DZone

THE DZONE GUIDE TO

Databases

Speed, Scale and Security

VOLUME IV

BROUGHT TO YOU IN PARTNERSHIP WITH

Couchbase

Datical

DBmaestro
DevOps for Database

FairCom

influxdata

MariaDB

redgate
ingeniously simple

redislabs
home of redis

Studio 3T

DEAR READER,

Data. Everything we do today generates more of it. Whether it's taking a ride in Uber or Lyft, adjusting the lights in your home with your Alexa, or sharing that family video on Facebook, we're creating data at an accelerating rate every day. With that, 90% of the Internet has been created since 2016. But this isn't a Big Data guide – it's a guide focused on databases, one of the primary places where we have to put all the data that we're generating.

Welcome to the *2017 DZone Guide to Databases*. This year, we'll take you on a guided tour looking at how to make your databases run faster, scale more easily, and more effectively secure the data that you store in them. At DZone.com, our Database Zone has been growing steadily, fueled by a resurgence of interest in relational databases, the emergence of SQL layers on other types of data stores, and the need to manage your databases like you manage the other parts of your infrastructure. This growth is driven largely by a convergence of database content with many of the different other topics we cover. In this guide, you'll find articles coverage database + DevOps, database + security, database + cloud, and database + agile.

Traditionally, changes to database technology have come slowly compared to the other components of modern web development. Businesses depend on databases to reliably store their data – everything else, including ease of use, come after that. Recently though, we've seen a relatively rapid evolution of the database space as the need to handle larger and ever-changing data has emerged. To be sure, the increase in adoption of DevOps practices has also had a significant impact on the pace at which databases and database technology evolve. When you deploy daily or even weekly, long gone are the days where you can wait for the DBA to make adjustments to your tables or to settings on the server. Today, you need to be able to apply changes to your database through version control and integrate your database directly into your CI and CD pipelines.

This move to DevOps is also driving the need to be more secure. With more potential changes being made to the data, more changes being deployed to the applications that leverage this data, and more bad actors out there, database security needs to be more flexible and offer a higher degree of comfort to the teams responsible for the safety of your data. I believe that this is a key area where we'll see AI step in and prove its worth.

Thank you for downloading the *2017 Guide to Databases: Speed, Scale, and Security*. We're proud of this one and I hope you enjoy reading it as much as we enjoyed making it.

BY MATT SCHMIDT
PRESIDENT, DZONE

TABLE OF CONTENTS

PRODUCTION

Chris Smith
DIRECTOR OF PRODUCTION

Andre Powell
SR. PRODUCTION COORDINATOR

G. Ryan Spain
PRODUCTION PUBLICATIONS EDITOR

Ashley Slate
DESIGN DIRECTOR

Billy Davis
PRODUCTION ASSISSTANT

MARKETING

Kellet Atkinson
DIRECTOR OF MARKETING

Lauren Curatola
MARKETING SPECIALIST

Kristen Pagàn
MARKETING SPECIALIST

Natalie Iannello
MARKETING SPECIALIST

Miranda Casey
MARKETING SPECIALIST

Julian Morris
MARKETING SPECIALIST

BUSINESS

Rick Ross
CEO

Matt Schmidt
PRESIDENT

Jesse Davis
EVP

Gordon Cervenka
COO

SALES

Matt O'Brian
DIRECTOR OF BUSINESS DEV.

Alex Crafts
DIRECTOR OF MAJOR ACCOUNTS

Jim Howard
SR ACCOUNT EXECUTIVE

Jim Dyer
ACCOUNT EXECUTIVE

Andrew Barker
ACCOUNT EXECUTIVE

Brian Anderson
ACCOUNT EXECUTIVE

Chris Brumfield
SALES MANAGER

Ana Jones
ACCOUNT MANAGER

Tom Martin
ACCOUNT MANAGER

EDITORIAL

Caitlin Candelmo
DIRECTOR OF CONTENT AND COMMUNITY

Matt Werner
PUBLICATIONS COORDINATOR

Michael Tharrington
CONTENT AND COMMUNITY MANAGER

Kara Phelps
CONTENT AND COMMUNITY MANAGER

Mike Gates
SR. CONTENT COORDINATOR

Sarah Davis
CONTENT COORDINATOR

Tom Smith
RESEARCH ANALYST

Jordan Baker
CONTENT COORDINATOR

Anne Marie Glen
CONTENT COORDINATOR

Want your solution to be featured in coming guides?
Please contact research@dzone.com for submission information.

Like to contribute content to coming guides?
Please contact research@dzone.com for consideration.

Interested in becoming a dzone research partner?
Please contact sales@dzone.com for information.

Special thanks to our topic experts, Zone Leaders, trusted DZone Most Valuable Bloggers, and dedicated users for all their help and feedback in making this guide a great success.

Executive Summary

BY MATT WERNER
PUBLICATIONS COORDINATOR, DZONE

Database technology is one of the absolute requirements for software development. Most developers learn how to incorporate databases fairly early on in their training. While database technology doesn't traditionally make evolutionary leaps compared to other areas of the industry, such as hardware or cloud computing, there are shifts occurring between relational and NoSQL, the cloud and on-premise, persistence, DevOps, and security. To find out how they were using database technology, we asked 528 software professionals to share their thoughts and organizational practices to share with you in this guide.

RELATIONAL DATABASES ARE STILL KING

DATA

The most common databases used in production are MySQL (48%), Oracle (45%), Microsoft SQL Server (32%), PostgreSQL (32%), and MongoDB (28%).

IMPLICATIONS

While NoSQL databases have been on the market for years now, the most popular choices for enterprise development are the old guard of relational databases. MySQL being the most popular is hardly surprising considering its popularity with developers of all skill levels.

RECOMMENDATIONS

Regardless of your skill level, chances are you're already familiar with at least one of these popular databases, probably MySQL. It'll be important to keep your skills sharp, as there doesn't seem to be an end to their usefulness inside and outside of enterprise development.

NOSQL STILL INTRIGUES DEVELOPERS

DATA

32% of respondents plan to adopt new data persistence technology in the next six months. The most commonly considered databases are MongoDB (38%), Cassandra (32%), Neo4j (22%), and Couchbase (17%).

IMPLICATIONS

Developers who are considering new database technologies are looking to non-relational or NoSQL databases first, since so many are already familiar at least with MySQL.

RECOMMENDATIONS

First, know the use case that needs to be met for your application. Will storing data as JSON files be more efficient for your application than a standard table? Second, a significant number of people use more than one database for their applications (72%), so it's entirely possible and reasonable to add a NoSQL database without replacing your current infrastructure. DZone's infographic, "All 'Bout Database, Database, NoSQL" highlights the differences between the four main kinds of NoSQL databases: document, graph, key-value, and column store.

THE CLOUD CATCHES ON

DATA

Developers using cloud storage increased from 17% in 2016 to 24% in 2017, and those using hybrid storage grew from 16% in 2016 to 21% in 2017. The use of on-premise storage decreased to 47% from 62% a year ago. Those using Oracle or Microsoft SQL Server in production are more likely to say that data is stored on premise compared to those using PostgreSQL, MySQL, or MongoDB.

IMPLICATIONS

It seems that database technology in general is slower to move on new trends and technologies than areas like DevOps or Cloud technologies. Though the concept of cloud computing has been around for a long time, databases are only making the move slowly, particularly Oracle and SQL Server enterprises, compared to more flexible, open source or free databases like MySQL and MongoDB.

RECOMMENDATIONS

Cloud-based databases are probably the simplest way to easily scale your database, since you're using hardware as a service that you can add to when needed. It also allows you to save time by letting your provider handle security (as long as they're actually handling it!) and reduces the workload on your DBA. If any of these have been problems, consider adding a database-as-a-service to your app or moving existing databases to the cloud.

Key Research Findings

BY G. RYAN SPAIN
PRODUCTION COORDINATOR, DZONE

DEMOGRAPHICS

528 software professionals completed DZone's 2017 Database Technology survey. Respondent demographics are as follows:

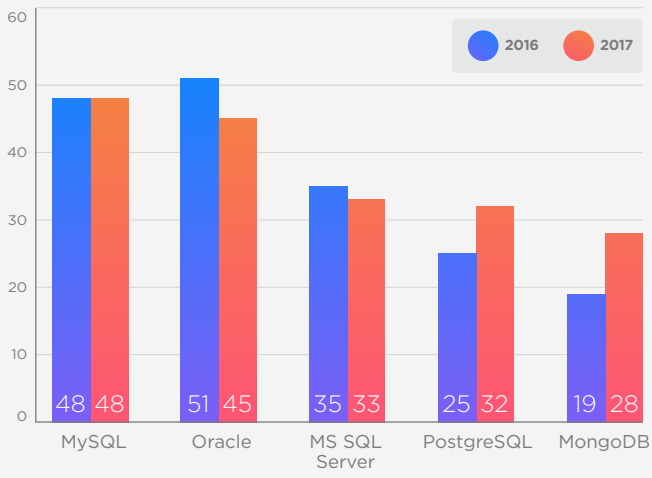
- 38% of respondents identify as developers or engineers, 17% identify as developer team leads, and 15% identify as software architects.
- The average respondent has 13 years of experience as an IT professional. 52% of respondents have 10 years of experience or more; 16% have 20 years or more.
- 38% of respondents work at companies headquartered in Europe; 33% work in companies headquartered in North America.
- 21% of respondents work at organizations with more than 10,000 employees; 19% work at organizations between 1,000 and 10,000 employees; and 25% work at organizations between 100 and 1,000 employees.
- 82% develop web applications or services; 48% develop enterprise business apps; and 23% develop native mobile applications.

TRENDS IN THE DATABASE TOOLING LANDSCAPE

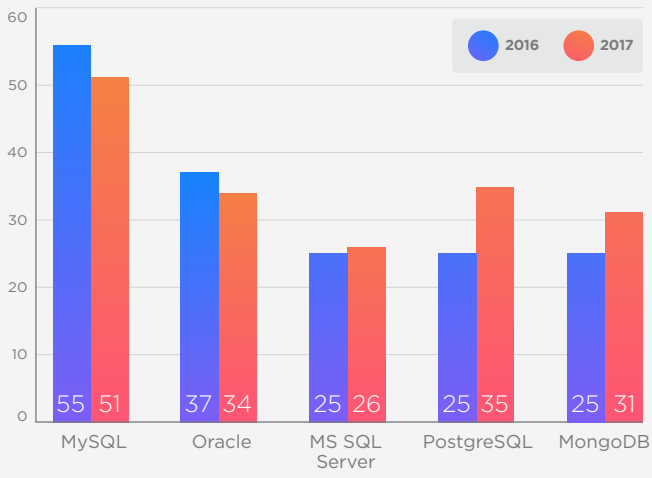
Responses in this year's Database Technology survey revealed some shifting trends in the database tooling landscape. For databases used in production environments, respondents who said they use Oracle DB dropped 6% since the 2016 survey (51% to 45%); while respondents using MySQL in production remained unchanged from last year (48%), this drop in Oracle usage put MySQL as the #1 production database this year. Usage of Microsoft SQL Server, the third most popular production database among survey respondents this year and last, dropped slightly (35% to 33%), while PostgreSQL, still in fourth place from last year, saw a 7% increase in usage, from 25% to 32%, nearly tying it with MS SQL Server. At fifth place again this year, MongoDB, the most popular non-relational database among survey respondents, saw a dramatic increase in adoption, jumping from 19% in 2016 to 28% in 2017.

For non-production database usage, MySQL remains in first place this year, though its non-production usage decreased from 55% in 2016 to 51% in 2017. A 10% increase in non-production PostgreSQL usage, from 25% to 35%, puts Postgres in the number two position this year, surpassing Oracle DB, which dropped from 37% last year to 34% this year. Last year's fourth place database in non-production environments was a tie between MS SQL Server and MongoDB at 25%; this year, MongoDB usage increased to

What database management systems do you use in production?



What database management systems do you use in non-production environments?



31%, giving it the definitive fourth place position over MS SQL Server's 26% for this year's survey.

SQL VS. NOSQL

As suggested by the previous section, non-relational databases in general do not see the popularity that relational databases do. In both production and non-production environments, MongoDB is the only non-relational database that broke the top 5. The next most popular non-relational database among survey respondents was wide column store Apache Cassandra, with 12% usage in production (up from 7% in 2016) and 13% usage in non-production environments (up from 8% in 2016). NoSQL databases Couchbase, Neo4j, Elasticsearch, and Redis each had 2% of responses for in-production use. Outside of production, Neo4j had 6% of responses, and Couchbase had 4%. While 96% of respondents use at least one relational database, only 50% of respondents use at least one non-relational DB, and only 40% use at least one in production. Still, NoSQL adoption is up from last year; the 2016 database survey showed 30% of respondents using a non-relational database in production, and 40% using at least one in all environments. 32% of respondents said they are planning to adopt a new database technology within 6 months, and of those responses, MongoDB (38%) and Cassandra (32%) were the two most popular choices. Respondents were inclined to prefer relational databases over non-relational, however, with 84% of respondents who expressed a database preference when asked what database they most enjoy using selecting a SQL database. Finally, the majority of respondents (55%) said they spend greater than 75% of their database-related development time working with SQL databases.

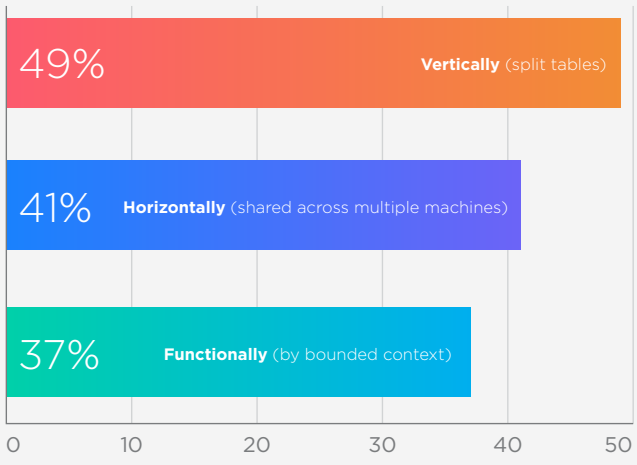
POLYGLOT PERSISTENCE

Last year's database survey saw 40% of respondents using a single persistent storage model in a typical application. This year, more respondents said they use two persistent storage models for a typical application (31%) than just one (30%), and 16% of respondents say they use 3 storage models. Most respondents also work mainly with persistent data; 42% of respondents said less than 10% of data they work with is ephemeral, and another 19% said that only 11% to 25% of the data they work with is ephemeral. The use of different databases has also slightly increased. This year, respondents said they use an average of 3.1 distinct DBMSes, compared to 2.9 in 2016.

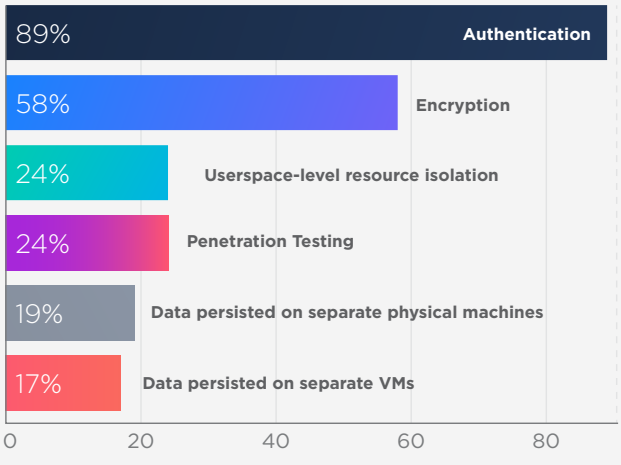
CLOUD STORAGE

Database management systems and storage models aren't the only technologies affecting the database landscape. The increasing popularity of cloud storage is also changing the ways data is stored and accessed. Cloud storage adoption has increased rather significantly over the past year, with 21% of respondents saying the data they work with is typically stored in a hybrid cloud storage model (up from 16% in 2016) and 24% of respondents saying they typically work with data in the cloud (up from 17% in 2016). On-premise storage decreased from 62% in 2016 to 47% in 2017. 55% of survey respondents said they believe that the cloud has made working with databases easier (33% had no opinion). Respondents using commercially licensed databases Oracle and MS SQL Server in production, however, were much more likely to say they typically work with on-premise data (58% and 56%, respectively) versus respondents who use open-source databases PostgreSQL, MySQL, and MongoDB (43%, 40%, and 31%, respectively).

