Respostas imutáveis ao longo do tempo

Auditoria, JSON e Replicação

Marcos Pegoraro - F10 Software
marcos@f10.com.br

- **Histórico de um Registro**
  - Cliente
- **Histórico de um Processo**
  - Cliente, Contrato e Parcelas
- **Histórico de um grupo de Processos**
  - Todos os Contratos em um determinado momento

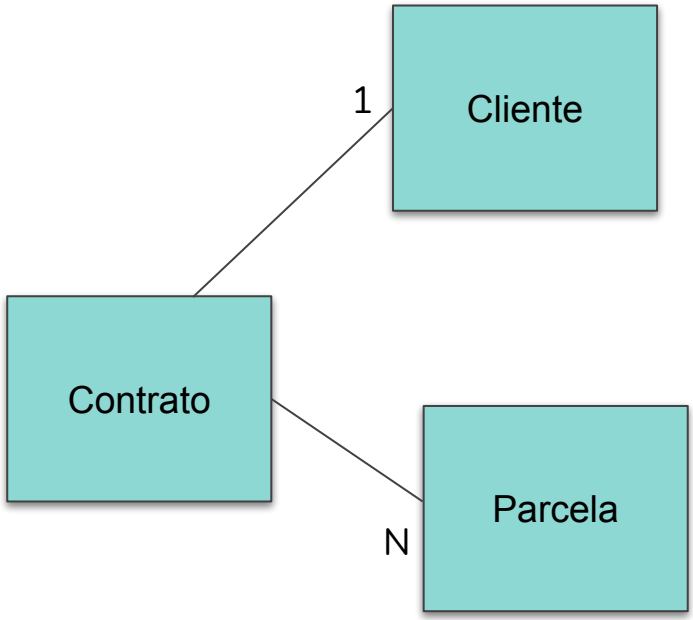
Modelos de auditoria

- O que armazenar:
  - Inclusões, Alterações e Exclusões
  - Apenas Alterações e Exclusões
- Function:
  - Tabela
  - Genérica
- Armazenamento:
  - Campo por Campo
  - JSON
- Consultas:
  - Mesclando dados entre tabela e auditoria
  - Dados provenientes somente da auditoria

Importância de uma boa Auditoria

- Alterações normais dos dados
- Alterações ou Exclusões acidentais
- Alteração nos dados muda o resultado da consulta ao longo do tempo
- Respostas imediatas para problemas de atualizações e usuários

