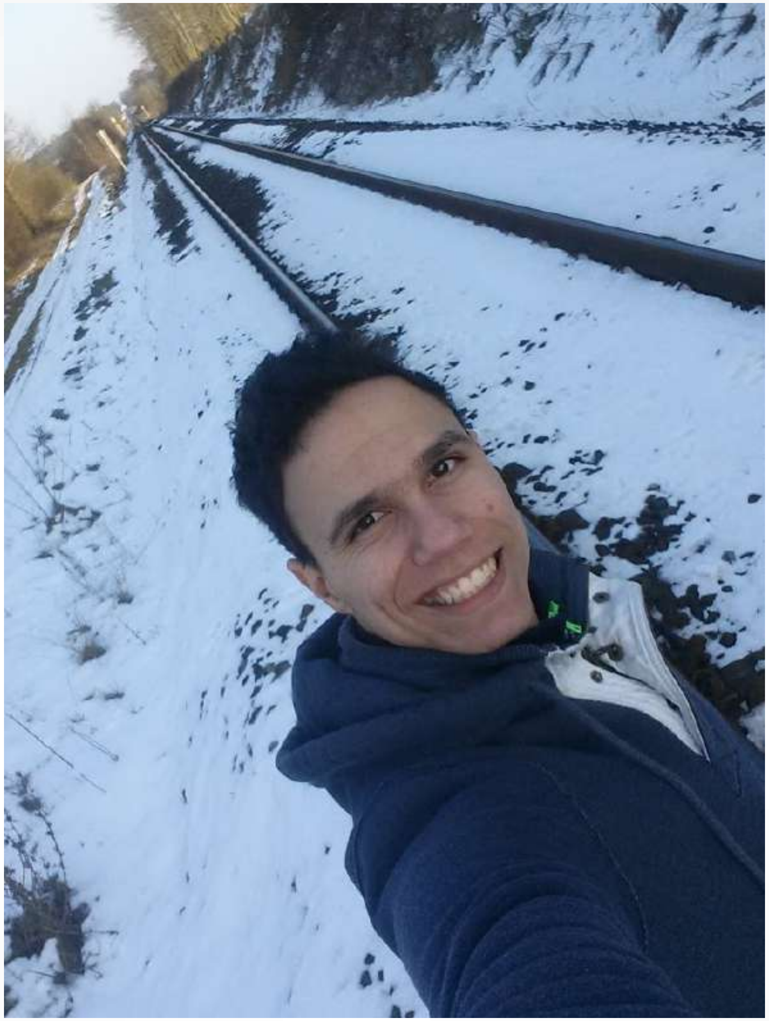
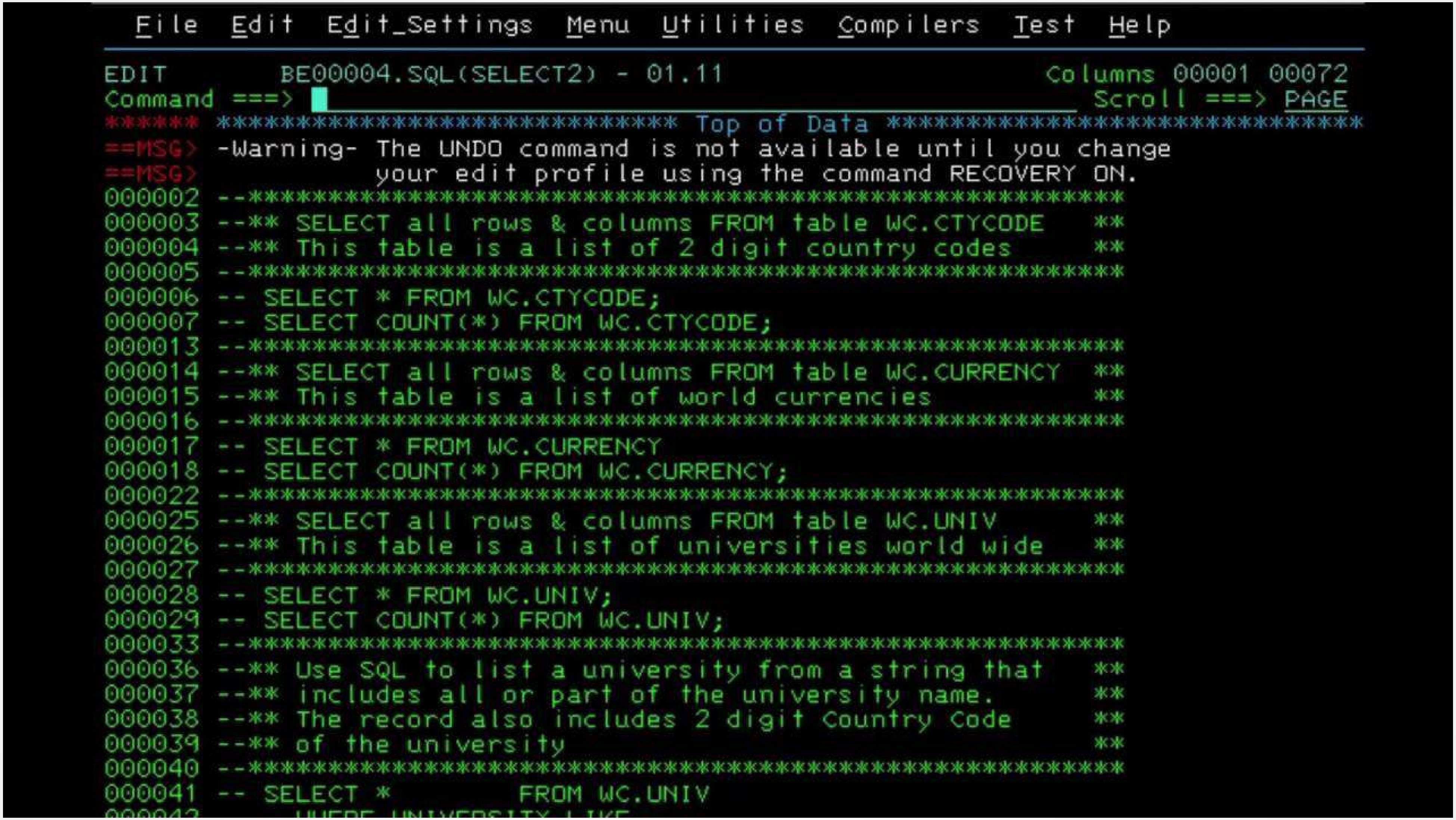
citusdata

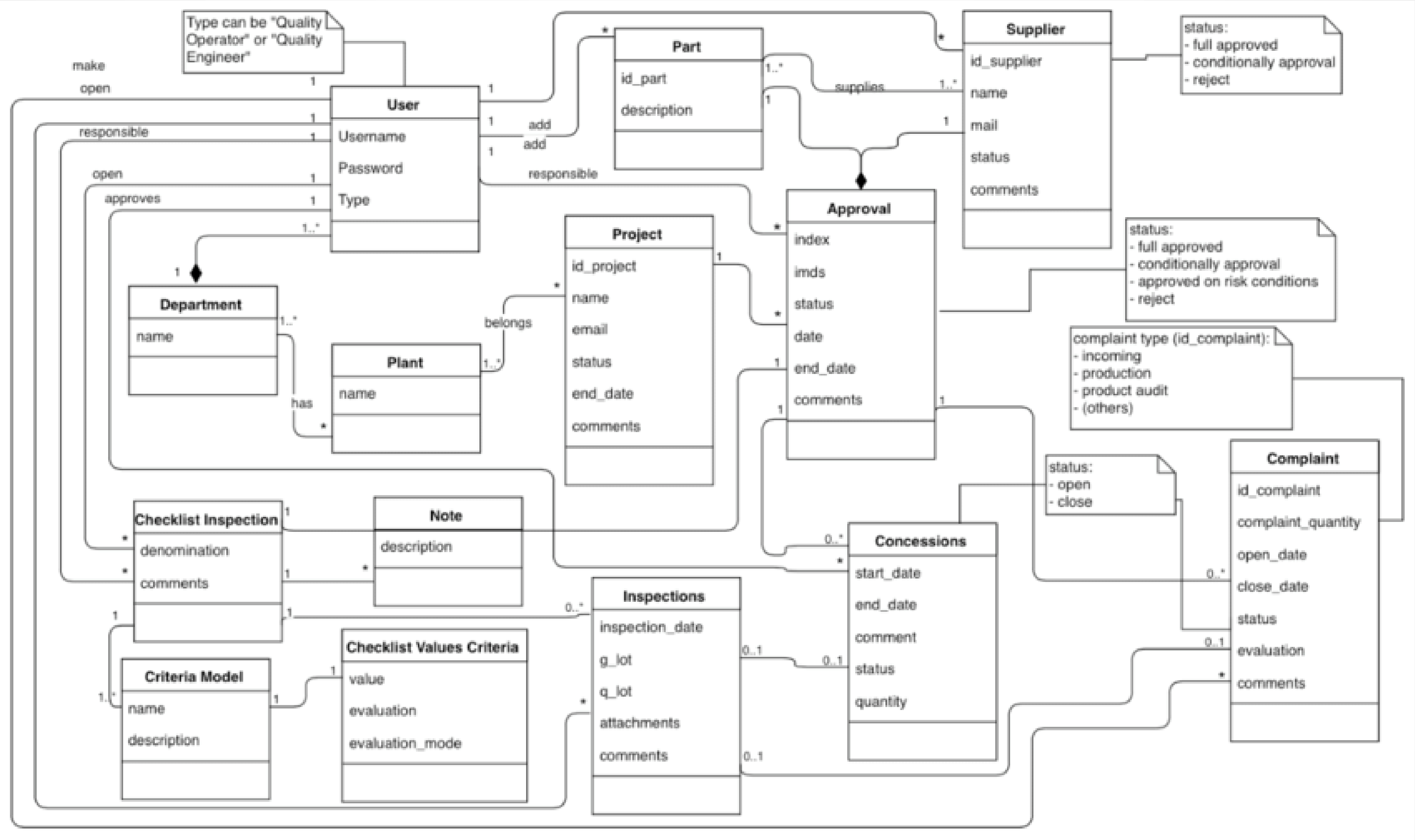
Sharding e replicação com Citus





```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
EDIT       BE00004.SQL(SELECT2) - 01.11                    Columns 00001 00072
Command ===>                                                  Scroll ===> PAGE
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000002 --*******************************************************
000003 --** SELECT all rows & columns FROM table WC.CTYCODE   **
000004 --** This table is a list of 2 digit country codes     **
000005 --*******************************************************
000006 -- SELECT * FROM WC.CTYCODE;
000007 -- SELECT COUNT(*) FROM WC.CTYCODE;
000013 --*******************************************************
000014 --** SELECT all rows & columns FROM table WC.CURRENCY  **
000015 --** This table is a list of world currencies          **
000016 --*******************************************************
000017 -- SELECT * FROM WC.CURRENCY
000018 -- SELECT COUNT(*) FROM WC.CURRENCY;
000022 --*******************************************************
000025 --** SELECT all rows & columns FROM table WC.UNIV      **
000026 --** This table is a list of universities world wide   **
000027 --*******************************************************
000028 -- SELECT * FROM WC.UNIV;
000029 -- SELECT COUNT(*) FROM WC.UNIV;
000033 --*******************************************************
000036 --** Use SQL to list a university from a string that   **
000037 --** includes all or part of the university name.      **
000038 --** The record also includes 2 digit Country Code     **
000039 --** of the university                                 **
000040 --*******************************************************
000041 -- SELECT *        FROM WC.UNIV
000042      WHERE UNIVERSITY LIKE
```

Relational Model



Type can be "Quality Operator" or "Quality Engineer"

User
Username
Password
Type

Part
id_part
description

Supplier
id_supplier
name
mail
status
comments

status:
- full approved
- conditionally approval
- reject

Approval
index
imds
status
date
end_date
comments

status:
- full approved
- conditionally approval
- approved on risk conditions
- reject

Project
id_project
name
email
status
end_date
comments

Department
name

Plant
name

complaint type (id_complaint):
- incoming
- production
- product audit
- (others)

status:
- open
- close

Complaint
id_complaint
complaint_quantity
open_date
close_date
status
evaluation
comments

Checklist Inspection
denomination
comments

Note
description

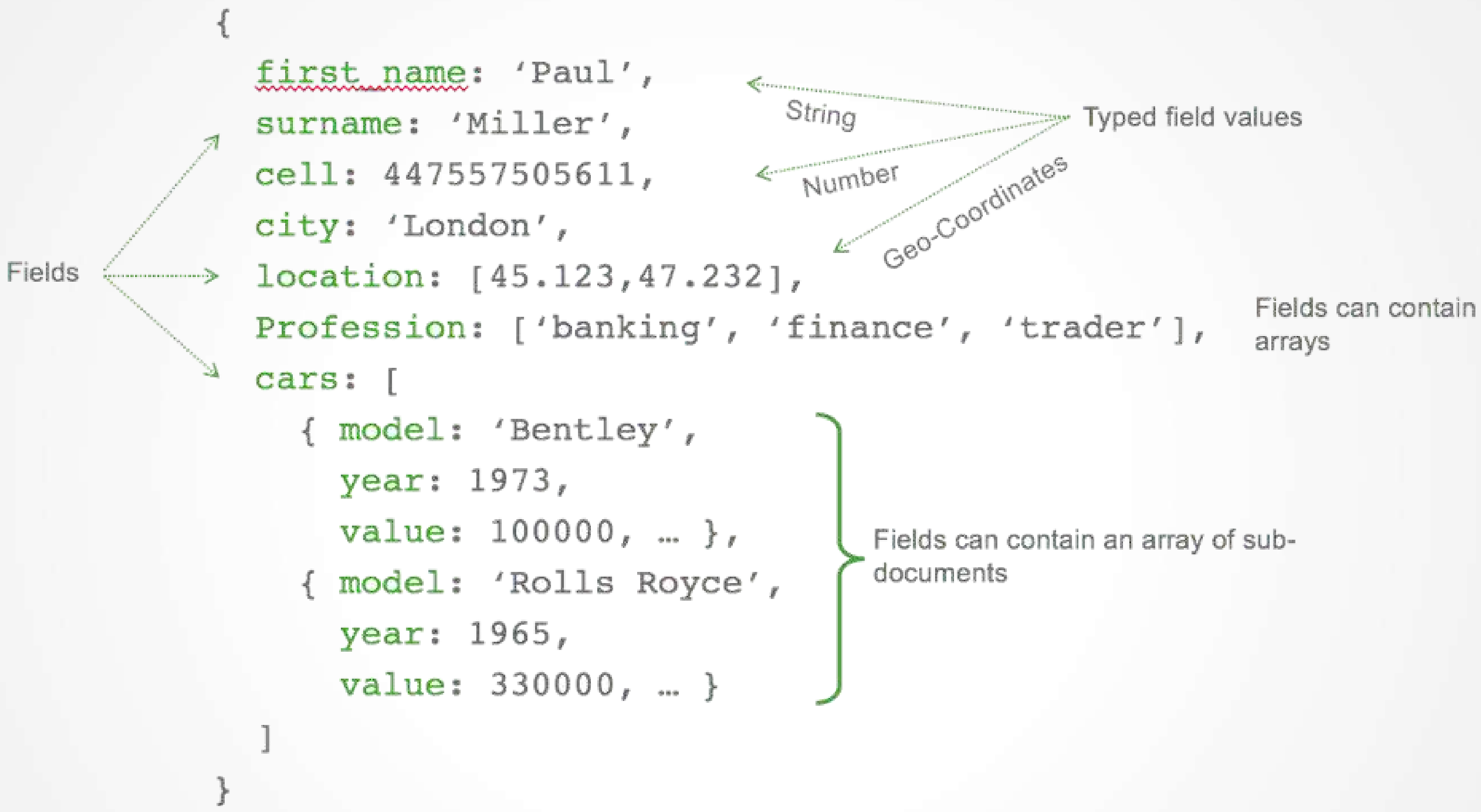
Concessions
start_date
end_date
comment
status
quantity

