

# Inleiding procedurele SQL



[Wim.bertels@ucll.be](mailto:Wim.bertels@ucll.be)

# ISO Standaard

- Sinds 1996, 1999, ..., 2023
- SQL/PSM (Persistent Stored Modules)
- Veel producten richten zich op de standaard, maar geen enkele is volgt deze helemaal.
  - Bekijk de documentatie van het concrete product.

# Postgresql

- Ondersteuning voor verschillende andere programmeertalen.
  - SQL
  - PL/pgSQL (standaard)
  - PL/Tcl
  - PL/Perl
  - PL/Python
  - PL/Java (uitbreiding)

# “Vetrouwde vs. Niet-Vetrouwde”

- Trusted (vertrouwd): deze talen zijn beperkt tot het wat is toegelaten door de databank (DBMS)
- Untrusted (niet-vertrouwd): deze talen zijn daartoe niet beperkt. Superuser toegang nodig om deze te gebruiken

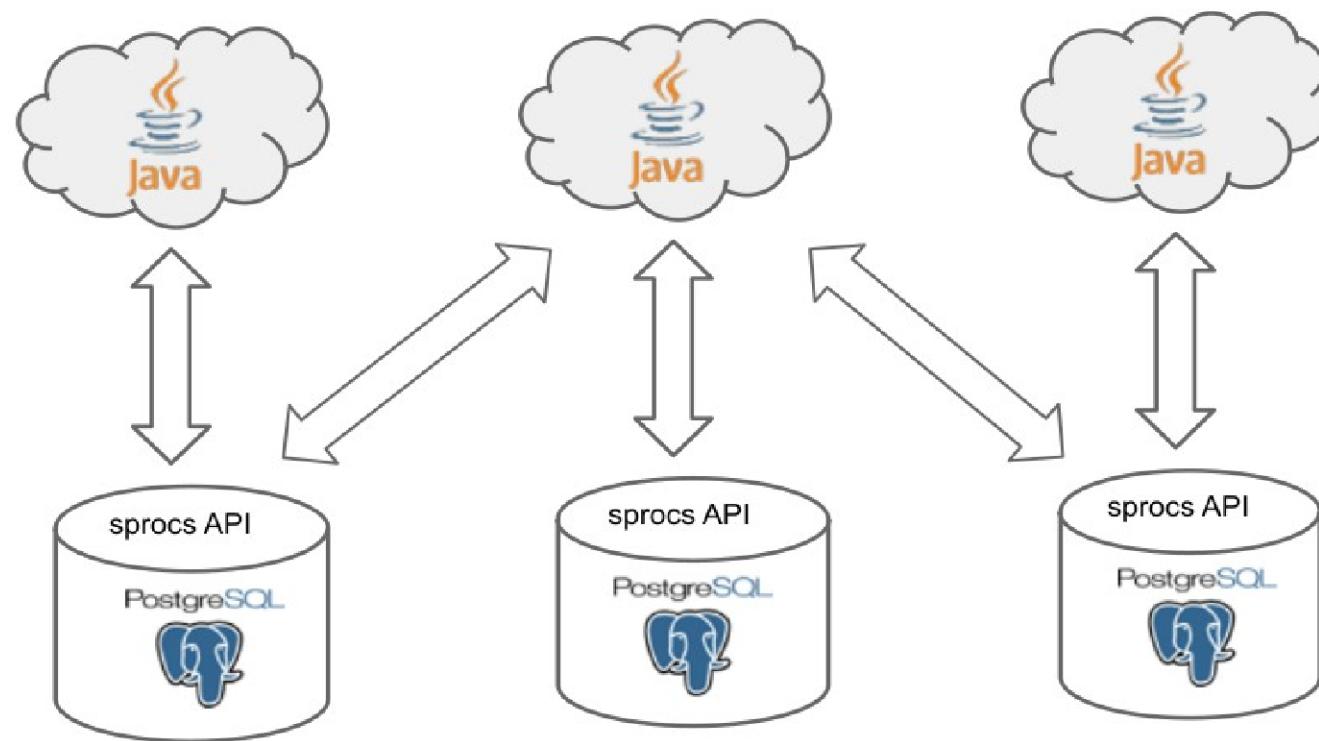
# Extra beschikbare modules

Postgresql komt met sommige extra modules zoals bijvoorbeeld:

- Debugger (bv pldebugger voor pgadmin)
- Profiler (bv plprofiler, of algemener pg\_stat\_statements)

# How Trustly and Zalando are using PostgreSQL

Applications access data in PostgreSQL databases via calls to stored procedures.



(slide copied from the excellent 2014.pgconf.eu presentation by Alexey Klyukin @ Zalando)

# Referenties

- <https://www.postgresql.org/docs/current/static/server-programming.html>
- <https://www.postgresql.org/docs/current/sql-createlanguage.html>
- <https://www.postgresql.org/docs/current/contrib.html>
- <https://en.wikipedia.org/wiki/SQL/PSM>
- "How we use Postgresql at Trustly, 'Joel Jacobson', October 2014, Madrid

Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International Public License

# Procedurele SQL met (plpgsql)



Wim.bertels@ucll.be

# Objecten op de server

- Stored procedures
- Stored functions
- Triggers
- Def. : hoeveelheid code die opgeslagen is in de catalogus van een DB en die geactiveerd kan worden
- Vergelijkbaar met wat je in programmeren een (statische) methode zou noemen.

# Voorbeeld: stored function

```
CREATE OR REPLACE FUNCTION increment(i INT)
RETURNS INT
AS
$$
BEGIN
RETURN i + 1;
END;
$$ LANGUAGE 'plpgsql';
```

-- Een voorbeeld van hoe de functie op te roepen:

```
SELECT increment(10);
```

# Verwerking

## Verwerking :

- Vanuit programma wordt procedure/functie opgeroepen
- DBMS ontvangt oproep en zoekt procedure
- Procedure wordt uitgevoerd waarbij de instructies op de database verwerkt worden
- M.b.v. een code wordt aangegeven of procedure correct verwerkt is (sqlcode)

# Parameters Algemeen

- Communicatie met buitenwereld
- 3 soorten (signatuur methode) :
  - Invoerparameters
  - Uitvoerparameters
  - Invoer/uitvoerparameters
- Parameter best andere naam dan kolom van tabel !
- Return (kan enkel bij functions)

