Complementi di Basi di Dati

Dati non relazionali (non strutturati)

- I dati richiedono un'elaborazione specifica.
- Aggiornamenti totali
- **Modello basato sulla classifica:** i risultati vengono classificati per grado di pertinenza e l'utente ammette possibili "errori". Non è possibile determinare con precisione se un risultato è completamente o per niente rilevante
    - Modello booleano non utilizzabile data la natura delle query, la grande quantità di risposte possibili e dato che più documenti possono rispondere più o meno bene ai requisiti espressi nella domanda
- **Information Retrieval:** studia i dati senza schema (oggetti multimediali e file di testo narrativo). I DBMS hanno già integrato alcune funzionalità derivate dall'Information Retrieval, come l'indicizzazione di colonne che contengono solo testo (CLOB) o colonne per dati multimediali (BLOB).
- Query semplici composte da elenchi di **parole chiave**, come: "**Return documents that contain the world 'battle'**"
    - Il risultato di una query non prevede la manipolazione dei dati, ma la **selezione** di alcuni di essi e **l'ordinamento per rilevanza**.
    - Proprietà dei dati, delle query e dei risultati differenti da quelle riscontrabili nei sistemi relazionali.

Dati relazionali (strutturati)

- Chiara **distinzione** tra schema e dati
- Basato sul concetto di **set**
- **Non ordinato** e non annidato
- Linguaggio di **query**

```
Select name, Count(distinct project), Sum(months)
From Person Natural Join Allocation
Where name like 'M%' AND Age > 40
Group by name
```

- **Modello booleano** (correttezza, completezza): le query esprimono requisiti precisi e ogni tupla della soluzione soddisfa tali requisiti; **una tupla è presente o non è presente nel risultato**.
- Aggiornamenti parziali e query
- Può essere rappresentato in **XML**

Dati semi-strutturati

- Mostrano le proprietà dei dati sia strutturati che non strutturati.
- Schema irregolare o parziale, costruito in retrospettiva, molto ampio e in rapida evoluzione (il cambiamento viene comunicato ai dati).
- Il regime non impone vincoli inappellabili.
- Le query riguardano anche lo schema.
- Dati rappresentati da liste (ordinati e nidificati)

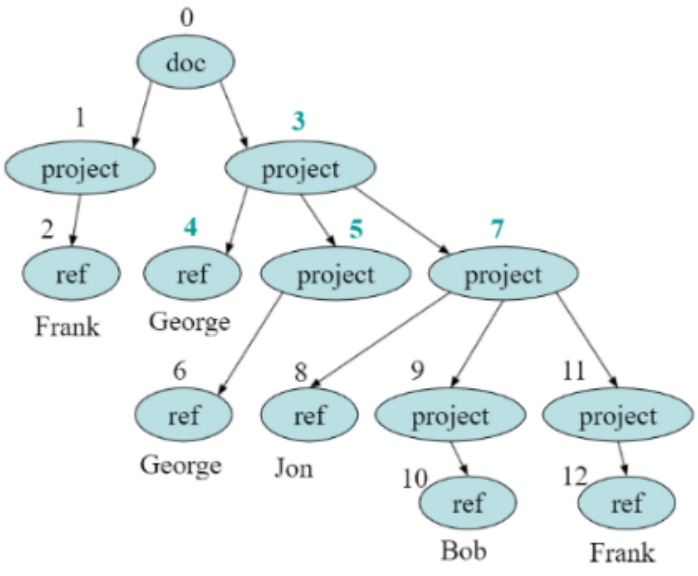
- Il formato principale per la rappresentazione dei dati è **XML**: creato per scambiare dati tra applicazioni e per rappresentare dati comprensibili per gli esseri umani.
- Rappresenta dati semi-strutturati, sfruttando la flessibilità e la capacità di indicare sia i dati, sia lo schema.

```
<doc>
    <project><ref>Frank</ref></project>
    <project>
        <ref>George</ref>
        <project><ref>George</ref></project>
        <project><ref>Jon</ref>
            <project><ref>Bob</ref></project>
            <project><ref>Frank</ref></project>
        </project>
    </project>
</doc>
```

- Il modello dei dati è più complesso del formato originale.
- Alcune query "ragionevoli" non possono essere scritte in SQL senza utilizzare la ricorsione, oppure possono essere inefficienti (richiedendo più tempo per l'accesso alla stessa tabella).

id	name
0	doc
1	project
2	ref
3	project
4	ref
5	project
6	ref
7	project
8	ref
9	project
10	ref
11	project
12	ref

id	child
0	1
0	3
1	2
3	4
3	5
3	7
5	6
7	8
7	9
7	11
9	10
11	12



XML

e**X**tensible **M**arkup **L**anguage deriva da SGML (**S**tandard **G**eneralized **M**arkup **L**anguage) e con entrambi è possibile definire linguaggi di markup specifici per diversi domini, come finanza o matematica.

Prolog: contiene informazioni utili per l'interpretazione del documento.

```
<?xml version="1.0"?>                          PROLOG
<?tex doctype[report] ?>
<doc isbn="2-266-04744-2">                     DOCUMENT
    <!-- editor is missing! -->                BODY
    <author>T. Harris</author>
    <title xml:lang="en">The silence of the lambs</title>
    <title xml:lang="fr">Le silence des agneaux</title>
    <comment>
        A book full of <i>suspance</i>.
    </comment>
    <price currency="euro">7</price>
</doc>
```

Dichiarazione tipo documento (Document **T**ype **D**efinition**,** DTD**)**: facoltativo

- **Valida** il contenuto del documenti XML.
- È costituito da **dichiarazioni di markup**, che determinano quali elementi possono essere inclusi nel documento, come possono essere utilizzati, quali sono i valori predefiniti degli attributi degli elementi e altri vincoli.
- **Vincola** un documento XML, ma è possibile specificare vincoli più complessi di quello consentito dalla DTD (ad esempio chiavi importate, vincoli di unicità o domini di elementi e attributi). Lo **Schema XML** permette di specificare questo tipo di vincoli, ed è quindi un'alternativa al DTD.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE greetings [
   <!ELEMENT greetings (#PCDATA)>
   <!ATTLIST greetings
          id      ID     #REQUIRED>
]>
<greetings id="0001">Hello,
world!</greetings>
```

DTD

```
<?xml version="1.0"?>
<!DOCTYPE greetings SYSTEM "hello.dtd">
<greetings>Hello, world!</greetings>
```

Document Body: il corpo del documento è costituito da un **elemento**, che a sua volta può contenere altri elementi nidificati nel suo contenuto e anche **commenti**.

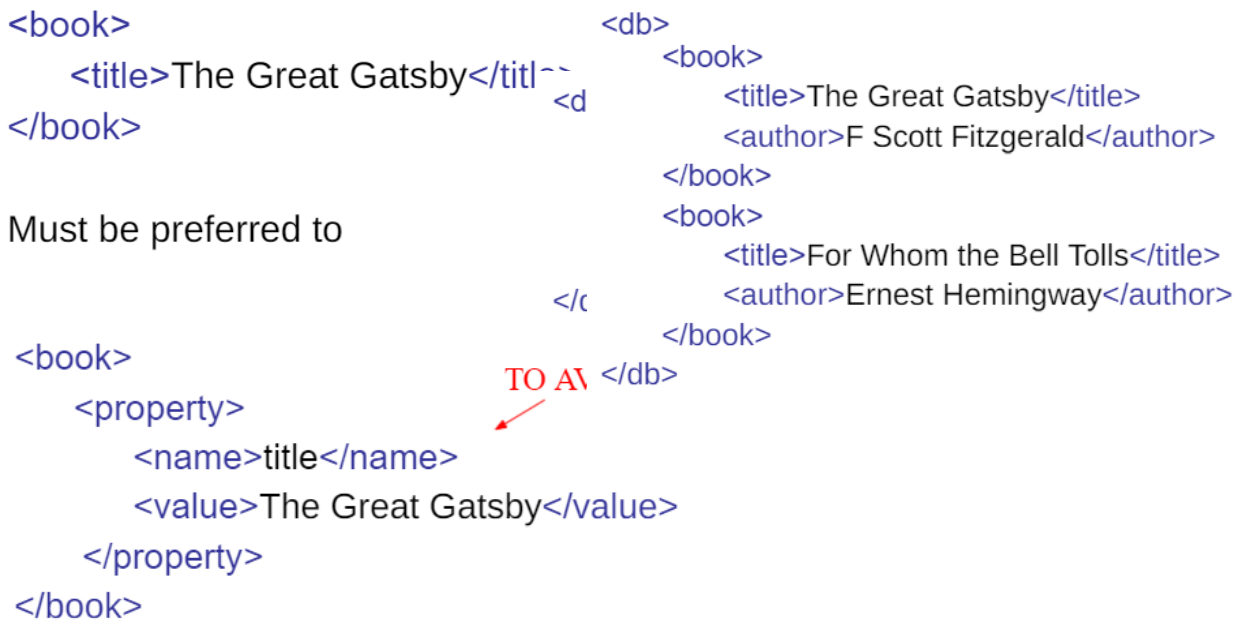
Elementi

- Ogni elemento deve essere contenuto tra un **tag** di apertura e un **/tag** di chiusura o con una forma abbreviata.

```
<name attributes_list>        (extended form)
   content...
<\name>

<name attributes_list />      (short form)
```

- I **nomi** degli elementi e degli attributi fanno distinzione tra maiuscole e minuscole.
- Gli elementi devono essere **nidificati** correttamente ed esiste **un solo elemento** che contiene tutti gli altri elementi.
- I **valori** degli attributi devono essere contenuti tra virgolette o doppie virgolette.

- Un elemento non può avere più di un **attributo** con lo stesso nome.
- I **dati** devono essere memorizzati nel contenuto degli elementi.



Dati relazionali e XML

XML viene utilizzato principalmente per **scambiare dati tra applicazioni** e rappresentare **dati semi-strutturati**. Quando i dati scambiati tra diverse applicazioni sono archiviati localmente in database relazionali, è necessario un "**ponte**" tra questi due formati (**SQL/XML** fornisce un linguaggio comune per **convertire** i dati relazionali in XML).

Per utilizzare una combinazione di dati relazionali e XML sono necessarie due funzioni principali:

1. **Estrazione XML** da una o più tabelle relazionali (semplice)
- **Mapping:** qualsiasi tipo di dato che può essere rappresentato in una tabella può essere rappresentato anche in XML.
    - Il **nome** della tabella diventa il nome del documento. Ogni riga è inclusa in un elemento **<row>** e ogni valore (colonna) è incluso in un elemento con il nome dell'attributo (i valori null sono rappresentati utilizzando l'attributo **xsi:nil = "true"**).
- Estrarre i dati utilizzando le tecnologie XML **(XQuery)**: trasforma i dati dalla tabella (SQL) in dati XML (SQL/XML); lo standard definisce uno schema XML con le definizioni di ogni tipo di dato in SQL e ogni elemento XML.

2. **Archiviazione XML** in una o più tabelle relazionali (complesso in generale).

- Colonne **Object-Relational** per archiviare interi frammenti XML in un **unico campo**: un intero documento XML viene visualizzato come un tipo di dati SQL e quindi memorizzato in un attributo (colonna) di una tabella.
- **Frammentare** i documenti, per i quali sono memorizzati elementi diversi in campi diversi, con alcune limitazioni, un documento XML può essere suddiviso in frammenti e archiviato in pezzi.

Il linguaggio SQL/XML: estensione di SQL (**S**tructured **Q**uery **L**anguage), comprende costruttori, routine, funzioni per supportare la manipolazione e l'archiviazione di XML in un database SQL.

Operatori

XMLELEMENT: permette di creare un elemento XML con argomenti

```
SELECT i.id, XMLELEMENT(
    NAME "emp",          → name of the element
    i.name ) AS result   → content
FROM EMPLOYEES i
```

Id	*result*
emp0001	<emp>John</emp>
emp0002	<emp>Jack</emp>
emp0003	<emp>Donald</emp>
emp0004	<emp>Samuel</emp>

- **Nome dell'elemento:** fornito esplicitamente utilizzando una costante.
- **Contenuto dell'elemento:** è possibile specificare più oggetti, sia elementi che stringhe di caratteri (operatore di concatenazione **||**).
- Elenco opzionale di **attributi**.

XMLATTRIBUTES: ogni parametro è inserito in un **attributo** che, se non dichiarato esplicitamente, prende il nome della colonna relazionale da cui è stato selezionato.

```
SELECT i.id, XMLELEMENT(
    NAME "emp",
    XMLATTRIBUTES(i.salary as "sal"),
    i.name || ' ' || i.surname ) AS result
FROM EMPLOYEES i
```

id	*result*
emp0001	<emp sal="20000">John Doe</emp>
emp0002	<emp sal="18000">Jack Black</emp>
emp0003	<emp sal="15000">Donald Mason</emp>
emp0004	<emp sal="15000">Samuel Wood</emp>

```
SELECT i.id,
   XMLELEMENT(
      NAME "emp",
      XMLFOREST(
         i.name,                  ← If this explicit name is missing, the name of the corresponding column is used instead.
         i.surname AS "co",
         i.department AS "dip")
   ) AS result
FROM EMPLOYEES i
```

XMLFOREST: produce rapidamente un **elenco** di elementi semplici (il comportamento dei suoi parametri è lo stesso di XMLATTRIBUTES).

XMLCONCAT: concatena i suoi argomenti, producendo una foresta di elementi (può concatenare anche elementi costruiti utilizzando XMLELEMENT).

```
SELECT i.id,
   XMLCONCAT(
      XMLELEMENT(
         NAME "co",
         i.surname),
      XMLELEMENT(
         NAME "dep",
         i.department)
   ) AS result
FROM EMPLOYEES i
```

```
SELECT XMLELEMENT(
      NAME "department",
      XMLATTRIBUTES(i.department AS "name"),
      XMLAGG(
         XMLELEMENT(
            NAME "emp",
            i.surname) )
   ) AS result
FROM EMPLOYEES i
GROUP BY department
```

XMLAGG: permette di **raggruppare** più tuple a seconda di uno o più attributi utilizzando GROUP BY

XMLGEN: specifica direttamente il **codice XML**, inserendo i dati tramite **variabili** (utilizzabili anche per i nomi degli elementi), espresse con "{" e "}".