Inspections
inspection_date
g_lot
q_lot
attachments
comments

Criteria Model
name
description

Checklist Values Criteria
value
evaluation
evaluation_mode

Document Store



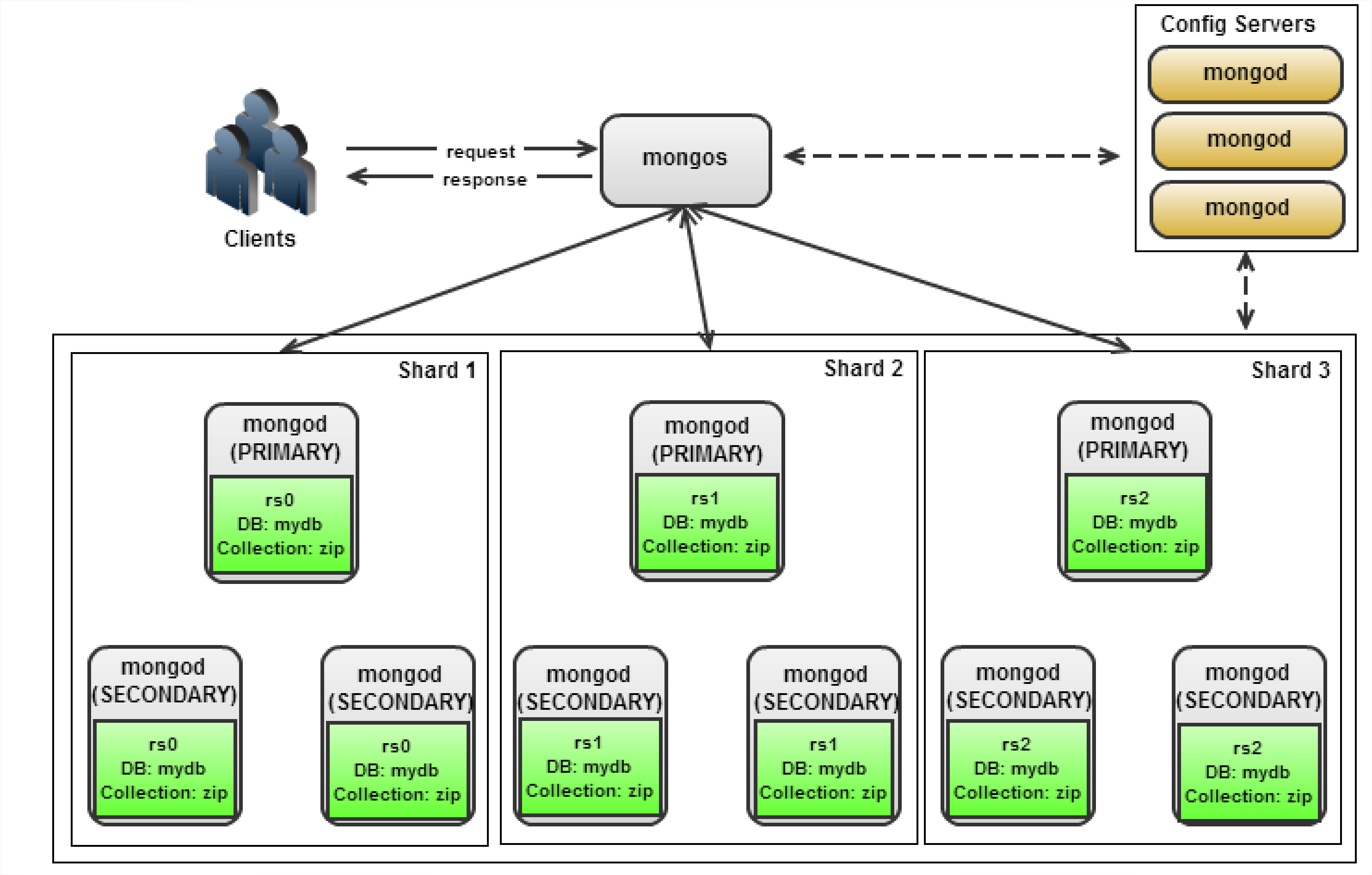
```
{
  first_name: ‘Paul’,
  surname: ‘Miller’,
  cell: 447557505611,
  city: ‘London’,
  location: [45.123,47.232],
  Profession: [‘banking’, ‘finance’, ‘trader’],
  cars: [
    { model: ‘Bentley’,
      year: 1973,
      value: 100000, … },
    { model: ‘Rolls Royce’,
      year: 1965,
      value: 330000, … }
  ]
}
```

Fields

Typed field values

Fields can contain arrays

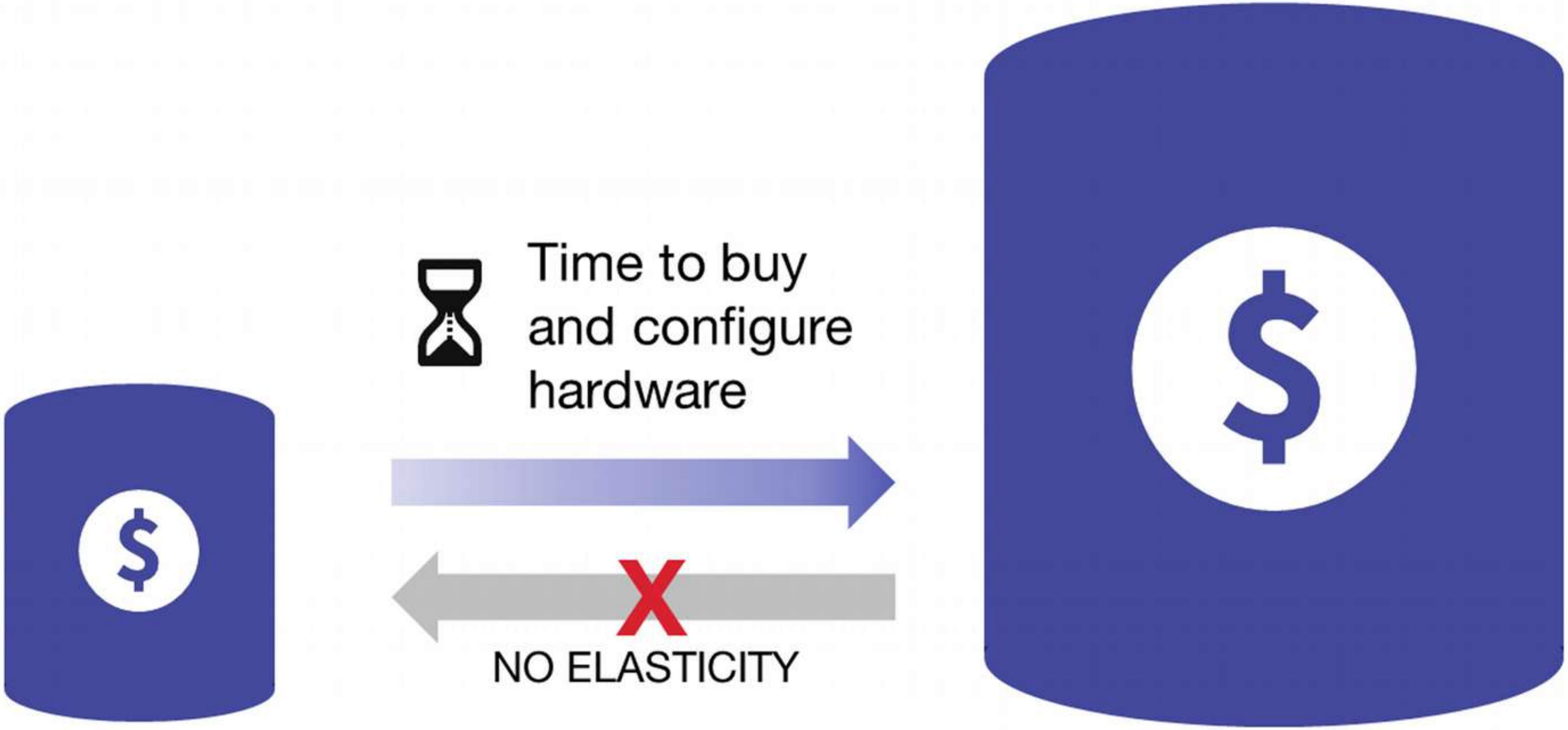
Fields can contain an array of sub-documents


Config Servers
mongod
mongod
mongod
request
response
mongos
Clients
Shard 1
Shard 2
Shard 3
mongod (PRIMARY)
rs0 DB: mydb Collection: zip
mongod (SECONDARY)
rs0 DB: mydb Collection: zip
mongod (SECONDARY)
rs0 DB: mydb Collection: zip
mongod (PRIMARY)
rs1 DB: mydb Collection: zip
mongod (SECONDARY)
rs1 DB: mydb Collection: zip
mongod (SECONDARY)
rs1 DB: mydb Collection: zip
mongod (PRIMARY)
rs2 DB: mydb Collection: zip
mongod (SECONDARY)
rs2 DB: mydb Collection: zip
mongod (SECONDARY)
rs2 DB: mydb Collection: zip

“RDBMS não escalam“

- ➔Single-server
- ➔Escala apenas verticalmente
- ➔Flexibilidade X Performance
- ➔Disponibilidade

Scaling a Relational Database



Expensive single servers

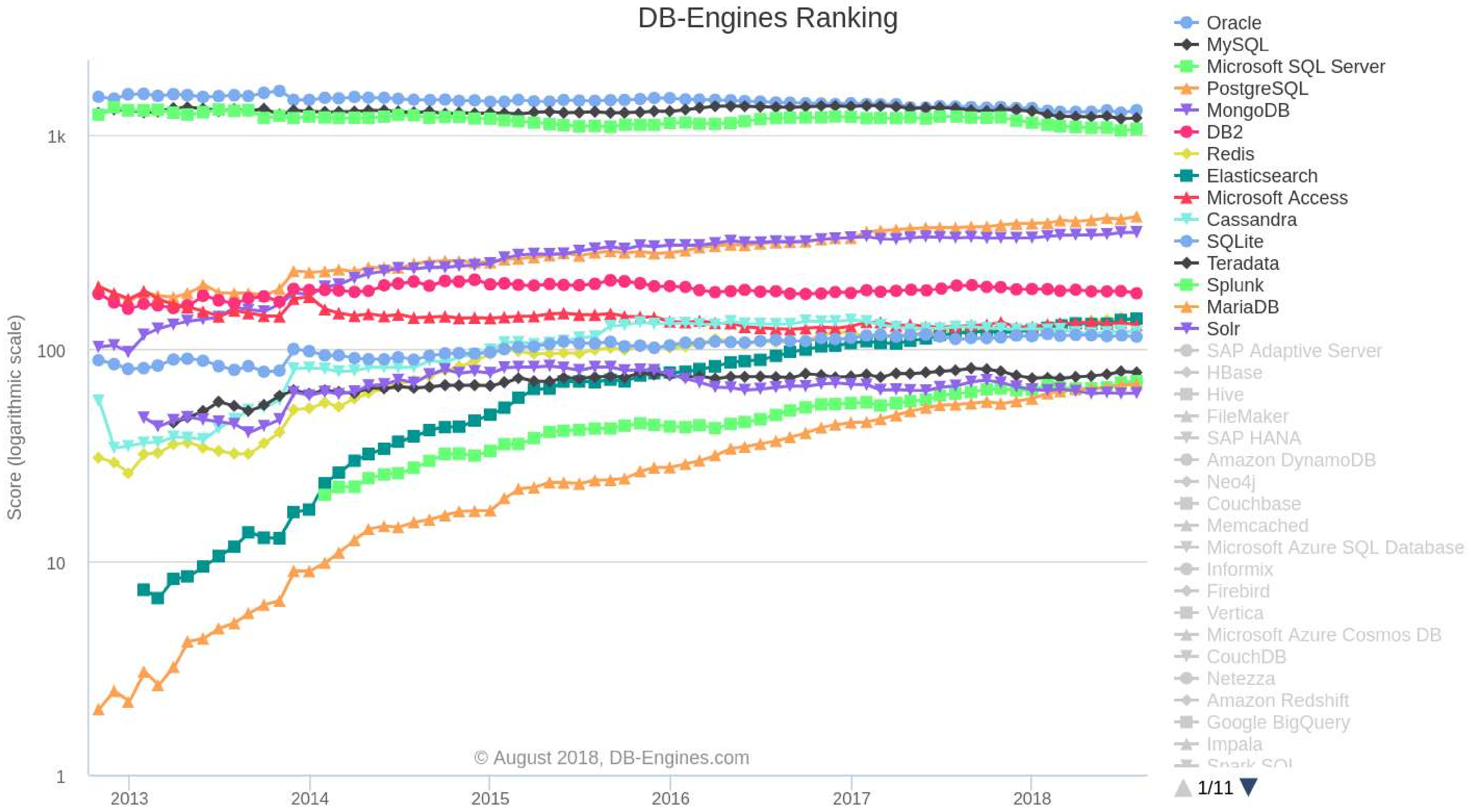
Time to buy and configure hardware

NO ELASTICITY

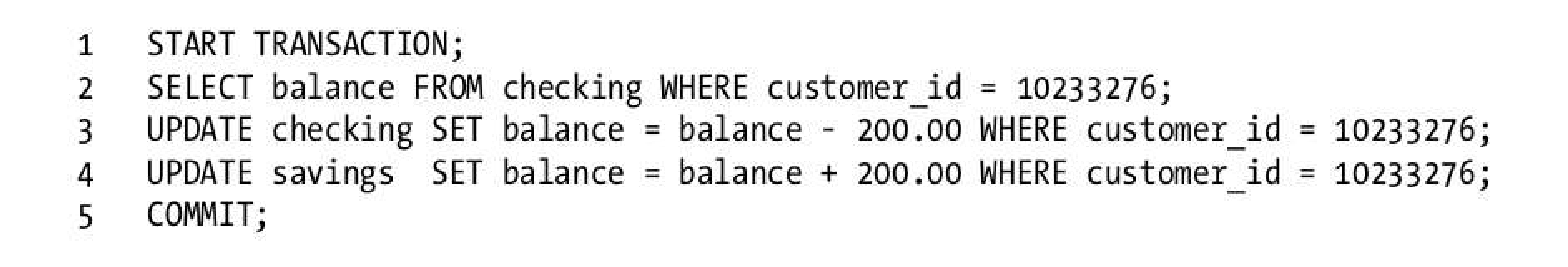
MBs

GBs

NoSQL cresceu!

