Βάσεις Δεδομένων

(12ο μάθημα)

Ηρακλής Βαρλάμης
varlamis@hua.gr

Εναλλακτικά μοντέλα υλοποίησης

- Τι άλλο υπάρχει εκτός από το Σχεσιακό;
  - Αντικειμενοστραφές - Αντικειμενο-σχεσιακό
  - Ιεραρχικό
  - Δικτυωτό
- NoSQL Databases
  - Graph Databases
  - Document Databases
  - Key-value stores
  - Column stores

Object-relational DBMS (ORDBMS)

ORDBMSs: Ορισμός και Παραδείγματα

Ορισμός: Ένα ORDBMS είναι ένα υβρίδιο που συνδυάζει αντικειμενοστρέφεια με τις παραδοσιακές σχεσιακές ΒΔ.

Παραδείγματα τέτοιων βάσεων είναι οι:

- Oracle Database
- PostgreSQL
- SQL Server
- Informix (IBM)

Πως επεκτείνουν το σχεσιακό μοντέλο

- Επιτρέπουν σύνθετα αντικείμενα και σύνθετες σχέσεις
- Επιτρέπουν σύνθετους τύπους δεδομένων
- Ο κώδικας διαχωρίζεται από τα δεδομένα
- Δεν υποστηρίζονται όλες οι ιδιότητες του αντικειμενοστραφούς μοντέλου (κληρονομικότητα, ενθυλάκωση κλπ.

PL/SQL Block Types

Anonymous

```
[DECLARE]


BEGIN
   --statements

[EXCEPTION]

END;
```

Procedure

```
PROCEDURE name
IS

BEGIN
   --statements

[EXCEPTION]

END;
```

Function

```
FUNCTION name
RETURN datatype
IS
BEGIN
   --statements
   RETURN value;
[EXCEPTION]

END;
```

Τύποι οριζόμενοι από το χρήστη

```
CREATE TYPE <typename> AS OBJECT
(
        <list of attribute-type pairs>
);
/

CREATE TYPE BarType AS (
        name CHAR(20),
        addr CHAR(20)
 );
/
```

Παράδειγμα

```
CREATE OR REPLACE TYPE ADDRESS_TYPE AS OBJECT
(STREET VARCHAR2(25),
CITY VARCHAR2(25),
COUNTRY VARCHAR2(25));
/

CREATE TABLE STUDENT
(AM NUMBER(4),
NAME VARCHAR2(20),
ADDRESS ADDRESS_TYPE);
```

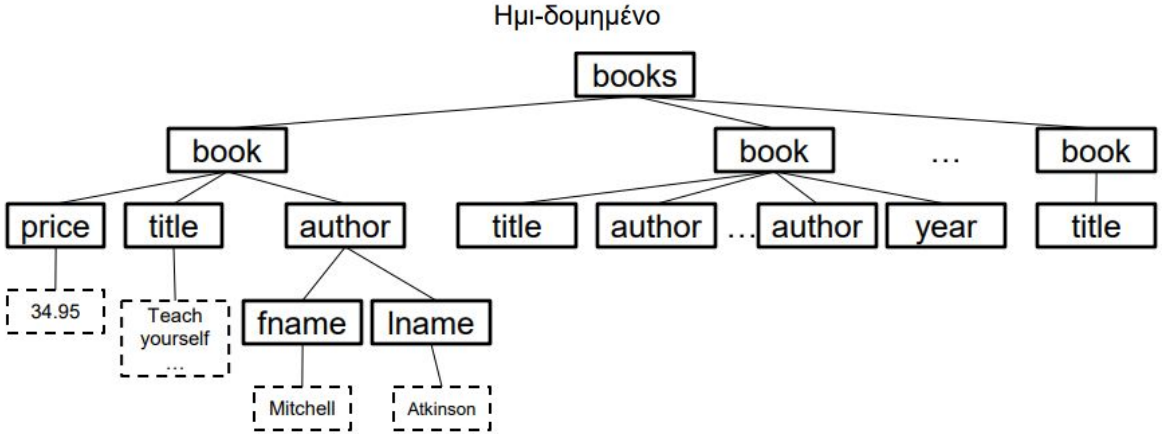
Εισαγωγή δεδομένων

INSERT INTO STUDENT
VALUES (1234, ‘George Georgiou’,
ADDRESS_TYPE(‘Ermou 10’, ‘Athina’, ‘Hellas’));

Ιεραρχικό μοντέλο
XML DBMS

Ιεραρχικό μοντέλο

- Αναπαράσταση μιας συλλογής βιβλίων
- Δεν έχουμε για όλα τα βιβλία
- Τις ίδιες τιμές
- Την ίδια δομή πληροφορίας



Η γλώσσα HTML

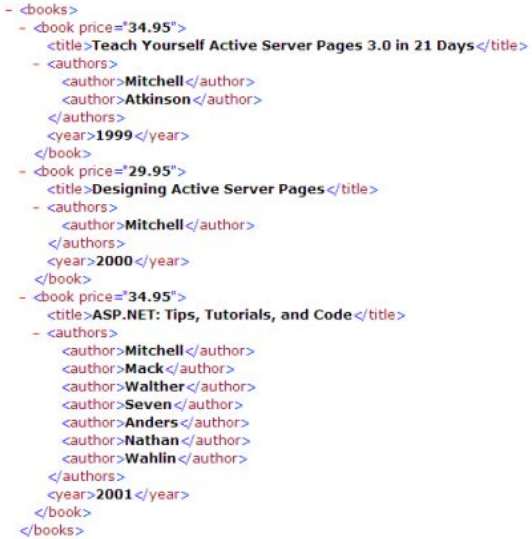
- Η βασική γλώσσα που χρησιμοποιούν οι ιστοσελίδες
- Έχει συγκεκριμένη δομή
- Έχει περιορισμένο αριθμό από ετικέτες (tags) κατά συνέπεια μπορούμε να αναπαραστήσουμε συγκεκριμένη πληροφορία
- Οι ετικέτες αφορούν τη μορφοποίηση της εμφάνισης
- Έχει ορισμένους περιορισμούς σύνταξης αλλά δεν έχει αυστηρή δομή για τα δεδομένα, ούτε δυνατότητες αυτο-περιγραφής