- Usuário usando a ferramenta de forma no mínimo inesperada

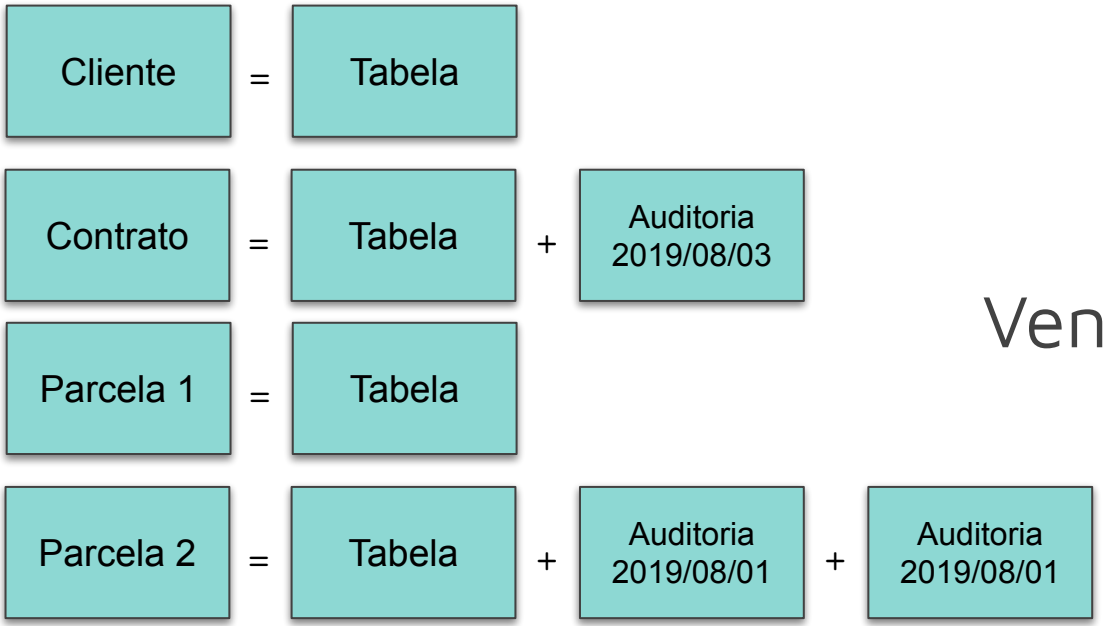


Cliente:
Inserido - ‘2019/03/15’
Alterado - ‘2019/07/01’ Status, CEP

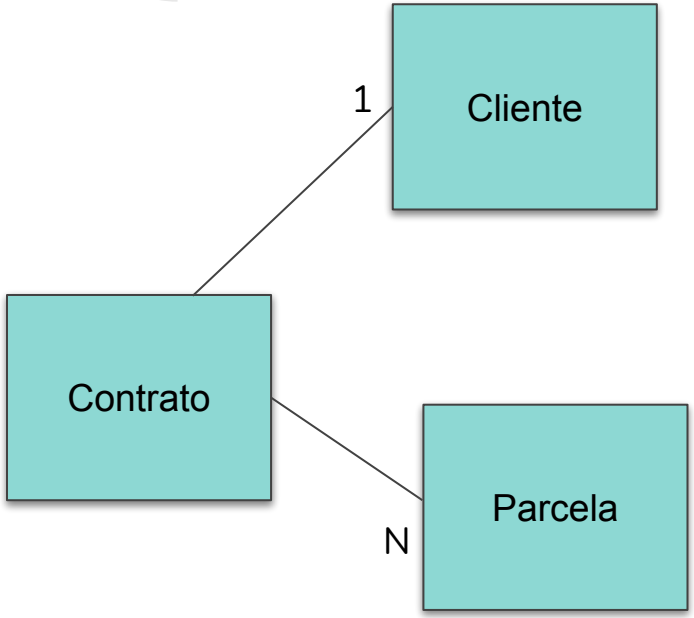
Pedido:
Inserido - ‘2019/07/30’
Alterado - ‘2019/08/03’ Cancelado

Item1:
Inserido - ‘2019/07/30’
Item2:
Inserido - ‘2019/07/30’
Alterado - ‘2019/08/01’ Valor
Alterado - ‘2019/08/01’ Valor

Dados a partir da Tabela + Auditoria ou somente da Auditoria



Vendas em 31/07

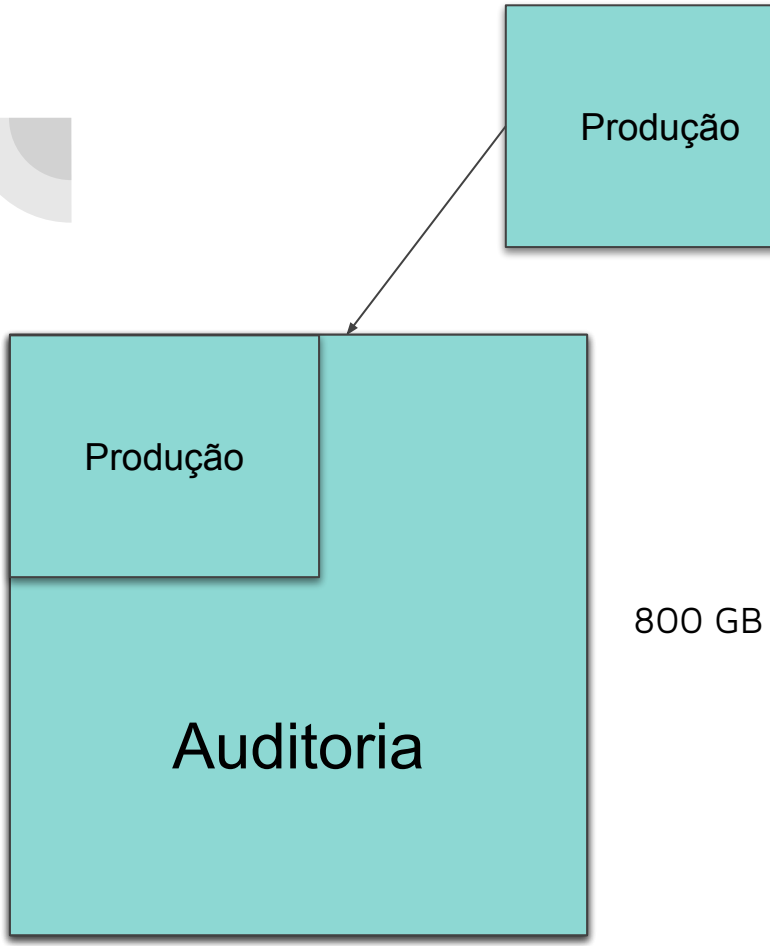


Cliente:
Criação - ‘2019/01/05..2019/03/01’ (JSON)
Alteração - ‘2019/03/01..infinity’ (JSON)

