LABORATÓRIO DE BANCO DE DADOS

IBD100

Programa

- Histórico da AR + SQL
  - Origem
  - Conceitos fundamentais de AR
  - O projeto da linguagem → Sequel
  - A evolução ➔ Square (for a da proposta da AR !!)
  - Voltando ao Sequel → Sequel2 ➔ SystemR
  - Evoluindo → SQL (~1980)
  - Variações da proposta original
  - Símbolos adaptados
  - As linguagens dos SGBDs e a SQL (indicando caminhos de evolução)
  - Os padrões SQL (ANSI).
- As operações de AR → SQL
  - Onde praticar? Base PostgreSQL
  - Os comandos padrões de SQL ≠ Comandos dos SGBDs (por que?)
  - Os módulos de linguagens dos SGBD e os Padrões SQL

IBD100
ÁLGEBRA RELACIONAL
+ SQL

João Maurício Hypólito
prof.oao.hypolito@gmail.com

- Histórico da AR – Origem
  - 1970 – Edgard F. Cood apresenta o Modelo Relacional e muda a forma de desenvolver a modelagem de dados. Conceito: Teoria dos conjuntos. TABELA = {tuplas}
  - Para processar as tabelas ele apresenta a Álgebra Relacional
  - As tuplas são '**elementos**' dos '**conjuntos**' (tabelas), então as operações de elementos de conjuntos expressam um raciocínio de processamento de tuplas das tabelas.
- Histórico da AR – Conceitos fundamentais:
  - Tabela={ tuplas }
  - Tabelas são definidas por seus campos.
    - Entre tabelas com igual ESQUEMA é possível fazer: **U, –** e ∩
    - ESQUEMA igual (campos com mesmo **nome**, na mesma **ordem** (na tabela), do mesmo **tipo de dado** e com o mesmo **tamanho** na tabela) ➔ tabelas União Compatível
  - Operações: Seleção, Projeção, Produto Cartesiano e Junção.
    (mas sempre com a ideia de conjunto gerando conjunto)
  - Permitem a combinação de dados das tabelas.

João Maurício Hypólito - prof.joao.hypolito@gmail.com

IBD100-ÁLGEBRA RELACIONAL + SQL

- Histórico da AR – Variações da Proposta Original
  - A Álgebra Relacional foi apresentada com uma notações de símbolos gregos: π σ X, ⋈, etc.
  - Esta notação de símbolos não é adequada para edição de operações usando editores de texto simples. Por isso, usuários fizeram a proposta de símbolos em texto simples para as operações da AR.
  - π – projeção – passa a ser escrita com o nome da tabela seguido de [] delimitando os nomes de campos.
  - σ – seleção – passa a ser escrita com o nome da tabela seguido da condição escrita dentro de []
  - X – produto cartesiano – passa a ser escrita com os nomes das tabelas escritos de cada lado do 'X'.
  - ⋈ – Junção - passa a ser escrita com os nomes das tabelas de cada lado de [] e com a condição dentro dos [].
  - ∩ passa a ser ^
  - **U** passa a ser U,
  - **-** passa a ser – e
- Assim as operações podem ser reapresentadas assim:

Operação	Símbolo adaptados
Seleção	Tab[condição]
Projeção	Tab[lista de campos]
Produto Cartesiano	Tab1 X Tab2
Junção	Tab1[condição]Tab2
União	Tab1 U Tab2
Subtração	Tab1 – Tab2
Intersecção	Tab1 ^ Tab2

João Maurício Hypólito - prof.joao.hypolito@gmail.com

IBD100-ÁLGEBRA RELACIONAL + SQL

- Histórico do SQL.
  - Consequência da Álgebra Relacional:
    - Criação da Linguagem de Manipulação para pesquisa dos dados nos Banco de Dados
  - 1974: Donald D. Chamberlin e Raymond Boyce (pesquisadores do "IBM San Jose Research Center"): artigo sugerindo a forma da linguagem de consulta estruturada. A linguagem foi chamada de SEQUEL;
  - 1975: Chamberlin, Boyce, King e Hammer apresentam a SQUARE (com símbolos matemáticos), semelhante a SEQUEL (mas esta tinha termos em inglês);
  - 1976: Chamberlin apresenta a SEQUEL2 que passa a ser usada como linguagem de consulta para o banco de dados que se pesquisava na IBM, o ***sistema R***: criam-se grupos de usuários, a linguagem começa a evoluir rapidamente ;
  - 1977: O SystemR-IBM (com SEQUEL/2) torna-se operacional + Oracle (Relacional)
  - 1980: Chamberlin sumariza as experiências dos usuários com a linguagem (seu nome a esta altura já era SQL – Structured Query Language).
  - 1983: IBM lança o DB2. Outros produtos no mercado: Sybase, Ingres, Progress), SQL é padrão de fato;
  - 1986: SQL é homologada como linguagem padrão ANSI para tratamento de dados em BD Relacionais basicamente a linguagem desenvolvida para o SystemR-IBM

