

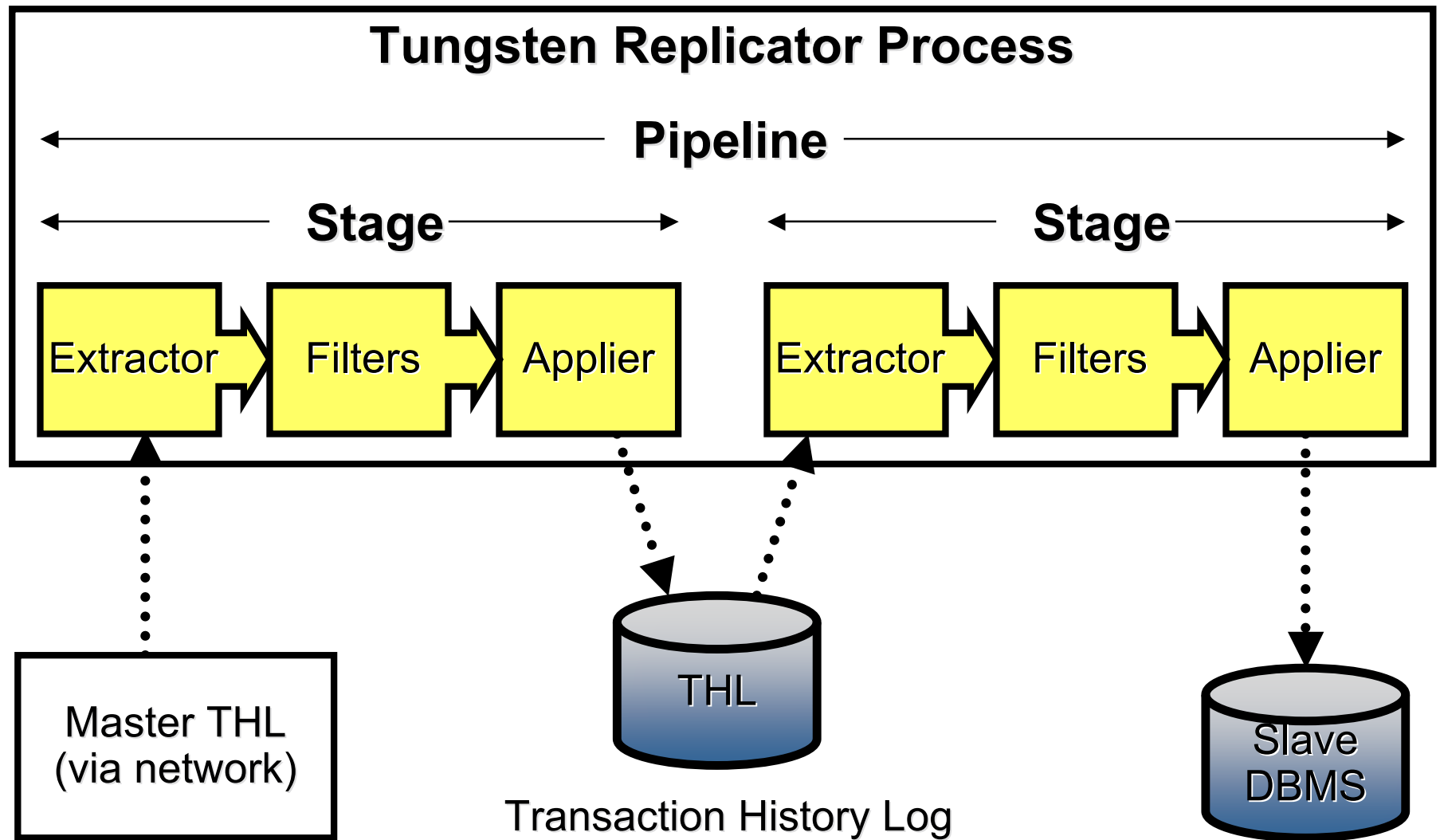
Heterogeneous Replication



© Continuent 2011

continuent
Open. Always Available.

