DECSAI
Departamento de Ciencias de la Computación e I.A.
Universidad de Granada

Seguridad en bases de datos

© Fernando Berzal, berzal@acm.org

Seguridad en bases de datos

- Bases de datos relacionales (SQL)
  - Permisos SQL
  - SQL Call-Level Interface
  - Ataques por inyección
  - Herramientas de O/R Mapping
- Bases de datos NoSQL
  - Consistencia eventual
  - Ataques por inyección NoSQL
- Middleware: DDS [Data Distribution Service]

Seguridad en SQL

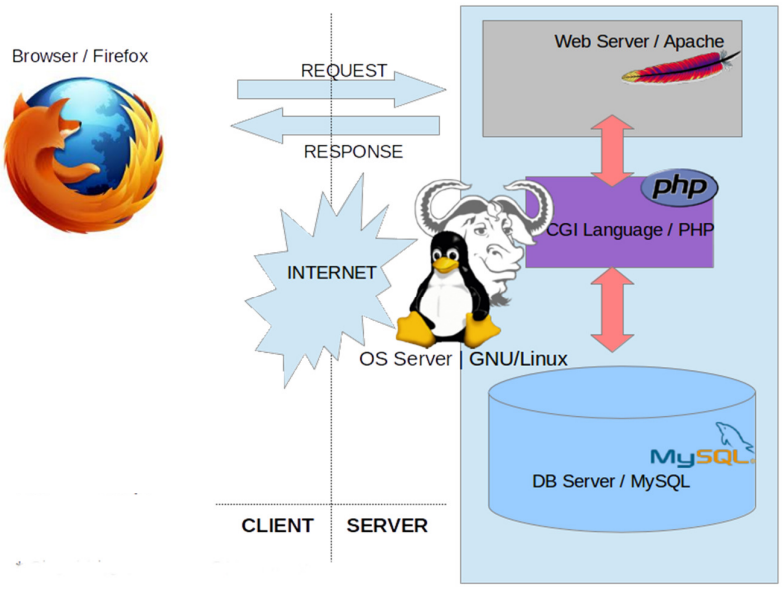
- SQL [Structured Query Language]
- Permisos SQL
- SQL Call-Level Interface [CLI]
- Ataques por inyección
- Herramientas de O/R Mapping

SQL

LAMP stack

SQL

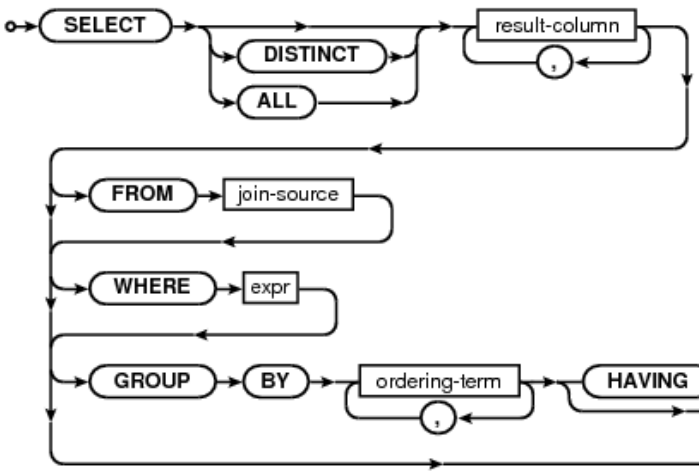
LAMP stack



SQL: Acceso a bases de datos

SQL

Structured Query Language



Permisos SQL

Usuarios y roles

- Usuarios identificados con contraseña mediante un proceso de login (p.ej. usuario específico para cada aplicación que usa la base de datos).

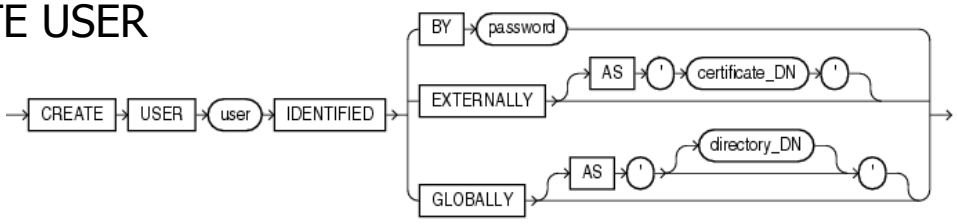
  p.ej. **Invitado**: guest
  Administrador: sys/system (Oracle), db2admin (IBM DB2), sa/dbo (SQL Server), root (MySQL), sysdba (InterBase)
  ...

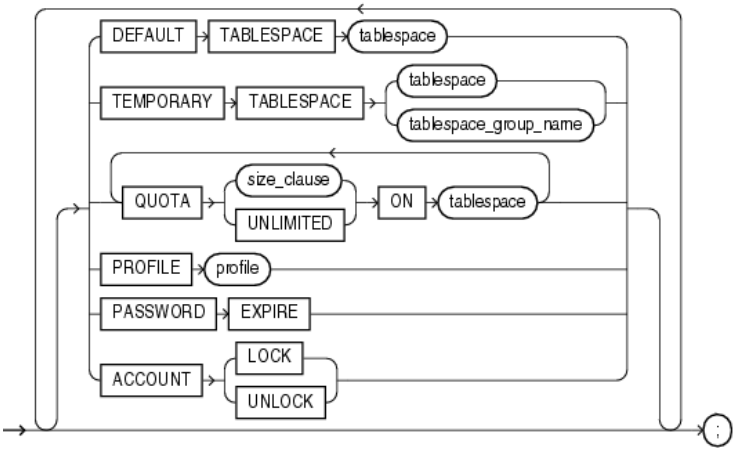
- Roles para simplificar la gestión de permisos:
  Un usuario puede estar asociado a múltiples roles.

Permisos SQL

Usuarios y roles

CREATE USER

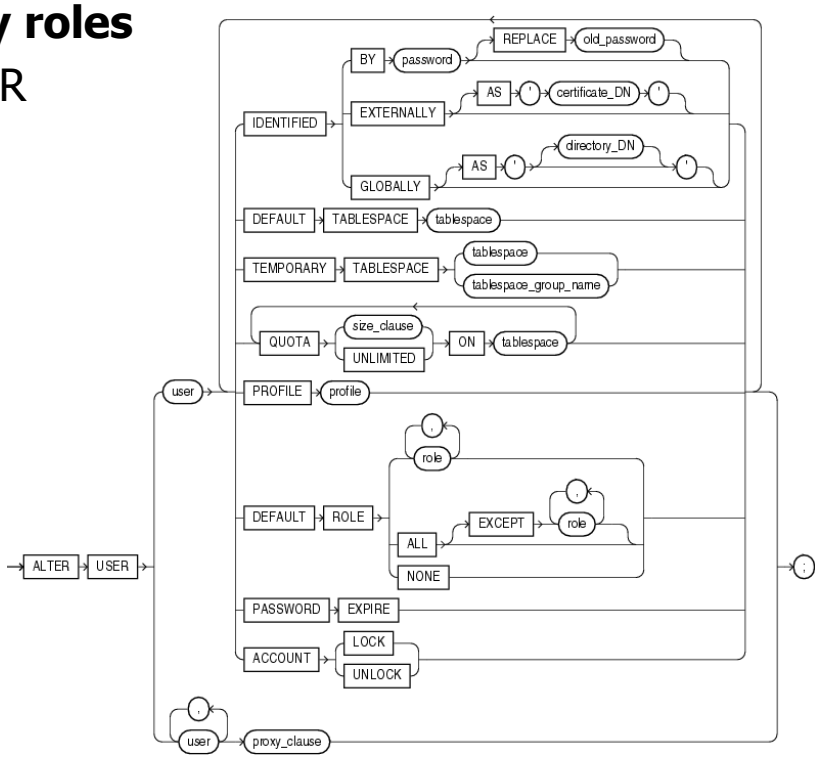




Permisos SQL

Usuarios y roles

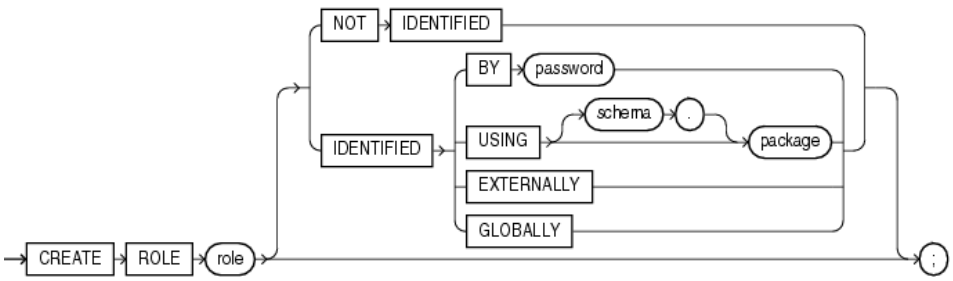
ALTER USER



Permisos SQL

Usuarios y roles

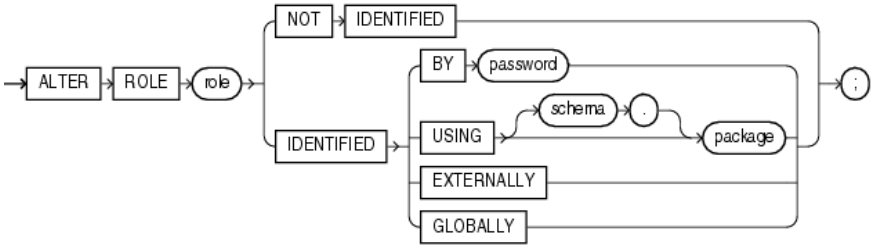
CREATE ROLE



Permisos SQL

Usuarios y roles

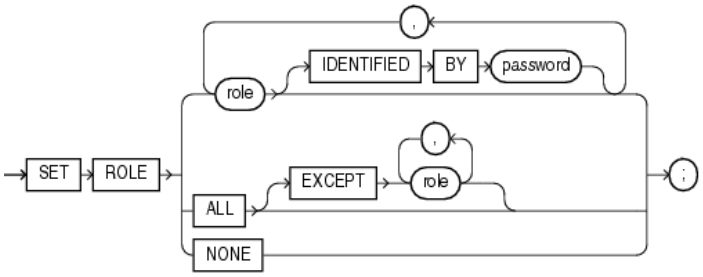
ALTER ROLE



Permisos SQL

Usuarios y roles

SET ROLE

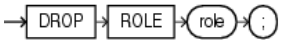


Permisos SQL

Usuarios y roles

DROP USER

DROP ROLE



Permisos SQL

Configuración de permisos

- Sobre tablas y vistas:
  SELECT, INSERT, UPDATE, DELETE
- Sobre procedimientos almacenados y funciones:
  EXECUTE
- Sobre bases de datos:
  CONNECT, BACKUP…

Permisos SQL

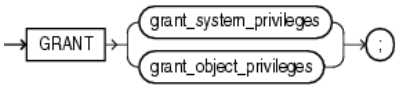
Configuración de permisos

- GRANT concede un permiso.
- DENY impide la concesión de un permiso (SQL Server).
- REVOKE elimina la concesión/prohibición de un permiso establecida mediante GRANT/DENY.

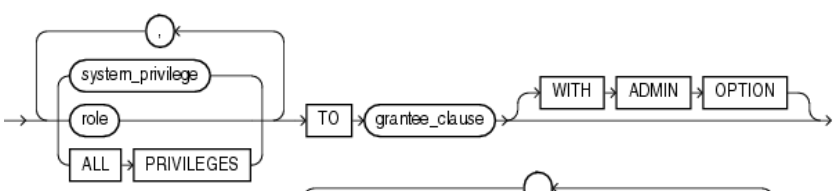
Permisos SQL

Configuración de permisos

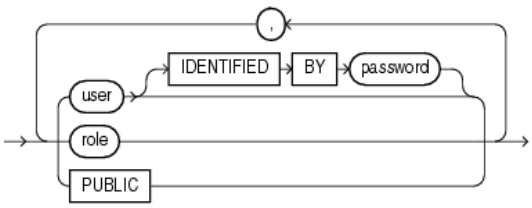
GRANT



System privileges



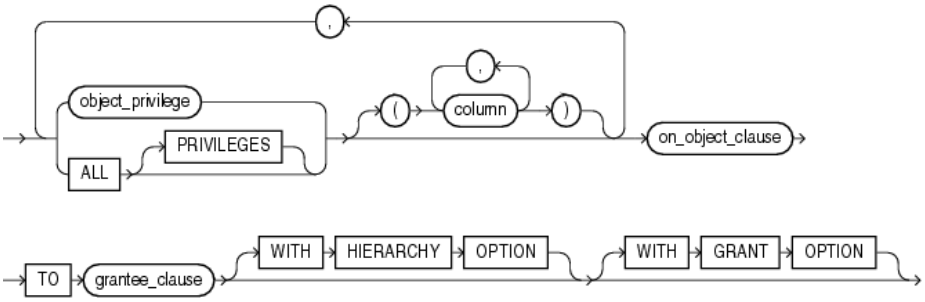
grantee clause



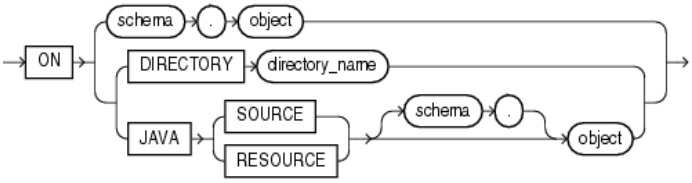
Permisos SQL

Configuración de permisos

Object privileges



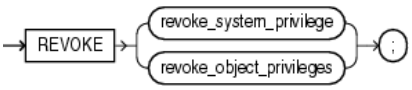
on object clause



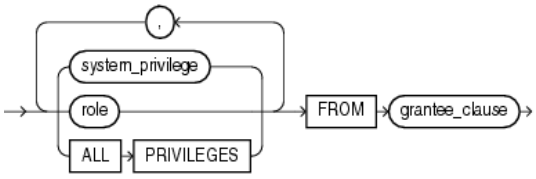
Permisos SQL

Configuración de permisos

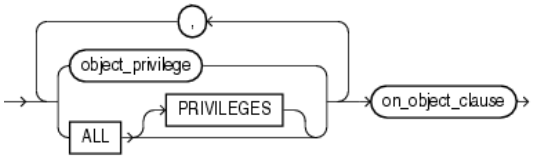
REVOKE

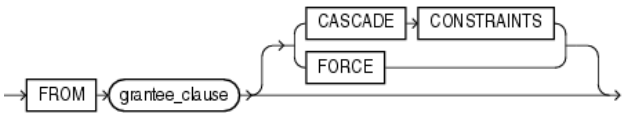


System



Objects





Permisos SQL

Buenas prácticas

- Mantenimiento mínimo (la vida lo más sencilla posible para el administrador de la base de datos).
  p.ej. Uso de roles (grupos de usuarios)
- Superficie mínima (reducir el número de lugares a través de los cuales recibir ataques).
  p.ej. Deshabilitar todo lo que no se utilice.
- Privilegios mínimos (si no lo necesitan, no se les concede el permiso).
  p.ej. Permisos sobre vistas o SP, no tablas.

SQL Call-Level Interface [CLI]

ISO SQL/CLI (estándar SQL-92)

API [Application Programming Interface] para acceder a una base de datos relacional utilizando sentencias SQL desde el código de una aplicación:

- ODBC [Open DataBase Connectivity] de Microsoft.
- JDBC [Java DataBase Connectivity] en Java.
- ADO.NET [ActiveX Data Objects] para .NET.
- DB-API en Python.

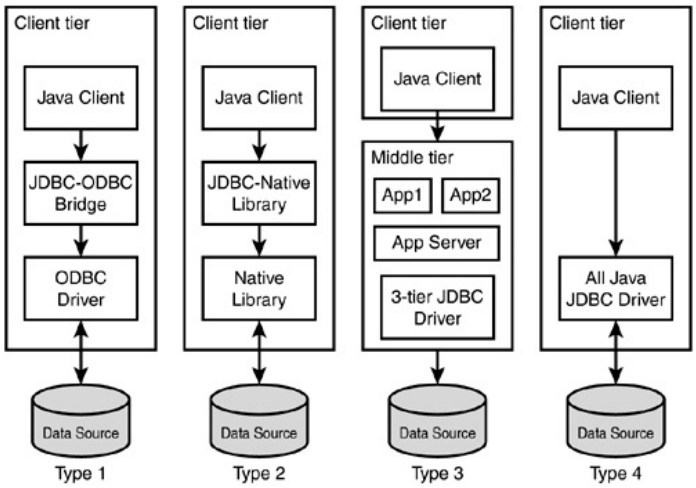
…

SQL Call-Level Interface [CLI]

JDBC [Java DataBase Connectivity]

Drivers JDBC

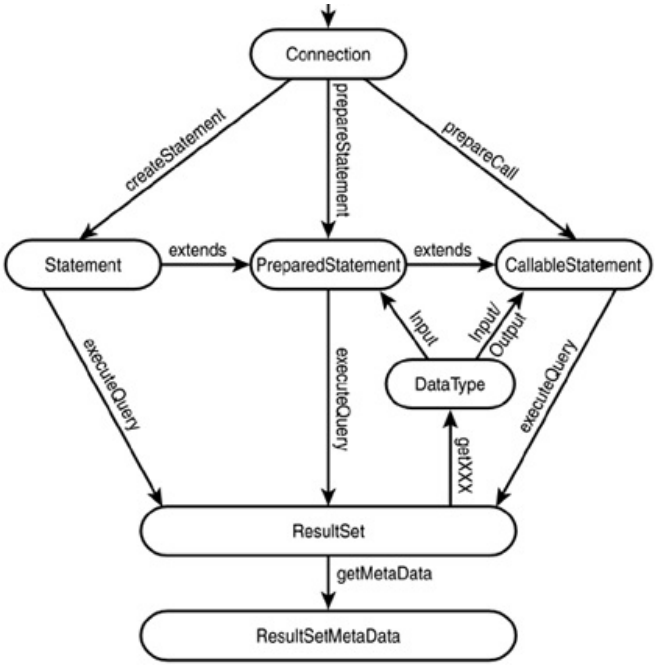
- JDBC-ODBC bridge
- Native API driver
- Middleware driver
- Pure Java driver



SQL Call-Level Interface [CLI]

JDBC [Java DataBase Connectivity]

Principales clases
(paquete java.sql)



SQL Call-Level Interface [CLI]

JDBC [Java DataBase Connectivity]

Establecimiento de conexiones

```
try {
  // 1. Cargamos el driver JDBC de nuestro DBMS (p.ej. Oracle)
  Class.forName("oracle.jdbc.driver.OracleDriver");
  // 2. Establecemos una conexión con la BD
  Connection connection = DriverManager.getConnection(
    "jdbc:oracle:thin:@localhost:1521:SID",
    "usuario", "password");
  …
} catch (ClassNotFoundException driverError) {
    // Driver no encontrado
} catch (SQLException sqlError) {
    // Error SQL (p.ej. usuario/clave incorrectas)
}
```


SQL Call-Level Interface [CLI]

JDBC [Java DataBase Connectivity]

Ejecución de sentencias SQL

```
Statement statement = connection.createStatement();
ResultSet set = statement.executeQuery("SELECT * FROM clients");

// Resultado de la consulta
while (set.next()) {
    … set.getString("name"); …
    … set.getString("address"); …
    … set.getDate("birthdate"); …
    … set.getBigDecimal("balance"); …
}
```

