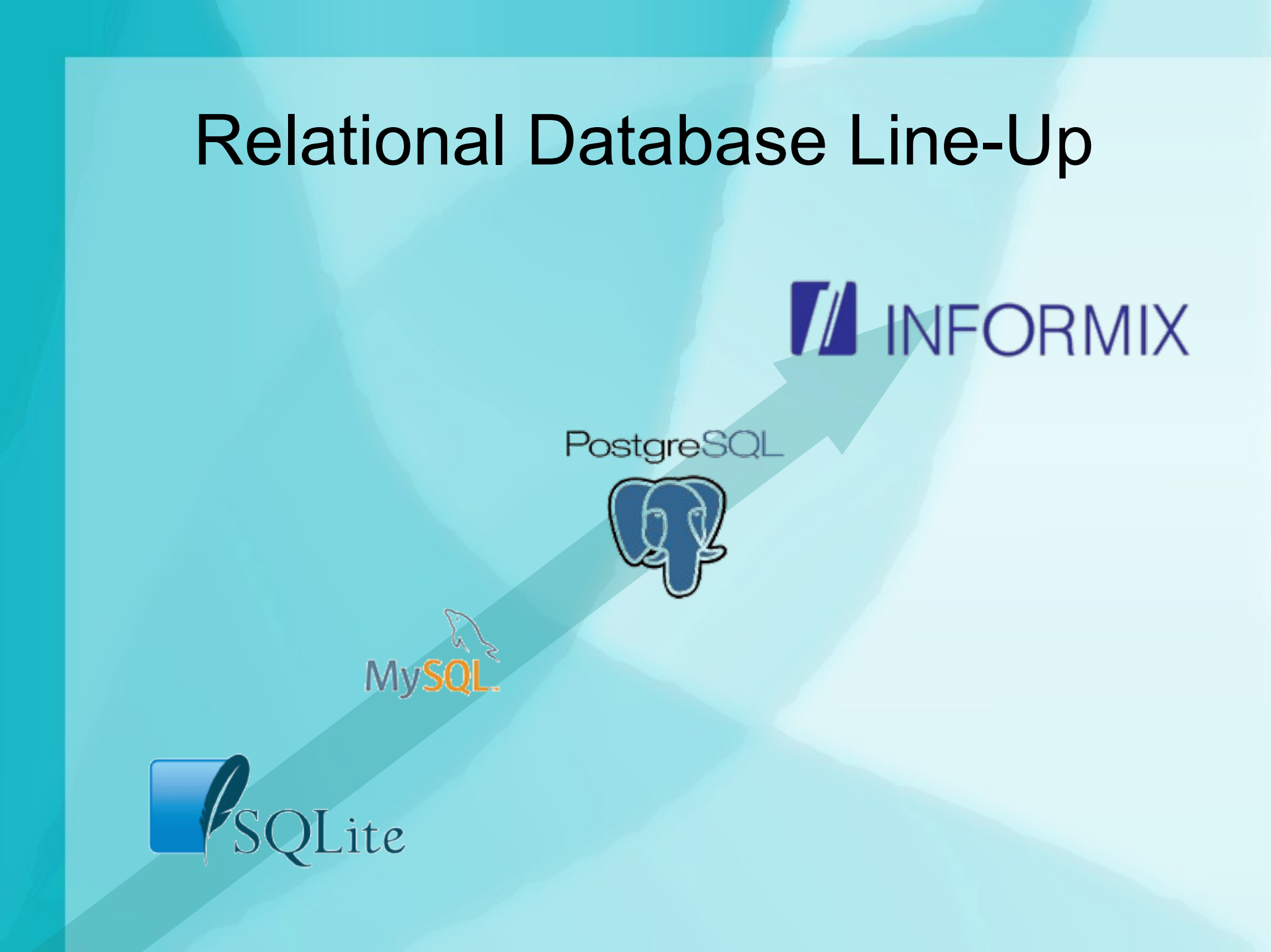
Relational Database Line-Up



SQLite: a different animal

- Engineless
  - Meant for 'local' applications.
  - Simplicity not concurrency.
  - No setup.
  - Has real limitations...which may not matter

Informix: Innovation Edition

- Free Download & License
  - http://ibm.co/oBz9kw
- Limited to 4 cores / 2GB RAM
- Unlimited Storage
- All replication options supported
- GIS “datablade” included.

Beware Benchmarks

- Benchmarks often hide more truth than they reveal.
- Performance depends on many things.
- Configuration matters
  - PostgreSQL default settings are ridiculous.
  - MySQL benchmarks often performed using MyISAM storage.