Replicator Review: Pipelines



Tungsten Capabilities

**Tungsten can replicate
transactionally
consistent data in real
time from MySQL to
PostgreSQL/Greenplum
and Oracle***

***But it's sometimes not simple**



A Word About MySQL Replication



© Continuent 2011

continuent
Open. Always Available.

Writeable slaves

	Slave <u>readable</u>	Slave <u>writeable</u>
MySQL Statement Based	+	+
MySQL Row Based	+	+
MySQL Mixed	+	+
PostgreSQL WAL Shipping	-	-
PostgreSQL Streaming Replication (Hot Standby)	+	-

/ Tungsten for MySQL

- You can both read and write to the DB operating in slave mode
- This, potentially, allows you to break consistency
- In some scenarios that is a plus

/ Tungsten for PostgreSQL

- In Streaming Replication you can read from slaves
- Still, you cannot write to them
- In effect, you cannot have different data/structure on a slave



Statement and Row Based Replication

/ Statement Based Replication



```
CREATE TABLE t (id INT, name  
VARCHAR(12));
```

```
ALTER TABLE t ADD COLUMN id2 INT;
```

```
INSERT INTO t VALUES (1, 'Sun', 999);
```

```
UPDATE t SET name = 'Moon';
```



/ Row Based Replication



```
CREATE TABLE t (id INT, name  
VARCHAR(12));
```

```
ALTER TABLE t ADD COLUMN id2 INT;
```

```
{ INSERT, t, (1, 'Sun', 999) };
```

```
{ UPDATE, t, id == 1, (1, 'Moon', 999) };
```



Logical vs. Physical Replication

	Logical	Physical
MySQL Statement Based	x	
MySQL Row Based	x	
MySQL Mixed	x	
PostgreSQL WAL Shipping		x
PostgreSQL Streaming Replication		x
Filters (data transformation) possible	+	-
Different data/structure on slave possible	+	-