How do you partition your database?



What security protocols do you implement to protect the data in the databases you are working with?



Bringing Continuous Delivery to the Database

BY **MATT HILBERT**

TECHNOLOGY WRITER, **REDGATE SOFTWARE**

QUICK VIEW

01 Companies and organizations are realizing that a siloed approach, where application and database development are two separate processes, hinders their ability to stay competitive.

02 Continuous Delivery enables a process that speeds up development, reduces errors, and allows new features to be released faster.

03 Rather than adding to your existing infrastructure with unfamiliar processes and enforced policies, Continuous Delivery for the database can be implemented alongside the systems already in place.

Databases are often seen as the poor relative in software development. The clunky bits at the backend that store the data while the front-end applications do their magic. The grey boxes that are updated at the conclusion of the software development process and often cause problems.

It's irritating to admit it, but also true. I've even heard of a Database Administrator (DBA) being presented with a pile of change scripts at the last minute and having to wade through them, line by line, while the developers fume impatiently for their latest feature to be pushed live.

It doesn't have to be that way.

Companies and organizations are realizing that this siloed approach, where application development and database development are two separate processes, is hindering their ability to stay competitive. Instead, they need to be considered as two sides of the same coin: separate but also connected.

It's beginning to happen, too. The 2017 State of Database DevOps report from Redgate revealed that in 75% of the 1,000 companies and organizations which took part, developers were responsible for both application and database development. Perhaps surprisingly, developers also built the database deployment scripts in 60% of respondents.

So, database development is moving from the dark silo of the DBA into the application development environment. When Visual Studio 2017 was launched, the Enterprise Edition also included, for the first time, third-party database development tools in the installer, so application developers can write database code in the IDE they're already comfortable working with.

When it comes to Continuous Delivery, however, a slightly different picture emerges. The Database DevOps report showed that 81% of developers use version control for their application code, 39% use Continuous Integration, and 33% have automated deployments to ease the process. It also disclosed that 47% of respondents have already adopted elements of Continuous Delivery across some or all of their IT projects, and a further 33% plan to do so in the next two years, all of which looks promising. Continuous Delivery enables a process to be established that speeds up development, reduces errors, and allows new features to be released faster:

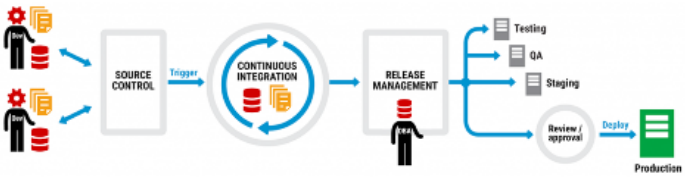
Version controlling code means one source of truth is maintained in development, preventing conflicts later on. Each time changes are committed to version control, a Continuous Integration process is triggered that builds and tests the changes, so errors are picked up immediately. Automated deployment tools then compile the code and create a release artifact that can be reviewed before it's deployed.

This kind of workflow is increasingly being adopted, and version control systems like Team Foundation Server, Subversion, and Git are commonplace, as are Continuous Integration tools like Jenkins, TeamCity, and Bamboo.

There's good reason, too. As the 2017 State of DevOps Report from DORA and Puppet shows, companies and organizations which embrace DevOps practices like Continuous Delivery can typically deploy changes, updates, and improvements 46 times more frequently. Their change failure rate is also 5 times lower, and they can recover from failures when they do occur 96 times faster.

But – and this is a big but – the Database DevOps report goes on to reveal that database development is lagging behind in Continuous Delivery. Here, 58% of developers use version control, and it falls to 20% and 21% for Continuous Integration and automated deployments, respectively. So, a lot of companies and organizations are missing out on the advantages of Continuous Delivery for the database.

And the advantages are there to be gained. Including the database makes it possible to move from big, infrequent, and often troublesome releases to smaller, regular releases that are virtually error-free. Importantly, if you introduce it in the right way, it can make both the application and database development process smoother:



As can be seen, rather than adding to your existing infrastructure with unfamiliar processes and enforced policies, Continuous Delivery for the database can be implemented alongside the systems already in place.

That said, what are the steps you need to take, and what do you need to watch out for in order to introduce Continuous Delivery to database development?

INCLUDE YOUR DATABASE IN VERSION CONTROL

42% of companies out there still aren't version controlling their database. Some probably rely on a backup of the production database when they need the latest version of the database, which is cumbersome at best. Others may have a folder of handwritten change scripts which they present to the DBA at deployment time (remember the poor DBA I mentioned earlier).

With the database in version control, it's a different story. The development team can reconstruct any version of the database, and migrate from one version to another, from the source-controlled database object scripts. That's a big plus-point because changes can be reverted at any time, and errors can be tracked back to specific changes.

Remember that everything required to recreate a working version of the database should be in version control. That's the database creation scripts, configuration scripts, schema objects like tables and stored procedures, reference data, and migration scripts.

There are many proven database version control tools available, but it's advisable to choose one which integrates with and works alongside the infrastructure you already have in place. Introducing database version control will mean developers have to change the way they work. Limiting that change and letting them work with the tools they're already accustomed to and like will make it easier to implement.

INCLUDE YOUR DATABASE IN CONTINUOUS INTEGRATION

In application development, it's becoming increasingly common for developers to commit changes to version control, at which point a Continuous Integration server will build it and run automated unit tests to see if anything is broken. Developers see immediately if there are any errors and, because they've just been working on the code and know what changed, it's easier to fix.

With the database in version control, you're able to include it in the build process as well. When changes are committed to version control, the build process can create a new database, create its objects, load any reference data, and check that the SQL syntax is valid and dependencies are in order. This not only highlights problems earlier, it also stops breaking database changes from ever reaching production where the impact can cause far bigger problems.

There are, again, mature development tools available to introduce Continuous Integration for the database that plugs into the systems used in application development. This makes it easier to introduce and implement and, importantly, lets you run automated tests across both the application and database, ensuring the tests are working against the latest version of each.

INCLUDE YOUR DATABASE IN RELEASE MANAGEMENT

Release management tools like TeamCity, Visual Studio Team Services, and Octopus Deploy are becoming more and more popular in application development because they automate even the most complicated deployments.

There are similar tools available for the database which automatically generate deployment scripts to update the database, and include static data and migration scripts that specify how to deploy complex database changes, such as table splits.

Choose one that works with your existing build and release management tools and you'll complete your Continuous Delivery journey by making deployments repeatable and reliable rather than error-prone and uncertain.

Matt Hilbert is a technology writer at Redgate Software with 20 years' experience working for lots of the world's biggest tech companies – and many of the smallest. He has a particular fascination for emerging technologies and is on a continuing mission to decompile, untangle, and explain techspeak to a wider audience, and excite people about the endless possibilities technology offers.

DevOps Meets Database

Software is reshaping every industry – from healthcare to finance to retail and beyond. The pace of change is accelerating, and companies must master their application release process in order to survive in the digital era. In the software economy, innovation and speed are the new standards.

As companies have adopted automation to help increase the pace of software delivery, a new constraint has emerged: data. The database has long been a unique, separate discipline, not part of the DevOps tool chain. In fact, most companies—even those are using the latest DevOps automation tools—are still managing and deploying database changes manually, anchoring development teams.

That's why we developed Datical DB. Our database release automation solution is designed to break the application release logjam created by today's manual database deployment processes. Datical offers a great opportunity to improve productivity and performance, allowing development, test, and DBA staff to focus on more important projects and initiatives. Database release automation also helps to eliminate otherwise unavoidable incidents of human error, while improving data security, application performance, and reliability. Put simply, database release automation from Datical helps speed the delivery of better performing applications into production faster, safer, and with more reliability.

Datical is strategically guiding some of the world's most admired companies through their database deployment modernizations efforts and delivers business innovation through database automation. Learn more here: A Guide to Digital Transformation, Enterprise Applications and the Database.

WRITTEN BY BEN GELLER
VICE PRESIDENT MARKETING, **DATICAL**

PARTNER SPOTLIGHT

Datical

Our goal was to decrease the release cycle of our applications from 5 weeks to 2. That was impossible because it took more than 2 weeks just to do the database portion.

CATEGORY
Continuous Delivery and Release Automation

NEW RELEASES
Every 3 weeks

OPEN SOURCE
No

CASE STUDY

While one of the world's leading entertainment and media companies had been successful in bringing DevOps Automation to many of the manual processes in modern software development, the same was not true for the data underpinning the application. Aligning database changes to the application changes they support was a manual process that added considerable friction to the application release process and slowed the pace of innovation. The DevOps team knew they had to automate their database deployment process alongside their application release process in order to achieve optimal DevOps. The team now uses Datical to bring DevOps to the database. As a result they have been able deliver innovation faster by decreasing the release cycle of their applications from 5 weeks to 2 weeks.

STRENGTHS

- Treat database code just like application code. Create database Continuous Delivery by unifying application and database changes.
- Capture DBA expertise in programmatic rules that can be enforced during the application build and release process.
- Simulate the impact of database changes before they are deployed
- Automatically track and report the status of every database deployment across the enterprise.

NOTABLE CUSTOMERS

- GE Transportation
- Sony
- AIG
- NBCUniversal
- Humana
- LabCorp

WEBSITE datical.com | **TWITTER** @Datical | **BLOG** datical.com/blog

QUICK VIEW

01 You should use change management tools to maintain your database schema

02 There are great open-source change management tools available

03 It will make your life easier, especially when you are using continuous integration.

Schema Migration for Databases

BY **BLAINE CARTER**

DEVELOPER ADVOCATE FOR OPEN SOURCE, **ORACLE**

It is generally accepted that you should always use proper version control when developing software. Why wouldn't you do the same with your database changes?

The question is, how do you manage changes to the database? The quick answer is, "write .sql scripts and check them into the version control system." As usual, things aren't always as simple as they first seem. Maintaining "build from scratch" scripts in version control is easy enough, but what do you do when you need to modify an existing database object?

THE HARD WAY

One method I've used in the past is to maintain two sets of scripts, each set in a different directory: one for new installations and one for upgrades.

NEW INSTALLATION SCRIPTS

In this directory, I have scripts for creating database objects. I prefer to have a single script handle a single object such as a table, view, or stored procedure. I also include separate scripts that will populate the default data in the master tables.

UPGRADE FROM VERSION X TO Y

In this directory are the scripts that will alter the database objects, upgrading them from one version to the next. When needed, this will include any data migration scripts.

In both directories, there is a master script that runs the change scripts in the correct order. Each script *should* contain validation logic to ensure they only run when they are supposed to and they raise any errors if something goes wrong.

While this approach can work, it's not perfect and there are some problems that are difficult to address.

- How sure can we be that everything ran?
- If there's a problem somewhere in the process, how do we rollback the changes?
- Can I run some scripts only for testing and not in production?

Each of these problems has the same answer: Write more scripts.

USE A SCHEMA CHANGE MANAGEMENT (SCM) TOOL INSTEAD

Typically, an SCM (sometimes called a migration tool) will function like a combination of "the hard way" directories above. Every change you make is defined in a change log. You'll start with the very first "create table" change log and add a new change log for all changes made from then on, such as adding or dropping a column, index, or constraint. The SCM tool will keep track of what has or hasn't been run and implement only the new changes. If you're doing

a brand-new install, it will start at the beginning and run them all.

There are a lot of great SCM tools available, and many of them are open source.

If you're using a programming framework in your application, such as Python's Django or Ruby on Rails, it may have built-in SCM tools. My first real experience with an SCM tool was a few years ago. I was writing a small Ruby on Rails application using Active Record, and since then, I have been a strong advocate for SCM tools.

I love exploring many different programming languages — which can make it difficult to get very deep in any language specific SCM tool. Fortunately, there are also standalone SCM tools you can use such as Flyway and Liquibase. Both of these tools are open-source and well-supported.

Most SCM tools work with many different databases — most of which implement slightly different versions of SQL. The tools will use their own object definitions in a common format such as JSON, XML, YAML, and others. The SCM tool will translate these definitions into the correct SQL for the database you're deploying your changes to. For my examples, I will be using an Oracle Database, but the same definitions should work with many different databases.

EXAMPLES

I've used a few of these tools, some more than others, but the one I'm most familiar with is Liquibase, so I'll use it for the following examples. Usually, the Liquibase examples you'll find online are written in XML, but I find XML to be a bit too pointy so I'll be using JSON.

INSTALLATION AND CONFIGURATION

In my database, I've created a schema called lb_demo to use with the examples. If you run these examples, **make sure you use a schema that is safe to experiment in.** Locate (or download) the JDBC driver for your database. I'll be using the Oracle driver ojdb7.jar.

Liquibase refers to the database changes as "change sets" and the files that contain them as "change log files." A change log master lists the change logs in the order that they will be run in.

Create the following directories and files:

```
-LiquibaseExample
      |- liquibase.properties
      |- changelog
            |-  db.changelog-master.json
            |-  db.changelog-1.0.json
            |-  db.changelog-2.0.json
```

Liquibase is "just" a Java application, so you don't actually need to do anything more than download the zip file, extract it, and run it. (Make sure your Java is version 1.6 or higher. You can run it directly from Java:

```
java $JAVA_OPTS -jar /opt/liquibase/liquibase.jar
--driver=oracle.jdbc.OracleDriver
--classpath="/home/example/jdbcDrivers/ojdbc7.jar"
--url=jdbc:oracle:thin:lb_demo/password@dbaccess
--changeLogFile=changelog/db.changelog-master.json
update
```

The Liquibase download includes a shell script and batch file that can be used in place of running the .jar file. You can simplify your process by including the extracted Liquibase directory in your path. For these examples, I will be using the shell script. To make it a little easier, I will include the parameters from above in a properties file.

Open liquibase.properties and add the following text substituting the values for your configuration:

```
#Liquibase.properties
driver: oracle.jdbc.OracleDriver
#your database driver
classpath: /home/example/jdbcDrivers/ojdbc7.jar
#jdbc driver location
url: jdbc:oracle:thin:lb_demo/password@dbaccess
#jdbc connect string
changeLogFile: changelog/db.changelog-master.json
#master change log location
```

EXAMPLE 1: CREATE TWO TABLES

Open db.changelog-master.json and add the following JSON:

```
{
  "databaseChangeLog": [{
      "include": {
        "file": "changelog/db.changelog-1.0.json"
      }
    },
    {
      "include": {
        "file": "changelog/db.changelog-2.0.json"
      }
    }
  ]
}
```

The first change set creates a table called `lb_groups` with three columns: `id`, `name`, and `description`.

Open db.changelog-1.0.json and add the following JSON:

```
{
  "databaseChangeLog": [{
    "preConditions": [{
      "runningAs": {
        "username": "lb_demo"
      }
    }]
  }, {
    "changeSet": {
      "id": "Two-Table-1",
      "author": "BlaineCarter",
      "comment": "Add table lb_groups",
      "changes": [{
        "createTable": {
          "tableName": "lb_groups",
          "columns": [{
              "column": {
                "name": "id",
                "type": "int",
                "autoIncrement": true,
                "constraints": {
                  "primaryKey": true,
                  "nullable": false
                }
              }
            },
            {
              "column": {
                "name": "name",
                "type": "varchar(50)",
                "constraints": {
                  "unique": true,
                 "uniqueConstraintName": "uk_lb_groups_name"
                }
              }
            },
            {
              "column": {
                "name": "description",
                "type": "varchar(200)"
              }
            }
          ]
        }
      }]
    }
  }]
}
```

The second change set creates a table called `lb_people` with four columns: `id`, `firstname`, `lastname`, and `group_id`. A foreign key to the `lb_groups` table is created using group_id.

