# Roadmap SQL – CNPJ Alfanumérico (Fase 1)

**Contexto:** Sistema legado C# WinForms + SQL Server (Mobios CCRP). Preparação para adoção de **CNPJ alfanumérico** a partir de 2026.

**Escopo desta fase:** Somente **Banco de Dados (SQL Server)**. App/API serão tratados depois.

## 1) Objetivo

Adequar o **banco** para aceitar e validar CNPJ no novo formato **[A-Z0-9]{12}[0-9]{2}** (14 posições fixas), mantendo o cálculo de DV em **módulo 11** e usando **(ASCII - 48)** para letras, **sem alterar PKs** hoje não utilizadas.

## 2) Premissas e Decisões

* Não há dados existentes nas colunas CNPJ (Total=0) → nenhuma conversão de dados agora.
* **Não mexer** nas PKs TB\_SB\_CREDITO.CNPJ e TB\_SB\_LOGS.CNPJ nesta fase (sem uso, sem FKs).
* Foco em: **tipos de colunas em uso**, **funções de validação**, **procedures/views**.
* **Retrocompatibilidade**: aceitar CNPJ numérico antigo (tudo dígito) e o novo alfanumérico.

## 3) Entregáveis

1. Inventário SQL de colunas, índices e objetos que usam CNPJ.
2. Scripts DDL para padronizar tipos de colunas de CNPJ (quando necessárias) para VARCHAR(14) (ou NVARCHAR(14)).
3. Nova função de validação **dbo.fncValida\_CNPJ\_Alpha** (módulo 11 com ASCII-48).
4. Refactor das SPs/Views/Functions que manipulam CNPJ (filtros, JOINs, SQL dinâmico).
5. Massa de teste (CNPJs numéricos e alfanuméricos válidos/ inválidos) + casos de teste.
6. Checklist Go/No-Go, Plano de Rollback e Métricas de sucesso.

## 4) Cronograma sugerido (4 semanas)

* **S1 – Descoberta & Planejamento**
  + Rodar inventários e mapear impacto.
  + Aprovar escopo de alterações.
* **S2 – Implementação Técnica (Funções + DDL mínima)**
  + Criar fncValida\_CNPJ\_Alpha.
  + Ajustar colunas **não-PK** que ainda estão numéricas.
* **S3 – Refactor SPs/Views + Testes**
  + Trocar validações, retirar dependências de ISNUMERIC, CAST numérico, etc.
  + Executar suíte de testes com massa alfanumérica.
* **S4 – Homologação & Deploy**
  + Homologação por área usuária.
  + Deploy em produção com plano de rollback.

## 5) Passo a passo detalhado

### 5.1 Inventário (descoberta)

**a) Colunas suspeitas (nome + tipo)**

-- Colunas com nomes típicos de CNPJ/CGC/RAIZ e tipos numéricos ou de texto longo  
SELECT  
 t.name AS Tabela,  
 c.name AS Coluna,  
 ty.name AS Tipo,  
 c.max\_length,  
 c.is\_nullable  
FROM sys.columns c  
JOIN sys.tables t ON t.object\_id = c.object\_id  
JOIN sys.types ty ON ty.user\_type\_id = c.user\_type\_id  
WHERE c.name LIKE '%CNPJ%'  
 OR c.name LIKE '%CGC%'  
 OR c.name LIKE '%RAIZ%'  
ORDER BY t.name, c.column\_id;

**b) Objetos que referenciam CNPJ**

-- Localiza SPs/Views/Functions que usam 'CNPJ' em qualquer parte do texto  
SELECT  
 o.type\_desc,  
 SCHEMA\_NAME(o.schema\_id) AS [schema],  
 o.name,  
 m.definition  
FROM sys.sql\_modules m  
JOIN sys.objects o ON o.object\_id = m.object\_id  
WHERE m.definition LIKE '%CNPJ%'  
ORDER BY o.type\_desc, [schema], o.name;

Resultado esperado: lista para revisão e classificação (alterar / só revisar / ignorar).

### 5.2 Padronização de tipos de coluna (somente se necessário agora)

**Regra:** Colunas **em uso** que armazenam CNPJ e ainda são numéricas → alterar para VARCHAR(14) (ou NVARCHAR(14) se houver internacionalização). **Não alterar PKs sem uso nesta fase**.

**Script-modelo (ajuste tabela/coluna conforme inventário):**

-- Exemplo: alterar tipo de coluna numérica para VARCHAR(14)  
ALTER TABLE dbo.SUA\_TABELA  
 ALTER COLUMN CNPJ VARCHAR(14) NOT NULL; -- use NULL/NOT NULL conforme o atual  
  
-- Opcional: Índice para pesquisas por igualdade  
CREATE INDEX IX\_SUA\_TABELA\_CNPJ ON dbo.SUA\_TABELA(CNPJ);

Observação: se a coluna possuir **CHECK** antigo (só dígitos), revisar para aceitar A–Z0–9 (ver 5.3.3).