Αδόμητο

```
<html>
<body>
   <h1>My First Heading</h1>
   <p>My first paragraph.</p>
   <table border="1">
   <tr>
   <td>row 1, cell 1</td>
   <td>row 1, cell 2</td>
   </tr>
   <tr>
   <td>row 2, cell 1</td>
   <td>row 2, cell 2</td>
   </tr>
   </table>
</body>
</html>
```

Τι είναι η XML

- EXtensible Markup Language
- Markup: Είναι δηλωτική γλώσσα, χρησιμοποιεί ετικέτες με συγκεκριμένη σημασία.
- Extensible: Είναι επεκτάσιμη. Οι ετικέτες της XML δεν είναι προκαθορισμένες. Καθορίζονται απ το χρήστη.
- Η XML είναι αυτο-περιγραφική.
- Η XML σχεδιάστηκε για να περιγράφει τα δεδομένα.

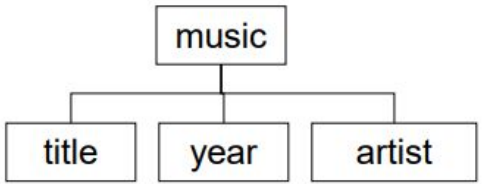


Η χρήση της XML

- Η XML χρησιμοποιείται για να αποθηκεύσουμε και να οργανώσουμε τα δεδομένα όχι για την παρουσίαση των δεδομένων (όπως κάνει η HTML).
- Η XML χρησιμοποιείται για ανταλλαγή δεδομένων
  - Το πρόγραμμά μου εξάγει τα δεδομένα του σε αρχείο XML.
  - Ένα άλλο πρόγραμμα μπορεί να διαβάσει τα XML αρχεία πιο εύκολα απ' ότι τα απλά αρχεία κειμένου.
  - Σε ένα XML έγγραφο, διατηρείται η δομή των δεδομένων μου.
- Η XML χρησιμοποιείται για αποθήκευση δεδομένων.
  - Τα XML αρχεία μπορούν να αντικαταστήσουν τις ΒΔ. Αρκεί να υπάρχουν τρόποι να κάνουμε επερωτήσεις και ενημερώσεις.

Παράδειγμα

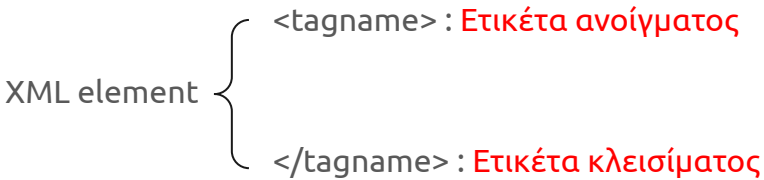
```
<?xml version="1.0"?>
<music>
   <title>Nevermind</title>
   <year>1991</year>
   <artist>Nirvana</artist>
</music>
```



- Η πρώτη γραμμή δηλώνει την έκδοση της XML για το συγκεκριμένο έγγραφο. Μπορεί να περιέχει πληροφορίες για την κωδικοποίηση (encoding) του εγγράφου.
- Η δεύτερη γραμμή ορίζει το πρώτο στοιχείο του εγγράφου (τη ρίζα του) που είναι μοναδικό.
- Οι υπόλοιπες 3 γραμμές ορίζουν τα στοιχεία που βρίσκονται κάτω από τη ρίζα (title, year, artist).
- Η τελευταία γραμμή κλείνει το στοιχείο στη ρίζα. (</...>)

XML Elements

- Όλα τα στοιχεία (elements) της XML πρέπει να έχουν ετικέτες ανοίγματος και κλεισίματος
  - <artist>Nirvana</artist>
- Οι ετικέτες της XML είναι case sensitive
  - Η ετικέτα <Year> είναι διαφορετική από την </year>
    - <Year>1991</year> //Λάθος
    - <year>1991</year> //Σωστό
- Τα elements στην XML πρέπει να είναι σωστά φωλιασμένα.
  - <music><title>Nevermind</music></title> //Λάθος
- Όλα τα XML έγγραφα πρέπει να έχουν μία ρίζα.
  - Η ρίζα πρέπει να είναι μοναδική. Όλα τα υπόλοιπα στοιχεία θα βρίσκονται φωλιασμένα σ' αυτήν.
    - Δεν μπορούμε να έχουμε δύο music στοιχεία



XML Attributes

- Τα attributes χρησιμοποιούνται για να περιγράψουν τα χαρακτηριστικά των στοιχείων της XML.
- Περιέχονται πάντα στην αρχική ετικέτα του element σε ζευγάρια της μορφής: όνομα="τιμή”.
  - <music code="1545">
- Οι τιμές των χαρακτηριστικών (attributes) πρέπει να βρίσκονται σε εισαγωγικά.
  - <music code=1545> … </music> //Λάθος
  - <music code="1545"> … </music> //Σωστό
- Τα attributes έχουν περιορισμούς στον τύπο που πρέπει να έχουν τα δεδομένα που περιέχουν, γι’ αυτό και προτιμούμε τα elements
- Τα attributes δεν μπορούν να περιγράψουν σύνθετες τιμές

XPath - Επιλογή τμημάτων ενός XML εγγράφου

- Ένα συντακτικό για να προσδιορίζουμε μέρη ενός εγγράφου XML
- Έχει παρόμοια λογική με το σύστημα προσδιορισμού αρχείων του λειτουργικού συστήματος
- Δεν είναι γραμμένη σε XML, αλλά ενσωματώνεται σε άλλες γλώσσες συνοδευτικές της XML
- Είναι πρότυπο του W3C