Open db.changelog-2.0.json and add the following JSON:

```
{
  "databaseChangeLog": [{
    "preConditions": [{
      "runningAs": {
        "username": "lb_demo"
      }
    }]
  }, {
  "changeSet": {
```

Code continued

```
      "id": "Two-Table-2",
      "author": "BlaineCarter",
      "comment": "Add table lb_people",
      "changes": [{
        "createTable": {
          "tableName": "lb_people",
          "columns": [{
              "column": {
                "name": "id",
                "type": "int",
                "autoIncrement": true,
                "constraints": {
                  "primaryKey": true,
                  "nullable": false
                },
              }
            },
            {
              "column": {
                "name": "firstname",
                "type": "varchar(50)"
              }
            },
            {
              "column": {
                "name": "lastname",
                "type": "varchar(50)",
                "constraints": {
                  "nullable": false
                },
              }
            },
            {
              "column": {
                "name": "group_id",
                "type": "int",
                "constraints": {
                  "foreignKeyName": "groupFK",
                  "references": "lb_groups(id)"
                },
              }
            }
          ]
        }
      }]
    }
  }]
}
```

Most of the properties here are fairly self-explanatory and similar to what you'd use in an SQL script to create a table. Let's take a look at some of the extra properties.

Liquibase uses Id and author combined with the file name and package to uniquely identify a change set. The Id and author values can be any string you'd like, but you should use values that will have meaning in your process.

The preConditions array is a list of conditions that Liquibase will check before running the changes. In this example, I'm making sure that I'm connected to the correct schema "lb_demo" before adding the tables. The linked documentation has a long list of other conditions you can use.

And finally, the comment value will be included in the Liquibase databasechangelog table (discussed below) and in the documentation generated by Liquibase.

RUN IT

Warning: When you run Liquibase, it executes the changes you defined. **There is no "Are you Sure?"** prompt and there are not separate commits.

Run this command: `liquibase update`

Connect to your database with your favorite SQL tool and inspect the changes. If this is the first time Liquibase has been run, you should have two new tables that Liquibase uses to control and log the changes: `databasechangeloglock` and `databasechangelog`. Take a look at the `databasechangelog` table to see what data Liquibase tracks for the changes. You should also see the two new tables, `lb_groups` and `lb_people`, defined in the change sets above.

EXAMPLE 2: ROLL BACK CHANGES

Liquibase includes the ability to automatically roll back certain changes. Let's add a column and roll it back.

First add a new file "db.changelog-3.0.json" to the changelog directory.

```
{
  "databaseChangeLog": [{
    "preConditions": [{
      "runningAs": {
        "username": "lb_demo"
      }
    }]
  }, {
    "changeSet": {
      "id": "RollbackTag-3",
      "author": "BlaineCarter",
      "comment": "Add rules column to lb_groups table",
      "tagDatabase": {
        "tag": "ver-1"
                "columns": [{
            "column": {
              "name": "rules",
              "type": "varchar(255)"
            }
          }]
        }
      }]
    }
  }]
}
```

Include the new file at the end of db.changelog-master.json.

```
    {
      "include": {
        "file": "changelog/db.changelog-3.0.json"
      }
    }
```

Run the update: `liquibase update`

Connect to your database and verify that the column "rules" has been added to the `lb_groups` table. Look in the `databasechangelog` table to see that the new change has been logged.

Liquibase has the option to roll back to a date, a tag, or a certain number of changes. When you roll back changes, they are rolled back in the reverse order that they were run. Run the following command to roll back the last change: liquibase rollbackCount 1

Connect to your database and verify that the column "rules" has been removed from the lb_groups table. Look in the `databasechangelog` table to see that the record for change log 3 has been removed. Liquibase only tracks changes that have been applied to the database — when you roll back a change, it is no longer in the log.

OTHER SCHEMA CHANGE MANAGEMENT FEATURES

- Branching and merging.
- Reverse-engineer your current database into change logs.
- Compare differences between schema.
- Load data from a change set or .csv file.
- Auto-generate schema documentation.
- Use labels and contexts to conditionally include or exclude specific changes.

SCHEMA CHANGE MANAGEMENT VS. "THE HARD WAY"

These examples were just a small taste of what you can do with a good SCM tool. Let's compare them to the way I used to do it.

Both methods produce a similar number of files. The SCM method only requires maintaining the changes in one place instead of files in two different directories.

The .json files are a bit more verbose than the .sql scripts but if you're like me, I find JSON to be more readable.

The SCM method automatically keeps track of what has run and only runs the new changes.

The SCM method gives you the ability to automatically rollback most changes.

Perhaps the biggest difference is the ease of using and SCM tool in a continuous development pipeline versus trying to manage a large directory of .sql files. This alone should be worth exploring a schema change management option.

Blaine Carter For most of my career, I've used Oracle tools to build applications; mostly SQL and PL/SQL. Lately, I've been exploring open-source projects and used quite a few languages and liked almost all of them. My combined experience with Oracle Database and open source led me to a brand new position at Oracle: Oracle Developer Advocate for Open Source. I help database developers improve their workflow using open-source tools and I promote the use of open source inside and outside of Oracle.

HIGH AVAILABILITY

SEAMLESS SCALING

Redis Enterprise for
Real-Time Streaming Data

redislabs
home of redis

redislabs.com

Real-Time Data Streaming with Redis

Ingest of streaming data in large volume and high velocity presents architectural challenges for big data use cases. Timely and accurate data-driven decisions are core to the fields of IoT, e-commerce, security, communications, entertainment, finance, and retail.

Redis, the in-memory database, delivers over a million read/write operations per second, with sub-millisecond latency on a modest-sized server, making it the most resource-efficient for large volumes of data. It offers built-in data structures such as Lists, Sets, Sorted Sets, and Hashes for efficient data processing. It also supports messaging services and client libraries in all popular programming languages.

Some techniques to perform fast data ingest using Redis include:

1. **Pub/Sub**: This relatively simple solution uses Redis' Pub/Sub feature, which allows applications to publish and subscribe to messages.
   Pros: Easy, suits geo distributed services
   Cons: Is connection oriented, publishers don't resend messages
2. **List**: In this solution, the producer pushes every message to a queue(represented as a List data structure), and the subscriber pulls new messages from the queue as they arrive.
   Pros: Reliable to connection loss
   Cons: Stores multiple copies of the data, the data is lost when consumed
3. **Sorted Set**: The messages are stored, sorted by their timestamps or in the order in which they were received. The consumers pull data by querying the Sorted Set data structure.
   Pros: Stores only one copy of the message, is reliable to connection loss
   Cons: Complex to implement, consumes more space

For more detailed information, download the whitepaper, Redis for Fast Data Ingest.

WRITTEN BY ROSHAN KUMAR
SR. PRODUCT MANAGER, **REDIS LABS**

PARTNER SPOTLIGHT

Redis Enterprise

redislabs home of redis

Redis Enterprise enhances open source Redis with stable high performance, true high availability, and seamless scaling

CATEGORY
In-memory database platform

NEW RELEASES
Typically, a major release in about two years, and minor releases every few months

OPEN SOURCE
Yes

CASE STUDY

Xignite, Inc. provides cloud-based financial market data APIs to help enterprises deliver financial market data to their websites and apps in real-time. Redis Enterprise, with the capacity to handle massive volume of financial market data with extremely low latency, is a critical technology layer for Xignite.

Xignite chose Redis for its versatile data structures and commands making it the simplest, fastest way to crunch the data. They deployed Redis Enterprise in their production for maximum throughput, scalability, always-on availability, and automatic failover.

The low operational hassle associated with Redis Enterprise ensures that the Xignite team can focus on development issues and not worry about Redis.

STRENGTHS

- High performance with sub millisecond latency supporting over a million writes per second
- Built-in data structures, message broker with commands optimized for faster resultsSimulate the impact of database changes before they are deployed
- Shared nothing cluster architecture with always-on availability, effortless scaling
- High throughput by running multiple Redis instances on multiple cores
- 24x7 support backed by expertise in managing and scaling over 250K Redis databases in production

NOTABLE CUSTOMERS

- DreamWorks
- eHarmony
- Overstock.com
- General Mills
- Home Depot

WEBSITE redislabs.com | **TWITTER** @RedisLabs | **BLOG** redislabs.com/blog

Monitoring MySQL

QUICK VIEW

01 MySQL is one of the most common open sourced databases used today.

02 A working and performant database is key to application performance.

03 Monitoring your database is crucial for maintaining performance.

04 There are several key database metrics you should be monitoring.

BY **MICHAEL COBURN**
PRODUCT MANAGER, **PERCONA**

INTRODUCTION

The Internet permeates our entire culture: we buy things online, take classes online, consume media using applications and websites, and communicate and keep in touch with people via social media and email on websites and applications. Databases contain all the raw data that applications and websites use to assemble a user-friendly access method for online and application interaction.

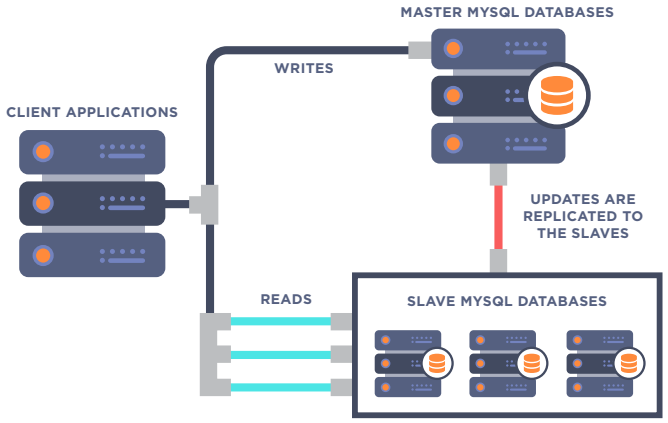
Databases are the backbone of the our modern Internet. There are many database options out there, but one of the most ubiquitous and most used database is MySQL.

WHAT IS MYSQL?

MySQL is the world's most popular free and open-source relational database. It is now over 20 years old. As an open-source software product, it is free to download, use, and reuse. Much of the development is done in the open-source community (outside of the company that is generally responsible for MySQL: Oracle).

MySQL is a fast, stable, multi-user, multi-threaded SQL database server, generally compliant with ANSI SQL 99. It represents the "M" in LAMP stack (and many other stack models). It is used by small and large websites, including Facebook, Google, YouTube, and Twitter.

A single instance of MySQL uses replication to update the data it contains. Replication is the concept of a master and slave using a binary log to copy database changes. Changes made to the master are written to the binary log, and then these logs are used to propagate changes to the slaves connected to the master.



Another important feature of MySQL (and any database, really) is high availability. This means guaranteeing that the database is working and available for applications and websites to query as much as possible (preferably, "always"). In high-traffic environments or under certain conditions, databases can get either overloaded or cut off from communication. This causes applications and websites to cease working or to slow down to the point that the user experience is greatly affected.

For MySQL, high availability is achieved via:

- Floating IP address between dual masters (so that if one goes down or becomes unresponsive, another takes over)

- Using a proxy, such as:
  - HAproxy — TCP only (layer 4)
  - ProxySQL — MySQL protocol aware (layer 7)
- Using other software technologies, like Galera Cluster:
  - This software has multiple writeable masters and virtually asynchronous replication
  - Galera Cluster itself is open-source and has been incorporated into various software options such as:
    - MariaDB Galera Cluster
    - Percona XtraDB Cluster

"Replication is the concept of a master and slave using a binary log to copy database changes. Changes made to the master are applied to the binary log, and then these logs are used to propagate changes to the slaves connected to the master."

WHY MONITOR MYSQL?

Your MySQL database is one tier in your typical website or application, and it is where the data is safely persisted to disk. The database tends to be the "black box" for most operations people: queries go in, results come out.

Monitoring, as a whole, is aimed at demystifying what metrics are key to your MySQL database performance. If you don't monitor objective metrics like CPU, load, or queries per second, you cannot make predictions.

Predictions allow you to provision your environment correctly to ensure the best performance for your customer user case.

For example, you might want to predict:

- The saturation point of a single server — or how long can we continue to increase traffic at the current rate before MySQL becomes a bottleneck?
- What are the deviations from a specified benchmark — or how much work are you doing right now vs. how much is the most you can feasibly deliver?
  - Benchmarks are the key to determining your theoretical database maximum load.

You want your database platform to be accessible by many types of users in your operations, engineering, and even business units:

- DBAs care about:
  - Low-level metrics on a per-MySQL basis.
  - Cluster-level metrics such as: How is replication? How is cluster membership in Percona XtraDB Cluster? Is flow control occurring?
  - Which queries are running in my database server? How often are they running? Which are the slowest performing queries? What can I do to improve the performance?
- Application developers care about:
  - What is the performance of my newly released query? What can I do to improve the performance, and does it need an index or a re-write?
  - What is the performance of the database tier in the application stack?
- Business managers care about:
  - Is the database tier presenting a bottleneck to user response time? Is the database tier generating errors in my application?

WHAT TO MONITOR IN MYSQL

So, what should you be monitoring in MySQL? There are many ways to answer this question, depending on what you are trying to do. However, for general performance issues, there are a few fairly standard metrics you can use to verify that your MySQL performance is being maintained:

- Relationship of Threads_connected to Max_connections
  - As you approach Max_connections, your database will stop responding to new connections from your application.
    - Your website or application will go down.
- Threads_running as a factor of the count of cores in your database server
  - As your workload increases, you should pay attention to the ratio of Threads_running vs. the count of cores.

  - Note that this ratio depends on how quickly your queries execute: if you are operating an OLTP application with ~100 milliseconds or faster query response time, you can generally sustain a higher average Threads_running because your queries enter and leave the innoDB kernel more quickly.
  - Conversely, with an OLAP system and queries > 1 second, you will find that you cannot sustain much more than a 1:1 ratio of Threads_running to count of CPU cores as subsequent queries will be waiting in the queue to execute.

- Uptime < 1800
  - This is an indication that your database server has recently restarted.
  - Restarts should be a rare occurrence, and always should be investigated.

- Replication lag
  - MySQL replication works based on a master writing database changes to a local file called the Binary Log.
  - This Binary Log is copied down to slave(s) (your Read Replicas) and written into the slave's Relay Log.
  - The slave's Relay Log is then read and database changes applied locally.
  - This all occurs in an asynchronous fashion, which by design can lead to a Slave being "behind" in replication — events that occurred on the master at Time 0 may not be applied on the slave until Time 5.
  - The implication is that if you write a record to the master and immediately trigger a read from a replica, you might not yet see this data. Therefore, you want to pay attention to the slave's server variable Seconds_behind_master and ensure you don't read from a Slave that is "too far behind" (this is dependent on your particular application's needs).

"Monitoring, as a whole, is aimed at demystifying what metrics are key to your MySQL database performance. If you don't monitor objective metrics like CPU, load, or queries per second, you cannot make predictions."

PERCONA MONITORING AND MANAGEMENT

Percona Monitoring and Management (PMM) is one way to monitor your database. It is a free, open-source database monitoring and performance optimization tool for MySQL and MongoDB. The PMM software is built on other open-source projects (for example, Grafana, Prometheus, and Consul) that are deployed in your environment on your equipment (not a SaaS).

PMM provides secure communications using SSL between clients and server. It has the following features:

- QAN for MySQL
  - Observe the queries consuming the most amount of time in MySQL
  - Visualize query characteristics such as:
    - Query response time, query count
    - Rows sent vs. rows examined
    - InnoDB operations: reads, waits
    - Exposes MySQL query plan information via EXPLAIN in table and JSON format so you can evaluate the level of optimization
    - Review the CREATE TABLE, SHOW TABLE STATUS, and SHOW INDEXES for your query so you can make appropriate tuning modifications (index addition, column type changes, etc.)