### 5.3 Nova validação de CNPJ (módulo 11, ASCII-48)

#### 5.3.1 Função utilitária de normalização

CREATE OR ALTER FUNCTION dbo.fncOnlyAZ09\_Upper (@s VARCHAR(200))  
RETURNS VARCHAR(200)  
AS  
BEGIN  
 -- Mantém apenas A-Z e 0-9, e upper-case  
 DECLARE @r VARCHAR(200) = '';  
 DECLARE @i INT = 1, @ch NCHAR(1);  
 SET @s = UPPER(@s);  
 WHILE @i <= LEN(@s)  
 BEGIN  
 SET @ch = SUBSTRING(@s, @i, 1);  
 IF (@ch LIKE '[A-Z]' OR @ch LIKE '[0-9]') SET @r += @ch;  
 SET @i += 1;  
 END  
 RETURN @r;  
END;

#### 5.3.2 Função de validação (14 posições, DV módulo 11 com ASCII-48)

Mantém a lógica clássica de pesos do CNPJ para os dois dígitos verificadores; para letras, o valor é (ASCII - 48).

CREATE OR ALTER FUNCTION dbo.fncValida\_CNPJ\_Alpha (@cnpj\_in VARCHAR(200))  
RETURNS BIT  
AS  
BEGIN  
 DECLARE @cnpj VARCHAR(14) = dbo.fncOnlyAZ09\_Upper(@cnpj\_in);  
 IF LEN(@cnpj) <> 14 RETURN 0;  
  
 -- Padrão: 12 primeiras [A-Z0-9], 2 últimas [0-9]  
 IF @cnpj NOT LIKE REPLICATE('[A-Z0-9]',12) + REPLICATE('[0-9]',2) COLLATE Latin1\_General\_BIN2 RETURN 0;  
  
 -- Converte char -> valor (ASCII - 48)  
 DECLARE @i INT, @val INT;  
 DECLARE @sum1 INT = 0, @sum2 INT = 0;  
 DECLARE @pesos1 TABLE (pos INT, p INT);  
 DECLARE @pesos2 TABLE (pos INT, p INT);  
  
 -- Pesos do CNPJ clássico  
 INSERT INTO @pesos1 (pos,p) VALUES  
 (1,5),(2,4),(3,3),(4,2),(5,9),(6,8),(7,7),(8,6),(9,5),(10,4),(11,3),(12,2);  
 INSERT INTO @pesos2 (pos,p) VALUES  
 (1,6),(2,5),(3,4),(4,3),(5,2),(6,9),(7,8),(8,7),(9,6),(10,5),(11,4),(12,3),(13,2);  
  
 -- Primeiro DV sobre as 12 primeiras  
 SET @i = 1;  
 WHILE @i <= 12  
 BEGIN  
 SET @val = ASCII(SUBSTRING(@cnpj, @i, 1)) - 48; -- dígitos 0-9 => 0..9; letras A-Z => 17..42  
 SELECT @sum1 += @val \* p FROM @pesos1 WHERE pos = @i;  
 SET @i += 1;  
 END  
 DECLARE @dv1 INT = (@sum1 % 11);  
 SET @dv1 = CASE WHEN @dv1 < 2 THEN 0 ELSE 11 - @dv1 END;  
  
 -- Segundo DV: usa as 12 + dv1  
 SET @i = 1; SET @sum2 = 0;  
 WHILE @i <= 12  
 BEGIN  
 SET @val = ASCII(SUBSTRING(@cnpj, @i, 1)) - 48;  
 SELECT @sum2 += @val \* p FROM @pesos2 WHERE pos = @i;  
 SET @i += 1;  
 END  
 -- Posição 13 (dv1) com peso 2 (já previsto na tabela @pesos2 como pos=13)  
 SELECT @sum2 += @dv1 \* p FROM @pesos2 WHERE pos = 13;  
  
 DECLARE @dv2 INT = (@sum2 % 11);  
 SET @dv2 = CASE WHEN @dv2 < 2 THEN 0 ELSE 11 - @dv2 END;  
  
 -- Compara com os 2 últimos caracteres (devem ser dígitos)  
 IF (SUBSTRING(@cnpj, 13, 1) = CONVERT(VARCHAR(1), @dv1)  
 AND SUBSTRING(@cnpj, 14, 1) = CONVERT(VARCHAR(1), @dv2))  
 RETURN 1;  
  
 RETURN 0;  
END;