- Histórico do SQL.
  - Os padrões da linguagem
    - 1987: Padrão ANSI é aceito pelo ISO (ANSI-SQL:86)
    - 1989: SQL (ANSI-SQL:89) – incorpora comandos para junções abertas e completas
      - Apenas dois anos depois do primeiro padrão e já apresentam uma revisão?
        Isso porquê o mundo ainda estava vivendo a primeira onda de código aberto e muitos usuários de SGBD relacionais contribuíam com a SQL.
    - 1992: Os comitês ISO e ANSI apresentam SQL2 (SQL:1992) incorpora características de tratamento de integridade (de CP e CE), tabelas temporárias, novas funções, expressões nomeadas, valores únicos, instrução CASE, etc.
    - 1999: SQL:1999 – implementação de expressões regulares, recursos de orientação a objetos, queries recursivas, triggers, novos tipos de dados (boolean, LOB, array e outros), novos predicados etc.
    - 2003: SQL:2003: - inclui suporte básico ao padrão XML, sequências padronizadas, instrução MERGE, colunas com valores auto-incrementais etc.
    - 2006: SQL:2006: - Não inclui mudanças significativas para as funções e comandos SQL. Reforça a interação SQL ←→ XML

João Maurício Hypólito - prof.joao.hypolito@gmail.com

- Depois que desenvolveram a SQL, surgiu a necessidade apresentarem os símbolos para *junção aberta*. Na simbologia fundamental bastaria retirar as barras desenhadas nos lados do símbolo ⋈.
- Mas, essa escrita ficaria complicada na simbologia adaptada. Então isso foi resolvido usando simplesmente o símbolo [[ ]] no lugar de [ ]
- Então a AR com os Símbolos adaptados para as operações (estendidas).

Operação	Símbolo adaptados
Seleção	Tab [[condição]]
Projeção	Tab [[lista de campos]]
Produto Cartesiano	Tab1 X Tab2
Junção Natural (fechada)	Tab1 [[condição]] Tab2
União	Tab1 U Tab2
Subtração	Tab1 – Tab2
Intersecção	Tab1 ^ Tab2
Junção aberta pela direita	Tab1 [[condição][Tab2
Junção aberta pela esquerda	Tab1][condição]] Tab2
Junção completa (aberta pelos dois lados)	Tab1][condição][Tab2

João Maurício Hypólito - prof.joao.hypolito@gmail.com

IBD100-ÁLGEBRA RELACIONAL + SQL

- A Evolução dos SGBDs
  - Os SGBDs são o resultado do amadurecimento das necessidades empresariais para suporte no desenvolvimento de Sistemas de Informação
  - Sistemas de Informação são um conjunto de computadores e outros dispositivos que tem a finalidade de fornecer meios de entrada, armazenamento, processamento e apresentação de saídas de dados que permitam estabelecer parâmetros que melhorem a ***tomada de decisão*** pelas organizações (empresas, escolar, etc).
  - OS SGBD devem dar suporte ao projeto do SI, portanto devem permitir:
    - Criar os arquivos que armazenam os dados e administrar estes arquivos
    - Implementar os programas aplicativos (que contém as regras de negócio das organizações).
  - No grupo de linguagens os SGBDs devem ter os comandos em três categorias:

Categoria	Descrição
DDL - *Data Defination Language*	Permite que sejam caracterizados a estrutura dos arquivos onde se gravam os dados da organização. É possível definir o tamanho dos campos, do arquivo como um todo e seus índices associados
DCL - *Data Control Language*	Linguagem de Definição de Usuários e Grupos de usuários e suas atribuições de acesso aos Dado
DML - *Data Manipulation Language*	Linguagem de Manipulação de Dados. É o grupo de comandos que permite incluir, pesquisar alterar e exclui linhas de uma tabela. A IBM ainda subdivide este grupo em: DQL – Comandos de CONSULTA e DTL – Comandos de ATUALIZAÇÃO – inserir, alterar e excluir.

João Maurício Hypólito - prof.joao.hypolito@gmail.com

- Os padrões da linguagem e os comandos (em categorias) dos SGBDs
  - De 1975 até meados da década de 1990 o ANSI estudava o SQL e indicava o caminho que os SGBD iriam trilhar.
  - O ritmo de desenvolvimento dos padrões SQL andou “atrás” do desenvolvimento dos SGBD a partir da década de 1990.
  - Isso fez com que os padrões posteriores à 1992 (SQL:1992) passassem a ter baixa adesão pelos SGBDs (cada um desenvolvia os seus comandos – parecidos com o SQL).
  - Mesmo os SGBDs lançados após 1992 passaram a adotar o SQL:1992 como seu SQL referencial, embora pudessem ter mais funções.
  - Desta forma os SGBD passaram a ter três blocos de linguagens:
    - A linguagem Nativa (incorporando funções específicas do SGBD)
    - A linguagem padrão ANSI (na maioria das vezes: SQL:1992)
    - A linguagem SQL-Nativa – que têm comandos no ‘formato’ SQL mas com a incorporação de funções do SGBD.