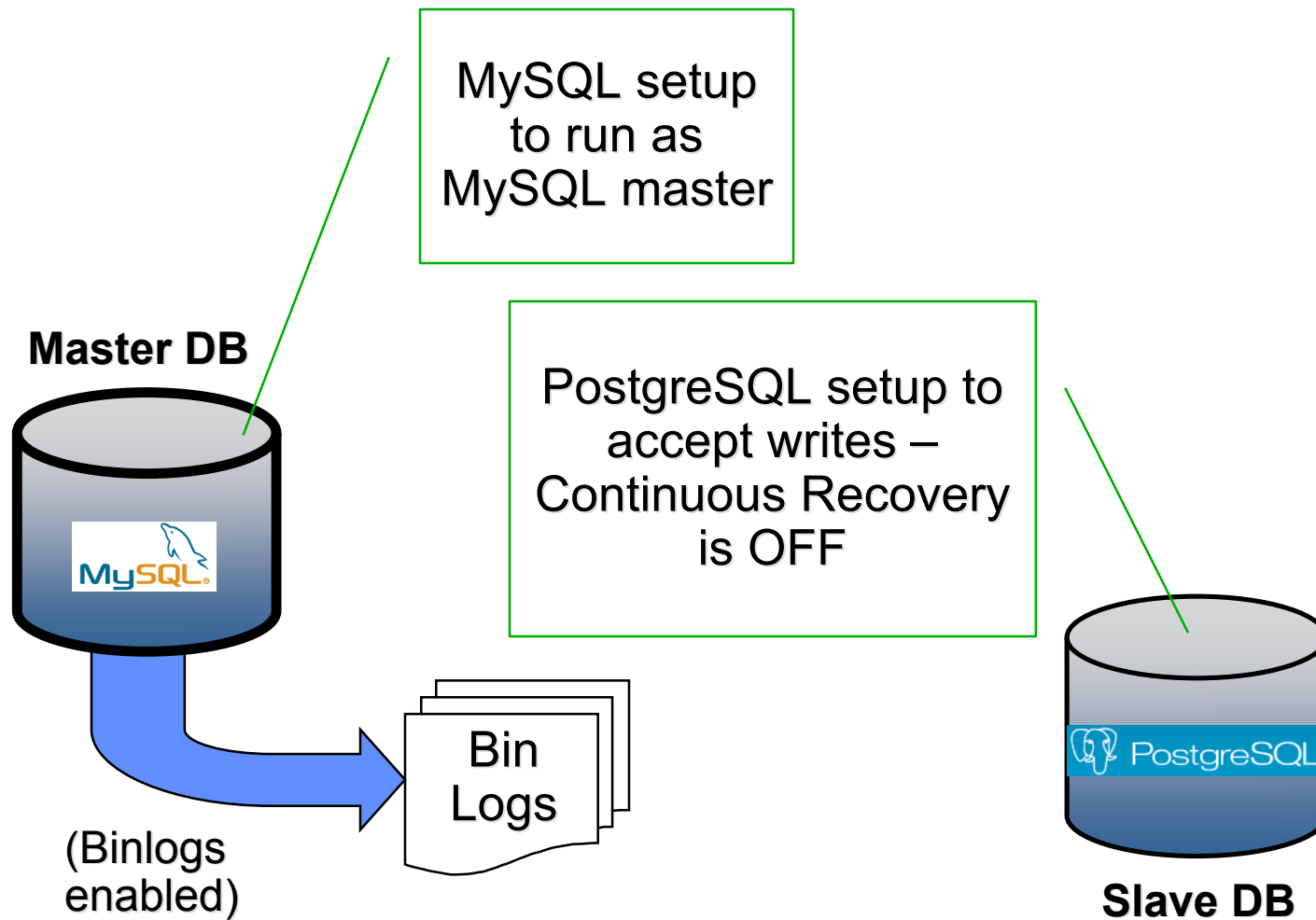
MySQL to PostgreSQL/Oracle Replication



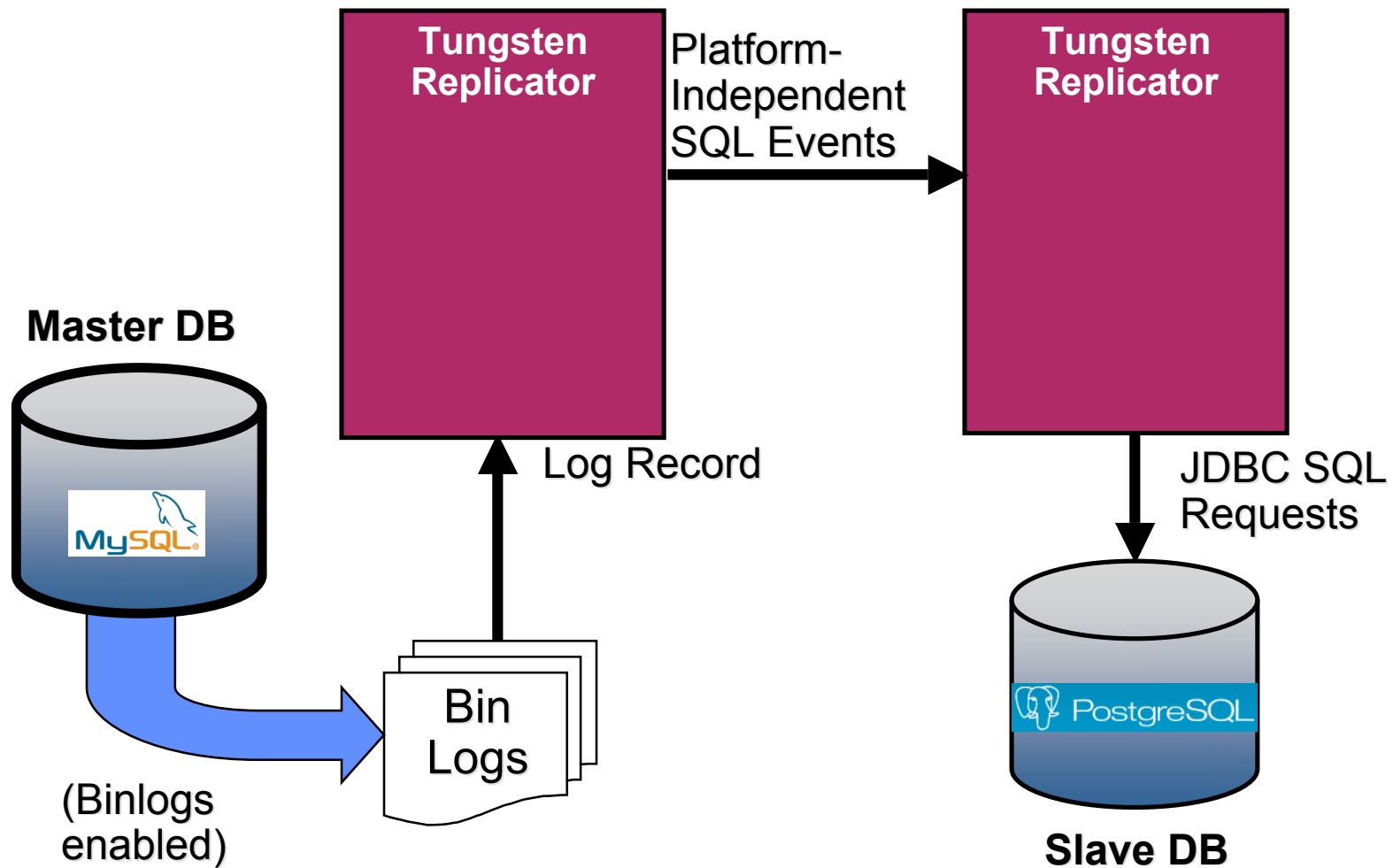