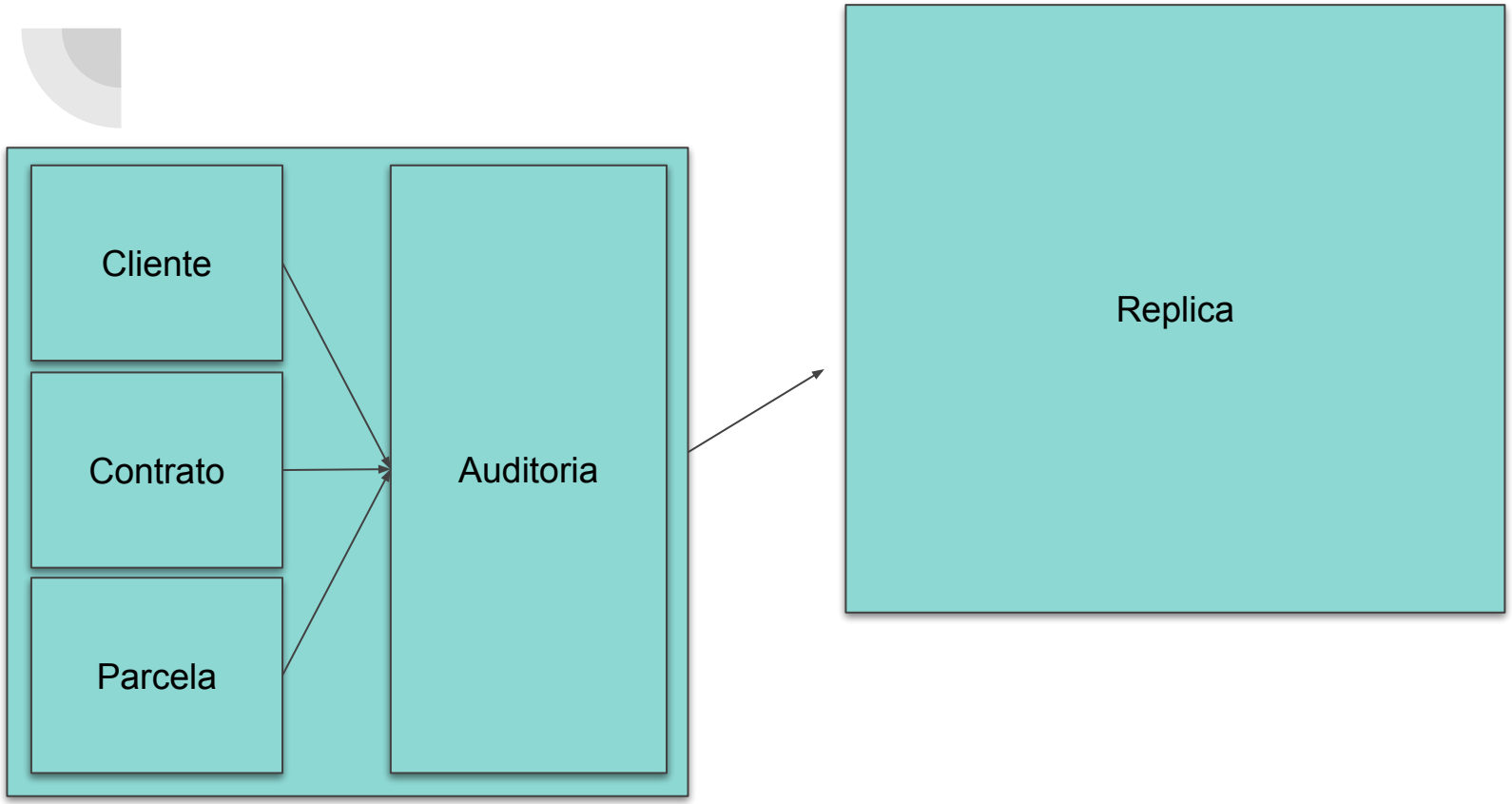
Contrato:
Criação - ‘2019/07/30..2019/08/03’ (JSON)
Alteração - ‘2019/08/03..infinity’ (JSON)