# Voorbeeld IN OUT

```
CREATE FUNCTION dup(in int, out f1 int, out f2 text)
```

```
AS
```

```
$$
```

```
    SELECT $1, CAST($1 AS text) || ' is text'
```

```
$$
```

```
LANGUAGE SQL;
```

```
SELECT * FROM dup(42);
```

# Signatuur: Functions vs Procedures

- Algemeen:
  - Het grote verschil is dat functies altijd iets teruggeven
  - Terwijl procedures nooit een return hebben
  - In de praktijk afhankelijk van het concrete DBMS dat je gebruikt

# Waarom?

- Historisch verschil tussen berekeningen (functies) en manipulaties (procedures)
- Daarnaast:
  - Transacties kunnen enkel binnen procedures
  - Zo kan de planner hiermee rekening houden

# Instructie mogelijkheden (de essentie)

- Declaratie
- Sequentie
- Selectie
  - IF .. THEN .. (ELSE ..) END IF
  - CASE .. WHEN .. THEN ... END CASE
- Iteratie
  - FOREACH .. LOOP .. END LOOP
  - FOR .. IN .. LOOP .. END LOOP

# Structuur: functies

```
CREATE FUNCTION function_name(arg1,arg2,...)
```

```
    RETURNS type
```

```
    AS
```

```
'
```

```
    BEGIN
```

```
        -- logic
```

```
    END;
```

```
'
```

```
    LANGUAGE language_name;
```

# Structuur: functies

- Specifieer naam van functie
- Lijst van parameters achter naam van functie
- Definieer return type
- Gevolgd door code binnen begin- en end block
- Procedurele taal meegeven

# Verwijderen stored functions

- **DROP FUNCTION:** verwijdert functie
- Vb.
  - `DROP FUNCTION function_name;`
  - `DROP FUNCTION function_name(signatuur);`

# SELECT INTO voorbeeld

- Enkel indien een rij als uitvoer! Vb.

```
create function som_boetes_speler(p_spelersnr integer)
    returns decimal(8,2)
AS
$eenderwat$
    declare    som_boetes decimal(8,2);
begin
    select sum(bedrag)
        into som_boetes
     from boetes
    where spelersnr = p_spelersnr;
    return som_boetes ;
end;
$eenderwat$
language plpgsql;

select som_boetes_speler (27);
```

# \$\$ delimiter dialect

```
create function som_boetes_speler(p_spelersnr integer)
    returns decimal(8,2)
    AS
```

**\$eenderwat\$**

```
declare som_boetes decimal(8,2);
begin
    select sum(bedrag)
    into som_boetes
    from boetes
    where spelersnr = p_spelersnr
    return som_boetes;
end;
```

**\$eenderwat\$**

```
language plpgsql;
```

```
select som_boetes_speler (27);
```

# PERFORM

- Resultaten van een sql statement moeten opgevangen worden, bv via INTO variabele.
- PERFORM alternatief voor SELECT waarbij het resultaat niet wordt opgevangen
  - Bv SELECT now(); zal een fout geven in een code blok
  - PERFORM now(); niet
- Vergelijkbaar met het void maken van een functie in andere programmeer talen

# FOUND globale variabele

- Boolean
- Bijvoorbeeld:
  - PERFORM spelersnr FROM spelers ;
  - IF FOUND THEN .. END IF ;

# RETURN(S)

- Signatuur :
  - RETURNS type
  - (RETURNS setof type)
  - RETURN TABLE (column\_name type,...)
- Code :
  - RETURN scalair..
  - RETURN QUERY ..
  - RETURN NEXT .. + RETURN

# Foutberichten

- Foutberichten :
- SQL-error-code : beschrijvende tekst
- SQLSTATE: code (getal)

```
BEGIN
    -- code
    RAISE DEBUG 'A debug message % ', variable_that_will_replace_percent;
EXCEPTION
    -- welke fout, bv
    WHEN uniqueViolation THEN
        -- code
    WHEN division_by_zero THEN
        RAISE .. -- eventueel omzetten naar uniqueViolation;
    WHEN others THEN
        -- ?
        NULL;
END
```

# RAISE

- RAISE;
- RAISE division\_by\_zero;
- RAISE SQLSTATE '22012';
- RAISE DEBUG/INFO/..  
    -- SET client\_min\_messages TO debug;
- RAISE .. USING
  - ERRCODE = 'uniqueViolation',
  - HINT = 'suggestie voor de reden voor de gebruiker',
  - DETAIL = 'meer detail fout',
  - MESSAGE = 'gedrag van de uniqueViolation';

# ASSERT

- ASSERT condition [, message];

```
do
$$
declare
    film_count integer;
begin
    select count(*)
    into film_count
    from film;
    assert film_count > 0, 'No films found, check the film table';
end
$$;
```

-- do is dialect, <https://www.postgresql.org/docs/current/sql-do.html>

# SECURITY

- GRANT EXECUTE ON haha() TO jomeke ;
- INVOKER : standaard, veilig
- DEFINER : via de rechten van de eigenaar

```
CREATE OR REPLACE FUNCTION haha()
```

```
    RETURNS text
```

```
    AS
```

```
$code$
```

```
BEGIN
```

```
    DROP TABLE ola();
```

```
    RETURN 'pola';
```

```
END
```

```
$code$
```

```
LANGUAGE plpgsql
```

```
EXTERNAL SECURITY DEFINER;
```

# LEVEL of immutability

CREATE FUNCTION add(integer, integer) RETURNS integer

AS 'select \$1 + \$2;'

LANGUAGE SQL

**IMMUTABLE**

RETURNS NULL ON NULL INPUT;

- **IMMUTABLE** (alleen afhankelijk van de signatuur)
- **STABLE** (geen aanpassingen)
- **VOLATILE** (standaard)

# Structuur: stored procedure

CREATE OR REPLACE PROCEDURE

<procedure\_name>( <arguments> )

AS <block-of-code>

LANGUAGE <implementation-language>;

# Transactie voorbeeld: stored procedure

CREATE OR REPLACE PROCEDURE voorbeeld(invoer text )

AS

\$code\$

BEGIN

...

**IF** invoer = 'niet doen' **THEN** RAISE WARNING 'Abort, the ship is sinking';

**ROLLBACK;**

**ELSEIF** invoer = 'doen' **THEN** RAISE INFO 'Gaon met die banaan';

**COMMIT;**

**END IF;**

...

