Porto Alegre. Bookman. 2008.