



Relatório Completo do Banco de Dados PostgreSQL Neon

Data da Análise: 28 de Outubro de 2025

Banco de Dados: neondb

Status da Conexão:  **SUCESSO**



1. Informações de Conexão

Detalhes da Conexão

- **Host:** ep-weathered-base-acxmst6l-pooler.sa-east-1.aws.neon.tech
- **Banco de Dados:** neondb
- **Usuário:** neondb_owner
- **Região:** sa-east-1 (São Paulo, AWS)
- **Tipo de Conexão:** Pooler (Connection Pooling)
- **SSL:** Obrigatório (sslmode=require)
- **Channel Binding:** Obrigatório

Status da Conexão

 **Conexão estabelecida com sucesso!**



2. Informações Gerais do Banco

Versão do PostgreSQL

PostgreSQL 17.5 (6bc9ef8) on x86_64-pc-linux-gnu
Compilado com: gcc (Debian 12.2.0-14+deb12u1) 12.2.0, 64-bit

Estatísticas do Banco

Métrica	Valor
Nome do Banco	neondb
Tamanho Total	7424 KB (~7.2 MB)
Usuário Atual	neondb_owner
Conexões Ativas	2
Transações Commitadas	390
Transações com Rollback	17
Blocos Lidos do Disco	2,646
Blocos Lidos do Cache	48,693
Tuplas Retornadas	103,161
Tuplas Buscadas	22,714
Tuplas Inseridas	64
Tuplas Atualizadas	16
Tuplas Deletadas	0

3. Schemas Disponíveis

Schema	Owner	Descrição
public	pg_database_owner	Schema padrão (vazio)
neon_auth	neondb_owner	Schema de autenticação Neon
information_schema	cloud_admin	Schema de metadados
pg_catalog	cloud_admin	Catálogo do sistema
pg_toast	cloud_admin	Sistema TOAST



4. Tabelas Existentes

4.1 Tabela: `neon_auth.users_sync`

Descrição: Tabela de sincronização de usuários do sistema de autenticação Neon.

Estrutura da Tabela

Coluna	Tipo	Nullable	Descrição
raw_json	jsonb	NOT NULL	Dados brutos do usuário em JSON
id	text	NOT NULL	ID do usuário (gerado automaticamente)
name	text	NULL	Nome de exibição do usuário
email	text	NULL	Email principal do usuário
created_at	timestamp with time zone	NULL	Data de criação da conta
updated_at	timestamp with time zone	NULL	Data da última atualização
deleted_at	timestamp with time zone	NULL	Data de exclusão (soft delete)

Colunas Geradas Automaticamente

- **id** : Extraído de `raw_json->>'id'`
- **name** : Extraído de `raw_json->>'display_name'`
- **email** : Extraído de `raw_json->>'primary_email'`
- **created_at** : Convertido de `raw_json->>'signed_up_at_millis'`

Índices

1. **users_sync_pkey** (PRIMARY KEY)
 - Tipo: btree
 - Coluna: `id`
2. **users_sync_deleted_at_idx**
 - Tipo: btree
 - Coluna: `deleted_at`
 - Propósito: Otimizar consultas de soft delete

Estatísticas

- **Total de Registros:** 1
- **Tamanho da Tabela:** 48 KB
- **Owner:** `neondb_owner`

Permissões do Usuário `neondb_owner`

- ✓ SELECT
- ✓ INSERT
- ✓ UPDATE
- ✓ DELETE
- ✓ TRUNCATE
- ✓ REFERENCES
- ✓ TRIGGER



5. Extensões Instaladas

Extensão	Versão	Schema	Descrição
plpgsql	1.0	pg_catalog	Linguagem procedur-al PL/pgSQL

Extensões Recomendadas para o Projeto TimmingLoveU

Com base no schema do Prisma encontrado no projeto, recomendo instalar as seguintes extensões:

```
-- Para geração de UUIDs (usado no Prisma com cuid())
CREATE EXTENSION IF NOT EXISTS "uuid-oss";

-- Para busca full-text em português
CREATE EXTENSION IF NOT EXISTS "unaccent";

-- Para funções de criptografia (útil para senhas)
CREATE EXTENSION IF NOT EXISTS "pgcrypto";
```



6. Views e Sequences

Views

✗ **Nenhuma view encontrada** no schema `neon_auth` ou `public`.

