Documento – Views de Manutenção do Banco Yoov (`yoov_db`)

1. Visão Geral do que foi feito

No banco `yoov_db` foram criadas três views voltadas para **organização e manutenção**:

1. `vw_tabelas_por_modulo` – catálogo organizado das tabelas/views por módulo.
2. `vw_relacoes_modulos` – mapa de relacionamentos (FKs) entre tabelas e módulos.
3. `vw_inconsistencias_geral` – painel único de inconsistências de dados por módulo.

Além disso, foram adicionados **comentários nas tabelas/views** deixando claro a qual **módulo** cada uma pertence. Essas views usam esses comentários por trás.

---

2. `vw_tabelas_por_modulo`

2.1. O que ela faz

É um **catálogo técnico do banco**, mostrando:

- Nome da tabela/view (`tabela`)
- Comentário/módulo (`modulo`)
- Tipo do objeto (`tipo_objeto`: TABELA ou VIEW)
- Quantidade de colunas
- Triggers ligadas àquela tabela
- Quantidade de FKs
- Dono do objeto
- Tamanho em bytes / tamanho legível

Ela ajuda a responder perguntas do tipo:

- "Quais objetos existem no módulo VENDAS?"
- "Essa tabela tem trigger? Tem FK?"
- "Quais tabelas/views estão sem módulo definido?"

2.2. Uso básico

Ver tudo organizado por módulo:

```
SELECT * FROM vw_tabelas_por_modulo;
(retorna todas as tabelas e views do schema public, agrupadas por comentário/
módulo)
```

2.3. Exemplos de uso com filtro

1) Somente o módulo VENDAS

```
SELECT *
FROM vw_tabelas_por_modulo
WHERE modulo LIKE 'MÓDULO VENDAS%';
(filtra apenas as tabelas/views cujo comentário começa com "MÓDULO VENDAS")
```

2) Ver apenas views

```
SELECT *
FROM vw_tabelas_por_modulo
WHERE tipo_objeto = 'VIEW';
(filtra apenas objetos do tipo VIEW; troque 'VIEW' por 'TABELA' se quiser só
tabelas)
```

3) Procurar uma tabela específica

```
SELECT *
FROM vw_tabelas_por_modulo
WHERE tabela = 'vendas';
(trocar 'vendas' pelo nome da tabela/view que você quer inspecionar)
```

4) Achar tabelas sem módulo definido (para ajuste de comentários)

```
SELECT *
FROM vw_tabelas_por_modulo
WHERE modulo = 'SEM MÓDULO DEFINIDO';
(lista objetos que não têm comentário ou ainda não foram classificados por
módulo)
```

---

3. `vw_relacoes_modulos`

3.1. O que ela faz

Essa view é o **mapa de relacionamentos (FKs)** entre tabelas e módulos.
Para cada **chave estrangeira** do banco, ela mostra:

- `tabela_origem`, `modulo_origem`, `coluna_origem`
- `tabela_destino`, `modulo_destino`, `coluna_destino`
- Nome da constraint (`nome_constraint`)
- Definição completa da constraint (`definicao_constraint`)

Ela responde perguntas como:

- "Quais tabelas apontam para `estoque_produtos`?"
- "Quais relações cruzam módulos diferentes (ex.: VENDAS x ESTOQUE)?"

- “Se eu mexer na tabela X, quem quebra junto?”

3.2. Uso básico

Ver todas as FKs do banco (entre módulos):

```
SELECT * FROM vw_relacoes_modulos;
(retorna todas as chaves estrangeiras do schema public, com origem, destino
e módulos)
```

3.3. Exemplos de uso com filtro

1) Ver tudo que depende de `estoque_produtos`

```
SELECT *
FROM vw_relacoes_modulos
WHERE tabela_destino = 'estoque_produtos';
(trocar 'estoque_produtos' pela tabela-alvo cuja dependência você quer ver)
```

