# Relatório sobre os CHECKs de integridade (Banco de Dados)

Israel Efraim Silva e Sousa – 2DB

Introdução  
  
Este relatório apresenta uma análise pormenorizada de cada restrição do tipo CHECK definida nas tabelas do esquema disponibilizado. Para cada constraint são descritos: a finalidade, o impacto sobre a integridade dos dados, potenciais limitações e sugestões de aprimoramento.  
  
O objetivo é proporcionar compreensão técnica e recomendações práticas para robustecer a modelagem do banco de dados, preservando coerência e prevenindo inserções inválidas.

## Tabela: IMOVEL — Constraint: check\_id\_imovel

SQL:

ALTER TABLE IMOVEL ADD CONSTRAINT check\_id\_imovel CHECK (id\_imovel > 1);

Objetivo:

Assegurar que o identificador do imóvel seja maior que 1 — evita valores nulos, 0 ou negativos.

Comentário e Sugestões:

Observação: não impede duplicidade se a PK não for bem definida; considera-se também garantir auto-incremento se aplicável.

## Tabela: IMOVEL — Constraint: check\_endereco

SQL:

ALTER TABLE IMOVEL ADD CONSTRAINT check\_endereco CHECK (UPPER(endereco) REGEXP '/ (AC|AL|AP|AM|BA|CE|DF|ES|GO|MA|MT|MS|MG|PA|PB|PR|PE|PI|RJ|RN|RS|RO|RR|SC|SP|SE|TO)$');

Objetivo:

Validar que o texto do endereço termine com a sigla de estado brasileiro (UF), em caixa alta.

Comentário e Sugestões:

Observação: a expressão regular e a função UPPER assumem formato específico; pode falhar com variações de formatação ou ausência de espaço antes da sigla. Recomenda-se normalizar dados no momento da inserção ou usar colunas separadas (cidade, uf).

## Tabela: IMOBILIARIA — Constraint: check\_cnpj

SQL:

ALTER TABLE IMOBILIARIA ADD CONSTRAINT check\_cnpj CHECK (cnpj > 1);

Objetivo:

Impor que o CNPJ seja um número positivo superior a 1, evitando 0 ou valores negativos.

Comentário e Sugestões:

Observação: não valida estrutura do CNPJ (14 dígitos nem dígito verificador). Para validação robusta, implementar check de formato ou trigger que chame função de validação.

## Tabela: IMOBILIARIA — Constraint: check\_email

SQL:

ALTER TABLE IMOBILIARIA ADD CONSTRAINT check\_email CHECK (email LIKE '%@%');

Objetivo:

Garantir que o campo e-mail contenha pelo menos o caractere '@', evitando strings sem formato mínimo de e-mail.

Comentário e Sugestões:

Observação: teste simples; admite e-mails inválidos (ex.: 'a@b'). Para validação melhor, usar expressão regular ou lógica em aplicação.

## Tabela: REGISTRO — Constraint: check\_id\_registro

SQL:

ALTER TABLE REGISTRO ADD CONSTRAINT check\_id\_registro CHECK (id\_registro > 1);

Objetivo:

Evitar identificadores de registro nulos, 0 ou negativos.

Comentário e Sugestões:

Observação: semelhante ao check de id\_imovel; considerar auto-incremento para PK.

## Tabela: REGISTRO — Constraint: check\_data\_registro

SQL:

ALTER TABLE REGISTRO ADD CONSTRAINT check\_data\_registro CHECK (data\_registro >= '2025-01-01');

Objetivo:

Forçar que a data do registro seja a partir de 1º de janeiro de 2025.

Comentário e Sugestões:

Observação: esta restrição é temporal e pode tornar-se obsoleta; confirmar se intenção era aceitar datas anteriores. Se a finalidade for impedir datas futuras, utilizar data\_registro <= CURDATE().

## Tabela: ANUNCIA — Constraint: check\_data\_anuncio

SQL:

ALTER TABLE ANUNCIA ADD CONSTRAINT check\_data\_anuncio CHECK (data\_anuncio >= '2021-01-01');

Objetivo:

Permitir apenas anúncios com data a partir de 1º de janeiro de 2021 — evita anúncios demasiadamente antigos.

Comentário e Sugestões:

Observação: data fixa também é sensível ao tempo; avaliar se a regra deve ser relativa (ex.: não anterior a X anos) ou proibir datas futuras.

## Tabela: ANUNCIA — Constraint: check\_id\_imovel

SQL:

ALTER TABLE ANUNCIA ADD CONSTRAINT check\_id\_imovel CHECK (id\_imovel > 0);

Objetivo:

Assegurar que o id\_imovel referenciado seja positivo.

Comentário e Sugestões:

Observação: a integridade referencial já é guardada pela FK; este check adiciona redundância útil para entradas locais.

## Tabela: SITE — Constraint: check\_id\_positivo

SQL:

ALTER TABLE SITE ADD CONSTRAINT check\_id\_positivo CHECK (id > 0);

Objetivo:

Impor que o identificador 'id' associado ao site seja positivo.

Comentário e Sugestões:

Observação: se 'id' for usado como FK em outras tabelas, considerar torná-lo PK e aplicar auto-incremento para maior segurança.

## Tabela: SITE — Constraint: check\_email\_valido

SQL:

ALTER TABLE SITE ADD CONSTRAINT check\_email\_valido CHECK (emailcontt LIKE '%@%');

Objetivo:

Verificar presença de '@' no contato de e-mail do site.