SQL Call-Level Interface [CLI]

JDBC [Java DataBase Connectivity]

Tipos de datos SQL

Tipo de dato SQL	Método JDBC
CHAR/VARCHAR	String getString()
DECIMAL/NUMERIC	java.math.BigDecimal getBigDecimal()
FLOAT/DOUBLE	double getDouble()
INTEGER	int getInt()
DATE	java.sql.Date getDate()
TIME	java.sql.Time getTime()
TIMESTAMP	java.sql.Timestamp getTimestamp()
BINARY	byte[] getBytes()
BLOB	java.io.InputStream getBinaryStream() java.sql.Blob getBlob()

SQL Call-Level Interface [CLI]

JDBC [Java DataBase Connectivity]

Peligro: Inyección de código SQL

```
String sql = "select * from user where username='" + username
           + "' and password='" + password + "'";
stmt = conn.createStatement();
rs = stmt.executeQuery(sql);
if (rs.next()) {
   out.println("Successfully logged in");
} else {
   out.println("Invalid username and/or password");
}
```

Entrada del usuario: username = admin' OR '1'='1

```
select * from user
where username='admin' OR '1'='1' and password=' '
```

SQL Call-Level Interface [CLI]

JDBC [Java DataBase Connectivity]

Para evitar ataques por inyección de código SQL…

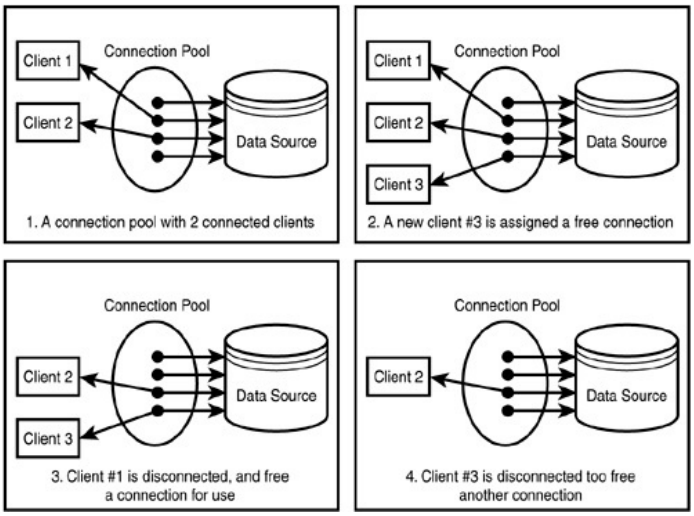
PreparedStatement

```
PreparedStatement statement = connection.prepareStatement (
  "UPDATE clients SET address = ? WHERE ID = ?");

statement.setString (1, "Nueva dirección");
statement.setInt (2, 123456 );
statement.execute();

// Resultado
… statement.getUpdateCount() …  // getResultSet() para consultas
```


SQL Call-Level Interface [CLI]

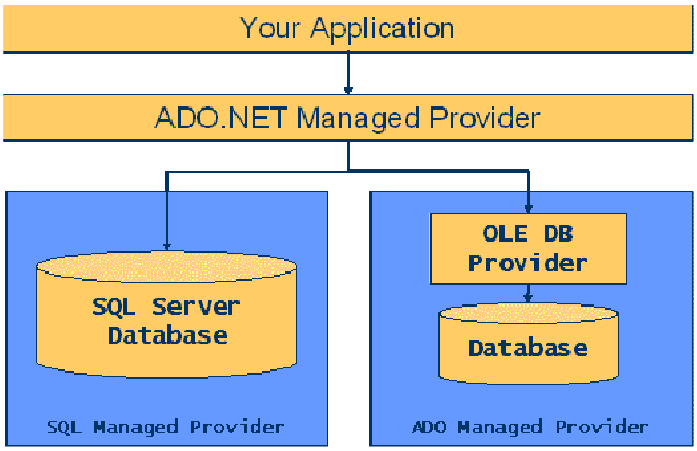
JDBC [Java DataBase Connectivity]

Pool de conexiones



SQL Call-Level Interface [CLI]

ADO.NET (plataforma .NET)



SQL Call-Level Interface [CLI]

ADO.NET (plataforma .NET)

Establecimiento de conexiones

```
string connectionString = "User ID=sa;Initial Catalog=MYDB;"
                        + "Data Source=MYSERVER";
SqlConnection connection = new SqlConnection(connectionString);
```

Ejecución de consultas (usando DataSet)

```
SqlDataAdapter adapter = new SqlDataAdapter();
DataSet dataset = new DataSet();

string sqlQuery = "SELECT * FROM Customers";
adapter.SelectCommand = new SqlCommand(sqlQuery, connection);

connection.Open();
adapter.Fill(dataset);
connection.Close();
```

SQL Call-Level Interface [CLI]

ADO.NET (plataforma .NET)
Ejecución de consultas (usando DataReader)

```
string sqlQuery = "SELECT Name FROM Users";
SqlCommand sqlCommand = new SqlCommand(sqlQuery, connection);

connection.Open();
SqlDataReader reader = sqlCommand.ExecuteReader();

while (reader.Read()) {
   … reader.GetString(0) …
}

myReader.Close();
connection.Close();
```


SQL Call-Level Interface [CLI]

ADO.NET (plataforma .NET)
Ejecución de sentencias SQL

```
string sqlInsert = "INSERT INTO Clients(Name) VALUES (@Name)";
SqlCommand sqlCommand = new SqlCommand(sqlInsert, connection);

SqlParameter  param = sqlCommand.Parameters.Add (
    new SqlParameter("@Name", SqlDbType.VarChar, 100));

param.Value = ...

connection.Open();
sqlCommand.ExecuteNonQuery();
connection.Close();
```

SQL Call-Level Interface [CLI]

SQLite (Android)

Esquema (local)

```
public final class FeedReaderContract {
  public FeedReaderContract() {}

  /* Inner class that defines the table contents */
  public static abstract class FeedEntry implements BaseColumns {
    public static final String TABLE_NAME = "entry";
    public static final String COLUMN_NAME_ENTRY_ID = "entryid";
    public static final String COLUMN_NAME_TITLE = "title";
    public static final String COLUMN_NAME_SUBTITLE = "subtitle";
    ...
  }
}
```


SQL Call-Level Interface [CLI]

SQLite (Android)

Base de datos (local): Creación

```
public class FeedReaderDbHelper extends SQLiteOpenHelper {
    // If you change the database schema,
    // you must increment the database version.
    public static final int DATABASE_VERSION = 1;
    public static final String DATABASE_NAME = "FeedReader.db";

    public FeedReaderDbHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(SQL_CREATE_TABLE);
    }
```

SQL Call-Level Interface [CLI]

SQLite (Android)

Base de datos (local): Actualizaciones

```
    public void onUpgrade
      (SQLiteDatabase db, int oldVersion, int newVersion) {
        // Data cache, just discard the data and start over
        db.execSQL(SQL_DROP_TABLE);
        onCreate(db);
    }
    public void onDowngrade
      (SQLiteDatabase db, int oldVersion, int newVersion) {
        onUpgrade(db, oldVersion, newVersion);
    }
}
```


SQL Call-Level Interface [CLI]

SQLite (Android)

Almacenamiento de datos

```
FeedReaderDbHelper mDbHelper
  = new FeedReaderDbHelper(getContext());
SQLiteDatabase db
  = mDbHelper.getWritableDatabase();

// Map of values, where column names are the keys
ContentValues values = new ContentValues();
values.put(FeedEntry.COLUMN_NAME_ENTRY_ID, id);
values.put(FeedEntry.COLUMN_NAME_TITLE, title);
values.put(FeedEntry.COLUMN_NAME_CONTENT, content);

// Insert the new row
long newRowId = db.insert(FeedEntry.TABLE_NAME,
         FeedEntry.COLUMN_NAME_NULLABLE, values);
```

SQL Call-Level Interface [CLI]

SQLite (Android)

Consulta de datos

```
SQLiteDatabase db = mDbHelper.getReadableDatabase();

Cursor c = db.query(        // .. or db.rawQuery(sqlStatement)
    FeedEntry.TABLE_NAME,   // The table to query
    projection,             // The columns to return
    selection,              // The columns for the WHERE clause
    selectionArgs,          // The values for the WHERE clause
    null,                   // don't group the rows
    null,                   // don't filter by row groups
    sortOrder );            // The sort order

cursor.moveToFirst();
long itemId = cursor.getLong(
    cursor.getColumnIndexOrThrow(FeedEntry._ID));
```


SQL Call-Level Interface [CLI]

SQLite (Android)

Borrado y actualizaciones

```
// 'where' clause
String selection = FeedEntry.COLUMN_NAME_ENTRY_ID + " LIKE ?";
String[] selectionArgs = { String.valueOf(rowId) };

// SQL statement
db.delete(table_name, selection, selectionArgs);

// New values
ContentValues values = new ContentValues();
values.put(FeedEntry.COLUMN_NAME_TITLE, title);
// SQL statement
int count = db.update(FeedReaderDbHelper.FeedEntry.TABLE_NAME,
    values, selection, selectionArgs);
```

SQL Call-Level Interface [CLI]

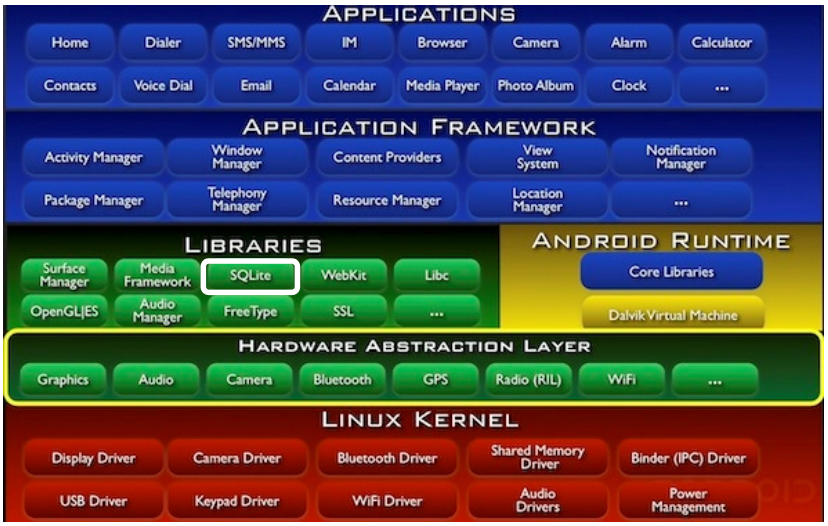
SQLite (Android)

Utilizando SQL directamente

```
// SQL query
Cursor cursor = db.rawQuery (
                  "SELECT id, name FROM people WHERE id = ?",
                  new String[] {"1234"} );

// SQL statement
db.execSQL ( string );

   // NOTA: Google recomienda utilizar
   // - db.insert(String, String, ContentValues)
   // - db.update(String, ContentValues, String, String[])
   // - db.delete(String, String, String[])
```


SQL Call-Level Interface [CLI]

SQLite en la plataforma Android



SQL Call-Level Interface [CLI]

SQLite (Android)
PELIGRO

- A diferencia de un DBMS "completo", SQLite está diseñado para sistemas empotrados, por lo que la seguridad de los datos depende de los controles establecidos por el sistema operativo: no requiere login.

- No se puede limitar el acceso a parte de los datos: es imposible establecer permisos de usuarios (grant/revoke) y resulta complejo establecer cuotas (puerta abierta a posibles ataques).

SQL Call-Level Interface [CLI]

SQLite (Android)
PELIGRO

- Los datos, por defecto, se almacenan sin encriptar, en un fichero que puede ser fácil de robar (dispositivos Android "rooteados" y aplicaciones web mal configuradas).

- POSIBLE SOLUCIÓN: Existen extensiones que permiten encriptar los ficheros de la base de datos (no en Android)

  p.ej. SQLCipher
  http://sqlcipher.net/
  SEE [SQLite Encryption Extension]
  http://www.hwaci.com/sw/sqlite/see.html

SQL Call-Level Interface [CLI]

SQLite: SQLCipher

http://sqlcipher.net/

```
% hexdump -C unencrypted-sqlite.db
00000000  53 51 4c 69 74 65 20 66  6f 72 6d 61 74 20 33 00  |SQLite format 3.|
00000010  04 00 01 01 00 40 20 20  00 00 00 02 00 00 00 03  |.....@  ........|
000003b0  00 00 00 00 00 00 00 00  00 00 00 00 00 41 01 06  |.............A..|
000003c0  17 1b 1b 01 5b 74 61 62  6c 65 73 65 63 72 65 74  |....[tablesecret|
000003d0  73 73 65 63 72 65 74 73  03 43 52 45 41 54 45 20  |ssecrets.CREATE |
000003e0  54 41 42 4c 45 20 73 65  63 72 65 74 73 28 69 64  |TABLE secrets(id|
000003f0  2c 20 70 61 73 73 77 6f  72 64 2c 20 6b 65 79 29  |, password, key)|
00000bd0  00 00 00 00 00 00 00 00  00 00 00 00 00 21 01 04  |.............!..|
00000be0  25 1d 1f 4c 61 75 6e 63  68 20 43 6f 64 65 73 70  |%..Launch Codesp|
00000bf0  61 24 24 77 6f 72 64 70  72 6f 6a 65 74 69 6c 65  |a$$wordprojetile|
```

```
% hexdump -C encrypted-sqlcipher.db
00000000  de ab bc 3a 40 2b 5d 00  b0 d2 9e 3b 75 91 76 73  |...:@+]....;u.vs|
00000010  bc 41 70 0c 8c ab a0 7a  37 eb a2 a8 a9 27 a5 0a  |.Ap....z7....'..|
00000020  38 c9 0b 9c 06 57 78 96  67 a2 e5 78 f8 8c 58 f3  |8....Wx.g..x..X.|
00000030  ea 7c c6 23 14 8a 75 33  d0 a5 2c 30 2e e1 a4 96  |.|.#..u3..,0....|
00000040  b1 c6 5a 21 67 0a 31 bb  3b de a2 d4 80 b4 60 e3  |..Z!g.1.;.....`.|
00000050  05 b0 75 04 f2 26 66 ed  c7 4e 7e 9c ac 2e ec 1d  |..u..&f..N~.....|
00000060  2d fc 31 b4 32 ce 24 0a  d0 23 71 b0 1f 21 12 2c  |-.1.2.$..#q..!.,|
00000070  92 af 8e d9 de ac 76 e6  20 62 56 c6 f5 05 f5 b3  |......v. bV.....|
00000080  53 d0 5f 4c 5e ec 5b 8a  be e7 d1 46 f0 d9 dc b9  |S._L^.[....F....|
00000090  a3 59 d6 63 a4 ae cf d8  e4 82 29 83 dd c7 86 13  |.Y.c......).....|
```

Full Database Encryption for SQLite

Open source extension that provides transparent 256-bit AES encryption of database files.

SQL Call-Level Interface [CLI]

SQLite (Android)

¿Y qué hacemos en Android?

- Encriptar los datos antes de introducirlos en la base de datos (aunque eso puede interferir con nuestra capacidad para realizar consultas).
- Mantener los datos en un servidor, no en el dispositivo móvil (no siempre es una solución viable).
- Android admite almacenar datos como pares (clave,valor) mediante "Shared Preferences": ¿seguro? Ummm…
  https://developer.android.com/guide/topics/data/data-storage.html#pref

Ataques por inyección

Top 10: Open Web Application Security Project

https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

1. **Inyección** (SQL, ORM, XML, SSI, buffer overflow…)
2. Autentificación y gestión de sesiones de usuario
3. XSS [Cross Site Scripting]
4. Referencias directas a objetos
5. Fallos de configuración (p.ej. PHP)
6. Exposición de datos sensibles
7. Controles de acceso (p.ej. URLs de ficheros PDF)
8. CSRF [Cross Site Request Forgery]
9. Redirecciones no validadas (redirect/forward)
10. Componentes con vulnerabilidades conocidas

Ataques por inyección

Ataques por inyección

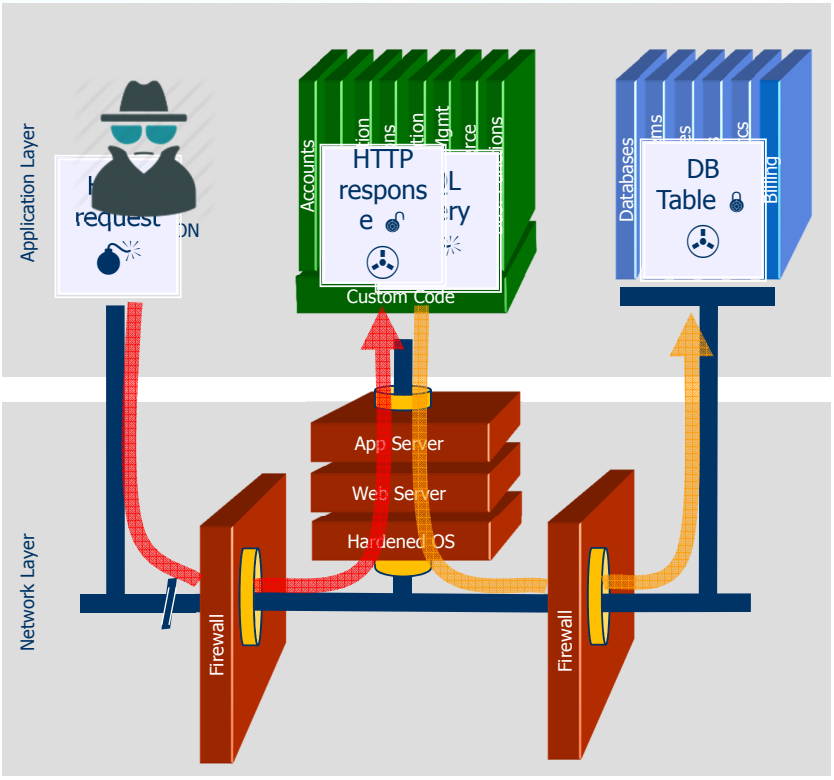
INYECCIÓN: Conseguir que una aplicación ejecute comandos por medio del envío de datos a un intérprete.