João Maurício Hypólito - prof.joao.hypolito@gmail.com

IBD100-ÁLGEBRA RELACIONAL + SQL

- Então os comandos SQL 'distribuídos' nas categorias de comandos dos SGBDs são:

DDL:	DCL:	DML:
CREATE	GRANT	SELECT
ALTER	REVOKE	INSERT
DROP		DELETE
		UPDATE

Todas as Operações da AR começam com este comando SQL

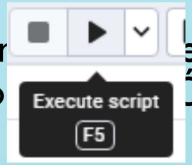
As operações de AR são:

```
SELEÇÃO
PROJEÇÃO
PRODUTO CARTESIANO
JUNÇÃO: NATURAL (FECHADA)
        ABERTA DIREITA
        ABERTA ESQUERDA
        ABERTA DIR←→ESQ (COMPLETA)
UNIÃO
SUBTRAÇÃO
INTERSECÇÃO
```

- Para prática é necessário instalar o SGBD PostgreSQL:
  - BAIXAR O SCRIPT que cria as tabelas para teste de SQL (site e Teams).
  - Acesse o aplicativo pgAdmin 4
  - CRIAR uma base de dados (sugestão de nome: **praticasql**). Nesta base de dados...
  - EXECUTAR o script para criar as tabelas
- Vamos fazer esta prática depois de exercitar algumas recuperações de dados em exercícios teóricos com 'SQL' escritos mesmo À MÃO (mesmo).

João Maurício Hypólito - prof.joao.hypolito@gmail.com

- Ok! Mas por onde começamos a executar os comandos em SQL?
- Acesse seu pgAdmin 4 (se estiver nos laboratórios, CRIE e EXECUTE o Script do seu último BACKUP – por isso foi sugerido você escrever a data de backup no nome do arquivo).
- Acesse a sua base de dados (marque o nome da tabela)
- Acesse o botão secundário do mouse e escolha "Query Tool".
  - Na janela do painel de detalhe existem duas áreas, a superior é para escrita do comando e a inferior é para visualizar o resultado do processamento.
  - Depois de escrever o comando clique no botão



  - Se na área superior você escrever dois ou mais cor executá-los, na área inferior será exibido o resultado do último comando. Para ver o resultado Ó comando marque o comando na área superior e depois clique no botão de executar.
  - Agora começaremos com os comandos de ATUALIZAÇÃO de linhas das tabelas: INSERT, DELETE e UPDATE.

João Maurício Hypólito - prof.joao.hypolito@gmail.com

- Def.: Comando de DML que permite inserir uma ou mais linhas em uma tabela.
  - Nota: foi implementado em SQL sem uma representação formal da AR. Quando se escreve, por exemplo, A ← Funcionarios [[ANO(dtnasc)='1960']] o resultado da operação é INSERIDO em A.
- Sintaxe:

```
INSERT INTO <Tab> [(Campo1, Campo2, ...)]
       VALUES ('valordocampo1', 'valordocampo 2', ...) [,
              ('valordocampo 1, 'valordocampo 2', ...),
               ...,
              ('valordocampo 1', 'valordocampo 2', ...)];
```

  - Exemplo:

```
INSERT INTO DEPARTAMENTOS
       VALUES ('A11','CONTROLE DE CUSTOS1','40','A01','-'),
              ('A12','CONTROLE DE CUSTOS2','120','A01','-');
```


João Maurício Hypólito - prof.joao.hypolito@gmail.com

- Notas:
  - Como o comando foi implementado nos SGBDs ANTES do Padrão ANSI/92, alguns SGBD criaram variações do comando que podem ser interessantes. Estas variações podem ser particulares de um SGBD e não rodarem em outros (teste no seu SGBD).
- Sintaxe:

```
INSERT INTO   <Tab>
       SET  Campo1='valordocampo1', Campo2=valordocampo2 , … ;
```

- Exemplo:

```
INSERT INTO departamentos
           SET DEPTNO='A02',
           DEPTNAME='Prova que dá certo',
           MGRNO='000090',
           ADMRDEPT='A01',
           LOCATION='-';
```

MAS e se quisermos inserir mais de uma linha?
PESQUISE e veja se é possível.

