Implementando o SQLRDD

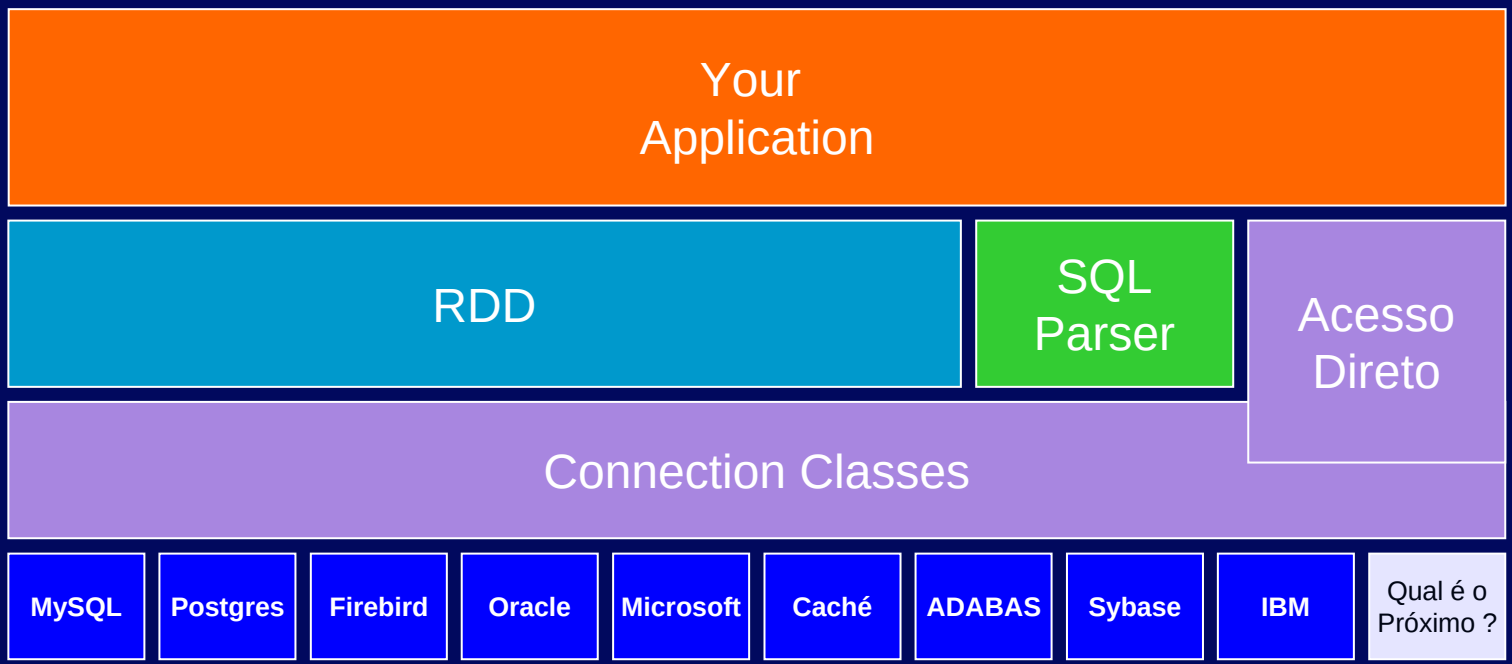
Por que migrar para SQL ?

- Segurança
- Integridade física
- Integridade lógica (controle transacional)
- Performance de acesso não deve ser a principal razão!

Nomenclaturas

- Arquivo, banco de dados **TABELA**
- Campo **COLUNA**
- Registro **LINHA**
- Diretório **BANCO DE DADOS**
- **DML** – Manipulação de dados (seek, replace, skip, etc..)
- **DDL** – Definição de dados (dbCreate(), INDEX, etc.)

Principais componentes do SQLRDD



RDD (Replaceable Database Driver)

- Compatível com DBFCDX
- Suporte a DDL e DML
- Converte o xBase (ISAM) em SQL
  - Cache Workareas
  - Paging Workareas
- Conecta-se aos bancos de dados usando “Connection Classes”