EJEMPLOS: SQL, shell SO, XPath, Hibernate…

La inyección de código SQL es muy común, pese a ser muy sencilla de prevenir :-(

https://www.owasp.org/index.php/Top_10_2013-A1-Injection

Ataques por inyección



Account: ' OR 1=1 --

SKU:

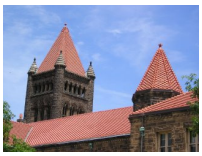
Submit

1. Application presents a form to the attacker
2. Attacker sends an attack in the form data
3. Application forwards attack to the database in a SQL query
4. Database runs query containing attack and sends encrypted results back to application
5. Application decrypts data as normal and sends results to the user

Ataques por inyección

Ataques por inyección

EJEMPLO @ http://elvex.ugr.es



```
[pid 4652:tid 900] [client 184.107.172.42:38400]
 AH00128: File does not exist: …/php-ofc-library/ofc_upload_image.php
...
```

Ejecución de código PHP ("Open Flash Chart")

http://www.securityfocus.com/bid/37314/exploit

http://www.example.com/libs/open-flash-chart/php-ofc-library/ofc_upload_image.php?name=shell.php&HTTP_RAW_POST_DATA=<?system($_GET['cmd']);?>

Ataques por inyección

Ataques por inyección

EJEMPLO @ http://elvex.ugr.es

```
92.63.97.93 - - [29/Oct/2015:02:19:47 +0100] "POST /cgi-bin/php/%63%67
%69%6E/%70%68%70?%2D%64+%61%6C%75%6F%6E+%2D%64+%6D%6F%64+%2D%64+%73%75
%68%6F%6E%3D%6F%6E+%2D%64+%75%6E%63%74%73%3D%22%22+%2D%64+%64%6E%65+
%2D%64+%61%75%74%6F%5F%70%72%%74+%2D%64+%63%67%69%2E%66%6F%72%63%65
%5F%72%65%64%69%72%65%63%74%3D%30+%2D%64+%74%5F%3D%30+%2D%64+%75%74
+%2D%6E HTTP/1.1" 404 218 ...
```

Ejecución de código PHP

URL decodificada

/cgi-bin/php/cgin/php?-d+aluon+-d+mod+-d+suhon=on+-d+uncts=""+-d+dne+-d+auto_pr%t+-d+cgi.force_redirect=0+-d+t_=0+-d+ut+-n

Ataques por inyección

Ataques por inyección

RECOMENDACIONES

- Evitar el intérprete por completo, o bien…
- Utilizar una interfaz que permita el uso de variables para distinguir entre código y datos (p.ej. "prepared statements", procedimientos almacenados…)
- Codificar siempre las entradas provenientes del usuario antes de pasárselas al intérprete (p.ej. "escaping")
- Validar todas las entradas del usuario.
- Minimizar los privilegios de la aplicación en la BD para reducir el impacto del fallo de seguridad.

Ataques por inyección

Ataques por inyección

EJEMPLO EN JAVA: JDBC

VERSIÓN NO SEGURA

```
String query = "SELECT account_balance FROM user_data"
  +" WHERE user_name = "+ request.getParameter("customerName");
Statement statement = connection.createStatement( … );
ResultSet results = statement.executeQuery( query );
```

VERSIÓN SEGURA: PREPARED STATEMENTS

```
String custname = request.getParameter("customerName");
String query = "SELECT account_balance FROM user_data"
  +" WHERE user_name = ? ";
PreparedStatement pstmt = connection.prepareStatement( query );
pstmt.setString( 1, custname);
ResultSet results = pstmt.executeQuery( );
```


Ataques por inyección

Ataques por inyección

EJEMPLO EN JAVA: JDBC

VERSIÓN SEGURA: PROCEDIMIENTO ALMACENADO

```
String custname = request.getParameter("customerName");
CallableStatement cs = connection.prepareCall(
  "{call sp_getAccountBalance(?)}");
cs.setString(1, custname);
ResultSet results = cs.executeQuery();
```

VERSIÓN SEGURA: VALIDACIÓN DE ENTRADAS (WHITE LISTS)

```
switch(PARAM) {
  case "Value1": tableName = “Table1"; break;
  …
  default:       throw new InputValidationException
                    ("unexpected value for table name");
}
```

Otros problemas habituales

Fallos de configuración

En cualquier parte del sistema que da soporte a la aplicación web: sistema operativo, servidor HTTP, contenedor / servidor de aplicaciones, intérpretes (p.ej. PHP), DBMS, intranet…

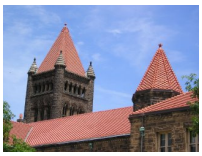
Impacto:

- Instalación de puertas traseras [backdoors].
- Accesos no autorizados
- …

Otros problemas habituales

Fallos de configuración

EJEMPLO @ http://elvex.ugr.es



```
[pid 4616:tid 848] [client 75.119.221.245:34465]
 AH00128: File does not exist: …/www/elvex/wp-admin/
[pid 4100:tid 912] [client 85.25.136.37:18395]
 AH00128: File does not exist: …/www/elvex/test/wp-admin/
[pid 4100:tid 912] [client 180.210.204.141:59963]
 AH00128: File does not exist: …/www/elvex/wordpress/wp-admin/
[pid 4100:tid 868] [client 108.171.217.244:35884]
 AH00128: File does not exist: …/www/elvex/blog/wp-admin/
[pid 4100:tid 904] [client 216.104.160.77:60732]
 AH00128: File does not exist: …/www/elvex/wp/wp-admin/
[pid 4100:tid 872] [client 193.143.77.22:38799]
 AH00128: File does not exist: …/www/elvex/old/wp-admin/
...
```

Intentos de acceso a una aplicación web (WordPress)

Otros problemas habituales

Exposición de datos sensibles

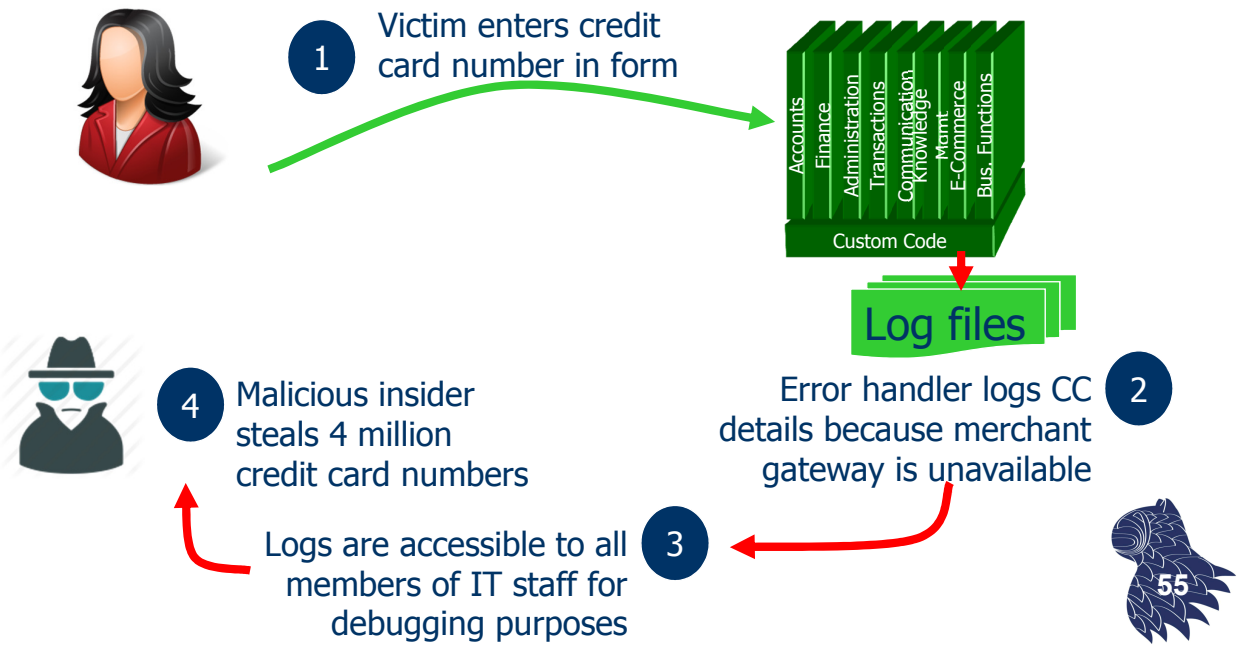
- Falta de seguridad en el almacenamiento (bases de datos, ficheros, logs, copias de seguridad…) o la transmisión de datos sensibles (uso de HTTPS).

Impacto:

- Modificación y robo de datos confidenciales.
- Pérdida de confianza de los clientes.
- Costes derivados de la brecha de seguridad.
- Problemas legales.

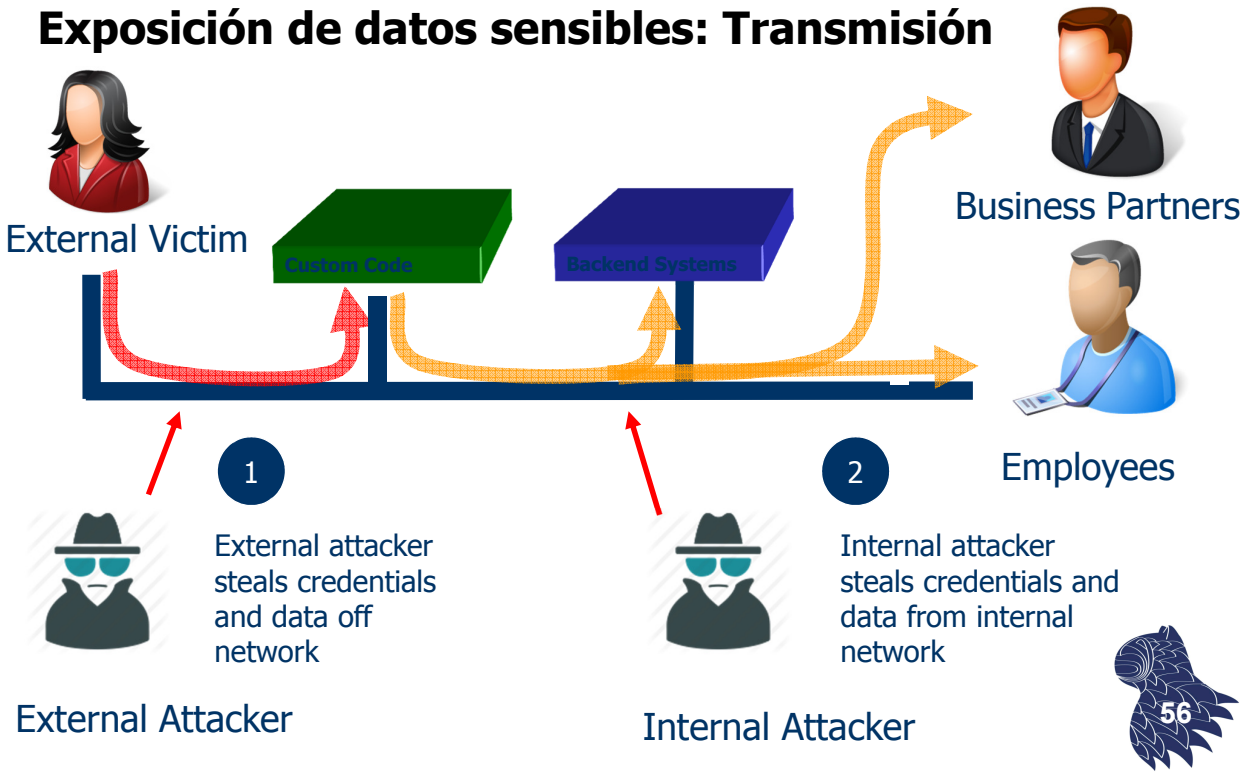
Otros problemas habituales

Exposición de datos sensibles: Almacenamiento



Otros problemas habituales

Exposición de datos sensibles: Transmisión



Otros problemas habituales

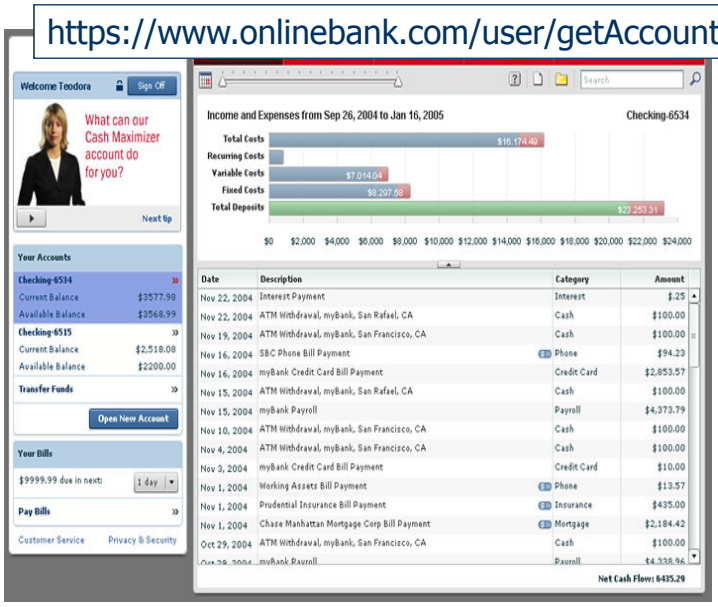
Controles de acceso

- Protección del acceso a URLs o a funciones de la aplicación a las que se hace referencias por medio de URLs+parámetros.
- Error habitual:
  Control de acceso en la capa de presentación
  Conociendo la URL usada por la aplicación, el atacante puede tener acceso a funciones a las que no debería.

Otros problemas habituales

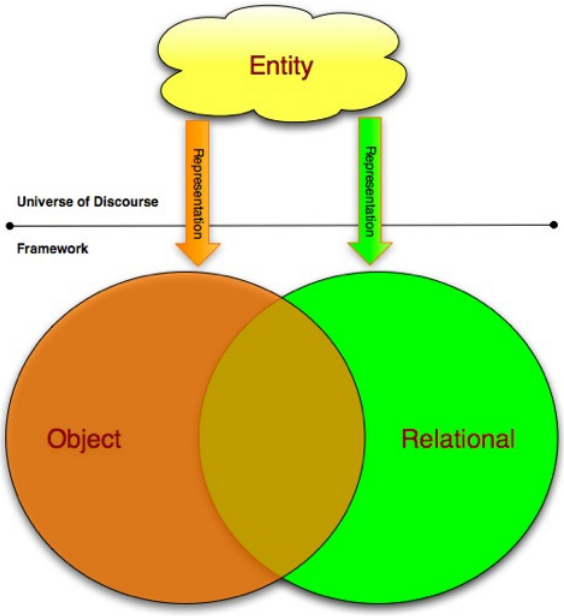
Controles de acceso

https://www.onlinebank.com/user/getAccounts



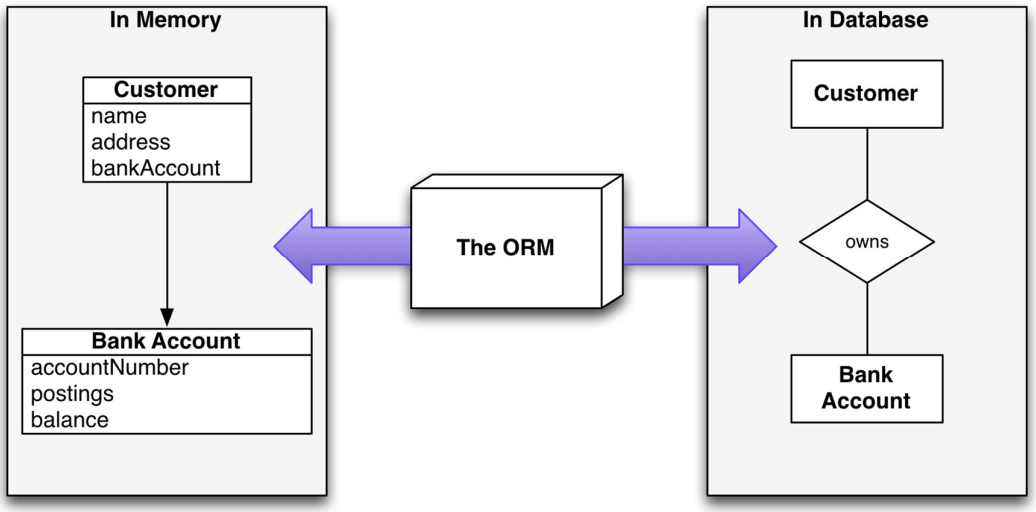
- Attacker notices the URL indicates his role
  /user/getAccounts
- He modifies it to another directory (role)
  /admin/getAccounts, or
  /manager/getAccounts
- Attacker views more accounts than just their own

O/R Mapping



The object-relational impedance mismatch
http://impedancemismatch.com/

O/R Mapping



O/R Mapping

Ejemplo: C#

Usando ADO.NET (CLI estándar para .NET):

```
String sql = "SELECT ... FROM clientes WHERE id = 10";
DbCommand cmd = new DbCommand(connection, sql);
Result res = cmd.Execute();
String name = res[0]["FIRST_NAME"];
```

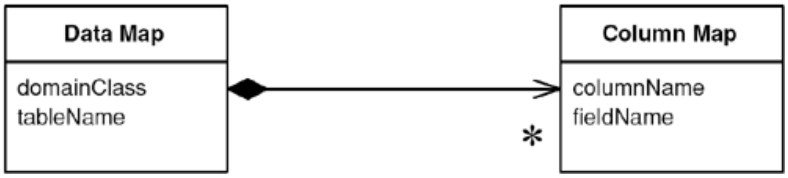
Usando ORM:

```
Client client = repository.GetClient(10);
String name = client.getFirstName();
```

O/R Mapping

Herramientas de O/R Mapping

En vez de programar manualmente la correspondencia entre objetos y tablas, se pueden utilizar metadatos para especificar la correspondencia y automatizar el proceso.