© Continuent 2011

continuent
Open. Always Available.

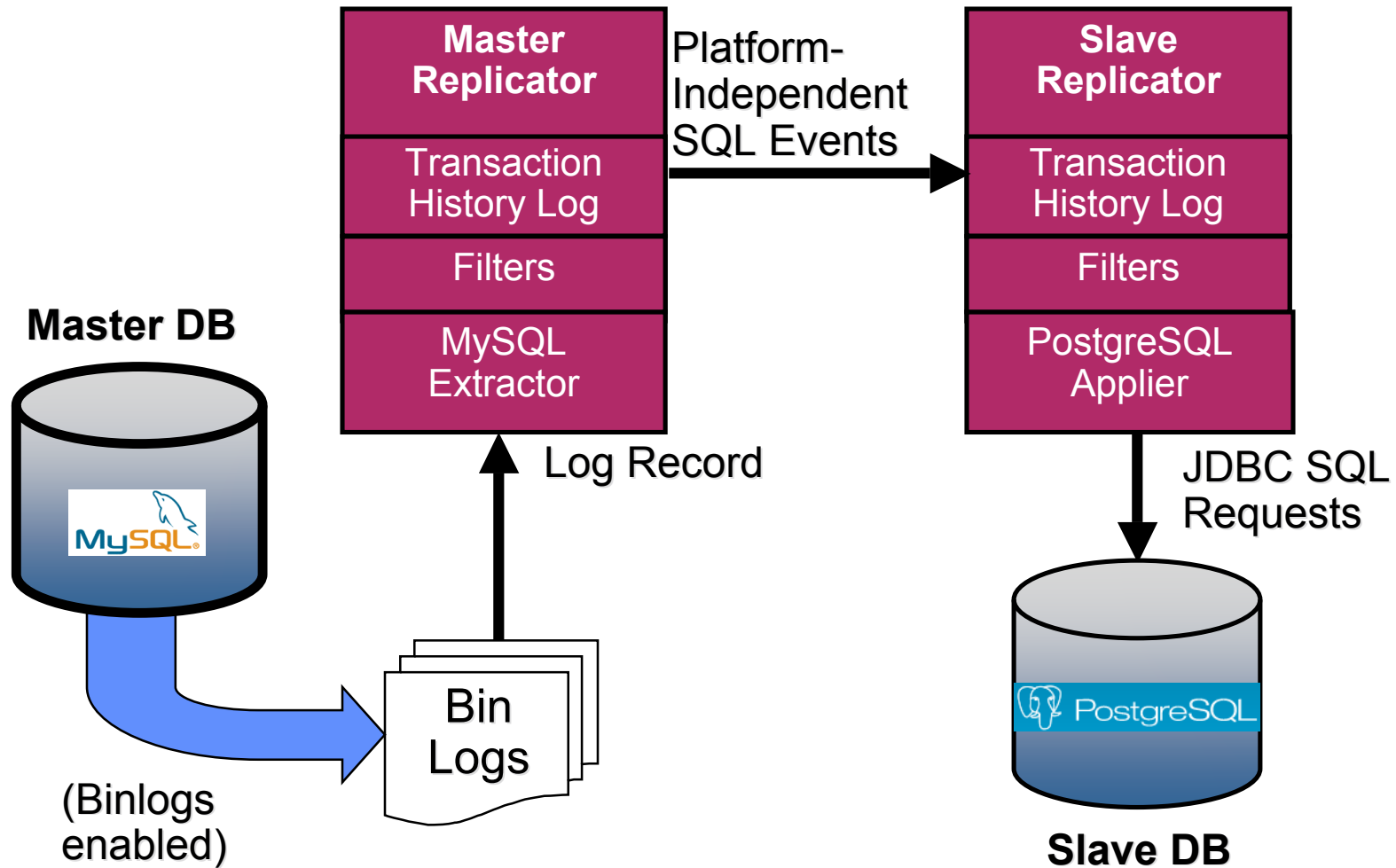
Who's Who?