- Metrics Monitor
  - Visualize activity over time
    - System-level resources: CPU, Load, Memory
    - MySQL: Queries Per Second (QPS), replication, storage engine-specific (for example, InnoDB, Aria, MyISAM)
    - Cluster: Galera Cluster, Percona XtraDB Cluster
    - ProxySQL overview

CONCLUSION

Monitoring your MySQL database allows you to understand what is happening as MySQL processes queries, provides insight into performance and what affects performance, and allows you to make informed predictions about what is needed in your database environment.

Michael Coburn joined Percona in 2012 and is the Product Manager for PMM (Percona Monitoring and Management). Michael's prior roles at Percona include MySQL Principal Architect, Consulting Manager, Technical Account Manager, and Technical Support Engineer."

Diving Deeper

INTO DATABASE

TOP #DATABASE TWITTER ACCOUNTS


@lukaseder


@KimberlyLTripp


@BrentO


@PaulRandal


@sqlgirl


@SQLEspresso


@felipehoffa


@_AlexYates_


@sql_williamd


@dmcghan

DATABASE-RELATED ZONES

Database dzone.com/database

The Database Zone is DZone's portal for following the news and trends of the database ecosystems, which include relational (SQL) and non-relational (NoSQL) solutions such as MySQL, PostgreSQL, SQL Server, NuoDB, Neo4j, MongoDB, CouchDB, Cassandra and many others.

Big Data dzone.com/bigdata

The Big Data/Analytics Zone is a prime resource and community for Big Data professionals of all types. We're on top of all the best tips and news for Hadoop, R, and data visualization technologies. Not only that, but we also give you advice from data science experts on how to understand and present that data.

Cloud dzone.com/cloud

The Cloud Zone covers the host of providers and utilities that make cloud computing possible and push the limits (and savings) with which we can deploy, store, and host applications in a flexible, elastic manner. The Cloud Zone focuses on PaaS, infrastructures, security, scalability, and hosting servers.

TOP DATABASE REFCARDZ

An Overview of GraphQL

As an alternative to REST, GraphQL is quickly gaining popularity as a tool for building APIs. This Refcard introduces GraphQL concepts such as the GraphQL type system and schema, and demonstrates how to build a GraphQL service that connects to a database using JavaScript.

NoSQL and Data Scalability

This Refcard covers the range of NoSQL database types available today, whether NoSQL is right for you, and several real-life use cases for using a NoSQL database.

Getting Started With Redis

In this Refcard, you will get started on your Redis journey with a simple guide to installing and utilizing Redis, a complete glossary of data types, and the various commands to operate in Redis with detailed examples.

TOP DATABASE PODCASTS

Graphinista

player.fm/series/graphistania-podcast-for-neo4j-graph-database-community

This podcast from the Neo4j graph database community features interviews with a variety of database experts.

Voice of the DBA

player.fm/series/voiceofthedbas-podcast

Look at the world from the perspective of a data professional.

Dear SQL DBA

player.fm/series/dear-sql-dba

In this style of an advice segment, this podcast answers the burning questions of SQL Server database administrators.

TOP DATABASE RESOURCES

Learn SQL: Codecademy

In this completely free and interactive course, learn the basics of database fundamentals, including tables, queries, aggregate functions, and constructing advanced database queries.

Database Design for Mere Mortals

by Michael J. Hernandez

In this free introductory course, learn the fundamentals of artificial intelligence, building intelligent agents, solving real AI problems with Python, and more.

Database Lesson #1 of 8: Introduction to Databases

Learn about the reasons for using a database, the components of a database system, the elements of a database, the purpose of database management systems, and the functions of a database.

In the Age of Customer Experience, Isn't it Time for a New Database?

If you still harbor any doubts about the power of digital disruption, or whether it's going to transform your business, consider this one simple fact: Today's S&P 500 is swapping out old companies for new at the mind-boggling rate of one every two weeks.

Digital disruptors in the mold of LinkedIn, Uber, and eBay are reinventing one business model after another, and they're doing it first and foremost by revolutionizing the customer experience. As a result, the ability to deliver exceptional customer experiences has become the single most important competitive differentiator in practically every industry.

The problem is, traditional databases simply weren't built to deliver the fast, seamless, and personalized experiences that modern customers demand. Old-school analytical systems are backwards-facing, while transactional systems are focused only on the moment of purchase. But, today's customer engagements depend on a lengthy and complex sequence of interactions that need to be executed flawlessly in real time.

That's why Couchbase specifically designed the world's first and only Engagement Database to let any business – from startup to enterprise – create and execute brilliant customer experiences every time. The Engagement Database responds instantly to fast-changing data, market factors, and business demands. It scales quickly and easily to handle spikes and rapid growth without loss of performance or outages. And it's incredibly flexible and easy to manage, even with limited resources. Now, you can be next to start a revolution.

WRITTEN BY RAVI MAYURAM
SVP OF ENGINEERING , CTO, **COUCHBASE**

PARTNER SPOTLIGHT

Couchbase Data Platform

Couchbase

From digital transformation to customer engagement – start a revolution with the world's first Engagement Database

CATEGORY
NoSQL Engagement Database

NEW RELEASES
Annual

OPEN SOURCE
Yes

CASE STUDY

In a world where customer expectations are constantly rising, businesses have to deliver continuously amazing digital experiences if they want to stay on top. That's why hundreds of leading companies from nearly every industry count on Couchbase to solve their toughest digital innovation challenges.

Couchbase designed the world's first Engagement Database with the most powerful NoSQL technology, so that any organization can drive digital innovation with ease. Designed to be a joy to work with, only Couchbase delivers unparalleled performance at any scale, while also providing the unmatched agility and manageability that today's businesses require to continually reinvent their customer experiences and thrive with an adaptive competitive edge.

STRENGTHS

- JSON with N1QL at any scale – Leverage existing SQL skills with N1QL (SQL for JSON)
- Memory-first speed – Rich data access, in-memory replication, 99.999% availability
- Security across the stack – Encrypt, audit, protect, and secure your data anywhere
- Cloud-native for global scale – Cross datacenter replication (XDCR) in any cloud
- Fully synced and offline-first – Real-time data sync even when users are offline

NOTABLE CUSTOMERS

- AT&T
- Cisco
- General Electric
- Marriott
- eBay

WEBSITE couchbase.com | **TWITTER** @couchbase | **BLOG** blog.couchbase.com

Majority Report: NoSQL and the Age of the Data Heist

QUICK VIEW

01 Past predictions of the future data-scape were wrong

02 Privatized data theft is becoming an epidemic

03 We rely on privateers to take down the pirates, but it's a community-wide responsibility

BY **RICHARD FOX COLLINS**
CHEIF MARKETING OFFICER, **STUDIO3T**

It's good to see a whole series of Philip K Dick's short stories being dramatized for television. 'Electric Dreams' is a multi-director anthology of short films celebrating the imagination of the Paranoiac-in-Chief. But for a writer who spent more time than most worrying about data, there is one data trick that neither he, nor Orwell, nor Huxley (not to mention Wells and Kafka before them) ever really entertained: data theft for purely criminal gain. Instead, all of them imagined data abuse in the hands of controlling elites, not outlaw, hole-in-the-wall geek gangs. Half-time score: 20th Century Paranoia – 0, The Future – 1. And it's shaping up to be a fascinating second half.

So maybe we should celebrate the fact that liberal free market capitalism has democratized the business of data theft and abuse? Surely an occasional infection of privatised data hacking is preferable to Big Brother calling all the shots from a gimongous MPP server somewhere in the Arctic Circle. As it is, anyone with a rig and a copy of the Hacker's Handbook can go dig, with the modern-day equivalent of the Gold Rush sluice box. Though, as Nic Roeg imagined in 'Eureka', this particular rush of gold moves fast. The Data Rush is indeed a rush of a different kind.

THROUGH DATA DARKLY

Gold was for the '49ers a natural resource, static, waiting to be found and fashioned and flogged with a bit of added value. Data is really none of these things. Data is what we humans are in the process of becoming. It is important that we don't allow the process to play itself out to an unanticipated and uncomfortable end-point. We want future generations to grow into something more than a resource whose relative preciousness is evaluated by a commodities exchange algorithm.

Data is also a fluid, but with hard bits in it. The recent cack-handed announcement by Equifax of their July failure in which 143 million US citizens' personal data was stolen, reminds us once again of the curious substance that data is; almost as curious as the human condition. It's fluid and constantly on the move, being transformed into different data-types, being indexed in a cascade of useful and pointless ways, being deleted, merged and updated. Addresses change, names get ditched and adopted, job titles and passwords are a Brownian mist of randomness in constant flux. But Social Security Numbers, now they're not in flux at all. SSNs are like the white blood cells in your data bloodstream, those DNA-carrying immune system sleuths that check invaders for passports. Similarly, your SSN (or National Insurance number in the UK) is a permanent ID stamp and ensures you maintain appropriate access to support services. If anything, SSNs are even less likely to mutate than your genome is.

THE SCALE OF THE DATA BREACH EPIDEMIC

The Identity Theft Resource Center is one of many organizations dedicated to tracking and reporting on the ever

increasing volumes of data breaches. Their mid-year report for 2017 showed a nearly 30% increase in breaches over 2016. Their projection is for 1,500 by year's end. The Breach Level Index is another tracker that estimates over 5 million data records are stolen every day.

> We want future generations to grow into something more than a resource whose relative preciousness is evaluated by a commodities exchange algorithm.

Until we can all catch up with doughty Estonia and get all of our citizens' private data encrypted in a publicly distributed blockchain, data breaches are not an occasional and annoying infection. They are more like a radically mutated Spanish influenza that puts the health of our digital world at huge risk. The blockchain investment by certain Baltic states is a brilliant pre-emptive strike to protect their citizens' identities by hiding in plain sight. Maximum shared visibility of the fact of being on the ledger means it's much, much harder for the fact of your citizenship to be challenged or undermined.

In September, and for the second time this year, a massive ransomware attack was hurled at unsecured instances of the MongoDB document store platform. Attacks quickly spread to a range of other NoSQL platforms including CouchDB and Cassandra, and in October, not for the first time, MySQL was also targeted. Back in January of 2017, when the first big wave of bitcoin ransom demands rolled across the NoSQL data-scape, Victor Gevers and his colleagues at G.D.I. took steps.

HACKERS SANS FRONTIERES

G.D.I is a non-profit organization fighting to protect the freedom of communication across the internet and provide help to victims who need their data back. Hospitals and other health organizations are not more vulnerable than other institutions but the potential damage caused by a data breach is orders of magnitude greater. In conversation with bleepingcomputer.com, Gevers characterised the latest wave of ransomware as a potentially alarming refinement over January:

"The amount of (new) attackers went down compared with the beginning of the year, but the destructive reach (in regards to victims) per attack went up in numbers. So it looks like there are fewer attackers but with a larger impact."

MongoDB has a straightforward security checklist that can keep your data safe from ransomware attacks. In addition, using a visual MongoDB IDE like Studio 3T to manage your data security, permissions, users and authentication protocols, makes the process that much more visible and shared. It's a lot harder to overlook a gaping security hole when it's staring not just you in the face but everyone on your team.

Thankfully, Gevers and his team have gone the extra mile in making recommendations on improved security for the NoSQL platform and MongoDB have announced imminent improvements.

"Beginning with development release version 3.5.7, localhost-only binding is implemented directly in the MongoDB server, making it the default behavior for all distributions. This will also be incorporated into our upcoming production-ready 3.6 release."
SecurityWeek's Ionut Arghire

And then there's another thing the community as a whole can do to help stem the data bleed. Microsoft MVP Troy Hunt writes and trains globally on several topics, including data security. He makes the valid point that calling out poor corporate responses to a data breach is not sufficient. Equally important is to give public kudos to organizations that get it right. It's easier to follow a good example than to avoid a bad one.

"We all jumped on "the Equifax dumpster fire bandwagon" recently and pointed to all the things that went fundamentally wrong with their disclosure process. But it's equally important that we acknowledge exemplary handling of data breaches when they occur because that's behaviour that should be encouraged."
Disqus demonstrates how to do Breach Disclosure Right

THE COMMUNITY STRIKES BACK

Philip K Dick's original 'Minority Report' short story (the Tom Cruise movie version is significantly different) described a Cold War world where data management, precognitive modelling and subsequent abuse of that data, were all in the hands of an authoritarian state. The world as it has actually turned out is thankfully far less authoritarian than Dick predicted. But it's also far more diverse and complex. Gevers and his G.D.I colleagues do us all a very big service by reminding us that building our shared immunity to data hacks is also a shared obligation. And as the ubiquity of data theft spreads daily, theirs is truly the majority report.

Richard Fox Collins is currently Chief Marketing Officer at 3T Software Labs in Berlin. He has worked in database and motion graphic technologies for over twenty years and is a regular documentary producer for BBC Radio 4.

The Return of Relational (with JSON)

There was a time when developers and architects had to choose between relational and non-relational databases. While relational databases provide strong consistency, non-relational databases deliver flexibility and scalability. However, the line between relational and non-relational databases is blurring as consistency and flexibility/scalability are no longer mutually exclusive.

While leading non-relational databases have yet to add transactions and schema constraints, a number of relational databases have added support for semi-structured data. MariaDB Server 5.3 introduced dynamic columns, enabling different rows to have different columns. MariaDB Server 10.2 introduced a comprehensive set of SQL functions for reading, writing and querying JSON documents stored in columns.

MariaDB Server provides true flexibility, the ability to support both structured and semi-structured data with the same database. The data can be structured or semi-structured, or it can be partially structured and partially semi-structured. While structured data can be modeled as columns, semi-structured data can be modeled as JSON documents. For example, relational models for fixed data can be extended with JSON documents to support variable data.

In addition, MariaDB Server includes a distributed storage engine (Spider) for read, write and storage scalability, distributing native partitions across multiple database servers. With JSON functions and Spider, MariaDB Server provides consistency, flexibility and scalability. MariaDB TX, built on MariaDB Server, includes the world's most advanced database proxy (MariaDB MaxScale) and administrative tools for managing MariaDB Server deployments.

WRITTEN BY SHANE JOHNSON
SENIOR DIRECTOR OF PRODUCT MARKETING, **MARIADB CORPORATION**

PARTNER SPOTLIGHT

MariaDB TX

MariaDB

Born of the community. Raised in the enterprise. It's the result of engineering leadership and community innovation – a modern database for everyone.

CATEGORY
DBMS

NEWEST RELEASE
MariaDB TX 2.0, May 2017

OPEN SOURCE
Yes

CASE STUDY

Red Hat developed Red Hat Single Sign-On (based on Keycloak), and chose to run it on MariaDB TX because they needed scalability and high availability. The solution required a resilient architecture based on hybrid cloud infrastructure to prevent user sessions from being interrupted in the event of a major outage. MariaDB TX, with synchronous, multi-master replication (MariaDB Cluster), enabled Red Hat to create a multi-site, hybrid-cloud solution by deploying a database cluster with nodes running in the datacenter and on public cloud infrastructure. With a full open source stack from the OS to the database to the middleware, Red Hat created a single sign-on solution to meet modern scalability and high availability demands.

STRENGTHS

- Supports JSON and GeoJSON with a comprehensive set of SQL functions
- Includes the world's most advanced database proxy, MariaDB MaxScale
- Features best-in-class security with a built-in firewall and data masking
- Provides complete HA/DR with replication, clustering and restore/rollback
- Includes advanced SQL: common table expressions and window functions

NOTABLE CUSTOMERS

- DBS Bank
- Deutsche Bank
- Red Hat
- Verizon
- O2

WEBSITE mariadb.com

TWITTER @mariadb

BLOG mariadb.com/resources/blog

I'M ALL 'BOUT Database, Database NoSQL

Most of us are familiar with relational databases. In fact, when we surveyed 528 DZone members for our Database survey, we found most of you were already familiar with MySQL, Oracle, PostgreSQL, and MS SQL Server. However, we also saw that there was a lot of interest in exploring NoSQL (not only SQL) databases like MongoDB, Cassandra, Neo4j, and Couchbase. As these databases may not be as familiar to you, we wanted to showcase the four major kinds of NoSQL databases, how they store data, and why they might help you knock your next project out of the park.