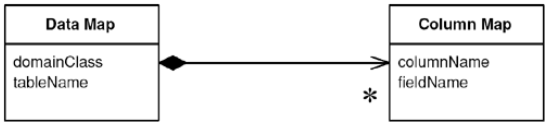


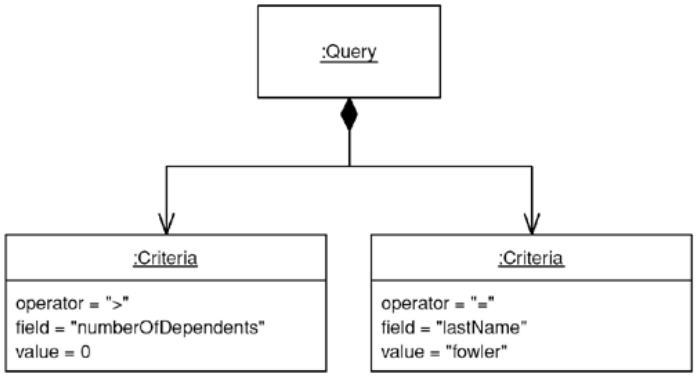
O/R Mapping

Herramientas de O/R Mapping

Realización de consultas:

"Query objects"

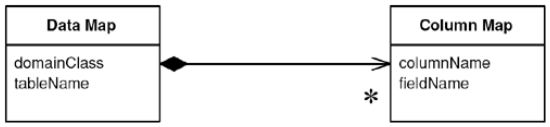


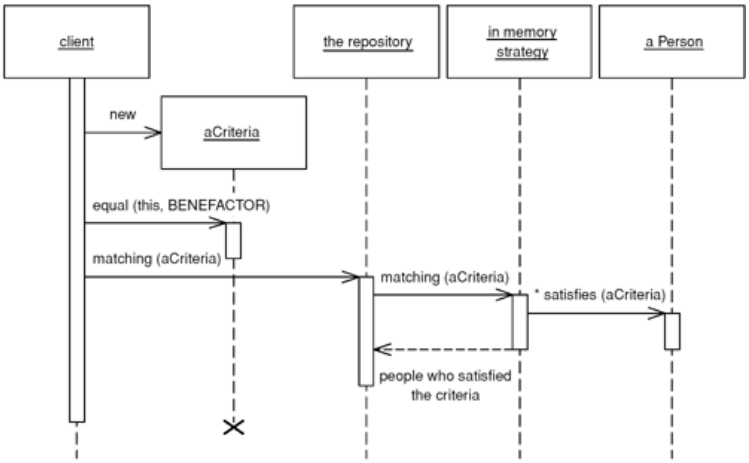


O/R Mapping

Herramientas de O/R Mapping

Almacenamiento de datos:
"Repositories"

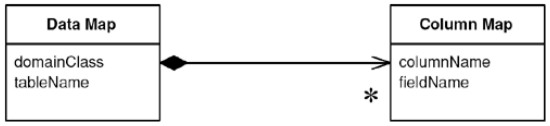




O/R Mapping

Herramientas de O/R Mapping

- JDO [Java Data Objects]



- JPA [Java Persistence API]: Lenguaje de consulta JPQL
- DataNucleus (JDO & JPA), e.g. Google App Engine
- Hibernate (Java, JPA) & Nhibernate (.NET)
- iBATIS (Java, .NET, Ruby) → MyBATIS (Java)

Herramientas de O/R Mapping

DataNucleus (JDO & JPA)

1. Clases en Java

```
public class Product
{
    String name;
    String description;
    double price;
    …
}
```

```
public class Book extends Product
{
    String author;
    String isbn;
    String publisher;
    …
}
```


Herramientas de O/R Mapping

DataNucleus (JDO & JPA)

2a. Persistencia (anotaciones)

```
@PersistenceCapable
public class Product
{
    String name;
    String description;
    double price;
    …
}
```

```
@PersistenceCapable
public class Book extends Product
{
    String author;
    String isbn;
    String publisher;
    …
}
```

Herramientas de O/R Mapping

DataNucleus (JDO & JPA)

2b. Persistencia (XML)

```
<?xml version="1.0"?>
<!DOCTYPE orm PUBLIC "-//Sun Microsystems, Inc.//DTD Java Data Objects Metadata 2.0//EN"
    "http://java.sun.com/dtd/orm_2_0.dtd">
<orm>
  <package name="org.datanucleus.samples.jdo.tutorial">
    <class name="Product" identity-type="datastore" table="JDO_PRODUCTS">
      <inheritance strategy="new-table"/>
      <field name="name">
         <column name="PRODUCT_NAME" length="100" jdbc-type="VARCHAR"/></field>
      <field name="description">
         <column length="255" jdbc-type="VARCHAR"/></field></class>
    <class name="Book" identity-type="datastore" table="JDO_BOOKS">
      <inheritance strategy="new-table"/>
      <field name="isbn">
          <column length="20" jdbc-type="VARCHAR"/></field>
      <field name="author">
          <column length="40" jdbc-type="VARCHAR"/></field>
      <field name="publisher">
          <column length="40" jdbc-type="VARCHAR"/></field></class>
    </package>
</orm>
```


Herramientas de O/R Mapping

DataNucleus (JDO & JPA)

3. Instrumentación de las clases: JDO "Enhancers"

Usando Ant

```
ant enhance
```

Usando Maven

```
mvn datanucleus:enhance
```

Manualmente

```
java -cp … org.datanucleus.enhancer.DataNucleusEnhancer *.java
```

Herramientas de O/R Mapping

DataNucleus (JDO & JPA)

4. Generación automática del esquema de la base de datos

Fichero de configuración (datanucleus.properties)

```
javax.jdo.PersistenceManagerFactoryClass=org.datanucleus.jdo.JDOPersistenceManagerFactory
javax.jdo.option.ConnectionDriverName=org.hsqldb.jdbcDriver
javax.jdo.option.ConnectionURL=jdbc:hsqldb:mem:nucleus1 javax.jdo.option.ConnectionUserName=sa
javax.jdo.option.ConnectionPassword= javax.jdo.option.Mapping=hsql
datanucleus.autoCreateSchema=true datanucleus.validateTables=false
datanucleus.validateConstraints=false
```

Usando Ant

```
ant createschema
```

Usando Maven

```
mvn datanucleus:schema-create
```

Manualmente

```
java –cp … org.datanucleus.store.rdbms.SchemaTool
     -props datanucleus.properties -create *.java
```

Herramientas de O/R Mapping

DataNucleus (JDO & JPA)

5. Uso desde una aplicación: CREATE

```
PersistenceManagerFactory pmf =
  JDOHelper.getPersistenceManagerFactory("datanucleus.properties");
PersistenceManager pm = pmf.getPersistenceManager();
Transaction tx=pm.currentTransaction();
try {
    tx.begin();
    Product product = new Product(“iPad", “Apple tablet", 649.99);
    pm.makePersistent(product);
    tx.commit();
} finally {
    if (tx.isActive()) tx.rollback();
    pm.close();
}
```

Herramientas de O/R Mapping

DataNucleus (JDO & JPA)

5. Uso desde una aplicación: READ

```
Transaction tx=pm.currentTransaction();
try {
    tx.begin();
    Extent e = pm.getExtent(Product.class, true);
    Query q = pm.newQuery(e,"price < 1500.00");
    q.setOrdering("price ascending");
    Collection c = (Collection) q.execute();
    for (Product p: c) { … }
    tx.commit();
} finally {
    if (tx.isActive()) tx.rollback();
    pm.close();
}
```


Herramientas de O/R Mapping

DataNucleus (JDO & JPA)

5. Uso desde una aplicación: DELETE

```
Transaction tx = pm.currentTransaction();
try {
    tx.begin();
    ...
    pm.deletePersistent(product);
    tx.commit();
} finally {
    if (tx.isActive()) tx.rollback();
    pm.close();
}
```

Herramientas de O/R Mapping

Hibernate

HIBERNATE

1. Clase en Java [POJO: Plain Old Java Object]

```
public class Employee
{
  private int id;
  private String firstName;
  private String lastName;
  private int level;
  …
  // Métodos get & set
  …
}
```


Herramientas de O/R Mapping

Hibernate

HIBERNATE

2. Tabla en la base de datos relacional (p.ej. MySQL)

```
create table EMPLOYEE (
   id INT NOT NULL auto_increment,
   first_name VARCHAR(20) default NULL,
   last_name  VARCHAR(20) default NULL,
   level INT default NULL,
   PRIMARY KEY (id)
);
```

Herramientas de O/R Mapping

Hibernate

3. Fichero de configuración (Employee.hbm.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD//EN"
 "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
   <class name="Employee" table="EMPLOYEE">
      <meta attribute="class-description">…</meta>
      <id name="id" column="id" type="int">
         <generator class="native"/>
      </id>
      <property name="firstName" column="first_name" type="string"/>
      <property name="lastName" column="last_name" type="string"/>
      <property name="level" column="level" type="int"/>
   </class>
</hibernate-mapping>
```


Herramientas de O/R Mapping

Hibernate: CREATE

```
public int addEmployee (String fname, String lname, int level) {
      Session session = sessionFactory.openSession();
      Transaction tx = null;
      Integer employeeID = null;
      try{
         tx = session.beginTransaction();
         Employee employee = new Employee(fname, lname, level);
         employeeID = (Integer) session.save(employee);
         tx.commit();
      } catch (HibernateException e) {
         if (tx!=null) tx.rollback();
      } finally {
         session.close();
      }
      return employeeID;
}
```

Herramientas de O/R Mapping

Hibernate: READ

HIBERNATE

```
public List listEmployees () {
  List employees;
  Session session = factory.openSession();
  Transaction tx = null;
  try {
    tx = session.beginTransaction();
    employees = session.createQuery("FROM Employee").list();
    tx.commit();
  } catch (HibernateException e) {
    if (tx!=null) tx.rollback();
  } finally {
    session.close();
  }
  return employees:
}
//  for (Employee employee: employees) …
```


Herramientas de O/R Mapping

Hibernate: UPDATE

HIBERNATE

```
public void updateEmployee (int EmployeeID, int level) {
  Session session = factory.openSession();
  Transaction tx = null;
  try {
    tx = session.beginTransaction();
    Employee employee = (Employee)session.get(Employee.class, EmployeeID);
    employee.setLevel ( level );
    session.update(employee);
    tx.commit();
  } catch (HibernateException e) {
    if (tx!=null) tx.rollback();
  } finally {
    session.close();
  }
}
```

Herramientas de O/R Mapping

Hibernate: DELETE


```
public void deleteEmployee (int EmployeeID) {
  Session session = factory.openSession();
  Transaction tx = null;
  try {
    tx = session.beginTransaction();
    Employee employee = (Employee)session.get(Employee.class, EmployeeID);
    session.delete(employee);
    tx.commit();
  } catch (HibernateException e) {
    if (tx!=null) tx.rollback();
  } finally {
    session.close();
  }
}
```


Herramientas de O/R Mapping

iBATIS → MyBatis

MyBatis

Acopla objetos en Java
con sentencias SQL
o llamadas a procedimientos almacenados

Opción A: Usando anotaciones

```
public interface BlogMapper {
    @Select("select * from Blog where id = #{id}")
    Blog selectBlog(int id);
}
```

```
BlogMapper mapper = session.getMapper(BlogMapper.class);
Blog blog = mapper.selectBlog(101);
```

Herramientas de O/R Mapping

iBATIS → MyBatis

MyBatis

Acopla objetos en Java
con sentencias SQL
o llamadas a procedimientos almacenados

Opción B: Usando ficheros XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="BlogMapper">
    <select id="selectBlog" parameterType="int" resultType="Blog">
        select * from Blog where id = #{id}
    </select>
</mapper>
```

```
Blog blog = session.selectOne("BlogMapper.selectBlog", 101);
```


Herramientas de O/R Mapping

JPA [Java Persistence API]

Clases anotadas

```
@Entity
public class Book {
    @Id
    private Integer id;
    private String title;
    private String isbn;

    @ManyToOne
    private Publisher publisher;

    @ManyToMany
    private List<Author> authors;
}
```

```
@Entity
public class Publisher {
    @Id
    private Integer id;
    private String name;
    private String address;
    @OneToMany(mappedBy = "publisher")
    private List<Book> books;
}
```

```
@Entity
public class Author {
    @Id
    private Integer id;
    private String firstName;
    private String lastName;
    @ManyToMany
    private List<Book> books;
}
```

Herramientas de O/R Mapping

JPA [Java Persistence API]

Lenguaje de consulta JPQL

```
import javax.persistence.EntityManager;
import javax.persistence.Query;
…
public List<Author> getAuthorsByLastName(String lastName)
{
  String queryString = "SELECT a FROM Author a" +
                       " WHERE a.lastName IS NULL" +
                       " OR LOWER(a.lastName)=LOWER(:lastName)";
  Query query = getEntityManager().createQuery(queryString);
  query.setParameter("lastName", lastName);
  return query.getResultList();
}
```


Herramientas de O/R Mapping

- Las herramientas de O/R mapping **no** garantizan la inmunidad frente a ataques por inyección de código.
- Ya sea a través del API proporcionado por la herramienta o mediante el lenguaje de consulta soportado por la herramienta (p.ej. HQL en Hibernate), un atacante puede conseguir manipular la base de datos relacional sobre la que trabaja la herramienta de O/R mapping.

Seguridad en Hibernate: Consulta con parámetros

```
Session s = factory.openSession();
Transaction tx;
try {
    tx = s.beginTransaction();
    Query q = s.createQuery("from Clientes c where c.name=?");
    q.setString(0, "López"); // Parámetro posicional
    tx.commit();
} catch (Exception e) {
    if (tx!=null) tx.rollback();
    throw e;
} finally {
    s.close();
}
```


Herramientas de O/R Mapping

Seguridad en Hibernate: Consulta con parámetros

```
Session s = factory.openSession();
Transaction tx;
try {
  tx = s.beginTransaction();
  Query q = s.createQuery("from Clientes c where c.name=:name");
  q.setString("name", "López"); // Parámetro con nombre (mejor)
  tx.commit();
} catch (Exception e) {
    if (tx!=null) tx.rollback();
    throw e;
} finally {
    s.close();
}
```

Herramientas de O/R Mapping

Seguridad en Hibernate

- Ninguna comunicación con la BD debería producirse fuera de una transacción (problemas de sincronización).

  NOTA: Es importante no olvidar el rollback() de la transacción y descartar la sesión en caso de error.

- createQuery() es vulnerable frente a ataques por inyección de código SQL/HQL, por lo que siempre se deben utilizar consultas con parámetros (?) y los métodos setXXX correspondientes.

Herramientas de O/R Mapping

Seguridad en Hibernate

- En el caso de las aplicaciones web, también hay que tener cuidado con los datos que se llegan a almacenar, no sólo con la forma de ejecutar consultas.
  p.ej. XSS [Cross-Site Scripting]

```
String firstname = request.getParameter("firstname");
String lastname  = request.getParameter("lastname");
…
Person person = session.load(Person.class, new Long(69));
person.setFirstname(firstname);
person.setLastname(lastname);
session.save(thePerson);
session.getTransaction().commit();
```

Seguridad en NoSQL

- NoSQL
  - Key-value stores
  - Wide column stores
  - Document stores
  - Graph-database systems
- Teorema CAP: Consistencia eventual
- Problemas de seguridad: inyección de código NoSQL

NoSQL

SQL = DBMS relacional (solución tradicional)
NoSQL = "**Not only SQL**" = DBMS no relacional

Motivación

Existen aplicaciones para las que las bases de datos relacionales no son la mejor solución…

… o para las cuales no todo se resuelve mejor usando exclusivamente una base de datos relacional.

NoSQL

Lo que ofrece un DBMS:

Característica	**DBMS relacional**
Conveniencia	Modelo de datos simple Lenguaje de consulta declarativo
Multi-usuario	Transacciones
Seguridad	Control de acceso a los datos
Fiabilidad	Replicación
Persistencia	Almacenamiento en ficheros
Volumen de datos ++	
Eficiencia (según para qué) +++	

NoSQL

Sistemas NoSQL

Alternativas a los DBMS relacionales

Pros:

- Flexibilidad a la hora de definir esquemas.
- Más sencillos de configurar.
- Más baratos.
- Escalabilidad.
- Consistencia relajada ➔ Mayor eficiencia/disponibilidad.

Contras:

- Sin lenguaje de consulta declarativo ➔ Más programación.
- Consistencia relajada ➔ Menores garantías.

NoSQL

Ejemplos de uso

Análisis de weblogs

- Registros (IP, timestamp, URL, …)
- Consultas altamente paralelizables.

Análisis de redes sociales

- Datos en forma de red (grafo con atributos).
- Consultas complejas (no adecuadas para SQL).

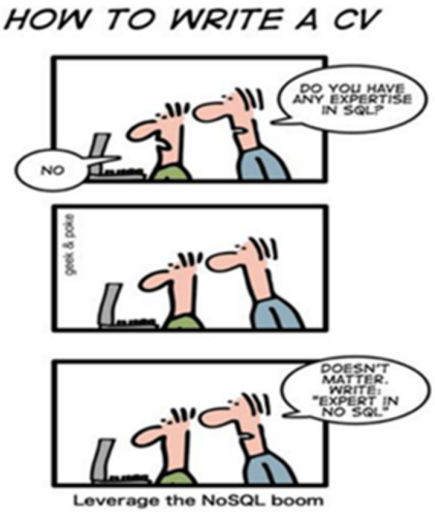
Wikipedia y otras colecciones de documentos

- Combinación de datos estructurados y no estructurados.
- Consultas y operaciones flexibles.

NoSQL

Alternativas de implementación

- Key-value stores
- Wide column stores
- Document stores
- Graph database systems



Key-value stores

OLTP [OnLine Transaction Processing]

DMBS con la interfaz más simple posible:

- Modelo de datos:
  pares <clave, valor>

- Operaciones: CRUD
  insert (key, value)
  fetch (key)
  update (key, value)
  delete (key)

Key-value stores

Implementación

Eficiente, escalable y tolerante a fallos

- Tabla hash distribuida: Registros almacenados en distintos nodos en función de su clave.

- Replicación de datos.

- Transacciones de un único registro:
  "consistencia eventual" (tras una actualización, eventualmente todos los accesos a la misma clave obtendrán el valor actualizado).

Key-value stores

Consistencia eventual: BASE vs. ACID

BASE [Basically Available, Soft state, Eventual consistency]

- Única garantía: "liveness".
- Incrementa la complejidad de los sistemas distribuidos.

ACID [Atomicity, Consistency, Isolation, Durability]

- Garantías tradicionales: "safety".
- DBMS relacional (commit/rollback).

Key-value stores

Ejemplos

- **Redis** (ANSI C)
  http://redis.io/

- **Memcached**
  http://memcached.org/

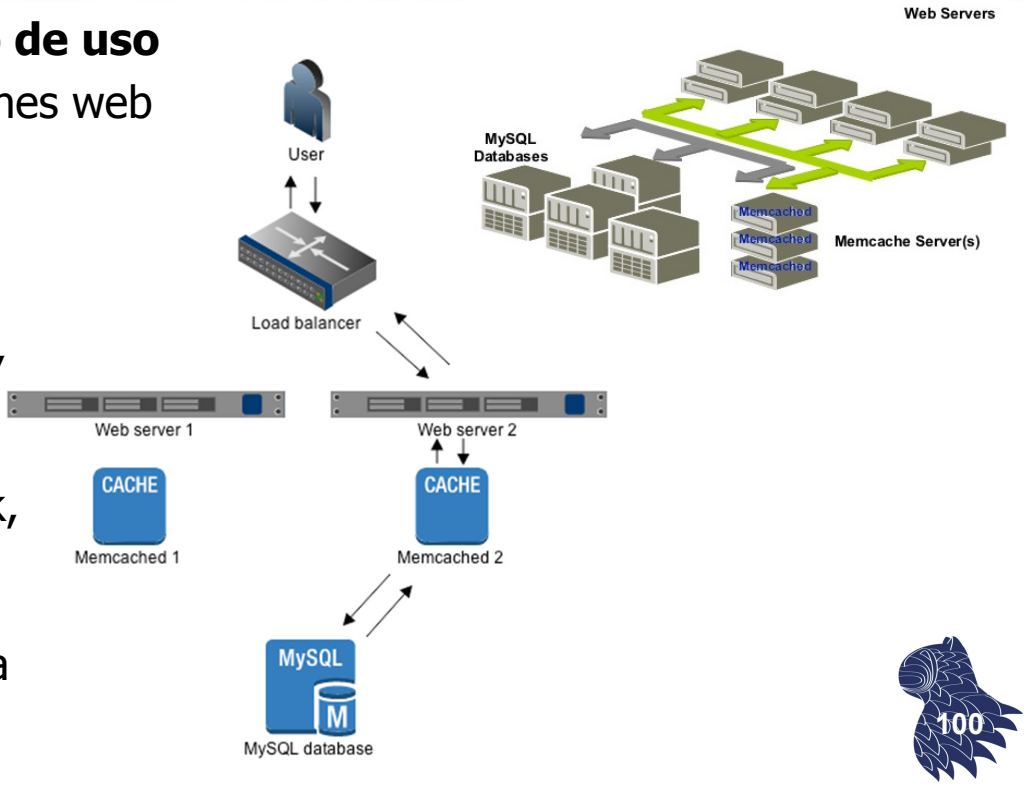
- **Amazon DynamoDB**
  http://aws.amazon.com/dynamodb/

Key-value stores

Ejemplo de uso

Aplicaciones web

p.ej.
YouTube,
Reddit,
Zinga,
Facebook,
Twitter,
Tumblr,
Wikipedia
...



Key-value stores

Algunas implementaciones permiten ordenar las claves, lo que permite realizar consultas sobre rangos de valores y procesar las claves en orden.

Muchos sistemas de este tipo incluyen extensiones que los acercan a otros tipos de sistemas NoSQL:

- Wide column stores
- Document stores

Wide column stores

Como en las bases de datos relacionales,
los datos se almacenan en tablas, con filas y columnas.