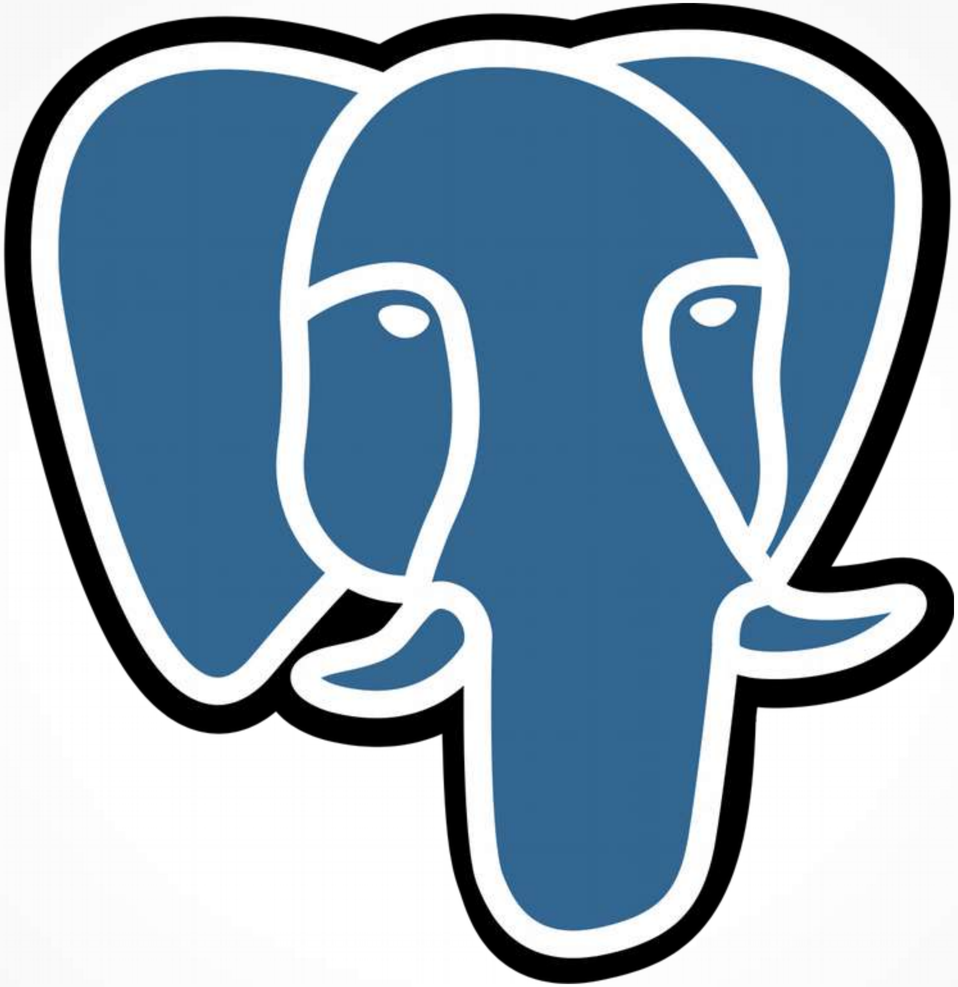


ACID



```sql
START TRANSACTION;
SELECT balance FROM checking WHERE customer_id = 10233276;
UPDATE checking SET balance = balance - 200.00 WHERE customer_id = 10233276;
UPDATE savings  SET balance = balance + 200.00 WHERE customer_id = 10233276;
COMMIT;
```

PostgreSQL: Melhorias



PostgreSQL: Melhorias

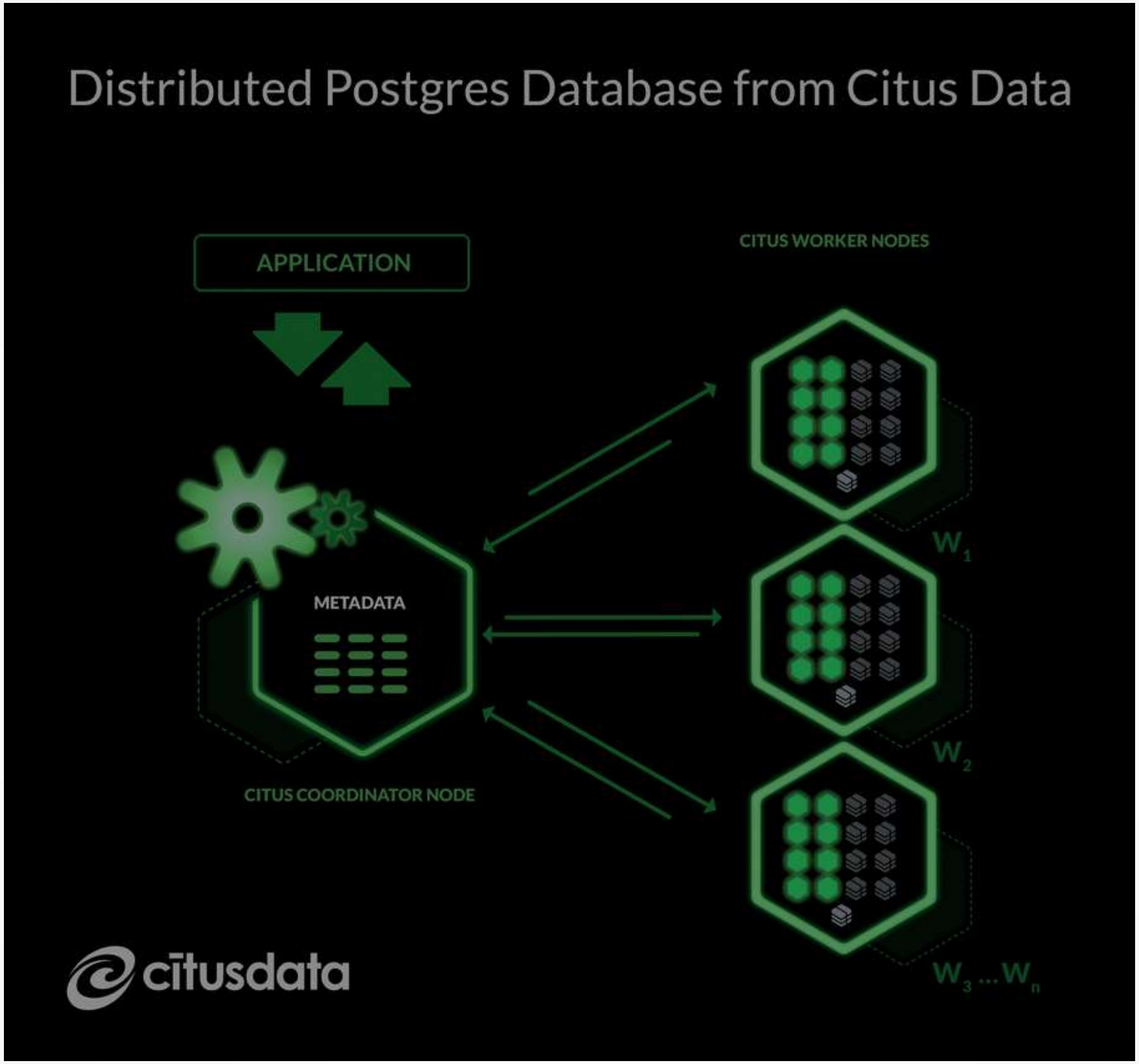
- ➔Particionamento nativo
- ➔FDW
- ➔FTS para JSON e JSONB
- ➔Queries paralelas
- ➔Quorum Commit

PostgreSQL + Vertical scaling

Como aliar PostgreSQL à necessidade de escalar horizontalmente?

citusdata

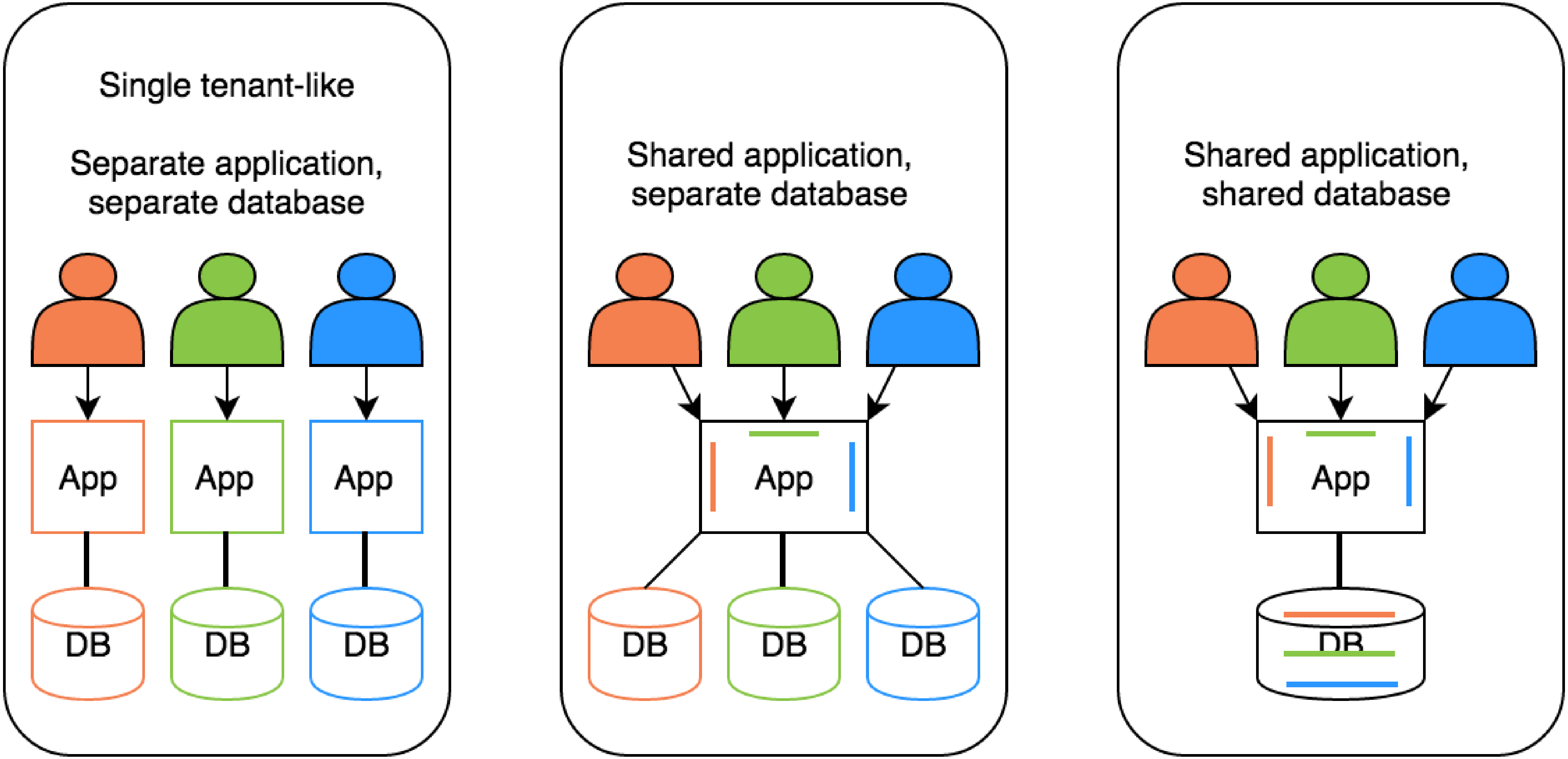
Citus é uma extensão!