Sequences

✗ **Nenhuma sequence encontrada** no schema `neon_auth` ou `public`.



7. Análise do Schema do Projeto TimmingLoveU

7.1 Modelos Identificados no Prisma Schema

O projeto possui **9 modelos principais**:

1. Autenticação (NextAuth.js)

- `Account` - Contas de provedores OAuth

- `Session` - Sessões de usuário
- `User` - Usuários do sistema
- `VerificationToken` - Tokens de verificação

2. Funcionalidades do TimmingLoveU

- `CouplePage` - Páginas de casais
- `PlanoAssinatura` - Planos de assinatura
- `UserSubscription` - Assinaturas de usuários
- `Transaction` - Transações financeiras
- `Media` - Fotos e vídeos dos casais

7.2 Tabelas que Precisam Ser Criadas

⚠ **IMPORTANTE:** O banco de dados está **VAZIO** (exceto pela tabela `neon_auth.users_sync`).

Todas as tabelas do projeto precisam ser criadas através de uma migração do Prisma:

```
# No diretório do projeto TimmingLoveU
npx prisma migrate dev --name init
```

7.3 Estrutura Esperada Após Migração

Tabelas de Autenticação (4)

1. `accounts` - Contas OAuth
2. `sessions` - Sessões ativas
3. `users` - Usuários principais
4. `verificationtokens` - Tokens de verificação

Tabelas do Negócio (5)

1. `couple_pages` - Páginas de casais
2. `planos_assinatura` - Planos disponíveis
3. `user_subscriptions` - Assinaturas ativas
4. `transactions` - Histórico de pagamentos
5. `media` - Galeria de mídia



8. Segurança e Permissões

Configurações de Segurança Ativas



- ✓ SSL obrigatório (`sslmode=require`)
- ✓ Channel binding obrigatório
- ✓ Connection pooling ativo
- ✓ Usuário com permissões completas no schema `neon_auth`

Recomendações de Segurança

1. ✓ **SSL está ativo** - Conexão criptografada
2. ⚠ **Considerar criar usuários separados** para diferentes ambientes (dev, staging, prod)
3. ✓ **Connection pooling ativo** - Melhor performance
4. 📝 **Implementar backup automático** - Verificar configurações no Neon

9. Performance e Otimização

Métricas de Performance Atuais

- **Cache Hit Ratio:** 94.8% (48,693 / 51,339)
-  **Excelente!** Mais de 94% das leituras vêm do cache
- **Transações por Segundo:** ~3.25 (390 commits / 120s estimado)
- **Taxa de Rollback:** 4.4% (17 / 390)
-  **Aceitável** - Taxa baixa de rollbacks

Recomendações de Otimização

Para o Modelo `CouplePage`

```
-- Índice para busca por slug (já existe UNIQUE, mas explícito)
CREATE INDEX idx_couple_pages_slug ON couple_pages(slug_publico);

-- Índice para páginas ativas de um usuário
CREATE INDEX idx_couple_pages_user_active ON couple_pages(user_id, ativo);

-- Índice para ordenação por views
CREATE INDEX idx_couple_pages_views ON couple_pages(views DESC);
```

Para o Modelo `Media`

```
-- Índice composto para galeria de um casal
CREATE INDEX idx_media_couple_ordem ON media(couple_page_id, ordem, ativo);

-- Índice para busca por tipo de mídia
CREATE INDEX idx_media_tipo ON media(tipo, ativo);
```

Para o Modelo `UserSubscription`

```
-- Índice para assinaturas ativas
CREATE INDEX idx_subscriptions_active ON user_subscriptions(user_id, status, current_period_end);
```

10. Próximos Passos Recomendados

10.1 Configuração Inicial (URGENTE)

1. Executar Migrações do Prisma

```
bash
cd /home/ubuntu/timiming_project/TimmingLoveU
npx prisma migrate dev --name init
```

2. Verificar Variáveis de Ambiente

```
bash
# Verificar se DATABASE_URL está configurada corretamente
cat .env | grep DATABASE_URL
```