A diferencia de las bases de datos relacionales,
los nombres y el formato de las columnas puede variar
de una fila a otra dentro de la misma tabla

Row ID	Columns...		
1	Name	Website	
	Preston	www.example.com	
2	Name	Website	
	Julia	www.example.com	
3	Name	Email	Website
	Alice	example@example.com	www.example.com

Wide column stores

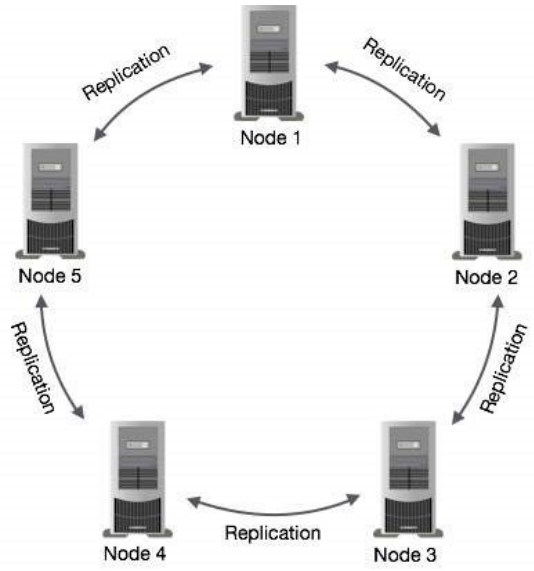
Ejemplos

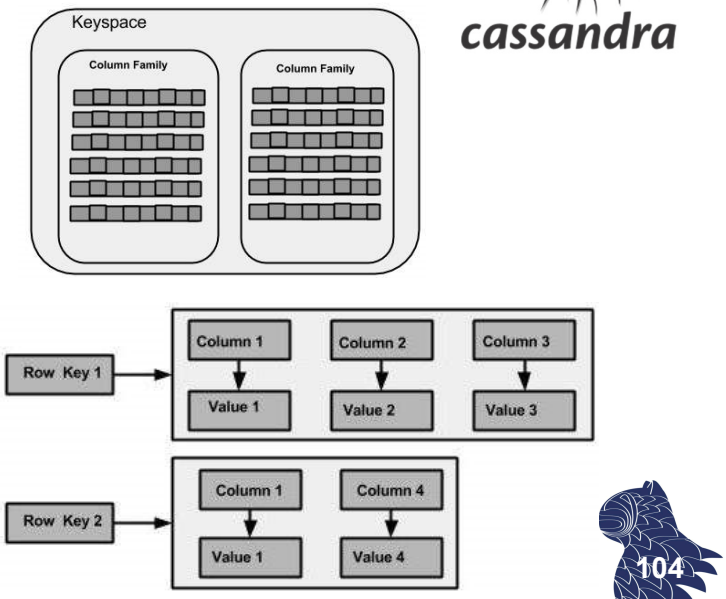
- **Google BigTable** (OSDI'2006)
  https://cloud.google.com/bigtable/
- **Apache Cassandra** (Facebook)
  http://cassandra.apache.org/
- **Apache HBase** (Java, sobre HDFS, como Google BigTable sobre GFS)
  http://hbase.apache.org/
- **Apache Accumulo** (NSA)
  https://accumulo.apache.org/

Wide column stores

Apache Cassandra

Architectura P2P





Wide column stores

Apache Cassandra

CQL [Cassandra Query Language]

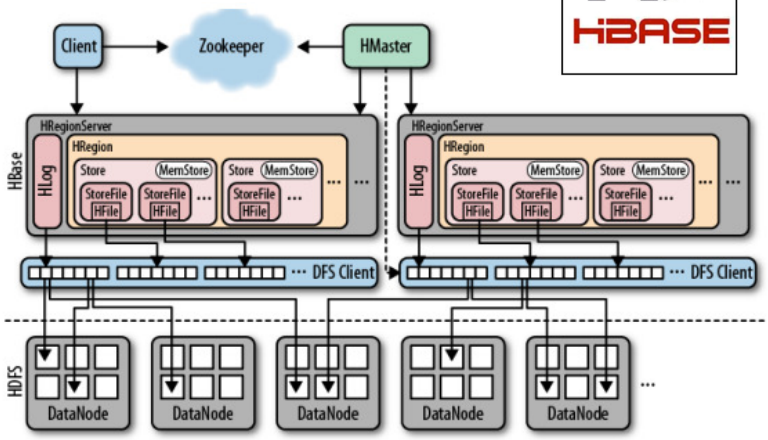
http://www.tutorialspoint.com/cassandra/

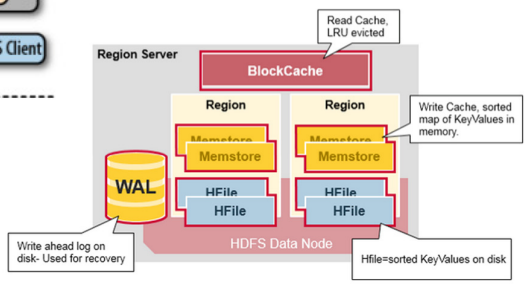
```
CREATE KEYSPACE MyKeySpace
  WITH REPLICATION = { 'class' : 'SimpleStrategy',
                       'replication_factor' : 3 };

USE MyKeySpace;

CREATE COLUMNFAMILY MyColumns
   (id text, Last text, First text, PRIMARY KEY(id));

INSERT INTO MyColumns (id, Last, First)
               VALUES ('1', 'Doe', 'John');

SELECT * FROM MyColumns;
```

Usuarios: Apple (>10PB, 100 000 nodos), Netflix…

Wide column stores

Apache HBase





Información detallada…

https://www.mapr.com/blog/in-depth-look-hbase-architecture

Wide column stores

Apache HBase

HBase shell

http://www.tutorialspoint.com/hbase/

```
create 'emp', 'personal data', 'professional data'
put 'emp','1','personal data:name','Jose'
put 'emp','1','personal data:city','Granada'
put 'emp','1','professional data:designation','manager'
…
scan 'emp'
```

COLUMN FAMILIES

Row key	personal data		professional data	
empid	name	city	designation	salary
1	raju	hyderabad	manager	50,000
2	ravi	chennai	sr.engineer	30,000
3	rajesh	delhi	jr.engineer	25,000

Usuarios: Facebook (desde 2010), LinkedIn, Spotify…

Document stores

a.k.a. document-oriented databases

No existe un esquema de la base de datos:

- Cada registro puede tener una estructura diferente.
- Tipos de las columnas variables de un registro a otro.
- Las columnas pueden tener más de un valor (arrays).
- Los registros pueden tener estructura propia [nested].

Representación de los datos utilizando JSON o XML.

Document stores

Implementación

- Como los almacenes clave-valor, salvo que ahora el valor es un documento semiestructurado (JSON, XML).

- Operaciones básicas de un almacén clave-valor:
  insert (key, value)
  fetch (key)
  update (key, value)
  delete (key)

- Consultas limitadas sobre el contenido de los documentos (dependientes del sistema concreto).

Document stores

Ejemplos más populares

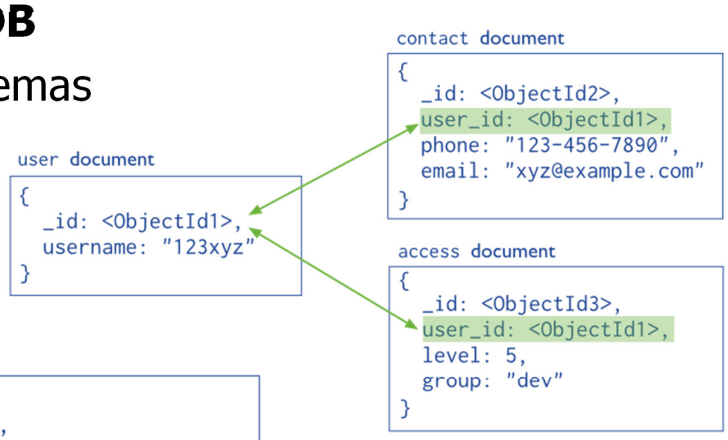
- **MongoDB** (C/C++, Javascript)
  https://www.mongodb.org/
- **Couchbase** (C/C++, Erlang)
  http://www.couchbase.com/
- **CouchDB** (Erlang)
  http://couchdb.apache.org/
- **Google Datastore**
  https://cloud.google.com/datastore/
- **Amazon DynamoDB**
  http://aws.amazon.com/dynamodb/
- **MarkLogic**
  http://www.marklogic.com/

Document stores

MongoDB

Sin esquemas

JSON




```
{
  _id: <ObjectId1>,
  username: "123xyz",
  contact: {
             phone: "123-456-7890",
             email: "xyz@example.com"
           },
  access: {
            level: 5,
            group: "dev"
          }
}
```

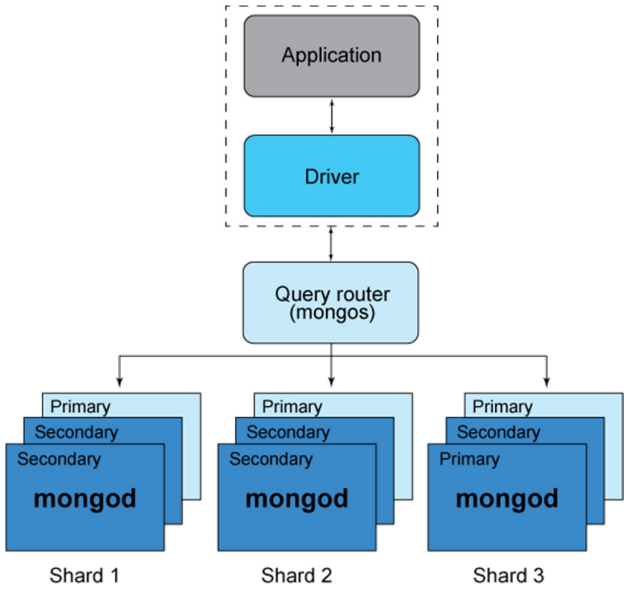
Embedded sub-document

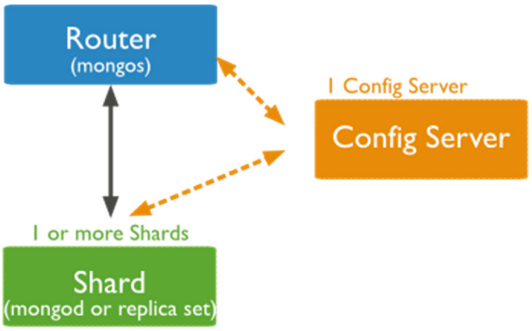
Embedded sub-document

Document stores

MongoDB

Arquitectura

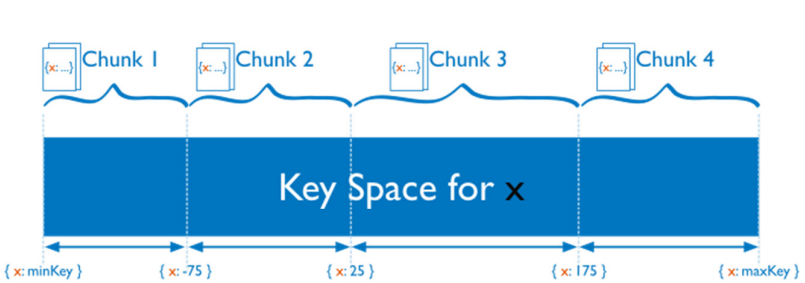


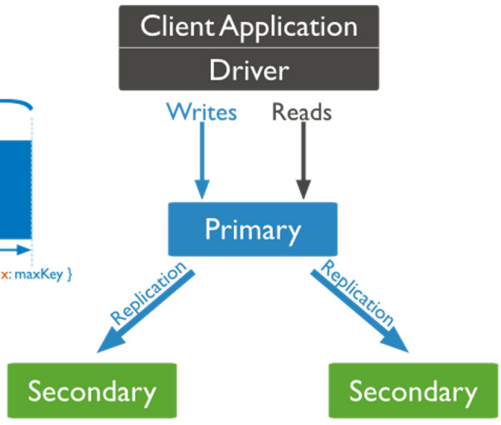


Document stores

MongoDB

Sharding

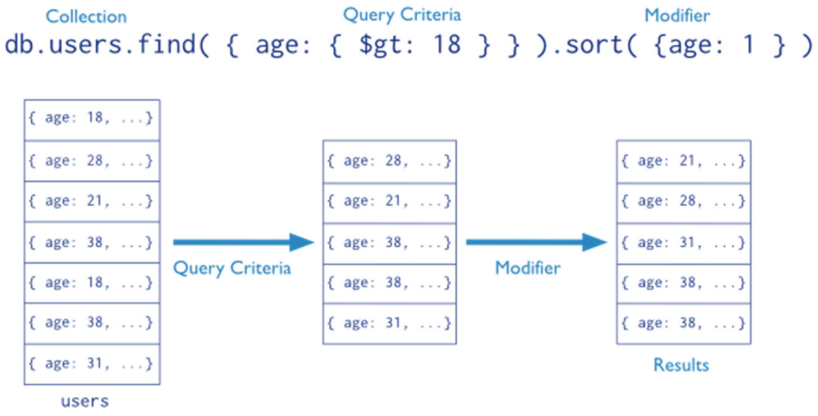




Document stores

MongoDB

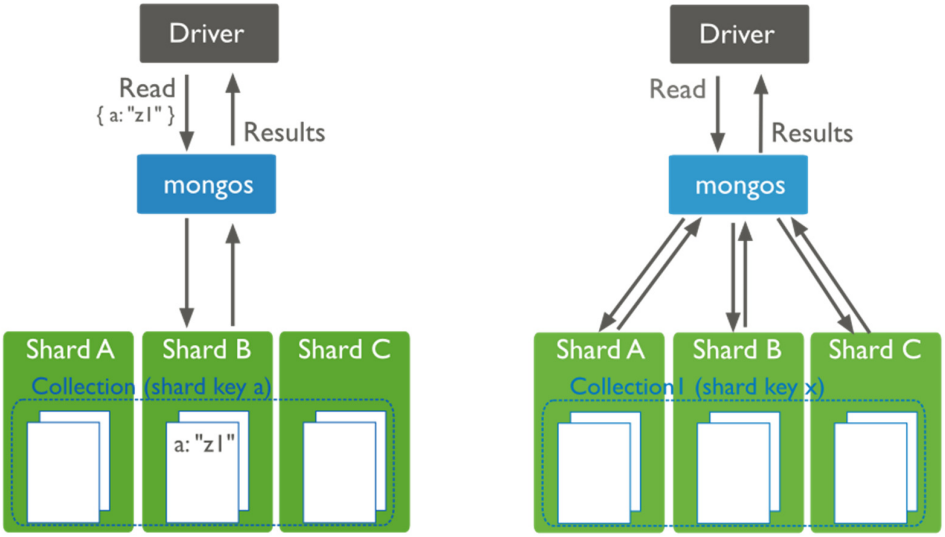
CRUD: Consultas



Document stores

MongoDB

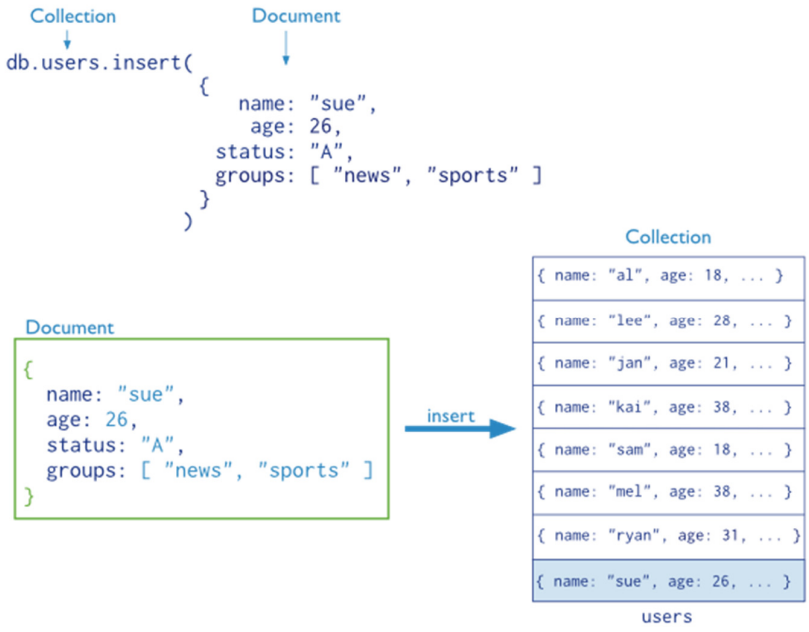
CRUD: Consultas



Document stores

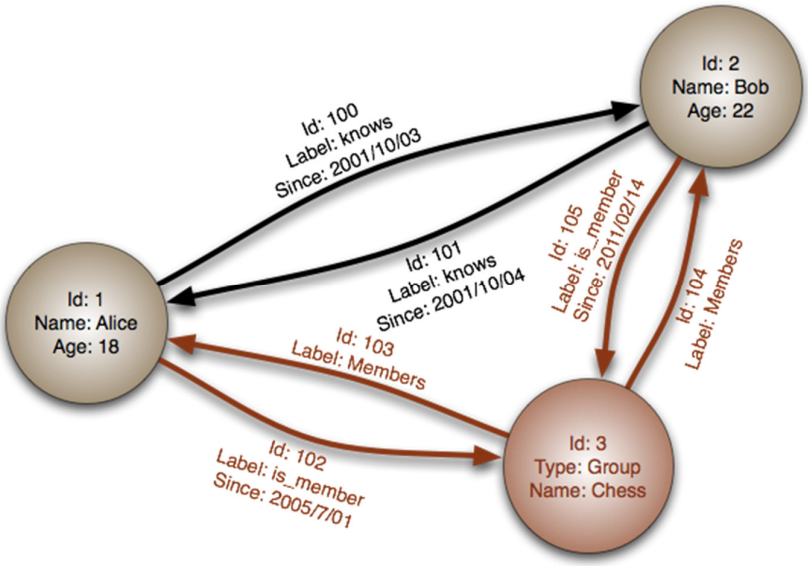
MongoDB

CRUD: Actualizaciones (insert|update|remove)



Graph database systems

Utilizan grafos con atributos para almacenar los datos:



Graph database systems

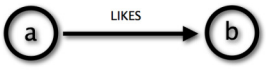
Sistemas

- **Neo4j** (Java)
  http://neo4j.com/
- **OrientDB** (Java, multi-modelo)
  http://orientdb.com/
- **Titan** (Java)
  http://thinkaurelius.github.io/titan/

Graph database systems

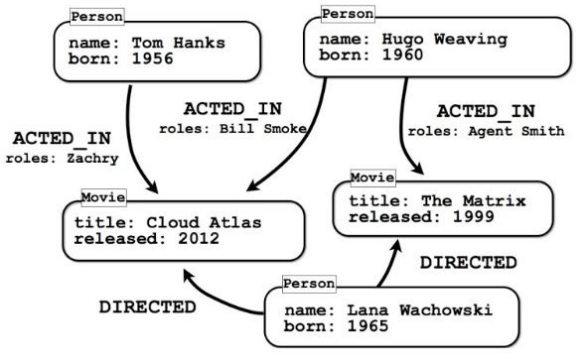
Neo4j

Cypher (query language)
http://neo4j.com/docs/stable/cypher-refcard/



Cypher

```
(a) -[:LIKES]-> (b)
```

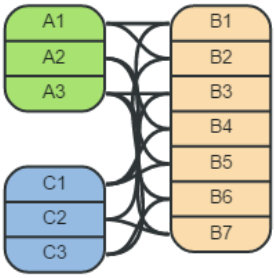



```
MATCH (actor:Person)-[:ACTED_IN]->(movie:Movie)
WHERE movie.title =~ "T.*"
RETURN movie.title as title,
       collect(actor.name) as cast
ORDER BY title ASC LIMIT 10;
```