FACTOID 1

The most popular databases to use outside of production are MySQL (51%), PostgreSQL (35%), Oracle (34%), and MongoDB (31%)

FACTOID 2

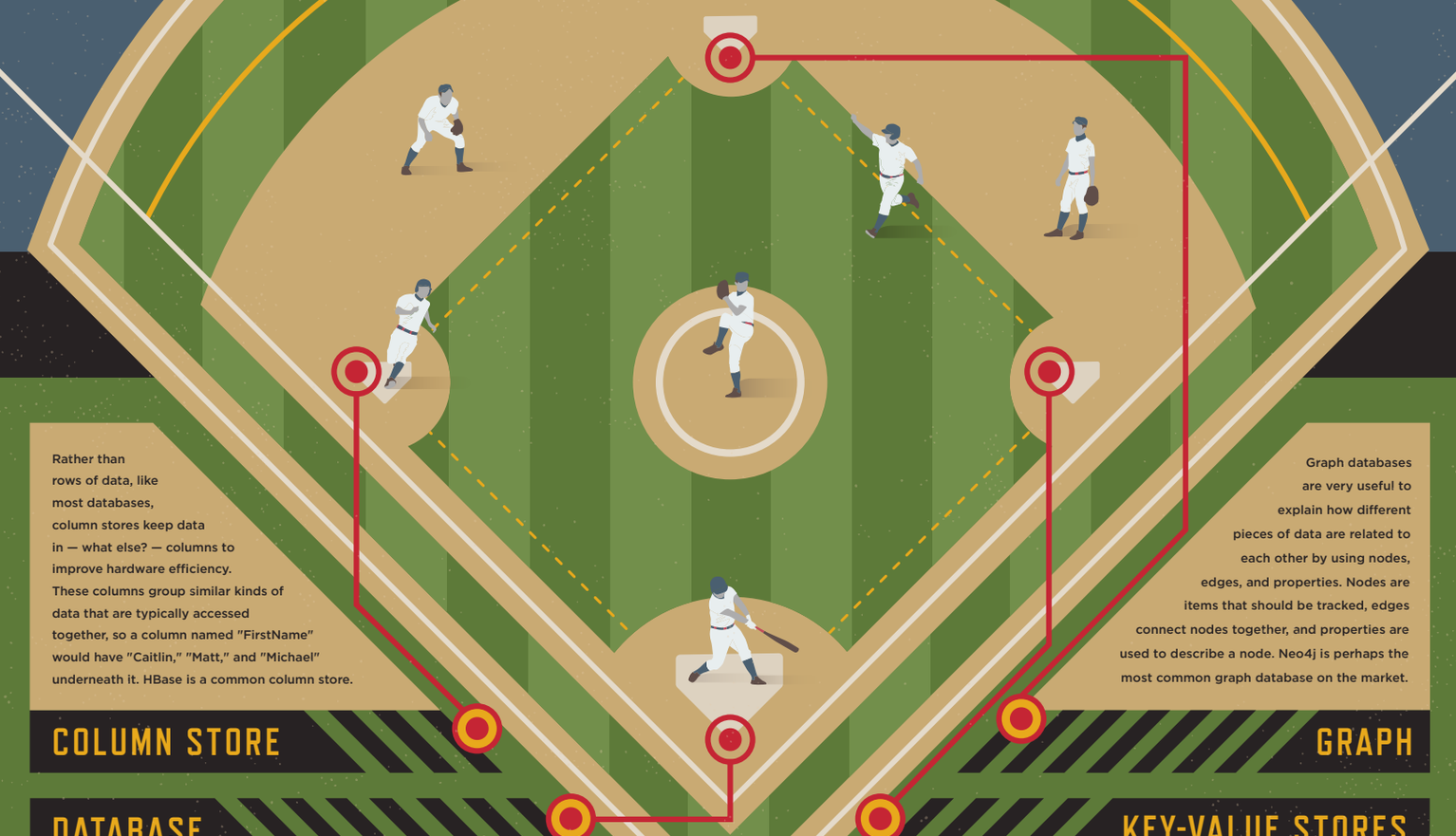
When a database needs to be partitioned, 49% of people elect to do it vertically

FACTOID 3

Only 27% of respondents version control their databases

FACTOID 4

32% of DZone users are planning to adopt a new database within six months



COLUMN STORE

Rather than rows of data, like most databases, column stores keep data in — what else? — columns to improve hardware efficiency. These columns group similar kinds of data that are typically accessed together, so a column named "FirstName" would have "Caitlin," "Matt," and "Michael" underneath it. HBase is a common column store.

GRAPH

Graph databases are very useful to explain how different pieces of data are related to each other by using nodes, edges, and properties. Nodes are items that should be tracked, edges connect nodes together, and properties are used to describe a node. Neo4j is perhaps the most common graph database on the market.

DATABASE

Data is stored in a database and then is assigned a key, similar to a KV store, but the data that is stored can be structured or semi-structured data in a XML, JSON, or BSON document. These databases can store more complicated information than simple strings. MongoDB, the most widely-used NoSQL database, is a document database.

KEY-VALUE STORES

In key-value stores, data is kept in the database and then assigned a key. Unlike relational databases, key-value stores treat all data as part of one collection, without relating two pieces of data to each other. Redis and Cassandra are popular key-value stores.

Missing Out on Speed? Speak SQL to NoSQL and Get the Best of Both Worlds.

When NoSQL hit its hipster stride around 2009, it came with a lot of flag waving. For an ambitious developer whose change-script or update requests were regularly rejected by their ornery DBA, MongoDB (or one of the other early non-relational options) arrived a bit like Batman. Fearsomely powerful and very much at the developer end of DevOps, it was a database that, in the app-coder's hands, could be deployed in ten minutes flat and spoke more-or-less fluent Javascript. It was installed by the million.

But that was almost ten years ago. The early adopters have moved on to the next cool thing, be that containerization, Golang, or some other hairy critter climbing the vines of the hype cycle. Meanwhile, MongoDB has grown up fast, and now runs increasingly large projects and sprawling systems that once were experimental but are now business critical. It's the early majority that is now embracing MongoDB.

But unlike their visionary cousins, the majority is pragmatic. They're not interested in hacking or debugging someone else's code, they just need it to work the first time, every time. And that's why a shared MongoDB IDE like Studio 3T is vital to the daily grind. Now your SQL-fluent DBA can run SQL queries directly against a MongoDB collection. Your JavaScript developer can run JSON queries against the same collection. Your code-shy Project Manager can drag-and-drop queries against it too. And you? You can set a read-only connection on production, unlock dev and QA, and marvel as your team gets more done, fast.

Find out more about bridging the SQL/NoSQL gap and developing faster with Studio 3T.

WRITTEN BY TOMASZ NAUMOWICZ
CO-FOUNDER, **STUDIO 3T**

PARTNER SPOTLIGHT

Studio 3T - the IDE for MongoDB

Studio 3T

Get days more work done with the world's most widely used IDE for MongoDB.

CATEGORY
Database management

NEWEST RELEASE
Continuous

OPEN SOURCE
No

CASE STUDY

Spoiler Alert identified an opportunity to use software to help solve the problem of food waste in the USA. Currently a shocking 40% of the food produced in the USA ends up being thrown away. Spoiler Alert decided to build a platform to bring together wholesale food distributors and non-profits that are looking for healthy food to distribute to their members and customers. They chose MongoDB as the database for their solution because of its flexibility and speed. They chose Studio 3T as their IDE for managing the data, exploring, testing and debugging. "We needed a tool that is really simple to use," says CTO Marty Sirkin, "that is obvious, intuitive, that is quick, that is flexible, and Studio 3T was all those things and more in the big things and in the small things."

STRENGTHS

- Team productivity-focused IDE for MongoDB
- High performance GUI and desktop client rolled into one
- Features include in-place editing, read-only connection strings, user and role management plus tree, JSON, and table views of data
- Multiple query modes include visual drag-and-drop query builder, Intellishell, SQL Query, and Aggregation Editor
- SQL import and export enables rapid ETL with SQL platforms

NOTABLE CUSTOMERS

- Tesla Motors
- Atlassian
- Deloitte
- Spoiler Alert
- Microsoft
- HBO
- Adobe
- Cisco

WEBSITE studio3t.com | **TWITTER** @Studio3T | **BLOG** studio3t.com/whats-new

The Secret Sauce of Graph Databases

QUICK VIEW

01 Graph databases have become the fastest-growing category of NoSQL databases

02 Modeling data as a graph may be difficult at first, but it's easy to add new nodes and relationships

03 There are several use cases for graph databases including fraud detection, recommendations, and more.

BY **MAX DE MARZI**
GRAPH DATABASE EXPERT

There is no magic in computer science. There are only two things: algorithms and data structures. Choose the right combinations and they will help you solve the problem. Choose the wrong ones and they will make the problem seem insurmountable.

Imagine you've been tasked with keeping a list of people, a list of things, and a list of things each person likes. If you use a relational database, you would have 3 tables: one table of people with a primary key of personId, one table of things with a primary key of thingId, and a join table between them holding foreign keys to both tables. Let's say we start with personId 83. How do we find the names of the things this person likes? We would scan down the join table looking for all the entries that had personId 83 in them, and collect the thingIds they referred. Then, we would scan down the things table, getting the name property of each of the thingIds we collected. This algorithm would work, but it would not be very efficient.

USING INDEXES

We could add an index to the keys of our tables. Now, instead of scanning the join table looking for entries that have personId 83, we can search a B-tree. A B-tree can perform a search operation in logarithmic time, using Big O notation: O(log n). As the data grows 10x in size, our search speed would only slow down by 2x. Starting with personId 83, we would perform one search operation to find all the thingIds they liked, and for each thingId, we would perform another search operation to get the name property of each one. Much better than scanning down the whole table.

Alternatively, we could have used a key-value store. KV stores use one big table and hashes to search in an average of O(1) time, and a worst case of O(n). Starting once again at personId 83, we would perform one hash operation to get to a list of thingIds they liked, and for each thingId, we would perform another hash operation to get the name property of each one. This would be faster, but we run into a problem: overhead from keeping a large hash table and the inevitable hash collisions that would creep up our search time.

USING POINTERS

Three tables didn't work well. One table didn't work well. What if we tried 2 tables?

Imagine one table to hold both people and things, and another table to hold the relationships between them. The entry for personId 83 would have a pointer to its first relationship. That first relationship would point to the entry for the thing that was liked as well as another pointer to the next relationship of personId 83. Instead of searching in a B-tree or hashing keys, we would chase pointers in two arrays which is always an O(1) operation.

That's the secret to graph databases. It is called "index free adjacency" but all it really means is that objects are connected at all times without having to search. This is an important distinction from other databases as it makes data "intelligent."

Imagine a row in a relational database. It has no idea which join tables are pointing at it and what else those join tables are pointing to. In the graph, a row is transformed into a Node and it knows exactly what it connects to and what is connecting to it by Relationships that are now first-class data citizens and hold referential integrity by design.

"That's the secret to graph databases. It is called "index free adjacency" but all it really means is that objects are connected at all times without having to search."

REINVENTION

"There is nothing new except what has been forgotten," said Rose Bertin to Marie Antoinette in 1785 and it applies here as well. Charles Bachman built the Integrated Data Store (IDS) in 1963, which managed relationships between records using chains of pointers. This later evolved into the network database model from CODASYL, which fell out of favor in the 1970s as relational databases emerged and eventually took over in the 1980s. We are in many ways still trying to build the technology showcased in 1968 by Douglas Engelbart in "the mother of all demos." But before we get lost in the past, let's jump back to the present and see where graph databases can help us today.

USE CASES

A graph database is a general-purpose database. Anywhere you use a relational database you can use a graph database, but it really shines when you can reimagine your data as a graph. There are hundreds of use cases, but the most popular ones seem to be real-time recommendations, fraud detection, master data management, network and IT operations, product hierarchies, rules engines, routing, ancestry, and data lineage. A big shift in capabilities occurs when you can take a query that wasn't able to perform in real time and suddenly have it complete in milliseconds. There are many accounts of companies that had failed to achieve results in Teradata or Oracle for years, which are now finding success in weeks using graph databases.

MODELING

To take full advantage of a graph database you must forget what you know about modeling in relational databases. There is no third normal form for graph databases. The shape of your data model will be heavily dictated by the kind of queries you need to run. It is a mistake to try to simply translate a relational model to the graph. It is also a mistake to see the graph in a flat two-dimensional plane. A graph allows for any number of dimensions and the ability to quickly traverse in and out of them. Yes, it can be more complicated to model things correctly initially, but the graph is pretty forgiving as it allows new types of nodes and relationships to easily be added. Careful constructs in direction and relationship types allow for a kind of data sharding or built-in pagination not easily doable in relational databases. The end goal is to ensure each query looks at the least amount of the graph as possible in order to find the correct answer quickly.

QUERIES

Getting the data into the graph is one thing, getting the answers out is another problem altogether. A language land war is currently being fought amongst the vendors. On one side you have Gremlin, a powerful imperative language built by a genius level developer to be used by genius level developers. On the other side you have Cypher, an ASCII art inspired declarative language that aims to be as easy to learn as SQL once was. Regardless of which side you choose, the real win is being able to express your ideas in a graph-like way, not the manner in which those ideas are interpreted. Much like the in the movie "Arrival," thinking in graphs requires a kind of graph epiphany, but once achieved, it opens up a new world of possibilities.

FUTURE

Once dismissed as the red-headed stepchild of the NoSQL world, over the last four years, graph databases have become the fastest growing category with new vendors emerging and existing vendors adding graph capabilities to their current solutions. As our industry continues to mature and evolve, it is important to know when to use the right tool for the job. There are over 300 databases listed on db-engines.com and you owe it to yourself to explore some graph databases. But be warned, once you experience their power and flexibility, you may never wish to go back to relational databases ever again.

Max De Marzi is a graph database expert. His graph power level is over 9000, but he is terrible at just about everything else. Over the past five years, he has authored about 200 github repositories and written over 100 blog posts on graph use cases on his blog at maxdemarzi.com where he waxes poetically and obsessively over these collections of points and lines. If you have a question about graphs, don't be shy...connect.

2018 Database DevOps Survey

Benchmarks & Best Practices Based on Data from 244 DevOps and Database Professionals

RESULTS ARE IN!
GET YOUR FREE COPY

Now you can compare your database DevOps operations and adoption rates to those of your industry peers with this comprehensive report.

The 2018 Database DevOps Survey Report presents up-to-date statistics based on responses from over 200 DevOps and database professionals.

Download this FREE report to get the newest data on:

- Database DevOps adoption rates
- Database deployment change frequencies
- Top risks and reasons for database errors
- And more…

DBmaestro
DevOps for Database

Download the FREE report

Modernize Database Processes with DevSecOps

Enterprises with digital aspirations now realize it's DevOps or die. The increase in agility gained from short, automated deployment cycles is staggering, and it's a matter of when, not if, software engineering teams make the switch.

The problem is that, although enterprises spend a lot of time and effort on a DevOps transformation, they often leave database development processes outside of the DevOps and DevSecOps loops.

Since the dawn of the DevOps revolution, various tools have come (and gone) which pioneered techniques to bring database power back to "the people." Today's developers enjoy a variety of offerings that provide assistance and guidance in their database interactions.