3. Gerar Cliente Prisma

```
bash
npx prisma generate
```

10.2 Seed do Banco de Dados

1. Criar Planos de Assinatura Iniciais

```
bash
npx prisma db seed
```

10.3 Validação

1. Verificar Tabelas Criadas

```
bash
npx prisma studio
# Ou via psql
psql 'postgresql://...' -c "\dt"
```

2. Testar Conexão da Aplicação

```
bash
npm run dev
# Verificar logs de conexão com o banco
```

10.4 Monitoramento

1. Configurar Monitoramento no Neon

- Acessar dashboard do Neon
- Ativar alertas de uso
- Configurar backups automáticos

2. Implementar Logging de Queries

```
typescript
// No prisma client
const prisma = new PrismaClient({
  log: ['query', 'error', 'warn'],
})
```



11. Comandos Úteis

Conectar ao Banco via psql

```
psql 'postgresql://neondb_owner:npg_0CBHJVFEpZ9L@ep-weathered-base-acxmst6l-pooler.sa-east-1.aws.neon.tech/neondb?sslmode=require&channel_binding=require'
```

Listar Todas as Tabelas

```
\dt *.*
```

Ver Estrutura de uma Tabela

```
🔍 d+ schema_name.table_name
```

Ver Tamanho das Tabelas

```
SELECT
    schemaname,
    tablename,
    pg_size_pretty(pg_total_relation_size(schemaname||'.'||tablename)) as size
FROM pg_tables
WHERE schemaname NOT IN ('pg_catalog', 'information_schema')
ORDER BY pg_total_relation_size(schemaname||'.'||tablename) DESC;
```

Verificar Conexões Ativas

```
SELECT
    datname,
    username,
    application_name,
    client_addr,
    state,
    query
FROM pg_stat_activity
WHERE datname = 'neondb';
```

! 12. Avisos e Considerações

Avisos Importantes

1. **! Banco de Dados Vazio**
 - O banco está praticamente vazio (apenas 1 tabela de sistema)
 - É necessário executar as migrações do Prisma antes de usar a aplicação
2. **! Credenciais Expostas**
 - As credenciais do banco estão visíveis neste relatório
 - Recomendo rotacionar a senha após a configuração inicial
 - Nunca commitar credenciais no Git
3. **! Connection Pooling**
 - O endpoint usa pooler (pooler.sa-east-1.aws.neon.tech)
 - Ideal para aplicações serverless
 - Limite de conexões pode variar conforme o plano Neon

Limitações do Plano Neon

Verificar no dashboard do Neon:

- Limite de armazenamento
- Limite de conexões simultâneas
- Limite de compute hours
- Política de backup

13. Suporte e Recursos

Documentação Oficial

- **Neon:** <https://neon.tech/docs>
- **Prisma:** <https://www.prisma.io/docs>
- **PostgreSQL 17:** <https://www.postgresql.org/docs/17/>

Comandos de Diagnóstico

```
# Verificar versão do Prisma
npx prisma --version

# Validar schema do Prisma
npx prisma validate

# Ver status das migrações
npx prisma migrate status

# Resetar banco (CUIDADO: apaga todos os dados)
npx prisma migrate reset
```

14. Checklist de Configuração

- ☒ [x] Conexão com o banco estabelecida
- ☒ [x] Credenciais validadas
- ☒ [x] Schema do Prisma analisado
- ☐ [] Migrações executadas
- ☐ [] Cliente Prisma gerado
- ☐ [] Seed do banco executado
- ☐ [] Planos de assinatura criados
- ☐ [] Testes de conexão da aplicação
- ☐ [] Monitoramento configurado
- ☐ [] Backups configurados

Conclusão

O banco de dados PostgreSQL Neon está **funcionando corretamente** e pronto para receber as migrações do projeto TimmingLoveU.

Status Geral:  **PRONTO PARA CONFIGURAÇÃO**

Próxima Ação Recomendada:

```
cd /home/ubuntu/timming_project/TimmingLoveU
npx prisma migrate dev --name init
```

Relatório gerado em: 28 de Outubro de 2025

Ferramenta: PostgreSQL Client 15.14

Analista: DeepAgent AI