Ορολογία

```
- <books>
  - <book price="34.95">
      <title>Teach Yourself Active Server Pages 3.0 in 21 Days</title>
    - <authors>
        <author>Mitchell</author>
        <author>Atkinson</author>
      </authors>
      <year>1999</year>
    </book>
  - <book price="29.95">
      <title>Designing Active Server Pages</title>
    - <authors>
        <author>Mitchell</author>
      </authors>
      <year>2000</year>
    </book>
  - <book price="34.95">
      <title>ASP.NET: Tips, Tutorials, and Code</title>
    - <authors>
        <author>Mitchell</author>
        <author>Mack</author>
        <author>Walther</author>
        <author>Seven</author>
        <author>Anders</author>
        <author>Nathan</author>
        <author>Wahlin</author>
      </authors>
      <year>2001</year>
    </book>
</books>
```

- Το books είναι γονιός τριών book;
- Το book είναι γονιός των title, authors και year
- Τα δύο author είναι παιδιά του πρώτου authors και αντίστοιχα τα υπόλοιπα
- Κάθε author κάτω από ένα authors είναι αδερφός (sibling) με όλα τα υπόλοιπα author κάτω από το ίδιο authors
- Τα books, book και authors είναι πρόγονοι (ancestors) του author
- Το author είναι απόγονος (descendent) του books

Μονοπάτια

/books = το root element

/books/book/authors/author = κάθε συγγραφέας σε οποιαδήποτε λίστα συγγραφέων σε οποιοδήποτε βιβλίο στη συλλογή

author = κάθε συγγραφέας που βρίσκεται κάτω από το τρέχον element,

. = το τρέχον element

.. = ο γονιός του τρέχοντος element

/books/book/* = όλα τα elements ενός βιβλίου στη συλλογή

Slashes

- Ένα path που ξεκινά με / είναι απόλυτο μονοπάτι που ξεκινά από την ρίζα του εγγράφου
  - π.χ. : /books/book/authors/author
  - Μπορεί να επιστρέψει πολλά elements (που βρίσκονται κάτω από διαφορετικού γονείς)
  - Μια σκέτη / σημαίνει όλο το έγγραφο
- Ένα path που δεν αρχίζει με / δηλώνει ότι ξεκινά από το τρέχον element
  - π.χ. : authors/author
- Ένα path που ξεκινά με // μπορεί να ξεκινά από οπουδήποτε στο κείμενο
  - π.χ.: //date/year επιλέγει τα έτη από όλες τις ημερομηνίες
  - Ψάχνει εξαντλητικά όλο το κείμενο (μπορεί να είναι αργό)

Αγκύλες και last()

- Ένας αριθμός σε αγκύλες προσδιορίζει το α/α του element που ταιριάζει στο path (η μέτρηση ξεκινά από 1)
  - /books/book[1] το πρώτο βιβλίο στη συλλογή
  - //authors/author[1] ο πρώτος συγγραφέας κάθε ομάδας συγγραφέων
  - //book/chapter[1]/section[2]
  - Μετράμε μόνο τα elements που ταιριάζουν στο path μέχρι το τρέχον σημείο
- Η last() εντός των αγκυλών επιστρέφει το τελευταίο element που ταιριάζει
  - /books/book/authors/author[last()]
- Υποστηρίζονται απλές πράξεις
  - /books/book/authors/author[last()-1]

Stars

- Ένα * δηλώνει όλα τα elements του επιπέδου
  - /books/book/authors/* επιλέγει κάθε συγγραφέα κάθε βιβλίου στη συλλογή
  - //book/* επιλέγει κάθε element κάθε book (title, authors, year)
  - /*/*/*/author επιλέγει κάθε author σε βάθος 3 στο δέντρο του XML εγγράφου
  - //* επιλέγει κάθε element σε όλο το έγγραφο (χάνεται η έννοια του βάθους)

Attributes

- Μπορούμε να επιλέξουμε attributes μόνα τους ή elements που περιέχουν συγκεκριμένα attributes
  - Το attribute είναι ένα ζεύγος όνομα-τιμή <chapter num="5">
  - Για να επιλέξουμε το attribute, βάζουμε @ μπροστά στο όνομα
  - π.χ.: @num επιλέγει κάθε attribute που λέγεται num
  - π.χ.: //@* επιλέγει κάθε attribute, οπουδήποτε στο έγγραφο
- Για να επιλέξουμε elements που έχουν κάποιο attribute, βάζουμε το attribute στις αγκύλες
  - π.χ.: //chapter[@num] επιλέγει κάθε chapter element (οπουδήποτε στο κείμενο) που έχει attribute με όνομα num
  - π.χ.: //chapter[not(@num)] επιλέγει κάθε chapter element (οπουδήποτε στο κείμενο) που δεν έχει attribute με όνομα num
  - π.χ.: //chapter[@num='3'] επιλέγει κάθε chapter element με attribute num το οποίο έχει τιμή 3

Δικτυωτό μοντέλο
RDF και Triplestores

RDF (Resource Description Framework)

- Ένα αφηρημένο, εννοιολογικό μοντέλο δεδομένων
- Είναι ανεξάρτητο από το πεδίο εφαρμογής
- Αναπαριστά τα δεδομένα με τη μορφή κατευθυνόμενου γράφου με ετικέτες.
- Τα δεδομένα αναπαρίστανται με XML, με τριπλέτες (πλειάδες τριών τιμών), με JSON κλπ
- Οι επίσημες οδηγίες για την RDF δίνονται από το

  http://www.w3.org/RDF/s

Ταυτότητα και περιγραφή

- Η RDF χρησιμοποιεί το URI ως id
  - Οτιδήποτε έχει URI είναι Resource
- Κάθε **resource** έχει ιδιότητες
  - Μια ιδιότητα (property) έχει ένα όνομα και μια τιμή
    - Όνομα: Author, Book, Address, Client, Product
    - Τιμή (property value): “Ora Lassila”, "rdf-syntax", http://www.someplace.com
  - Κάθε τιμή ιδιότητας μπορεί να είναι ένα άλλο resource

Πώς θα περιγράψω το ακόλουθο;

- "The author of http://www.w3schools.com/RDF is Jan Egil Refsnes".