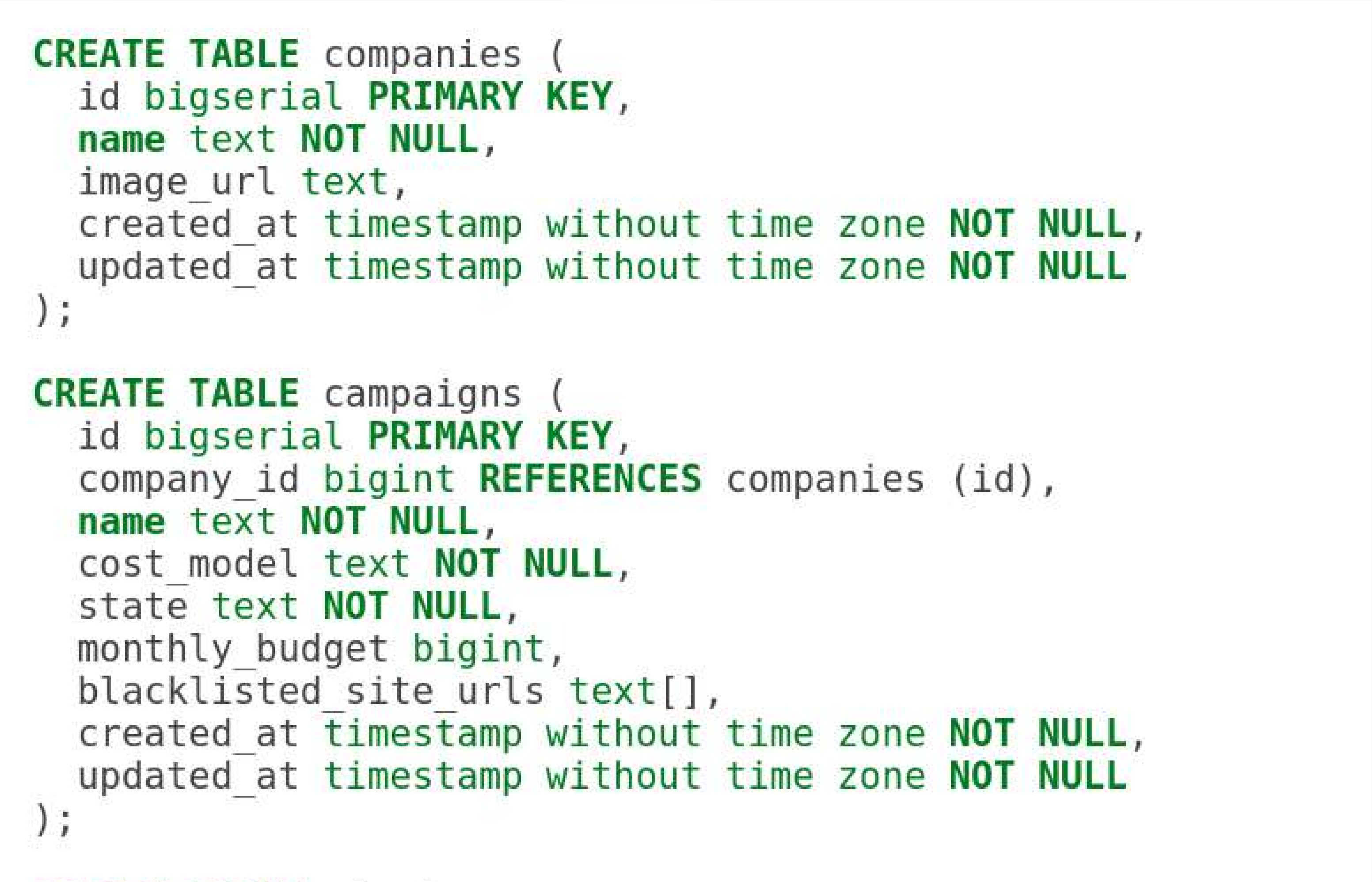
Problemas

- Multi-tenancy
- Real-time analytics

Multi-tenancy

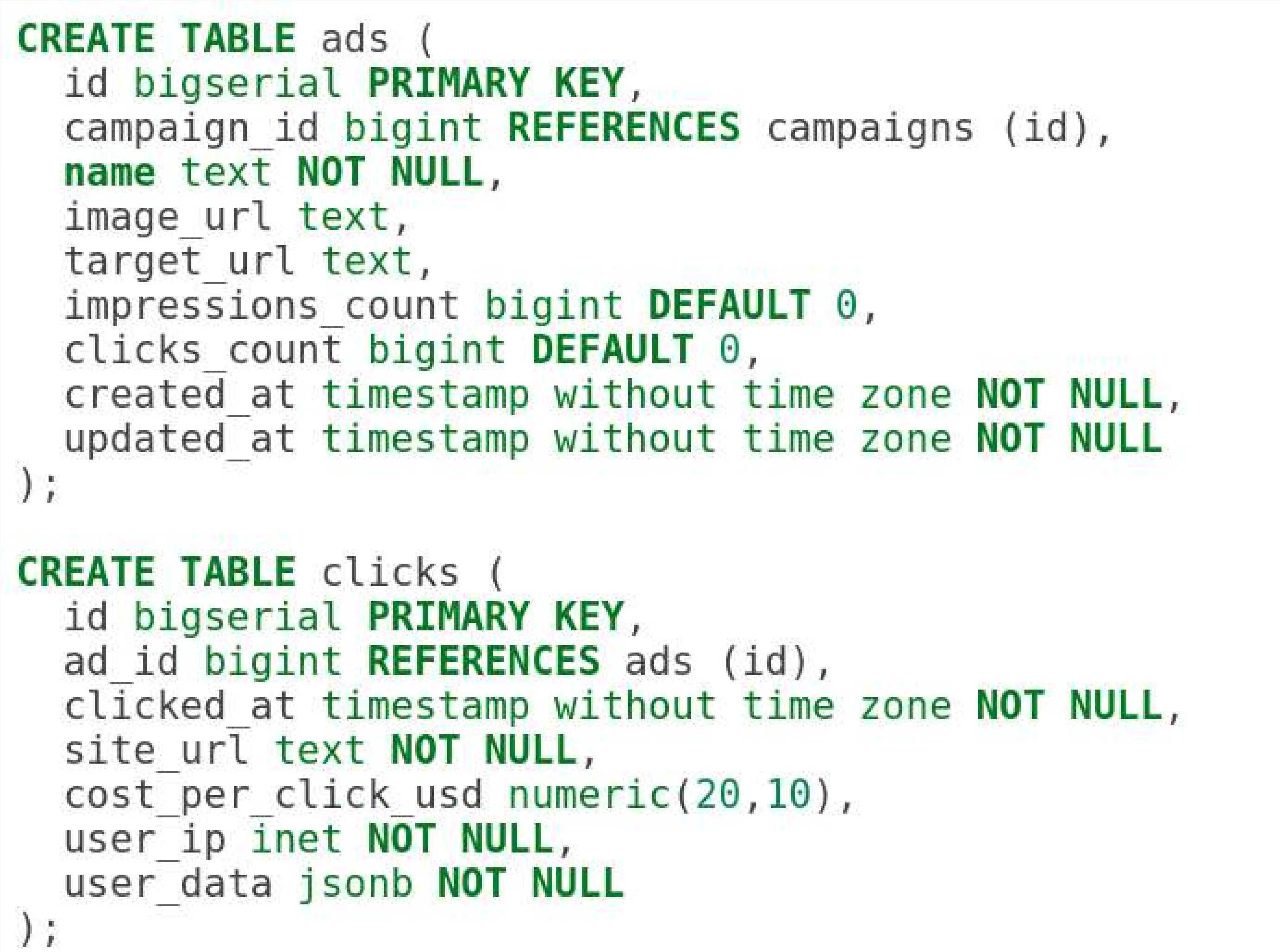

Single tenant-like
Separate application, separate database
App
App
App
DB
DB
DB
Shared application, separate database
App
DB
DB
DB
Shared application, shared database
App
DB

Ad Analytics



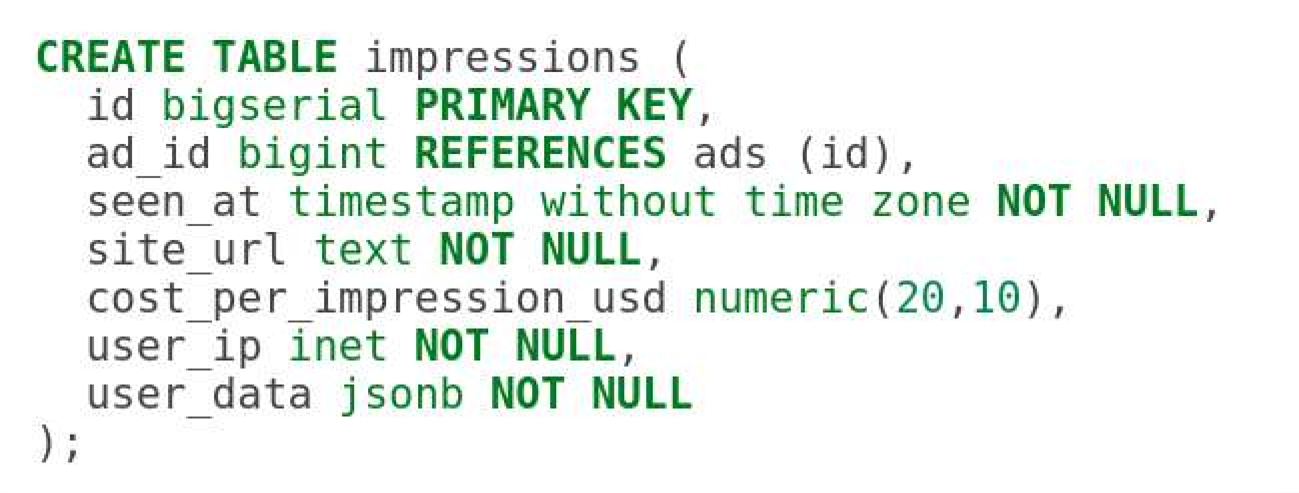
```
CREATE TABLE companies (
  id bigserial PRIMARY KEY,
  name text NOT NULL,
  image_url text,
  created_at timestamp without time zone NOT NULL,
  updated_at timestamp without time zone NOT NULL
);

CREATE TABLE campaigns (
  id bigserial PRIMARY KEY,
  company_id bigint REFERENCES companies (id),
  name text NOT NULL,
  cost_model text NOT NULL,
  state text NOT NULL,
  monthly_budget bigint,
  blacklisted_site_urls text[],
  created_at timestamp without time zone NOT NULL,
  updated_at timestamp without time zone NOT NULL
);
```

Ad Analytics



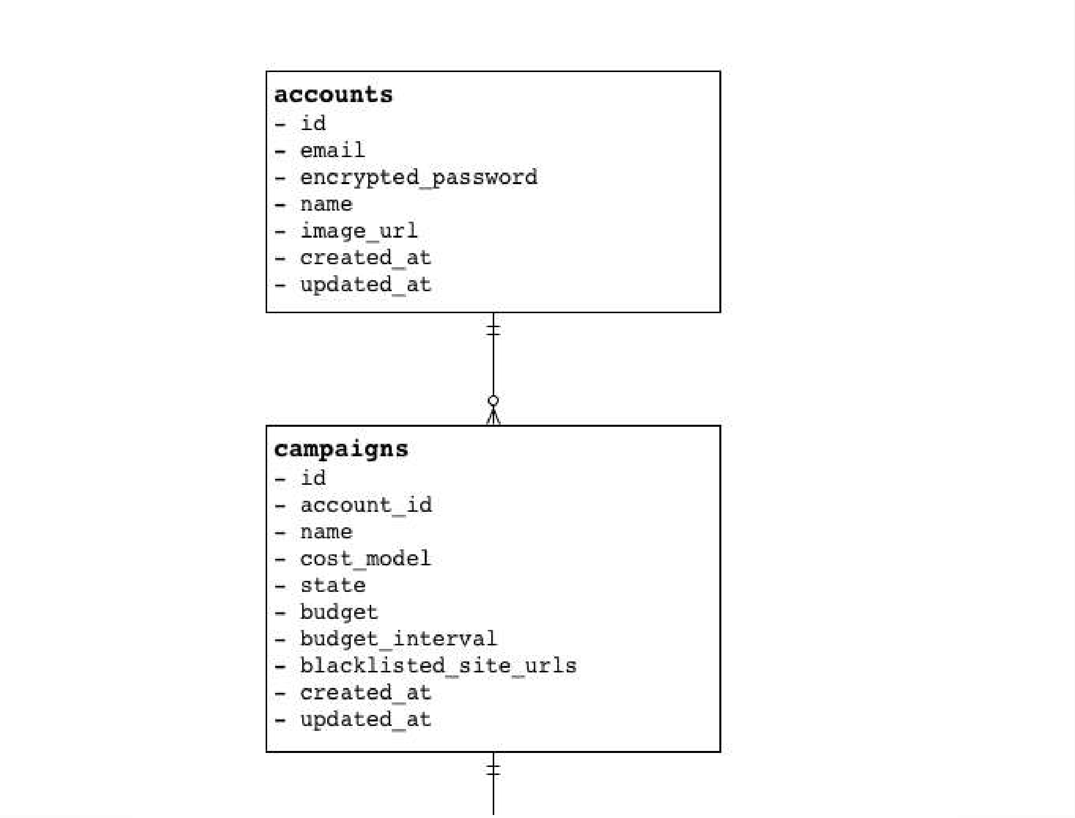
```
CREATE TABLE ads (
  id bigserial PRIMARY KEY,
  campaign_id bigint REFERENCES campaigns (id),
  name text NOT NULL,
  image_url text,
  target_url text,
  impressions_count bigint DEFAULT 0,
  clicks_count bigint DEFAULT 0,
  created_at timestamp without time zone NOT NULL,
  updated_at timestamp without time zone NOT NULL
);

CREATE TABLE clicks (
  id bigserial PRIMARY KEY,
  ad_id bigint REFERENCES ads (id),
  clicked_at timestamp without time zone NOT NULL,
  site_url text NOT NULL,
  cost_per_click_usd numeric(20,10),
  user_ip inet NOT NULL,
  user_data jsonb NOT NULL
);
```

Ad Analytics



```
CREATE TABLE impressions (
  id bigserial PRIMARY KEY,
  ad_id bigint REFERENCES ads (id),
  seen_at timestamp without time zone NOT NULL,
  site_url text NOT NULL,
  cost_per_impression_usd numeric(20,10),
  user_ip inet NOT NULL,
  user_data jsonb NOT NULL
);
```

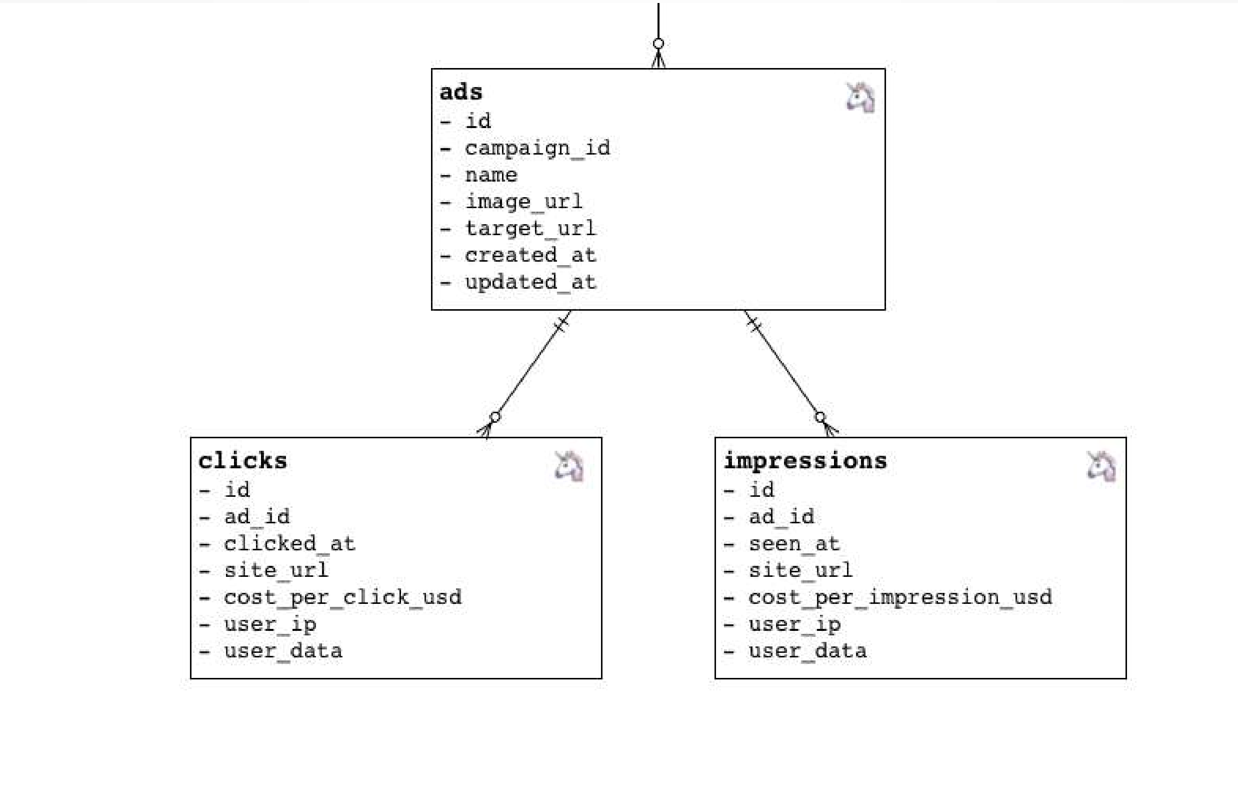
Ad Analytics



accounts
- id
- email
- encrypted_password
- name
- image_url
- created_at
- updated_at

campaigns
- id
- account_id
- name
- cost_model
- state
- budget
- budget_interval
- blacklisted_site_urls
- created_at
- updated_at

Ad Analytics



ads

- id
- campaign_id
- name
- image_url
- target_url
- created_at
- updated_at

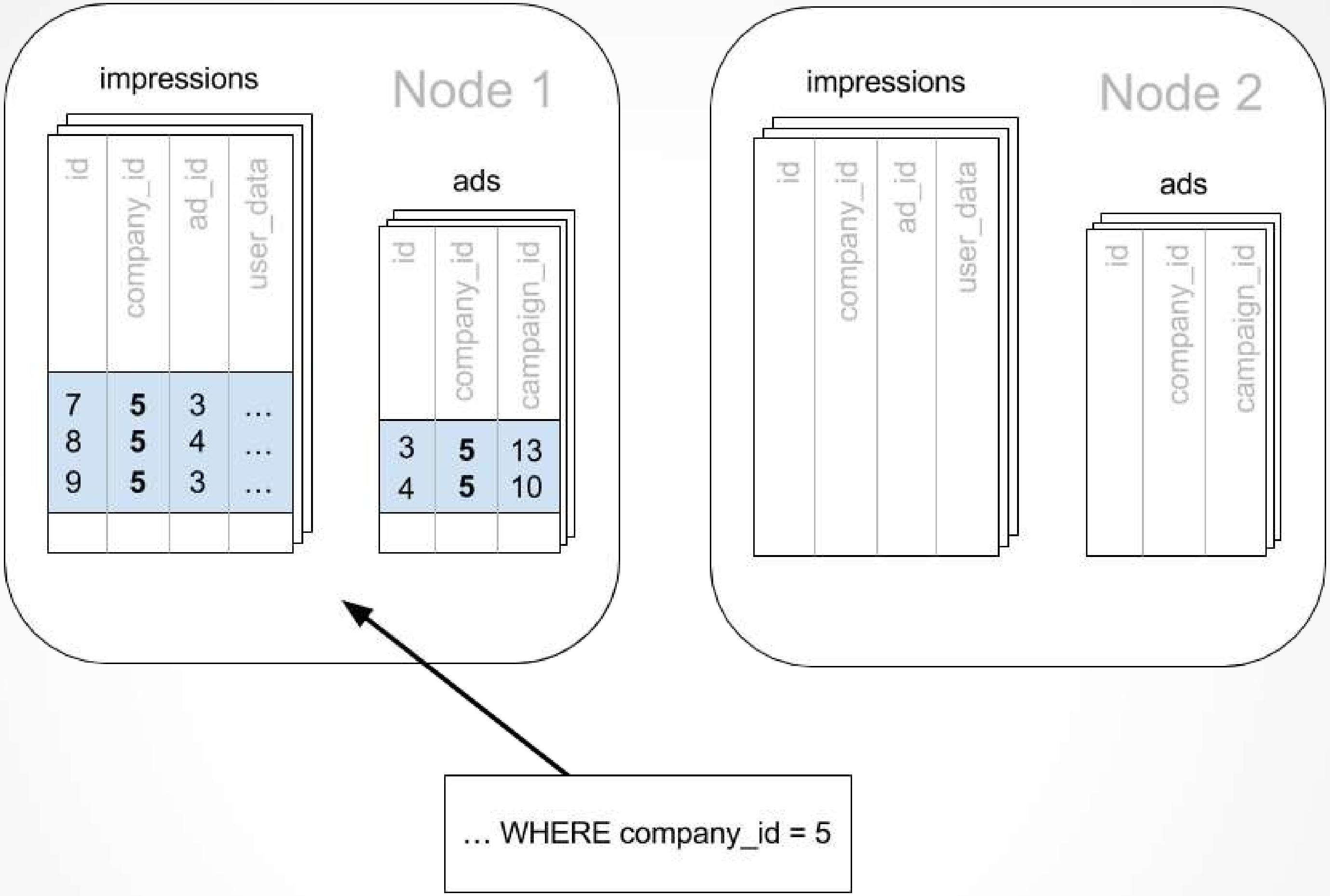
clicks

- id
- ad_id
- clicked_at
- site_url
- cost_per_click_usd
- user_ip
- user_data

impressions

- id
- ad_id
- seen_at
- site_url
- cost_per_impression_usd
- user_ip
- user_data

