# **Checklist atualizado: Supabase + n8n: Operações CRUD em Postgres + Integração com Agente de IA**

Este checklist foi criado para te ensinar como **conectar o Supabase (banco SQL) ao n8n**, **como realizar as 4 operações CRUD clássicas**, **como integrar o Supabase como ferramenta (Tool) em um Agente de IA** e realizarmos uma comparação do uso de banco direto vs via agente.

**1. Criando uma conta no Supabase**

1. Acesse [supabase.com](https://supabase.com/) e crie uma conta
2. Clique em **New Project**
3. Preencha os dados e aguarde a criação
4. Crie uma nova tabela chamada gastos:

CREATE TABLE gastos (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

item TEXT NOT NULL,

valor FLOAT8 NOT NULL,

fornecedor TEXT,

tipo\_despesa TEXT,

data DATE NOT NULL

);

5. Agora insira 10 registros

INSERT INTO gastos (item, valor, fornecedor, tipo\_despesa, data)

VALUES

('Pizza', 85.50, 'Dominos', 'pessoal', '2025-06-01'),

('Uber', 29.90, 'Uber Brasil', 'pessoal', '2025-06-02'),

('Notebook', 4500.00, 'Dell', 'empresarial', '2025-06-03'),

('Café', 12.00, 'Starbucks', 'pessoal', '2025-06-04'),

('Hospedagem', 350.00, 'Booking.com', 'empresarial', '2025-06-05'),

('Formação Profissional', 1999.90, 'Jornada de dados', 'educacao', '2025-06-06'),

('Internet', 120.00, 'Vivo', 'pessoal', '2025-06-07'),

('Ferramenta SaaS', 99.00, 'ZapFlow', 'empresarial', '2025-06-08'),

('Supermercado', 320.00, 'Pão de Açúcar', 'pessoal', '2025-06-09'),

('Combustível', 210.00, 'Posto Shell', 'pessoal', '2025-06-10');

**2. Configurando credenciais (Supabase e postgres)**

Podemos usar o node do postgres ou o node do supabase.

O **Postgres Node** conecta diretamente ao banco de dados PostgreSQL via SQL (usando pg no backend). O **Supabase Node** se conecta à API REST exposta pelo Supabase (usando Supabase JS Client).

O que isso muda?

O Postgres Node tem acesso direto e completo ao banco: você pode rodar qualquer SELECT, JOIN, GROUP BY, UPDATE, DELETE, INSERT, UPSERT, etc.

O Supabase Node tem limitações impostas pela API REST (por exemplo, queries complexas, agregações ou JOINs são mais difíceis).

### **O que é igual nos dois nós?**

| **Característica** | **Postgres Node** | **Supabase Node** |
| --- | --- | --- |
| Inserir dados (Create) | ✅ sim | ✅ sim |
| Atualizar dados (Update) | ✅ sim | ✅ sim |
| Deletar dados | ✅ sim | ✅ sim |
| Buscar dados | ✅ sim (via SQL) | ✅ sim (via parâmetros ou filtros) |
| Retornar dados como JSON | ✅ sim | ✅ sim |
| Suporte a credenciais | ✅ sim (host, user, pass etc.) | ✅ sim (API URL + Key) |
| Pode ser usado como Tool do Agent | ✅ sim (direto como tool) | ✅ sim (direto como tool) |

## **Cadastro de Credencial no n8n: Supabase**

### **Passo a passo:**

1. Acesse o n8n
2. Vá em **Credentials** > **New Credential**
3. Escolha o tipo **Supabase**
4. Preencha os campos:

| **Campo** | **Descrição** |
| --- | --- |
| **API URL** | A URL base do seu projeto Supabase. Ex: https://xyzcompany.supabase.co |
| **API Key** | Sua chave de API (anon ou service\_role) — encontrada no painel do Supabase |

**Cadastro de Credencial no n8n: Postgres**

### **Passo a passo:**

1. Vá em **Credentials** > **New Credential**
2. Escolha o tipo **Postgres**
3. Preencha os campos:

| **Campo** | **Descrição** |
| --- | --- |
| **Host** | Ex: db.xyzcompany.supabase.co (pegue no painel do Supabase) |
| **Port** | Geralmente 5432 |
| **User** | Nome do usuário (geralmente postgres ou definido no projeto) |
| **Password** | Senha gerada no painel Supabase |
| **Database** | Nome do banco (definido ao criar o projeto) |