  Resource είναι το

  - http://www.w3schools.com/RDF

- "The homepage of http://www.w3schools.com/RDF is http://www.w3schools.com".

  Resource είναι το

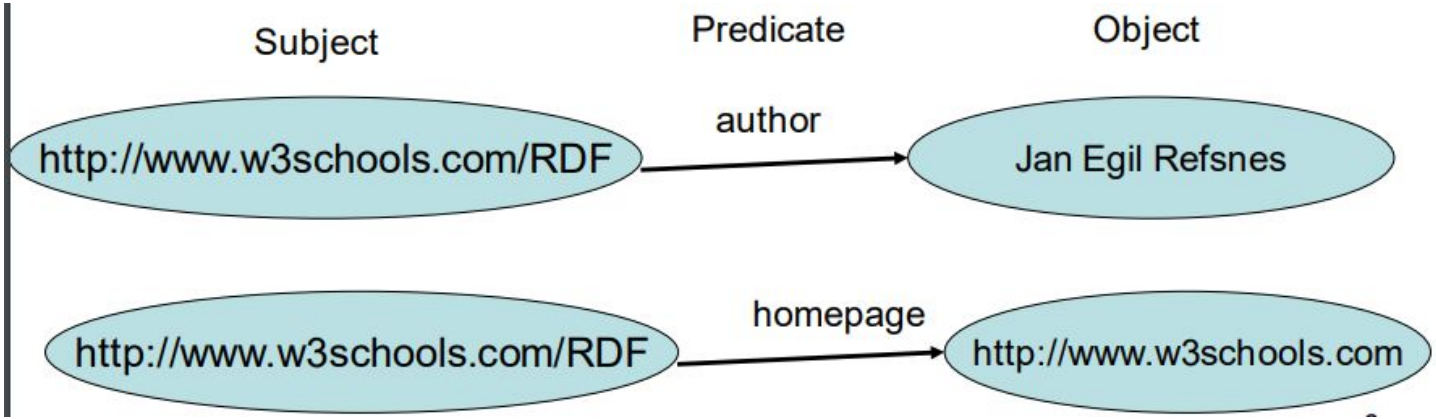
  - http://www.w3schools.com/RDF

  και το

  - http://www.w3schools.com

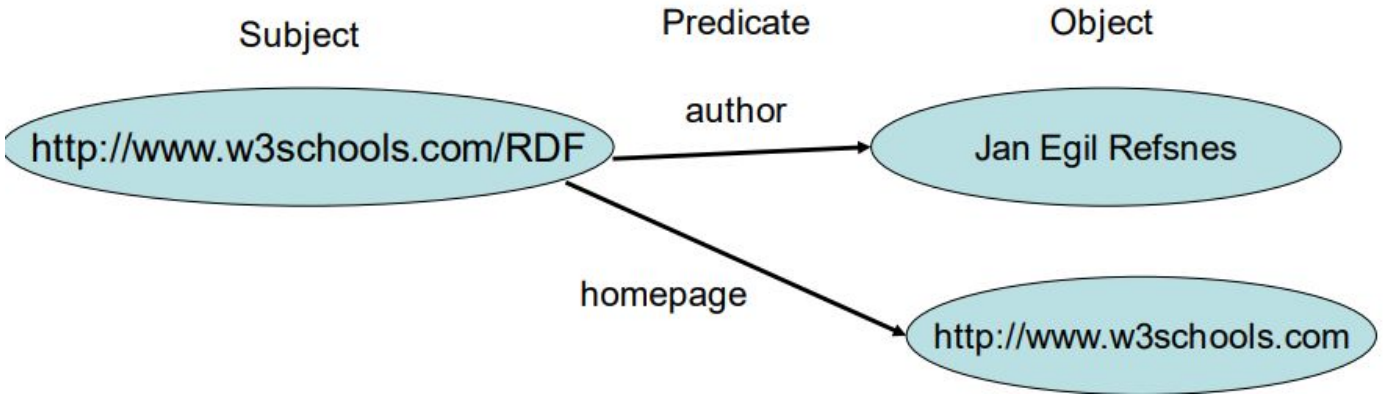
Κατηγορούμενο - Predicate

- Τα predicates προσδιορίζουν μια σχέση ανάμεσα σε ένα υποκείμενο (subject) και ένα αντικείμενο (object)
- Η RDF έχει μόνο δυαδικά predicates
- Στο γράφο μου κάθε predicate είναι μια ακμή από το subject στο object

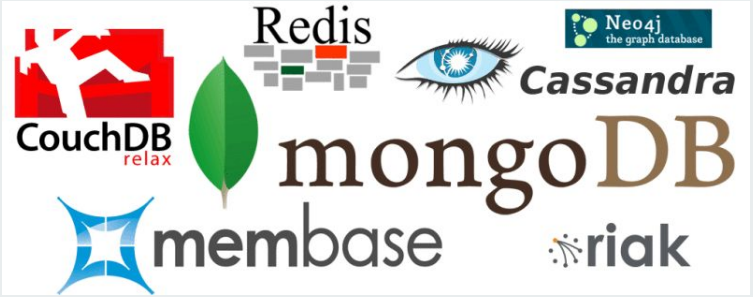


Κατηγορούμενο - Predicate

- Τα predicates προσδιορίζουν μια σχέση ανάμεσα σε ένα υποκείμενο (subject) και ένα αντικείμενο (object)
- Η RDF έχει μόνο δυαδικά predicates
- Στο γράφο μου κάθε predicate είναι μια ακμή από το subject στο object