```
SELECT XMLGEN(
      '<employee>
         <name>{$name}</name>
         <salary>{$sal}</salary>
      </employee>',
      i.name,
      i.salary AS "sal"
   ) AS result
FROM EMPLOYEES i
WHERE salary > 15000
```

Reference by name

Per calcolare il risultato:

1. Viene considerato solo **SQL** e viene calcolata una tabella come se fosse un **SELECT ***
2. Viene costruito il risultato, selezionando gli attributi richiesti **(Attr1, Attr2...)** e costruendo il codice XML PER OGNI TUPLA.

Linguaggio XQuery

Può essere utilizzato per accedere a dati espressi in XML in documenti strutturati e semi-strutturati e ha funzionalità simili a XSLT (e**X**tensible **S**tylesheet **L**anguage **T**ransformations); distingue tra maiuscole e minuscole

Modello Dati XQuery

Sequenze

XQuery opera sulle sequenze, che possono contenere **valori atomici** (come la stringa "ciao" o l'intero 3) e **nodi** (documenti e frammenti XML rappresentati come alberi).

Un'espressione XQuery riceve nessuna (nel caso di costruttori) o più sequenze e produce una sequenza **ordinata** [(1, 2) è diverso da (2, 1)] e **non nidificata** [((), 1, (2, 3)) è uguale a (1, 2, 3)]. Non c'è differenza tra un elemento e una sequenza con lo stesso elemento: (1) è uguale a 1.

Operatori: **,** (virgola), **to, union** (equivalente a **|**), **intersect**, **except**

- **Equivalenti:** (1, 2, 3); (1, (), (2, 3)); 1 (from 1 to 3); (1, from 2 to 3)
- **(A) union (A, B) -> (A, B)**
- **(A, B) intersect (B, C) -> (B)**
- **(A, B) except (B) -> (A)**

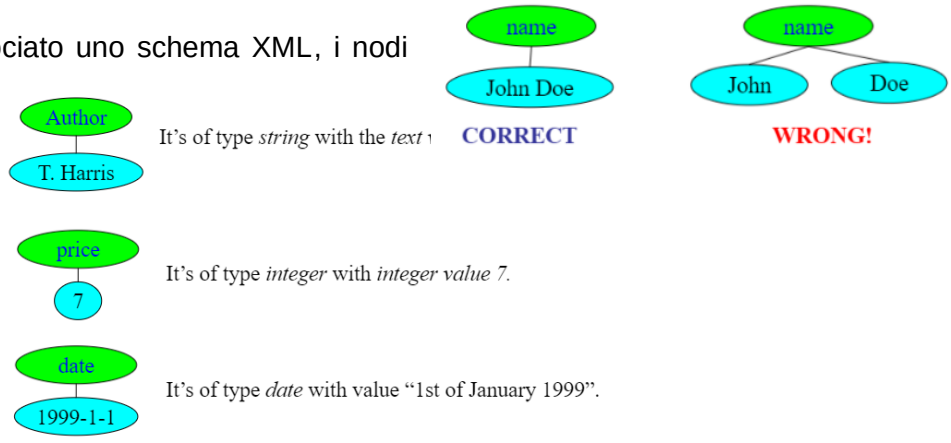
Nodi

Attributi: se vengono estratti utilizzando XQuery l'ordine non corrisponde necessariamente a quello in cui si trovano nel documento XML di origine (**non ordinati**), una volta estratti però, vengono inseriti in una sequenza, di cui mantengono l'ordine.

Text: il valore della stringa dei nodi dei tipi **document** ed **element** corrisponde alla concatenazione dei valori di testo di tutti i suoi discendenti testuali, nell'ordine in cui si trovano nel documento.

Due **nodi di testo non possono essere adiacenti**: un elemento <name>John Doe</name> è quindi rappresentato così

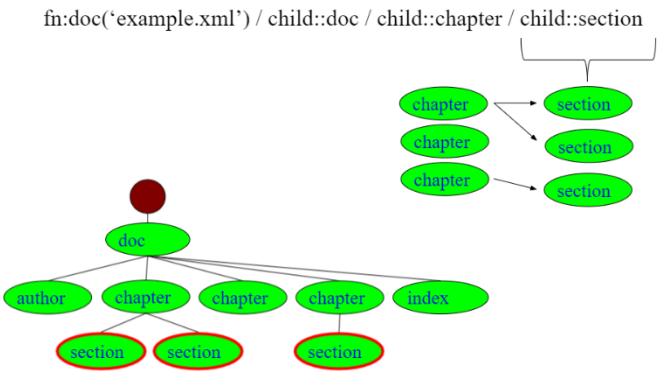
Se al documento è associato uno schema XML, i nodi



possono avere un tipo

Espressioni di percorso (Path Expressions)

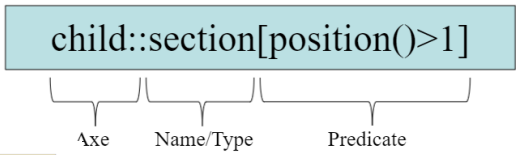
Utilizzate per estrarre valori da nodi e alberi XML e per verificarne le proprietà; consiste in una **serie di passaggi**, separati dal carattere */*.



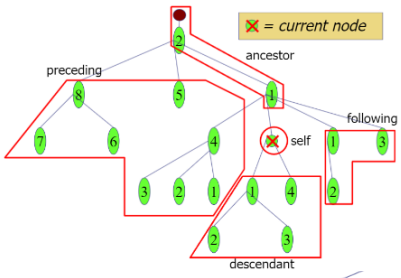
Ogni passaggio viene valutato in un **contesto**, ovvero una sequenza di nodi (con informazioni aggiuntive, ad esempio la posizione del nodo) e produce una **sequenza**.

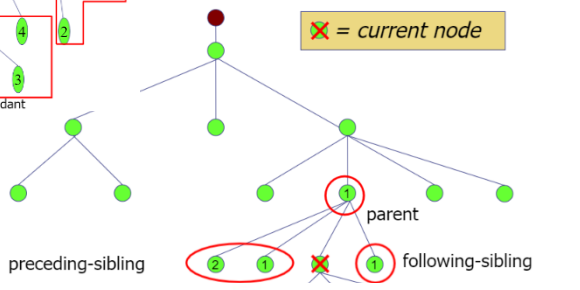
Il passaggio successivo viene **valutato** utilizzando come contesto la sequenza di nodi prodotta dal passaggio precedente.

Struttura a gradini: il passaggio di un'espressione di percorso può essere composto da tre parti principali:

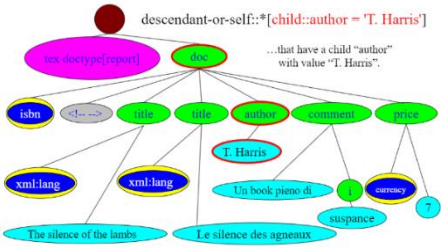


1. **Axe**: seleziona i nodi a seconda della sua posizione rispetto alla posizione del nodo di contesto.
    - self::
    - child::
    - parent::
    - ancestor::
    - descendant::
    - following-sibling:
    - preceding-sibling::
    - attribute::
    - descendant-or-self::
    - ancestor-or-self::





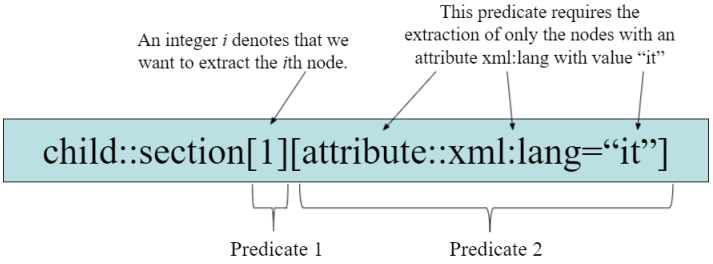
2. **Test** che **filtra** i nodi in base a:
    - **Nome:**
        - **child::section** restituisce solo gli elementi child con tag <section>.
        - **child::*** restituisce tutti gli elementi figlio.
        - **attribute::xml:lang** restituisce l'attributo xml:lang.



    - Oppure **Tipo:**
        - **descendant::node()** restituisce tutti i nodi discendenti.
        - **descendant::text()** restituisce tutti i nodi discendenti di tipo testo.
        - **descendant::element()** restituisce tutti i nodi di elementi discendenti.
    - O **entrambi: descendant::element(person, xs:decimal)** – elementi persona di tipo decimale.

3. **Predicate**: **filtra** ulteriormente i nodi in base a criteri più **generici**, ogni passaggio può terminare con uno o più predicati (in congiunzione con e), compresi tra “[” e “]”, per filtrare ulteriormente i nodi selezionati utilizzando assi e test su nomi/tipi.

Valutazione dei predicati



Il predicato ha valore **TRUE**:

- Se l'espressione restituisce un singolo **valore intero**
- Se la **posizione** del nodo soggetto nell'espressione valutata corrisponde al **valore intero**: **child::chapter[2]** restituisce solo il secondo child con il tag <chapter>
- Se il **primo** elemento è un **nodo**
    - **child::chapter[child::title]** restituisce tutti i figli con tag <chapter> che hanno almeno un figlio con tag <title>.
    - **child::chapter[attribute::xml:lang = "it"]** restituisce tutti i child con tag <chapter> che hanno un attributo xml:lang con valore "it"

Il predicato ha valore **FALSE** se l'espressione restituisce una **sequenza non vuota**

Espressioni di percorso completo: può iniziare con il carattere / (sequenza di input che contiene la **radice dell'albero**) oppure può iniziare con i caratteri // (sequenza di input che contiene **tutti i nodi** nel documento).

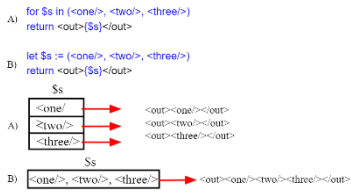
- **/descendant:figure[fn:position() = 42]:** seleziona la 42a cifra del documento.
- **/child::book/child::chapter[5]/child::section[2]:** seleziona la seconda sezione del quinto capitolo.
- **//self::chapter[child::title]:** seleziona tutti i capitoli che hanno almeno un figlio <title>.

Sintassi breve: per scrivere espressioni più compatte è possibile utilizzare alcune scorciatoie

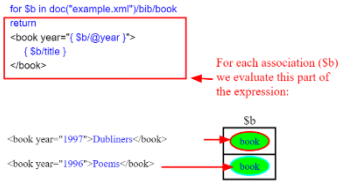
- **Omissione** del **child**::axe
    - **child::**section/**child::**paragraph -> section/paragraph
- **Sostituzione** dell'attributo **axe**:: con il carattere @
    - para[**attribute::**type="warning"] -> para[@type="warning"]
- **Sostituzione** di **descendant-or-self::node**() con doppia barra (**//**)
    - div**/descendant-or-self::node()/child::paragraph** -> div**//paragraph**
- **Sostituzione** di **self::node**() con un punto (**.**):
    - **self::node()/descendant-or-self::node()/child::para** -> **.//para**
- **Sostituzione** di **parent::node**() con due punti (**..**):
    - **parent:node()/child::section** -> **../section**

Espressioni FLWOR ("flower")

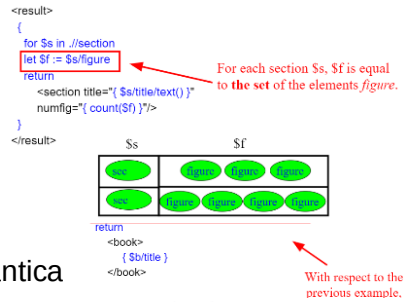
Simile a un'istruzione SQL Select-From-Where, ma definita in termini di **variabili** vincolanti e composta da cinque parti, alcune delle quali sono opzionali:



1. **Return:** crea il risultato dell'espressione FLWOR.
   - **For** e **Let** in XQuery creano un elenco con tutte le possibili associazioni (**tuple**)



2. **For:** associa una o più variabili alle espressioni.
   - **Iterazioni di elementi:** selezioniamo tutti i libri e poi **ogni libro** lo associamo alla variabile **$b**
3. **Let:** crea un alias dell'intero risultato di un'espressione.
   - **Espressioni con funzioni aggregate:** raggruppa elementi per contarli o per calcolare il loro valore medio, minimo o massimo se si tratta di numeri.
4. **Where:** filtra l'elenco delle associazioni in base a una condizione.
5. **Order by:** ordina l'elenco delle associazioni e può essere specificato insieme a diversi parametri per alterarne la semantica (**sorting**)




```
for $l in $books//book[price < 50]
order by $l/title
return $l
```

Qualsiasi espressione XQuery, può essere commentata con i simboli “(:” e “:)”: **(: Questo è un commento (: questo è un commento annidato... :) :)**

Join

Supponiamo di avere due file XML, uno descrive il programma di un concerto, l'altro descrive i compositori (il compositore ha un identificatore comune).

Vogliamo produrre un programma di proiezioni, che elenchi le canzoni, con l'autore e le date di nascita e morte.

```
<program>
  <song>
     <title>Ciaccona</title>
     <of>J.S.Bach</of>
     <date>1685-1750</date>
  </song>
  <song>
     <title>Polacca</title>
     <of>F.Chopin</of>
     <date>1810-1849</date>
  </song>
   ...
```

```
<program> {
for $comp in doc('comp.xml')/comp/composer,
    $song in doc('conc.xml')/concert/song
where $comp/id eq $song/author
return
   <song>
      <title>{$song/title}</title>
      <di>{$comp/name}</di>
      <date>{$comp/bio}</date>
   </song>
}</program>
```

Espressioni condizionali

I linguaggi di programmazione imperativi (come il C), forniscono un'espressione condizionale nella forma **if-then-else** e in XQuery possiamo anche usare costrutti di questo tipo.

```
if ($product1/price < $product2/price)
then $product2
else $product1
```

Operatori di confronto

Operatori che agiscono su **sequenze** composte da più elementi: **=, !=, <, <=, >, >=**

Per esempio, ‘**$book1/author = "Joyce“’** restituisce TRUE se **almeno uno dei nodi** selezionati “author” ha un valore di testo uguale a "Joyce"

Queste operazioni **non** sono **transitive**: (1, 2) = (2, 3) e (1, 2) != (2, 3) [ritornano TRUE]

XPath 2.0 introduce nuovi operatori per confrontare sequenze composte da un singolo elemento: **eq, ne, lt, le, gt, ge**

$book1/author eq "Joyce" è vero solo se è stato selezionato un **singolo nodo autore**, in caso contrario, viene segnalato un errore.

Operatori logici e aritmetici

- XQuery fornisce i seguenti operatori logici: **or, and**
- Una funzione standard per la negazione logica: **fn:not()**
- E operatori aritmetici: **+, -, * div, idiv** (divisione intera), **mod** (resto della divisione).

Espressioni con quantificatori

In XQuery, una variabile può essere associata a un singolo valore (**$price** può avere valore 102), ma spesso le variabili sono associate a insiemi di oggetti.

Ad esempio, l'espressione **for $lib in doc books.xml/books/book** potrebbe associare la variabile **$lib** a diversi elementi <book>.

Sono necessari operatori specifici per verificare le proprietà di insiemi di oggetti: **some $emp in //employee**

- **satisfies ($emp/salary > 13000):** vera **se almeno un** dipendente riceve uno stipendio maggiore di 13000.

Il risultato opposto si ottiene utilizzando il quantificatore **every $imp in //emplyee**

- **satisfies ($imp/salary > 13000):** vera se **tutti i** dipendenti ricevono uno stipendio maggiore di 13000.

Funzioni standard

Funzioni di ingresso: necessarie per ottenere il codice XML su qui fare la query.

- **fn:doc('bib.xml'):** utilizzato per accedere a un singolo documento
- **fn:collection('compositori'):** utilizzato per accedere a una sequenza di documenti

Funzioni su sequenze di nodi

- **fn:position():** posizione del nodo corrente
- **fn:last():** numero di nodi
- **fn:count($elements):** cardinalità della sequenza di nodi nell'argomento

Funzioni aggregate

$seq3 = (3, 4, 5)
count($seq3) returns 3
avg($seq3) returns 4
max($seq3) returns 5
min($seq3) returns 3
sum($seq3) returns 12

XQuery su DB2, Oracle e Server SQL

Documenti di esempio

Departments.xml

```
<depts>
   <dept deptno="10" dname="Administration"/>
   <dept deptno="20" dname="Marketing"/>
   <dept deptno="30" dname="Purchasing"/>
   <dept deptno="40" dname="Publishing"/>
   <dept deptno="50" dname="Transport"/>
</depts>
```

Employees.xml

```
<emps>
   <emp empno="1" deptno="10" ename="John" salary="21000"/>
   <emp empno="2" deptno="10" ename="Jack" salary="310000"/>
   <emp empno="3" deptno="20" ename="Jill" salary="100001"/>
   <emp empno="4" deptno="30" ename="Andrew" salary="50000"/>
   <emp empno="5" deptno="30" ename="Smith" salary="123000"/>
   <emp empno="6" deptno="30" ename="James" salary="34000"/>
   <emp empno="7" deptno="40" ename="Tom" salary="210000"/>
   <emp empno="8" deptno="40" ename="Derek" salary="223000"/>
   <emp empno="9" deptno="50" ename="Alice" salary="120000"/>
   <emp empno="10" deptno="50" ename="Sandra" salary="12000"/>
</emps>
```

XQuery su DB2

DB2 permette di inserire colonne di tipo **XML** in tabelle relazionali dove si possono inserire dati utilizzando i comandi INSERT abbinati alla funzione **XMLPARSE**, che prende come input una stringa e la converte in un frammento XML che DBMS può gestire (il database utilizza la codifica **UTF-8**).

Le query **XQuery** vengono inserite nella clausola SELECT delle query SQL, utilizzando la funzione **XMLQUERY** e restituendo frammenti XML.

È possibile utilizzare funzioni per combinare query relazionali e query XQuery: **XMLEXISTS** può essere utilizzato nella clausola WHERE e **XMLTABLE** è utilizzato nella clausola FROM.

DB2 fornisce il supporto completo per tutti gli axe dell'**espressione del percorso**, tutti i tipi di nodo e tutti i test richiesti dallo standard XPath.

- **/descendant-or-self::book[attribute::year>1992]:** seleziona tutti gli elementi **'book'** appartenenti all'axe **'descendant-or-self'** che hanno un **attribute** 'year' con valore maggiore di 1992
- **/child::bib/child::book/child::text():** seleziona tutti gli elementi **di tipo testo** figli di un elemento **'book'** che a sua volta è figlio di un elemento **'bib'**

Creare una tabella XML: basta specificare XML come tipo di dati di una colonna. Nel nostro caso vogliamo due tabelle, ognuna conterrà uno dei documenti visti in precedenza.

Inserire i dati XML nelle tabelle: utilizziamo **la funzione XMLPARSE** all'interno di una clausola

Database examples

```
CREATE TABLE employees (data XML)
CREATE TABLE departments (data XML)
```

INSERT, passando **una stringa che rappresenta il documento XML** che vogliamo inserire nella tabella. È possibile scegliere se **mantenere o meno lo spazio bianco**.

Insert example

```
INSERT INTO departments(data) VALUES
 (XMLPARSE
   (document cast
     ('<depts>
   <dept deptno="10" dname="Administration"/>
   ....
   <dept deptno="50" dname="Transport"/>
</depts>' as clob)
   preserve whitespace)
 )
```

Estraiamo i nomi di tutti i reparti e **inseriamoli** negli **elementi XML <deptName>**. Usiamo la funzione XMLQUERY, tramite **SELECT**, per inserire la query **XQuery** in un'espressione SQL.

XQuery 1

```
SELECT XMLQUERY
    ('for $d in $list//dept
    return <deptName>{ $d/@dname }</deptName>'
    passing dtable.data as "list")
FROM departments dtable
```

La clausola **passing** viene utilizzata per specificare su quale frammento XML stiamo lavorando. In questo caso **passiamo il contenuto della tabella dei reparti come "list"**, in modo da **creare una variabile $list** che potremo utilizzare all'interno della query per fare riferimento al frammento XML che abbiamo precedentemente inserito nella tabella.

Nel **SELECT** vogliamo estrarre **i nomi e gli stipendi di tutti i dipendenti** con **stipendio superiore a 50.000** tramite il comando **where**. Inoltre, vogliamo che **i risultati vengano inseriti negli elementi <empSalary>** e **ordinati per stipendio** tramite il comando **order by**.

XQuery 2

```
SELECT XMLQUERY
    ('for $e in $list//emp
    where $e/@salary > 50000
    order by $e/@salary
    return <empSalary>{ $e/@ename }{$e/@salary}</empSalary>'
    passing etable.data as "list")
FROM employees etable
```

Nel **SELECT**, tramite gli operatori **let e avg**, vogliamo estrarre **per ogni dipendente** il nome, lo stipendio e **lo stipendio medio del dipartimento in cui lavora**.

La clausola let verrà eseguita ad ogni ciclo for, calcolando per ogni <emp> la media del valore di @salary di tutti gli elementi <emp> con lo stesso @deptno.

XQuery 3

```
SELECT XMLQUERY
    ('for $e in $list//emp
    let $avgsal := avg($list//emp[@deptno = $e/@deptno]/@salary)
    return
<averages>{$e/@ename}{$e/@salary}{$avgsal}</averages>'
    passing etable.data as "list")
FROM employees etable
```

Creiamo attraverso gli **operatori condizionali** un elemento <salaries> che è **vuoto se l'impiegato John guadagna più dell'impiegato Smith** e **che contiene lo stipendio di Smith altrimenti**.

Tramite l'**operatore union**, estraiamo i nomi di tutti i dipendenti **che lavorano nel reparto con**

XQuery 4

```
SELECT XMLQUERY
    ('let $john := $list//emp[@ename="John"]
```

XQuery 5

```
SELECT XMLQUERY
    ('let $emps30 := $list//emp[@deptno = 30]
    let $empspoor := $list//emp[@salary < 100000]
    for $e in ($emps30 union $empspoor)
    return <empName>{$e/@ename}</empName>'
    passing etable.data as "list")
FROM employees etable
```

@deptno 30 e di quelli che guadagnano meno di 100.000 (**let** è al di fuori della clausola **for**, quindi viene calcolata solo una volta).

Attraverso l'**operatore some**, vogliamo **estrarre i nomi e gli stipendi di tutti i dipendenti**, ma **solo se c'è almeno un dipendente che ha uno stipendio maggiore di 100000**.

Selezioniamo il nome di tutti i dipendenti **che lavorano nel reparto denominato "Purchasing"**. Affinché la **join** funzioni, utilizziamo la clausola passing per trasferire sia il **contenuto dei dipendenti** sia il **contenuto dei reparti**.

XQuery 7

```
SELECT XMLQUERY
```

XQuery 8

```
SELECT XMLQUERY
    ('for $d in $dlist//dept
      let $sumsalary :=
          sum($elist//emp[@deptno = $d/@deptno]/@salary)
      return <deptCost>{$d/@dname} {$sumsalary}</deptCost>'
     passing etable.data as "elist", dtable.data as "dlist")

FROM employees etable, departments dtable
```

```
FROM employees etable
```

Attraverso **join e sum**, vogliamo selezionare **per ogni reparto**, il suo nome e **la somma degli stipendi dei dipendenti che vi lavorano**.

Tramite **SELECT**, vogliamo selezionare per ogni dipartimento **il numero dei suoi dipendenti** e **il loro stipendio medio** e vogliamo che i risultati siano **ordinati in base al costo totale (la somma degli stipendi) di ciascun reparto** e che **solo i reparti con più di un dipendente debbano essere inclusi nella risposta**.

XQuery 9

```
SELECT XMLQUERY
    ('for $d in $dlist//dept
    let $emps := $elist//emp[@deptno = $d/@deptno]
    where count($emps) > 1
    order by sum($emps/@salary) descending
    return
        <big-dept>
        {$d/@dname}
        <headcount> {count($emps)}</headcount>
        <avgsal>{avg($emps/@salary)}</avgsal>
        </big-dept>'
  passing etable.data as "elist", dtable.data as "dlist")

FROM employees etable, departments dtable
```

XQuery in Oracle DB

Funzionalità non supportate

- **Codifica della versione:** non è possibile specificare la codifica utilizzata all'interno di un'espressione XQuery.
- **xml:id:** viene generato un errore.
- **xs:duration:** in alternativa è possibile utilizzare xs:yearMonthDuration o xs:DayTimeDuration.
- **Funzionalità di convalida dello schema** (opzionale nello standard).
- **Caratteristica del modulo** (opzionale di serie).
- **Oracle DB e XPath 2.0:** non puoi utilizzare le funzioni standard per la definizione delle espressioni regolari, ma devi usare quelle integrate all'interno del DBMS
    - Le funzioni **fn:id** e **fn:idref** non sono supportate.
    - La funzione **fn:collection** senza argomenti non è supportata.

Creazione di una tabella XML

Come in DB2, in Oracle DB possiamo **inserire colonne di tipo XML nelle tabelle relazionali**, utilizzando la parola chiave **XML Type**. È inoltre possibile **creare tabelle che siano esse stesse di tipo XML**.

Examples:

```
Creation of an XML table:
CREATE TABLE employees OF XMLType

Creation of a relational table with an XML column:
CREATE TABLE departments
    (id NUMBER(4),
    data XMLType)
```

Due opzioni per l'implementazione di XML Type

1. **LOB (Large Object):** come una semplice stringa.
2. **Storage Strutturato**: il DBMS costruirà automaticamente le tabelle e le viste che seguono lo schema XML del documento e le utilizzerà per contenere i singoli nodi del documento, mantenendo la conformità al suo DOM (**D**ocument **O**bject **M**odel).

La scelta dell'implementazione non ha alcuna conseguenza sui **metodi** che l'utente utilizzerà per gestire i Tipi XML, ma ha conseguenze sull'**esecuzione** delle query. Non è possibile stabilire quale modalità di conservazione sia "superiore": per ottenere le migliori prestazioni bisogna provare caso per caso.

Inserire dati XML all'interno di una tabella di colonna: utilizziamo la funzione **XML Type** all'interno di un comando INSERT, passando **il frammento XML che vogliamo inserire**.

Esempio:

```
INSERT INTO departments VALUES
 (XMLType
  ('<depts>
   <dept deptno="10" dname="Administration"/>
   <dept deptno="20" dname="Marketing"/>
   <dept deptno="30" dname="Purchasing"/>
   <dept deptno="40" dname="Publishing"/>
   <dept deptno="50" dname="Transport"/>
```

Example of XQuery

```
SELECT XMLQUERY
    ('for $e in //emp
    where $e/@salary > 50000
    order by $e/@salary
    return <empSalary>{ $e/@ename }{$e/@salary}</empSalary>'
    passing etable.object_value
    returning content)
FROM employees etable
```

Mostriamo una **query** che abbiamo già eseguito in DB2 dove si vuole estrarre **il nome e lo stipendio** di tutti i dipendenti con stipendio superiore a 50000, ordinati dal meno pagato al più pagato. A parte alcune **differenze** nella clausola di **passaggio** e nell'aggiunta della clausola di **restituzione del contenuto**, il sistema di query è lo stesso di DB2.

Mostriamo una query con una **join** tra due tabelle XML: anche in questo caso la sintassi è simile a quella utilizzata in DB2. Lo scopo della query è estrarre tutti i dipendenti che lavorano nel reparto denominato "Purchasing".

Example of XQuery with join

```
SELECT XMLQUERY
    ('for $e in $elist//emp
    let $d := $dlist//dept[@dname = "Purchasing"]
    where $e/@deptno = $d/@deptno
    return $e'
    passing etable.object_value as "elist",
            dtable.object_value as "dlist"
  returning content)

FROM employees etable, departments dtable
```

Se non si vogliono inserire dati nelle tabelle, è possibile utilizzare la **funzione doc()** per **lanciare query direttamente sui file XML**. Vediamo un esempio della stessa query appena vista, ma applicata al documento **employees.xml**.

La tabella **DUAL** è presente di default in tutti i database **Oracle**, ed è vuota: serve solo a compensare il fatto che la clausola FROM è obbligatoria in tutte le query SQL, anche nel caso in cui i dati siano presi con altri mezzi (come in questo caso, la funzione **doc()**).

```
SELECT XMLQUERY
    ('for $e in doc("employees.xml")//emp
    where $e/@salary > 50000
    order by $e/@salary
    return <empSalary>{ $e/@ename }{$e/@salary}</empSalary>'
    returning content)
FROM DUAL
```

XQuery in SQL Server 2012

Sequenze omogenee: possono essere composte solo da **nodi** o **valori atomici**. Non è possibile utilizzare gli operatori **union, intersect, except** per combinare sequenze formate da nodi.

Espressioni di percorso: è possibile utilizzare gli axe **child, descendant, parent, attribute, self, descendant-or-self** e sono supportati i tipi di nodo comment(), node(), processing-instruction(), text().

Espressioni FLWOR (completamente supportate)**:** le espressioni assegnate a una variabile tramite la clausola LET verranno inserite nella query **ogni volta che la variabile è menzionata in essa**. Ciò significa che l'espressione non verrà eseguita una sola volta, ma **verrà ricalcolata ogni volta che c'è un riferimento a quella variabile**.

Altri operatori supportati sono quelli di confronto, logici (**ad eccezione di idiv e not**), i costrutti condizionali **if-then-else**, tutte le **funzioni di aggregazione** e le espressioni con quantificatori **some/satisfies** ed **every/satisfies**.

SQL Server

Per gestire i dati **XML** dobbiamo prima **creare una tabella** che li contenga e inserire una colonna XML al suo interno.

Example of tables

```
CREATE TABLE employees(
    col1 int primary key,
    data XML)

CREATE TABLE departments(
    col1 int primary key,
    data XML)
```

Per **inserire i dati nella tabella** appena creata è sufficiente inserire nell'operazione INSERT una stringa contenente il frammento **XML**.

Example of INSERT

```
INSERT INTO departments VALUES
 (1,
  ('<depts>
   <dept deptno="10" dname="Administration"/>
   <dept deptno="20" dname="Marketing"/>
   <dept deptno="30" dname="Purchasing"/>
   <dept deptno="40" dname="Publishing"/>
   <dept deptno="50" dname="Transport"/>
</depts>'))
```

Mostriamo la stessa query che abbiamo fatto per DB2 e OracleDB: vogliamo estrarre il nome e lo stipendio dei dipendenti che guadagnano più di 50.000, classificati per stipendio. A parte alcune **differenze** nel passaggio dei dati (invece di utilizzare una clausola di passing, applichiamo direttamente **la funzione query() alla colonna XML "data"** contenuta **nella tabella emplyee**), la sintassi è la stessa di sempre.

XQuery in Oracle Berkeley DB XML

Example of XQuery:

```
SELECT data.query
    ('for $e in //emp
    where $e/@salary > 50000
    order by $e/@salary
    return <employee>{ $e/@ename }{$e/@salary}</employee>')
AS result
FROM employees
```

Oracle Berkeley non è un vero DBMS, ma **un insieme di librerie** che possono essere utilizzate all'interno dei linguaggi di programmazione più comuni, per gestire database XML (non fornisce supporto ai database relazionali). È possibile operare anche da un'interfaccia terminale testuale che valuta direttamente i comandi.

Operazioni principali

Creare un contenitore: due diverse modalità di archiviazione dei dati:

1. **Per documento:** i documenti vengono archiviati esattamente come sono stati consegnati al sistema
2. **Per nodi:** i documenti vengono scomposti nei nodi di cui sono composti, che poi vengono archiviati separatamente; questo consente di modificare il documento in futuro, oltre a garantire prestazioni migliori.

Example (through the console):

```
dbxml> createContainer esempio.dbxml
```

Una volta creato un **contenitore** (paragonabile alle tabelle dei DBMS relazionali), è possibile **inserirvi documenti** XML con l'operazione: **"putDocument <namePrefix> <string> [f|s|q)"**

Examples (through the console):

```
dbxml> putDocument book1
'<book><title>Title1</title><author>Author1</author><pages>
100</pages></book>' s

dbxml> putDocument book1'Document.xml' f
```

- Inserire "f" come terzo parametro denota che **la stringa contiene il percorso del file XML** che vogliamo inserire
- Inserire "s" come terzo parametro denota che **la stringa contiene il frammento XML** che vogliamo passare

È possibile **vincolare** uno schema ad un documento XML: saranno consentiti solo i documenti XML che soddisfano il **vincolo** dello schema. È anche possibile specificare vincoli diversi per documenti diversi, all'interno dello stesso contenitore.

È possibile creare **indici** che aumentano le prestazioni.

Per **recuperare** i documenti inseriti in un contenitore, vengono utilizzate le **query XQuery**. **Oracle Berkley** eseguirà una ricerca su tutti i documenti contenuti nel contenitore.

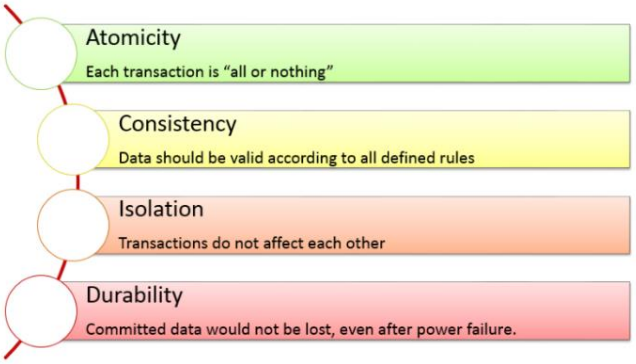
Examples (through the console):

```
Retrieve the whole content of the collection:
dbxml> query 'collection("esempio.dbxml")'

Retrieve the title of all the books that have John as
author:
dbxml> query
'collection("esempio.dbxml")/book[author="John"]/title'