Considerations

- Licensing
- Platform
- Concurrency
- Scalability / Availability
- Query Types
- Security
- Instrumentation

Platform

- Good support in Java is ubiquitous.
- PHP has a strong MySQL bias.
- PostgreSQL well supported in most environments.
- Support for commercial engines varies.

Licensing

System	License
SQLite	Public Domain Proprietary
MySQL	GPL (CE[1]) Proprietary
PostgreSQL	PostgreSQL[2]
Informix/DB2	Proprietary

[1]Community Edition
[2]Equivalent to BSD & MIT

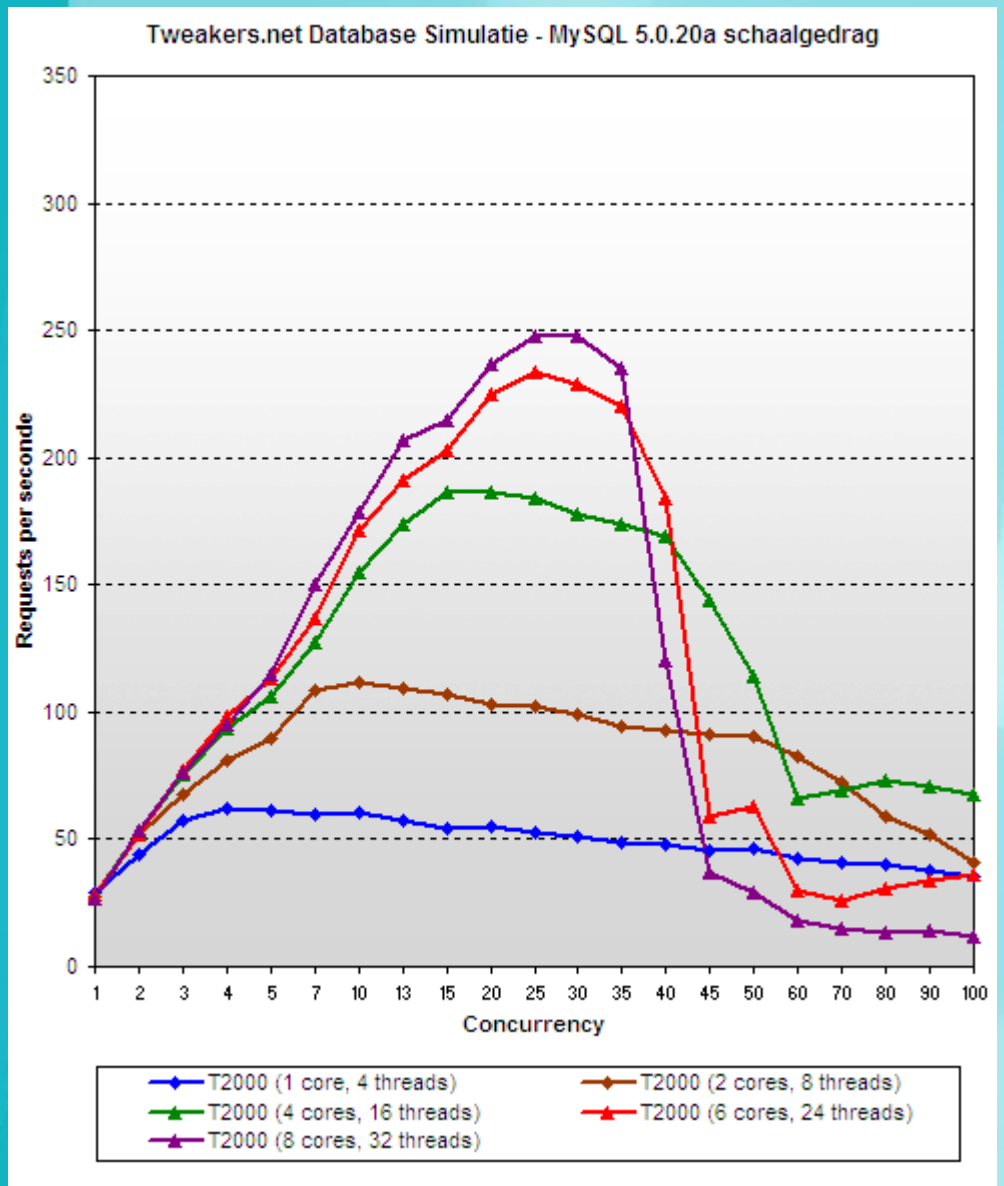
Concurrency

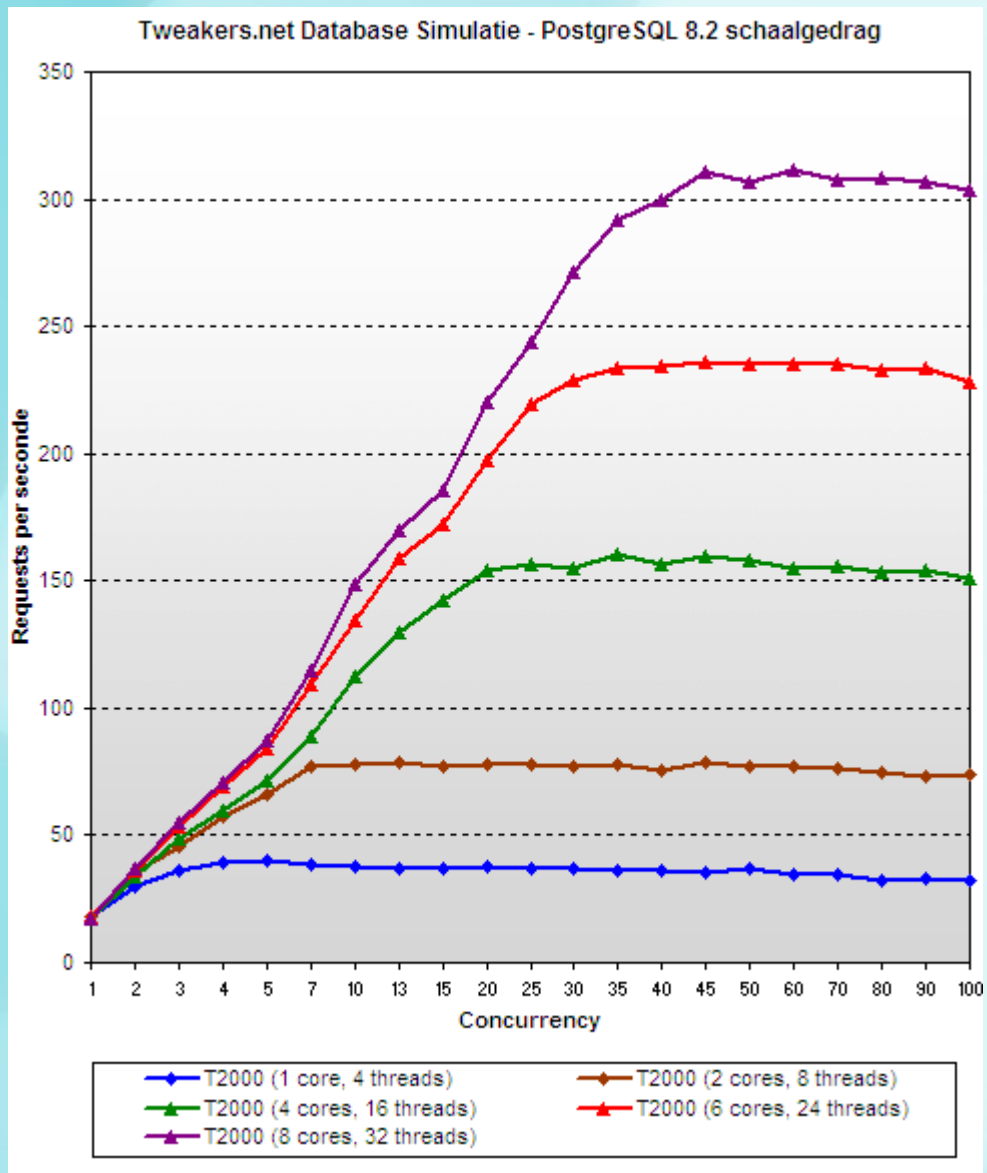
System	General	Isolation	Note
SQLite	Low (File Based)	READ COMMITED READ UNCOMMITED	Writes lock tables
MySQL	Good	READ COMMITED READ UNCOMMITED REPEATABLE READ SERIALIZABLE	Impacts performance significantly.
PostgreSQL	Excellent	READ COMMITED READ UNCOMMITED REPEATABLE READ SERIALIZABLE	
Informix/DB2	Excellent	READ COMMITED READ UNCOMMITED REPEATABLE READ SERIALIZABLE	

Isolation Level	Dirty Read	Non-repeatable Read	Phantom Read
READ UNCOMMITED	Yes	Yes	Yes
READ COMMITED	No	Yes	Yes
REPEATABLE READ	No	No	Yes
SERIALIZABLE	No	No	No

MySQL vs. PostgreSQL

Maybe; beware benchmarks, but...