NoSQL Databases



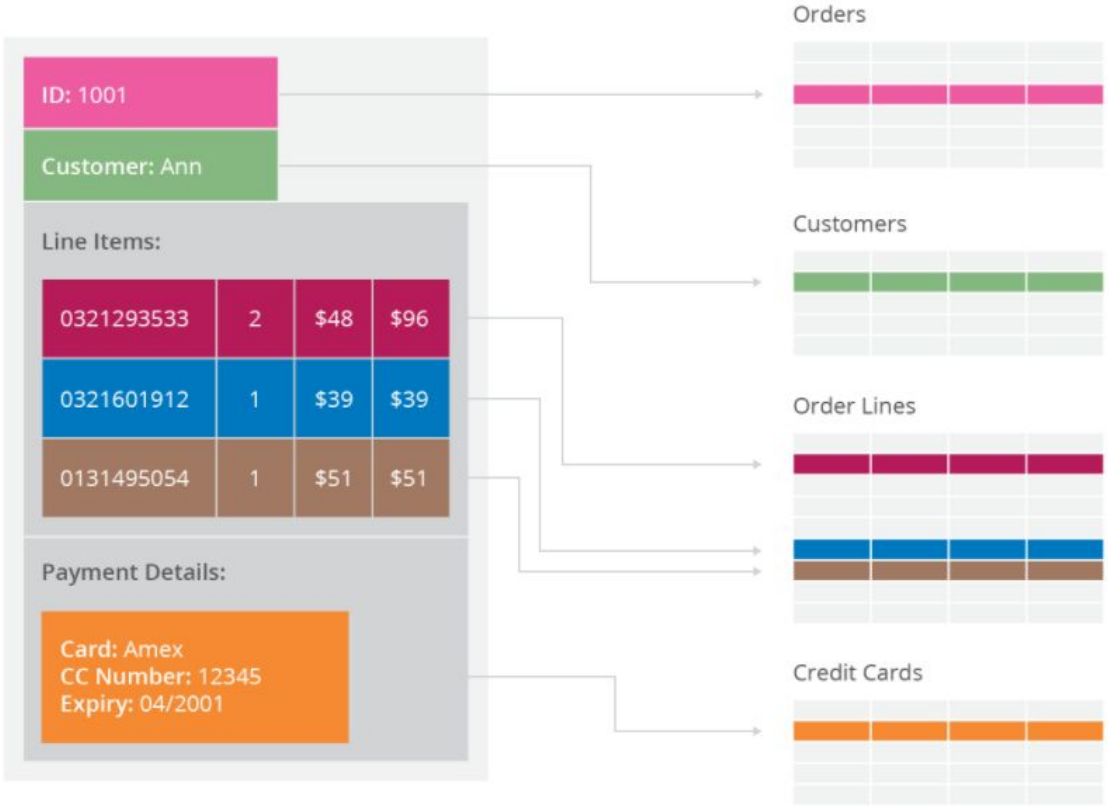
Ορισμός

Μια NoSQL ΣΔΒΔ παρέχει ένα μηχανισμό αποθήκευσης και ανάκτησης δεδομένων που διαφέρει από τον παραδοσιακό μηχανισμό που έχουν οι σχεσιακές ΒΔ.

Τι είναι τα NoSQL συστήματα

- Ο όρος NoSQL, μεταφράζεται ως «Not Only SQL»
- Κύριο χαρακτηριστικό τους είναι το ότι δεν τηρούν το RDBMS μοντέλο (Relational Database Management System)
- Είναι κατανεμημένες, μη σχεσιακές (non-relational) βάσεις, σχεδιασμένες για αποθήκευση δεδομένων μεγάλης κλίμακας και παράλληλη επεξεργασία δεδομένων μοιρασμένα σε έναν μεγάλο αριθμό από servers.
- Δεν χρησιμοποιούν κάποιο δομημένο σύστημα για τα στοιχεία που περιλαμβάνουν, όπως για παράδειγμα πίνακες
- Χρησιμοποιούν αποκλειστικά non-relational τρόπους οργάνωσης και ανάλυσης των δεδομένων
- Τα κύρια χαρακτηριστικά τους συνοψίζονται στα εξής :
    - Η μη χρήση της SQL ως query language,
    - Δεν μπορούν να εγγυηθούν ότι οι διεργασίες στην βάση δεδομένων θα γίνονται αξιόπιστα
- Έχουν μια ιδιαίτερη αρχιτεκτονική και φιλοσοφία λειτουργίας

Γιατί NoSQL Databases



Γιατί, που και πότε είναι χρήσιμες

- Τα NoSQL συστήματα δεν εμφανίστηκαν για να αντικαταστήσουν τα σχεσιακά, αλλά για να τα συμπληρώσουν.
- Τα NoSQL συστήματα διαχείρισης βάσεων δεδομένων είναι εξαιρετικά χρήσιμα όταν κάποιος δουλεύει με τεράστιο όγκο δεδομένων, η φύση των οποίων δεν απαιτεί κάποιο σχεσιακό μοντέλο.
- Η κατανεμημένη τους φύση τα καθιστά ιδανικά για μαζική επεξεργασία δεδομένων (πχ ενοποίηση, φιλτράρισμα, διαλογή, στατιστικές ενέργειες κλπ).
- Είναι επίσης πολύ καλά για ανάκτηση και ανταλλαγή δεδομένων μεταξύ μηχανημάτων (machine-to-machine), καθώς και για την επεξεργασία συναλλαγών μεγάλου όγκου.