Retrieve the title of all the books with a price greater
than 100:
dbxml> query
'for $book in collection("esempio.dbxml")
where $book/price > 100
return $book/title'
```

NoSQL

DBMS relazionale: Proprietà

- **Solide fondamenta**
- **Altamente strutturato:** righe, colonne, tipi di dati
- **Linguaggio di query strutturato**
- **Proprietà ACID (A**tomicity **C**onsistency **I**solation **D**urability**)**
- **Join:** nuove visualizzazioni dalle relazioni



Le cinque **V** di **Big Data**:

1. **V**olume: dati distribuiti per volume
2. **V**elocità: elevata disponibilità per la velocità
3. **V**arietà
4. **V**ariabilità: schema flessibile o nessuno schema
5. **V**eridicità

Debolezze

- **Join:** non scalabili.
- **Transazioni:** le operazioni di lettura e scrittura sono lente a causa del blocco delle risorse.
- **Definizioni fisse (schema):** difficile lavorare con dati altamente variabili.
- **Integrazione dei documenti:** difficile creare report basati su dati strutturati e non strutturati.

NoSQL: perché, cosa e quando

Nel 2004 Google e Amazon hanno creato i propri database **non relazionali** (senza chiavi primarie/esterne, JOIN o calcoli relazionali di alcun tipo), progettati per scalare fino a **petabyte** di dati su **migliaia di macchine** (Google Big Table e Dynamo). Nel 2008 Facebook rilascia il proprio database non relazionale, con un design simile a Google Big Table.

Altri motivi importanti: le licenze SQL **costano** per centinaia di migliaia di macchine.

#NoSQL era un hashtag di Twitter e fa riferimento a "non SQL", "non relazionale", ma non non esiste **una definizione rigorosa** per NoSQL (ci sono attualmente più di 225 sistemi di database NoSQL).

NoSQL DBMSS

Proprietà dei sistemi NoSQL

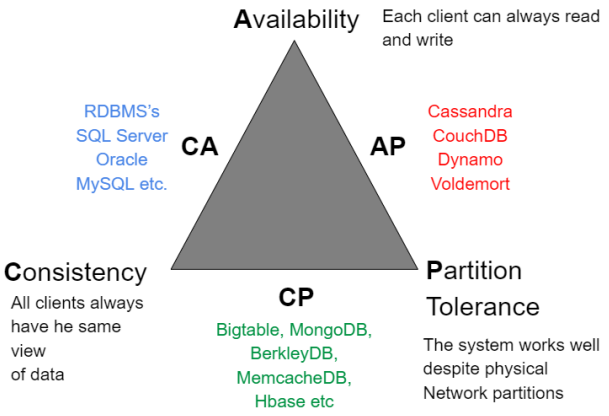
- **BASE**: la disponibilità è la cosa più importante, non seguono le proprietà ACID e sono disposti a sacrificare altre proprietà per questo.
  - Consistenza più debole.
  - Più potenza
  - Semplice e veloce.
  - Ottimista.
- **Privi di schema**: hanno un modello di dati non relazionale (ad es. valore-chiave, documento, grafico).
- **Fondamentalmente disponibili:** il sistema garantisce la disponibilità dei dati per quanto riguarda il **Teorema CAP (triangolo)**. Ci sarà una risposta a qualsiasi richiesta, ma tale risposta potrebbe essere 'fallimento' e i dati potrebbero essere in uno stato incoerente o modificati.
- **Stato morbido:** lo stato del sistema potrebbe sempre cambiare nel tempo. Anche durante i periodi senza input ci possono essere cambiamenti in corso a causa dell'eventuale consistenza.
- **Consistenza finale:** il sistema alla fine diventerà coerente una volta che smette di ricevere input.
  - I dati si propagheranno ovunque prima o poi.

- Il sistema continuerà a ricevere input nel frattempo.
- Il sistema non verifica la consistenza di ogni transazione prima di passare a quella successiva.

Il compromesso CAP (triangolo)

1. **Coerenza:** un sistema funziona completamente o no (stessa idea di ACID).
2. **Disponibilità**: il servizio o sistema fornito è disponibile quando richiesto? Ogni richiesta ottiene una risposta?
3. **Tolleranza (Partition Tolerance):** un sistema continua a funzionare anche in circostanze di perdita di dati o guasto del sistema. Un errore di un singolo nodo non causa il collasso dell'intero sistema.

In sistemi su larga scala, distribuiti e non relazionali, hanno bisogno di disponibilità e tolleranza delle partizioni; quindi, la coerenza ne risente e **l'ACID crolla**.



Solo **due proprietà su tre** possono essere soddisfatte nello stesso modello di dati!

I sistemi NoSQL sono privi di schema

Nei Database **relazionali**:

- Non puoi aggiungere un **record** che non si adatta allo schema.
- Devi aggiungere **NULL** agli elementi inutilizzati in una riga.
- Considera i **tipi** di dati: non puoi aggiungere una stringa a un campo int.
- Non puoi aggiungere più **elementi** in un campo (dovresti creare un'altra tabella: chiave primaria, chiave esterna, join, normalizzazione, ...).

Nei database **NoSQL**: **non c'è uno schema**, **non ci sono celle inutilizzate** e **i tipi di dati sono impliciti**. La maggior parte delle considerazioni si esegue al **livello dell'applicazione**.

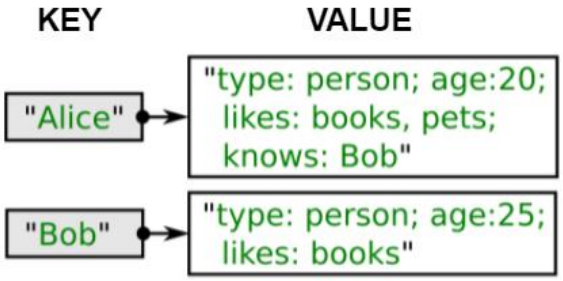
- **Aggregazione:** si raccolgono tutti gli elementi in modo aggregato (documento, il termine deriva da **D**omain **D**riven **D**esign).
- **Aggregato:** cluster di oggetti di dominio che possono essere trattati come una singola unità. Per esempio, un documento web è un aggregato con titolo, corpo, immagine all'interno del corpo con la sua didascalia, ecc. Gli aggregati sono **l'elemento base** del **trasferimento** dell'**archiviazione** dei dati (si richiede di caricare o salvare interi aggregati). Le transazioni non devono superare i confini aggregati. Questo meccanismo riduce le operazioni di join a un livello minimo (le cose correlate sono già insieme).
- **Persistenza poliglotta:** un approccio ibrido alla persistenza dei dati. Diversi modelli di dati sono progettati per risolvere problemi diversi: diversi tipi di dati e diverse esigenze di accesso (temporaneo, velocissimo, storico). L'utilizzo di un unico motore di database per tutti i requisiti porta a soluzioni non performanti.

Key-Value: si utilizza una tabella hash **per memorizzarlo** in memoria o nella persistenza del disco (gestisce enormi carichi di dati e diverse versioni)

- **Key:** stringa
- **Value:** qualsiasi formato di dati, in genere un blocco di dati non interpretato (JSON o un documento binario). Viene memorizzato come "**blob**": non c'è bisogno di sapere cosa c'è dentro o da quanto tempo; l'applicazione è responsabile della comprensione dei dati.

Quattro **operazioni** di base:

1. Insert(key,value)
2. Fetch(key)
3. Update(key)
4. Delete(key)



Modello dati del documento (XML, JSON, testo o BLOB binario)**:** qualsiasi struttura ad albero può essere rappresentata come un **documento** XML o JSON (simile al key-value, tranne che il valore è un documento). Ecco alcuni **vantaggi:**

- **Modello di dati ricco** (rappresentazione naturale dei dati)**:** incorpora dati correlati in documenti secondari e array; supporta indici e query avanzate su qualsiasi elemento.
- **Dati aggregati in un'unica struttura** (pre-JOINed)**:** programmazione semplice e prestazioni fornite su larga scala.
- **Schema dinamico:** i modelli di dati possono evolversi facilmente e adattarsi rapidamente ai cambiamenti (metodologia agile).

Modello relazionale vs modello documentale: operazioni

Applicazione	Azione Relazionale	Azione Modello di documento
Crea record prodotto	INSERT a (n) tabelle (descrizione prodotto, prezzo, produttore, ecc.)	insert() in 1 documento con sotto-documenti, array
Visualizza la registrazione del prodotto	SELECT and JOIN (n) tabelle di prodotto	find() documento aggregato
Aggiungi recensione prodotto	INSERT alla tabella di "revisione" chiave esterna al record del prodotto	insert() per "rivedere" la raccolta, riferimento al documento del prodotto

Modello di dati a colonne: archiviano le tabelle per **colonne** di dati anziché per righe.

Colonnare: estensione alle strutture delle tabelle tradizionali.

- Supporta set variabili di colonne (famiglie di colonne) ed è ottimizzato per operazioni a livello di colonna (come conteggio, somma e media).
- Più valori (colonne) per chiave.
- La colonna è l'istanza di dati più piccola.
- È una tupla che contiene un nome, un valore e un timestamp.

ID	Last	First	Bonus
1	Doe	John	8000
2	Smith	Jane	4000
3	Beck	Sam	1000

ROW-BASED	COLUMN-BASED
1,Doe,John,8000;2,Smith,Jane ,4000;3,Beck,Sam,1000;	1,2,3;Doe,Smith,Beck;John ,Jane,Sam;8000,4000,1000;

Modello dati grafico: vengono usati algoritmi grafici, la struttura dei dati è basata sulla teoria dei grafi (nodi, relazioni, proprietà).

- Scala verticalmente, nessun raggruppamento.
- Transazioni.
- ACID.

Recupero delle Informazioni (**I**nformation **R**etrieval, **IR**)

Consiste nel **trovare materiale** di **natura non strutturata** che soddisfa un **bisogno di informazioni** all'interno di **grandi raccolte**. Viene usata ad esempio per la **ricerca sul web**, e-mail/social media, sul tuo laptop, ecc.

Assunzioni base

- **Documento** (pagine web, e-mail, libri, notizie, articoli accademici, messaggi di testo, Word, PowerPoint, PDF, post su forum, brevetti, ecc.)
    - **Proprietà** comuni: contenuto di testo significativo e per lo più **non strutturato** (esempio di struttura: titolo, autore, data dell'articolo)
- **Collezione:** una serie di documenti (supponiamo per il momento che sia una raccolta statica, non ipertestuale)
- **Query:** insieme di parole chiave per esprimere un bisogno informativo
- **Obiettivo:** recuperare documenti con informazioni rilevanti per le esigenze dell'utente e aiutare l'utente a completare un'attività

Recupero dati vs IR

Un sistema di Data Retrieval (DBMS) si occupa di dati di una struttura **ben definita (schema di database)**

Un sistema di Information Retrieval si occupa di testi scritti in **linguaggio naturale**, spesso non strutturati e ambigui **(documenti di testo)**

Registro del database	Documento di testo
Il contenuto è strutturato in campi digitati	Il contenuto non è strutturato
Valori coerenti con i loro tipi di dati definiti	Può contenere qualsiasi tipo di dato (codici, date, prezzi) tutto in formato testo
I dati sono archiviati in formati predefiniti	Le informazioni sono espresse in linguaggio naturale

Un sistema di recupero dati (DBMS) recupera i dati utilizzando un **linguaggio di query** formalmente definito (ad es. SQL, algebra relazionale, ecc.)

Un sistema di Information Retrieval recupera documenti tramite un insieme di parole chiave **(query)** espresse in linguaggio naturale

Linguaggio di query	Linguaggio di ricerca
Basato su una grammatica formale	Basato su parole chiave del linguaggio naturale
La risposta non dipende dall'implementazione del sistema	Può essere interpretato in modo diverso dai diversi sistemi IR
Produce come risposta un insieme prevedibile di record che sono veri sotto la query	Produce un elenco classificato di risultati in base alla pertinenza della query

Testo a confronto

Confrontare il testo della **query** con il testo del **documento** e determinare quale sia una buona **corrispondenza** è il **problema** principale del recupero delle informazioni.

La **corrispondenza esatta** delle parole non è sufficiente, perché ci sono tanti modi diversi per scrivere la stessa cosa in un "linguaggio naturale". È necessaria una nozione di **pertinenza** invece di una corrispondenza esatta.

Modelli di recupero

1. **Modelli classici:** recupero booleano, modello spaziale vettoriale
2. **Modelli probabilistici:** BM25, modelli linguistici
3. **Combinazione di prove:** reti di inferenza, imparare a classificarsi

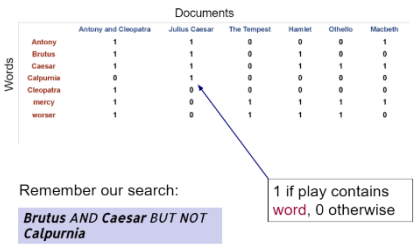
1. Modello classico: recupero booleano

Il modello di recupero booleano è il più semplice ed è in grado di creare query che utilizzano AND, OR e NOT per combinare i termini di ricerca

- Visualizza ogni documento come **un insieme di parole (Bag-of-words)**
- È preciso: il documento **corrisponde alla condizione o no**.

Principale strumento di ricerca commerciale fino ai primi anni '90. Molti sistemi di ricerca che usi ancora sono booleani: e-mail, catalogo della libreria, ecc.

Quali opere di Shakespeare contengono le parole Bruto E Cesare MA NON Calpurnia? **Lento** (per molti dati) e NON Calpurnia **non** è **banale**. Abbiamo bisogno di un modo per rappresentare i documenti come **insiemi di parole**



Vettori di incidenza

"Bruto E **Cesare** MA NON **Calpurnia":** abbiamo un vettore 0/1 per ogni parola.

Risultato: prendi i vettori per Bruto, Cesare e Calpurnia → AND bit per bit: 110100 AND 110111 AND 101111 = **100100**

Answer:	1 Antony and Cleopatra	0 Julius Caesar	0 The Tempest	1 Hamlet	0 Othello	0 Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

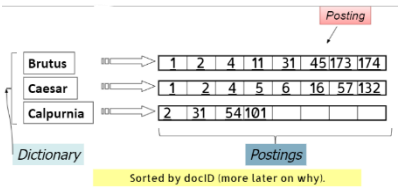
Words

Dimensione matrice

1. Consideriamo N = 1 milione di documenti, ciascuno con circa 1000 parole.
2. Media 6 byte/parola inclusi spazi/punteggiatura (6GB di dati nei documenti).
3. Supponiamo che ci siano M = 500K termini **distinti** tra questi.
4. La matrice M x N = 500K X 1M ha mezzo trilione di 0 e 1.
5. Ma non ha più di un miliardo di 1 (matrice **estremamente scarsa**).
6. **Registriamo solo le posizioni 1.**

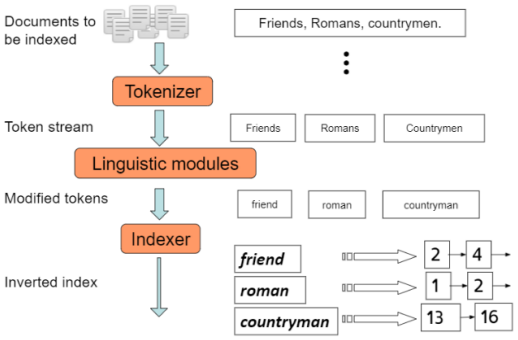
Indice invertito

Per ogni termine t, dobbiamo memorizzare un elenco di tutti i documenti che contengono t (identificare ogni documento con un docID, un numero di serie del documento).



Se usassimo array di dimensioni fisse, molti di questi sarebbero per lo più vuoti (**spreco di memoria**) e la raccolta non dovrebbe cambiare.

Abbiamo bisogno di una struttura più dinamica e migliore su disco, come le **liste di annunci (Posting-lists)** di dimensioni variabili: nella memoria principale, è possibile utilizzare elenchi collegati o array di lunghezza variabile (**compromessi** in termini di dimensioni e facilità di inserimento).



Pre-elaborazione del testo

Tokenization: taglia la sequenza di caratteri in simboli di parole, sequenza di coppie (Token modificato, ID documento)

Normalization: mappa testo e termini della query nella forma "normale" e il risultato è un **termine,** parola normalizzata nel nostro dizionario del sistema IR

- Definiamo implicitamente classi di equivalenza di termini, **eliminando** i punti e i trattini per formare un termine: **U.S.A., USA → USA; anti-discriminatorio, antidiscriminatorio → antidiscriminatorio**
- **Contenitore pieghevole:** riduci tutte le lettere in minuscolo tranne per le lettere maiuscole a metà frase, ad esempio: General Motors, Fed = fed, VELA = vela (spesso è meglio scrivere tutto in **minuscolo**).
- Esempio di Google di lunga data: **[risolto nel 2011...]**
    - **Query**: C.A.T.
    - **Risultato**: numero 1 è "cat" non per Caterpillar Inc.

Lemmatization: riduce le forme flessive/varianti alla forma base "corretta" del lemma del dizionario, ad esempio:

- am, are, is → be
- car, cars, car's, cars' → car
- the boy's cars are different colors → the boy car be different color

Stemming: riduci i termini alle loro "radici" prima dell'indicizzazione con un taglio postfisso grezzo che dipendente dalla lingua (automate(s), automatic, automation ridotti ad automat).

Stop words: con una stop list, escludi completamente dal dizionario le parole più comuni (o no) e che hanno poco contenuto semantico (il, a, i, di, the, a, and, to, be). Ma la tendenza recente è lontana dal fare questo:

- Buone tecniche di **compressione** significano che lo spazio per l'inclusione di parole non significative in un sistema è molto piccolo.
- Buone tecniche di **ottimizzazione** delle query significano che paghi poco al momento della query per includere le parole non significative.
- Ti servono per:
    - **Query di frase**: "Re di Danimarca"
    - **Vari titoli di canzoni**, ecc.: "Let it be", "To be or not to be"
    - **Query "relazionali":** "voli per Londra"

Dizionario: lessico del corpus indicizzato, ad esempio, tutte le parole sul web, tutti i nomi, acronimi, ecc. (compresi gli errori di ortografia)

- Specifico della lingua: **Webster** (inglese); **WordNet** è un database lessicale per la lingua inglese (raggruppa le parole inglesi in insiemi di sinonimi chiamati synset).
- Specifico del dominio

- L'**Unified Medical Language System (UMLS)** è un compendio di molti vocabolari controllati nelle scienze biomediche.
- **MeSH** è un vocabolario controllato completo allo scopo di indicizzare articoli di riviste e libri nelle scienze della vita.

Classifica

Limiti del modello booleano: buono per **utenti esperti** con una comprensione precisa delle loro esigenze. Ottimo anche per le applicazioni che possono facilmente utilizzare migliaia di risultati per ulteriori elaborazioni. Finora, le nostre query sono state tutte booleane (**corrispondono o no**).

Non va bene per la maggior parte degli utenti: non sono in grado di scrivere query booleane (o lo sono, ma pensano che sia troppo lavoro) e non vogliono guardare migliaia di risultati (particolarmente vero per la ricerca sul web).

Esempio: le query booleane spesso danno un **numero insufficiente** di risultati (= 0) o un **numero eccessivo** di risultati (1000). Supponiamo che un utente abbia problemi con la sua scheda di rete:

- Query 1: "utente standard dlink 650" -> 200.000 risultati
- Query 2: "utente standard dlink 650 nessuna scheda trovata": 0 risultati

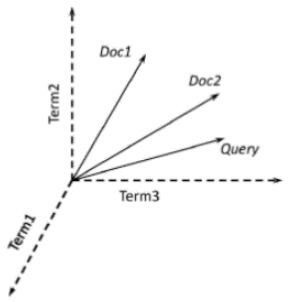
Ci vuole molta abilità per elaborare una query che produca un numero gestibile di risultati: **AND** ne dà troppo pochi, **OR** ne dà troppi

Recupero classificato: piuttosto che un insieme di documenti che soddisfano un'espressione di query, nel **recupero classificato**, il sistema restituisce un ordinamento sui documenti **(in alto)** nella raccolta per una query.

Quando un sistema produce un set di risultati classificato, le **grandi dimensioni non sono un problema:** mostra solo i **primi k** (≈ 10) risultati e non travolge l'utente.

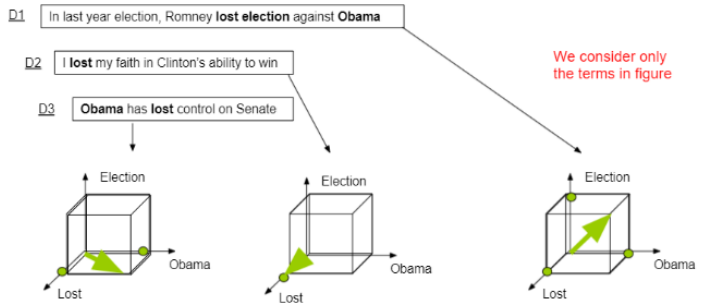
Vector Space Model (VSM): modello di classificazione

Come nel modello bag-of-word, i documenti sono **rappresentati da un vettore** o matrice di "**pesi dei termini**": d_{ij}, è il peso di Doc_i, per il $Term_j$



Rappresentazione vettoriale

Esempio con 3 termini e 3 documenti.



$$D_i = (d_{i1}, d_{i2}, \ldots, d_{it}) \quad Q = (q_1, q_2, \ldots, q_t)$$

	$Term_1$	$Term_2$	$\ldots$	$Term_t$
Doc_1	d_{11}	d_{12}	$\ldots$	d_{1t}
Doc_2	d_{21}	d_{22}	$\ldots$	d_{2t}
$\vdots$	$\vdots$			
Doc_n	d_{n1}	d_{n2}	$\ldots$	d_{nt}

Documenti come vettori

- |Spazio vettoriale |V|-dimensionale|
- I termini sono assi dello spazio
- I documenti sono punti o vettori in questo spazio
- Decine di milioni di dimensioni quando applicato a un motore di ricerca web (la maggior parte delle voci è vuota).

Query come vettori

- Rappresenta le query come vettori nello spazio
- Classifica i documenti in base alla loro vicinanza alla query nello spazio
  - Vicinanza = **somiglianza dei vettori**
  - Vicinanza = inverso della distanza
- **Classifica** i documenti più **rilevanti** più in alto rispetto a quelli meno rilevanti

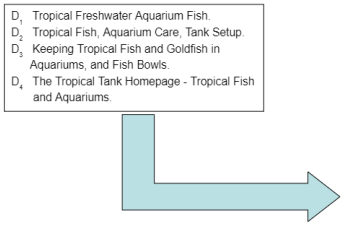
Termine Frequenze (TF)

Finora, i vettori erano binari e rappresentavano la **presenza o** l'**assenza** di termini in un documento.

Assegniamo un **peso** per ogni termine in uno spazio vettoriale:

- Il termine frequenza $tf_{t,d}$ del termine t nel documento d è definito come il numero di volte che t si verifica in d.
- Vogliamo usare la **f**requenza del **t**ermine durante il calcolo dei punteggi di corrispondenza del documento di query.

Esempio: TF come peso del termine

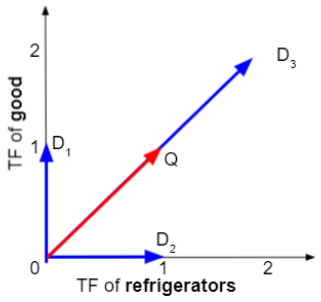


	Documents			
Terms	D_1	D_2	D_3	D_4
Aquarium	1	1	1	1
Bowl	0	0	1	0
Care	0	1	0	0
Fish	1	1	2	1
Freshwater	1	0	0	0
Goldfish	0	0	1	0
Homepage	0	0	0	1
Keep	0	0	1	0
Setup	0	1	0	0
Tank	0	1	0	1
Tropical	1	1	1	2

Le immagini tridimensionali sono utili, ma possono essere fuorvianti per lo spazio ad alta dimensione (più di 3 termini considerati)

Distanza spaziale vettoriale

Esempio di query Q: "Buoni frigoriferi"



- D_1: "Buongiorno a tutti voi."
- D_2: “Non mettete la pizza nei frigoriferi”
- D_3: “Recensione dei buoni frigoriferi: i primi cinque buoni frigoriferi.”

Mostriamo solo le due dimensioni interessanti, "good" e "refrigerators", ma esiste una dimensione per ogni termine indicizzato (pizza, mattina, ...)

Distanza euclidea: la distanza in linea retta tra i punti finali

- La distanza tra Q e D_1 e la distanza tra Q e D_2 è 1
- La distanza tra Q e D_3 è $\sqrt{2} \cong 1.414$ (**D_3 è il meno simile**, non è giusto)

Angolo: prendi un documento d e aggiungilo a sé stesso molte volte. Chiama questo documento più lungo d’. "Semanticamente" d e d' hanno lo stesso contenuto, ma la distanza euclidea tra i due documenti può essere piuttosto grande. Invece l'angolo tra i due documenti è 0, **corrispondente alla massima somiglianza**.

Idea chiave: classifica i documenti in base **all'angolo** con la query.

Esempio: utilizzo degli angoli

Consideriamo ora l'angolo tra i vettori

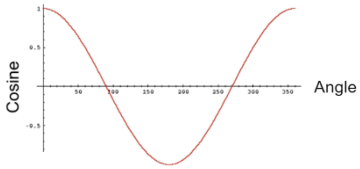
- L'angolo tra Q e D_1 e l'angolo tra Q e D_2 è 45°
- L'angolo tra Q e D_3 è 0° (**D_3 è il più simile**, è ciò di cui abbiamo bisogno)

Dagli angoli ai coseni

Le due nozioni seguenti sono equivalenti.

- Classifica i documenti in ordine decrescente dell'angolo tra query e documento
- Classifica i documenti in ordine crescente di coseno (query, documento)

Il coseno è una funzione monotona **decrescente** per l'intervallo [0°, 180°], il che significa che è inversamente proporzionale all'angolo



Somiglianza coseno

$$\cos(\vec{q},\vec{d}) = \frac{\vec{q} \bullet \vec{d}}{|\vec{q}||\vec{d}|} = \frac{\vec{q}}{|\vec{q}|} \bullet \frac{\vec{d}}{|\vec{d}|} = \frac{\sum_{i=1}^{|V|} q_i d_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2} \sqrt{\sum_{i=1}^{|V|} d_i^2}}$$

Dot product — Unit vectors

- **q_i, è il peso tf del termine i nella query**
- **d_i è il peso tf del termine i nel documento**
- $\cos(\vec{q},\vec{d})$ è la somiglianza coseno di $\vec{q}$ e $\vec{d}$ ... o, equivalentemente, il coseno dell'angolo tra $\vec{q}$ e $\vec{d}$.

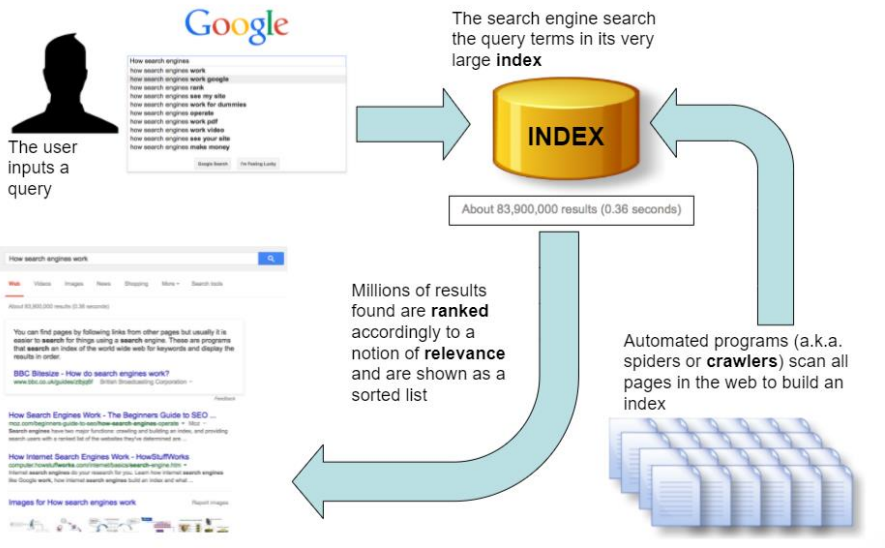
	Election	Lost	Obama
D1	1	1	1
D2	0	1	0
D3	0	1	1
Q	**0**	**0**	**1**

$$Sim(Q,D1) = \frac{\sum_{i=1}^{3} Q_i * D1_i}{\sqrt{\sum_{i=1}^{3} Q_i^2} * \sqrt{\sum_{i=1}^{3} D1_i^2}} = \frac{0*1 + 0*1 + 1*1}{\sqrt{1} * \sqrt{3}} = 0.577$$

$$Sim(Q,D2) = \frac{\sum_{i=1}^{3} Q_i * D2_i}{\sqrt{\sum_{i=1}^{3} Q_i^2} * \sqrt{\sum_{i=1}^{3} D2_i^2}} = \frac{0*0 + 0*1 + 1*0}{\sqrt{1} * \sqrt{1}} = 0.000$$

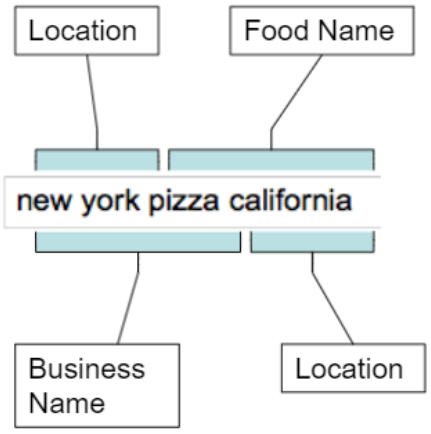
$$Sim(Q,D3) = \frac{\sum_{i=1}^{3} Q_i * D3_i}{\sqrt{\sum_{i=1}^{3} Q_i^2} * \sqrt{\sum_{i=1}^{3} D3_i^2}} = \frac{0*0 + 0*1 + 1*1}{\sqrt{1} * \sqrt{2}} = 0.707$$

Come funziona il motore di ricerca



Come può un motore di ricerca sapere cosa sta **cercando l'utente**?

- **Bisogno di informazione:** il desiderio di localizzare e ottenere informazioni per soddisfare un bisogno conscio o inconscio
- **Query utente:** l'insieme di termini consecutivi formulati da un utente per esprimere il suo bisogno di informazioni
- **Intento della query:** l'attività o l'intento di un utente, espresso da una query. La stessa query formulata da utenti diversi può essere il risultato di intenti diversi (**ambiguo**).



Classifica: Google Page-Rank

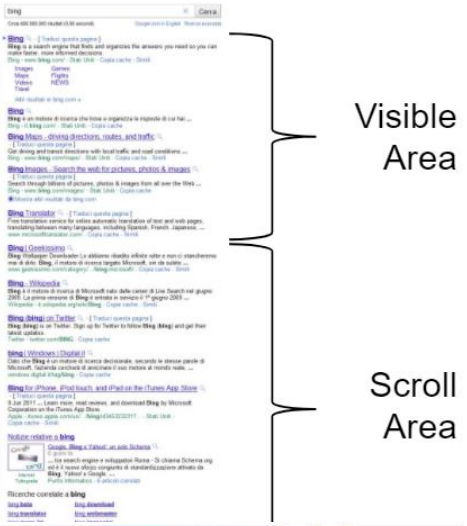
Una nozione di pertinenza basata solo sulle frequenze dei termini non è sufficiente per classificare miliardi di documenti; quindi, applichiamo misure complesse per valutare la **qualità**, l'**affidabilità** e l'**autorità** delle pagine web:

- Misure basate sulle **proprietà della rete topologica:** imposta un ciclo ricco per diventare più ricco, in base al quale pochi siti dominano i primi posti

- Misure basate su **diversi potenziamenti del campo** (titolo, sottotitolo, corpo hanno peso diverso)
- Misure basate **su semantica e proprietà spazio-temporali** (freschezza di una pagina)

Pagina dei risultati del motore di ricerca (SERP)

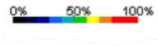
Mostra per impostazione predefinita **10 risultati** e di questi risultati, in media, i primi **5 sono visibili** senza scorrere verso il basso. Consideriamo il **comportamento degli utenti sulla prima SERP** dei motori di ricerca.



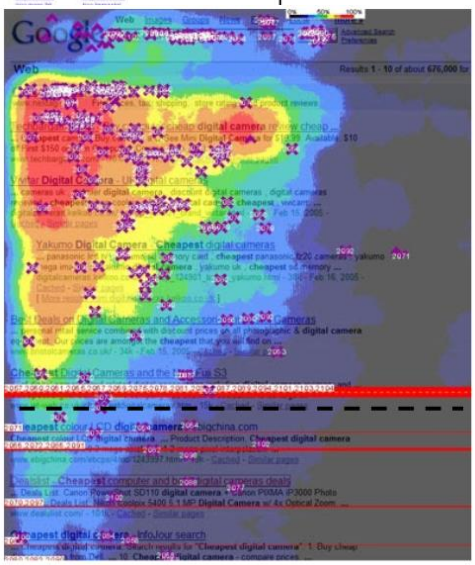
Distorsione di posizionamento dei risultati

Quando cerca informazioni, l'utente medio non guarda tutti i risultati, ma **semplicemente fa clic sui risultati migliori.** Sono stati condotti studi confrontando le risposte degli utenti quando hanno ricevuto elenchi di risultati nell'**ordine normale rispetto a un ordine inverso**

Eye-tracking: Triangolo d'Oro

Il colore 

rappresenta la percentuale di tempo trascorso a guardare l'area.



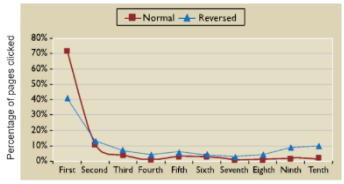
La X viola rappresenta un clic del mouse sulla pagina.

La linea tratteggiata **----** rappresenta il punto in cui la pagina si interrompe sullo schermo del computer (Area visibile/Area di scorrimento).

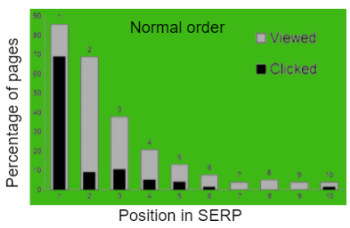
Le linee rosse indicano fino a che punto è stata fatta scorrere la pagina prima di lasciarla.

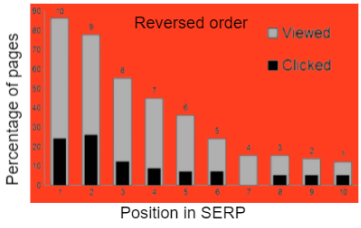
I risultati in prima posizione vengono cliccati **più di 10 volte** dei risultati in sesta posizione

Pregiudizio su clic ed eye-tracking



Anche se i risultati mostrano una certa consapevolezza dell'utente, un'**eccessiva fiducia** negli algoritmi di **ranking** può influire negativamente sui siti Web posizionati più piccoli



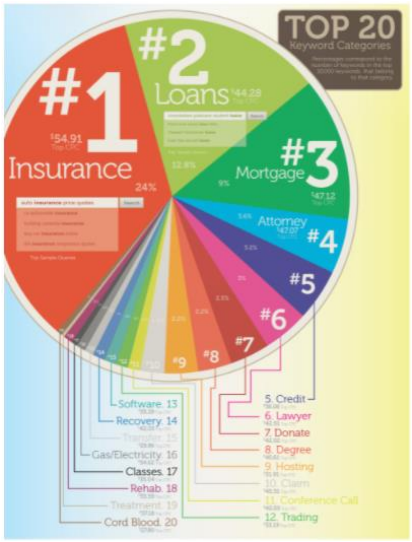


Impatto della posizione sul business

Supponiamo che la società A e la società B siano rispettivamente alla prima e alla sesta posizione per una parola chiave di valore: B dovrebbe acquistare dieci volte i clic che riceve A per ottenere lo stesso traffico!

Costo Per Clic (CPC) in media = 1$

Su Google alcune parole chiave possono essere molto costose **(parole chiave più costose di Google 2011)**:



1. **Assicurazione: 54,91 $**
2. Mutuo: 47,12 $
3. Procuratore: 47.07
4. Richiesta: 45,51 $
5. Prestiti: 44,28$
6. Avvocato: 42,51$

Bing è ancora più costoso **(parole chiave più costose di Bing 2015)**:

1. **Avvocati: 109,21$**
2. Procuratore: 101,77 $
3. Insediamenti strutturati: 78.39$

Rarità dei termini

I termini comuni sono meno informativi dei termini rari, quindi **pesano più le parole rare** rispetto alle parole comuni**.**

Considera una query come "buoni frigoriferi": “buoni” è una parola molto comune, che può essere trovata nella maggior parte dei documenti, mentre “frigoriferi” è meno comune e quindi più informativo per pertinenza

Frequenza Documento (DF): misura **quanto è comune un termine nell'intera raccolta di documenti.**