Agrupe os dados!



impressions

id	company_id	ad_id	user_data
7	**5**	3	…
8	**5**	4	…
9	**5**	3	…

ads

id	company_id	campaign_id
3	**5**	13
4	**5**	10

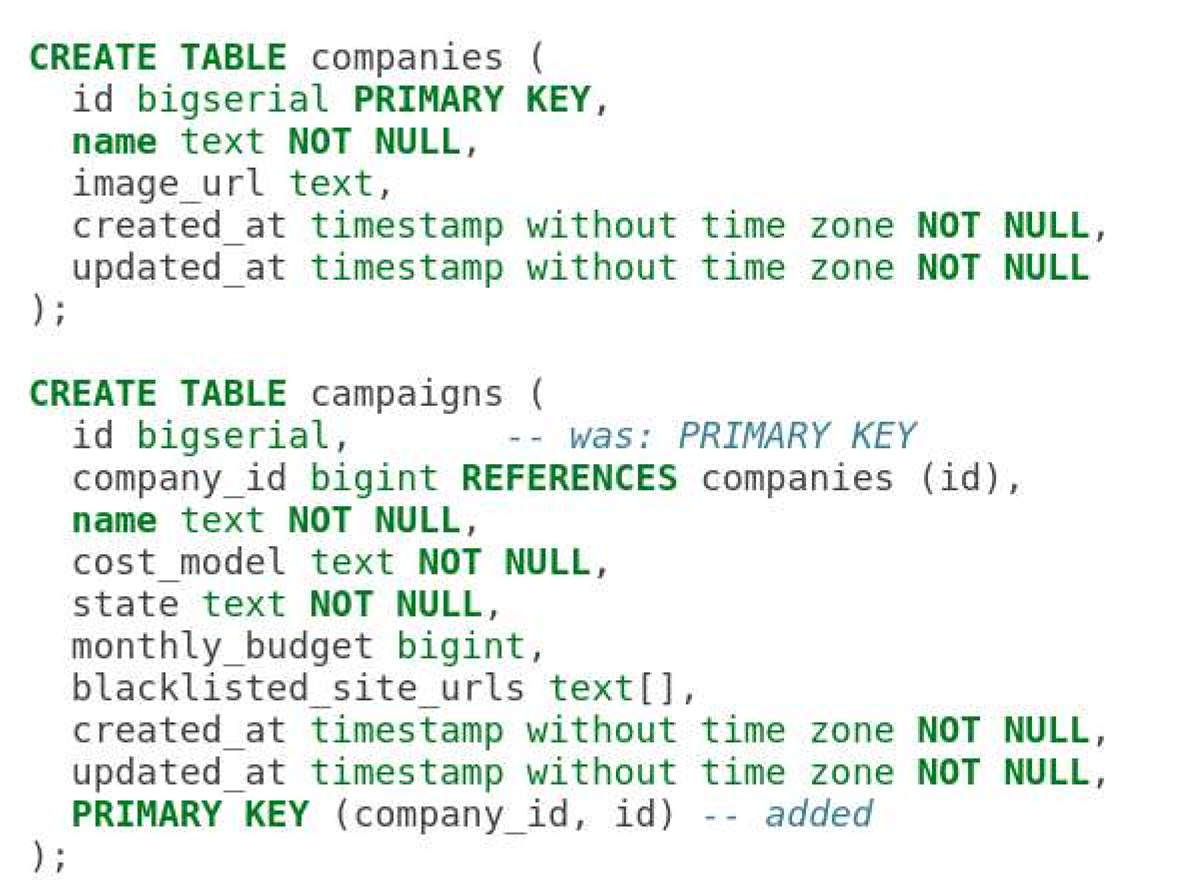
impressions

id	company_id	ad_id	user_data

ads

id	company_id	campaign_id

… WHERE company_id = 5

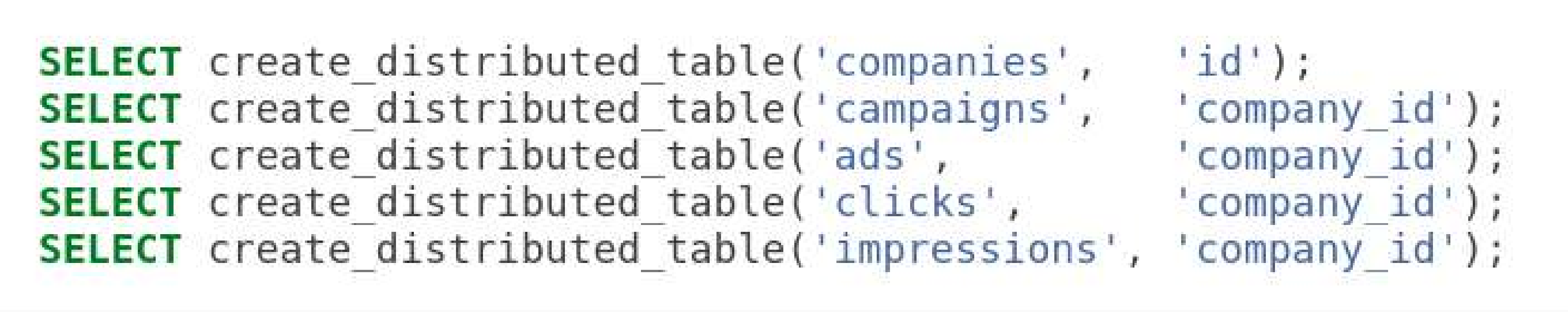


```sql
CREATE TABLE companies (
  id bigserial PRIMARY KEY,
  name text NOT NULL,
  image_url text,
  created_at timestamp without time zone NOT NULL,
  updated_at timestamp without time zone NOT NULL
);

CREATE TABLE campaigns (
  id bigserial,       -- was: PRIMARY KEY
  company_id bigint REFERENCES companies (id),
  name text NOT NULL,
  cost_model text NOT NULL,
  state text NOT NULL,
  monthly_budget bigint,
  blacklisted_site_urls text[],
  created_at timestamp without time zone NOT NULL,
  updated_at timestamp without time zone NOT NULL,
  PRIMARY KEY (company_id, id) -- added
);
```

```
CREATE TABLE ads (
  id bigserial,       -- was: PRIMARY KEY
  company_id bigint,  -- added
  campaign_id bigint, -- was: REFERENCES campaigns (id)
  name text NOT NULL,
  image_url text,
  target_url text,
  impressions_count bigint DEFAULT 0,
  clicks_count bigint DEFAULT 0,
  created_at timestamp without time zone NOT NULL,
  updated_at timestamp without time zone NOT NULL,
  PRIMARY KEY (company_id, id),         -- added
  FOREIGN KEY (company_id, campaign_id) -- added
    REFERENCES campaigns (company_id, id)
);
```

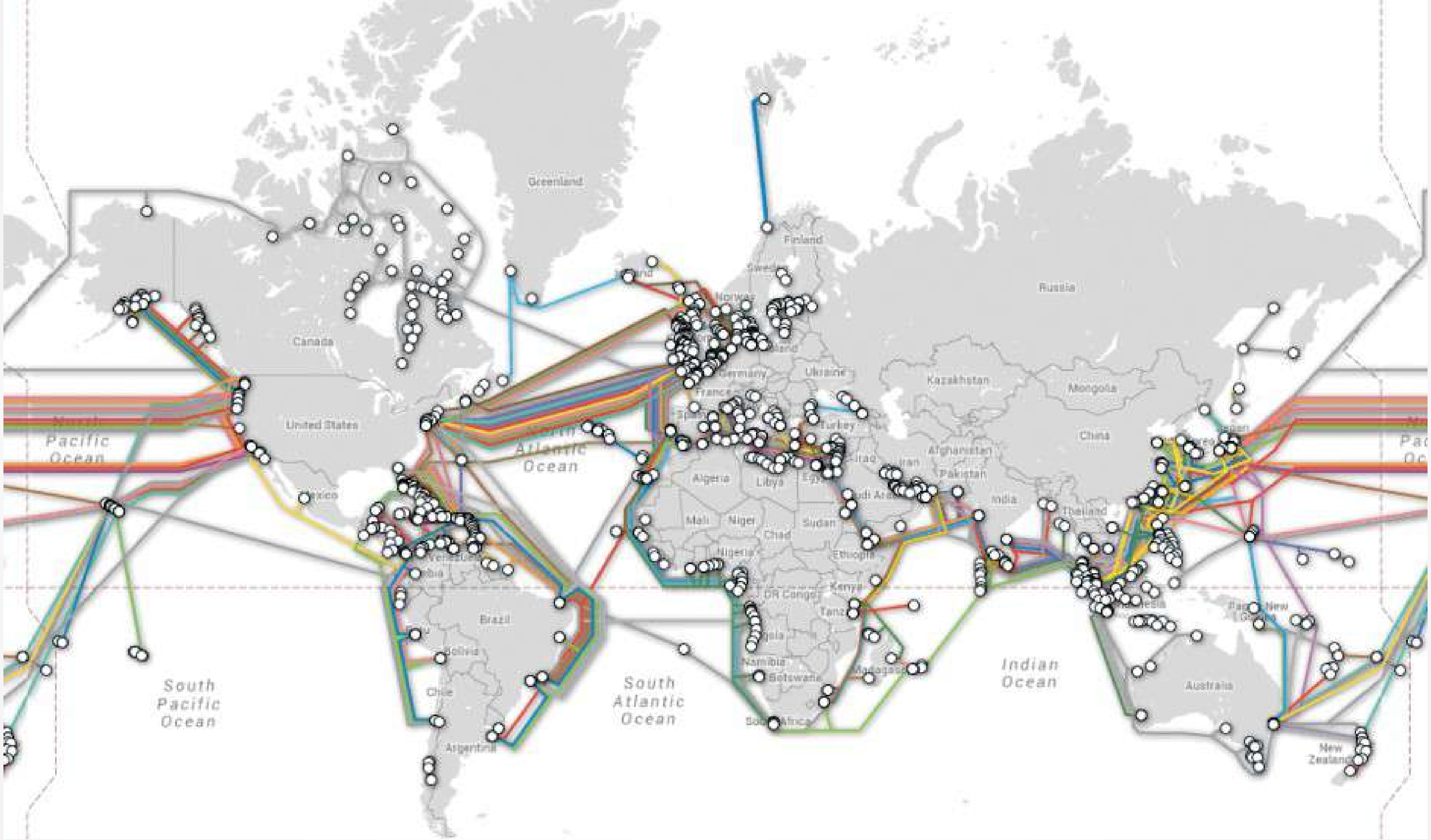
```sql
CREATE TABLE clicks (
  id bigserial,        -- was: PRIMARY KEY
  company_id bigint,   -- added
  ad_id bigint,        -- was: REFERENCES ads (id),
  clicked_at timestamp without time zone NOT NULL,
  site_url text NOT NULL,
  cost_per_click_usd numeric(20,10),
  user_ip inet NOT NULL,
  user_data jsonb NOT NULL,
  PRIMARY KEY (company_id, id),     -- added
  FOREIGN KEY (company_id, ad_id)    -- added
    REFERENCES ads (company_id, id)
);

CREATE TABLE impressions (
  id bigserial,         -- was: PRIMARY KEY
  company_id bigint,    -- added
  ad_id bigint,         -- was: REFERENCES ads (id),
  seen_at timestamp without time zone NOT NULL,
  site_url text NOT NULL,
  cost_per_impression_usd numeric(20,10),
  user_ip inet NOT NULL,
  user_data jsonb NOT NULL,
  PRIMARY KEY (company_id, id),      -- added
  FOREIGN KEY (company_id, ad_id)     -- added
    REFERENCES ads (company_id, id)
);
```

Distributed tables

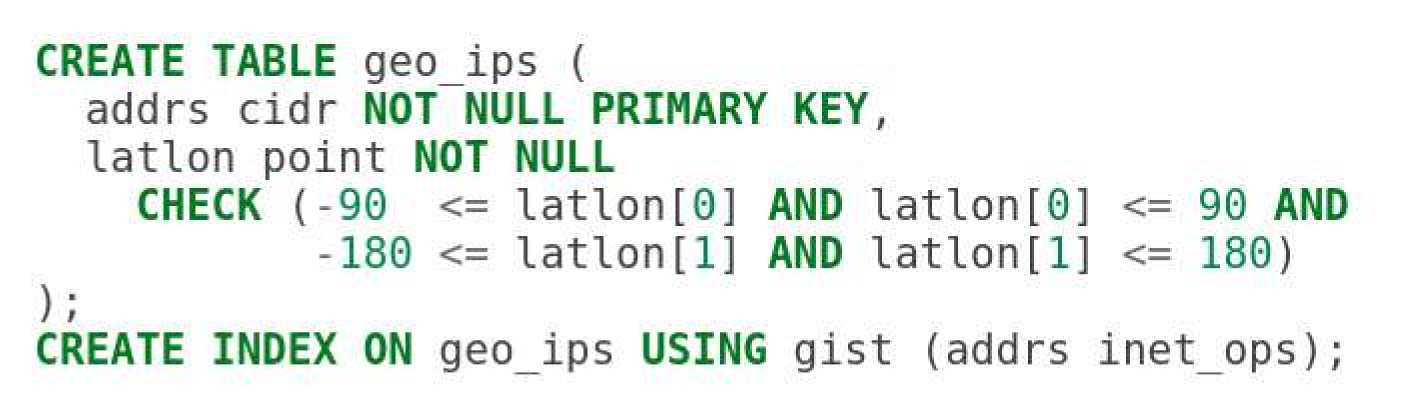


```sql
SELECT create_distributed_table('companies',   'id');
SELECT create_distributed_table('campaigns',   'company_id');
SELECT create_distributed_table('ads',         'company_id');
SELECT create_distributed_table('clicks',      'company_id');
SELECT create_distributed_table('impressions', 'company_id');
```