**END**

\$code\$

LANGUAGE plpgsql;

CALL voorbeeld('doen');

-- <https://www.postgresql.org/docs/16/plpgsql-transactions.html>

# Praktisch: script

BEGIN;

code en testen

ROLLBACK;

DROP [procedure|function|trigger] naam

CREATE [procedure|function|trigger] naam

ALTER [procedure|function|trigger] naam

CREATE OR REPLACE [procedure|function|trigger] naam

# Triggers

- Def. :  
hoeveelheid code die opgeslagen is in de catalogus en die geactiveerd wordt door het dbms indien een bepaalde operatie wordt uitgevoerd en een conditie waar is.
- Triggers worden door het dbms zelf automatisch opgeroepen (niet door vb. een call)

# PostgreSQL

- Triggers roepen  
trigger functies op
- **CREATE FUNCTION trigger\_functie...**  
**RETURNS TRIGGER**  
..
- **CREATE TRIGGER trigger\_trg ..**  
..  
**EXECUTE PROCEDURE trigger\_functie..**

# Voorbeeld: tabel

```
create table mutaties (
    gebruiker          varchar(30)  not null,
    mut_tijdstip       timestamp   not null,
    mut_spelersnr     smallint    not null,
    mut_type          char(1)      not null,
    mut_spelersnr_new smallint    ,
    primary key
        (gebruiker, mut_tijdstip, mut_spelersnr, mut_type)) ;
```

-- tabel om wijzigingen bij te houden

# Voorbeeld: trigger functie

```
create or replace function insert_speler() returns trigger as  
$body$
```

```
begin
```

```
    insert into mutaties values
```

```
        (user, current_date(), new.spelersnr, 'I', null) ;
```

```
end;
```

```
$body$
```

```
language sql;
```

# Voorbeeld: trigger

```
create or replace trigger insert_speler.trg
```

```
after insert
```

```
on spelers
```

```
-- when new.spelersnr < 10
```

```
for each row
```

```
execute procedure insert_speler();
```

-- OLD en NEW verwijzen naar de huidige toestand en de nieuwe toestand

-- welke verschillende onderdelen zie je hier die typisch voor een

-- trigger zijn ?

# Onderdelen Trigger

- Trigger-moment + Trigger-gebeurtenis:
  - Wanneer activeren?
    - AFTER : nadat triggering instructie is verwerkt
    - BEFORE : eerst de trigger-actie
    - INSTEAD OF : alleen de trigger-actie
  - Voor welke rij activeren ?
    - FOR EACH ROW : voor elke rij
    - FOR EACH STATEMENT : voor een statement
  - Voor welke gebeurtenis?
    - INSERT, UPDATE, DELETE, (TRUNCATE)
- Trigger-Conditie: WHEN
- Trigger-actie: wat doet de trigger?

# Syntax

```
CREATE [ OR REPLACE ] [ CONSTRAINT ] TRIGGER name { BEFORE | AFTER | INSTEAD OF } { event [ OR ... ] }
ON table_name
[ FROM referenced_table_name ]
[ NOT DEFERRABLE | [ DEFERRABLE ] [ INITIALLY IMMEDIATE | INITIALLY DEFERRED ] ]
[ REFERENCING { { OLD | NEW } TABLE [ AS ] transition_relation_name } [ ... ] ]
[ FOR [ EACH ] { ROW | STATEMENT } ]
[ WHEN ( condition ) ]
EXECUTE { FUNCTION | PROCEDURE } function_name ( arguments )
```

where event can be one of:

INSERT

UPDATE [ OF column\_name [, ... ] ]

DELETE

TRUNCATE

URL: <https://www.postgresql.org/docs/current/sql-createtrigger.html>

# Syntax

CREATE/ALTER/DROP TRIGGER

ALTER TABLE .. ENABLE/DISABLE TRIGGER ..

GRANT/REVOKE TRIGGER ON TABLE .. TO ..

( CREATE EVENT TRIGGER – DDL)

# Standaard

- De standaard laat meer acties dan een trigger functie toe (udf : user defined function)
- Bv
  - create trigger delete\_spelers  
after delete on spelers for each row  
begin  
    delete from spelers\_wed  
    where spelersnr = old.spelersnr;  
end ;  
-- geen verwijzing naar udf, maar rechtstreeks de sql code

# Notas

- Er zitten verschillen tussen producten :
  - Meerdere triggers op 1 tabel ? En wat met de volgorde ?
  - Kan een trigger een andere trigger activeren ? (waterval/domino !)
  - Wat mag een trigger allemaal doen ?
  - Hoe worden (complexere) triggers juist verwerkt ?
  - ..

# Gebruik?

- Denk gebeurtenis gestuurd
- Voorbeelden :
  - (Integriteits)regels
  - Audit
  - Consistentie
  - Beveiliging
  - Afgeleide waarden berekenen
  - (Data)validatie
  - ..

# Voordelen

- **Onderhoudbaarheid:**  
Vb. Snellere uitvoering door meer instructies in macro
- **Verwerkingsnelheid**  
Vb. minimaliseert netwerkverkeer
- « Precompilatie » bij stored procedures/functions/triggers
  - Planning en caching
  - Werkt in verschillende host-languages

# Nadelen

- Nadelenken over architectuur en code organisatie
- Algemeen zoals bij andere talen : logische fouten vs syntaxfouten
  - Bv Onverwachte neveneffecten bij gebruik van (veel) (overlappende) triggers

# Referenties

Slides: Stored Procedures Functions Triggers, P. Demazière, 2018

Slides: Procedurel SQL, H.Martens, W.Bertels,2014

Postgresql 11 Server Side Programming Quick Start Guide, L. Ferrari, 2018

<https://www.postgresqltutorial.com/postgresql-plpgsql>

<https://www.postgresql.org/docs/current/plpgsql-trigger.html>

<https://www.postgresql.org/docs/current/sql-grant.html>

<https://www.postgresql.org/docs/current/triggers.html>

<https://www.postgresql.org/docs/current/sql-createtrigger.html>

<https://www.postgresql.org/docs/current/plpgsql.html>

<https://www.postgresql.org/docs/current/xfunc-sql.html>

Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International Public License

# Beveiliging



[Wim.bertels@ucll.be](mailto:Wim.bertels@ucll.be)

# Security

- Hardware
- Het platform waar het op draait
- De databank software
- Binnen sql zelf (grant, revoke, view,...)
- Sql als “vertaler”
- Applicaties
- Gebruikers
- Algemeen terugkerende security problematiek (bv CIA, AAA, maar ook reserve kopies, ..)
- ..