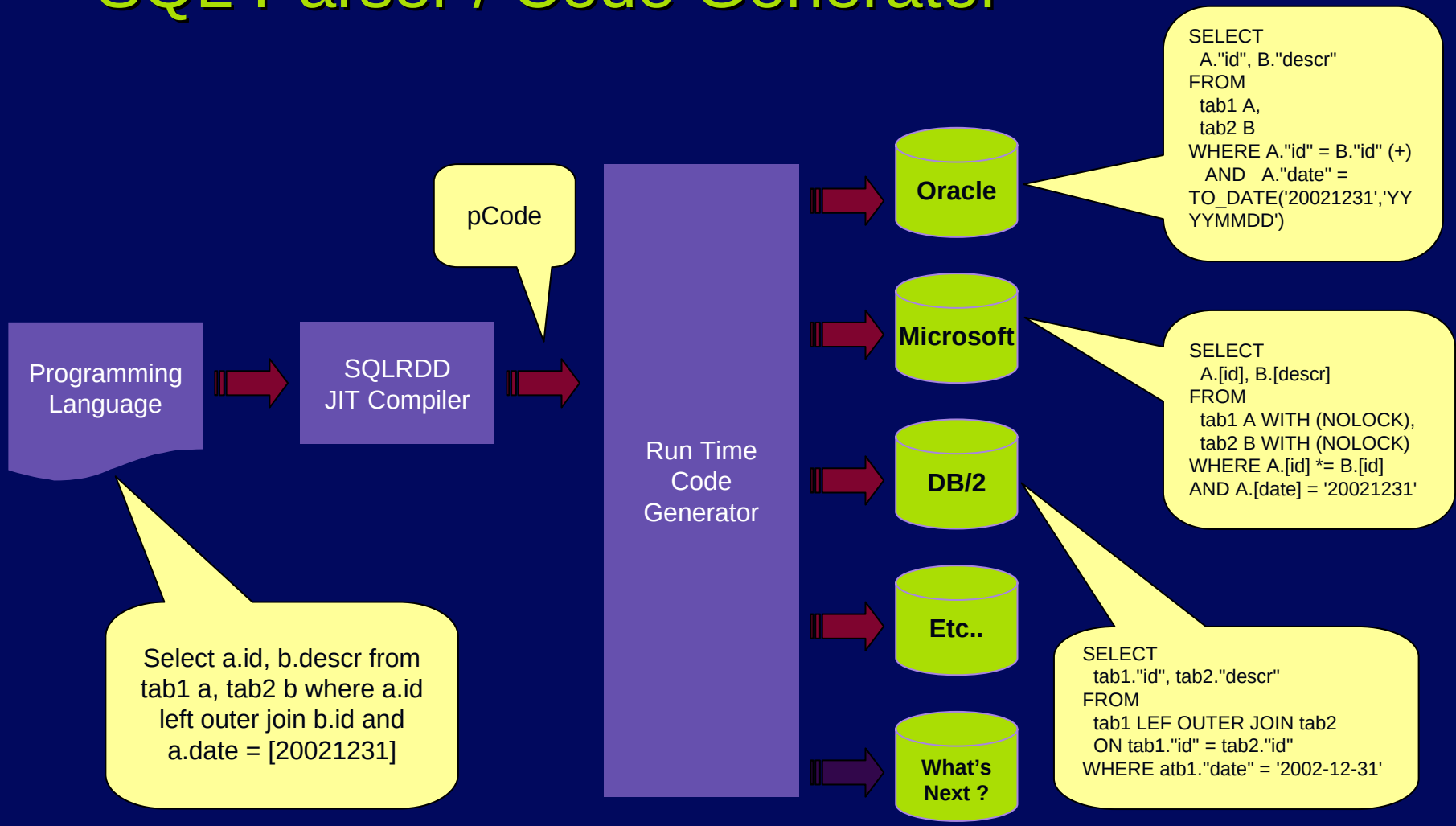
Connection Classes

- Conjunto de classes que provém o acesso aos bancos de dados diversos
- Manipulam diretamente os recod sets
- Provê acesso direto aos bancos de dados, porém com SQL nativo de cada um (não portável)

SQL Parser

- Provê uma linguagem SQL natural, independente do banco de dados
- Compila o SQL natural gerando um pCode
- Gera o código SQL específico de cada banco a partir do pCode
- Processa apenas DML (SELECT, INSERT, UPDATE, DELETE)

SQL Parser / Code Generator



Diferenciais

- Único produto do mercado que proporcional real portabilidade entre os bancos de dados diversos
- Gera aplicativos royalty free
- Não necessita de aplicativos Server ou Middleware
- Pouquíssimas mudanças no código fonte
- Dispões da maior gama de bancos de dados suportados do mercado
- Utiliza tipos de dados e índices nativos
- Pode compartilhar as bases de dados com outros ambientes e linguagens

XHarbour
eXtended xBase Compiler

Metodologia de Migração

1. Migração Instantânea
2. Ganhando Performance
3. Sintonia Fina (opcional)

Passo 01: Migração Instantânea

- Migrar de Clipper para xHarbour, ainda utilizando DBF
- Usar o dbf2sql para migrar os DBFs para o banco de dados escolhido
- Adicionar conexão ao database em seu programa principal
- Alterar abertura de tabelas para “VIA SQLRDD” ou alterar o RDD Default - RDDSetDefault(“SQLRDD”)
- Altera-se o file() por sr_file() onde necessário
- Coloca-se controle transacional em pontos estratégicos
- Configuração básica do Banco de Dados para o cenário de operação

Resultados do passo 01

- Telas e consultas ficam com performance aceitável
- Processamentos ficam aceitáveis com alguns "pontos localizados" de lentidão
- Relatórios ficam na maioria bastante lentos
- Melhora a confiabilidade pois acaba-se com a corrupção de arquivos, indices e transações críticas incompletas
- Com isto, já temos o aplicativo pronto para ser distribuído para os usuários que se encontram em estado de pânico absoluto
- Prazo estimado de 1 a 5 dias

Passo 02: Ganhando performance

- Muda-se os relatorios principais para queries SQL (ainda portáveis com o Parser do SQLRDD)
- Ajusta-se a abertura das tabelas para usar SET AUTOPEN ON
- Austa-se pontos do processamento onde trocamos os velhos:
  - Seek/DoWhile<condicao>/skip/EndDo por um bom UPDATE ... SET .. = .. WHERE <condicao>
  - Totalizações por queries com SELECT ... WHERE <condicao>

Resultados do passo 02

- Sistema com performance aceitável em todo o seu contexto e melhor do que DBF em diversos pontos
- Aplicativo pronto para ser distribuído para os usuários em geral
- Prazo Estimado: De 1 semana a 3 meses

Passo 03: Sintonia Fina (opcional)

- Modificar partes com “não conformidades”
- Adoção de filtros SERVER SIDE
- Uso de técnicas exclusivas do SQLRDD não suportadas por outros RDDs
  - SR_SetlGoTopOnFirstInteract(.F.)
  - SR_SetGoTopOnScope(.F.)
  - Indices Sintéticos
- Adição de integridade referencial e relacional no próprio banco de dados
- Tuning do banco de dados por DBA especializado

XHarbour
eXtended xBase Compiler

Resultados do passo 03

- Performance bem maior do que em DBF
- Sistema pronto para qualquer terreno
- Plataforma incontestável para a maioria dos DBAs