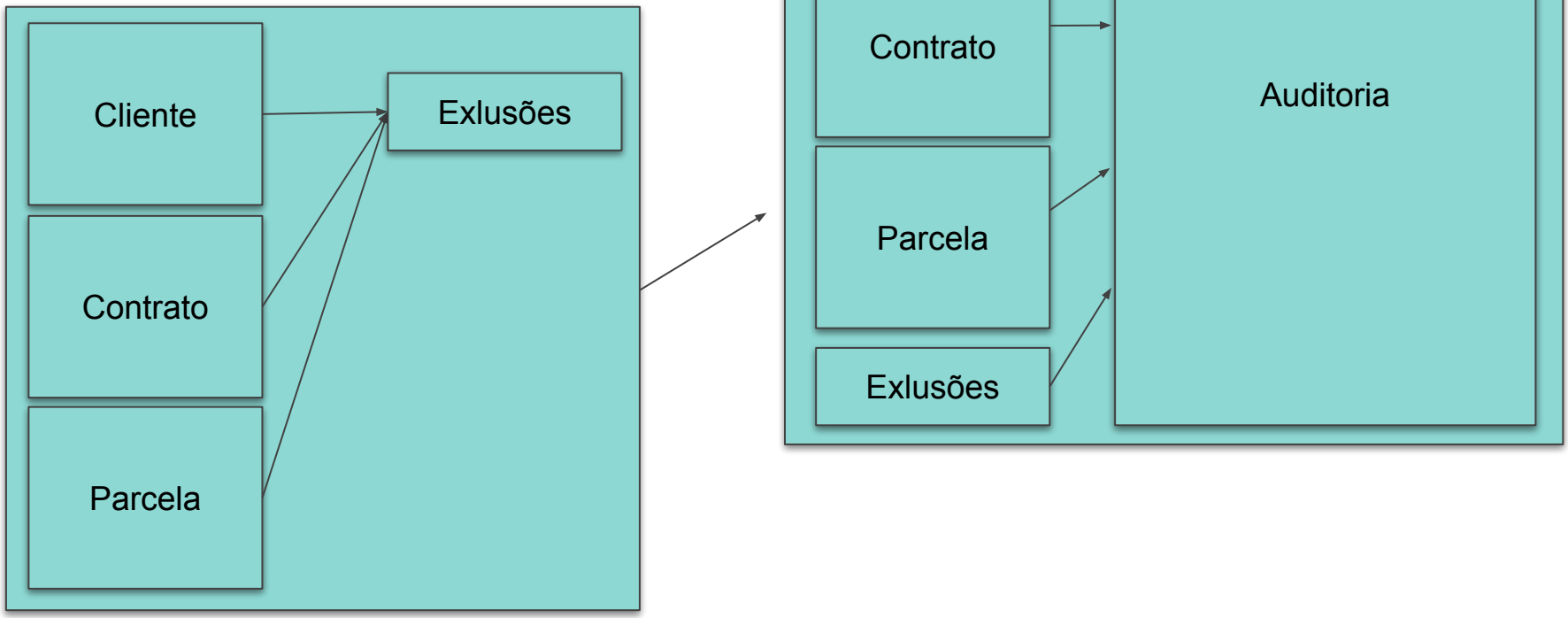
Parcela 1:
Criação - ‘2019/07/30..infinity’ (JSON)
Parcela 2:
Criação - ‘2019/07/30..2019/08/01’ (JSON)
Alteração - ‘2019/08/01..2019/08/01’ (JSON)
Alteração - ‘2019/08/01..infinity’ (JSON)

```
DataAlteração @> '2019/07/31'::Date
```



- Produção apenas identifica usuário e datahora da inserção, alteração ou exclusão
- Replicação executa trigger que armazena em JSONB e determina a validade daquele registro e invalida o registro anterior
- Tamanho depende muito. Tipo de informação, quantidade de alterações e exclusões e triggers


Cliente
Contrato
Parcela
Auditoria
Replica


Cliente
Contrato
Parcela
Exlusões
Cliente
Contrato
Parcela
Exlusões
Auditoria

- Produção
  - Tabela Exclusão
  - Triggers em todas as tabelas definindo usuário e data hora da operação.

- Replicação
  - Tabela Exclusão
  - Trigger na tabela de Exclusão atualiza o registro como excluído.
  - Tabela Auditoria
  - Trigger que audita em todas as tabelas
  - Carregamento inicial
    - Disable subscription
    - Insert na auditoria como está o registro inicialmente
    - Enable subscription

- SQL realizado exclusivamente na Auditoria
  - Considerar sempre a tabela principal da consulta
  - Caso alguma tabela filha seja demasiado grande, buscar antes da tabela principal

Vamos escrever ...