João Maurício Hypólito - prof.joao.hypolito@gmail.com

- Insert com sub query:
  - Em 92 o padrão ANSI formalizou o processamento de sub queries. SUB-QUERY é uma forma de altera a precedência de execução de um comando. Isso se faz com a delimitação de um comando com os sinais: (). Simples assim. As sub queries são executadas na ordem de leitura do comando.
- Sintaxe:

```
INSERT INTO   <Tab> (SUB QUERY);
```

- Exemplo:

```
INSERT INTO deptosbauru
                (SELECT * FROM DEPARTAMENTOS WHERE CDCIDADE='BAURU');
```

Mas E SE quisermos executar DUAS ou + sub queries para gerar um 'bloco de linhas' para o INSERT?
PESQUISE, teste e veja se é possível.

João Maurício Hypólito - prof.joao.hypolito@gmail.com

IBD100-ÁLGEBRA RELACIONAL + SQL – [ATUALIZAÇÃO → **DELETE**]

- Def.: Comando de DML que permite excluir uma ou mais linhas em uma tabela.
  - Nota: foi implementado em SQL sem uma representação formal da AR.
- Sintaxe:

```
DELETE FROM <Tab> [WHERE <condição>];
```

  - Exemplo:

```
DELETE FROM ClienteBOM WHERE ClienteBOM.Limite < 1500;
```

- Notas:
  - **SE o sub comando WHERE for OMITIDO será entendido como WHERE 1=1; Analise: EM quais linhas o 1 é igual à 1? EM TODAS. Então, algo assim: DELETE FROM <tab>; Deleta TODAS as linhas de <tab>.**

prof.joao.hypolito@gmail.com

- Nota: Os comandos de atualização, por um certo tempo, agiam somente em uma tabela. Com o passar do tempo algumas implementações tornaram-se necessárias. Algo como: Deletar os pedidos de funcionários que foram desligados de uma empresa. Com a evolução do SQL (e antes de se tratar as linhas de uma tabela pelo valor de chaves estrangeiras) se tornou possível usar subqueries para tratar esta situação. Comando de DML que permite excluir uma ou mais linhas em uma tabela através do retorno verdadeiro de um relacionamento pode ser estruturado na seguinte forma:

```
DELETE FROM d2 where exists
       (SELECT 1 FROM d1 WHERE d1.CampoDeLig=d2.CampoDeLig);
```

- Exemplo:

```
DELETE FROM pedidos where exists
       (SELECT 1 FROM pedidos WHERE pedidos.CeFunc=Funcionarios.CpFunc);
```

Faça uma análise do comando dentro dos parênteses e veja o 'retorna' do comando. O retorno verdadeiro indica um registro que relaciona os dados de pedidos e funcionarios e essa ligação indica quais registros de pedidos podem ser excluídos.

Veremos outras formas de uso de sub queries (em outras aplicações).

João Maurício Hypólito - prof.joao.hypolito@gmail.com

- Def.: Comando de DML que permite alterar valores dos campos de uma ou mais linhas em uma tabela.
- Sintaxe:

```
UPDATE TABELA SET Atrib1=Valor1[, Atrib2=Valor2, ...Atribn=Valorn]
                  [WHERE condição]  ;
```

- Exemplo:

```
UPDATE Alunos SET Nome="José Antonio da Silva"
              WHERE Alunos.Matricula ="9871234";
```

Mas E SE quisermos executar um UPDATE em DUAS ou + TABELAS?
Bem, no conceito dos comandos de atualização somente uma tabela pode ser atualizada. MAS... Pode ser que o 'seu' SGBD implemente...
PESQUISE, teste e veja se é possível.

- Notas:
  - **SE o sub comando WHERE for OMITIDO será entendido como WHERE 1=1; Analise: EM quais linhas o 1 é igual à 1? EM TODAS. Então, algo assim: DELETE FROM <tab>; Deleta TODAS as linhas de <tab>.**

João Maurício Hypólito - prof.joao.hypolito@gmail.com

- E começa a miscelânea... Alguns SGBD implementaram comandos que facilitaram (ainda mais) o aprendizado.
- No UPDATE usamos o SET para declarar as atribuições... então... Não podemos usar o SET no comando INSERT? Assim:

INSERT INTO TABELA SET $Atrib_1$=$Valor_1$[, $Atrib_2$=$Valor_2$, ...$Atrib_n$=$Valor_n$];

- Pois então, no MariaDB funciona, teste.
- **Outra implementação interessante: o DELETE e UPDATE SEM o WHERE processa somente UMA linha. Se o programador quiser processar mais de uma linha deve escrever ALL depois da primeira palavra do comando, assim**
- **UPDATE ALL... ou**
- **DELETE ALL...**

Encerramos os comandos de Atualização de tabelas em uma base de dados.

LABORATÓRIO DE BANCO DE DADOS

IBD100