Tungsten Replicator Doing the Job



Tungsten Replicator Components



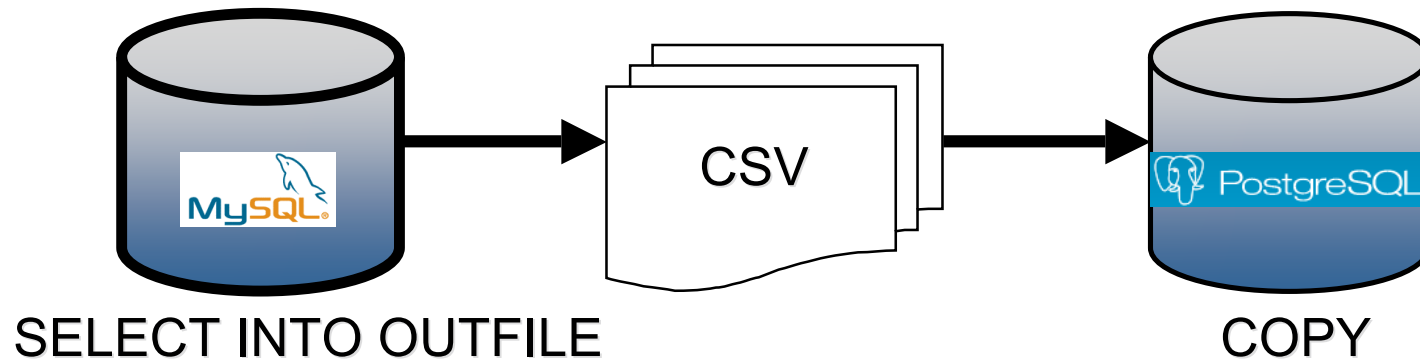
MySQL -> PostgreSQL/Oracle

- / **Provisioning**
- / **Data Type Differences**
- / **Database vs. Schema**
- / **Default (Implicitly Defined) Schema Selection**
- / **SQL Dialect Differences**
 - Statement Replication vs. Row Replication
- / **Character Sets and Binary Data**
- / **Case Sensitivity**
- / **Old Versions of MySQL**