Auditoria modelo 1

- *Relação Mestre Detalhe*
- *Registros do Detalhe são armazenados como text*
- *Function que faz auditoria compara campo por campo para auditar, portanto precisa de uma function para cada tabela*

```
create table if not exists auditoria.audit(
Audit ID bigint not null constraint pkAudit primary key,
Table Name text not null,     Primary Key integer,    User Name text not null,
IsUpdate boolean,             Date_Time timestamp,    Schema_Name text);
```

```
create table if not exists auditoria.AuditFields(
AuditField ID bigint not null constraint pkAuditFields primary key,
Audit ID bigint not null constraint fkauditaudit references auditoria.Audit,
Field_Name text, old_Value text, new_Value text);

create function cad_produto_aud() returns trigger language plpgsql as $$
declare
 vAudit_Id integer;
begin
 insert into auditoria.audit(Audit ID,Table Name, Primary_Key, IsUpdate, User_Name, Date_Time,
Schema Name) values (vAudit Id, 'cad produto',
          case TG OP when 'UPDATE' then new.produto id else old.produto id end, (TG OP = 'UPDATE'),
          Current User, clock_timestamp(), TG_TABLE_SCHEMA)returning Audit_Id into vAudit_Id;
 if (TG OP = 'UPDATE') then
   if (old.ativo is distinct from new.ativo) then
     insert into auditoria.auditfields(Audit ID, Field_Name, old_Value, new_Value)values (vAudit_Id,
     'ativo', old.ativo::text, new.ativo::text);
   end if;
   --Demais campos
 elseif (TG OP = 'DELETE') then
   if (old.ativo is not null) then
     insert into auditoria.auditfields(audit_id, field_name, old_Value)values (vAudit_Id, 'ativo',
     old.ativo::text);
   end if;
   --Demais campos
 end if;
 return null;
end; $$;
```

Auditoria modelo 1 - function usando JSON

- *Function que faz auditoria compara os valores para saber se houve alteração usando JSON, isso faz com que ela seja genérica.*
- *Funciona para todas as tabelas do banco, precisa somente criar o trigger apontando para esta function.*
- *Function realiza um único insert também comparando new e old values através de JSON.*
- *Nesse modelo não são auditadas inserções, portanto precisa buscar valores na tabela*

```
create function Auditoria_Por_JSON() returns trigger language plpgsql as $$
declare
 vAudit Id integer;
 new JSONB JSONB; old_JSONB JSONB;
begin
 if (TG OP = 'UPDATE') then --Só grava a auditoria se houver efetivamente alteração
   new JSONB = row to json(new)::jsonb - '{user name,datahoraalteracao}'::text[];
   old JSONB = row to json(old)::jsonb - '{user_name,datahoraalteracao}'::text[];
   if (new JSONB = old_JSONB) then
     return null;
   end if;
 end if;
 insert into auditoria.audit(Table Name, Primary Key, IsUpdate, User Name, Date Time,Schema Name)
   values (tg table name, (select Value from json each text(row to json(case when (TG_OP = 'UPDATE')
     then new else old end)) limit 1)::integer, (TG OP = 'UPDATE'),
       Current User, clock_timestamp(), tg_table_schema)returning Audit_Id into vAudit_Id;
 if (TG OP = 'UPDATE') then
   insert into auditoria.auditfields(audit id, field name, old Value, new_Value)
     select vAuditId, original.key, original.value, alterado.value
     from (select json each.key, json each.valuefrom json each text(row to json(new)) order by json each.key) original
     join (select json each.key, json_each.valuefrom json_each_text(row_to_json(old)) order by json_each.key) alterado on
(original.key = alterado.key)
     where alterado.value is distinct from original.value;
 else
   insert into auditoria.auditfields(audit id, field_name, old_Value)
     select vAuditId, original.key, original.value
     from (select json each.key, json_each.valuefrom json_each_text(json_strip_nulls(row_to_json(old))) order by
json each.key) original;
 end if;
 return null;
end; $$;
```