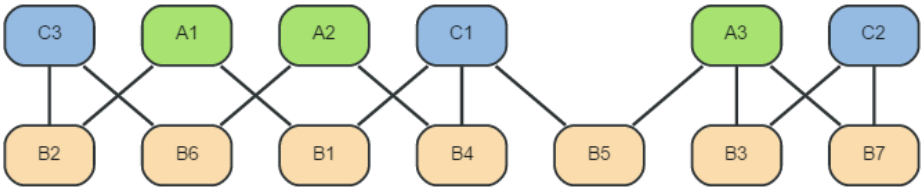
Graph database systems

neo4j

Relational DB



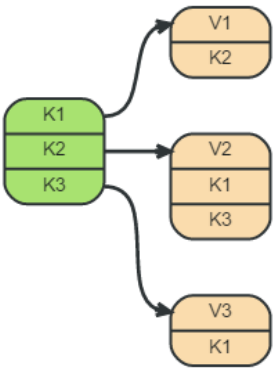
Graph DB



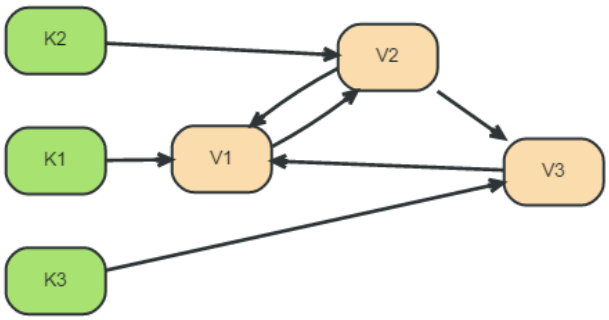
Graph database systems

neo4j

Key-value store



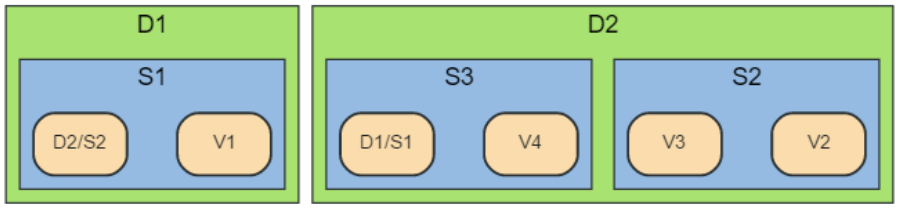
Graph DB



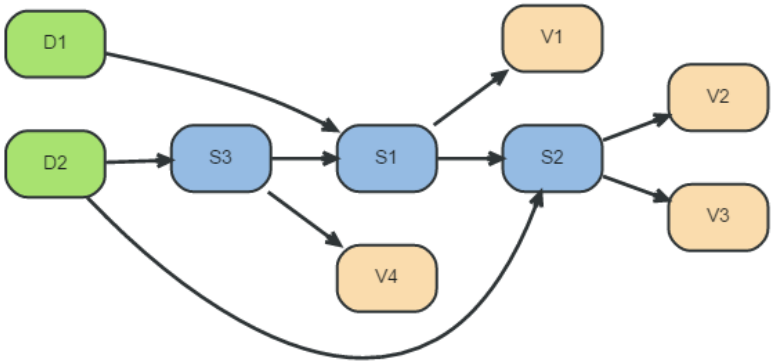
Graph database systems

neo4j

Document store



Graph DB

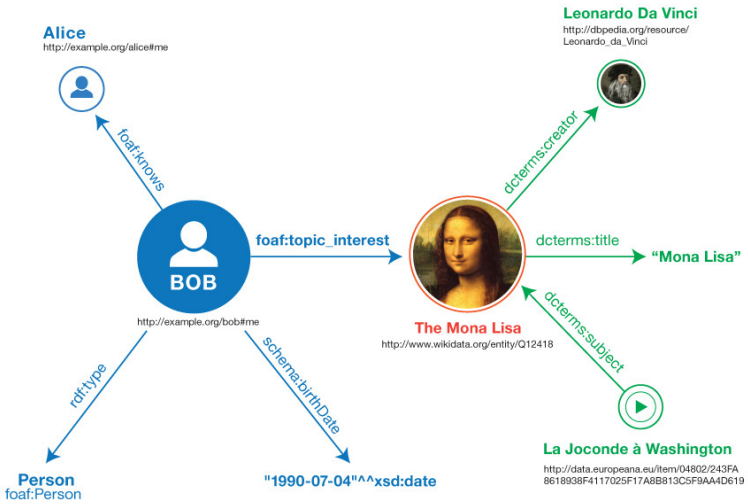


Graph database systems

Ejemplo: Web semántica

Tripletas RDF [Resource Description Framework]

`<subject> <predicate> <object>`



Graph database systems

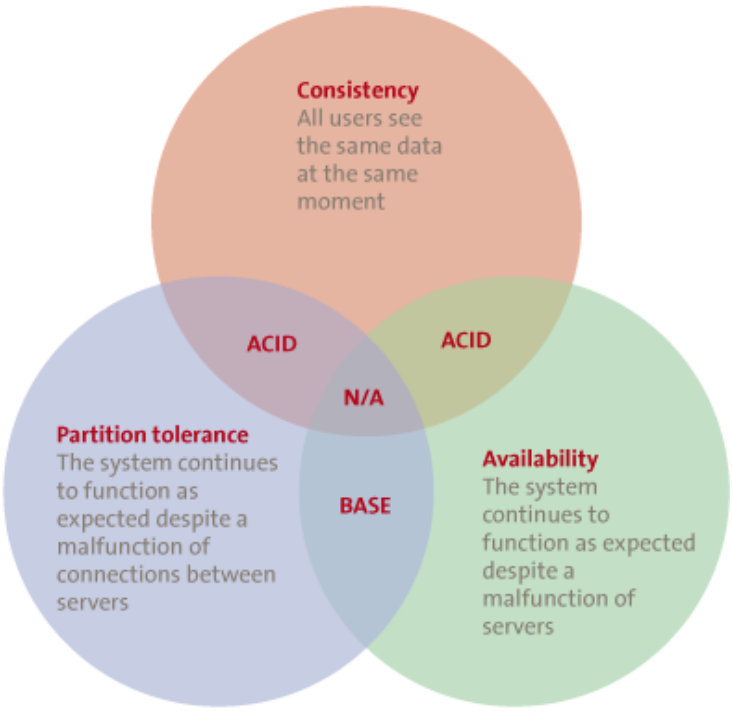
Ejemplo: Web semántica

Lenguaje de consulta SPARQL ["sparkle"]
p.ej. Neo4j, Virtuoso…

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?email
WHERE {
  ?person a foaf:Person.
  ?person foaf:name ?name.
  ?person foaf:mbox ?email.
}
```

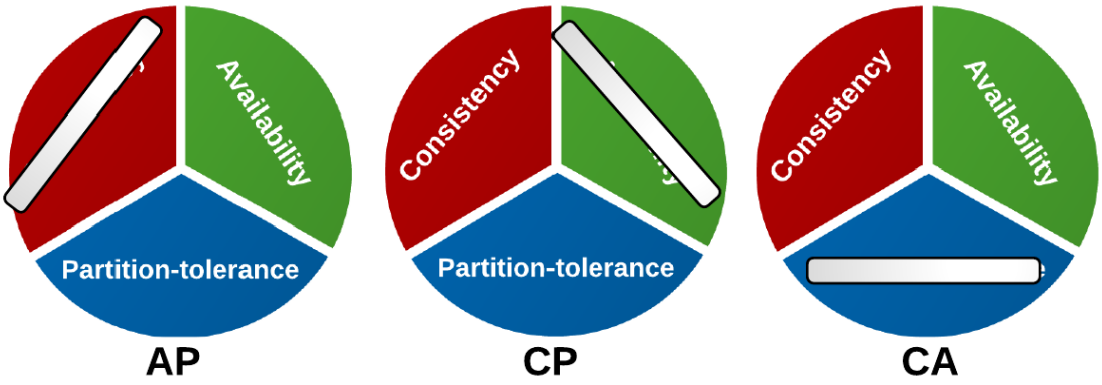
Acrónimo recursivo:
SPARQL Protocol and RDF Query Language

Bases de datos NoSQL



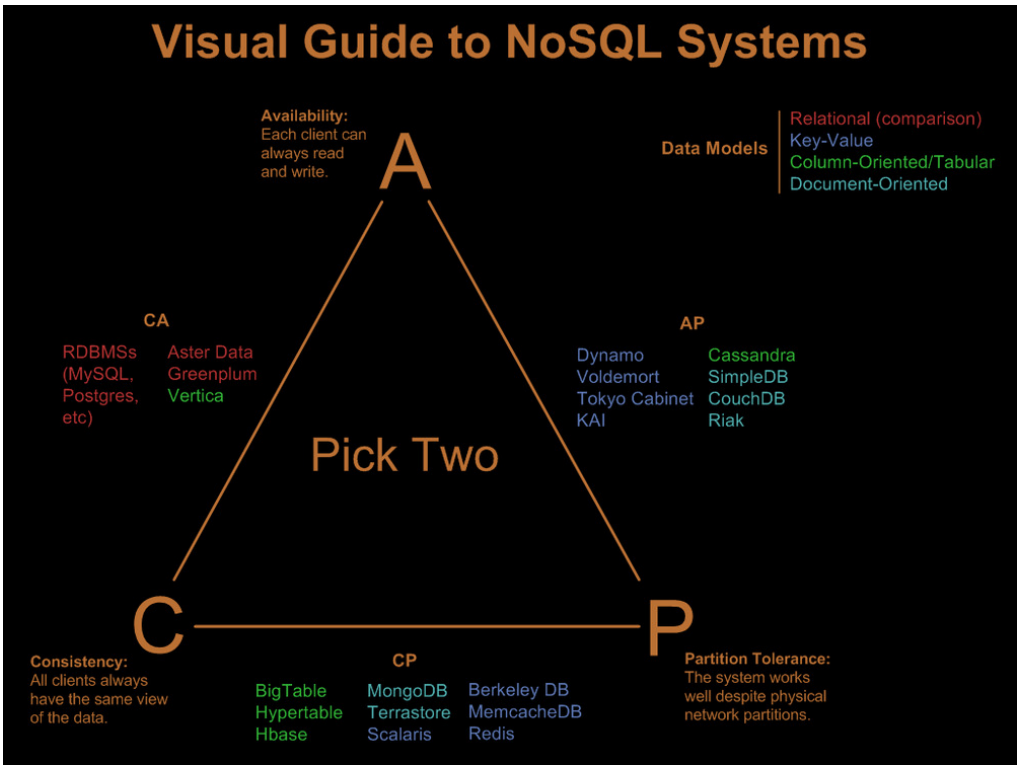
Bases de datos NoSQL

Teorema CAP



http://captheorem-jweaver.rhcloud.com/

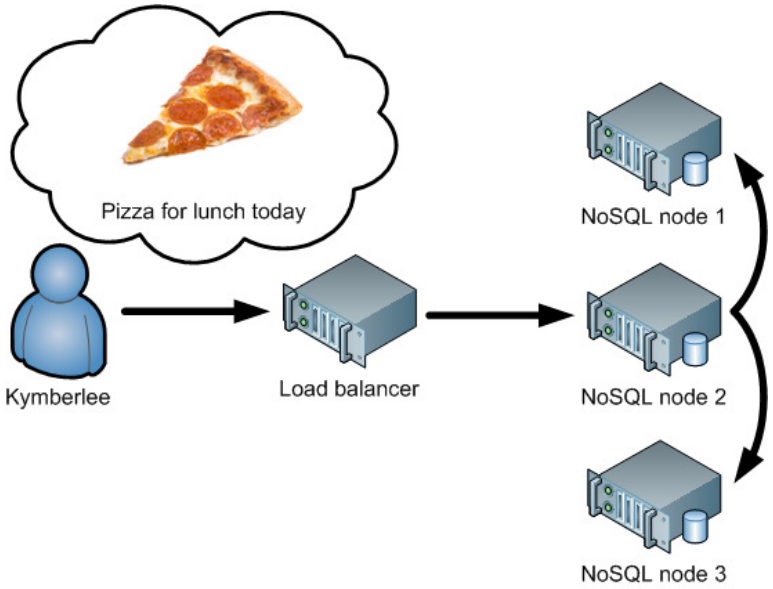
Bases de datos NoSQL



Bases de datos NoSQL

Consistencia eventual:

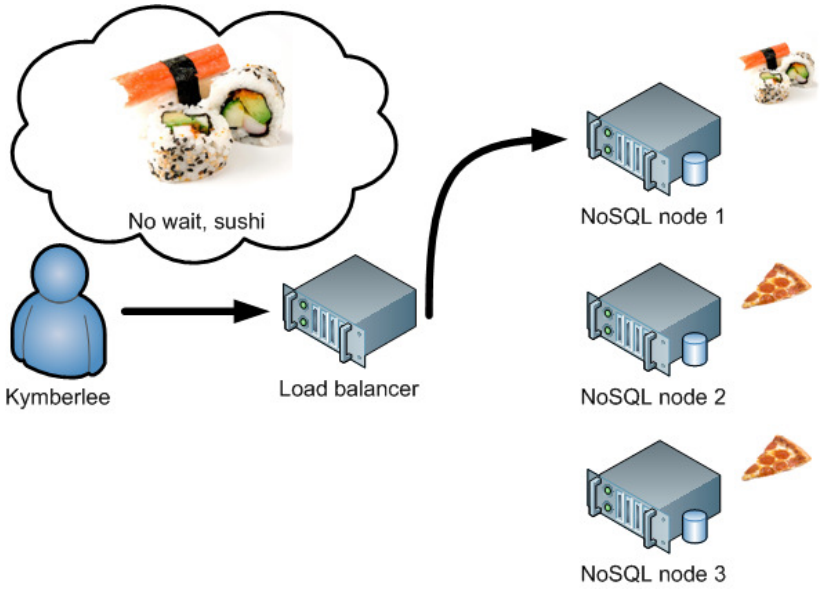
Las actualizaciones no se propagan inmediatamente



Bases de datos NoSQL

Consistencia eventual:

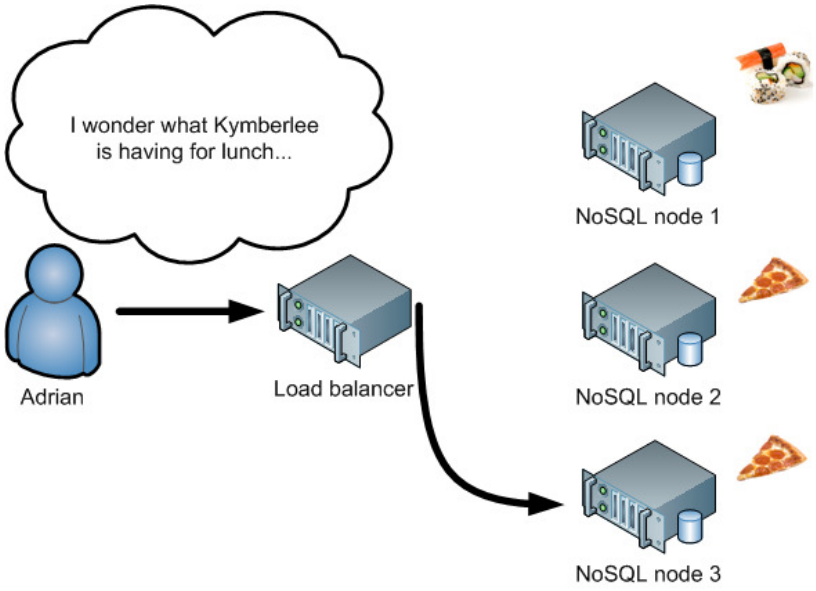
Las actualizaciones no se propagan inmediatamente



Bases de datos NoSQL

Consistencia eventual:

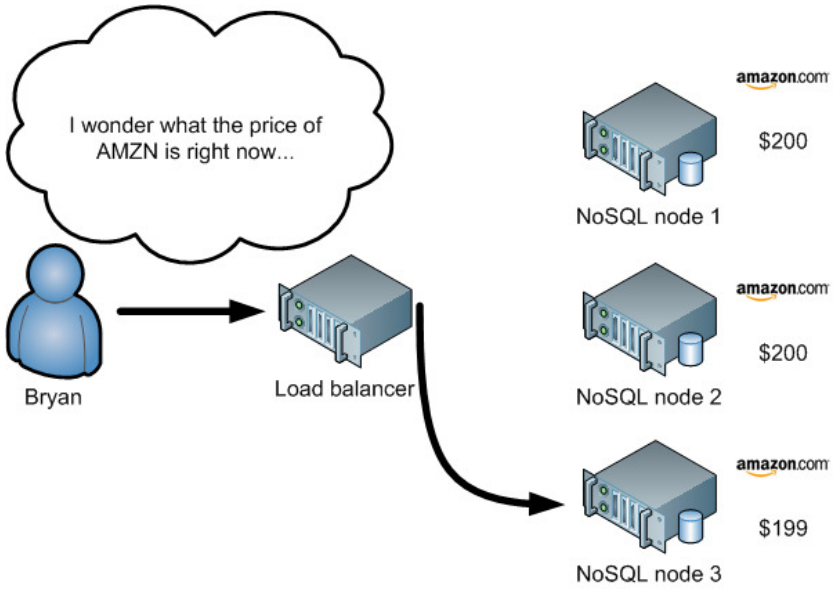
Las actualizaciones no se propagan inmediatamente



Bases de datos NoSQL

Consistencia eventual: Stale data

Las actualizaciones no se propagan inmediatamente



Bases de datos NoSQL

MongoDB

- C++
- Distribución: distintos nodos con réplicas de los datos.
- **Consistencia estricta**: uno de los nodos ejerce de nodo primario (todas las operaciones de escritura), los demás son nodos secundarios.

CouchDB

- Erlang
- Distribución: distintos nodos con réplicas de los datos.
- **Consistencia eventual**: se permiten operaciones de escritura sin esperar la confirmación de los demás nodos (copias incrementales de los cambios).

Bases de datos NoSQL

Compromiso consistencia-disponibilidad

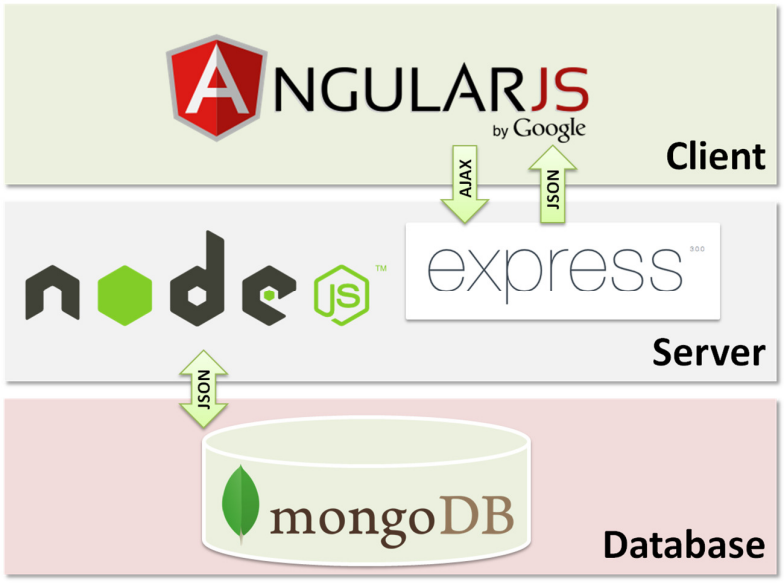
Más importante garantizar la consistencia de los datos…

Más importante la disponibilidad de los datos…

Arquitecturas típicas

MEAN stack

http://mean.io/#!/



mongoDB

express

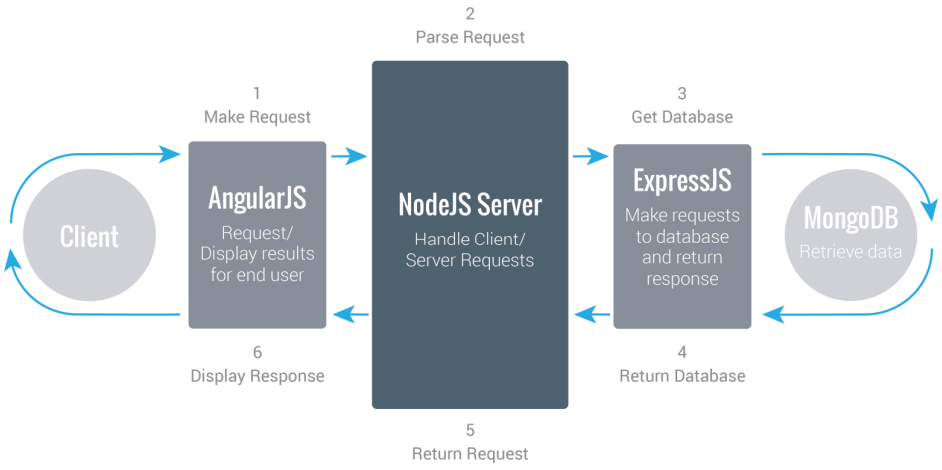
ANGULARJS by Google

node JS

Arquitecturas típicas

MEAN stack

http://mean.io/#!/



mongoDB

express

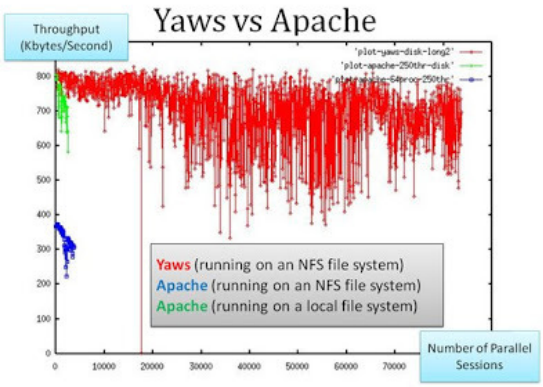
ANGULARJS by Google

node JS

Arquitecturas típicas

LYME/LYCE stack

- Linux
- Yaws (web server)
- Mnesia/CouchDB (database)
- Erlang (programming language)



Problemas de seguridad

Problemas habituales en muchos sistemas NoSQL:

- Autentificación no habilitada por defecto.
- Comunicación usando texto plano (MongoDB).
- Datos no encriptados [“at rest”].
- Vulnerabilidad frente a ataques por inyección.
- Vulnerabilidad frente a ataques por denegación.

Ataques por inyección NoSQL

- Incluso más graves que en bases de datos SQL, al utilizar en ocasiones lenguajes imperativos en vez de declarativos.

- A diferencia de los ataques sobre base de datos relacionales, donde la inyección de código se ejecuta en el DBMS, la inyección de código NoSQL se ejecuta allí donde se analiza y evalúa la cadena (en la capa de aplicación o en la base de datos, dependiendo del sistema).

Ataques por inyección NoSQL

- Incluso más graves que en bases de datos SQL, al utilizar en ocasiones lenguajes imperativos en vez de declarativos.

- Llamadas al API usando formatos estandarizados (XML, JSON, LINQ…), vulnerables al uso malicioso de caracteres especiales.

  p.ej XML `< > & ;`
  JSON `/ { } : .`

- Centenares de productos NoSQL, cada uno con sus peculiarides específicas (API, lenguaje de consulta, modelo de datos..)

Ataques por inyección NoSQL

MongoDB

Su API espera BSON [Binary JSON],
que admite expresiones en JavaScript en sus parámetros

- Filtro, estilo SQL:

```
db.myCollection.find(
  { $where: "this.credits == this.debits" } );
```

- Ejecución de código JavaScript:

```
db.myCollection.find(
  { $where: function()
            { return obj.credits - obj.debits < 0; } } );
```


Ataques por inyección NoSQL

MongoDB

Si el atacante puede manipular los datos pasados al operador $where, puede ejecutar código en JavaScript:

```
db.myCollection.find( { active: true, $where: function() {
  return obj.credits - obj.debits < $userInput; } } );;
```

Inyectando caracteres especiales para el lenguaje del API y observando los resultados, el atacante puede determinar hasta qué punto se controla correctamente la entrada, p.ej. MongoDB produciría un error en cuanto le llegasen determinados caracteres especiales `'" \;()`

Ataques por inyección NoSQL

MongoDB

Si el atacante puede manipular los datos pasados al operador $where, puede ejecutar código en JavaScript:

```
db.myCollection.find( { active: true, $where: function() {
  return obj.credits - obj.debits < $userInput; } } );;
```

Con la entrada `0;var date=new Date(); do{curDate = new Date();}while(curDate-date<10000)` el atacante mantendría ocupada la CPU al 100% durante 10 segundos:

```
function() { return obj.credits - obj.debits < 0;
  var date=new Date();
  do{curDate = new Date();}while(curDate-date<10000); }
```

Ataques por inyección NoSQL

MongoDB

Incluso aunque las consultas se parametricen y se controle perfectamente la entrada, existe una vía alternativa para inyectar código.

En MongoDB, $where es un operador, pero también es una variable válida en PHP, por lo que un atacante puede inyectar código si crea una variable $where en PHP (e.g. HTTP Parameter Pollution).

Cualquier modificación de $where podría generar un error en MongoDB que le permitiría al atacante detectar la vulnerabilidad para luego explotarla :-(

Ataques por inyección NoSQL

MongoDB: Payloads

```
true, $where: '1 == 1'
, $where: '1 == 1'
$where: '1 == 1'
', $where: '1 == 1'
1, $where: '1 == 1'
{ $ne: 1 }
', $or: [ {}, { 'a':'a
' } ], $comment:'successful MongoDB injection'
db.injection.insert({success:1});
db.injection.insert({success:1});return
1;db.stores.mapReduce(function() { { emit(1,1
|| 1==1
' && this.password.match(/.*/)//+%00
' && this.passwordzz.match(/.*/)//+%00
'%20%26%26%20this.password.match(/.*/)//+%00
'%20%26%26%20this.passwordzz.match(/.*/)//+%00
{$gt: ''}
[$ne]=1
```


Otros problemas de seguridad

Control de acceso

MongoDB documentation

"One valid way to run the Mongo database is in a trusted environment, with no security and authentication."

This "is the default option and is recommended"

Cassandra Wiki

"The default AllowAllAuthenticator approach is essentially pass-through"

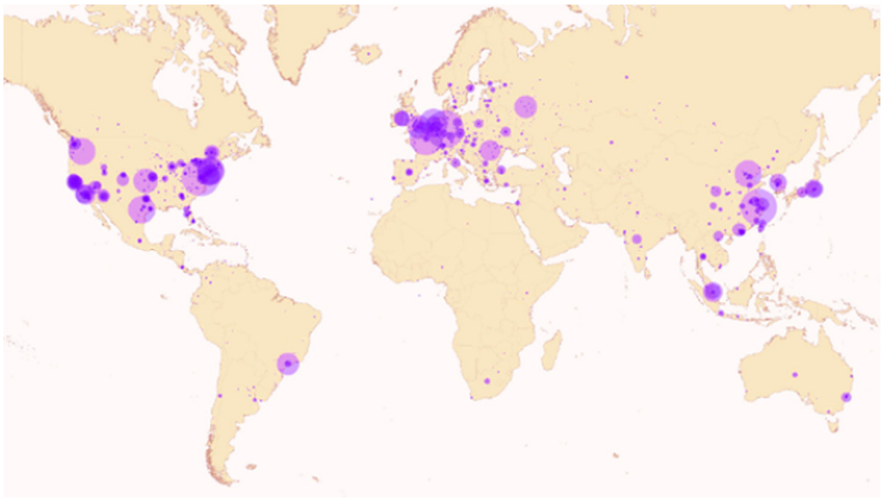
CouchDB: The Definitive Guide

The "Admin Party": Everyone can do everything by default.

Otros problemas de seguridad

Bases de datos MongoDB sin seguridad alguna en Internet

http://www.information-age.com/major-security-alert-40000-mongodb-databases-left-unsecured-internet-123459001/



Puerto TCP 27017

¡¡¡Incluyendo empresas con millones de clientes!!!

Otros problemas de seguridad

Al atacante le basta con encontrar un puerto abierto…

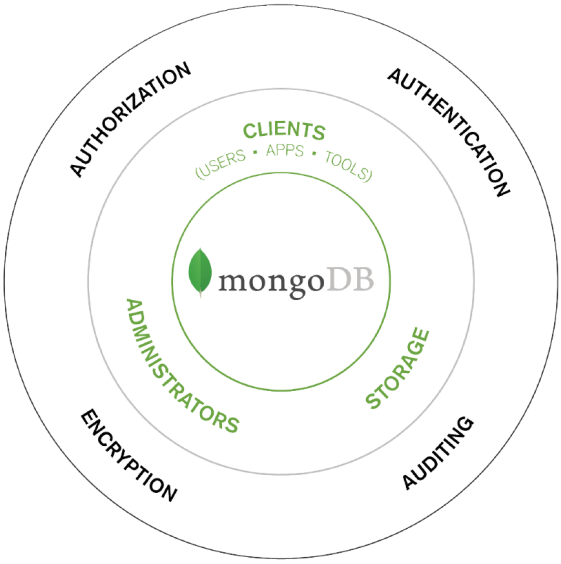
Base de datos	Puerto por defecto
MongoDB	27017 / 28017 / 27080
CouchDB	5984
Hbase	9000
Cassandra	9160
Neo4j	7474

Otros problemas de seguridad

Bases de datos MongoDB sin seguridad alguna en Internet

MongoDB Security Architecture

https://www.mongodb.com/collateral/mongodb-security-architecture

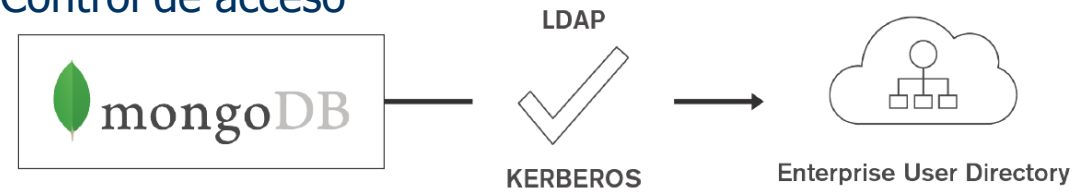


Otros problemas de seguridad

Bases de datos MongoDB sin seguridad alguna en Internet

MongoDB Security Architecture

Control de acceso



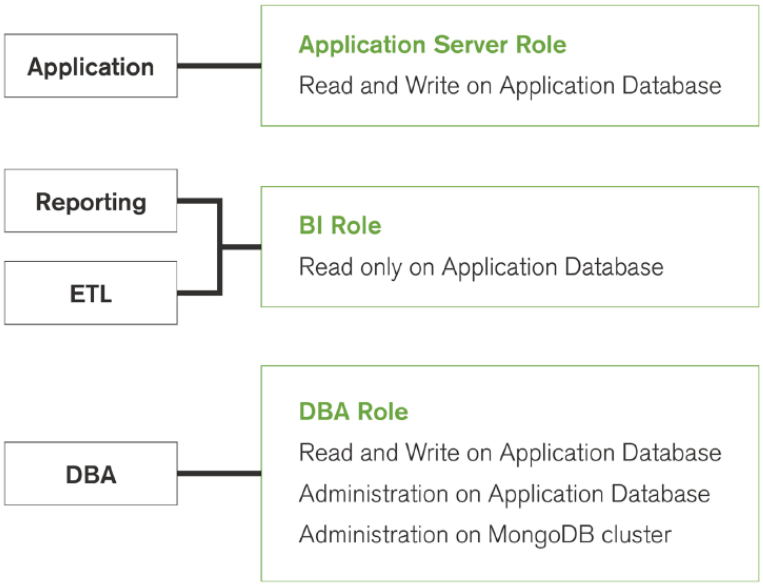
Auditoría

Otros problemas de seguridad

Bases de datos MongoDB sin seguridad alguna en Internet

MongoDB Security Architecture

Roles

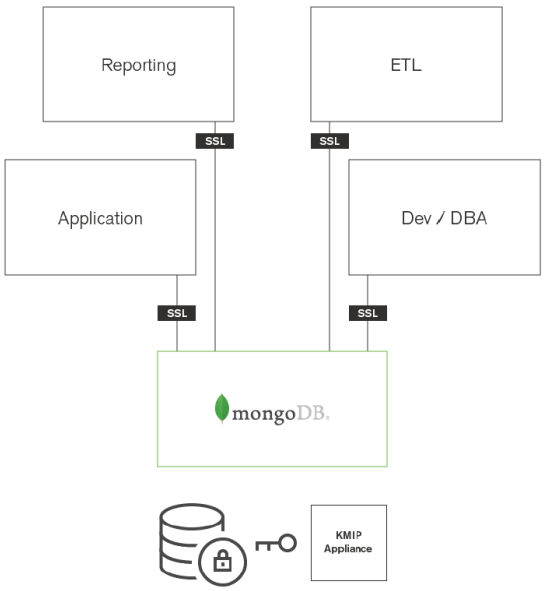


Otros problemas de seguridad

Bases de datos MongoDB sin seguridad alguna en Internet

MongoDB Security Architecture

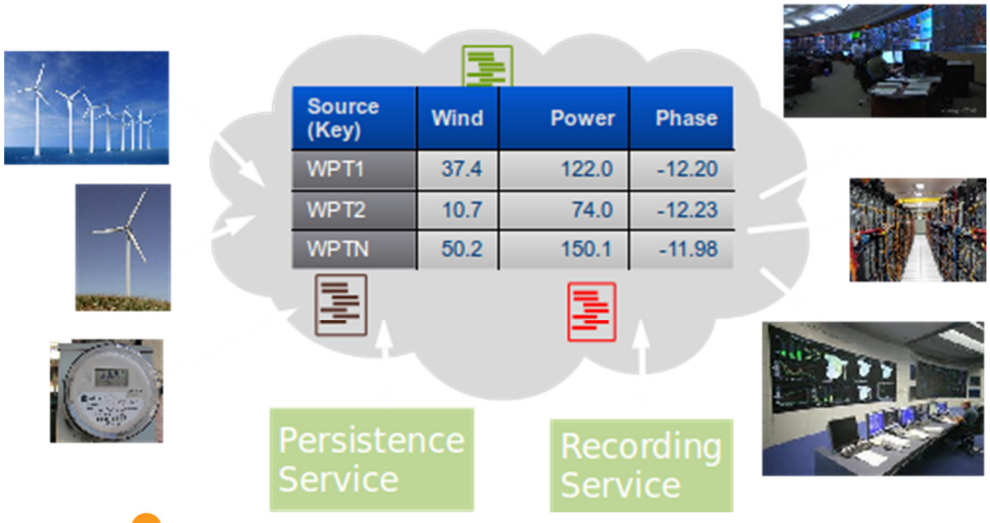
Criptografía



Seguridad en el middleware

DDS [Data Distribution Service]

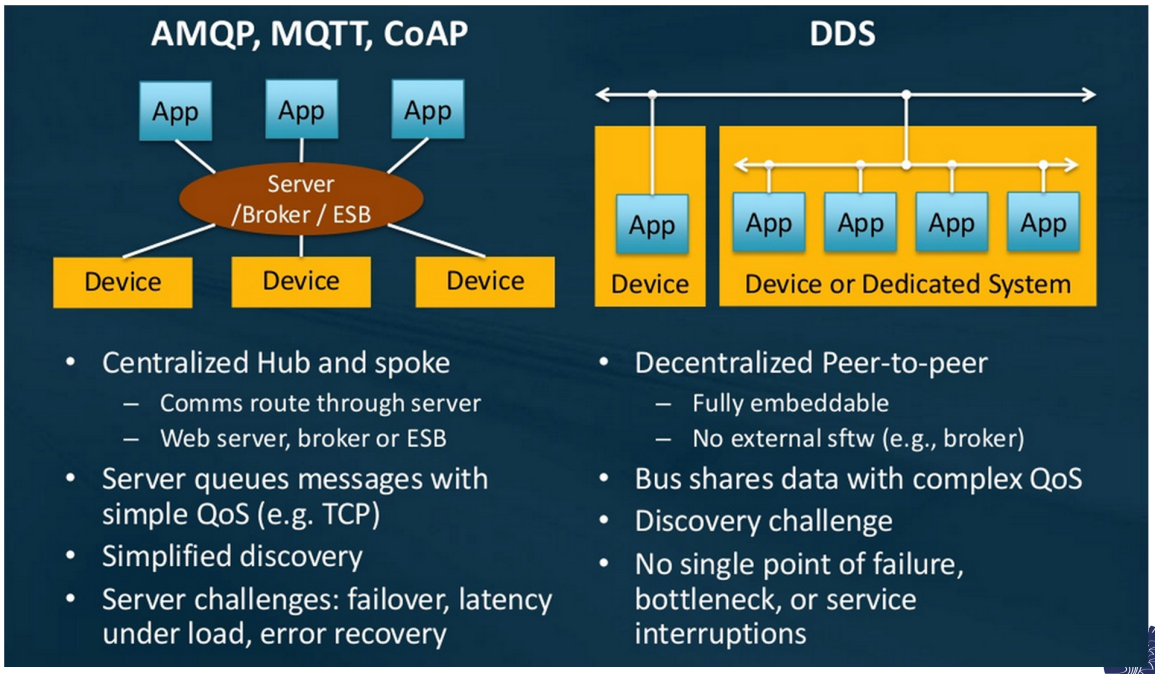
DATA-CENTRIC MIDDLEWARE



rti

OMG
OBJECT MANAGEMENT GROUP

DDS [Data Distribution Service]

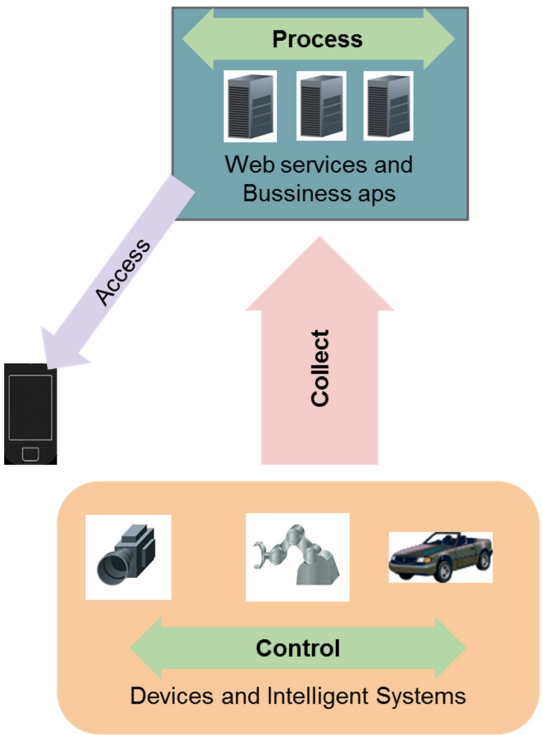


- Centralized Hub and spoke
  - Comms route through server
  - Web server, broker or ESB
- Server queues messages with simple QoS (e.g. TCP)
- Simplified discovery
- Server challenges: failover, latency under load, error recovery

- Decentralized Peer-to-peer
  - Fully embeddable
  - No external sftw (e.g., broker)
- Bus shares data with complex QoS
- Discovery challenge
- No single point of failure, bottleneck, or service interruptions

https://en.wikipedia.org/wiki/Data_Distribution_Service

DDS [Data Distribution Service]



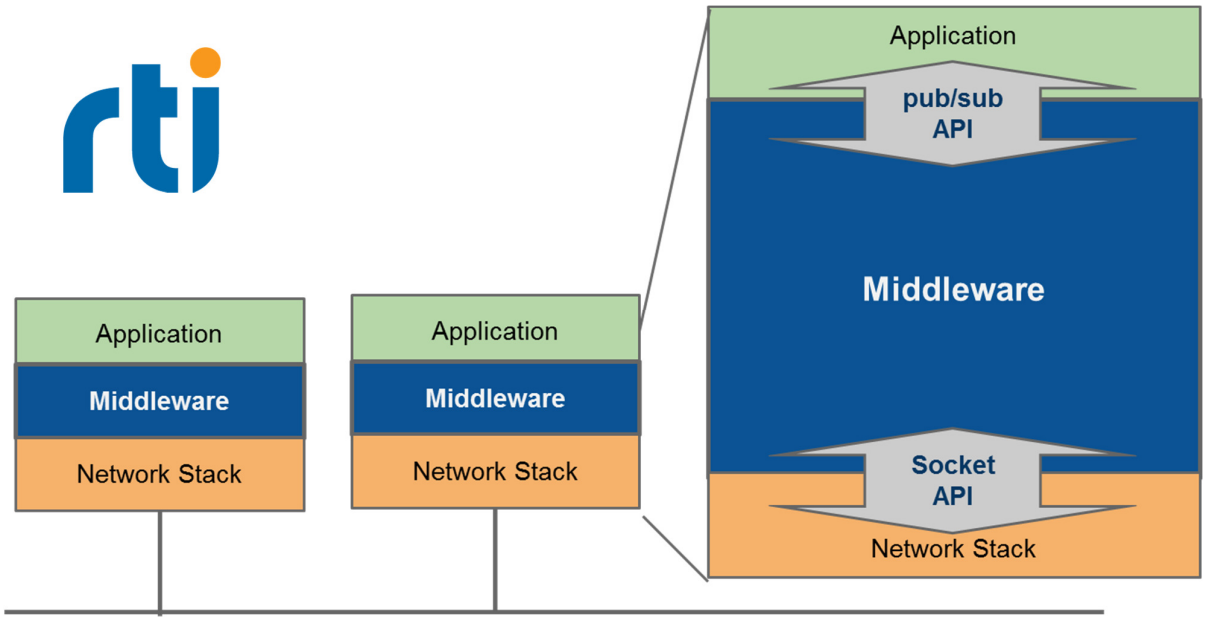
Access
- Link sparse endpoints
- **XMPP**

Process
- Biz intelligence
- Centralized/ESB
- ~100ms
- **MQ/AMQP**

Collect
- Collect data
- Hub & spoke
- ~10ms
- **MQTT/CoAP**

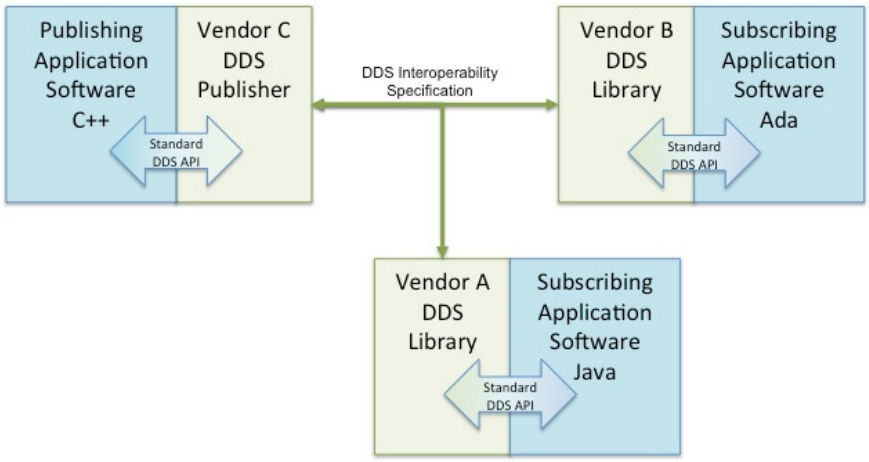
Control, distribute
- DataBus
- ~.01ms
- **DDS**

DDS [Data Distribution Service]



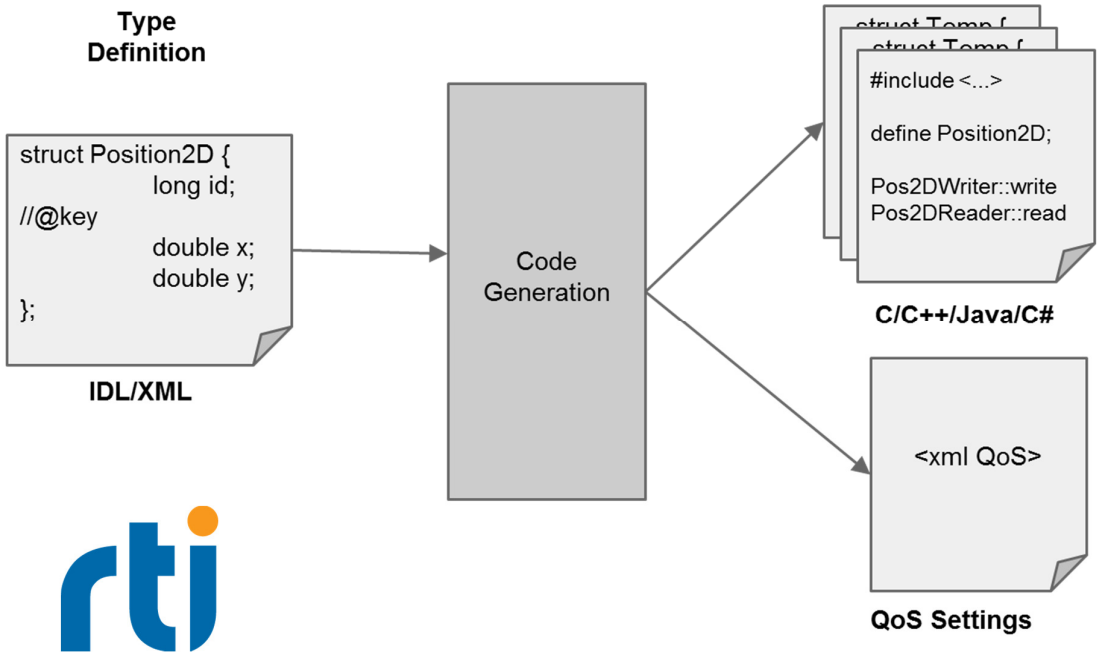
https://en.wikipedia.org/wiki/Data_Distribution_Service

DDS [Data Distribution Service]

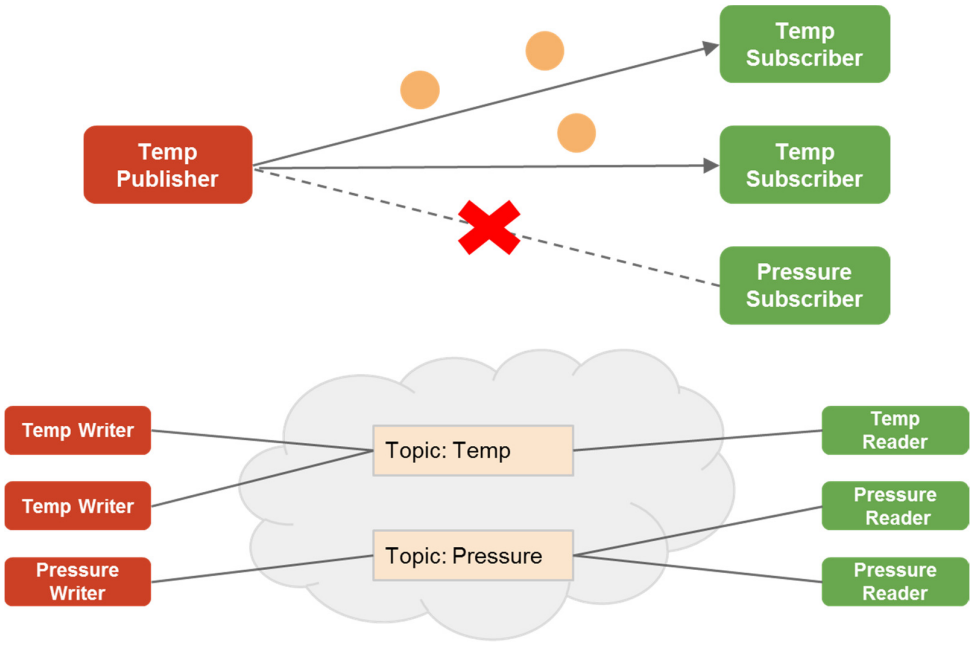


https://en.wikipedia.org/wiki/Data_Distribution_Service

DDS [Data Distribution Service]

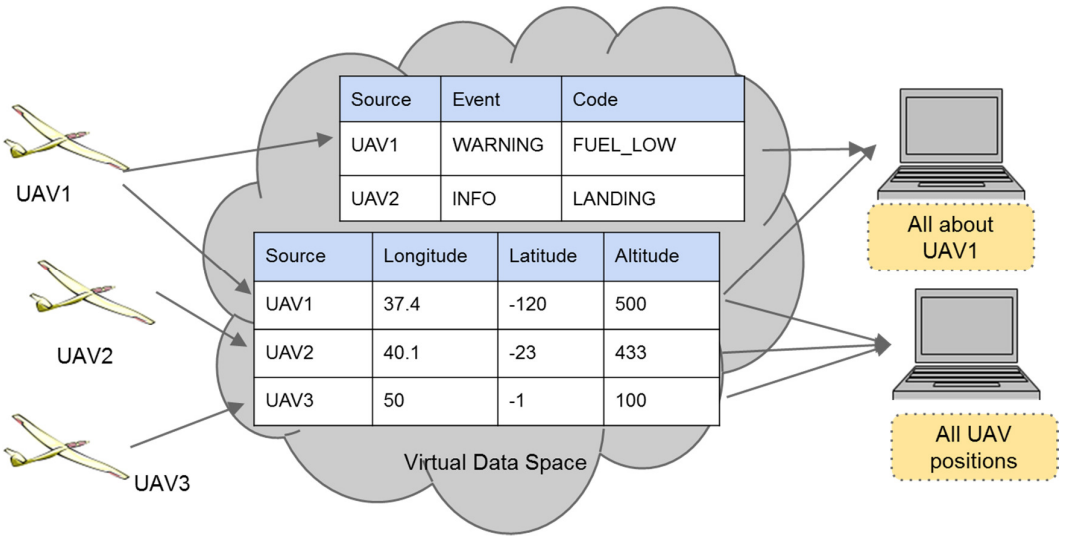


DDS [Data Distribution Service]



Your systems. Working as one.

DDS [Data Distribution Service]



Your systems. Working as one.

DDS [Data Distribution Service]

Publisher