**3. Criando nossos primeiros nodes**

Montar um workflow no n8n com dois caminhos paralelos:

* 🔹 Um usando o **nó Supabase** (via API REST)
* 🔹 Outro usando o **nó Postgres** (via SQL direto)

## 

### **Passo 1 – Gatilho Manual**

1. Adicione o nó **Manual Trigger**Nome: When clicking ‘Test workflow’
   * Isso permite que você execute o fluxo manualmente clicando em “Executar Fluxo”

### **Passo 2 – Configurando o nó Supabase (REST API)**

1. Adicione o node Supabase
2. Configure assim:
   * **Operation**: Get All
   * **Table ID**: gastos
   * **Filters**:
     + Field: valor
     + Condition: gt (greater than)
     + Value: 50

### **🛢️ Passo 3 – Configurando o nó Postgres (SQL direto)**

1. Adicione o nó Postgres
2. Configure assim:
   * **Operation**: Select
   * **Schema**: public
   * **Table**: gastos
   * **Where (Filter)**:
     + Column: valor
     + Condition: > (Greater than)
     + Value: 50

Resultado: executa internamente o SQL:

Além disso, há a operação **Execute Query**, que permite rodar uma query SQL arbitrária. Nesse caso: SELECT \* FROM public.gastos WHERE valor > 50;

### **4. Criando os fluxos CRUD no n8n (usando Postgres)**

Para cada operação, criaremos um workflow separado e eu vou apresentar o método usando Operation + Filters (modo declarativo no node). Tente também fazer via Execute Query e também pelo Supabase.

**4.1 Create (Inserir um novo gasto)**

### **Método: POST JSON para enviar:**

{

"item": "Pizza",

"valor": 85.5,

"fornecedor": "Dominos",

"tipo\_despesa": "pessoal",

"data": "2025-06-22"

}

## **4.2 Read (Listar últimas despesas)**

### **Método: GET Sem body necessário**

### **Usando Operation: Select com filtro**

### **Webhook (GET)**

### **Postgres Node**

### **Operation: Select**

### **Table: gastos Filters: Ordenar por data (descendente) Limite: 10**

### 

## **4.3 Update (Atualizar um gasto)**

### **📌 Método: POST 📦 JSON para enviar:**

{

"id": "c537f3d4-8d1d-4ccf-9419-4d0a7c0d13f5",

"item": "Pizza Calabresa",

"valor": 92.00,

"fornecedor": "Dominos",

"tipo\_despesa": "pessoal",

"data": "2025-06-22"

}

### 

### **Webhook (POST) Postgres Node**

### **Operation: Update**

### **Table: gastos**

### **Filtro:**

### **Coluna: id**

### **Condição: Equal**

### **Valor: {{$json["id"]}} Campos a atualizar: item, valor, fornecedor, tipo\_despesa, data**

## **4.4 Delete (Remover um gasto)**

### **📌 Método: DELETE 📦 JSON para enviar:**

{

"id": "c537f3d4-8d1d-4ccf-9419-4d0a7c0d13f5"

}

## **5. Configurando e Utilizando o AI Agent com Tools no n8n**

No n8n, um **AI Agent** é um nó especial (baseado em LangChain) que funciona como um agente autônomo capaz de tomar decisões e usar outras ferramentas para cumprir uma tarefa. Diferente de um simples fluxo fixo, o agente utiliza um modelo de linguagem para decidir quais ações tomar e quais *tools* (ferramentas) chamar para obter informações ou executar comandos. Em outras palavras, o agente analisa a pergunta ou comando que você envia e **dinamicamente determina** quais ferramentas usar para chegar a uma resposta.

**Supabase e AI Agents:** Para cenários de IA, Supabase é relevante tanto como um simples banco Postgres (por exemplo, armazenando dados que o agente vai consultar via SQL) quanto como um vetor de embeddings. O n8n possui inclusive um nó *Supabase Vector Store* para usar o recurso de vetor do Supabase (PGVector) no contexto de **RAG (Retrieval-Augmented Generation)**. Ou seja, você pode inserir embeddings de textos no Supabase e depois consultá-los via similaridade. Se seu caso de uso inclui IA buscar informações armazenadas, pode optar por ou consultar o Postgres diretamente com SQL gerado pela IA (como no exemplo anterior financeiro), ou indexar conhecimento no vector store do Supabase e usar um *Vector Store Tool* com o AI Agent. Ambas abordagens são possíveis e mencionadas em materiais do n8n (por exemplo, vídeos e tutoriais sobre Supabase + n8n para RAG).