Auditoria modelo 2 - Produção

- *No Banco de Produção precisamos apenas da tabela de exclusões, para que saibamos que registros foram excluídos, quando e por quem.*
- *Vantagem que não ocupamos espaço no banco de produção, assim como não afetamos sua performance já que nele serão somente gravados os registros excluídos.*
- *É obrigatório que o primeiro registro da tabela seja chave primária e que este seja numérico*

```
create table if not exists auditoria.exclusoes(
Exclusao_ID integer not null constraint pk_Exclusao_ID primary key,
Chave integer, UsuarioAlteracao integer, Cliente text, Tabela text, DataHoraAlteracao timestamp
);
```

```
CREATE OR REPLACE FUNCTION Auditoria.ExcluiRegistro() RETURNS TRIGGER AS $BODY$
DECLARE
 vChavePrimaria TEXT;
BEGIN
 IF (tg_when <> 'AFTER' and tg_op <> 'DELETE') THEN
   RAISE EXCEPTION 'Erro ao gerar a auditoria do registro!' USING HINT = Format('O trigger %L não pode ser executado no
evento %L em %s.%s', tg_name, tg_when, tg_table_schema, tg_table_name);
 END IF;
 vChavePrimaria = TG_ARGV[0];
 IF (coalesce(vChavePrimaria,'') <>'') AND (json_typeof(row_to_json(old.*)->(vChavePrimaria)) = 'number') THEN
   INSERT INTO Auditoria.ExcluiRegistro(Cliente, Tabela, Chave, usuarioalteracao, datahoraalteracao)
   VALUES (tg_table_schema, tg_table_name, (row_to_json(OLD.*)->>(vChavePrimaria))::INTEGER, carregaf10running(),
clock_timestamp());
 END IF;
 RETURN NULL;
END;$BODY$ LANGUAGE plpgsql;
```

Auditoria modelo 2 - Produção

- *Criação do trigger que realiza a inserção dos registros excluídos*

```
CREATE OR REPLACE FUNCTION Auditoria.AuditarRegistrosExcluidos()returns void AS $F$
DECLARE
 comando RECORD;
BEGIN
 FOR comando IN SELECT schemaname, relname,column name, Trim(Format($$
  DROP TRIGGER IF EXISTS excluiregistro ON %1$s.%2$s; CREATE TRIGGER EXLUIREGISTRO AFTER DELETE ON %1$s.%2$s FOR EACH ROW
EXECUTE PROCEDURE auditoria.excluiregistro(%3$L);
    $$, schemaname, relname,column name)) sql FROM
   ( SELECT schemaname, relname from pg_stat_user_tables WHERE schemaname !~* 'pg_catalog|public'
ORDER BY schemaname, relname ) tabelas
   INNER JOIN information schema.table_constraints constritores ON constritores.table_catalog = current_database AND
constritores.table schema =
     tabelas.schemaname AND constritores.table name = tabelas.relname AND constritores.constraint type = 'PRIMARY KEY'
   INNER JOIN information schema.key_column_usage chaves ON chaves.table_catalog = constritores.table_catalog AND
constritores.table schema =
     chaves.table schema AND constritores.table_name = chaves.table_name AND constritores.constraint_name =
chaves.constraint name
 ORDER BY schemaname, relname LOOP
     EXECUTE comando.sql;
 END LOOP;
END;$F$ LANGUAGE plpgsql;
```

Auditoria modelo 2 - Réplica

- *Os Registros que foram excluídos no banco de produção precisam ser adicionados a auditoria, para que saibamos a partir de quando esse registro não existe mais.*
- *Nesta function são inseridos na auditoria os registros excluídos na produção*

```
create table if not exists auditoria.exclusoes(
Exclusao_ID integer not null constraint pk_Exclusao_ID primary key,
Chave integer, Usuario_Alteracao integer, Cliente text, Tabela text, DataHoraAlteracao timestamp
);

create table if not exists auditoria.auditoria (
 Auditoria_ID bigint not null constraint pk_Auditoria_ID primary key,
 Usuario_Alteracao integer not null, Usuario_Exclusao integer, Chave integer, Cliente text, Tabela text,
 Erro text, Registro jsonb, Inclusao timestamp default clock_timestamp(),
 DataHoraAlteracao tsrange default tsrange((clock_timestamp())::timestamp without time zone, 'infinity'::timestamp without
time zone)
);
create function excluiregistro() returns trigger language plpgsql as $$
DECLARE
 vDataHoraAlteracao timestamp; vDataHoraAlteracaoAnterior TIMESTAMP; vAuditaRegistro_ID BIGINT; vErro Text;
BEGIN
 vDataHoraAlteracao = new.DataHoraAlteracao;
 SELECT Lower(DataHoraAlteracao)::TIMESTAMP, AuditaRegistro_ID into vDataHoraAlteracaoAnterior,
 vAuditaRegistro_ID FROM auditoria.AuditaRegistro WHERE Cliente = new.Cliente AND Tabela = new.Tabela
 AND Chave = new.Chave AND Upper(DataHoraAlteracao) = 'infinity' and usuario_Exclusao is null;
 UPDATE auditoria.AuditaRegistro SET DataHoraAlteracao = tsRange(Lower(DataHoraAlteracao), vDataHoraAlteracao), erro = vErro,
usuario_Exclusao = new.UsuarioAlteracao
 WHERE auditaregistro_id = vAuditaRegistro_ID;
 IF NOT FOUND THEN
   vErro = Format('[Alerta] Não foi encontrado registro de criação ou update válido para esta exclusão ocorrida em %L',
vDataHoraAlteracao);
   INSERT INTO auditoria.AuditaRegistro(Cliente, Tabela, Chave, UsuarioAlteracao, DataHoraAlteracao, erro)
   VALUES (new.Cliente, new.Tabela, new.Chave, new.UsuarioAlteracao, tsRange(vDataHoraAlteracao, vDataHoraAlteracao +
INTERVAL '1 microsecond'), vErro);
 END IF;
 RETURN NULL;
END; $$;
```