Comentário e Sugestões:

Observação: validação mínima; recomenda-se regex mais rigorosa ou validação na aplicação.

## Tabela: ENTREGA — Constraint: check\_data\_entrega

SQL:

ALTER TABLE ENTREGA ADD CONSTRAINT check\_data\_entrega CHECK (data\_entrega >= '2021-01-01');

Objetivo:

Permitir apenas entregas cuja data seja a partir de 1º de janeiro de 2021.

Comentário e Sugestões:

Observação: mesma observação sobre datas fixas — considerar restrições relativas ou impedir datas futuras conforme requisito.

## Tabela: ENTREGA — Constraint: check\_cnpj\_positivo

SQL:

ALTER TABLE ENTREGA ADD CONSTRAINT check\_cnpj\_positivo CHECK (cnpj > 0);

Objetivo:

Garantir que o CNPJ associado à entrega seja positivo.

Comentário e Sugestões:

Observação: redundante com FK; útil como validação adicional no momento da inserção.

## Tabela: CLIENTE — Constraint: check\_email\_valido\_cliente

SQL:

ALTER TABLE CLIENTE ADD CONSTRAINT check\_email\_valido\_cliente CHECK (email LIKE '%@%');

Objetivo:

Exigir o caractere '@' no e-mail do cliente.

Comentário e Sugestões:

Observação: validação mínima; para segurança, adicionar verificação de domínio ou regex.

## Tabela: CLIENTE — Constraint: check\_senha\_minima

SQL:

ALTER TABLE CLIENTE ADD CONSTRAINT check\_senha\_minima CHECK (CHAR\_LENGTH(senha) >= 6);

Objetivo:

Impor comprimento mínimo de 6 caracteres para a senha armazenada.

Comentário e Sugestões:

Observação: não garante complexidade (maiúsculas, números, símbolos). Além disso, por segurança, senhas não devem ser armazenadas em texto puro — usar hashing e, se possível, regras de complexidade na camada da aplicação.

## Tabela: PROPOSTA — Constraint: check\_valor\_positivo

SQL:

ALTER TABLE PROPOSTA ADD CONSTRAINT check\_valor\_positivo CHECK (valor > 0);

Objetivo:

Assegurar que o valor monetário de uma proposta seja estritamente positivo.

Comentário e Sugestões:

Observação: apropriado; confirmar também limites superiores ou consistência com o valor do imóvel em caso de vínculo lógico.

## Tabela: PROPOSTA — Constraint: chk\_resgprost\_valido

SQL:

ALTER TABLE PROPOSTA ADD CONSTRAINT chk\_resgprost\_valido CHECK (resgprost IN ('Aceita', 'Recusada', 'Em Análise'));

Objetivo:

Restringir o status da proposta a três valores permitidos, padronizando registros.

Comentário e Sugestões:

Observação: ótima prática. Se houver possibilidade de internacionalização, avaliar uso de códigos (ENUM ou tabela de lookup) para maior flexibilidade.

## Tabela: FAZ — Constraint: check\_cpf\_nao\_vazio

SQL:

ALTER TABLE FAZ ADD CONSTRAINT check\_cpf\_nao\_vazio CHECK (cpf <> '');

Objetivo:

Impedir que o campo cpf da relação fique vazio.

Comentário e Sugestões:

Observação: redundante com a FK para CLIENTE, mas adiciona validação imediata. Para robustez, validar formato do CPF (11 dígitos) em vez de apenas não vazio.

## Tabela: FAZ — Constraint: check\_id\_proposta\_positivo

SQL:

ALTER TABLE FAZ ADD CONSTRAINT check\_id\_proposta\_positivo CHECK (id\_proposta > 0);

Objetivo:

Garantir que o identificador da proposta na relação seja positivo.

Comentário e Sugestões:

Observação: integridade referencial pela FK complementa este check; manutenção é simples e válida.

## Tabela: COMPRA — Constraint: check\_valpag\_positivo

SQL:

ALTER TABLE COMPRA ADD CONSTRAINT check\_valpag\_positivo CHECK (valpag > 0);

Objetivo:

Assegurar que o valor pago na compra seja positivo.

Comentário e Sugestões:

Observação: pode-se também verificar coerência entre valpag e valor da proposta vinculada.

## Tabela: COMPRA — Constraint: check\_data\_compra

SQL:

ALTER TABLE COMPRA ADD CONSTRAINT check\_data\_compra CHECK (data\_compra <= CURDATE());

Objetivo:

Impedir inserção de datas de compra futuras.

Comentário e Sugestões:

Observação: regra prática e sensata; considerar timezone e consistência com logs de transação.

Conclusão  
  
As constraints do tipo CHECK descritas proporcionam um primeiro nível de validação diretamente no banco de dados, mitigando inserções indevidas e fortalecendo a integridade dos dados. Porém, muitas das verificações apresentadas são de caráter sintático ou estacionário (ex.: presença de '@' em e-mails ou datas a partir de uma data fixa). Recomenda-se complementar as restrições com validações mais robustas na camada de aplicação, adoção de triggers ou funções de validação para formatos complexos (CPF/CNPJ), e uso de tipos e colunas normalizadas (UF, cidade, campos monetários). Finalmente, avaliar a necessidade de tornar algumas regras relativas ao tempo (datas) mais flexíveis para evitar obsolescência futura.