This "democratization of database development" has helped, but it overlooks a simple, painful truth: Giving developers increased database power has made it easier than ever to deploy bad schemas, suboptimal procedures, and execution paths. This approach also contributes to a lack of security, control, auditing, and visibility—elements that are an integral part of every DevOps project, at all levels.

Tools must strike the right balance between developer liberty, your capabilities, DevSecOps management, and database control, so choose wisely.

Download our eBook to understand 4 simple foundations to help you build the infrastructure you need to modernize your database processes with DevSecOps and shift your database development into high gear.

Download the eBook.

WRITTEN BY YANIV YEHUDA
CO-FOUNDER, **DBMAESTRO**

PARTNER SPOTLIGHT

DBmaestro DevOps Suite

DBmaestro DevOps for Database

Simplifying and automating database deployment processes in an agile environment while reducing critical application downtime

CATEGORY
DevOps for Database

NEWEST RELEASE
Weekly

OPEN SOURCE
No

CASE STUDY

A leading insurance company went through a transformation project of adapting modern development tools. During the process, they realized that a large percentage of failures could be attributed to the database and to a conflict between development and DBA teams.

The company started using DBmaestro to simplify, accelerate, and modernize the database release processes. Within a few weeks, they defined a new, well-coordinated release cycle for a central application that incorporates the database in the application release process and which is also connected to Jenkins. They were able to define and enforce roles and policies for the database and minimize the probability of errors caused when multiple teams worked on the same database object.

STRENGTHS

- Automate database release process to accelerate overall application release cycle
- Increase productivity by 15% for development and DBA teams
- Gain complete security, policy compliance, and transparent audits of databases
- Support multi-database environments through multiple DevOps tool integrations

NOTABLE CUSTOMERS

- BBVA
- Sura
- Allianz
- Barclays
- T-Mobile

WEBSITE dbmaestro.com | **TWITTER** @DBmaestro | **BLOG** dbmaestro.com/blog

Database Clones and Containers for Enterprise Data Delivery

QUICK VIEW

01 Database clones deliver writable production database environments for Dev/Test in seconds.

02 This article surveys available solutions for use of database clones.

03 This article introduces a new software based database cloning solution with SQL Server containers, providing enterprises with an open "hub" for database environment delivery.

BY **PAUL STANTON**
VP AND CO-FOUNDER, **WINDOCKS**

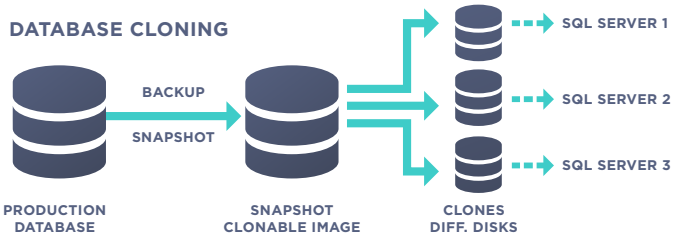
In the late 1980's, personal computers were poised to transform enterprise computing, and the Ethernet protocol would create a similar phenomenon in local area networking. We soon needed File Sharing, and in the early 1990's NetApp popularized network attached storage. Interestingly, Cisco and others followed a similar path as NetApp, delivering fixed purpose UNIX appliances. The IT infrastructure that persists to this day was born.

Rationalizing this disparate infrastructure is a continuing issue, and particularly as enterprises increasingly explore Digital Transformation and Cloud migration. Fortunately, new methods are available that combine use of database clones with containers for delivery of database environments, for both on premise and cloud.

My firm, Windocks, delivered an independent port of Docker's source to Windows, and earlier this year was the first to incorporate database cloning with containers. In this article we'll look at the use of database clones that are available on existing Storage Array Networks (SANs), as well as new Windows based clones.

DATABASE CLONES FOR DEV/TEST

Database clones are ideal for short lived environments in Dev, Test, and reporting. A writable terabyte class environment can be delivered in seconds, with a small storage footprint. Clones employ a Copy on Write or Redirect on Write method to accommodate changes, relying on a parent snapshot or image as a data source. Clones expand dynamically as changes are made.



STORAGE AREA NETWORK (SAN) SNAPSHOTS

Databases require data that are proximate to the host, and Storage Area Networks with iSCSI, Fibrechannel, or fast NFS and SMB networks are the current incumbent storage systems. NetApp, Pure Storage, EqualLogic, and others deliver writable clones in the form of fast snapshots. SANs are widely available, but are underutilized for database dev/test support due to the complex scripting required to access data. Use of SAN based snapshots involve provisioning of LUNs, volumes, and mount points, and ongoing management is needed to clean up mount points when the snapshots are no longer used.

Docker Volume plugins have been developed to allow containers to access SAN data, but the complexity of dealing with storage primitives remain. Time will tell if DBAs are will wade through the storage complexity to use this approach.

SAN V2: "COPY DATA MANAGEMENT"

Dev and Test environments don't share the performance requirements of production systems and new firms are focused on dev/test support. Secondary storage comprises roughly 80 percent of enterprise data, and the Gartner Group has dubbed this space "Copy Data Management." Cohesity, Rubrik, and others are delivering improvements in what has been a quiet part of the industry.

CDM systems continue the pattern of fixed purpose UNIX appliances, but with new capabilities. CDMs deliver data deduplication, incremental updates, and simplified access to data via restful APIs. These systems aim to provide "incremental forever," instantly available environments (doing away with full backups). APIs simplify access and use of data, and CDM system are available as software appliances on AWS, Azure, and other public clouds.

WINDOWS DATABASE CLONING

In addition to SAN and CDM appliances, Windows based cloning is now available from Windocks that combines SQL Server database clones with containers. Windows based clones employ the same storage virtualization techniques as the SAN and CDMs, and deliver writable Terabyte class environments in seconds, with minimal storage required for the clones. As a Windows based solution clones are infrastructure agnostic, and run on premise or on any public cloud.

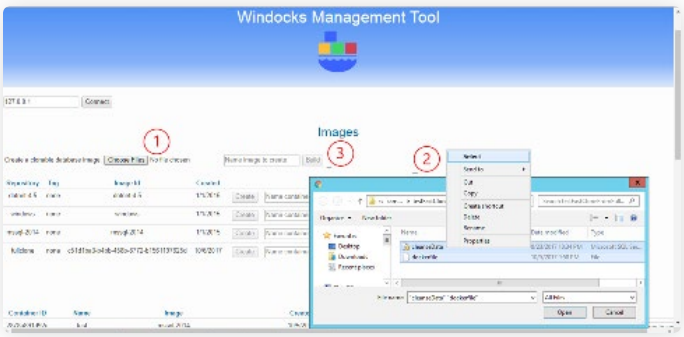
Building Windows-based clones starts with a plain text configuration file, listing the database backups or database files, along with any SQL Server scripts to be run. In this example, the backups are located on a network file share, and a SQL Server script is used to implement data masking.

dockerfile - Notepad

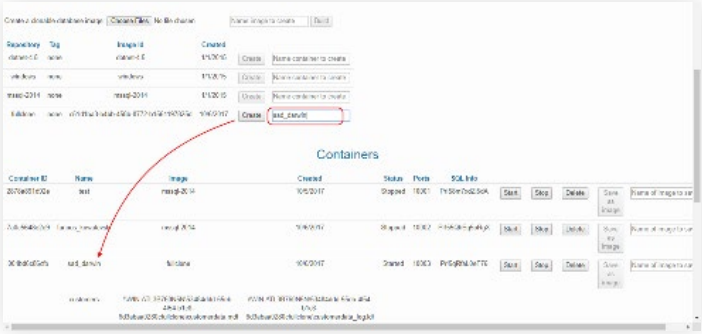
File Edit Format View Help

```
FROM mssql-2016
SETUPCLONING FULL customers   \\test\data\customerfile1.bak\\test\data2\customerfile2.bak \\test
\data3\customerfile3.bak
SETUPCLONING FULL operations  \\test\data\operations.bak
COPY cleanseData.sql  .
RUN cleanseData.sql
```

The image is built by submitting the configuration file and SQL script with the Windocks web UI. The file selector tool is used to select the files (1), which are highlighted and selected (2), and submitted with an image name (3).



Once the new image is available users provision containers by choosing a container name and "create." The cloned database is delivered in seconds, irrespective of the size of the parent image. Developers can specify fixed connection strings, allowing the back-end to be provisioned with fixed ports and SQL credentials, simplifying integration with .NET, Java, or other front-end or middle-tier applications.



A container based solution such as Windocks easily integrates with Jenkins and other Continuous Integration tooling, and supports delivery of multi-tier environments that can include .NET or Java with SQL Server back-ends. Windocks also supports delivery of snapshots sourced from SANs and CDM systems, and will soon support delivery of these cloned environments to Microsoft containers and other application environments.

MODERNIZE DATA DELIVERY WITH EXISTING INFRASTRUCTURE

Surveys indicate that limited access to database environments for testing is blocking efficient software development and delivery. These processes begin with a step-wise approach to implement and deliver cloned database environments. Clones are critical for supporting use of production database environments in Jenkins and other CI processes.

Database clones provide an opportunity to modernize development and test processes by using existing SAN infrastructure, or new CDM solutions that offer enterprise backup scalability, with deduplication, and improved ease of use by DBAs. For organizations that are interested to explore SQL Server DBA accessible tools, Windocks is a new option. As a software based approach, Windocks supports free evaluation with a Windocks Community Edition, which is available here.

Paul Stanton is a former Director at Microsoft, and has been involved in container initiatives, including Cloud Foundry, OpenShift, and more recently as co-founder of Windocks in delivery of an independent port of Docker's source to Windows. At Windocks Paul leads Product Management, Marketing, and industry partnerships.

Why Time Series matters for metrics, real-time, and sensor data

Download the e-book

"MySQL is not intended for time series data... I can testify it is like pounding nails with a screwdriver. It's definitely not what you want to do in any relational database."

John Burk, Senior Software Developer

influxdata®

The Modern Engine for Metrics and Events

Time Series: The Next Revolution of Technologies

Time Series Databases are the fastest-growing database category, according to independent research, and store measurements tracked over time. This could be metrics and events from servers, applications, networks, sensors, or even trades in a market. The difference with time series data from regular data is that it is time-stamped, and analytics on this data are around "change over time." It comes in two forms: regular and irregular. Regular (metrics) are measurements gathered at regular intervals (e.g. CPU usage every micro-second). Irregular (events) are driven by some trigger, for example a sensor that gets triggered if the temperature exceeds some threshold.

Time Series Platforms are becoming a critical architectural component in every environment because of three major trends which are fueling the growth of time-stamped data — we are witnessing the instrumentation of every available surface in the material and virtual worlds from wearable sensors, to computing microservices, and everything in between.

THE TIME SERIES WORKLOAD

Time Series data is very different from any other workload: It requires millions of writes per second, the ability to do real-time queries on huge datasets, time-based functions that help measure change over time, optimal disk compression with these large data sets, and the need to keep high value data accessible for real-time access, while storing older data with potentially different time precision available for historical analysis.

InfluxData is the leading Time Series Platform and comes out of the box ready to use. Learn more.

WRITTEN BY MARK HERRING
CMO, **INFLUXDATA**

PARTNER SPOTLIGHT

InfluxData

The modern engine for metrics and events

CATEGORY
Time Series Database

NEWEST RELEASE
Quarterly Release cycles

OPEN SOURCE
Yes

CASE STUDY

Coupa Software needed to create a custom DevOps Monitoring solution for their leading spend management cloud platform. With InfluxData they moved from pure data collection to predictive analytics and achieved a consistent track record of delivering close to 100% uptime SLA across 13 major product releases and 5 major product module offerings, as well as solving their data accessibility, aggregation, and retention challenges. Operational metrics are collected via Telegraf, stored in InfluxDB, and analyzed by Kapacitor. They use Grafana for visualization and have created a custom alerting framework. This has become the foundation to the path of building a system that is self-healing and can provide predictive analytics key to accurate forecasting.

STRENGTHS

- Deploy and begin building your application right on top in minutes
- Real action in real-time
- Developer happiness

NOTABLE CUSTOMERS

- Citrix
- Coupa
- Nordstrom
- IBM

WEBSITE influxdata.com

TWITTER @InfluxDB

BLOG influxdata.com/blog

Defending Your Database From Ransomware

BY **OREN EINI**
CEO, **HIBERNATING RHINOS LTD**

QUICK VIEW

01 It is all too easy have all your data in the open for any interested party to delete, change, or ransom.

02 The "default" deployment configuration for many databases is suited for development and favors ease of use over security.

03 This type of issue is the equivalent of leaving all your doors open. It is also something that is common and has impacted tens of thousands of companies.

04 Basic security measures can prevent such issues, but they need to actually be applied.

Some attacks are hard to defend against. Zero-day exploits, where there is no patch and sometimes no solution, leave organizations with very little that can be done to prevent attacks. In other cases, keeping up to date on patches will prevent such issues (as was the case with WannaCry, for example), but that can be hard and requires constant vigilance and attention.

In contrast, the recent wave of ransomware attacks on databases has been primarily the result of bad defaults and the inattention of administrators in regard to what is going on their systems. In the past few months, we have seen ransomware attacks on tens of thousands of MongoDB instances, thousands of Elasticsearch databases, and hundreds of CouchDB, Hadoop, and MySQL servers.

In a few of these cases, the ransomware attack has indeed been an unpatched vulnerability, but in the vast majority of cases, the underlying reason was much simpler. The gates were opened, and anyone and everyone was welcome to come in and do as they please. These databases, often containing sensitive and private information, have been left open to the public internet without even the most rudimentary of security mechanisms.

All of the above-mentioned databases are quite capable of controlling who has access and to what, but if you never close the door, the sophistication of the lock is not relevant to the thief, who can simply step in.

Here is how it typically goes. A developer needs a database for an application, so they download and install a database and run it on their own machine. Typically, in such a case, the database is only available to the local machine — only used by a single user — and any security in place will be a hindrance, rather than help, during the development process.

Because of this, most databases will happily run in a mode that basically accepts any connection to the database as a valid user with full privileges. It makes the development process much easier and increases the adoption rate, since you don't have to struggle with security decisions from the start.

This is a good thing. Developers usually don't have the required skillset to make proper security decisions for the organization, nor should they. That role should be at the hands of the operations staff, which control and maintain the production environments for the organization. At least, that is the way it *should* be.

In practice, in many cases, the person who developed the application is also responsible for setting it up in production. That can be a problem from a security point of view. The easiest thing to do is to simply repeat the same development setup procedure (after all, we know it works) in production and configure the database to accept remote connections as well.

Just a few words on the consequences of these issues. A database that was left open to the world in production can be the end of a business. Leaving aside the fact that you might need to ransom your data (being effectively offline for the duration), you'll also need to disclose the breech to customers

and possibly face regulatory action. As you can imagine, that can be...unpleasant.

A database that anyone can log into as an administrator is an excellent attack vector into an organization, so even if you aren't holding anything particularly interesting in that database, it can be used as a jumping off point from inside your firewall, bypassing a lot of the safeguards you rely on.

If you have any sort of experience with security, the last paragraph probably made you squirm. I'm sorry about that, and I dearly wish things were different. It is easy to blame the developer with negligence when you consider the implications (I'm picking a bit on developers here because they are usual suspects in such cases, although professional operation teams also make these sorts of mistakes quite often). One recent case with an IoT stuffed-toy exposed two million voice records of kids and their families along with close to a million accounts. That particular database has been left exposed to the public and been held for ransom at least three times. I would call *that* gross negligence, to be sure.

But the number of successful attacks also indicate another issue. Taking MongoDB as the leading example in these breaches, there is a security checklist for production deployments, which is great. The list includes 15 topics and goes on for 44 pages. For any organization that does not have a dedicated security stuff, expecting users to go through that before each deployment it not realistic.

The major problem here is that security is *hard*, requires expertise, and often runs in direct contradiction to ease of use. It is also true that whenever a user needs to jump a hurdle because of security, their first inclination is to just disable security completely.