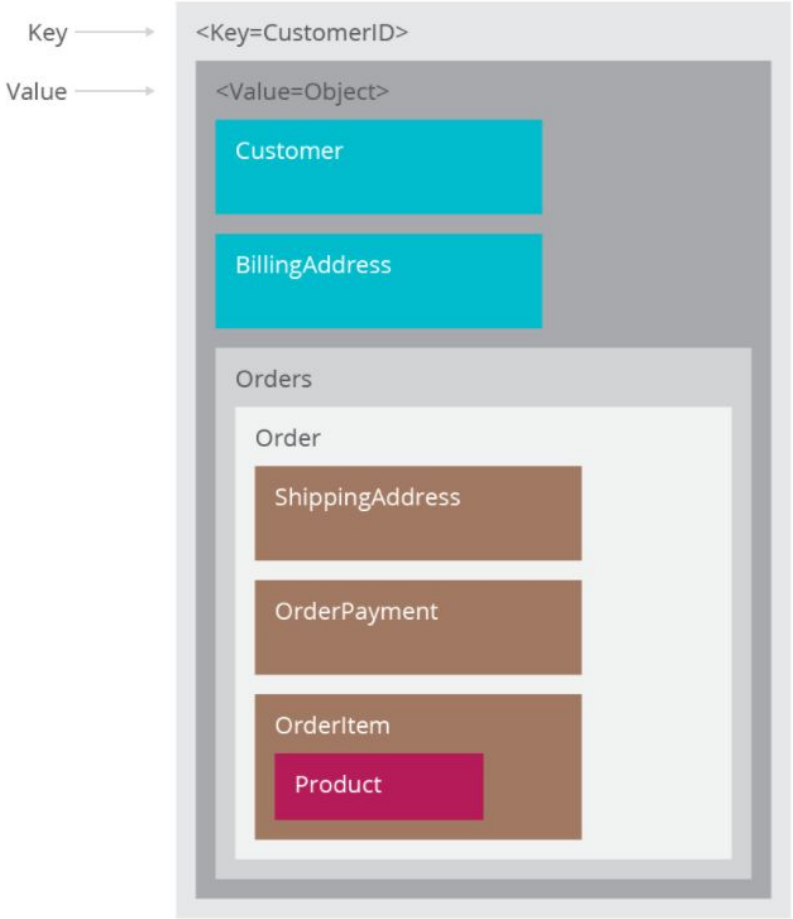
Γιατί, που και πότε είναι χρήσιμες

- Παρέχουν σχετικά φθηνή, υψηλής επεκτασιμότητας αποθήκευση μεγάλου όγκου
  - ιστορικά δεδομένα, αρχεία καταγραφής, αρχεία τηλεφωνικών δεδομένων, ενδείξεις αισθητήρων, αποθήκευση (semi-structured) ή (unstructured) δεδομένων, όπως αρχεία ηλεκτρονικού ταχυδρομείου, αρχεία XML, κλπ.
- Έχουν την ικανότητα της κλιμάκωσης
- Τα NoSQL συστήματα έχουν πιο αδύναμα μοντέλα συνέπειας των δεδομένων, μπορούν να «θυσιάσουν» την συνοχή για την αποδοτικότητα

Τύποι NoSQL ΣΔΒΔ

- **Key-value stores.** Η πιο απλή μορφή NoSQL ΣΔΒΔ. Κάθε στοιχείο στη βάση δεδομένων αποθηκεύεται με ένα όνομα κλειδί ('key'), μαζί με την τιμή του (value). Κάποια παραδείγματα key-value stores επιτρέπουν να ορίσουμε τον τύπο κάθε τιμής. Παραδείγματα: Oracle BerkleyDB, Redis (in memory).
- **Document databases**. Κάθε κλειδί (key) αντιστοιχεί σε μια σύνθετη δομή δεδομένων που ονομάζεται έγγραφο (document). Τα documents μπορούν να περιέχουν πολλαπλά ζεύγη key-value, ή key-array, ή φωλιασμένα έγγραφα. Παράδειγμα: ElasticSearch, MongoDB
- **Graph stores**. Χρησιμοποιούνται για να αποθηκεύσουμε πληροφορία για δίκτυα δεδομένων όπως π.χ. για τις συνδέσεις σε ένα κοινωνικό δίκτυο. Παράδειγμα: Neo4J, Giraph.
- **Column stores.** Έχουν βελτιστοποιηθεί για να απαντούν ερωτήματα σε μεγάλα σύνολα δεδομένων και αποθηκεύουν ενιαίες τις στήλες των δεδομένων (αντί για τις πλειάδες). Παραδείγματα: Cassandra, HBase.

Key-value stores



Key-value databases:

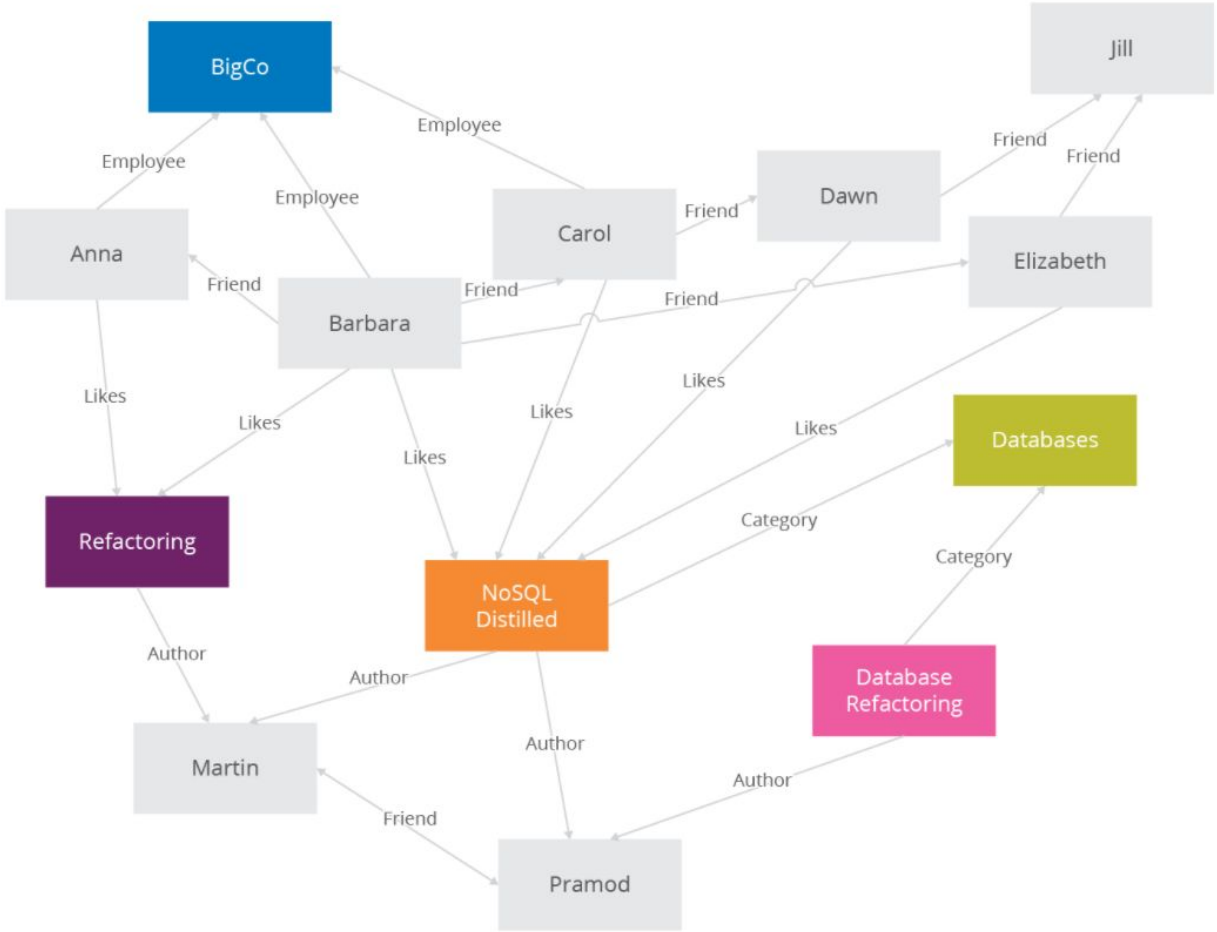
- Riak
- Redis
- Memcached
- Berkeley DB
- Amazon DynamoDB (not open-source)
- Project Voldemort and Couchbase.

Document databases

Document databases:

- MongoDB
- Elastic Search
- CouchDB
- Terrastore
- OrientDB
- RavenDB

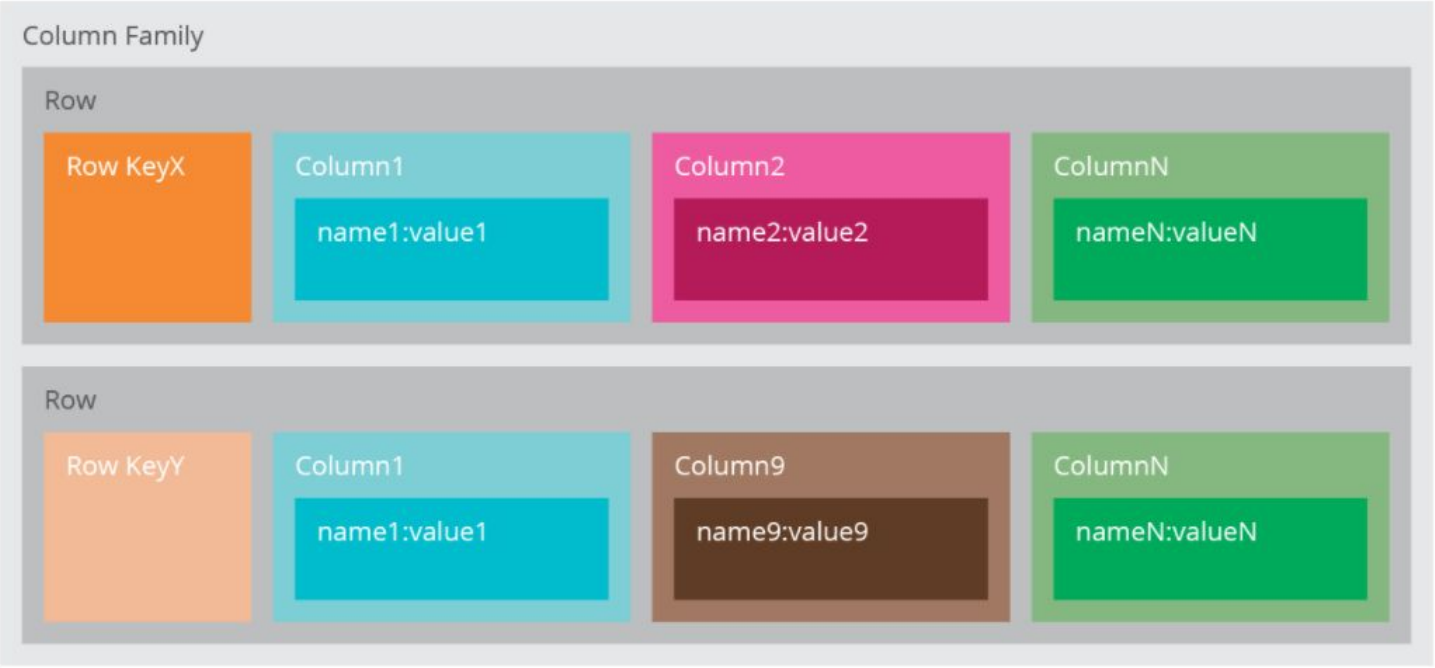
Graph stores



Graph databases:

- Neo4J
- RedisGraph
- OrientDB
- DGraph

Column stores



Column-store databases:

- Cassandra
- HBase
- Hypertable
- Amazon DynamoDB

Χαρακτηριστικά ανά κατηγορία NoSQL συστημάτων

Μοντέλο Δεδομένων	Απόδοση	Επεκτασιμότητα	Ευελιξία	Πολυπλοκότητα	Λειτουργικότητα
Key-value	high	high	high	none	Variable (none)
Column	high	high	moderate	low	minimal
Document	high	Variable (high)	high	low	Variable (low)
Graph	variable	variable	high	high	graph theory
Relational	variable	variable	low	moderate	relational algebra