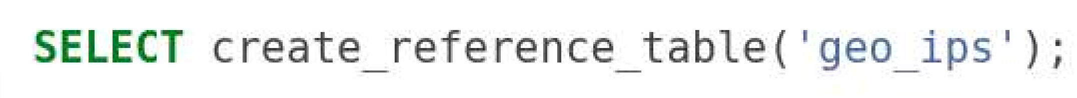
Reference tables



Reference tables

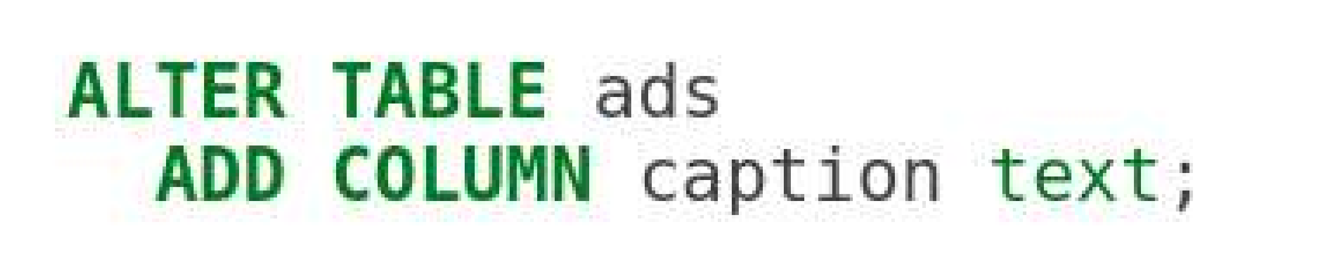


```
CREATE TABLE geo_ips (
  addrs cidr NOT NULL PRIMARY KEY,
  latlon point NOT NULL
    CHECK (-90  <= latlon[0] AND latlon[0] <= 90 AND
           -180 <= latlon[1] AND latlon[1] <= 180)
);
CREATE INDEX ON geo_ips USING gist (addrs inet_ops);
```



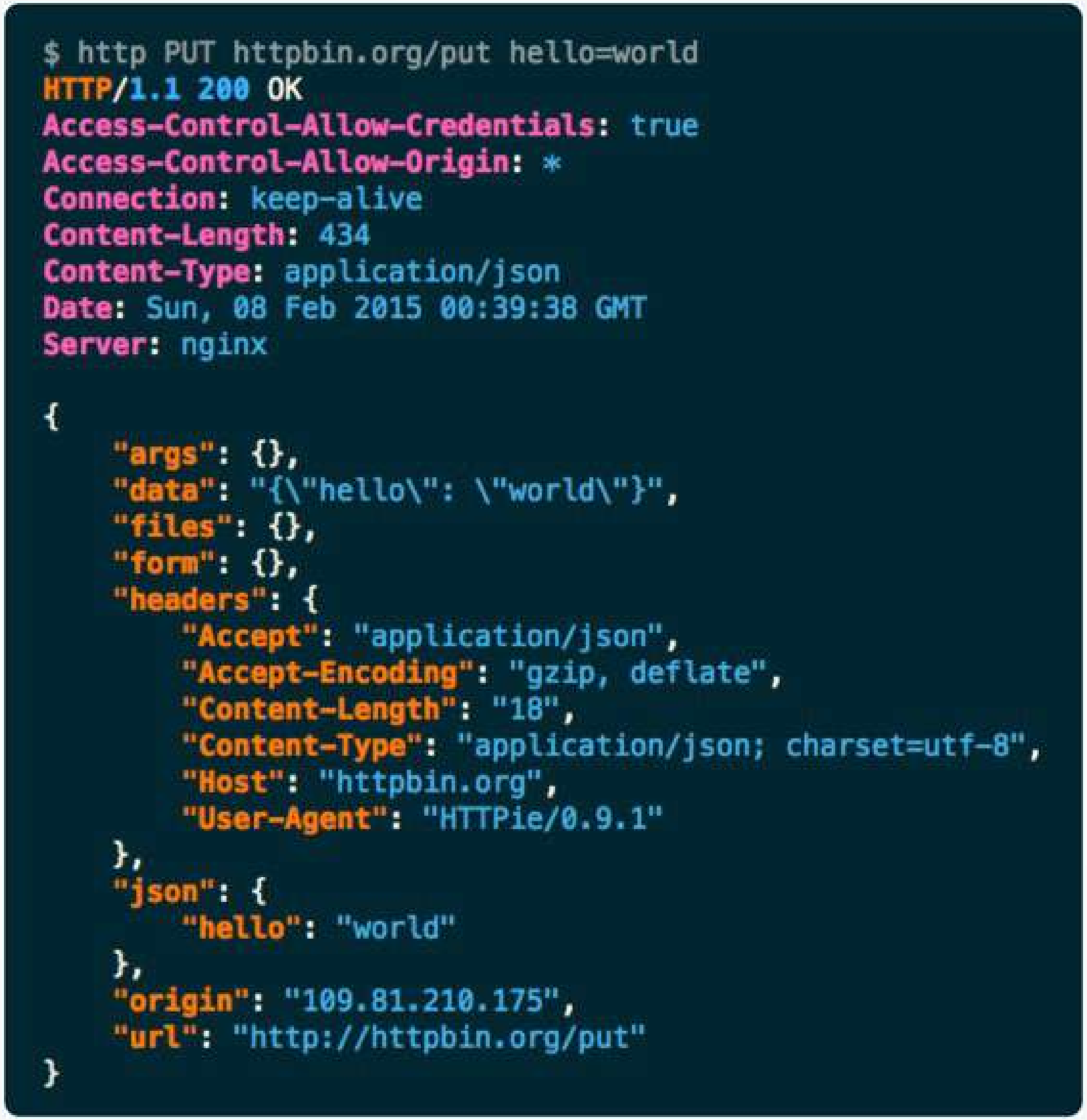
```
SELECT create_reference_table('geo_ips');
```

DDLs



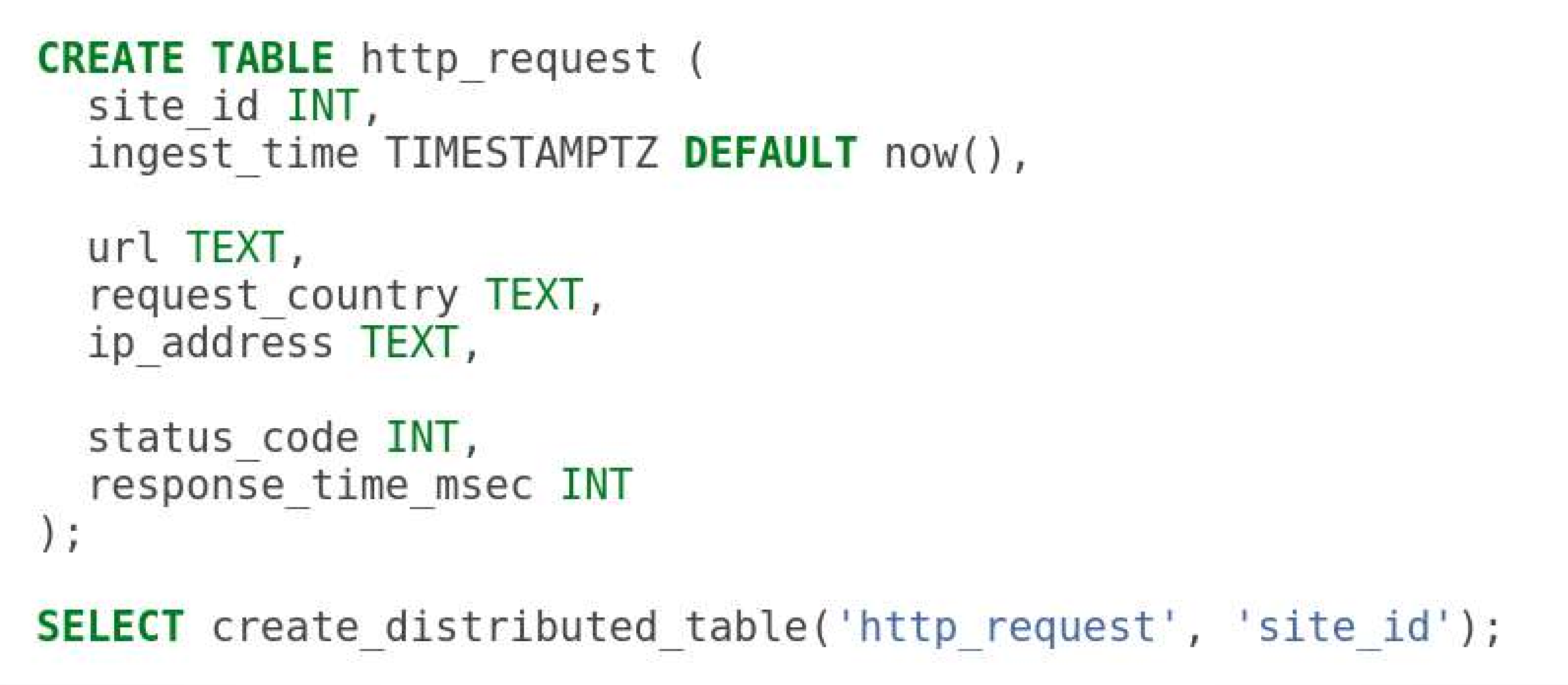
```
ALTER TABLE ads
  ADD COLUMN caption text;
```

Real-time analytics



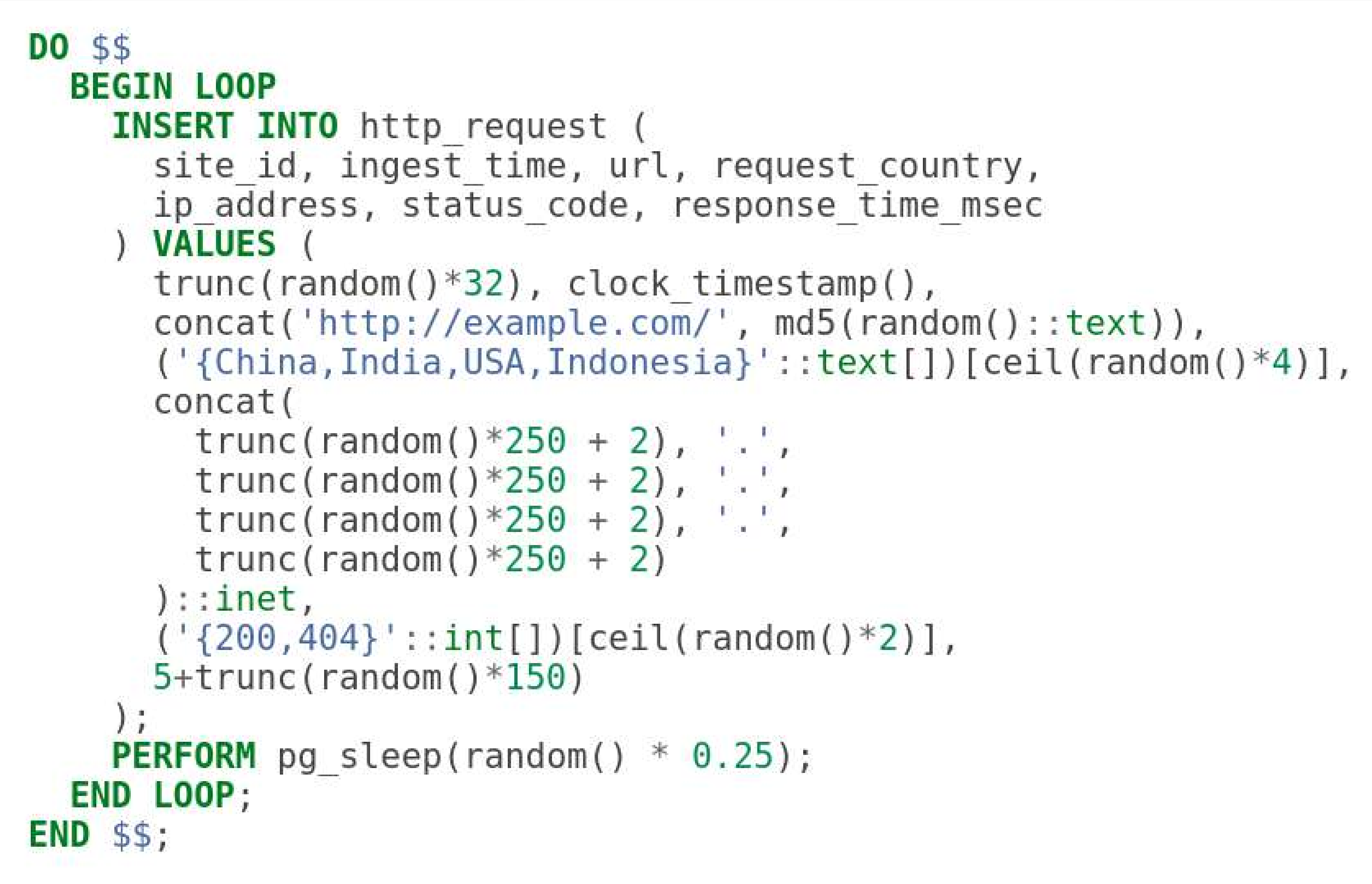
```
$ http PUT httpbin.org/put hello=world
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: *
Connection: keep-alive
Content-Length: 434
Content-Type: application/json
Date: Sun, 08 Feb 2015 00:39:38 GMT
Server: nginx

{
    "args": {},
    "data": "{\"hello\": \"world\"}",
    "files": {},
    "form": {},
    "headers": {
        "Accept": "application/json",
        "Accept-Encoding": "gzip, deflate",
        "Content-Length": "18",
        "Content-Type": "application/json; charset=utf-8",
        "Host": "httpbin.org",
        "User-Agent": "HTTPie/0.9.1"
    },
    "json": {
        "hello": "world"
    },
    "origin": "109.81.210.175",
    "url": "http://httpbin.org/put"
}
```

Data model



```
CREATE TABLE http_request (
  site_id INT,
  ingest_time TIMESTAMPTZ DEFAULT now(),

  url TEXT,
  request_country TEXT,
  ip_address TEXT,

  status_code INT,
  response_time_msec INT
);

SELECT create_distributed_table('http_request', 'site_id');
```