$\boldsymbol{df_t}$ è il numero di documenti che contengono t

Usiamo **DF** per formulare il concetto **inverso**, per esprimere la **rarità** di una parola.

Frequenza Documento Inversa (IDF): più una parola è **comune**, **meno informativa** sarà per la pertinenza (caso estremo: stop word).

$$\boldsymbol{idf_t = \log_{10}(N/df_t)}$$

- N: numero di documenti nella raccolta.
- $\boldsymbol{\log_{10}(N/df_t)}$: applichiamo il logaritmo per alleggerire il fenomeno **esponenziale** della linguistica (legge di Zipf), altrimenti parole molto comuni avrebbero IDF pari a zero (per esempio, una parola comune come "casa" può avere una frequenza di **1 milione** di volte **più grande** rispetto a una parola rara come "fenestrazione").

Effetto dell'IDF sulla classifica

IDF influisce sulla classificazione dei documenti per le query con almeno due termini: per la query **persona capricciosa**, la ponderazione IDF fa sì che le occorrenze di **capriccio** contino molto di più nella classifica del documento finale rispetto alle occorrenze di **persona**

IDF viene sempre utilizzato insieme TF per una migliore stima della rilevanza del documento

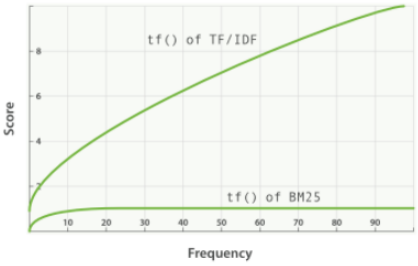
TF-DF (nomi alternativi: TF.IDF, **TF x IDF**)**:** schema di ponderazione più noto nel recupero delle informazioni. Il peso **TF-IDF** di un termine è il prodotto del peso **TF** (aumenta con il numero di occorrenze all'interno di un documento) e del peso **IDF** (aumenta con la rarità del termine nella raccolta).

$$w_{t,d} = tf_{t,d} * \log_{10}\left(\frac{N}{df_t}\right)$$

2. Modelli probabilistici

La **graduatoria dei documenti** è **in ordine decrescente di probabilità di rilevanza,** stimata nel modo più accurato possibile utilizzando i dati disponibili → **migliore efficacia**

1. **Okapi BM25 (schema di ponderazione):** modello probabilistico sensibile alla **frequenza del termine,** alla **rarità del termine** (simile a IDF) e alla **lunghezza del documento**
   - In Okapi BM25, il punteggio di similarità è sempre compreso tra **0** e **1** (i valori più alti di frequenza dei termini non influiscono sul punteggio)
   - **Parametri**
     1. **k1:** controlla la **saturazione** della **frequenza del termine**. Il valore predefinito è 1.2 perché valori più bassi producono una saturazione più rapida e valori più alti una saturazione più lenta
     2. **b:** controlla quanto effetto dovrebbe avere la normalizzazione della lunghezza del documento (il valore predefinito è 0,75)
        - Un valore di 0,0 disabilita completamente la normalizzazione
        - Un valore di 1,0 normalizza completamente il documento



2. **Modello linguistico:** un modello di linguaggio statistico assegna una probabilità a una sequenza di m **parole** P(W_1, ..., W_m) mediante una **distribuzione di probabilità**.
   - **Un modello linguistico separato è associato a ogni documento in una raccolta. I documenti sono classificati in base alla probabilità della query Q nel modello di linguaggio del documento P(Q|M_d).**
   - **Modello di linguaggio Unigram:** la generazione del testo consiste nell'estrarre le parole da un "secchio" secondo la distribuzione di probabilità e sostituirle
   - **Modello di linguaggio a N-grammi:** alcune applicazioni utilizzano modelli di linguaggio bigram e trigram in cui le probabilità dipendono dalle parole precedenti

Aspetti di efficienza del ranking

Il **calcolo** del punteggio di pertinenza utilizzando modelli non booleani può essere piuttosto **impegnativo** dal punto di vista computazionale:

- Il documento viene recuperato attraverso un **punteggio di pertinenza**, anche se non sono presenti tutti i termini della query.
- Le **formule** di pertinenza complesse richiedono **calcoli** aggiuntivi per calcolare il **punteggio**

Strategie per controllare l'efficienza:

- Il **modo** in cui accediamo ai documenti influisce sui tempi di calcolo
- Il più delle volte non siamo interessati ai documenti meno **rilevanti**

Punteggio efficiente

Una grande frazione del lavoro della CPU per ogni query di ricerca è dedicata solo al **calcolo del punteggio** di pertinenza

- Generalmente, abbiamo un budget limitato sulla latenza (ad esempio 250ms) per rispondere alla query, altrimenti l'utente non sarà felice

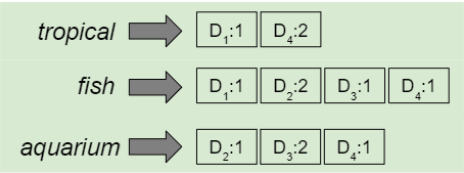
- Il provisioning della CPU non consente di assegnare un punteggio completo a ogni documento per ogni query

Idea di base: evita di segnare documenti che non arriveranno tra i migliori K

Tecniche TAAT vs DAAT

TAAT (termine alla volta): punteggi per tutti i documenti calcolati contemporaneamente, un termine di query alla volta. **Esempio:**

- **Query:** acquario di pesci tropicali
- **Indice invertito**
- **Frequenza** dei termini per ogni documento:



  - D_1 = 1 + 1 = 2
  - D_2 = 2 + 1 = 3
  - D_3 = 1 + 2 = 3
  - D_4 = 2 + 1 + 1 = 4

DAAT (documento alla volta): punteggio totale per ogni documento (inclusi tutti i termini della query), calcolato prima di procedere al successivo

Ognuno ha implicazioni su come l'**indice** di recupero è strutturato e memorizzato.

Classifica sicura: utilizzata per metodi che garantiscono che i K documenti restituiti siano i **K** documenti con **punteggio più alto in assoluto.** L'obiettivo è quello di eseguire meno calcoli dando buoni risultati, approssimativi, ma abbastanza buoni da soddisfare l'utente.

Classifica non sicura: la funzione di ranking è solo un proxy per la felicità dell'utente. I documenti vicini alla prima K **potrebbero** andare bene.

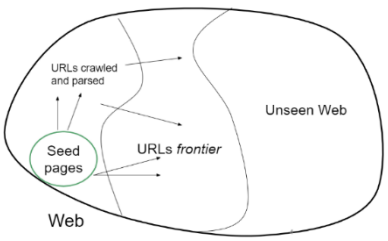
- "**Eliminazione**" **dell'indice:** considera solo **termini di query con IDF elevato (rari)**, scarta quelli comuni o considera solo i **documenti che contengono molti termini di query** e scarta quelli con pochi.
- **Elenchi dei campioni:** per ogni termine, i punteggi dei principali documenti rilevanti sono precalcolati (prima che venga emessa la query)

Recupero informazioni web

Scansione web: raccoglie in modo **rapido** ed efficiente quante più **pagine Web** utili possibili, insieme alla **struttura di link** che le collega

Funzionamento di base del crawler (ragno)

1. Inizia con gli **URL** "seme" noti
2. **Recupera e analizza:** estrae gli URL a cui puntano e li posiziona in una coda, chiamata frontiera degli URL
3. Recupera ogni URL alla **frontiera** e ripete



Requisiti del crawler

Robustezza: scansione del Web distribuita su più macchine

- **Pagine dannose:** spam e trappole per ragni (recupero di un numero infinito di pagine in un particolare dominio), incluse quelle generate dinamicamente

- Anche le **pagine non dannose** pongono delle sfide: la **latenza**/larghezza di banda ai server remoti varia, **condizioni** dei webmaster, "**profondità**" della scansione nella gerarchia di URL di un sito, **Mirror** del sito e pagine duplicate

Politeness (educazione): politiche dei server Web (regola la velocità dei crawler)

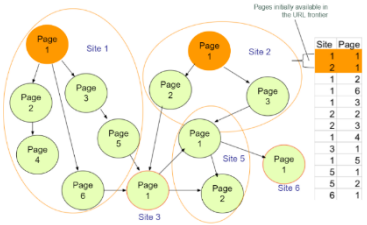
- **Implicita:** anche senza specifiche, evita di visitare qualsiasi sito troppo spesso.
- **Esplicita:** specifiche del sito Web su quali parti del sito possono essere sottoposte a scansione
    - **robots.txt:** protocollo per fornire agli spider ("robot") un accesso limitato a un sito Web. Il **sito crea un file** chiamato robots.txt che specifica le **restrizioni** di accesso tramite una serie di **regole**, che dovrebbero essere seguite dai **client**

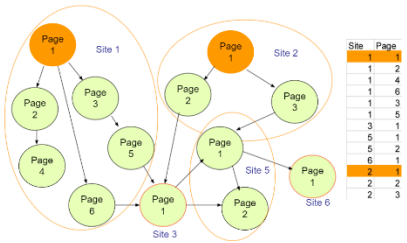
Esempio robots.txt: accesso vietato per **tutti** all'URL che inizia con "/yoursite/temp/", ad **eccezione** dello user-agent (nomi client) chiamato "searchengine" a cui è **tutto accessibile**

User-agent	Disallow
*	/yoursite/temp/
searchengine	

Frontiera URL

Strategia Breadth first: data una pagina Web nella **frontiera URL**, **aggiunge tutte le pagine collegate dalla pagina corrente**. La copertura è ampia ma superficiale.





Strategia Deep First: data una pagina Web nella **frontiera URL**, **segue il primo collegamento nella pagina corrente** fino alla prima pagina senza collegamenti.

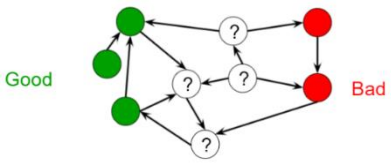
Cercando nel web: supponiamo che tu debba visitare eBay e non sai che www.ebay.com è l'URL. Ci sono milioni di pagine web che contengono il termine "eBay" e siti web con più frequenza sul termine "eBay" rispetto a eBay stesso. Abbiamo bisogno di una nozione di **popolarità**, insieme a una nozione di pertinenza.

Recupero di informazioni sul Web, costruiscono il ranking combinando:

- **Punteggio del contenuto:** grado di corrispondenza di una pagina. Può essere calcolato utilizzando uno dei modelli di recupero delle informazioni descritti finora (probabilistici)
- **Punteggio di popolarità:** grado di importanza di una pagina. Può essere calcolato da un'analisi della struttura dei **collegamenti ipertestuali** delle pagine indicizzate utilizzando uno o più modelli di **analisi dei collegamenti**.

Analisi dei collegamenti semplice: i nodi **Buoni** non indicheranno i nodi **Cattivi**

- **Cattivo:** se indichi un nodo **Cattivo**
- **Buono:** se un nodo **Buono** punta a te



Analisi delle citazioni: la frequenza delle citazioni è una stima della popolarità

- Frequenza di accoppiamento bibliografica: gli articoli che co-citano gli stessi articoli sono correlati
- Indicizzazione delle citazioni: da chi è citato questo autore?
- Anteprima Page-Rank: quali riviste sono autorevoli?

Page-Rank: assegna un **punteggio numerico compreso tra 0 e 1** a ogni nodo e dipende dalla **struttura dei link** del grafico web. Data una query, un motore di ricerca web calcola un **punteggio composito** per ogni pagina web che combina centinaia di caratteristiche come la somiglianza del coseno, insieme al punteggio Page-Rank. Questo punteggio composito viene utilizzato per fornire un **elenco classificato di risultati** per la query.

La surfista casuale: prendiamo in considerazione Alice, che vaga senza meta tra le pagine web per sempre, partendo da Google (con il pulsante "**sorprendimi**" andrà su una pagina Web casuale e usandolo all'infinito raggiungerà ogni pagina di internet). Ogni volta che viene caricata una pagina web, lei sceglie se fare clic su un collegamento **casuale** nella pagina o fare clic sul pulsante sorprendimi. Alice naviga sul Web utilizzando questo **algoritmo**:

1. Sceglie un numero casuale r compreso tra 0 e 1.
2. Se r > λ: fa clic sul pulsante "sorprendimi"
3. Se r $\leq \lambda$: fa clic su un collegamento casuale nella pagina corrente.
4. Ricomincia.

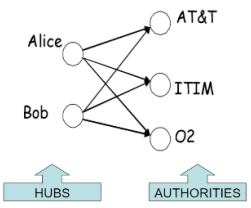
Mentre Alice sta navigando, tu entri e guardi la pagina web sul suo schermo. La **probabilità** che stia guardando il sito web di eBay è il **Page-Rank** del sito.

Utilizzo del Page-Rank in Google

Il Page-Rank è **solo uno dei tanti** fattori che determinano il punteggio finale di una pagina Web in Google e fa parte di un sistema di classificazione molto più ampio che si ritiene rappresenti più di **200 diversi "segnali"**:

- caratteristiche della lingua (frasi, sinonimi, errori di ortografia, ecc.)
- funzioni di query relative a funzioni linguistiche, termini/frasi di tendenza
- caratteristiche legate al tempo (le query relative alle "notizie" potrebbero ricevere una risposta migliore da documenti indicizzati di recente)
- funzioni di personalizzazione, che riguardano la cronologia delle ricerche, il comportamento e l'ambiente sociale.

Ricerca di argomenti indotta da collegamenti ipertestuali (HITS): adatto per query di "argomento ampio", perché raggiunge una fetta ampia di *opinione* comune; in risposta a una query trova due serie di **pagine correlate**



- **Pagina hub** (elenco di collegamenti su un argomento): *punta* a molte buone pagine autorevoli per quell'argomento
- **Pagina di autorità**: indicata da molti buoni hub per quell'argomento

Definizione circolare: la trasformerà in un calcolo iterativo.

Information Retrievial: recupero dei documenti e il resto dipende da te

Ricerca semantica: esegue una **ricerca grafica** sulla conoscenza strutturata (grafico della conoscenza di Google, ricerca di grafici di Facebook)

Valutazione IR

Misurare la rilevanza

1. Raccolta di documenti di riferimento
2. Suite di query di riferimento
3. **Valutazione** umana di rilevante o non per ogni query e ogni documento

L'esigenza dell'utente è tradotta in una **query**, ma la pertinenza viene valutata in base alle **esigenze dell'utente**, non alla **query.** Ad esempio, bisogno di informazioni: *il fondo della mia piscina sta diventando nero e deve essere pulito.*

- **Query: pulitore di piscine**
- **Valuta** se il documento risponde al **bisogno** sottostante, non se contiene le parole

Giudizi di pertinenza

Binario (rilevante o non) nel caso più semplice o (0, 1, 2, 3, ...) negli altri.

Se per ogni query consideriamo tutto l'insieme dei documenti da giudicare, la valutazione della pertinenza può essere enorme e costosa.

La soluzione di **deep-k pooling**: prendi in considerazione i **primi k** (es. 100) documenti di **N** (es. 100) diversi sistemi di recupero delle informazioni. Gli esseri umani devono giudicare un "pool" di non più di **k** x **N** documenti (ad es. 10.000), che è di gran lunga inferiore all'intera raccolta di documenti (potrebbero essere milioni di documenti).

Collezioni di test qualificati

Collezioni Text Retrieval Conference (TREC): la più grande raccolta Web facilmente disponibile per scopi di ricerca, che include 25 milioni di pagine. Il NIST degli Stati Uniti ha condotto un'ampia serie di valutazioni del banco di prova IR dal 1992. All'interno di questo quadro, ci sono state **molte tracce** su una serie di **diverse raccolte di test**.

Collection	NDocs	NQrys	Size (MB)	Terms/Doc	Q-D Rel/Ass
ADI	82	35			
AIT	2109	14	2	400	>10,000
CACM	3204	64	2	24.5	
CISI	1460	112	2	46.5	
Cranfield	1400	225	2	53.1	
LISA	5872	35	3		
Medline	1033	30	1		
NPL	11,429	93	3		
OSHMED	34,8566	106	400	250	16,140
Reuters	21,578	672	28	131	
TREC	740,000	200	2000	89-3543	» 100,000

Typical TREC

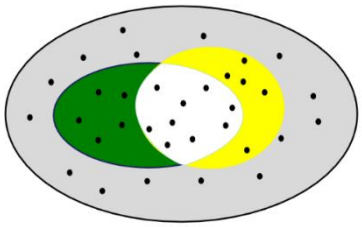
Turco meccanico

Presentare coppie di query-documento a manodopera a basso costo su piattaforme di crowdsourcing online (più economico rispetto all'assunzione di valutatori qualificati). Ricevi qualche segnale, ma la **varianza** nei giudizi risultanti è **molto alta**

Misure di efficacia

Per **valutare l'efficacia di un sistema IR**, cioè la qualità dei suoi risultati di ricerca, ci sono due parametri sui risultati restituiti dal sistema per una query:

- **Precisione:** frazione di documenti **restituiti** rilevante per la necessità di informazioni
  - $$\textbf{\textit{Precisione}} = \frac{Documenti\ Recuperati\ Rilevanti}{Documenti\ Recuperati}$$
- **Richiamo:** frazione di documenti **rilevanti** nella raccolta restituita dal sistema
  - $$\textbf{\textit{Richiamo}} = \frac{Documenti\ Recuperati\ Rilevanti}{Documenti\ Rilevanti}$$



Documenti recuperati: ogni punto • è un documento della collezione

Documenti rilevanti: l'insieme in verde è l'insieme di tutti i documenti realmente **rilevanti** per la query e l'insieme in giallo è l'insieme di **tutti** i **documenti** restituiti dal sistema che vogliamo valutare.

Documenti recuperati rilevanti = documenti rilevanti ∩ documenti recuperati

Precision @K e Recall @K:

1. **Imposta** una soglia di rango **K**
2. **Calcola % rilevante** nel K superiore
3. **Ignora i documenti** classificati **inferiori a K**

Precisione media (AP): la **media** di Precision @K calcolata **solo per quei K in cui viene recuperato il risultato rilevante.** Si ferma solo quando vengono recuperati **tutti i documenti rilevanti** (Recall@K=1). **AP è in grado di rilevare che il Ranking #1 è migliore**, poiché classifica i documenti più rilevanti nelle posizioni più elevate.

Mean Average Precision (MAP): macro-media, cioè la **media misurata della precisione media** su **tutte le query** (ogni query conta allo stesso modo e le raccolte di test si estendono da 50 a 500 query). MAP presuppone che l'utente sia interessato a trovare **molti documenti pertinenti** per ogni query.

- **Precisone 0:** se un documento rilevante non viene mai recuperato.

Nozione binaria di rilevanza: un documento è rilevante per la query o non lo è. La rilevanza binaria è più comune e fornisce una buona stima per la valutazione IR.

Nozione non binaria: alcuni documenti possono essere **meno rilevanti** di altri, ma comunque rilevanti. **DCG** (**D**iscounted **C**umulative **G**ain) o **NDCG** (**N**ormalized **D**iscounted **C**umulative **G**ain).

Metodi Avanzati

Sinonimia: nella maggior parte delle raccolte, si può fare riferimento allo stesso concetto usando **parole diverse.** Questo problema ha un impatto sul **richiamo** della maggior parte dei sistemi di recupero delle informazioni (vorrai che la ricerca per **aeromobile** corrisponda anche alla parola **aeroplano**).

Espansione della query: aumentiamo la query con sinonimi di parole chiave e termini correlati.

- **Query utente:** "auto"
- **Query estesa:** "auto automobili automobile automobili auto"

È stata sviluppata una varietà di tecniche di espansione delle query automatiche o semiautomatiche con l'obiettivo di migliorare l'efficacia abbinando i termini correlati (le tecniche semiautomatiche richiedono l'interazione dell'utente per selezionare i migliori termini di espansione).

Termini correlati

- Vocabolari controllati **(Wordnet)**
- Raccolta di testi: termini **co-occorrenti**, da documenti pertinenti, da documenti recuperati o in una **finestra** adiacente di documenti rilevanti o recuperati.

Espansione query thesaurus: l'espansione automatica basata su un vocabolario generale controllato (thesaurus) **non** è **molto efficace**, perché non tiene conto del **contesto**

- Query: "acquari tropicali"
- **Query ampliata**: "acquari vasche per pesci tropicali"
- Query: "armatura per carri armati"
- **Query ampliata**: "armatura per acquari vasche"

Espansione query di ricorrenza: diverse misure di **co-occorrenza** possono essere utilizzate per trovare la parola chiave correlata (estratta dalle raccolte di testo). Le misure si basano su documenti interi o in parti (frasi, paragrafi, finestre); per semplicità, consideriamo interi documenti.

Dice’s coefficent

Supponiamo di voler trovare parole relative a "pesce"

Misura di co-occorrenza: $\frac{\text{Quante volte appaiono } \textbf{insieme}}{\text{Quante volte appaiono } \textbf{singolarmente}}$

Più alto è questo punteggio, più correlate dovrebbero essere le due parole!

Date due parole **a** e **b**, è formalmente definita come: $\frac{2n_{ab}}{n_a+n_b}$

- $\boldsymbol{n_a}$ è il numero di documenti contenenti la parola **a**.
- $\boldsymbol{n_b}$ è il numero di documenti contenenti la parola **b**.
- $\boldsymbol{n_{ab}}$ è il numero di documenti che contengono **entrambe** le parole **a** e **b**.

Mutual Information

Utilizzato in numerosi studi sulla collocazione delle parole, simile a Dice, è basato sulle probabilità.

Per due parole a e b, è definita come $\log\frac{P(a,b)}{P(a)P(b)}$

- **P(a)** è la probabilità che la parola **a** compaia in una finestra di testo.
- **P(b)** è la probabilità che la parola **b** compaia in una finestra di testo.
- **P(a, b)** è la probabilità che a e b compaiano nella stessa finestra di testo.

Un **problema** con **l'informazione reciproca** è che tende a favorire i termini a bassa frequenza. Per esempio:

- Considera due parole **a** e **b**:
  - $\boldsymbol{n_a = n_b = 10}$
  - Si verificano la **metà delle volte** (n_{ab} = 5).
  - L'informazione reciproca per questi due termini è $\boldsymbol{5 * 10^{-2}}$
- Considera due parole **c** e **d**:
  - $\boldsymbol{n_c = n_d = 1000}$
  - Si verificano **metà delle volte** (n_{cd} = 500).
  - L'informazione reciproca per questi due termini è $\boldsymbol{5 * 10^{-4}}$.

Entrambe le coppie coesistono la metà del tempo in cui si verificano. Tuttavia, hanno diverse informazioni reciproche: 0,05 vs 0,0005.

Expected Mutual Information

L'informazione mutua attesa affronta il **problema a bassa frequenza** pesando il valore dell'informazione mutua utilizzando la probabilità P(a, b).

Ci interessa principalmente il caso in cui ricorrano entrambi i termini, dando la formula: $\boldsymbol{P(a,b) * \log\frac{P(a,b)}{P(a)P(b)}}$

Chi quadrato di Pearson (X^2)

Confronta il numero di co-occorrenze di due parole con il numero previsto di co-occorrenze se le due parole fossero indipendenti e normalizza questo confronto per il numero atteso.

$$\frac{\left(n_{ab} - N * \frac{n_a}{N} * \frac{n_b}{N}\right)^2}{N * \frac{n_a}{N} * \frac{n_b}{N}}$$

$N * \frac{n_a}{N} * \frac{n_b}{N}$ è il numero previsto di co-occorrenze se i due termini si verificano indipendentemente.

Esempio di espansione della query

Utilizzando una **raccolta di notizie TREC**, le quattro misure di co-occorrenza vengono applicate a livello di documento e vengono mostrate le prime 5 parole correlate. La parola per la quale stiamo cercando termini correlati è "**fish"**

- L'informazione reciproca favorisce le parole molto rare e a volte errate.
- Chi-quadrato cattura anche parole insolite.
- **Il coefficiente di dadi e le informazioni reciproche previste sono più adatti per l'espansione della query IR**.

Dice's coefficient	Mutual information	Expected mutual information	Pearson's Chi-squared
species	zoologico	water	arsiq
wildlife	zapanta	species	happyman
fishery	wrint	wildlife	outerlimit
water	wpfmc	fishery	sportk
fisherman	wighout	sea	lingcod

Feedback sulla pertinenza (RF): tecnica di espansione e perfezionamento della query basata sul **riscontro degli utenti**.

1. L'utente invia una breve e semplice query.
2. Il sistema restituisce una serie iniziale di risultati.
3. L'utente contrassegna alcuni documenti restituiti come pertinenti o no.
4. Il sistema calcola una migliore rappresentazione del fabbisogno di informazioni in base al riscontro dell'utente.
5. Il sistema visualizza una serie rivista di risultati di recupero.

- **Esempio RF**

Query: *New space satellite applications*

Rank	Document Title	User Feedback
1	**NASA Hasn't Scrapped Imaging Spectrometer**	**YES**
2	**NASA Scratches Environment Gear From Satellite Plan**	**YES**
3	Science Panel Backs NASA Satellite Plan, But Urges Launches of Smaller Probes	NO
4	A NASA Satellite Project Accomplishes Incredible Feat: Staying Within Budget	NO
5	Scientist Who Exposed Global Warming Proposes Satellites for Climate Research	NO
6	Report Provides Support for the Critics Of Using Big Satellites to Study Climate	NO
7	Arianespace Receives Satellite Launch Pact From Telesat Canada	NO
8	**Telecommunications Tale of Two Companies**	**YES**

Documents relevant from the user feedback

#1	**NASA Hasn't Scrapped Imaging Spectrometer**
#2	**NASA Scratches Environment Gear From Satellite Plan**
#8	**Telecommunications Tale of Two Companies**

Recurring keywords in relevant documents

new, space, satellite, application, **nasa, eos, launch, aster, instrument, arianespace, bundespost, ss, rocket, scientist, broadcast, earth, oil, measure**

Pseudo RF (feedback di rilevanza cieca)**:** il feedback di pseudo pertinenza fornisce un metodo per il feedback di pertinenza **automatico**. Automatizza la parte manuale della RF, in modo che l'utente ottenga prestazioni di recupero migliorate **senza un'interazione estesa**.

1. Si esegue il **recupero** normale per trovare i documenti più rilevanti
2. Si suppone che i **primi k** documenti classificati siano **rilevanti**
3. Si **calcola RF** come prima sotto questa ipotesi.

Machine Learning (ML) e IR: combiniamo contemporaneamente la frequenza dei termini nel corpo e nel titolo del documento, la lunghezza del documento e la popolarità del documento (es. Page-Rank).

Un modello di **apprendimento per classificare** apprende i pesi da un insieme di funzionalità e giudizi di pertinenza. Vogliamo usare l'apprendimento automatico per costruire un classificatore che **classifichi** i documenti in **classi rilevanti e non rilevanti**

Dati di formazione limitati: è molto difficile raccogliere query su test e giudizi di pertinenza rappresentativi delle reali esigenze degli utenti. Le funzioni di classificazione tradizionali in IR

utilizzano un **numero molto ridotto di funzionalità** come la frequenza del termine e del documento inversa e la lunghezza del documento

Imparare a classificarsi: i sistemi moderni sul Web utilizzano un **gran numero di caratteristiche** e sono disponibili **molti dati di allenamento** da enormi registri di query raccolti dalle interazioni dell'utente, come la lunghezza dell'URL e della pagina, la frequenza della parola di query nel testo, ecc.

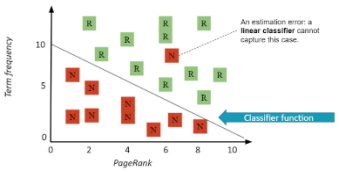
Query: "*fish tank price*"

INPUT				OUTPUT
Training sample	**Doc ID**	**Term frequency**	**PageRank**	**judgement**
001	37	11	3	1 (relevant)
002	37	0	8	0 (nonrelevant)
003	238	8	2	1 (relevant)
004	238	1	2	0 (nonrelevant)
005	1741	5	6	1 (relevant)
006	2094	18	1	1 (relevant)
007	3191	3	2	0 (nonrelevant)

- **Esempio:** supponiamo di considerare due **caratteristiche** in ogni documento. Il training set è composto da vettori **(TF, PR)** in input, ciascuno con un corrispondente giudizio di rilevanza e produce la pertinenza stimata.

Rilevanza feedback: RF è un semplice esempio di utilizzo dell'apprendimento automatico supervisionato nel recupero delle informazioni, i **dati di addestramento** (ovvero i documenti pertinenti e non identificati) vengono utilizzati per migliorare le prestazioni del sistema.

Apprendimento offline: utilizziamo i giudizi di pertinenza di una raccolta di test per addestrare un classificatore.



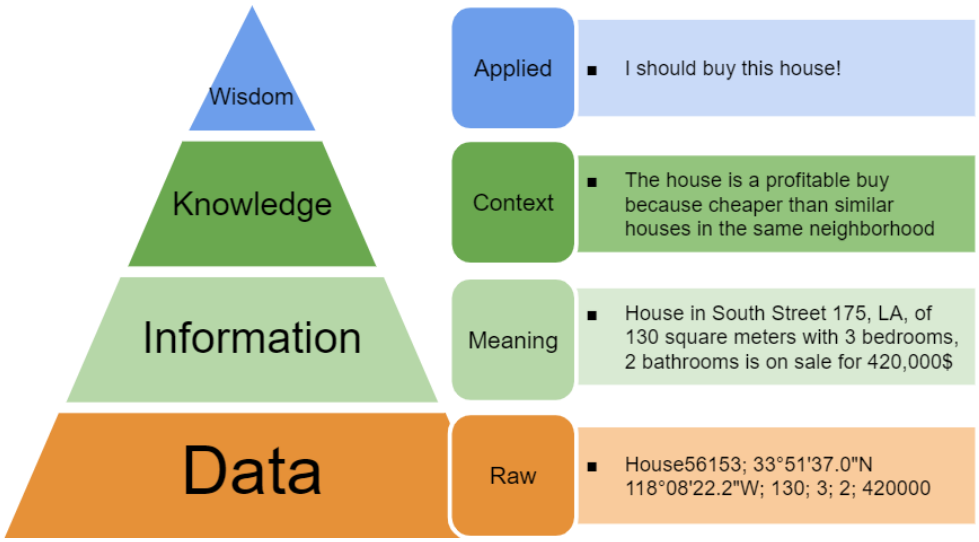
- **Esempio:** i **pesi** appresi sono i **coefficienti** di una funzione lineare che rappresenta il modello appreso che **separa** i documenti rilevanti (output 1) da quelli non rilevanti (output 0) in base alle due variabili di input.