Auditoria modelo 2 - Réplica

- *Todas as alterações realizadas na produção precisam ser auditadas.*
- *Aqui são auditados os registros inseridos e atualizados na produção.*

```
CREATE OR REPLACE FUNCTION Auditoria.AuditaRegistro()RETURNS TRIGGER AS $body$
DECLARE
 vChave                      INTEGER;
 vAuditoria ID               INTEGER;
 vDataHoraAlteracao          TIMESTAMP;
 vCompararDataHoraAlteracaoTIMESTAMP;
 vErro                       TEXT;
BEGIN
 IF tg op = 'UPDATE' THEN
   vDataHoraAlteracao =new.DataHoraAlteracao;
   SELECT auditaregistro id, Lower(DataHoraAlteracao)INTO vAuditoria ID, vCompararDataHoraAlteracao
   FROM Auditoria.AuditaRegistroWHERE Cliente = tg table schemaAND Tabela = tg_table_nameAND Chave = vChave AND
Upper(DataHoraAlteracao) = 'infinity'AND Registro IS NOT NULL;
   UPDATE auditoria.AuditaRegistroSET DataHoraAlteracao = tsRange(Lower(DataHoraAlteracao), vDataHoraAlteracao),
erro = vErro WHERE auditaregistro_id = vAuditoria_ID;
 END IF;
 INSERT INTO auditoria.AuditaRegistro(Cliente, Tabela, Chave, UsuarioAlteracao, DataHoraAlteracao, Registro)
 VALUES (tg table schema, tg table name, vChave,new.UsuarioAlteracao, tsRange(new.DataHoraAlteracao, 'infinity'),
json strip nulls(row_to_json(new.*)));
 RETURN NULL;
END;
$body$ LANGUAGE plpgsql;
```

Auditoria modelo 2 - Réplica

- *Criamos Triggers para todas as tabelas na réplica e para que o trigger seja disparado como desejamos é necessário* ENABLE REPLICA TRIGGER

```
SELECT string_agg(Format(
 'DROP TRIGGER IF EXISTS auditaregistro %$2s ON %1$s.%2$s;' |¢hr(13) ||
 'CREATE TRIGGER auditaregistro AFTER INSERT OR UPDATE ON %1$s.%2$s FOR EACH ROW EXECUTE PROCEDURE
auditoria.auditaregistro(%L);' ||chr(13) ||
 'ALTER TABLE %1$s.%2$s ENABLE REPLICA TRIGGER auditaregistro;'
 , schemaname, relname, column name
), chr(13)||chr(13) order by schemaname, relname)
FROM (
 select schemaname, relname from pg_catalog.pg_stat_user_tables where pg_stat_user_tables.schemaname !~*
'pg_catalog|public'
) tabelas
INNER JOIN information_schema.table_constraints constritores ON constritores.table_catalog = 'f10db' AND
constritores.table_schema = tabelas.schemaname
AND constritores.table_name = tabelas.relname AND constritores.constraint_type = 'PRIMARY KEY'
INNER JOIN information_schema.key_column_usage chaves ON chaves.table_catalog = constritores.table_catalog AND
constritores.table_schema = chaves.table_schema
AND constritores.table_name = chaves.table_name AND constritores.constraint_name = chaves.constraint_name;
```