Scalability / Availability

System	Failover (PITR)	Master / Slave	Master / Master	Mix
SQLite	n/a	n/a	n/a	n/a
MySQL	Yes	Yes	No	n/a
PostgreSQL	Yes	Yes	Maybe[1]	?
Informix/DB2	Yes	Yes	Yes	Yes

[1]Requires non-core components.

Query Types

- Query A is not Query B
  - “Data Warehouse” Queries
    - BIG results
    - Complex Correlations
  - Non-Relational Queries
    - Most modern database systems are hybrids.
    - Facilities for un-[or less]-structured data.

Advanced Data Storage

System	Partitioned Tales	Conditional Indexes	Functional Indexes	Query Priority
SQLite	No	No	No	No
MySQL	Yes	No	No	Very Limited
PostgreSQL	Yes (but odd)	Yes	Yes	No
Informix/DB2	Yes	Yes	Yes	Yes

Partitioned Table

MySQL

```
CREATE TABLE trb3 (id INT, name VARCHAR(50),
purchased DATE)
    PARTITION BY RANGE( YEAR(purchased) ) (
        PARTITION p0 VALUES LESS THAN (1990),
        PARTITION p1 VALUES LESS THAN (1995),
        PARTITION p2 VALUES LESS THAN (2000),
        PARTITION p3 VALUES LESS THAN (2005)
    );
```

Conditional Indexes

PostgreSQL

```
CREATE INDEX assigned_task_links
  ON obj_link(target_id)
  WHERE link_type = 'Preferred Job Executant'
```

Functional Index

Informix

```
CREATE FUNCTION toLower( name VARCHAR(255))
  RETURNS VARCHAR(255)
  WITH (NOT VARIANT);
  RETURN lower( name );
END FUNCTION;

CREATE INDEX test
  ON oemr(toLower(oe_oem_code));

SELECT *
  FROM oemr
  WHERE toLower(oe_oem_desc) = 'hyster';
```

Non-Traditional Data Types

System	"Full Text"	XML	GIS	Collections
SQLite	No	No	No	No
MySQL	Yes	As strings	Yes	ENUM SET (String)
PostgreSQL	Yes	Limited	Yes	ARRAY (Any) ROW / Structure[1] **ENUM**
Informix/DB2	Yes	Yes	YES	LIST[2] SET[3] MULTISET[4] ROW / Structure[1]

[1]A compound data-element.
[2]Elements are not unique and ordered
[3]Elements are unique and unordered
[4]Elements are not unique and unordered.

Non-Traditional Data-Types

Complex Type: PostgreSQL

```
CREATE TYPE inventory_item AS (
  name            text,
  supplier_id     integer,
  price           numeric);

CREATE TABLE inventory(item inventory_item, count
integer);

INSERT INTO inventory
VALUES (ROW('fuzzy dice', 42, 1.99), 1000);

CREATE TABLE orders
(order_id INT, items inventory_item[]);
```

PostgreSQL HSTORE

http://www.postgresql.org/docs/current/static/hstore.html

- Supports key value data as a field type.

```
CREATE TABLE user_profile (
    user_id INT NOT NULL PRIMARY KEY,
    profile HSTORE
);

CREATE INDEX user_profile_hstore ON user_profile USING GIN (profile);

INSERT INTO user_profile(user_id, profile)
  VALUES ( 5, hstore('"Home City"=>"San Francisco","Occupation"=>"Sculptor"');

SELECT akeys(profile) FROM user_profile
WHERE user_id = 5;

UPDATE user_profile SET profile = profile - 'Occupation' WHERE user_id = 5;

SELECT user_id FROM user_profile
WHERE (profile->'Occupation') IS NOT NULL;
```

Key-Values can be indexed.

Remove the “Occupation” key.

Security

System	Integrated Authentication[1]	LBAC	Field Encryption[1]	Version Control[1]
SQLite	n/a	n/a	No	No
MySQL	No	No	via methods	Possible
PostgreSQL	Yes	No	via methods (pgcyrpto)	Possible
Informix/DB2	Yes	Yes	via methods	Yes

[1]Integrated authentication, auditing, and/or data encryption may be a regulatory requirement for certain data elements under HIPPA, FERPA, PCI/DSS. All US corporations must implement a data-retention policy.

Instrumentation

- Everyone supports EXPLAIN
  - Use it!
  - PostgreSQL EXPLAIN output as JSON, YAML, or XML.
- MySQL & PostgreSQL both have excellent statistics.
  - You still have to go look.
  - Documentation available, but can be thin.
- Commercial instrumentation tends to be more “automated”.
  - Better supported by the tool-chain.