# Ondersteunend platform

- Bijdetijs (Up to date) houden
- Beveiligen (infrastructuur vakken)
- (D)DoS-attack's
- ...

# Databank software

- Up to date houden
- Configuratie
  - Local
  - Remote
    - Eg postgresql: pg\_hba.conf, postmaster.conf en postgresql.conf
- Known exploits -> Wat doe je?

# Binnen SQL zelf

- User management (roles)
- Privileges (grant / revoke)
- Views
- Stored procedures
- Row Security Policies

# SQL

- SQL wordt niet gecompileerd
- SQL wordt “vertaald” in de onderliggende/omsluitende laag
- Dit kan gebruikt worden om sql ongewenste instructies te laten uitvoeren
- Het is probleem dat bij elke taal die “vertaald”, terugkomt (eg php)

# SQL Injection

- Indien we de onderliggende sql code van bv een formulier niet kennen, dan gaan we proberen te raden wat de sql code is die erachter zit.
- Ipv gewone waarden gaan sql code meegeven met het formulier.
- Pas op serial, identity, autoincrement..?

# Incorrectly Filtered Escape Characters

- Fout: input is niet gefilterd op escape characters
- Bv
  - statement := "SELECT \* FROM users WHERE naam = " + userNaam + ";"
  - UserNaam:= a' or 't='t
  - Gevolg: SELECT \* FROM users WHERE naam = 'a' or 't='t';
  - Dus.. , andere voorbeelden?

# Incorrect Type Handling

- Fout: types van de input worden niet gecheckt
- Bv
  - statement := "SELECT \* FROM data WHERE id = "  
+ a + ";" (a wordt enkel verwacht als int)
  - Voor a := 1;DROP TABLE users;
  - Gevolg: SELECT \* FROM data WHERE id = 1;DROP TABLE users;
  - Dus..

# Oplossingen

- Escape character verwijderen uit de invoervelden
- Invoer validatie, controleren of het invoerveld bv wel het juiste type heeft
- Prepared statements
- Stored procedures
  - Type
  - ; en escaping
  - Grants..
  - Dicht bij de bron
  - ..

# Opgepast

- Enkel Escaping is niet voldoende:
- Bv
  - `SELECT * from items where userid=$userid;`
  - `$userid := "33 or userid is not null or userid=44";`
  - `SELECT * from items where userid=33 or userid is not null or userid=44;`
- Dus zeg eerder wat wel mag zijn als invoer, ipv wat niet mag; het eerste is eenvoudiger, er zijn (meestal) maar een eindig aantal mogelijkheden.

# Handige Functies

- `quote_ident ( text ) → text`
- `quote_literal ( anyelement ) → text`
  - `quote_nullable ( anyelement ) → text`

# Dieper

- <https://www.postgresql.org/docs/current/ddl-rowsecurity.html>

```
CREATE TABLE users (
```

```
    manager text,
```

```
    company text );
```

```
ALTER TABLE users ENABLE ROW LEVEL SECURITY;
```

```
CREATE POLICY users_manager
```

```
    ON users TO managers -- groep rol managers
```

```
    USING (manager = current_user);
```

- <https://www.postgresql.org/docs/current/sepgsql.html>
- <https://www.postgresql.org/docs/current/sql-security-label.html>

# Links

- <http://www.w3schools.com/>
- <http://www.postgresql.org/>
- <http://en.wikipedia.org/>

Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International Public License

# Replicatie

(CC BY-NC-SA 4.0)

[Wim.bertels@ucll.be](mailto:Wim.bertels@ucll.be)