Auditoria modelo 2 - Réplica

- *Como faremos as consultas apenas na auditoria precisamos fazer um insert inicial e a partir daí os triggers alimentarão a medida que as alterações acontecem.*
- *Select que gera um script a ser executado na Réplica o qual copia dados de todas as tabelas da replica para a auditoria.*

```
SELECT concat('/*', schemaname, '*/', chr(13), string agg(trim( chr(13)||chr(10) from Format($$
INSERT INTO auditoria.AuditaRegistro(usuarioAlteracao, chave, cliente, tabela, registro, dataHoraAlteracao)
SELECT coalesce(%2$s.usuarioalteracao, (SELECT (usesysid::INTEGER * -1) FROM pg user WHERE usename = current user)),
%2$s.%3$s, %1$L, %2$L, row to json(%2$s.*), tsrange(%2$s.datahoraalteracao::timestamp, 'infinity') FROM %1$s.%2$s;
$$, schemaname, relname,column_name)), chr(13) order by relname)
)as sql
FROM (
 SELECT schemaname, relname FROM pg catalog.pg_stat_user_tables
 WHERE schemaname !~* 'pg catalog|public'
 ORDER BY schemaname, relname
) tabelas
INNER JOIN information schema.table constraints constritores ON constritores.table catalog = 'f10db' AND
constritores.table_schema = tabelas.schemaname AND constritores.table_name = tabelas.relname AND constritores.constraint_type
= 'PRIMARY KEY'
INNER JOIN information schema.key column usage chaves ON chaves.table catalog = constritores.table_catalog AND
constritores.table schema = chaves.table schema AND constritores.table_name = chaves.table_name AND
constritores.constraint_name = chaves.constraint_name
GROUP BY schemaname;
```

Auditoria modelo 2 - Réplica

- *Tudo pronto, agora é só fazer os selects que se deseja na Réplica*
- *Tenha sempre em mente que precisa definir um schema específico, já que a auditoria acontece para todos eles.*
- *Você precisa definir de quando quer o resultado, por exemplo: Date_Trunc('month', current_date).*
- *Extraia os dados do JSON sempre por CTE para que você consiga montar a lista de tabelas com os valores e a partir delas fazer os joins e wheres necessários.*

```
EXPLAIN ANALYSE
WITH Const(cliente, data) as (
 values ('SchemaDesejado', '2019/08/01'::timestamp) --Informe qual Schema e Momento no tempo que deseja o resultado
), contratos as (
 select contrato id, numerocontrato, pessoa id, curso_id, fonte_id, status, vendedor_id, matricula, datacancelamento
   from auditaregistro contratos cross join Const
     /* Seria mais prático usar um type, mas pode ser usado somente os campos que interessam */
   JOIN LATERAL (select * from jsonb populate record(null::type mov Contrato,registro)) jsonb to recordon TRUE
   WHERE contratos.cliente = config.clienteand contratos.tabela = 'mov_contrato'and (registro->>'status')::INTEGER = 1 and
contratos.datahoraalteracao @> config.data
), parcelas as (
 select contrato id, receber_id, parcela, tiporecebimento, valor, lancamento, vencimento, cobranca, quitacao, promessa,
valorpago from config
   join auditaregistro p on TRUE
   join lateral jsonb to record(p.registro)as (receber id INTEGER, contrato id INTEGER, parcela INTEGER, tiporecebimento
smallint, valor numeric(15,2), lancamento DATE, vencimento DATE, cobranca DATE, quitacao timestamp, promessa DATE, valorpago
numeric(15,2)) on TRUE
 where p.cliente = config.clienteand p.tabela = 'fin_receber'and p.datahoraalteracao @> config.data
 order by vencimento, parcela
)
SELECT
 concat(row number() OVER porcontrato, '/', count(*) OVER porcontrato), contratos.*,  parcelas.* FROM
 parcelas JOIN contratos on (parcelas.contrato id = contratos.contrato_id)
WINDOW porcontrato as (PARTITION BY contratos.contrato_id);
```

Resumindo ...

- *Fazer a auditoria na réplica é melhor porque o servidor de produção não é sobrecarregado, nem em espaço nem em processamento.*
- *Por performance nem sempre seu select partirá da tabela principal, existem casos que a tabela dependente é a principal da CTE, dependendo do tipo de joins e wheres que você utilize.*
- *Para que se faça os joins no select da auditoria é necessário extrair a tabela inteira de dentro dos jsons já que tudo está dentro do JSON, exceto a PK.*
- *Considere usar indices GIN para o json, porém tenha também consciência que isto fará com que aumente consideravelmente o tamanho da auditoria.*
- *Verifique se não está havendo auditoria sem necessidade, por exemplo se houver diversos registros da mesma tabela e primary key. Isto quer dizer que há algo de errado na aplicação ou um update sendo disparado num trigger sem necessidade.*
    - *select Tabela, Chave, Date_Trunc('second',inclusao) from auditaregistro where Cliente = 'Schema_518' and tabela = 'contrato' group by 1,2,3 having count(*) > 1;*