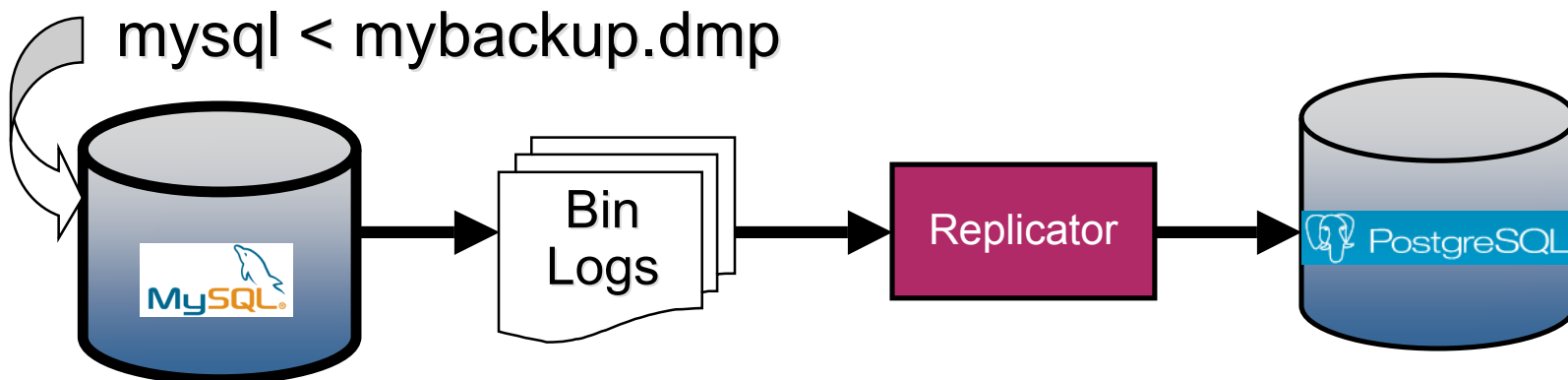


Provisioning

/ Harder way: Dump data explicitly



/ Easier way: Replicate a mysqldump backup



Data Types

/ Note the type differences between MySQL and PG

	MySQL	PostgreSQL
!	TINYINT	SMALLINT
	SMALLINT	SMALLINT
	INTEGER	INTEGER
	BIGINT	BIGINT
!	CHAR(1)	CHAR(5) = {'true', 'false'}
	CHAR(x)	CHAR(x)
	VARCHAR(x)	VARCHAR(x)
	DATE	DATE
	TIMESTAMP	TIMESTAMP
!	TEXT (diff. sizes, <i>treated as BLOB</i>)	TEXT (treated as TEXT)
!	BLOB	BYTEA
	...	



Database vs. Schema

/ In MySQL these are the same:

```
CREATE DATABASE foo
```

```
CREATE SCHEMA foo
```

/ In PostgreSQL these are very different:

```
CREATE DATABASE foo
```

```
CREATE SCHEMA foo
```

/ `<MySQL>.DatabaseMetaData.getColumns(schemaName, null, tableName, null)` vs.

`<PG>.DatabaseMetaData.getColumns(dbName, schemaName, tableName, null)`

/ Tungsten has logic and filters to rectify MySQL databases to PostgreSQL schemas



Default Schema

- / **MySQL: Trivial to use `USE`**
- / **MySQL: Going without `USE` generates different events**

	MySQL Implicit	MySQL Explicit
	CREATE SCHEMA s;	CREATE SCHEMA s;
	USE s;	
!	CREATE TABLE t (i int);	CREATE TABLE s.t (i int);
!	INSERT INTO t (1);	INSERT INTO s.t (1);

- / **PG: Extract the default schema from the event**
- / **PG: Set it before applying**

MySQL		PostgreSQL
USE s;	>	SET search_path TO s, "\$user";



SQL Dialect

- / Differences between DDL and DML statement SQL dialects
- / Row Replication resolves issues rising from differences in DML, but still leaves DDL to handle
- / Tungsten Replicator Filters come to the rescue!
 - Simple to develop Java or JavaScript extensions
 - Event structure IN -> Filter -> Event structure OUT

MySQL	PostgreSQL
<pre>CREATE TABLE complex (id <u>INTEGER AUTO_INCREMENT</u> PRIMARY KEY, i INT);</pre>	<pre>CREATE TABLE complex (id <u>SERIAL</u> PRIMARY KEY, i INT);</pre>
<pre>CREATE TABLE dt (i <u>TINYINT</u>);</pre>	<pre>CREATE TABLE dt (i <u>SMALLINT</u>);</pre>
...	