#### 5.3.3 CHECK Constraint (opcional, onde fizer sentido)

-- Garante padrão do novo CNPJ (sem validar DV). Útil para colunas de entrada.  
ALTER TABLE dbo.SUA\_TABELA WITH NOCHECK  
ADD CONSTRAINT CK\_SUA\_TABELA\_CNPJ\_FORMATO  
CHECK (CNPJ COLLATE Latin1\_General\_BIN2 LIKE REPLICATE('[A-Z0-9]',12) + '[0-9][0-9]');

### 5.4 Refactor de Procedures / Views

**Guidelines:** - **Remover** dependência de ISNUMERIC, TRY\_CONVERT(INT, CNPJ), CAST numérico, etc. - Usar sempre **comparações de string** (=/IN/JOIN) com collation case-insensitive ou UPPER() padronizado. - Onde houver validação, trocar para dbo.fncValida\_CNPJ\_Alpha(@cnpj). - Em **SQL dinâmico**, parametrizar e aplicar UPPER() + limpeza se necessário.

**Snippet de busca e substituição base:**

-- Exemplo: validar parâmetro de entrada  
IF dbo.fncValida\_CNPJ\_Alpha(@CNPJ) = 0  
BEGIN  
 RAISERROR('CNPJ inválido (formato ou DV)', 16, 1);  
 RETURN;  
END  
  
-- Exemplo: normalização em JOINs/filtros  
... WHERE UPPER(t.CNPJ) = UPPER(@CNPJ)

### 5.5 Testes

**Massa sugerida:** - 10 valores **válidos** numéricos (CNPJ clássico) e 10 **inválidos**. - 10 valores **válidos** alfanuméricos (fabricados com DV correto) e 20 **inválidos** (tamanho errado, letras nas duas últimas posições, DV errado, caracteres fora de [A-Z0-9]).

**Casos de teste:** 1. fncValida\_CNPJ\_Alpha retorna 1 para válidos e 0 para inválidos. 2. INSERT/UPDATE em tabelas com CHECK (quando houver) aceita válidos e rejeita inválidos. 3. Procedures com filtros por CNPJ: retornam linhas corretas com CNPJ alfa. 4. SQL dinâmico: sem erros de conversão; filtros funcionam com letras.

**Automação mínima:** criar uma #t\_casos e assertivas simples:

CREATE TABLE #t\_casos (cnpj VARCHAR(50), esperado BIT);  
INSERT INTO #t\_casos VALUES  
('A1B2C3D4E5F601', 1), -- exemplo fictício (ajuste DV conforme gerador)  
('00000000000000', 0),  
('ABCDEF123456AA', 0),  
('12AB34CD56EF78', 0);  
  
SELECT cnpj, esperado, dbo.fncValida\_CNPJ\_Alpha(cnpj) AS obtido  
FROM #t\_casos;

## 6) Critérios de aceite (Go/No-Go)

* Todos os objetos mapeados e revisados.
* Nenhuma SP/View usando ISNUMERIC/conversões numéricas para CNPJ.
* Função fncValida\_CNPJ\_Alpha implantada e coberta por testes.
* Massa de testes executada sem regressões.
* Documentação atualizada (inventário e decisões).

## 7) Plano de Deploy & Rollback

**Deploy:** 1. Criar função(ões) novas (fncValida\_CNPJ\_Alpha, utilitários) **idempotentes** (CREATE OR ALTER). 2. Aplicar DDL de alteração de colunas **não-PK**. 3. Atualizar SPs/Views dependentes. 4. Executar testes pós-deploy.

**Rollback:** - Scripts inversos: ALTER COLUMN de volta ao tipo anterior; DROP FUNCTION/DROP CONSTRAINT criados nesta fase. - Snapshot/backup do schema antes do deploy.

## 8) Riscos & Mitigações

* **Risco:** Alguma SP oculta (SQL dinâmico construído no app) não mapeada → **Mitigação:** rastrear no log de consultas recentes e revisar.
* **Risco:** Colação sensível a case/acentos gerando falso negativo → **Mitigação:** padronizar UPPER() e, se necessário, aplicar collation explícito.
* **Risco:** Performance em buscas por CNPJ após virar texto → **Mitigação:** índices em colunas de CNPJ mais usadas.

## 9) Métricas de sucesso

* 0 erros de conversão/validação em logs após deploy.
* 100% das SPs/Views relevantes sem dependências numéricas para CNPJ.
* Tempo médio de execução de consultas por CNPJ sem piora perceptível (>10%).

## 10) Próximos passos (fora do escopo desta fase)

* Definir se/como migrar as **PKs** quando (e se) passarem a ser usadas.
* Ajustes no **APP/API** (máscaras, validação no front/back, DTOs, contratos, documentação).
* Gerador oficial de CNPJ alfa para massa de testes (script dedicado).