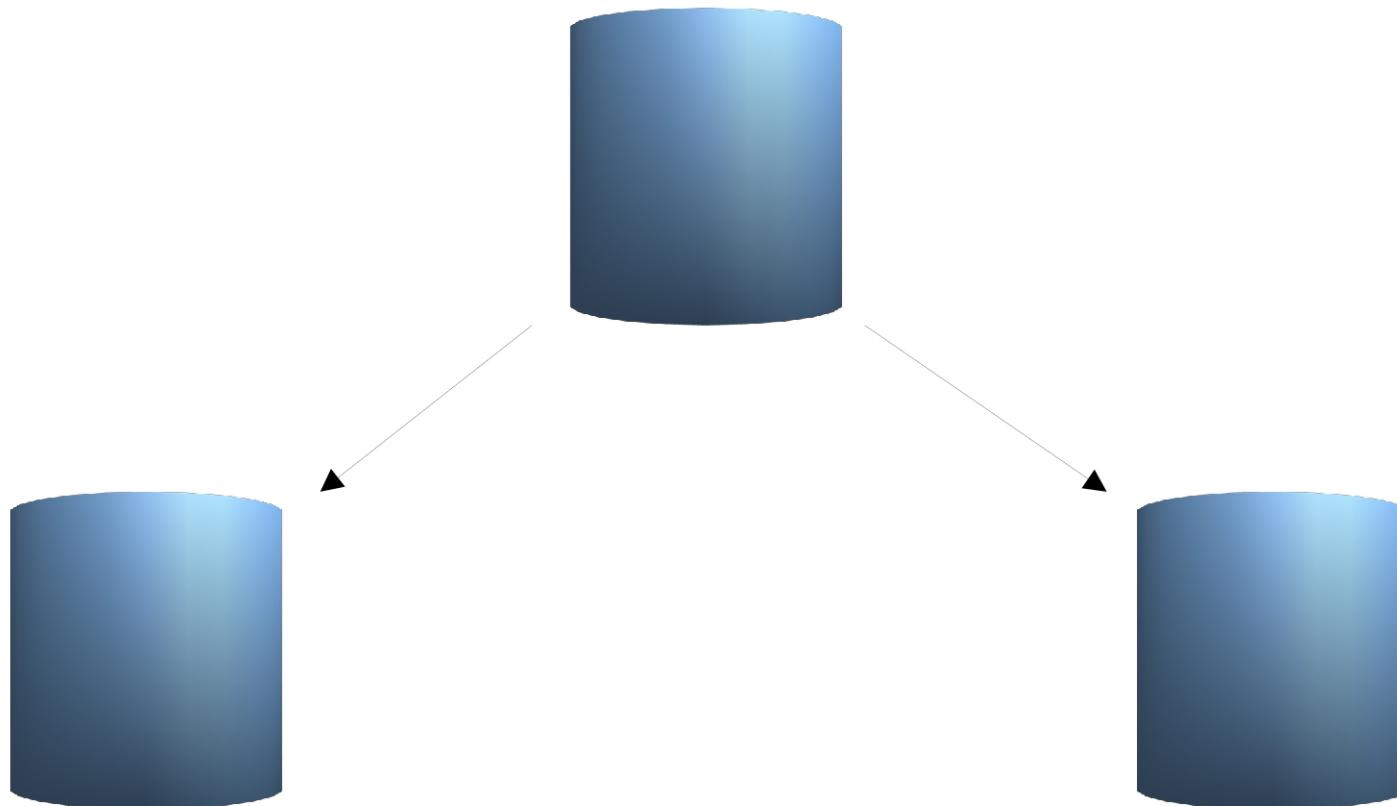
# Replicatie

- Verband CAP theorema
  - Hoge beschikbaarheid (naast data partitionering, parallele query verdeling over meerdere servers, ..)
- Fysische
- Logische
- Andere (bv triggergebaseerd, externe applicatie ea)

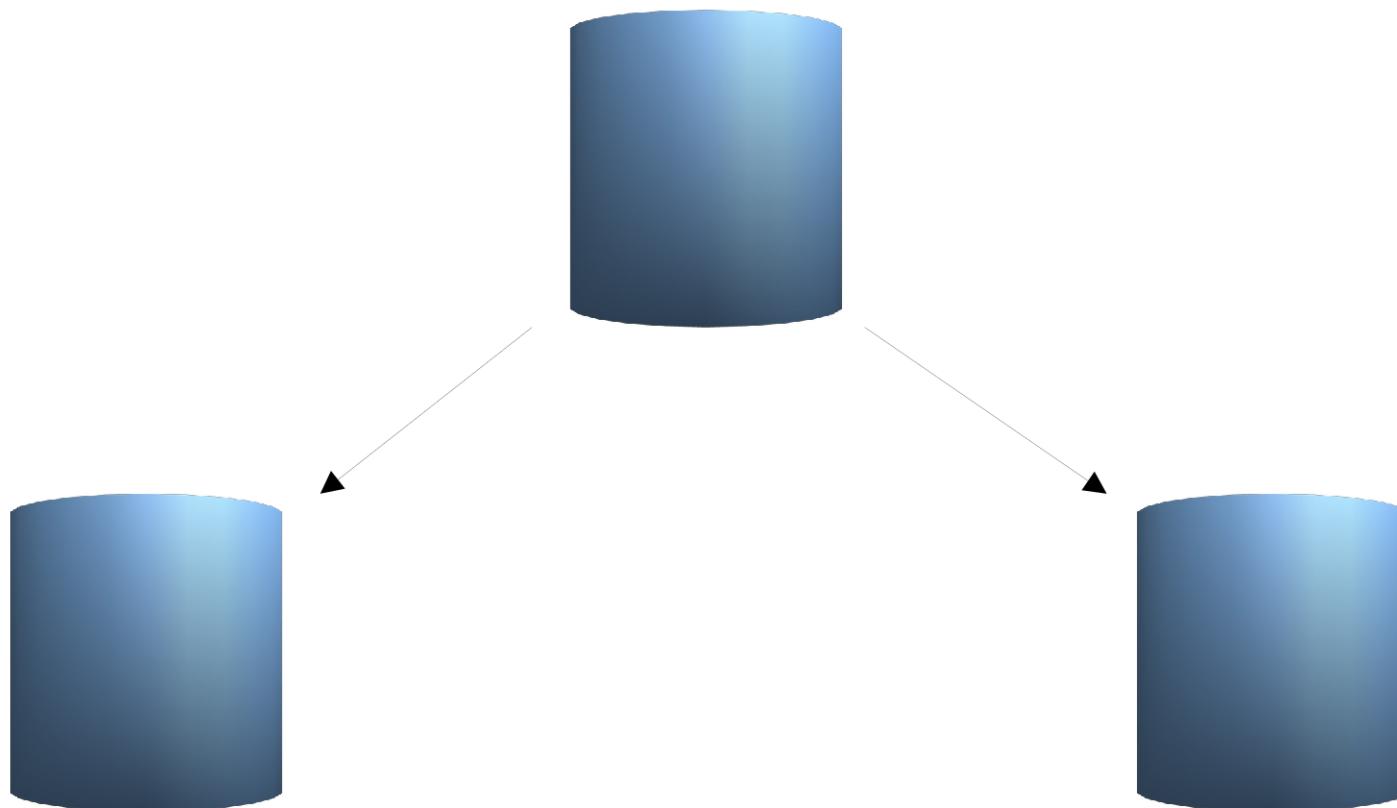
# CAP

- Consistentie:  
alle knooppunten in het systeem zien dezelfde data op hetzelfde moment
- Availability (beschikbaarheid):  
elke aanvraag krijgt een antwoord terug.
- Partitie tolerant:  
als een knooppunt uitvalt, dan blijft het systeem functioneren
  - > 2 van de 3
  - \* uitbreiding: pacelc (latency vs consistency)
  - \* sync/async