Data model

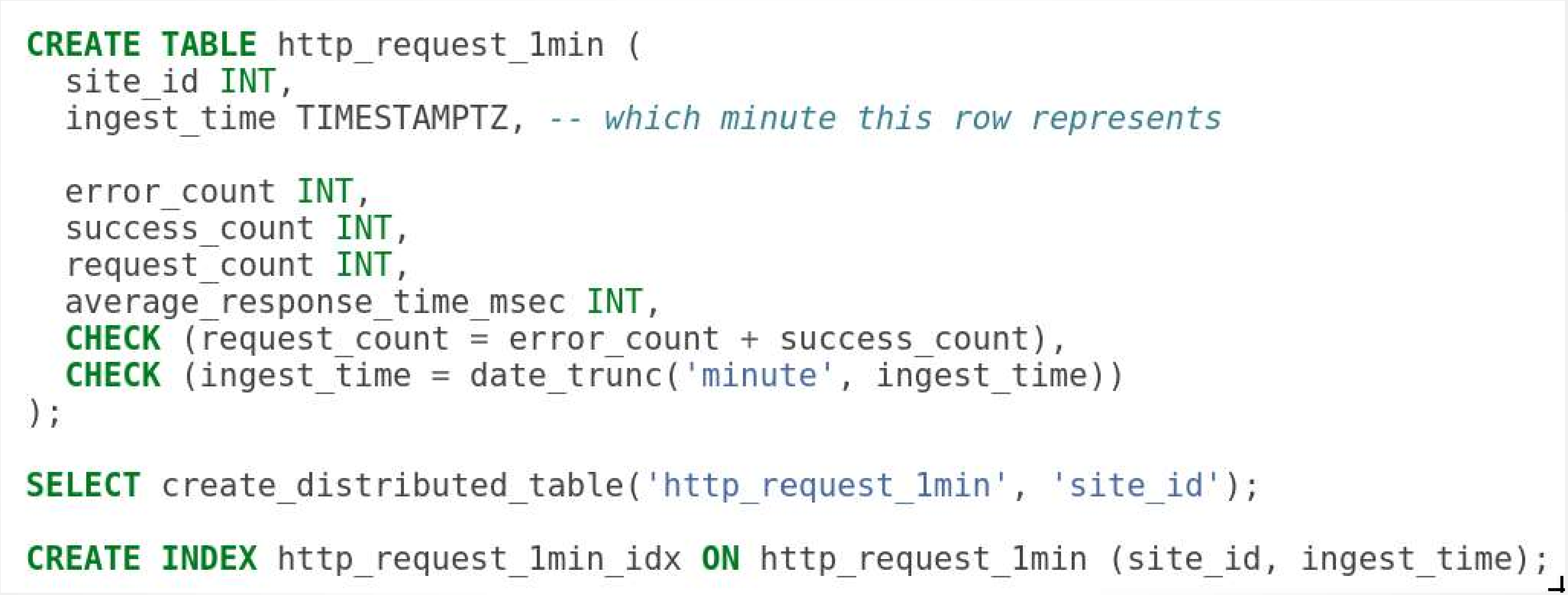


```
DO $$
  BEGIN LOOP
    INSERT INTO http_request (
      site_id, ingest_time, url, request_country,
      ip_address, status_code, response_time_msec
    ) VALUES (
      trunc(random()*32), clock_timestamp(),
      concat('http://example.com/', md5(random()::text)),
      ('{China,India,USA,Indonesia}'::text[])[ceil(random()*4)],
      concat(
        trunc(random()*250 + 2), '.',
        trunc(random()*250 + 2), '.',
        trunc(random()*250 + 2), '.',
        trunc(random()*250 + 2)
      )::inet,
      ('{200,404}'::int[])[ceil(random()*2)],
      5+trunc(random()*150)
    );
    PERFORM pg_sleep(random() * 0.25);
  END LOOP;
END $$;
```

Dashboard query

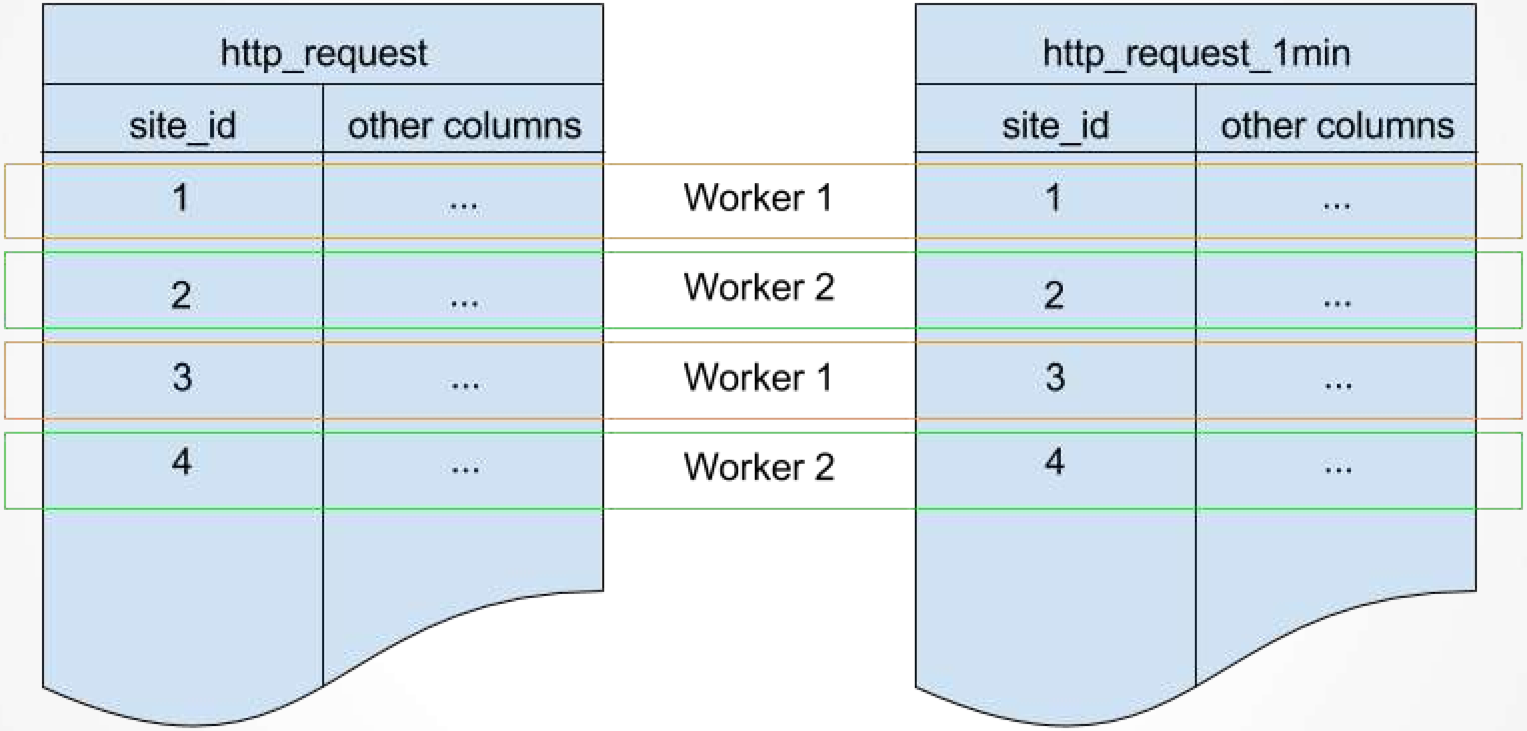
```
SELECT
  site_id,
  date_trunc('minute', ingest_time) as minute,
  COUNT(1) AS request_count,
  SUM(CASE WHEN (status_code between 200 and 299) THEN 1 ELSE 0 END) as success_count,
  SUM(CASE WHEN (status_code between 200 and 299) THEN 0 ELSE 1 END) as error_count,
  SUM(response_time_msec) / COUNT(1) AS average_response_time_msec
FROM http_request
WHERE date_trunc('minute', ingest_time) > now() - '5 minutes'::interval
GROUP BY site_id, minute
ORDER BY minute ASC;
```

ROLLUPS



```
CREATE TABLE http_request_1min (
  site_id INT,
  ingest_time TIMESTAMPTZ, -- which minute this row represents

  error_count INT,
  success_count INT,
  request_count INT,
  average_response_time_msec INT,
  CHECK (request_count = error_count + success_count),
  CHECK (ingest_time = date_trunc('minute', ingest_time))
);

SELECT create_distributed_table('http_request_1min', 'site_id');

CREATE INDEX http_request_1min_idx ON http_request_1min (site_id, ingest_time);
```

Co-location



http_request	
site_id	other columns
1	...
2	...
3	...
4	...

Worker 1

Worker 2

Worker 1

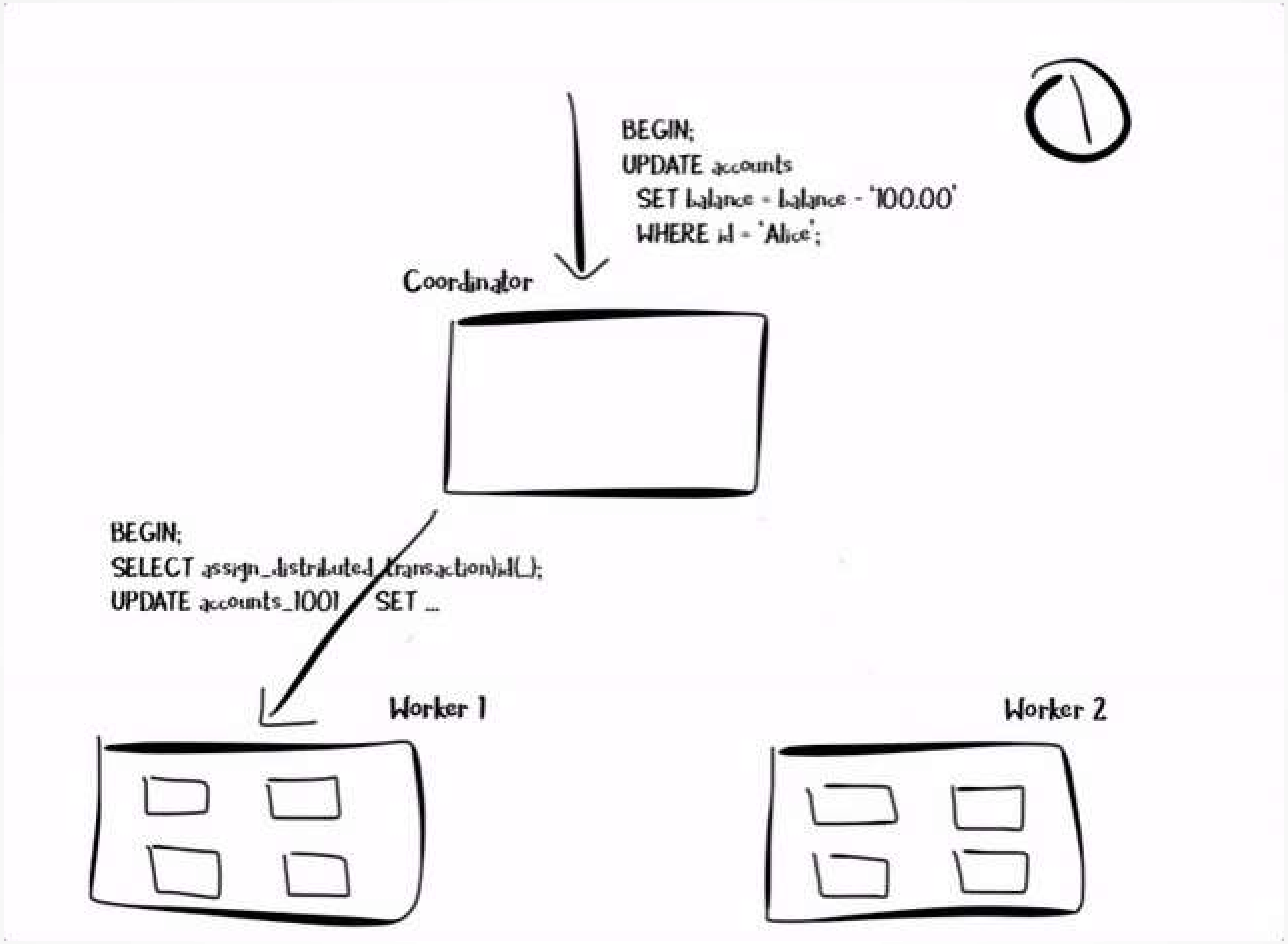
Worker 2

http_request_1min	
site_id	other columns
1	...
2	...
3	...
4	...

INSERT INTO SELECT

```
-- single-row table to store when we rolled up last
CREATE TABLE latest_rollup (
  minute timestamptz PRIMARY KEY,
  -- "minute" should be no more precise than a minute
  CHECK (minute = date_trunc('minute', minute))
);
-- initialize to a time long ago
INSERT INTO latest_rollup VALUES ('10-10-1901');
-- function to do the rollup
CREATE OR REPLACE FUNCTION rollup_http_request() RETURNS void AS $$
DECLARE
  current_time     timestamptz := date_trunc('minute', now());
  last_rollup_time timestamptz := minute from latest_rollup;
BEGIN
  INSERT INTO http_request_1min (
    site_id, ingest_time, request_count,
    success_count, error_count, average_response_time_msec
  ) SELECT
    site_id,
    date_trunc('minute', ingest_time),
    COUNT(1) as request_count,
    SUM(CASE WHEN (status_code between 200 and 299) THEN 1 ELSE 0 END) as success_count,
    SUM(CASE WHEN (status_code between 200 and 299) THEN 0 ELSE 1 END) as error_count,
    SUM(response_time_msec) / COUNT(1) AS average_response_time_msec
  FROM http_request
  -- roll up only data new since last_rollup_time
  WHERE date_trunc('minute', ingest_time) <@
          tstzrange(last_rollup_time, current_time, '(]')
  GROUP BY 1, 2;
  -- update the value in latest_rollup so that next time we run the
  -- rollup it will operate on data newer than current_time
  UPDATE latest_rollup SET minute = current_time;
END;
$$ LANGUAGE plpgsql;
```

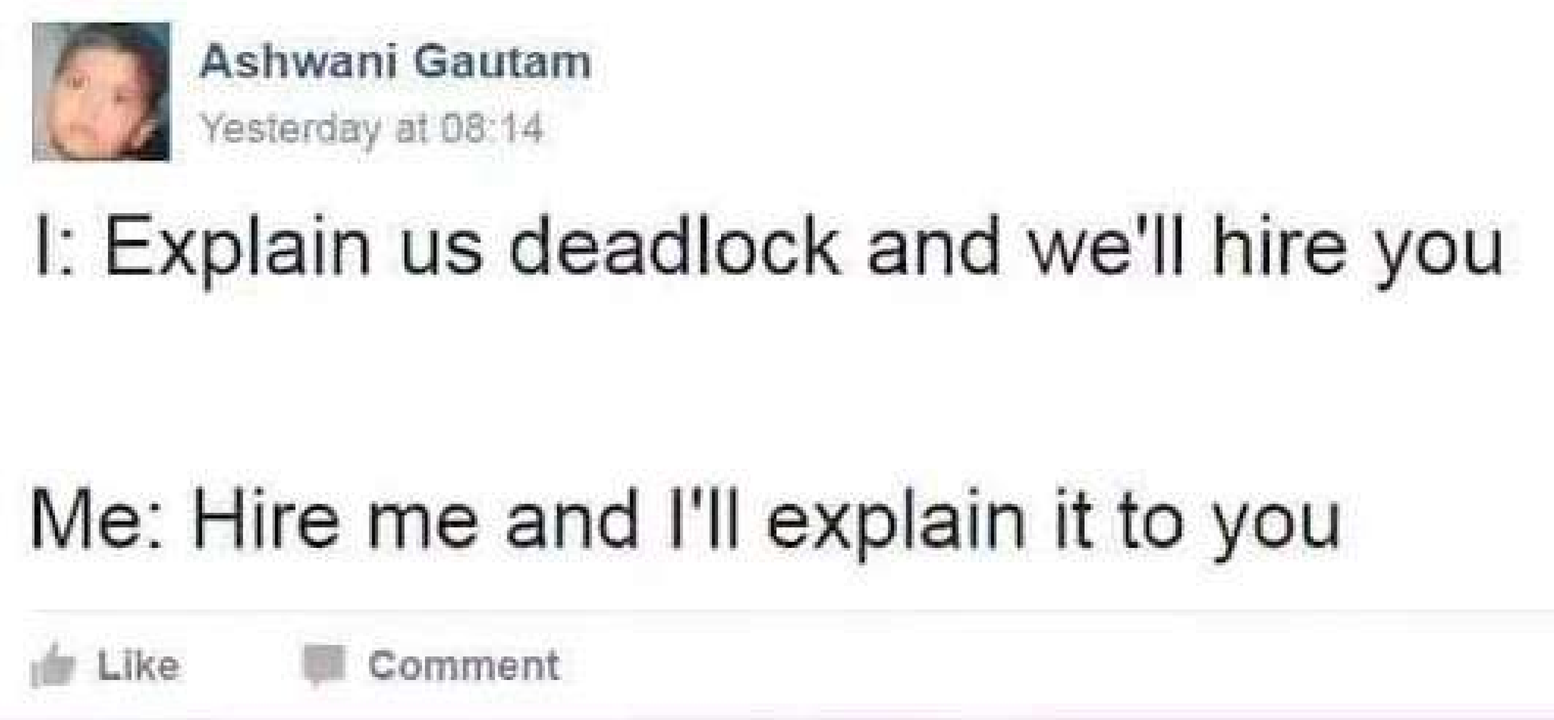
DISTRIBUTED TRANSACTIONS



```
BEGIN;
UPDATE accounts
  SET balance = balance - '100.00'
  WHERE id = 'Alice';
```

```
BEGIN;
SELECT assign_distributed_transaction_id(...);
UPDATE accounts_1001   SET ...
```