Character Sets

- / **Statement replication: MySQL syntax is “permissive”**
 - / **Embedded binary / alternate charsets**
 - / **Different charsets for different clients**
- / **Row replication: database/table/column charsets may differ**
- / **Answer: Stick with one character set throughout; use row replication to move binary data**

MySQL	PostgreSQL
<pre>INSERT INTO embedded_blob (key, data) VALUES (1, '?\0^Es\0^\0\''')</pre>	ARGH!!! (SQL statement fails)
<pre>CREATE TABLE xlate(id int, d1 varchar(25) character set latin1, d2 varchar(25) character set utf8);</pre>	ARGH!!! (no way to translate to common charset)



Case Sensitivity

/ MySQL case sensitivity differs:

- Table name case sensitivity depends on platform
- Column names are not case sensitive

/ PostgreSQL: column and table names are case sensitive

/ Oracle: column and table names upper cased (usually)

MySQL	PostgreSQL
<code>INSERT INTO test.users (firstName, familyName) ...</code>	ERROR: column "firstName" of relation "users" does not exist
<code>INSERT INTO test."users" ("firstName", "familyName") ...</code>	Success!



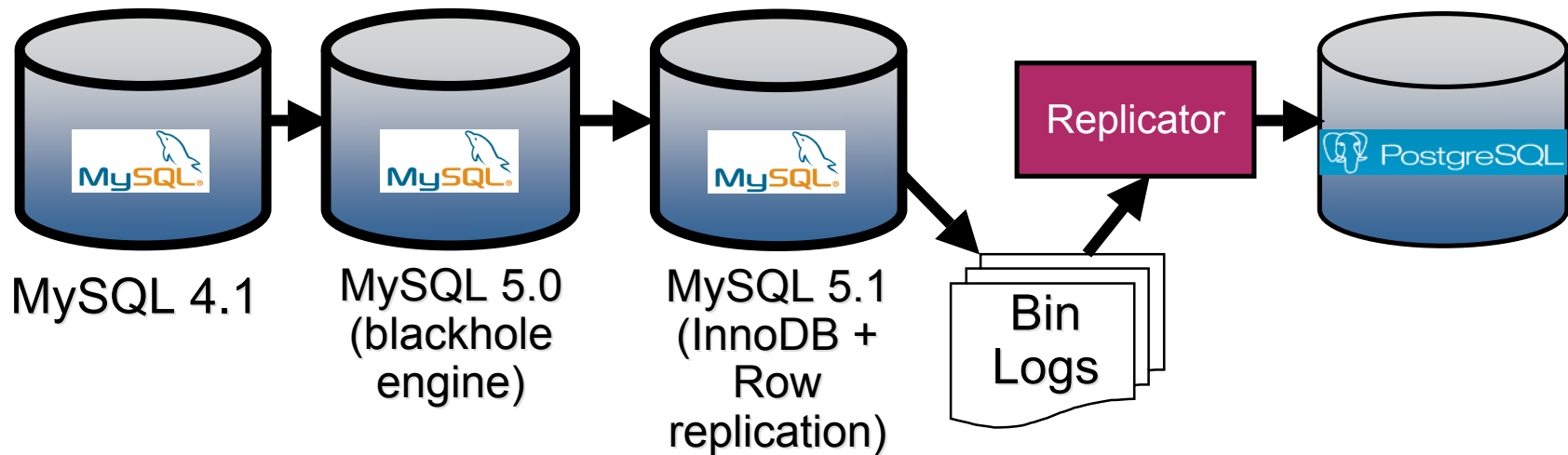
MySQL Versions

/ Problem: Data stored on hard-to-replicate MySQL versions or configurations

- Row replication not enabled (5.1)
- No row replication support (5.0, 4.1)
- Tungsten cannot read binlog (4.1)

/ MySQL blackhole replication can help

- (Blackhole = no store, just a binlog)
- Does not convert binlog records



What's Left?

- / We have covered the basics but there is more...
- / Greenplum: events must be “skinned” to not update the Distribution Key (OptimizeUpdatesFilter)
- / Oracle: Case transformation on schema and DBMS names
- / All: data transformation (more Javascript filtering)
- / Making it go **really** fast (think parallel replication)



Heterogeneous Replication Roadmap

- / **Automated installation for non-MySQL slaves**
- / **Complete type coverage**
- / **Easy-to-configure schema/data transformation filters**
- / **Replicate from other DBMS types to MySQL**