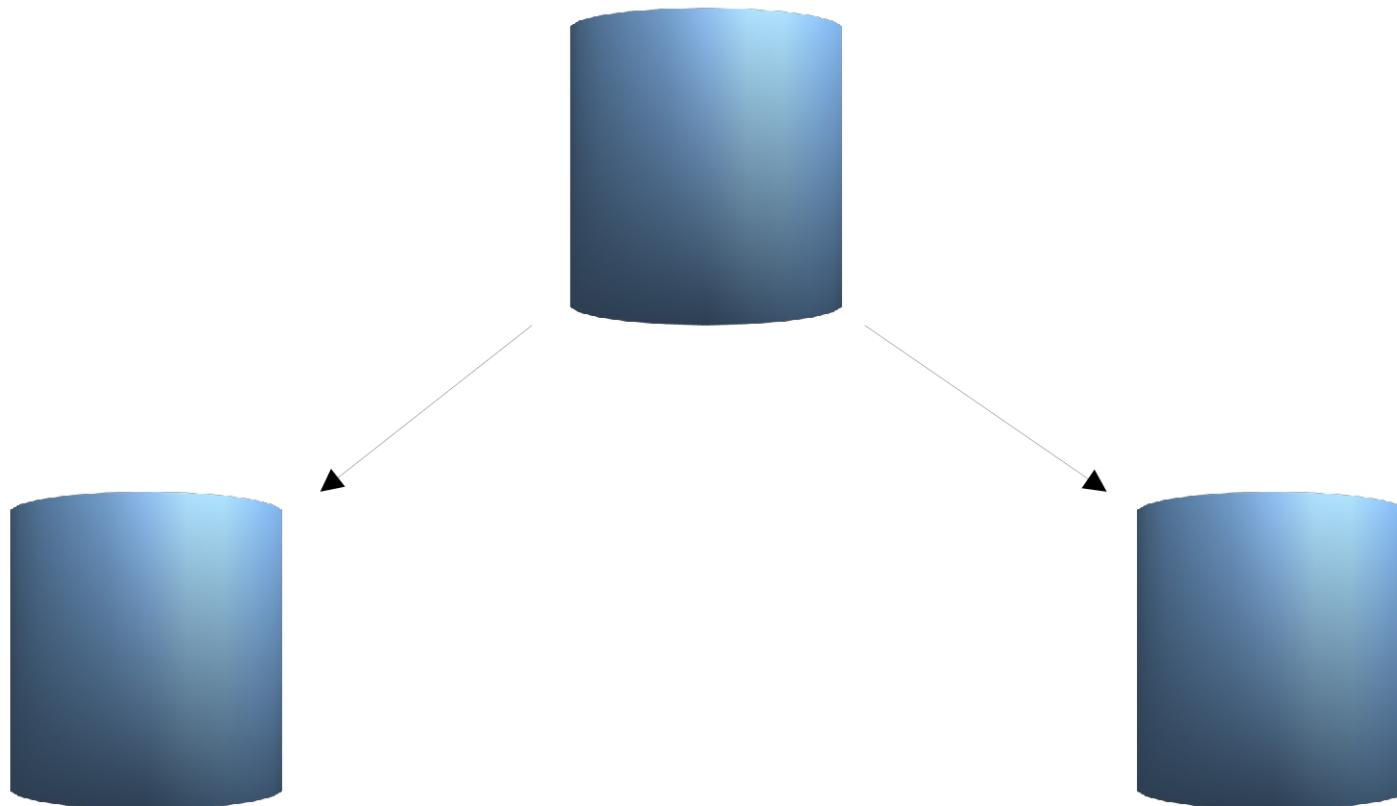
# CAP Voorbeeld 1



# CAP Voorbeeld 3



# CAP Voorbeeld 2



# Andere: Niet ingebouwd

- <https://www.symmetricds.org/>

HOME    ABOUT    DOWNLOAD    DOCUMENTATION    DEVELOPER    GET HELP

## Fast & Flexible Database Replication

*SymmetricDS is open source database replication software that focuses on features and cross platform compatibility.*



### CROSS PLATFORM

Replicate data across different platforms, with compatibility for many databases. Sync from any database to any database in a heterogeneous environment.

[READ MORE +](#)



### SCALE OUT PERFORMANCE

Optimized for performance and scalability, replicate thousands of databases asynchronously in near real time, and span replication across multiple tiers.

[READ MORE +](#)



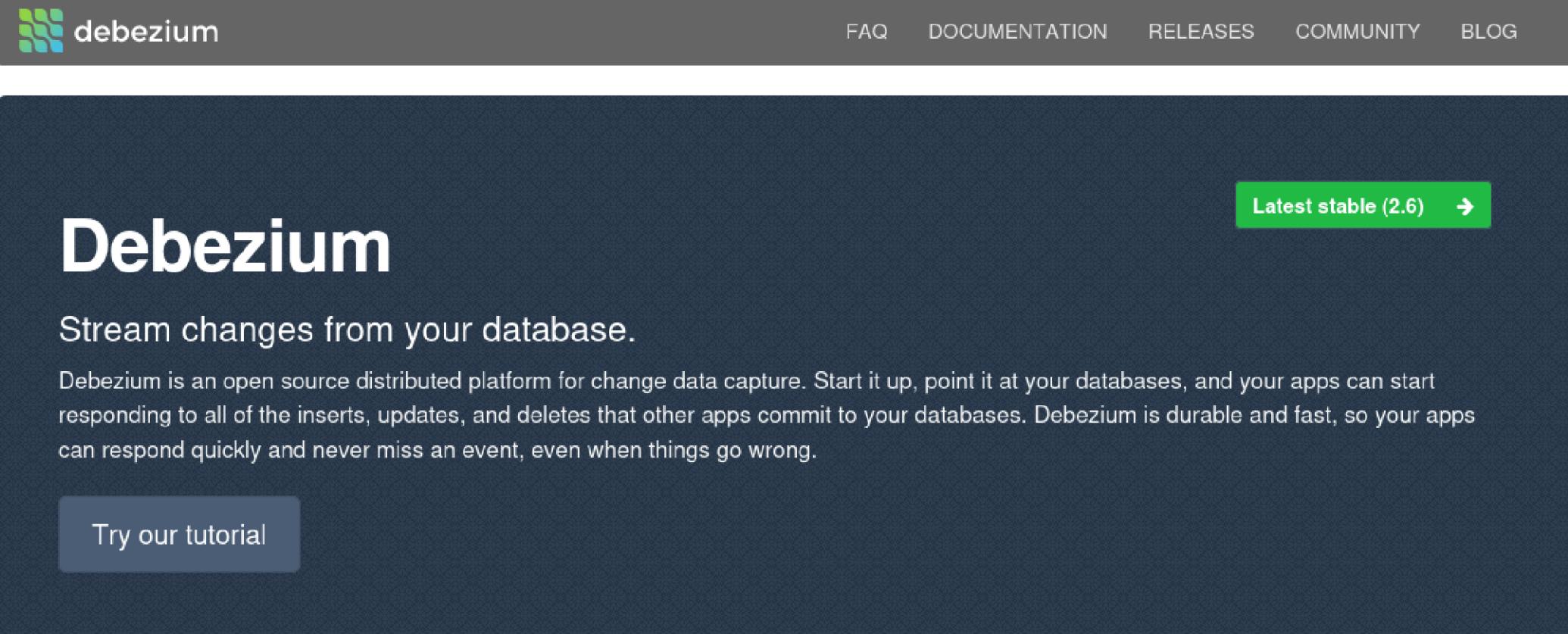
### FLEXIBLE CONFIGURATION

Configure which tables and columns to sync, and in which direction. Subset rows and distribute them across databases. Combine, filter, and transform data.