Apprendimento online: un esempio è l'utilizzo del feedback sulla pertinenza per ottimizzare i pesi del classificatore e migliorarne l'accuratezza.

Analisi dei dati

Dai dati alla saggezza

Piramide DIKW: in genere le informazioni sono definite in termini di **dati**, la conoscenza in termini di informazioni e la **saggezza** in termini di conoscenza

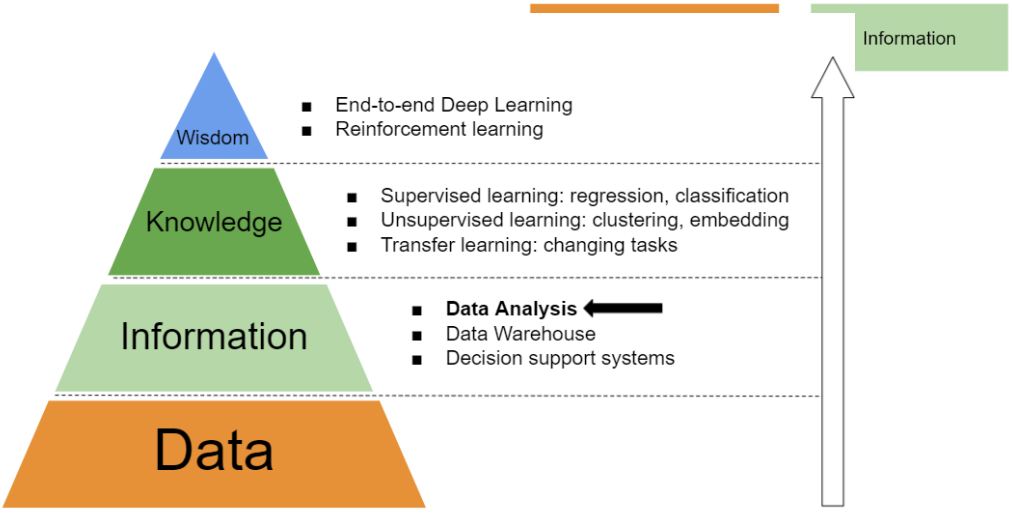


Partendo dal livello basso

Dati semi-strutturati: archivia e fai query tramite SQL sui dati senza avere uno schema rigido, relazionandoti ai dati strutturati. Possono fornire informazioni solo definendo manualmente query complesse

Recupero delle informazioni: recupera un sottoinsieme di documenti rispetto alle esigenze di informazioni dell'utente, **ricerca** nel WWW e **classifica** i documenti in base alla loro pertinenza rispetto a una query di testo e al riscontro dell'utente senza estrarle dai dati (i documenti sono già "informazioni" in linguaggio naturale).

Analisi dei dati: lo scopo è **estrarre informazioni** di base da raccolte di dati.

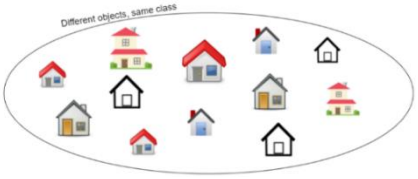


Le **informazioni** estratte possono essere:

- **Riassuntive:** ad esempio, la media da un insieme di valori numerici
- **Sull'associazione:** ad esempio, la relazione tra due insiemi di valori (prezzo delle case vs metri quadrati)

Popolazione: insieme di oggetti a cui siamo interessati. Ad esempio:

- Tutte le case di Los Angeles
- Tutti gli studenti dell'università
- Tutte le ricevute di un negozio di alimentari

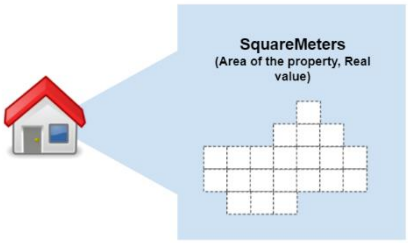


City	Latitude	Longitude	Bedrooms	SquareMeters	Price
Los Angeles	33°51'37.0"N	118°08'22.2"W	3	130	420000

Record (osservazione, caso)**:** tupla di valori che caratterizzano un elemento di una popolazione

Variabile (campo, caratteristica)**:** nome del valore di un record, ha un significato e un tipo comuni per tutti i record della popolazione

- **Variabile numerica** (quantitativa)**:** possiamo applicare su di essa operazioni aritmetiche. Ad esempio: il prezzo 420000
  - **Discreta**: i valori possono essere contati
  - **Continua**: risultato di una misura continua
- **Variabile categoriale** (qualitative)**:** non possiamo applicare su di essa operazioni aritmetiche. Ad esempio: le città di Los Angeles, New York, Roma



- **Ordinale**: esiste un ordine naturale sui valori possibili (ad esempio voti scolastici: A, B, C, D)
- **Nominale**: ordinato per nome (ad es. colori)

City	Latitude	Longitude	Bedrooms	SquareMeters	Price
Los Angeles	33°51'37.0"N	118°08'22.2"W	3	130	420000
Los Angeles	33°50'17.7"N	118°09'12.6"W	2	60	380000
Los Angeles	33°49'32.3"N	118°08'44.1"W	5	230	2500000
...	...	...	...	...	...
Albuquerque	35°12'06.1"N	106°58'31.1"W	2	105	190000
Albuquerque	35°15'17.0"N	106°59'26.8"W	4	225	440000
Albuquerque	35°14'22.0"N	106°26'26.2"W	2	140	220000
Albuquerque	35°32'23.0"N	106°38'21.2"W	3	150	250000

Un set di dati: la raccolta dei record (un dataset) assume la forma di un'unica tabella

Statistica descrittiva: fornisce indicatori di sintesi per **identificare**, con un unico valore, le **proprietà statistiche** di una popolazione rispetto ad una singola variabile

Indicatori di Centralità

Media aritmetica: utile per sostituire dati mancanti o errati senza modificarne la distribuzione complessiva, ma il suo valore non è ricavato dai dati disponibili ed è **sensibile alle anomalie**.

Sia **X** una variabile **numerica** del nostro set di dati (non possiamo estrarre la media da valori categorici), **n** è il **numero di record** nella nostra popolazione e $\boldsymbol{X_i}$ è il i-esimo record: $\boldsymbol{media} = \frac{\sum_{i=1}^{n} X_i}{n}$

City	Latitude	Longitude	Bedrooms	SquareMeters	Price
Los Angeles	33°51'37.0"N	118°08'22.2"W	3	130	**???**

- **Proprietà:** supponiamo di avere un record con un dato mancante (il prezzo). Per mantenere il record senza influenzare la variabile media si possono **sostituire i dati mancanti con la media** per quella variabile (non cambierà la media aritmetica per l'intero set di dati)

Mediana: valore effettivo dalle osservazioni, resistente alle anomalie, ma necessita di dati ordinali.

Data una popolazione di valori ordinati (es. la colonna "SquareMeters" ordinata per il suo valore), la *mediana* è il valore in posizione centrale ($X_{\frac{n}{2}}$ = 91)

- **Proprietà: indicatore robusto** (anomalie come valori molto grandi o molto piccoli non influiscono molto sul valore mediano).
  - Considera il seguente esempio: la mediana è ancora 4 (valore in posizione centrale) mentre la media è passata da 4 a 14,6!
    - 2,3,3,4,5,6,6 {Mediana:4; Media:4}
    - 2,3,3,4,5,6,80 {Mediana:4; Media:14,6}

Moda: valore effettivo, il più frequente. Robusto alle anomalie, non necessita di dati ordinali e può essere applicato su variabili categoriali.

Dato un insieme di valori di una variabile (ad esempio: 1,2,4,2,5,3,2,2,3,4,1,3,4,2,6,2,1,3,1,2), la moda è il valore con **frequenza** maggiore nell'insieme delle osservazioni, ovvero il "2", e si calcola contando le occorrenze di un valore: "1" viene ripetuto 4 volte, "2" viene ripetuto 7 volte, "3" viene ripetuto 4 volte, ...

- **Proprietà:** a differenza della media e della mediana, la moda ha senso anche sui dati **categoriali**: mode(Rome, Rome, Los Angeles, Albuquerque): Rome. Sebbene robusto per **anomalie** come la mediana, ha senso anche quando **non c'è** un **ordine lineare** sui possibili valori.

- In un sistema di **votazione** (ad esempio un insieme di molti classificatori diversi) la moda determina il risultato finale vincente.

Indicatori di Variazione

Deviazione al quadrato: misura la differenza tra ogni valore x_i e la media delle osservazioni $\bar{x}$**:** $\boldsymbol{dev = \sum_{i=1}^{n}(x_i - \bar{x})^2}$

- Più i valori sono lontani dalla media, maggiore è la deviazione ed è **influenzata dal numero di osservazioni**: più valori abbiamo, più alta è la deviazione.

Varianza (spesso rappresentata con s^2, σ^2 o Var): normalizza la deviazione al quadrato per il numero di osservazioni: $\boldsymbol{s^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 = \frac{1}{n}dev}$

Confronto: usiamo lo stesso esempio sui tre indicatori di variazione per osservare la loro differenza (la media è 15): **(4,6,10,40)**

- **Deviazione quadrata:** dev(4,6,10,40) = $\boldsymbol{\sum_{i=1}^{n}(x_i - 15)^2}$= 852
- **Varianza:** Var(4,6,10,40) = 852/4 = 213
- **Deviazione standard:** Stdev(4,6,10,40) = 14,6

Indicatori variabili singoli

- **Minimo** (min): è il valore minimo nelle osservazioni
- **Massimo** (max): è il valore massimo delle osservazioni
- **Range**: è la differenza tra il valore massimo e il valore minimo

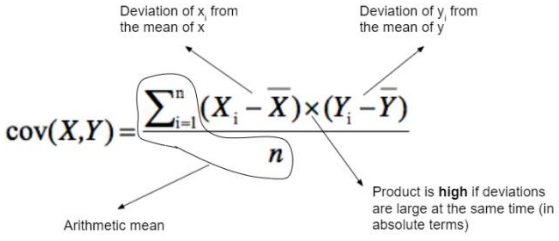
Indicatori a variabili multiple

Nella statistica descrittiva, le misure di associazione consentono di descrivere la **relazione tra due variabili**, cercando associazioni. Ad esempio, sono utili per determinare: se il **prezzo** delle case è associato ai **metri quadri**; se il **fumo** è associato a **malattie cardiache**; se il **budget** sulla pubblicità è associato al numero di **vendite.** Vedremo due misure: **covarianza** e **correlazione**

Misure di associazione

Covarianza: misura la forza della **relazione tra due variabili X e Y**. In particolare, è la **media** dei **prodotti** degli **scostamenti** dei valori: $\boldsymbol{cov(X,Y) = \frac{\sum_{i=1}^{n}(Y_i - \bar{Y}) * (Y_i - \bar{Y})}{n}}$

Significato della covarianza: la somma dei prodotti delle due deviazioni è alta se, ogni volta che X



devia dalla sua media, anche Y devia dalla sua media di **conseguenza**.

- Se Y non si discosta, la sua deviazione è bassa e così sarà il prodotto
- Se Y devia casualmente la somma si "annulla"
- **Direttamente proporzionale:** una covarianza **positiva** significa che X e Y deviano nella **stessa direzione**

- **Inversamente proporzionale:** una covarianza **negativa** significa che X e Y deviano in **direzioni opposte**

Un **problema** con la **covarianza** è che il suo valore è influenzato dall'**unità di misura**: se i valori sono grandi, la covarianza tende ad essere grande e se i valori sono piccoli, la covarianza tende ad essere piccola. Ad esempio, dato lo **stesso insieme** di prezzi, possiamo **diminuire** la covarianza di un fattore 1000, semplicemente esprimendo il prezzo come **migliaia di $**! Per superare il problema dell'unità di misura utilizziamo la **correlazione**.

Correlazione: divide la covarianza per il prodotto delle due deviazioni standard assicurando un valore compreso tra **-1** e **1.** Produce un risultato **indipendente dall'unità di misura**, perché tiene conto delle deviazioni standard di X e Y

$$\boldsymbol{Corr(X,Y) = \frac{Cov(X,Y)}{Stdev(X) * Stdev(Y)}}$$

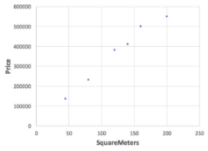
Significato di correlazione

- **Inversamente proporzionale:** un valore di correlazione è prossimo a **-1** se le due variabili tendono a variare in senso opposto
- **Direttamente proporzionale:** un valore di correlazione è prossimo a **1** se le due variabili tendono a variare nella stessa direzione
- **Relazioni lineari** (i punti formano una retta nel piano XY)**:** un valore di correlazione è vicino a 0 se le due variabili hanno variazioni indipendenti
- La correlazione può essere 0 anche se esiste una relazione forte ma **non lineare** tra X e Y.

Grafico a dispersione e regressione

Oltre a calcolare le misure di associazione, è utile visualizzare le coppie di valori delle due variabili in un piano XY.

SquareMeters	Price
120	380,000
200	550,000
80	250,000
160	500,000
45	135,000
140	410,000



Mentre le misure di associazione ci dicono **se** esiste un'associazione, i modelli di **regressione** stimano la **funzione effettiva** che mette in relazione le due variabili.

Data Warehouse: database operativo

Supporta le **applicazioni software** e ha grandi **repository** che consolidano i dati provenienti da **diverse** fonti (interne ed esterne all'organizzazione)

Viene **aggiornato offline** in tempo reale: aggiungere, modificare o eliminare dati

Segue il **modello dati multidimensionale**

Ha bisogno di query **complesse** per eseguire l'analisi e l'esplorazione dei dati: operazioni di **join** costose dal punto di vista computazionale (per unire i dati in un'unica tabella)

Implementato utilizzando **schemi a stella o fiocco di neve** per supportare in modo **efficiente** le **query OLAP**

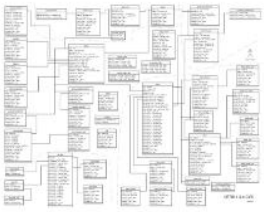
Potrebbe non contenere **dati storici** (non ne ha bisogno)

Processo decisionale basato sui dati

I database operativi vanno più che bene per i dati software, ma in un contesto **aziendale**, i dati dovrebbero anche diventare una ricca fonte di **informazioni**:

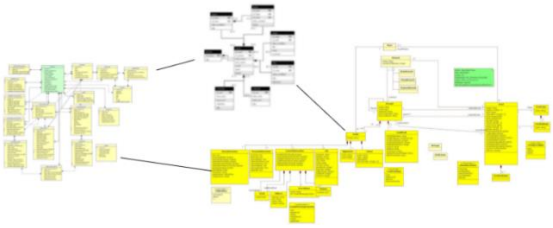
- Rappresenta la situazione e il comportamento aziendale
- Può supportare **decisioni informate** per i decisori aziendali

Processo decisionale da un database

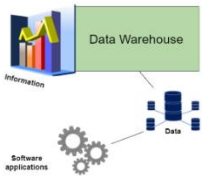


I decisori non possono **beneficiare** direttamente dei dati di un database senza una conoscenza approfondita e tecnica del **modello di dati**

Integrazione dei dati



Spesso i decisori vogliono **mettere insieme** dati da diversi database operativi (ad esempio i dati di vendita con dati pubblicitari). Integrare diversi database operativi è **estremamente** difficile: hanno definizioni e contenuti dei dati differenti



Data Warehouse (DW): raccolta di dati **orientati al soggetto**, **integrati**, non volatili e variabili nel tempo per supportare le **decisioni** di gestione

Proprietà del Data Warehouse

- **Orientato al soggetto:** si rivolge a uno o più soggetti di analisi in base alle esigenze analitiche dei manager ai vari livelli del processo decisionale
- **Integrato:** il contenuto risulta dall'integrazione di dati provenienti da vari sistemi operativi ed esterni
- **Non volatile: accumula** dati dai sistemi operativi per un lungo periodo di tempo, la **modifica e la rimozione non sono consentite,** l'unica operazione è l'eliminazione dei dati non più necessari e obsoleti
- **Variabile nel tempo:** tiene traccia di come i suoi dati si sono **evoluti nel tempo**, ad esempio può consentire di conoscere l'evoluzione delle vendite o delle scorte negli ultimi mesi/anni

Progettazione di banche dati

1. **Specifica dei requisiti:** le esigenze degli utenti vengono raccolte per creare uno schema di database
2. **Progettazione concettuale:** descrive i dati con il modello entità-relazione (ER)
3. **Progettazione logica:** paradigma di implementazione per il modello relazionale di applicazioni database
4. **Progettazione fisica:** implementazione specifica su DBMS

Il risultato finale è un **database relazionale**: altamente normalizzato per garantire la coerenza con aggiornamenti frequenti, solitamente ottenuto a un costo di query più elevato (la normalizzazione produce più tabelle).

Progettazione di un Data Warehouse

Il paradigma relazionale **non** è **appropriato** per i Data Warehouse: abbiamo bisogno di buone prestazioni per query complesse nell'analisi dei dati e di una tecnica di progettazione che:

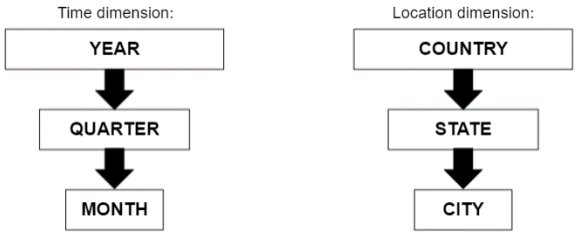
- Ha lo scopo di supportare le domande degli **utenti finali** (responsabili decisionali)
- È orientata alla **comprensibilità** dei risultati, senza la necessità di conoscere uno schema di dati complesso

- È orientata alle **prestazioni**, la ridondanza è accettata se porta un vantaggio in termini di prestazioni, è richiesto un grado inferiore di normalizzazione

Il paradigma utilizzato per progettare un Data Warehouse si chiama **modellazione multidimensionale** (o modellazione dimensionale)

Modellazione multidimensionale (DM): dati costituiti da **fatti** collegati alle **dimensioni**

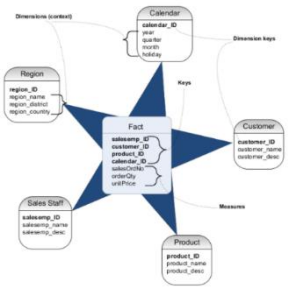
- Un **fatto** rappresenta il fulcro dell'analisi, l'oggetto principale che ci interessa (ad esempio nell'analisi delle vendite nei negozi, il fatto è la vendita).
- Le **misure** quantificano i fatti: le misure sono valori **numerici** (ad esempio l'importo delle vendite, il numero di prodotti, ecc.)
- Le **dimensioni** vengono utilizzate per visualizzare le misure da diverse prospettive, ad esempio:
  - **Dimensione temporale**: analizza i cambiamenti nelle vendite in periodi di tempo specifici
  - **Dimensione della posizione**: analizza le vendite in base alla distribuzione geografica dei negozi
  - Le dimensioni includono attributi che formano **gerarchie**, che consentono agli utenti decisionali di esplorare misure a vari **livelli di dettaglio**
  - **L'aggregazione** delle misure si verifica ogni volta che viene attraversata una gerarchia, ad esempio, lo spostamento da un mese all'altro aggregherà i valori delle vendite per ogni anno

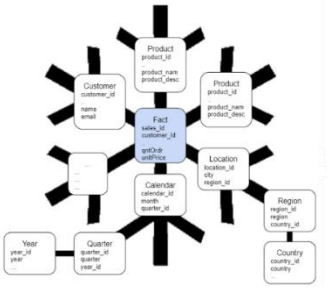


La modellazione multidimensionale è un modello concettuale per Data Warehouse. A livello **logico**, il modello multidimensionale è solitamente rappresentato da tabelle relazionali organizzate in **schemi a stella** e **schemi a fiocco di neve**. Questi schemi mettono al centro la tabella dei fatti, collegata a diverse tabelle dimensionali

Schema a stella

Ridondanza: utilizzano una **tabella univoca per ogni dimensione**, anche in presenza di gerarchie (tabella de-normalizzata)



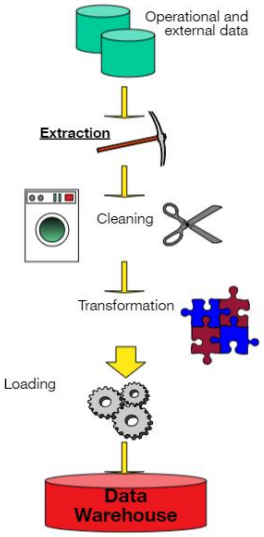


Schema a fiocco di neve

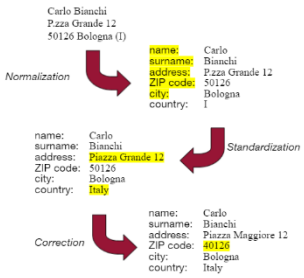
Nessuna ridondanza: utilizzano **più** tabelle **normalizzate** le gerarchie di una dimensione

Popolazione di un DW: ETL (Estrazione, Trasformazione e Caricamento)

È un processo cruciale per il successo di un progetto di Data Warehousing, richiede molti sforzi (l'ETL è l'80% del costo totale DW) e passa attraverso 3 passaggi:



1. **Estrazione:** i dati rilevanti vengono **selezionati** in base alla loro qualità ed **estratti** dalla fonte eterogenea interna o esterna all'organizzazione
   - Estrazione **statica:** viene eseguita quando il DW deve essere popolato per la prima volta e consiste nel fare una "istantanea" dei dati operativi.
   - Estrazione **incrementale:** utilizzata per aggiornare il DW e cattura solo le modifiche rispetto all'ultima estrazione (basato sul registro DBMS e sul timestamp)
2. **Trasformazione:** converte i dati dal formato operativo al formato standardizzato del DW. La corrispondenza con il livello della sorgente è resa più complessa dall'eterogeneità delle diverse sorgenti, che richiede una complessa fase di integrazione (dati che nascondono informazioni importanti e con formati diversi)



   a. **Pulizia: rimuove errori** e incongruenze nei dati e li **converte** in un formato standardizzato (dati duplicati, uso improprio di un campo o valori mancanti, impossibili, errati e incoerenti per la stessa entità o tra valori associati a causa di standard diversi o per errori di battitura)
   b. **Integrazione**: riconcilia i dati da diverse fonti di dati, sia a livello di schema che a livello di dati
   c. **Aggregazione**: riepiloga i dati ottenuti dalle origini dati in base alla granularità del Data Warehouse
3. **Caricamento:** alimenta il DW con i dati trasformati, incluso l'aggiornamento, ovvero la propagazione degli aggiornamenti dalle origini dati al Data Warehouse con una frequenza specificata

Sistemi OLTP: database operativi (elaborazione delle transazioni online)

Sistemi di **database tradizionali** progettati e ottimizzati per supportare **operazioni** quotidiane, come l'accesso rapido e simultaneo ai dati, l'elaborazione delle **transazioni** e il controllo della concorrenza avendo dati di aggiornamento online coerenti

Caratteristiche dei dati del DB **OLTP**:

- Dati **dettagliati**
- **Non includono** dati **storici**
- Altamente **normalizzati**
- Scarse prestazioni su query complesse, tra cui **join** un'aggregazione

L'analisi dei dati richiede un **nuovo paradigma**: **OLAP**

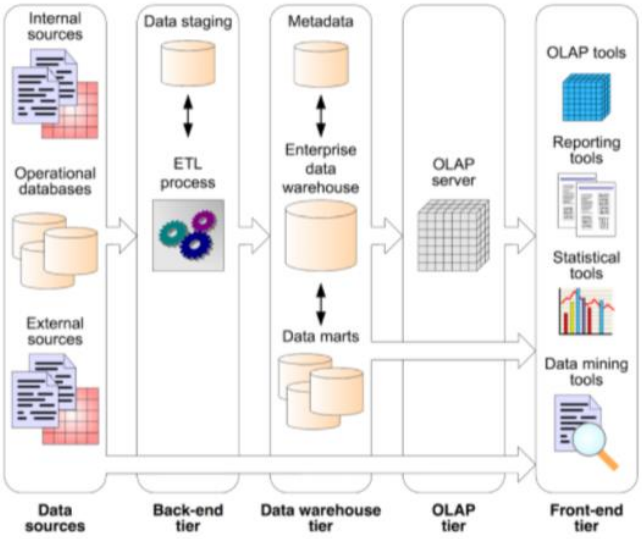
Sistemi OLAP (elaborazione analitica online)

- OLAP si concentra sulle **query analitiche**, dove la ricostruzione dei dati richiede un numero elevato di join (tabelle de normalizzate in OLAP)
- I database OLAP supportano un **carico** di query **pesante**
- Tecniche di indicizzazione OLTP non efficienti in OLAP: orientate ad accedere a pochi record
- Le query OLAP in genere includono l'**aggregazione**

OLTP vs OLAP

	OLTP	OLAP
User type	Operators, office employees	Managers, executives
Usage	Predictable, repetitive	Ad hoc, nonstructured
Data content	Current, detailed data	Historical, summarized data
Data organization	According to operational needs	According to analysis needs
Data structures	Optimized for small transactions	Optimized for complex queries
Access frequency	High	From medium to low
Access type	Read, insert, update, delete	Read, append only
Number of records per access	Few	Many
Response time	Short	Can be long
Concurrency level	High	Low
Lock utilization	Needed	Not needed
Update frequency	High	None
Data redundancy	Low (normalized tables)	High (denormalized tables)
Data modeling	UML, ER model	Multidimensional model

Architettura con più Livelli



L'origine dati (Data sources) non è un livello di un DW, ma più una raccolta di dati di diverso tipo (Data Lake)

Livello Back-end composto da:

- **Strumenti di estrazione, di trasformazione e di caricamento** (ETL): alimentano i dati nel DW da database operativi e origini dati interne ed esterne
- **Data staging** (Data-store Operativo): **database** intermedio in cui vengono eseguiti tutti i processi di **modifica** dei dati prima del caricamento nel DW

Livello Data Warehouse composto da:

- **Data Warehouse aziendale** centralizzato e che comprende un'intera organizzazione
- Diversi **Data mart:** DW di dipartimenti specializzati
- **Metadati**
    - **Metadati Aziendali:** descrivono la **semantica** dei dati e le **regole** organizzative, le politiche e i vincoli relativi ai dati, le **informazioni di sicurezza** (autorizzazione dell'utente e controllo degli accessi) e le informazioni di monitoraggio (statistiche di utilizzo, segnalazioni di errori, audit trail). Descrivono le **fonti dei dati**: schemi, proprietà, frequenze di aggiornamento, limitazioni legali, metodi di accesso
    - **Metadati Tecnici:** descrivono l'**archiviazione** dei dati nel sistema informatico e le **applicazioni** e i **processi** che manipolano i dati (descrivono l'**ETL**: derivazione dei dati, estrazione dei dati, pulizia, regole di trasformazione, ecc). Descrivono anche la **struttura**

del DW e dei Data mart, a livello concettuale/logico (fatti, dimensioni, gerarchie, ...) e a livello fisico (indici, partizioni, ...)
    - **Repository:** memorizza informazioni sul DW e sui suoi contenuti

Livello OLAP composto da un server che fornisce una vista multidimensionale dei dati, indipendentemente dall'effettivo modo in cui i dati vengono archiviati

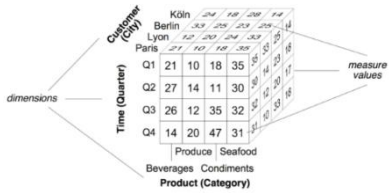
Livello Front-end: utilizzato per l'analisi e la visualizzazione dei dati. Contiene strumenti client che consentono agli utenti di sfruttare il contenuto del DW

- **Strumenti OLAP**: consentono l'esplorazione e la manipolazione interattiva dei dati di magazzino e la formulazione di complesse query ad hoc
- **Strumenti di reporting:** consentono la produzione, la consegna e la gestione dei report, che possono essere cartacei, interattivi o basati sul web. I report utilizzano **query predefinite** che richiedono informazioni specifiche in un formato specifico, eseguite regolarmente
- **Strumenti statistici**: utilizzati per analizzare e visualizzare i dati del cubo utilizzando metodi statistici
- **Strumenti di data mining:** consentono agli utenti di analizzare i dati per scoprire preziose conoscenze come i modelli e le tendenze e consentono anche di fare previsioni basate sui dati attuali

Design: modello multidimensionale (MultiDim)

Il modello multidimensionale visualizza i dati in uno spazio **n-dimensionale**: un cubo di dati composto da **dimensioni** (**prospettive** per analizzare i dati) e **fatti**

MultiDim: Esempio



Dimensioni (Cliente, Tempo, Prodotto)**:** gli **attributi** descrivono le **dimensioni;** la dimensione del prodotto può avere attributi ProductNumber e UnitPrice

Fatti (Vendite)**: ogni cella** (cubetto) **del cubo dati rappresenta la quantità di unità vendute per categoria, trimestre e città del cliente**

Misure (Quantità)**: valori numerici associati** alle celle o fatti di un cubo di dati

Le **istanze** di una dimensione sono chiamate **membri** (frutti di mare e bevande sono membri del prodotto nella categoria di granularità)

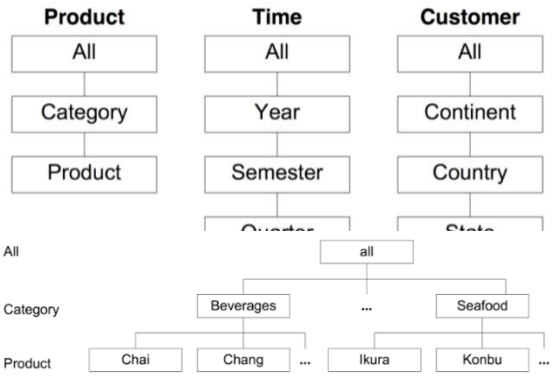
Un **cubo di dati** contiene diverse **misure**, ad es. Importo, che indica l'importo totale delle vendite. Un cubo di dati può essere **sparso** (caso tipico) o **denso**: non tutti i clienti potrebbero aver ordinato prodotti di tutte le categorie durante tutti i trimestri

Granularità dei dati: livello di dettaglio al quale vengono rappresentate le misure per ogni dimensione del cubo

Gerarchie: consentono di visualizzare i dati con **diverse granularità**

- Definire mappature che mettono in relazione concetti dettagliati di **livello inferiore (figlio)** con quelli di **livello superiore (genitore)**
- **Schema della dimensione:** la struttura gerarchica di una dimensione

- Un'**istanza di dimensione** comprende **tutti i membri a tutti i livelli** in una dimensione
- Nell'esempio, la granularità di ogni dimensione: Categoria per la dimensione **Prodotto**, Trimestre per **Tempo** e Città per **Cliente**
- Potremmo volere dati di vendita con una **granularità più fine** (Mese) o con una **granularità più grossolana** (Paese)



Nello stesso esempio, possiamo guardare i **membri** (istanze) **della gerarchia** Prodotto (Prodotto -> Categoria -> Tutto)

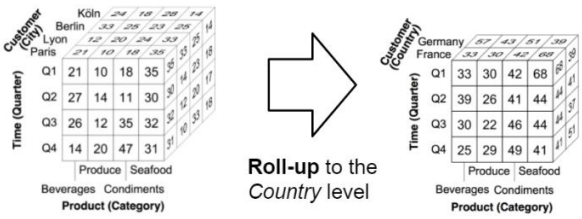
Misure e dimensioni: l'aggregazione delle misure modifica il livello di astrazione al quale vengono visualizzati i dati in un cubo. Le misure possono essere:

- **Additive**: può essere riassunta usando l'addizione lungo tutte le dimensioni (più comune)
- **Semi additive**: può essere riassunta usando l'addizione lungo alcune dimensioni. Ad esempio: quantità di inventario, che non può essere aggiunta lungo la dimensione Tempo
- **Non additiva**: non può essere riassunta in modo significativo utilizzando l'addizione attraverso qualsiasi dimensione. Ad esempio: prezzo dell'articolo, costo per unità e tasso di cambio
- **Distributive:** definite da una funzione di aggregazione che può essere calcolata in modo distribuito. Le funzioni **conteggio, somma, minimo** e **massimo** sono distributive, il **conteggio distinto no**
    - Es. S = {3,3,4,5,8,4,7,3,8} partizionato in sottoinsiemi {3,3,4}, {5,8,4}, {7,3,8} dà un risultato di 8, mentre la risposta sul set originale è 5
- **Algebriche:** definite da una funzione di aggregazione che può essere espressa come una funzione scalare di quelle distributive. Ad esempio: **media**, calcolata dividendo la somma per il conteggio
- **Olistiche:** non possono essere calcolate da altri sotto-aggregati (ad es. mediana, rango)

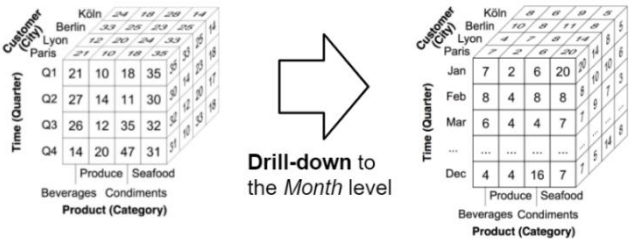
Query OLAP

Operazioni OLAP: prendendo lo stesso esempio, vediamo una serie di **operazioni** OLAP che possiamo applicare per **esplorare il cubo** delle vendite (in migliaia) per categoria di prodotto e città dei clienti per il 2003

Roll-up: calcoliamo le quantità di vendita per Paese, un'operazione di roll-up al livello Paese lungo la dimensione Cliente. Dopo il roll-up possiamo esaminare le misure a livello di Paese. Notiamo che le vendite di prodotti ittici in Francia per il primo trimestre sono significativamente più elevate. Per scoprire se ciò si è verificato durante un particolare mese del trimestre Q1, riportiamo il cubo al livello di aggregazione della città ed effettuiamo il **drill-down** lungo il livello Tempo fino al livello del mese...



Drill-down: si passa da una granularità più grossolana (trimestre) a una granularità mensile

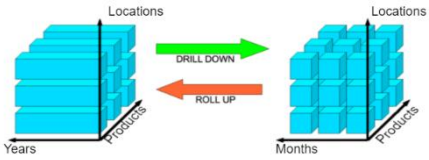


Esempio Roll-up/Drill-down in SQL

Roll-up:

```
SELECT SUM(F.vendite)
FROM Time T JOIN Fatto F JOI..
GROUP BY T.year, L.region, P.brand
WHERE T.year = 2018
```

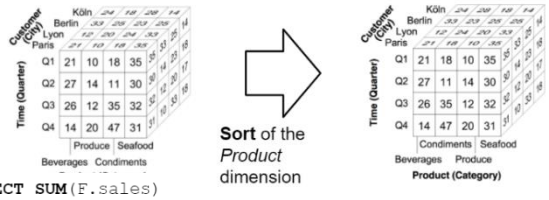
I prodotti sono ora in ordine casuale; vogliamo visualizzare il cubo originale con il prodotto ordinato...



Drill-down:

```
SELECT SUM(F.vendite)
FROM Time T JOIN Fatto F JOIN Product P JOIN Location L
GROUP BY T.month, L.region, P.brand
WHERE T.year = 2003
```

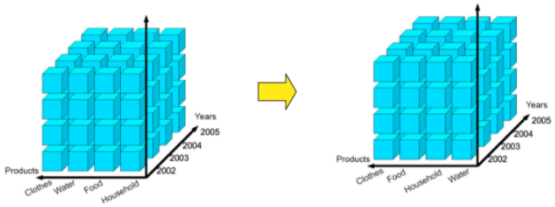
Sort: ordinando la dimensione Prodotto; i prodotti, a livello di Categoria, sono ordinati alfabeticamente.



Esempio Sort in SQL

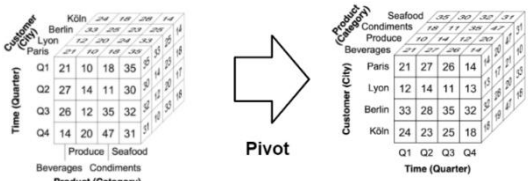
```
SELECT SUM(F.sales)
FROM Time T JOIN Fact F
JOIN Products P
JOIN Locations L
```