SQL Injection

Φάσεις ενός SQL Injection

- Έρευνα: Ο επιτιθέμενος επιχειρεί την υποβολή διάφορων μη αναμενόμενων τιμών, παρατηρεί πώς απαντά η εφαρμογή και αποφασίζει ένα πλάνο επίθεσης.
- Επίθεση: Ο επιτιθέμενος δίνει μία προσεκτικά επιλεγμένη τιμή εισόδου, η οποία όταν θα χρησιμοποιηθεί σαν παράμετρος ενός SQL ερωτήματος, θα “μεταφραστεί” σαν τμήμα της εντολής SQL, και η ΒΔ θα εκτελέσει το ερώτημα παρέχοντας στον επιτιθέμενο την πρόσβαση που του ορίζει το τροποποιημένο ερώτημα.

Τύποι επίθεσης & παραδείγματα

Τα SQL Injections μπορούν να προκαλέσουν μεγαλύτερη ζημιά από την παράκαμψη των αλγορίθμων login. Αυτές οι επιθέσεις περιλαμβάνουν:

- Διαγραφή δεδομένων
- Ενημέρωση δεδομένων
- Εισαγωγή δεδομένων
- Εκτέλεση εντολών στον διακομιστή για το κατέβασμα και την εγκατάσταση κακόβουλων προγραμμάτων, όπως Trojans
- Εξαγωγή πολύτιμων δεδομένων, όπως στοιχεία πιστωτικών καρτών, email, και κωδικών στον υπολογιστή του επιτιθέμενου

Τύποι επίθεσης & παραδείγματα

Υπάρχουν πολλές μέθοδοι για την εκτέλεση επιθέσεων SQL injection όπως:

- Μετατροπή του απλού λογαριασμού χρήστη σε διαχειριστή
- Επιθέσεις βασισμένες στα σφάλματα SQL (μέσω της επιστροφής μηνυμάτων σφάλματος από τον DB server)
- Επιθέσεις που αναγκάζουν τον DB server να παραμένει απασχολημένος για πολύ ώρα (Time-based Blind SQLi)
- Επιστροφή περισσότερων δεδομένων από αυτά που θα έπρεπε κανονικά

Επιστροφή περισσότερων δεδομένων από όσα θα έπρεπε

Ας υποθέσουμε πως ένας προγραμματιστής θέλει να επιστρέψει σε μία εφαρμογή τους αριθμούς λογαριασμού και τα πιστωτικά όρια με βάση το id του χρήστη. Σε Java λοιπόν μπορεί να έγραφε το παρακάτω block κώδικα:

```
String accountBalanceQuery =
  "SELECT accountNumber, balance FROM accounts WHERE account_owner_id = "
  + request.getParameter("user_id");

try {
    Statement statement = connection.createStatement();
    ResultSet rs = statement.executeQuery(accountBalanceQuery);
    while (rs.next()) {
        page.addTableRow(rs.getInt("accountNumber"), rs.getFloat("balance"));
    }
} catch (SQLException e) { ... }
```

Επιστροφή περισσότερων δεδομένων από όσα θα έπρεπε

Μπορεί για παράδειγμα ο χρήστης με ID 984 να έχει κάνει log in, και το URL στο οποίο πρέπει να πάει ο χρήστης είναι:
https://bankingwebsite/show_balances?user_id=984

Άρα το αντίστοιχο SQL query θα ήταν:
SELECT accountNumber, balance FROM accounts WHERE account_owner_id = 984;

Πώς μπορεί λοιπόν κανείς να εκμεταλλευτεί αυτό το query για να εκτελέσει ένα SQLi?

Επιστροφή περισσότερων δεδομένων από όσα θα έπρεπε

Μπορεί για παράδειγμα ο χρήστης με ID 984 να έχει κάνει log in, και το URL στο οποίο πρέπει να πάει ο χρήστης είναι:
https://bankingwebsite/show_balances?user_id=984

Άρα το αντίστοιχο SQL query θα ήταν:
SELECT accountNumber, balance FROM accounts WHERE account_owner_id = 984;

Μία τακτική είναι να παραποιήσεις τις παραμέτρους εισόδου ώστε για παράδειγμα να μεταφραστούν από τον μεταγλωττιστή ως:
0 OR 1=1

Αποτέλεσμα είναι το query που θα εκτελεστεί στον server να είναι:
SELECT accountNumber, balance FROM accounts WHERE account_owner_id = 0 OR 1=1

Τρόποι προστασίας απέναντι σε επιθέσεις SQL Injection

- **Τα δεδομένα εισόδου δεν πρέπει ποτέ να θεωρούνται δεδομένα** - Πρέπει πάντα να φιλτράρονται πριν τα SQL statements.
- **Stored procedures** – μπορούν να ενθυλακώσουν τα SQL statements και να διαχειριστούν τις παραμέτρους εισόδου.
- **Prepared statements** – μπορούν πρώτα να χτίσουν το SQL statement και μετά να χειριστούν τις παραμέτρους εισόδου.
- **Regular expressions** – μπορούν να χρησιμοποιηθούν για να ανιχνεύσουν πιθανώς κακόβουλο κώδικα και να τον αφαιρέσουν πριν την εκτέλεση των SQL ερωτημάτων.
- **Δικαιώματα σύνδεσης στον Database server** – κάθε χρήστης πρέπει να έχει μόνο τα απαραίτητα δικαιώματα χρήσης και τίποτα περισσότερο.
- **Μηνύματα λάθους** – Δεν θα πρέπει να αποκαλύπτουν ευαίσθητες πληροφορίες ή το ποιό ακριβώς σφάλμα προέκυψε.