```
connector = rti.Connector("MyParticipantLibrary::Sensor", 'Tutorial.xml')
writer = connector.getOutput("TempPublisher::TempWriter")
writer.instance.setString('id', sensor.id)
writer.write()
```

Subscriber

```
connector = rti.Connector("MyParticipantLibrary::Sensor", 'Tutorial.xml')
reader = connector.getInput("TempPublisher::TempWriter")
reader.read()
for i in nsamples
  if reader.infos.isvalid(i)
    ample = reader.samples.getDictionary(i)
```

Your systems. Working as one.

DDS [Data Distribution Service]

Configuración XML

1. QoS [Quality of Service]


```
<qos_library name="QosLibrary">
    <qos_profile name="DefaultProfile“ is_default_qos="true">
      <participant_qos>
        <transport_builtin>
          <mask>SHMEM</mask>  <!-- <mask>UDPV4</mask>-->
        </transport_builtin>
      </participant_qos>
      <datareader_qos>
        <!-- Modify reader values here -->
      </datareader_qos>
    </qos_profile>
  </qos_library>
```


DDS [Data Distribution Service]

Quality of Service	Quality of service
DURABILITY	USER_DATA
HISTORY	TOPIC_DATA
READER DATA LIFECYCLE	GROUP_DATA
WRITER DATA LIFECYCLE	PARTITION
LIFESPAN	PRESENTATION
ENTITY FACTORY	DESTINATION ORDER
RESOURCE LIMITS	OWNERSHIP
RELIABILITY	OWNERSHIP STRENGTH
TIME BASED FILTER	LIVELINESS
DEADLINE	LATENCY BUDGET
CONTENT FILTERS	TRANSPORT PRIORITY

Volatility

Infrastructure

Delivery

User

Presentation

Redundancy

Transport

DDS [Data Distribution Service]

Configuración XML

2. Definición de tipos

Your systems. Working as one.

```
<types>
  <struct name="Sensor" extensibility="extensible">
   <member name="id" stringMaxLength="128" id="0" type="string“
        key="true"/>
   <member name="value" id="1" type="long"/>
   <member name="timestamp" id="2" type="long"/>
 </struct>
</types>
```

```
struct Sensor {
        string id; //@key
        long value;
        long timestamp;
};
```

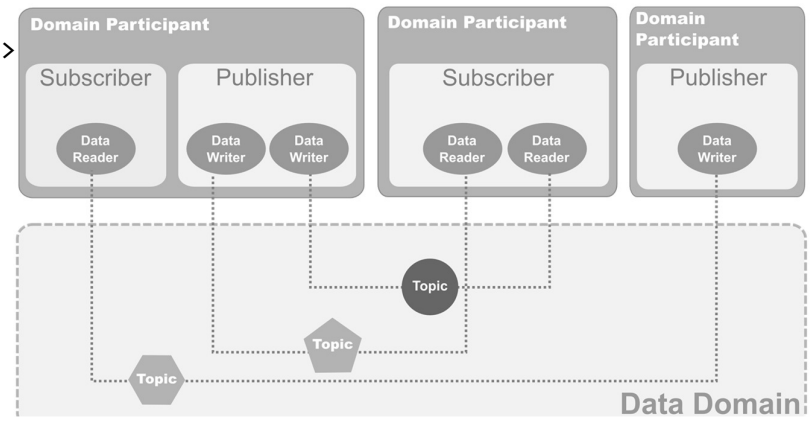

DDS [Data Distribution Service]

Configuración XML

3. Domains & Topics

Your systems. Working as one.

```
<domain_library name="MyDomainLibrary">
  <domain name="MyDomain" domain_id="0">
    <register_type name="Sensor" kind="dynamicData" type_ref="Sensor"/>
    <topic name="Temperature" register_type_ref="Sensor"/>
  </domain>
</domain_library>
```



DDS [Data Distribution Service]

Configuración XML

4. Entidades participantes


```
<participant_library name="MyParticipantLibrary">
 <domain_participant name="Console"
        domain_ref="MyDomainLibrary::MyDomain">
   <subscriber name="TempSubscriber">
     <data_reader name="TempReader" topic_ref="Temperature"/>
   </subscriber>
  </domain_participant>
  <domain_participant name="Sensor"
        domain_ref="MyDomainLibrary::MyDomain">
  <publisher name="TempPublisher">
    <data_writer name="TempWriter" topic_ref="Temperature"/>
    </publisher>
  </domain_participant>
</participant_library
```

DDS [Data Distribution Service]

Uso

- Filtrado (por contenido o tiempo).
- Configuración QoS:
  - Disponibilidad [availability]: liveliness/ownership.
    p.ej. sensor de respaldo (si falla el primario).
  - Durabilidad [durability]
    p.ej. historia reciente de los datos del sensor.
  - Particionamiento [data isolation/partition]
    p.ej. actualizaciones sólo de determinadas zonas.

DDS [Data Distribution Service]

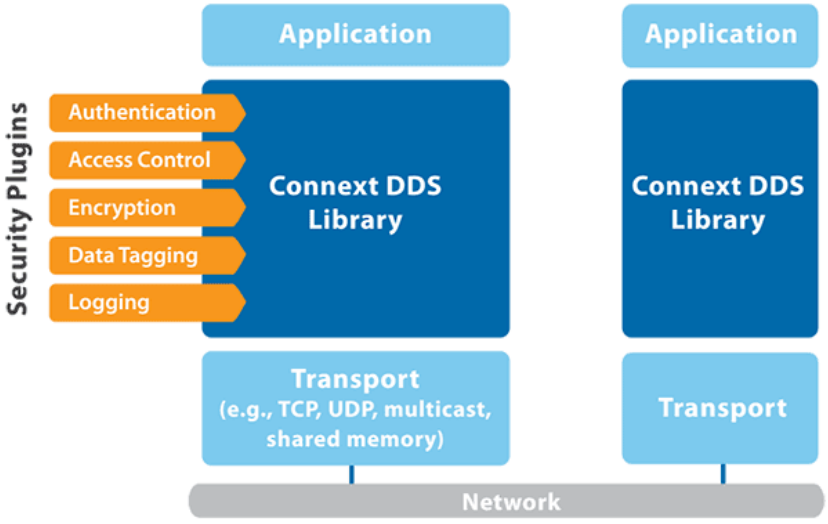
Uso: Problemas en la entrega de mensajes
https://blogs.rti.com/2016/03/02/where-is-my-data/

rti Your systems. Working as one.

- Proceso de descubrimiento
  Un DataReader puede perder las primeras muestras de un DataWriter si éste no ha descubierto al DataReader en el momento de publicar losprimeros datos (posible solución: mantener las muestras usando DurabilityQosPolicy=TRANSIENT_LOCAL).
- Comunicación fiable
  Aunque la comunicación sea fiable [ReliabilityQosPolicy=RELIABLE], en una red congestionada se pueden perder muestras si no las mantenemos en caché [HistoryQosPolicy=KEEP_ALL].
- DataReader marcado como inactivo
  Un DataReader se considera inactivo si no envía mensajes ACK/NACK en respuesta a n mensajes periódicos [heartbeat] del DaraWriter (parámetros max_heartbeat_retries & HB period).

DDS [Data Distribution Service]

rti Your systems. Working as one.



http://www.omg.org/spec/DDS-SECURITY/

Agradecimientos

Aída Jiménez Moscoso del Prado, Ph.D.
Senior Software Engineer
Real-Time Innovations