2) Ver todas as FKs que saem do módulo VENDAS

```
SELECT *
FROM vw_relacoes_modulos
WHERE modulo_origem LIKE 'MÓDULO VENDAS%';
(trocar 'MÓDULO VENDAS%' por outro módulo, ex.: 'MÓDULO ESTOQUE%', 'MÓDULO
AGENDA%')
```

3) Ver apenas relações que cruzam módulos diferentes

```
SELECT *
FROM vw_relacoes_modulos
WHERE modulo_origem <> modulo_destino;
(mostra apenas as FKs onde origem e destino pertencem a módulos diferentes)
```

4) Ver todas as FKs de uma tabela específica

```
SELECT *
FROM vw_relacoes_modulos
WHERE tabela_origem = 'vendas';
(trocar 'vendas' pela tabela de origem que você quer analisar)
```

---

4. `vw_inconsistencias_geral`

4.1. O que ela faz

É uma **view central de "saúde dos dados"**.
Ela junta, em uma só visão, inconsistências detectadas em:

- MÓDULO VENDAS
- `VALOR_LIQUIDO_NEGATIVO`
- `DESCONTO_NEGATIVO`
- `DESCONTO_MAIOR_QUE_VALOR_BRUTO`
- `CALCULO_INCONSISTENTE`
- `VENDA_SEM_ITENS`

- MÓDULO ESTOQUE

- `PRODUTO_ESTOQUE_NEGATIVO` (tabela `estoque_produtos`)
- `MOVIMENTACAO_GEROU_ESTOQUE_NEGATIVO` (tabela `estoque_movimentacoes`)

- MÓDULO FINANCEIRO

- `VALOR_NAO_POSITIVO`
- `PAGAMENTO_ANTES_DO_LANCAMENTO`
- `VENCIMENTO_ANTES_DO_LANCAMENTO`

- MÓDULO AGENDA

- `FIM_ANTES_DO_INICIO`

Cada linha traz:

- `modulo`
- `tabela`
- `id_registro` (PK do registro problemático)
- `id_unidade`
- `tipo_inconsistencia` (código)
- `mensagem` (texto explicando o problema)

Ela é perfeita para:

- rotina de checagem diária/semanal da base
- painel de "saúde da unidade"
- base pra IA sugerir correções

4.2. Uso básico

Ver todas as inconsistências de todos os módulos:

```
SELECT * FROM vw_inconsistencias_geral;
(lista todas as inconsistências conhecidas, de todos os módulos e tabelas)
```

4.3. Exemplos de uso com filtro

1) Ver tudo que está errado no módulo ESTOQUE

```
SELECT *
FROM vw_inconsistencias_geral
WHERE modulo = 'MÓDULO ESTOQUE';
(filtra apenas inconsistências cujo campo "modulo" é exatamente 'MÓDULO
ESTOQUE')
```

2) Ver apenas inconsistências de VENDAS

```
SELECT *
FROM vw_inconsistencias_geral
WHERE modulo = 'MÓDULO VENDAS';
(trocar 'MÓDULO VENDAS' pelo módulo desejado, ex.: 'MÓDULO FINANCEIRO')
```

3) Ver só um tipo específico de problema

```
SELECT *
FROM vw_inconsistencias_geral
WHERE tipo_inconsistencia = 'VENDA_SEM_ITENS';
(trocar 'VENDA_SEM_ITENS' pelo tipo que deseja analisar, ex.:
'PRODUTO_ESTOQUE_NEGATIVO')
```

4) Ver inconsistências de uma unidade específica

```
SELECT *
FROM vw_inconsistencias_geral
WHERE id_unidade = '11111111-1111-1111-1111-111111111111';
(trocar o UUID pela unidade que você quer inspecionar)
```

5) Ver um "resumo" por módulo (quantidade de problemas)

```
SELECT modulo, COUNT(*) AS qtd_inconsistencias
FROM vw_inconsistencias_geral
GROUP BY modulo
ORDER BY qtd_inconsistencias DESC;
(agrupa por módulo e mostra quantas inconsistências cada um tem)
```