Criar um **agente inteligente** que interpreta mensagens como:

**"Gastei 89 reais com Uber ontem"**

### **Configuração passo a passo**

#### **1. Crie um novo workflow no n8n**

* **Gatilho**: When Chat Message Received  
   (pode usar também Webhook, Telegram ou WhatsApp como input no futuro)

#### **2. Adicione o nó AI Agent**

* Esse será o **cérebro** do seu fluxo
* Configure o modelo (ex: OpenAI GPT-4o-mini)
* Conecte uma **memória** (Simple Memory ou Window Buffer Memory) se desejar manter contexto de conversas

#### **3. Adicione o nó Postgres Tool ao AI Agent**

* Tipo: Insert
* Tabela: gastos

### **System Message — Design refinado**

Aqui está um exemplo mais robusto de system prompt com clareza, estrutura e regras (padrão "Role + Task + Constraints + Output"):

[Role]

Você é um agente de IA especializado em controle financeiro pessoal.

[Objective]

Seu único objetivo é registrar despesas informadas pelo usuário no banco de dados.

[Tools]

Você vai usar o banco postgres para salvar os itens

[Task]

1. Identifique os seguintes campos na mensagem do usuário:

- item: o que foi comprado ou gasto

- valor: o custo total (em reais)

- data: a data da despesa (relativa ou explícita)

- fornecedor: o local ou serviço associado (opcional)

- tipo\_despesa: categoria como pessoal, empresarial, alimentação, etc.

2. Use a ferramenta de banco de dados para inserir os dados extraídos na tabela `gastos`.

3. Use a data atual se o usuário disser “hoje” e subtraia um dia se disser “ontem”.

[Constraints]

- Não responda perguntas fora do escopo de controle de despesas.

- Se a mensagem não contiver um valor claro, solicite que o usuário informe.

- Sempre utilize ponto (.) como separador decimal.

- Nunca tente adivinhar dados que não foram ditos.

[Output]

Após salvar os dados, responda com:

"✅ Despesa registrada: [item] – R$[valor] em [data]."

Se houver erro, diga:

"❌ Não consegui registrar a despesa. Pode repetir com mais detalhes?"

### **Teste prático**

Mensagem:

**"Ontem gastei 120 reais com gasolina no Posto Shell"**

#### **Esperado:**

* **item**: gasolina
* **valor**: 120
* **fornecedor**: Posto Shell
* **tipo\_despesa**: pessoal *(inferido ou padrão)*
* **data**: data de ontem (ex: 2025-06-22, se hoje é 23/06/2025)

### **6. Conclusão**

Ao construir automações inteligentes no n8n, você pode se deparar com duas abordagens: utilizar integrações diretas (nós específicos para cada serviço, sequenciados de forma determinística) ou empregar um **AI Agent com tools**, que decide dinamicamente quais ações tomar. Cada abordagem tem seus prós e contras, e a escolha depende do caso de uso.

**Integração direta (nós individuais):** Isso significa usar, por exemplo, o nó PostgreSQL, nó Google Sheets, etc., em sequência com lógica tradicional (ifs, loops) para atingir um objetivo fixo. As vantagens aqui são **confiabilidade e previsibilidade** – o fluxo fará sempre as mesmas operações, e você tem controle exato de cada passo. É ideal para processos bem definidos, onde não há ambiguidade no que deve ser feito. Também tende a ser mais **eficiente** em termos de desempenho, já que não envolve chamadas extras ao modelo de linguagem para decidir etapas. Por exemplo, para inserir um dado no Supabase quando um webhook chega, basta usar o nó Supabase diretamente – simples e eficaz.

No contexto de dados, usar um nó Postgres ou Supabase diretamente pode evitar sobrecarga cognitiva no modelo de IA. Por exemplo, se você precisa extrair 100 registros e fazer cálculos, talvez seja melhor o workflow fazer isso via SQL ou código do que “perguntar” a um agente para descobrir isso – o agente poderia ter dificuldade ou consumir muitos tokens.