This means that any security design needs to balance the ease of use of the system with its level of security and make sure that a reasonable tradeoff is made. It is also the case that large part of the security design of the system is actually entirely unrelated to security at all, but completely focused on the user experience. This is strange, because you'll typically find UX decisions at the user interface level, certainly not as one of the top items in the security stack.

But the truth is that if you want to have a secured solution, you need to make it easy to run your system in a secured manner, and you need to make that approach the default and most reasonable thing to do. From my point of view, this is a user experience problem, not a security problem.

In my day job, I'm working on building the RavenDB document database, and these kinds of threats can keep you up at night. Notice that at this point, we haven't even *discussed* any kind of security mechanism of any kind, because by default, none is usually engaged. And the problem is that the security policies that are mandatory in production are a major hindrance in development. This leads to an interesting dichotomy. Databases that install with unsecured defaults get more users (since they are easier to start with), but they also have a tendency to leak data and expose the organization to major risks because it is easy to miss changing the default configuration.

With RavenDB, we have implemented a two-step approach. As long as RavenDB is running on a developer machine and is configured to listen only to local connections (which is the default in the typical developer installation), there is no need to configure security. This is safe, since there is no way for an outside party to connect to the database. Only connections from within the same machine are allowed.

When you go to production, you'll need to reconfigure RavenDB to listen to connections from the network, and it is at this stage that the second step of the security design applies. RavenDB will refuse to run in such a configuration. Listening to the network while not having any authentication set up is a red flag for a highly vulnerable position, and RavenDB will reject such a configuration.

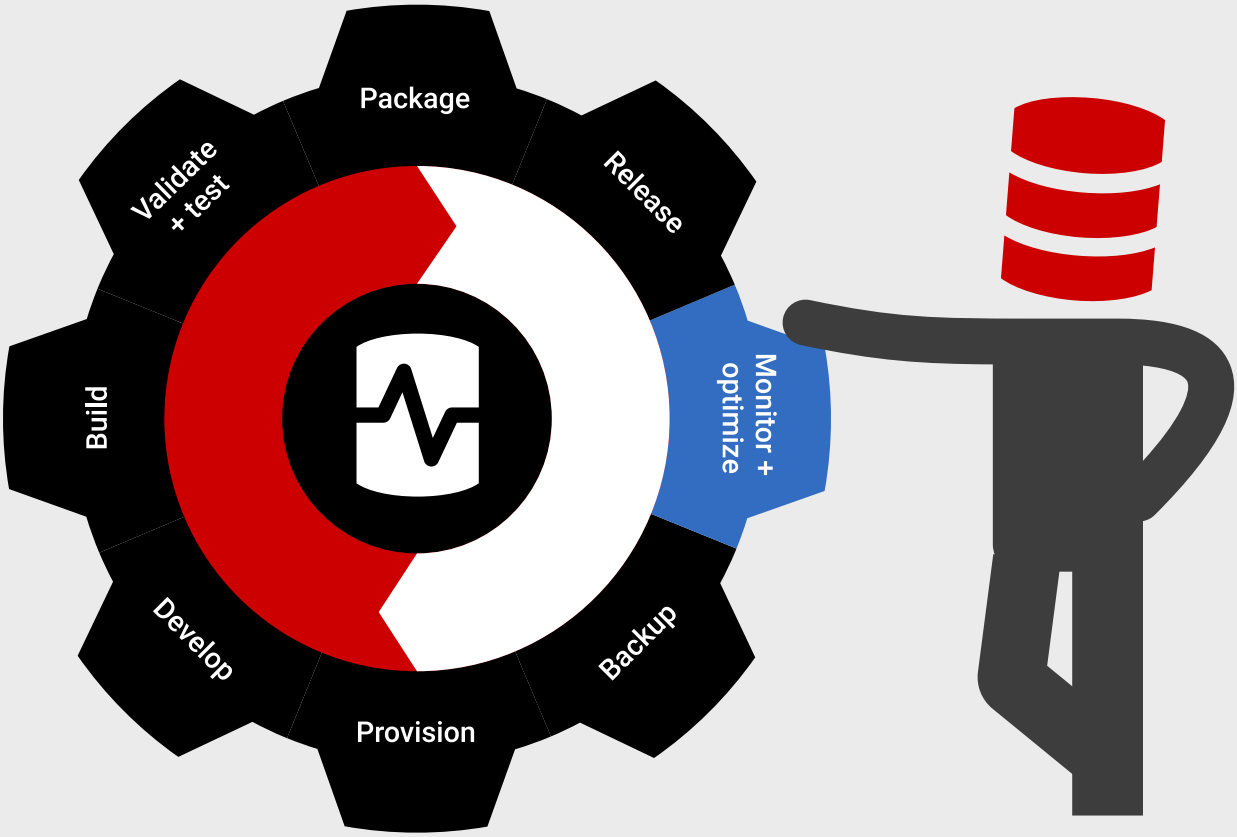
At this point, the administrator is no longer just operating by rote but is aware that they need to make an explicit security decision and set up the appropriate security and authentication. We have been using this approach for the past five years or so with great success. Making a proper distinction between a developer machine (easy to set up, no need for security) and production (more involved to set up and configuration and security are paramount) gives users the appropriate balance.

A quick look through Shodan shows over 50,000 MongoDB instances that are listening to the public internet and over 15,000 exposed instances of Elasticsearch and Redis each. It took me about two minutes to find a publicly addressable MongoDB instance that had no authentication at all (by all appearances, it had already been mauled by ransomware by the time I looked at it).

Combine that with the fact that tools such as ZMap can scan the entirety of the IPV4 range in a few minutes, and you can assume that any publicly visible instance is going to be probed within a few minutes of coming online.

Regardless of your choice of database, ensuring that you have taken the basic step of locking down production systems is just the first item on your list. I'm aware that a security article that basically states "don't leave your door wide open" is a bit redundant, but the tens of thousands of instances publicly available on the net shows that this is still a topic worth discussing.

Oren Eini is CEO of Hibernating Rhinos LTD, an Israeli-based company that produces RavenDB (NoSQL database) and developer productivity tools for OLTP applications such as NHibernate Profiler, Linq to SQL Profiler, Entity Framework Profiler (efprof.com), and more. Oren is a frequent blogger at ayende.com/Blog.



SQL Server performance monitoring for database DevOps

With the increased speed of delivery that database DevOps brings, you need to make sure you have the right information to react to any unexpected changes

www.red-gate.com/monitor-devops

redgate

Effective Database Monitoring is No Longer Optional

There was a time when DBAs could get away with only basic monitoring of SQL Server databases, with hand-rolled scripts or generalist monitoring tools like Microsoft Systems Center Operations Manager (SCOM) providing alerts to general performance and availability issues.

It's no longer enough. Environments have become more complex, with more instances and larger estates. Business is moving faster, and downtime can have a long-lasting as well as an immediate impact. Companies cannot afford their database performance to fall or stall.

Beyond being warned that a spike in CPU usage has occurred, for example, DBAs need richer monitoring that provides a focused set of performance metrics to quickly pinpoint the root cause of an issue, not just the symptoms, in five minutes or less.

If you're thinking of bringing DevOps to the database, it's even more important. Already popular in application development, it's been shown to speed up the deployment of changes, updates and improvements, while at the same time reducing the change failure rate. Tooling and processes have also emerged that make it possible to include the database in version control, continuous integration, and automated deployments.

By developing the database alongside the application, often plugging into and integrating with the systems already in place, deployments become reliable, robust and error-free. A lot of the onerous database development work like the creation of deployment scripts is then automated, freeing up time for more rewarding tasks such as performance tuning.

This increased speed of delivery, along with the automation that DevOps introduces, makes monitoring a crucial element. With DevOps, the database moves from being a relatively static code base to one that changes little and often, with updates constantly rolling from development through testing and staging environments before being deployed to production.

As a measure of the increased speed of delivery it offers, one notable company, Skyscanner, introduced DevOps for the database and went from releasing changes every six weeks to releasing them up to 95 times a day.

The increased speed of delivery... along with the automation that DevOps introduces, makes monitoring a crucial element.

This is where a performance monitoring tool really comes into its own because, however effective the testing regime is, it's only when changes hit the production environment under real load that their true effect can be monitored. If problems do occur, time is at a bigger premium than ever.

Give the development team access to such a tool and the advantages increase further because they can correlate their own changes to any issue that arises, discover why it happened and apply continuous improvements.

A good signpost of the importance of monitoring is the widely read and quoted State of Database DevOps report from Redgate. This showed that 50% of organizations have server performance monitoring in place – and the figure rises to 65% among those who have already adopted a DevOps approach.

So, if you're looking for a faster, more effective way to prevent problems affecting your users now, or you want to ensure the success of introducing a DevOps initiative, remember to include a performance monitoring tool.

WRITTEN BY BEN EMMETT - PRODUCT MANAGER, **REDGATE SOFTWARE**

Executive Insights on Databases

QUICK VIEW

01 Having the right tool to solve the business problem is just one of the keys to a successful database strategy.

02 Databases are critical to every company in every industry since they deliver immediate, personalized, data-driven applications and real-time analytics.

03 The greatest opportunities in the evolution of databases are the integration and convergence of technologies in the cloud using microservices.

BY **TOM SMITH**
RESEARCH ANALYST, **DZONE**

To gather insights on the state of databases today, and their future, we spoke to 27 executives at 24 companies who are involved in the creation and maintenance of databases.

EMMA MCGRATTAN S.V.P. OF ENGINEERING, ACTIAN

ZACK KENDRA PRINCIPAL SOFTWARE ENGINEER, BLUE MEDORA

SUBRA RAMESH VP OF PRODUCTS AND ENGINEERING, DATAGUISE

ROBERT REEVES CO-FOUNDER AND CTO, DATICAL

BEN GELLAR VP OF MARKETING, DATICAL

PETER SMALLS VP OF MARKETING AND BUSINESS DEVELOPMENT, DATOS IO

SHALABH GOYAL DIR. OF PRODUCT, DATOS IO

ANDERS WALLGREN CTO, ELECTRIC CLOUD

AVANTIKA MATHUR PROJECT MANAGER, ELECTRIC CLOUD

LUCAS VOGEL FOUNDER, ENDPOINT SYSTEMS

YU XU CEO, TIGERGRAPHSQL

AVINASH LAKSHMAN CEO, HEDVIG

MATTHIAS FUNKE DIR., OFFERING MANAGER, HYBRID DATA MANAGEMENT, IBM

VICKY HARP SENIOR PRODUCT MANAGER, IDERA

BEN BROMHEAD CTO, INSTACLUSTR

JULIE LOCKNER GLOBAL PRODUCT MARKETING, DATA PLATFORMS, INTERSYSTEMS

AMIT VIJ CEO AND CO-FOUNDER, KINETICA

ANOOP DAWAR V.P. PRODUCT MARKETING AND MANAGEMENT, MAPR

SHANE JOHNSON SENIOR DIR. OF PRODUCT MARKETING, MARIADB

DEREK SMITH CEO AND SEAN CAVANAUGH, DIRECTOR OF SALES, NAVEEGO

PHILIP RATHLE V.P. PRODUCTS, NEO4J

ARIFF KASSAM V.P. PRODUCTS, NUODB

WILLIAM HARDIE V.P. ORACLE DATABASE PRODUCT MANAGEMENT, ORACLE

KATE DUGGAN MARKETING MANAGER, REDGATE SOFTWARE LTD

SYED RASHEED DIR. SOLUTIONS MARKETING MIDDLEWARE TECHNOLOGIES, RED HAT

JOHN HUGG FOUNDING ENGINEER, VOLTDB

MILT REDER V.P. OF ENGINEERING, YET ANALYTICS

KEY FINDINGS

01 Keys to a successful database strategy are: 1) having the right tool to solve the business problem at hand; 2) performance and speed; 3) data security; 4) scalability; 5) automation; and, 6) the right data in the right format.

Specific projects need specific resources. You need to know the business problem you are attempting to solve to ensure you are using the right database solution. Doing so will have a tremendous impact on the performance or speed of the database, especially with regards to large-scale data sets.

Security is now on everyone's mind and is also a requirement for government, financial services, healthcare, and global retail customers given the regulatory compliance environment.

Automation and DevOps are required in order to accelerate the development cycle. DevOps processes require databases to be de-siloed so that everyone involved in data management and analytics has insight into the process. Lastly, data must be accessible but secure. That means knowing where data is and eliminating duplicate data that leads to significant storage expense. Enable data management to take advantage of the latest technology.

02 Companies in every vertical, especially those making a digital transformation, rely on databases to analyze and monetize data by reducing costs and identifying new revenue streams. They are critical to delivering immediate, personalized, data-driven applications and real-time analytics.

03 The cloud and containers, the transition from NoSQL back to SQL, and consolidation are the tree biggest changes in the database ecosystem over the past year or so. Companies are moving from traditional on-premise databases to the cloud. The exploding volume of data is driving the adoption of cloud-native databases that will scale with the data. Cloud solutions

are offering classical relational databases (e.g., Spanner and Dynamo) that are sophisticated solutions. Containers are disruptive with a new set of requirements and challenges like persistence, provisioning, and storage.

NoSQL is not as pervasive. There's been a shift away from NoSQL with Spanner, Cockroach, and NuoDB providing more consistent systems with SQL. SQL is coming back built on top of NoSQL.

The industry and technology are consolidating. People want the best of both worlds: distributed computing and SQL, with elastic SQL like Google Cloud Spanner that enables scalable apps to be built using SQL.

04 More than 60 different solutions were mentioned when we asked about the technical solutions they and their clients used for database projects. This affirms the breadth of platforms, architectures, languages, and tools that make up the ecosystem. The four most frequently mentioned were: Spark, Hadoop, Cassandra, and Oracle.

05 Databases are widely adopted in every industry and are used for accounting, enterprise resource planning, supply chain management, human capital management, and basic data management. There are industry specific solutions for multiple vertical that can process, scale, be available, and be secure. The three most frequently mentioned industries were financial services, healthcare, and retail. The first two are driven by the need to meet regulatory requirements while retailers are trying to serve customers their best product option, at the best price, in real time.

06 Lack of knowledge, the complexity of the ecosystem, and scalability are the major hurdles for companies implementing successful database strategies. There are 50 to 100 different types of database, and the customer needs to understand the benefits of each one. There's a need to always put requirements first and not compromise critical business performance with speed, scale, accuracy, and cost. Those unfamiliar with databases can quickly find themselves in over their heads and unable to see the potential roadblocks of scalability, consistency, and availability. The cost of switching technology is high and evaluating fit can be challenging.

Data is exploding in volume, along with the number of solutions to deal with it. The more data that is stored is inversely proportional to a company's ability to analyze the data. Companies have no clue how many copies of data they have, no idea of data lineage, and no idea of where all their data resides.

07 The biggest opportunities in the evolution of databases are with the integration and convergence of technologies in the cloud using microservices. As data grows, there will be greater adoption of modern platforms, with more functionality, and a more comprehensive data environment. More integrated platforms will be managed with APIs and offer both multi-dimensional and text analytics using artificial intelligence (AI), machine learning (ML), and predictive analytics. We will have polyglot or multi-model persistence to create a unified platform that performs hybrid transactional/analytical processing.

Distributed relational databases in the cloud are the future. The cloud provides a disposable infrastructure that lets the business owner focus on their core business. Microservices are becoming the norm and will change how systems are architected.

08 The biggest concerns around databases today is the persistence of legacy thinking and the lack of skills necessary to evaluate and manage new database solutions. Companies are still working on legacy idioms and data models, and cannot take advantage of parallelism. The fact that the database process has stayed the way it has is either negligence or criminal intent.

There's a skills and knowledge gap that's hard for companies to adapt because they do not have the specialists. The companies that do not carefully consider the new value and strengths of more established solutions when setting out to build something should be concerned. Users are sent on a wild goose chase because they don't understand the right technology for the job – either due to a lack of knowledge or because the vendors are misleading them with regards to non-native graph databases.