[READ MORE +](#)

# Andere: Niet ingebouwd

- <https://debezium.io/>



The screenshot shows the official Debezium website. At the top, there's a dark navigation bar with the Debezium logo on the left and links for FAQ, DOCUMENTATION, RELEASES, COMMUNITY, and BLOG on the right. Below the header, the word "Debezium" is prominently displayed in large white letters. To its right is a green button labeled "Latest stable (2.6)" with a white arrow pointing right. Underneath the title, the tagline "Stream changes from your database." is written in white. A detailed description follows: "Debezium is an open source distributed platform for change data capture. Start it up, point it at your databases, and your apps can start responding to all of the inserts, updates, and deletes that other apps commit to your databases. Debezium is durable and fast, so your apps can respond quickly and never miss an event, even when things go wrong." At the bottom left, there's a blue button with the text "Try our tutorial".

# Fysisch

- Exacte kopie
- “Transactielog” (xlog > pg:WAL)

# Logische

- Fijner
- “SQL” : logical decoding
- Typische gebruik:
  - Incrementeel veranderingen doorsturen
  - Verdere afhankelijkheden op de subscriber
  - Verzamelen van data van verschillende databanken
  - Replicatie over verschillende besturingssystemen en/of versies van db software
  - Toegangsbeleid (rechten op subscriber naar rollen)
  - Een deel van een db delen met andere dbn

# Overzicht binnen planeet pg

Feature	Shared Disk	File System Repl.	Write-Ahead Log Shipping	Logical Repl.	Trigger-Based Repl.	SQL Repl. Middle-ware	Async. MM Repl.	Sync. MM Repl.
Popular examples	NAS	DRBD	built-in streaming repl.	built-in logical repl., pglogical	Londiste, Slony	pgpool-II	Bucardo	
Comm. method	shared disk	disk blocks	WAL	logical decoding	table rows	SQL	table rows	table rows and row locks
No special hardware required		•	•	•	•	•	•	•
Allows multiple primary servers				•		•	•	•
No overhead on primary	•		•	•		•		
No waiting for multiple servers	•		with sync off	with sync off	•		•	
Primary failure will never lose data	•	•	with sync on	with sync on		•		•
Replicas accept read-only queries			with hot standby	•	•	•	•	•
Per-table granularity				•	•		•	•
No conflict resolution necessary	•	•	•		•	•		•

# Links

- <https://www.postgresql.org/docs/current/high-availability.html>
- <https://www.postgresql.org/docs/current/logical-replication.html>
- [https://en.wikipedia.org/wiki/CAP\\_theorem](https://en.wikipedia.org/wiki/CAP_theorem)

Wim Bertels

Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International Public License

# XML

(CC BY-NC-SA 4.0)

[Wim.bertels@ucll.be](mailto:Wim.bertels@ucll.be)

# XML

- eXtensible Markup Language
- DTD (of xml schema)
- 1996
- hierarchische structuren in een nieuw kleedje
- platte tekstbestanden
- ISO standaard
- einde RDBMS..?

# eenvoudige XML Syntax

- men gebruikt een tag om aan te geven waar men begint en eindigt
- elke document moet in 1 root tag omvat zijn
- beginnen met een cijfer mag niet
- case sensitive
- bv
  - <example> tekst </example>
- tags mogen genest worden

# Voorbeeld

```
<energie>  
  <gerecht>  
    <naam> frieten </naam>  
    <quitering> zeer wel </quitering>  
  </gerecht>  
  <gerecht>  
    <naam> sojapuddingzonderchoco </naam>  
    <quitering> twijfelachtig </quitering>  
  </gerecht>  
</energie>
```

# Doel

- xml (data)    vs    html (vorm)  
(xsl                 vs        css)
- om zowel hierarchische datastructuren te omschrijven als deze te bevatten
- om data uit wisselen tussen verschillende gegevensbronnen      (cf s-patroon:mediator)
  - een gemeenschappelijke taal
  - geef hiervan een voorbeeld.

# METADATA

- je kan attributen aan je tags meegeven, deze moeten tussen “ “ of ' ' staan
- gebruik dit enkel om METADATA weer te geven, bv eigenschappen
- de data zelf horen daar niet thuis
- bv

```
<example type='music'> lalala.mp3 </example>
```

# Uitbreidingen

- xml word bv ook gebruikt om configuraties van software bij te houden bv menubalk,..
- er zijn verschillende formaten die van deze standaard gebruik maken:

xhtml, xml dom, xsl, xslt, xpath, xsl-fo, xlink,  
xpointer, dtd, xsd, xforms, xquery, soap,  
wsdl, rdf, rss, svg, wap, smil, ..

# Pro Contra

- ISO
- uitwisselbaarheid
- eenvoud
- gn hierarchische engine nodig
- ..
- redundantie tov relationeel model, dus niet misbruiken in die zin
- sequentieel, traag
- ..

# RDBMS

- Verschillende RDBMS voorzien manieren om xml te gebruiken
  - <http://www.postgresql.org/docs/current/static/datatype-xml.html>
  - <http://www.postgresql.org/docs/current/static/functions-xml.html>

# Voorbeelden

```
SELECT xmlcomment('hallo');
```

xmlcomment

-----

```
<!--hallo-->
```

```
SELECT xmlelement(name jos,
    xmlattributes(current_date as ke), 'hal', 'lo');
```

xmlelement

---

```
<jos ke="2014-01-26">hallo</jos>
```

```
select xmlforest(divisie, teamnr) from teams;
```

xmlforest

---

```
<divisie>ere </divisie><teamnr>1</teamnr>
<divisie>tweede</divisie><teamnr>2</teamnr>
```

```
SELECT xmlpi(name php, 'echo "Patat";');
```

xmlpi

---

```
<?php echo "Patat";?>
```

```
..  
table_to_xml(tbl regclass, nulls boolean,  
tableforest boolean, targetns text)  
  
query_to_xml(query text, nulls boolean,  
tableforest boolean, targetns text)  
  
cursor_to_xml(cursor refcursor, count int, nulls  
boolean, tableforest boolean, targetns text)
```

..