**AI Agent com Tools:** Brilha em cenários de **flexibilidade e linguagem natural**. Essa abordagem permite que o usuário final interaja em linguagem comum (“Quais produtos tiveram aumento de vendas?”) e o agente descubra sozinho que isso implica, digamos, consultar uma planilha ou banco de dados e depois chamar uma API para contexto adicional, antes de responder. Ou seja, quando a sequência exata de passos não é conhecida de antemão (porque depende da pergunta do usuário), o agente com tools é extremamente poderoso. Ele combina a **inteligência do LLM** em interpretar pedidos complexos com a **capacidade de ação** fornecida pelos tools.

Um exemplo apontado pela comunidade n8n é no domínio de análise de dados financeiros: ao invés de pré-definir consultas, deixar um AI Agent gerar SQL sob demanda para responder perguntas permite muito mais flexibilidade – o usuário pode perguntar qualquer coisa sobre os dados, e o agente monta a query apropriada. Essa solução, inclusive, pode superar abordagens de *vector database* quando se trata de consultas numéricas exatas, conforme descrito no template do n8n: ao usar diretamente o Postgres via agente, obtém-se resultados precisos e rápidos, sem ter que converter tudo em embeddings e correr o risco de imprecisão em números.

**Trade-offs e Recomendações:** A principal desvantagem de usar um AI Agent é que você **depende do modelo de IA acertar** o que fazer. Há possibilidade de erros de “hallucination” (o modelo pode chamar um tool de forma errada, ou interpretar errado os dados). Por isso, para tarefas críticas ou que envolvem escrita em sistemas (ex: deletar algo em banco), deve-se ter muito cuidado ao delegar a um agente. Se optar por essa via, vale restringir o agente via system message e validar as saídas. Uma boa prática é começar usando o agente em modo “somente leitura” das ferramentas primeiro, até ter confiança, antes de permitir que ele faça ações destrutivas.

Enquanto isso, fluxos determinísticos não sofrem desse risco – eles só fazem o que foram programados para fazer. Em contrapartida, não entendem linguagem natural nem fogem do roteiro pré-definido.

Portanto, uma recomendação comum é: **use integração direta quando o processo é fixo, repetitivo e bem compreendido**, e **use AI Agent quando quiser proporcionar uma interface mais inteligente/flexível ou automatizar decisões complexas**. Muitas vezes, ambas abordagens se complementam. Por exemplo, você pode ter um workflow tradicional que, ao detectar certa necessidade, entrega controle para um AI Agent resolver sub-tarefas criativas. Ou um AI Agent que, ao invés de ter tools para tudo, apenas prepara dados e repassa para nós fixos (ex: o agente decide *o que* fazer e armazena numa variável, e então o fluxo normal executa aquela ação).

Um ponto citado anteriormente ilustra também a diferença: na integração Supabase, o nó Supabase pode falhar em retornar dados por causa de RLS, mas o nó Postgres direto funciona. Isso não é exatamente escolha agente vs direto, mas mostra que às vezes a via direta é mais adequada para acessar os dados sem restrição. Em contrapartida, se você quiser um chatbot que responde dúvidas sobre os dados do Supabase, aí sim um agente com tool (consultando Postgres ou a API Supabase) é o caminho para permitir perguntas arbitrárias.

**Desempenho:** Considerar que cada passo de um agente envolve rodar um modelo (possivelmente várias vezes) – isso pode ser **mais lento** e custoso (se for um modelo pago tipo GPT-4) do que um equivalente determinístico. Logo, não use um agente para algo trivial que poderia ser um simples nó – use quando agrega valor em termos de compreensão da linguagem ou tomada de decisão não trivial.

Em conclusão, conforme as observações na documentação e exemplos do n8n:

* Workflows tradicionais (nós integrados sequenciais) são ótimos para **ETLs, sincronizações, acionamentos simples**, onde sabemos exatamente o fluxo.
* AI Agents são indicados para **assistentes conversacionais, automação orientada a perguntas, busca inteligente de informações**, e cenários onde incorporar “raciocínio” do modelo vale a pena. Por exemplo, um agente de suporte que consulta Google Sheets e Postgres (via tools) conforme a pergunta do usuário – isso simplifica muito a construção, pois o desenvolvedor não precisa prever todas as combinações de perguntas; o agente lida com isso.

O n8n permite inclusive misturar as duas abordagens no mesmo workflow. A escolha deve ser guiada pela necessidade de **flexibilidade** vs **controle**. Sempre que possível, consulte a documentação oficial e os templates de referência – muitos trazem notas se preferiram usar um agente ou um fluxo fixo e por quê. Assim, você pode se inspirar nas melhores práticas aplicadas em casos reais.