Databases are now part of the DevOps fabric. It takes a different skill set to get business value from them these days. You have to be part programmer, part automator, part infrastructure administrator, and part database administrator. That's a unicorn these days.

09 According to our respondents, while there are a lot of things for developers to know to be proficient with databases, only knowledge of SQL and databases were mentioned by more than a couple of respondents. You need to know the difference between joins, because SQL is not going away. MongoDB, H-base Hadoop, and Cassandra all have SQL interfaces, and SQL is still the most well-known language. Understand how SQL impacts the performance of the data store, even though frameworks automate it.

Understand the different data models and the end goal so you can choose the right database for your use case. Look at the platform and tools and understand when to choose a relational (transactions) versus a NoSQL (reports and analytics) database.

Reinvent the art of evaluating technology. Everything changes so fast the fundamentals you learned five or 10 years ago may no longer apply. Something counter to your intuition could work because the underlying facts have changed. Modern networks have reduced latency from tens or hundreds of milliseconds to less than one millisecond. Bandwidth is 100-times greater than it was. Try to find the next set of standards.

Tom Smith is a Research Analyst at DZone who excels at gathering insights from analytics—both quantitative and qualitative—to drive business results. His passion is sharing information of value to help people succeed. In his spare time, you can find him either eating at Chipotle or working out at the gym.

NEW RELEASE! c-treeACE V11.5 Available Fall 2017

c-treeACE®

Multimodel Database

by FairCom®

FAST
RELIABLE
CUSTOM-ENGINEERED

NoSQL • ACID • Key-value Store • ISAM
Full ANSI SQL • 20+ Platforms • 22 APIs

ALL OVER YOUR NOSQL DATA

THESE COMPANIES TRUST C-TREEACE FOR THEIR TOUGHEST DATA MANAGEMENT JOBS

ups™ COMMVAULT® verizon BNY MELLON THOMSON REUTERS
FICO™ tealeaf an IBM Company VISA HENRY SCHEIN®

FairCom® Trusted Database Technology Since 1979

FairCom.com
800.234.8180

Does NoSQL and SQL Together Still Mean Two Data Copies?

To put it simply, the database world has been divided into two factions: SQL and NoSQL. The common belief has been the two could co-exist in an organization's database ecosystem, but typically only through some form of ETL (Extract, Transform, and Load) process that requires making a copy of the data and altering it to achieve SQL compatibility.

However, the downside of an ETL strategy is you now have two copies of the same data that you need to keep in sync, which is not optimal for several reasons. The major one is that if you are relying on the SQL version of the data to perform reports and make business decisions, those decisions will only be as current as your most recent ETL operation. Other items to consider include the cost of additional computing resources, storage space, and dealing with ETL error logs.

c-treeACE by FairCom has made it possible for NoSQL and SQL to seamlessly co-exist in the ever-changing database technology world by allowing NoSQL and SQL operations over one single instance of data. This is made possible because c-treeACE implements a lot of the database fundamentals at the NoSQL level that are required by a full relational DBMS. This is just one example of the powerful capabilities that the multimodel c-treeACE database can provide to organizations.

See what more than 40 percent of the Fortune 100 companies already know. It is no longer a NoSQL vs. SQL world. It is now a NoSQL and SQL world over one single instance of data with c-treeACE. Learn more at FairCom.com.

WRITTEN BY RANDAL HOFF
VICE PRESIDENT OF ENGINEERING SERVICES, **FAIRCOM CORPORATION**

PARTNER SPOTLIGHT

c-treeACE V11.5 by FairCom

Fast, reliable, custom-engineered.

CATEGORY
Multimodel NoSQL + SQL database

NEWEST RELEASE
Continuous

OPEN SOURCE
No

CASE STUDY

c-treeACE is a multimodel-database offering unique fast indexing technology for high-performance NoSQL and industry-standard SQL within the same application, over the same data. Customizability is what sets c-treeACE apart. With 498 configuration calls, it can be calibrated tightly to the application to meet the hardest and most unique challenges facing the tech industry, such as security and the balance of performance vs. reliability. With nearly 40 years of history, the rock-solid c-treeACE core backs every deployment and powers over 40% of the Fortune 100. Operating on over 20 platforms, 22 APIs and engineer-2-engineer support at every stage — development through production — no other database on the market empowers like c-treeACE.

STRENGTHS

- ACID-compliant, key-value store
- Customizable
- High speed and availability
- Dynamic schema management

NOTABLE CUSTOMERS

- Visa
- Verizon
- UPS
- FICO
- Thompson Reuters

WEBSITE FairCom.com

TWITTER @FairCom_Corp

BLOG faircom.com/developers

Solutions Directory

This directory contains databases and database performance tools to help you store, organize, and query the data you need. It provides free trial data and product category information gathered from vendor websites and project pages. Solutions are selected for inclusion based on several impartial criteria, including solution maturity, technical innovativeness, relevance, and data availability.

COMPANY	PRODUCT	CATEGORY	FREE TRIAL	WEBSITE
Actian	Actian X	RDBMS	30-day free trial	actian.com/data-management/ingres-sql-rdbms/
Actian	Versant Object Database	Object-Oriented	30-day free trial	actian.com/data-management/versant-nosql-object-database/
ActiveRecord	ActiveRecord	ORM	Included in Rails	guides.rubyonrails.org/active_record_basics.html
Aerospike	Aerospike Server	In-Memory, KV	Open source	aerospike.com/download/server/3.15.0.1/
Altibase	Altibase HDB	In-Memory, NewSQL	Free tier available	altibase.com
Amazon Web Services	DynamoDB	KV, Document, DBaaS	Free tier available	aws.amazon.com/dynamodb
Amazon Web Services	SimpleDB	Column Store	Available by request	aws.amazon.com/simpledb/
Apache Foundation	Apache Cassandra	KV, Wide Column	Open source	cassandra.apache.org
Apache Foundation	Apache Hbase	Wide Column	Open source	hbase.apache.org
Apache Foundation	Apache Ignite	In-Memory, Hadoop, Data Grid	Open source	ignite.apache.org
Apache Foundation	Apache OpenJPA	ORM	Open source	openjpa.apache.org
Apple	Core Data	ORM	Included in iOS and macOS	developer.apple.com/documentation/coredata

COMPANY	PRODUCT	CATEGORY	FREE TRIAL	WEBSITE
ArangoDB	ArangoDB	Graph, Document, KV	Open source	arangodb.com
Atlassian	ActiveObjects	ORM	Included in Jira and Confluence	developer.atlassian.com/docs/atlassian-platform-common-components/active-objects
Basho	Riak	Document, KV	Open source	basho.com/products/
CakeDC	CakePHP	ORM	Open source	cakephp.org
Canonical	Storm	ORM	Open source	storm.canonical.com
Clustrix	Clustrix OLTP Database	NewSQL, Transactional	Available by request	clustrix.com/oltp-database/
Couchbase	Couchbase Server	KV, Document, Data Caching	Open source	couchbase.com
CUBRID	CUBRID	RDBMS	Open source	cubrid.org
Datical	Datical	Application Release Automation for Databases	Demo available by request	datical.com
DBMaestro	Database Release and Automation	Continuous Delivery for Databases	14-day free trial	dbmaestro.com/database-release-and-automation/
Eclipse Foundation	EclipseLink	ORM	Open source	eclipse.org/eclipselink/
Embarcadero	InterBase	RDBMS	Available by request	embarcadero.com/products/interbase
EnterpriseDB	EDB Postgres Platform	RDBMS	Free tier available	enterprisedb.com/products/edb-postgres-platform
FairCom	c-treeACE	NewSQL, KV Direct Access	Available by request	faircom.com
Firebird	Firebird	RDBMS	Open source	firebirdsql.org
Google	BigTable	Column Store	Available by request	cloud.google.com/bigtable/
Google	Cloud Spaner	RDBMS, NewSQL	Available by request	cloud.google.com/spanner/
Gridgain	GridGain In-Memory Computing	In-Memory, Hadoop, Data Grid	Free tier available	gridgain.com/products/gridgain-products
Hazelcast IMDG	Hazelcast	In-Memory, Data Grid	Open source	hazelcast.com/products/
Hibernating Rhinos	RavenDB	Document	Open source	ravendb.net

COMPANY	PRODUCT	CATEGORY	FREE TRIAL	WEBSITE
IBM	IBM DB2	RDBMS	Free tier available	ibm.com/analytics/us/en/db2/
IBM	Informix	RDBMS, Transactional	Available by request	ibm.com/analytics/us/en/technology/informix/
InfluxData	InfluxEnterprise	Available by request	Available by request	influxdata.com/products/
Instaclustr	Managed Databases	Hosted Cassandra, Spark, Lucene, ElasticSearch, ScyllaDB, or Zeppelin	Available by request	instaclustr.com
Intersystems	Cache	Object-Oriented, Relational	Free tier available	intersystems.com/products/cache/#technology
JOOQ	JOOQ	ORM for Java	Free tier available	jooq.org
MariaDB	MariaDB TX	RDBMS, Document, MySQL Family	Open source	mariadb.com/products/solutions/oltp-database-tx
MarkLogic	MarkLogic	Transactional	Free tier available	marklogic.com/what-is-marklogic/
MemSQL	MemSQL	In-Memory, NewSQL	Free tier available	memsql.com/product/
Micro Focus	Vertica Enterprise	RDBMS, Column Store	Free tier available	vertica.com/product/on-premise/
Microsoft	Entity Framework	ORM	Part of .NET Framework	docs.microsoft.com/en-us/ef/
Microsoft	SQL Server 2016	RDBMS	180-day preview available	microsoft.com/en-us/sql-server/sql-server-2016
MongoDB	MongoDB	Document	Open source	mongodb.org
MyBatis	MyBatis	ORM	Open source	blog.mybatis.org/p/products.html
Neo4j	Neo4j	Graph	Free tier available	neo4j.com
Nhibernate	Nhibernate	ORM for .NET	Open source	nhibernate.info
NuoDB	NuoDB	NewSQL	Free tier available	nuodb.com
OpenText	OpenText Gupta SQLBase	RDBMS	Available by request	opentext.com/what-we-do/products/specialty-technologies/opentext-gupta-development-tools-databases/opentext-gupta-sqlbase
Oracle	JDBC	ORM	Free solution	oracle.com/technetwork/database/application-development/index-099369.html
Oracle	MySQL Community Edition	RDBMS	Open source	mysql.com
Oracle	Oracle Database	RDBMS, Document	Free solution	oracle.com/database/index.html

COMPANY	PRODUCT	CATEGORY	FREE TRIAL	WEBSITE
Oracle	Toplink	ORM	Free solution	oracle.com/technetwork/middleware/toplink/overview/index.html
OrientDB	OrientDB	Document, Graph, KV	Open source	orientechnologies.com
OrmLite	OrmLite	ORM	Open source	ormlite.com
Percona	Percona Server for MySQL	RDBMS, MySQL Family	Free solution	percona.com/software/mysql-database/percona-server
Pivotal	Gemfire	In-Memory, Data Grid	Free solution	pivotal.io
PostgreSQL	PostgreSQL	RDBMS	Open source	postgresql.org
Red Gate Software	SQL Toolkit	CI, Source Control, Change Management for Databases	14-day free trial	red-gate.com/products/
Red Hat	Hibernate	ORM	Open source	hibernate.org
Redis Labs	Redis	In-Memory, KV, Data Caching	Open source	redis.io
Redis Labs	Redis Cloud	In-Memory, KV, Data Caching, DBaaS	Free tier available	redislabs.com/products/redis-cloud/
SAP	SAP HANA Platform	In-Memory, Column-Oriented RDBMS	Available by request	sap.com/products/hana.html
ScaleOut	ScaleOut StateServer	In-Memory, Hadoop, Data Grid	30-day free trial	scaleoutsoftware.com/products/
Software AG	Adabas	Stream Processing	Free tier available	adabas.com
Solarwinds	Database Performance Analyzer	Database Performance Tuning	14-day free trial	solarwinds.com/database-performance-analyzer
Splice Machine	Splice Machine	NewSQL, RDBMS	Free tier available	splicemachine.com/product/features/
SQLite	SQLite	RDBMS	Open source	sqlite.org
Studio 3T	Studio 3T	IDE for MongoDB	Free solution	studio3t.com
SymmetricDS	SymmetricDS	Database Replication	Open source	symmetricds.org
Teradata	Aster Database	Specialist Analytic	Available by request	teradata.com
VoltDB	VoltDB	In-Memory, NewSQL	30-day free trial	voltdb.com

GLOSSARY

ACID (ATOMICITY, CONSISTENCY, ISOLATION, DURABILITY)
A term that refers to the model properties of database transactions, traditionally used for SQL databases.

BASE (BASIC AVAILABILITY, SOFT STATE, EVENTUAL CONSISTENCY)
A term that refers to the model properties of database transactions, specifically for NoSQL databases needing to manage unstructured data.

DATABASE MANAGEMENT SYSTEM (DBMS)
A suite of software and tools that manage data between the end user and the database.

DATA WAREHOUSE
A collection of individual computers that work together and appear to function as a single system. This requires access to a central database, multiple copies of a database on each computer, or database partitions on each machine.

DOCUMENT STORE
A type of database that aggregates data from documents rather than defined tables and is used to present document data in a searchable form.

FAULT TOLERANCE
A system's ability to respond to hardware or software failure without disrupting other systems.

GRAPH STORE
A type of database used for handling entities that have a large number of relationships, such as social graphs, tag systems, or any link-rich domain; it is also often used for routing and location services.

HADOOP
An Apache Software Foundation framework developed specifically for high scalability, data-intensive, distributed computing. It is used primarily for batch processing large datasets very efficiently.

HIGH AVAILABILITY (HA)
Refers to the continuous availability of resources in a computer system even after component failures occur. This can be achieved with redundant hardware, software solutions, and other specific strategies.

IN-MEMORY
As a generalized industry term, it describes data management tools that load data into RAM or flash memory instead of hard-disk or solid-state drives.

KEY-VALUE STORE
A type of database that stores data in simple key-value pairs. They are used for handling lots of small, continuous, and potentially volatile reads and writes.

NEWSQL
A shorthand descriptor for relational database systems that provide horizontal scalability and performance on par with NoSQL systems.

NOSQL
A class of database systems that incorporates other means of querying outside of traditional SQL and does not use standard relational structures.

OBJECT-RELATIONAL MAPPER (ORM)
A tool that provides a database abstraction layer to convert data between incompatible type systems using object-oriented programming languages instead of the database's query language.

PERSISTENCE
Refers to information from a program that outlives the process that created it, meaning it won't be erased during a shutdown or clearing of RAM. Databases provide persistence.

POLYGLOT PERSISTENCE
Refers to an organization's use of several different data storage technologies for different types of data.

RELATIONAL DATABASE
A database that structures interrelated datasets in tables, records, and columns.

REPLICATION
A term for the sharing of data so as to ensure consistency between redundant resources.

SCHEMA
A term for the unique data structure of an individual database.

SHARDING
Also known as "horizontal partitioning," sharding is where a database is split into several pieces, usually to improve the speed and reliability of an application.

STRONG CONSISTENCY
A database concept that refers to the inability to commit transactions that violate a database's rules for data validity.

STRUCTURED QUERY LANGUAGE (SQL)
A programming language designed for managing and manipulating data; used primarily in relational databases.

INTRODUCING THE

AI Zone

What's new with Predictive Analytics, Natural Language Processing, and Neural Nets? Check out DZone's new AI Zone – a central place for AI resources and tutorials.

Keep a pulse on the industry and go beyond the hype. This Zone gives you access to practical applications and use cases for AI technologies like: Machine Learning, Cognitive Computing, and Chatbots.

Visit the Zone

MACHINE LEARNING

COGNITIVE COMPUTING

CHATBOTS

DEEP LEARNING