# Links

- <http://www.postgresql.org/docs/current/static/datatype-xml.html>
- <http://www.postgresql.org/docs/current/static/functions-xml.html>

Wim Bertels

Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International Public License

# FDW

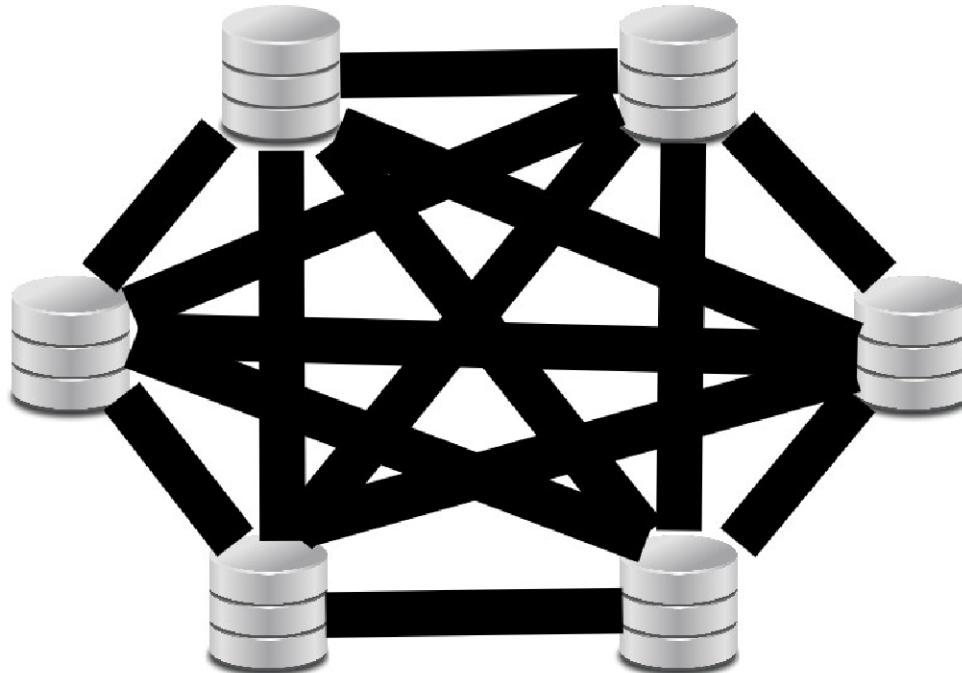
[wim.bertels@ucll.be](mailto:wim.bertels@ucll.be)

Naamsvermelding-NietCommercieel-GelijkDelen 4.0 Unported  
Licentie

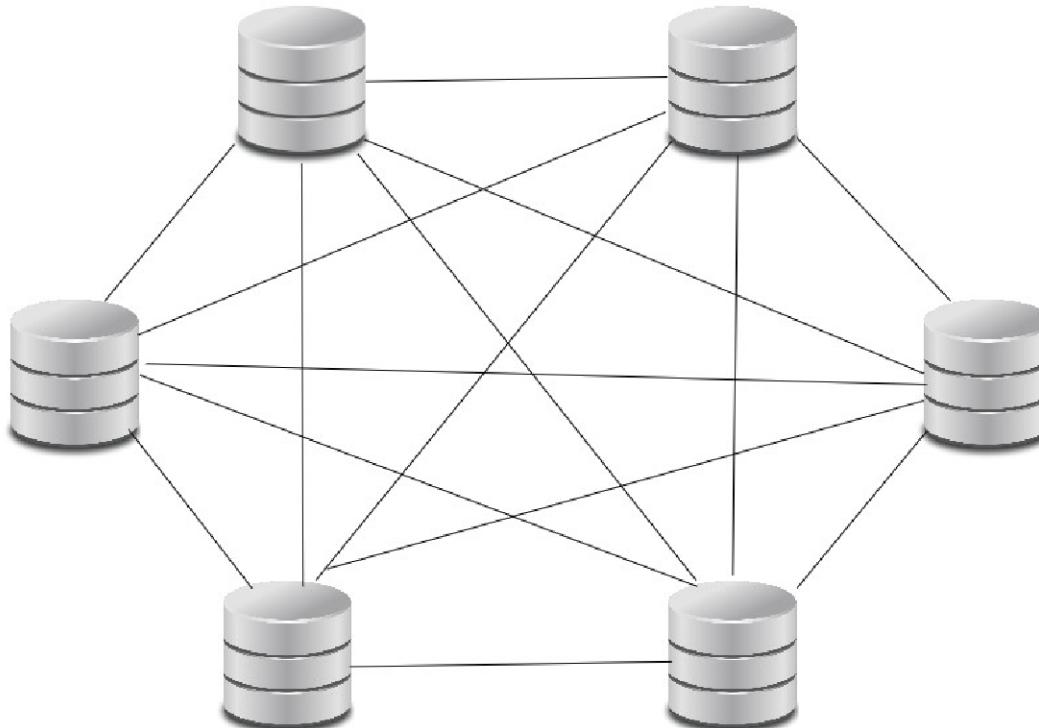
# Wat

- Foreign Data Wrapper (FDW)
- Onderdeel van SQL/MED (ISO)
  - Foreign Table
  - Datalink
    - <https://github.com/lacanoid/datalink>

# Opslag vs Netwerk



# Opslag vs Netwerk



# Bronnen

Local server



Remote servers



<https://www.interdb.jp/pg/pgsql04.html>

# Bronnen

- [https://wiki.postgresql.org/wiki/Foreign\\_data\\_wrappers](https://wiki.postgresql.org/wiki/Foreign_data_wrappers)
  - Databanken
  - Bestanden
  - GIS
  - LDAP
  - Web
  - BS
  - ..

# Basis

- Waar en hoe: server
  - **CREATE SERVER**
- Wie: beveiliging, gebruiker afbeelding
  - **CREATE USER MAPPING**
- Wat: welke tabellen
  - **CREATE FOREIGN TABLE**
- Nodig voor gebruik: bibliotheek
  - **CREATE EXTENSION**

# CREATE EXTENSION

- CREATE EXTENSION postgres\_fdw;
- CREATE EXTENSION file\_fdw;
- <https://gdal.org/>
  - <https://github.com/pramsey/pgsql-ogr-fdw>

# Referenties

- Universal Data Access with SQL/MED, David Fetter
- <https://wiki.postgresql.org/wiki/SQL/MED>