DEADLOCK DETECTION



Ashwani Gautam

Yesterday at 08:14

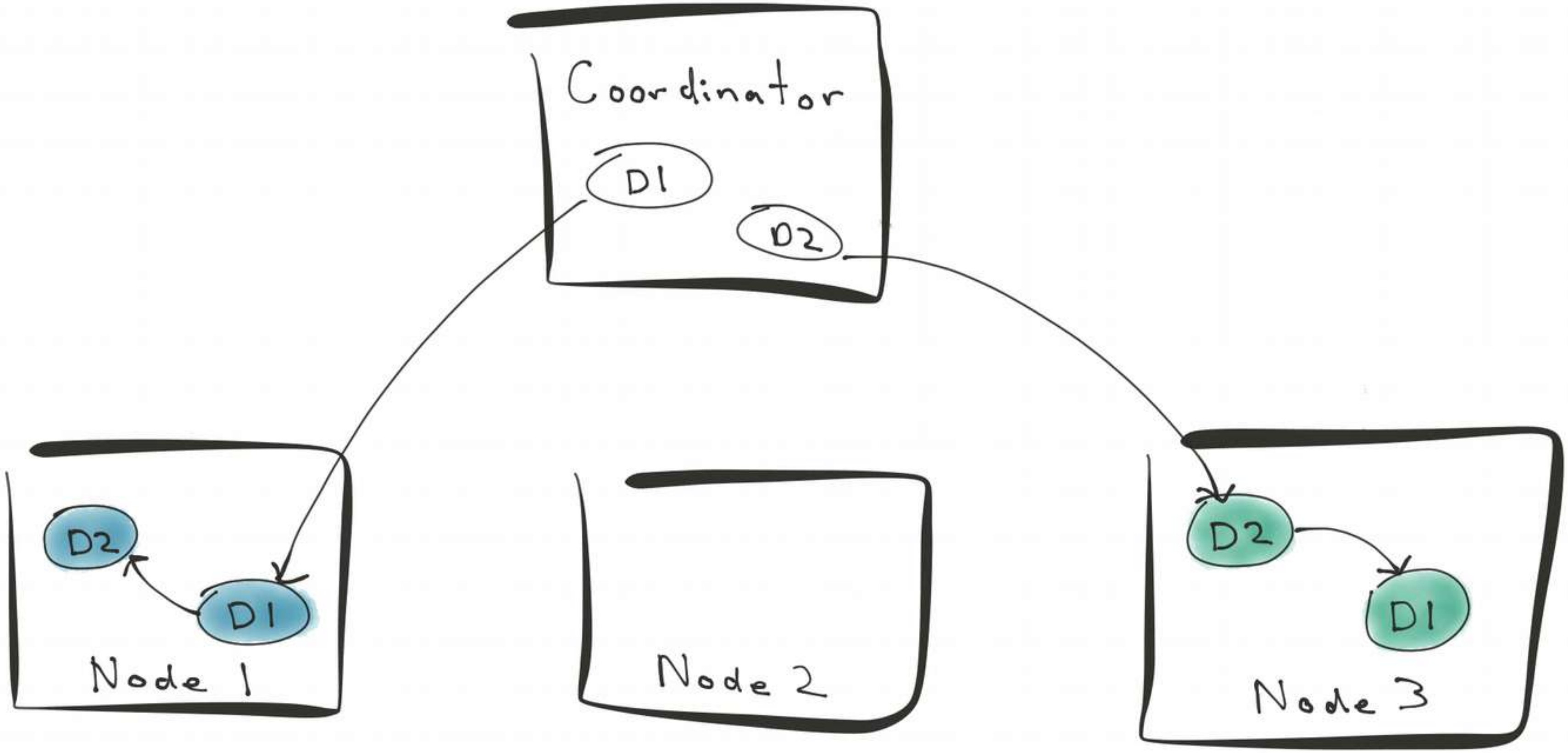
I: Explain us deadlock and we'll hire you

Me: Hire me and I'll explain it to you

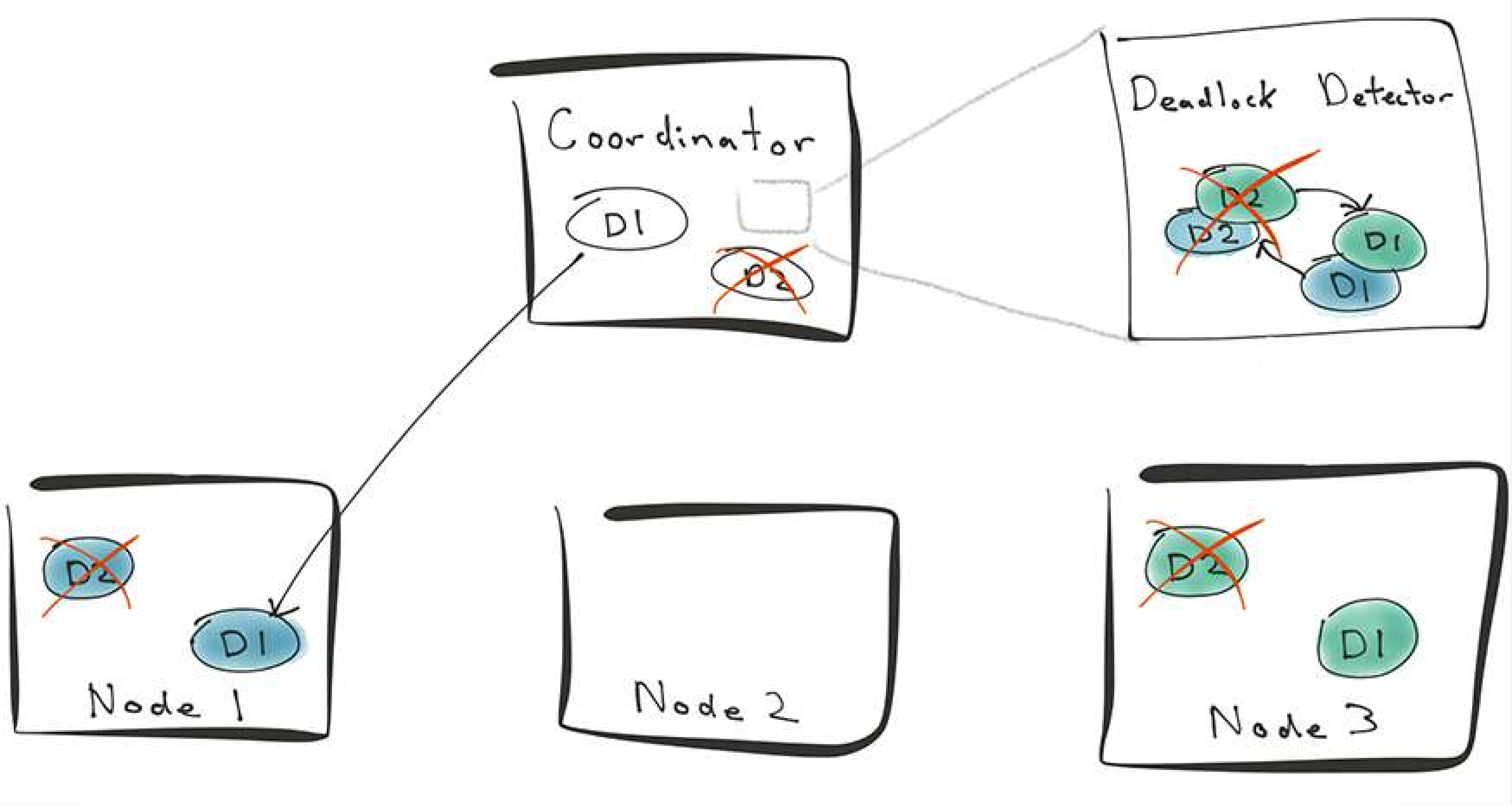
DEADLOCK DETECTION

```
S1:  BEGIN;  // session 1 starts transaction block
S1:  UPDATE accounts
     SET balance = balance - '100.00'
     WHERE id = 'Alice';  // S1 takes 'Alice' lock

S2:  BEGIN;  // session 2 starts transaction block
S2:  UPDATE accounts
     SET balance = balance + '100.00'
     WHERE id = 'Bob';  // S2 takes 'Bob' lock

S1:  UPDATE accounts
     SET balance = balance + '100.00'
     WHERE key = 'Bob';  // waits for 'Bob' lock held by S2

S2:  UPDATE accounts
     SET balance = balance - '100.00'
     WHERE key = 'Alice';  // deadlocks on 'Alice' lock held by S1
```

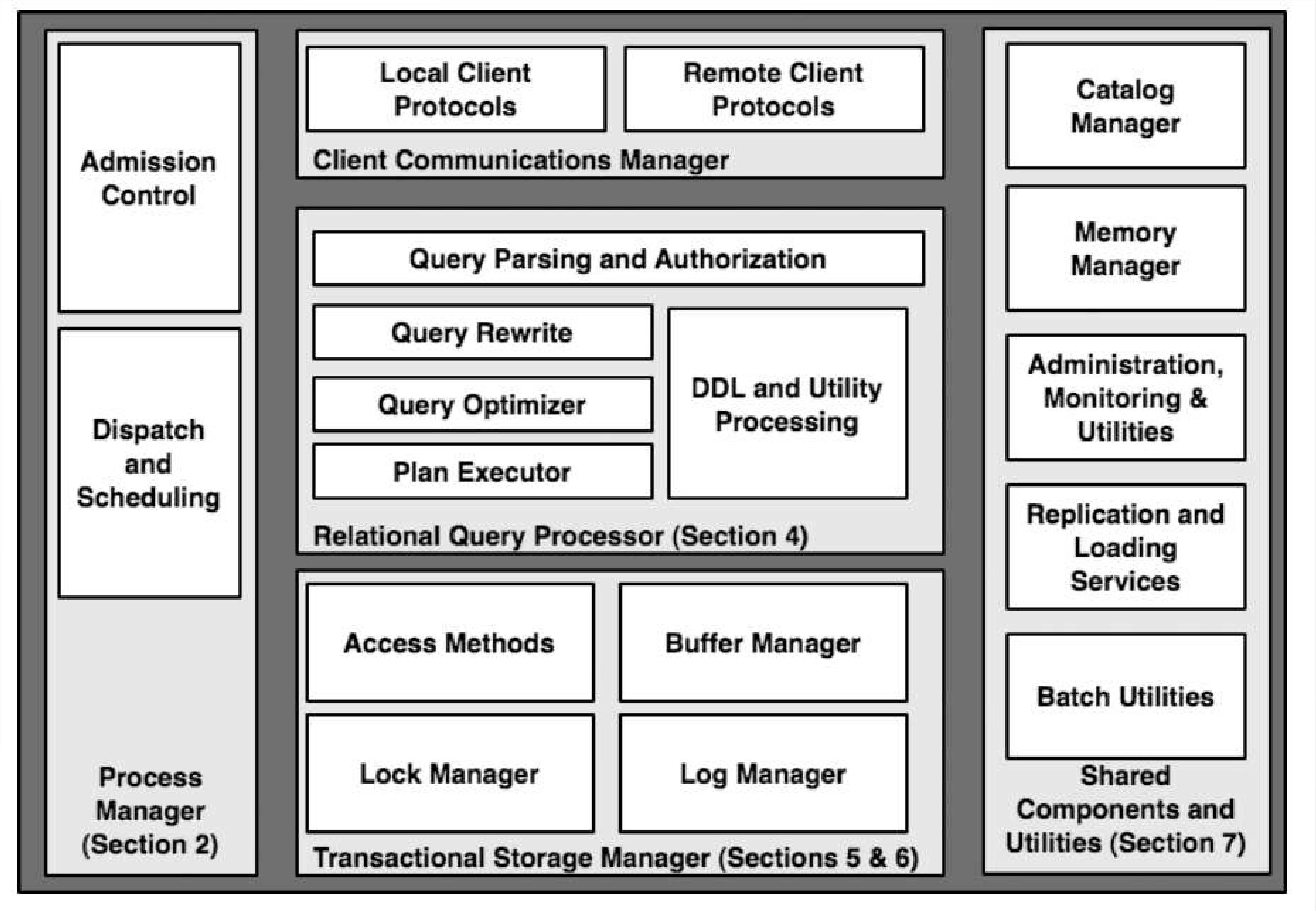
DEADLOCK DETECTION



DEADLOCK DETECTION



Por que extender?


Admission Control
Dispatch and Scheduling
Process Manager (Section 2)
Local Client Protocols
Remote Client Protocols
Client Communications Manager
Query Parsing and Authorization
Query Rewrite
Query Optimizer
Plan Executor
DDL and Utility Processing
Relational Query Processor (Section 4)
Access Methods
Buffer Manager
Lock Manager
Log Manager
Transactional Storage Manager (Sections 5 & 6)
Catalog Manager
Memory Manager
Administration, Monitoring & Utilities
Replication and Loading Services
Batch Utilities
Shared Components and Utilities (Section 7)

Casos de uso

CISCO

algolia

Bloomberg

PANOPLY™

HEAP

Casos de uso

„Hoje uma de nossas tabelas do PostgreSQL ultrapassou meio trilhão de registros [mais de 54TB de dados]. E ainda assim a maior parte de nossas queries rodam em menos de 600ms. Somente graças a @citusdata e o seu apoio incrível“ - Pex CEO

F.A.Q.

- Posso criar PKs?
- Posso adicionar mais nós?
- Falha dos workers?
- Falha dos coordinators?
- Posso misturar tabelas distribuídas e locais?
- Como criar roles, functions, extensions em workers?

F.A.Q.

- E se um só mudar de endereço?
- Posso distribuir por múltiplas colunas?
- Falha dos coordinators?

Conclusão

„...Quando escolhemos extender Postgres, nos disseram que SQL não escala. E acontece que é muito fácil dispensar um problema aparentemente intratável ao afirmar algo que banaliza o problema. E melhor forma de se resolver um problema complexo não é ao descartá-lo e sim quebrá-lo em pedaços menores e resolvê-los um por um...“ - Ozgun Erdogan, Citusdata CTO

Obrigado!!!

4Linux-Privado
Subutai
Conquer the Cloud
4LINUX
Linux
JOY DIVISION