```
SELECT SUM(F.sales)
FROM Time T JOIN Fact F
JOIN Products P
JOIN Locations L
ORDER BY P.category
```



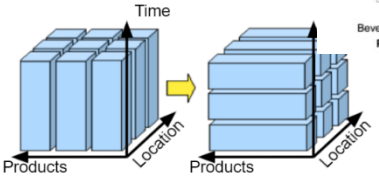
Per introdurre l'operazione successiva, supponiamo di voler ruotare il cubo per vedere la dimensione Tempo sull'asse X...

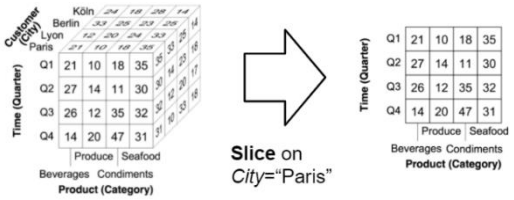
Pivot: utilizzando l'operazione **pivot**, la dimensione Tempo è ora sull'**asse X**, la dimensione Cliente è ora sull'**asse Y**, mentre la dimensione Prodotto è sull'**asse Z**.



Esempio Pivot in SQL

```
SELECT SUM(F.sales)
FROM Time T JOIN Fact F
JOIN Products P
JOIN Locations L
GROUP BY P.category
```

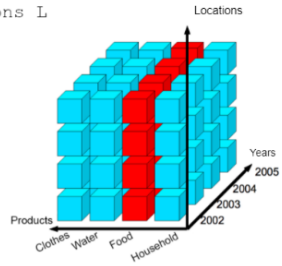




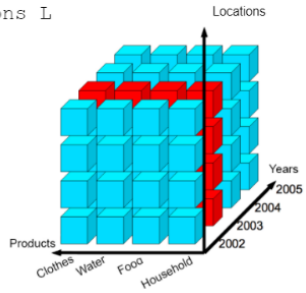
Slice: l'operazione slice visualizza i dati solo per Parigi, risultando in un "sotto-cubo" bidimensionale

Esempio Slice in SQL

```
SELECT F.sales
FROM Time T JOIN Fact F
JOIN Products P JOIN Locations L
WHERE P.category=”Food”
```

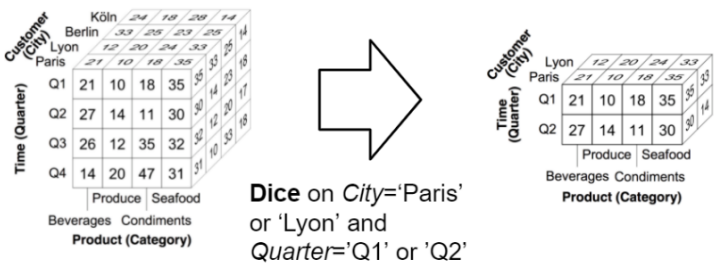



```
SELECT F.sales
FROM Time T JOIN Fact F
JOIN Products P JOIN Locations L
WHERE T.years=2003
```



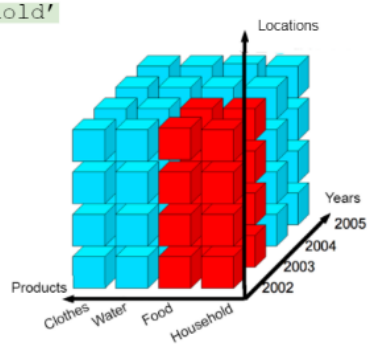
Slice consente di mostrare solo un singolo valore per una singola dimensione. E se volessimo selezionare più valori da dimensioni diverse?

Dice: con i **dadi** possiamo selezionare su più condizioni e più dimensioni, producendo un sotto-cubo tridimensionale.



Dice in SQL: esempio

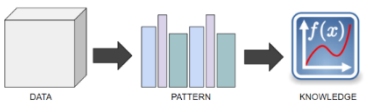
```
SELECT F.sales
FROM Time T JOIN Fact F
JOIN Products P JOIN Locations L
WHERE T.years<2003
AND P.category LIKE 'Food'
OR P.category LIKE 'Household'
```



Data Mining

"Il data mining è il processo informatico di scoperta di modelli in grandi set di dati che coinvolgono metodi all'intersezione tra **machine learning**, **statistiche** e **sistemi di database**". L'obiettivo è l'estrazione di **modelli** e **conoscenze** da una **grande quantità** di dati

Modelli: regolarità nei dati che si ripetono per tutte le osservazioni in modo prevedibile. **Esempi** di modelli:



- Il prezzo delle case in € è circa 2500 volte i loro metri quadrati
- Spesso si comprano latte e cereali insieme
- È stato osservato che le persone acquistano oggetti digitali in questo ordine: Personal Computer, fotocamera digitale e scheda di memoria

Conoscenza: una volta scoperto un modello, possiamo **comprendere** la logica dietro il modello, in modo da poter **descrivere** il modello e **prevedere** fenomeni simili su un dato argomento. **Esempi** di conoscenza scoperta:

- Come vengono fissati i prezzi delle case a seconda dei metri quadrati
- Quali prodotti dovrebbero essere messi più vicini al supermercato
- In quale ordine gli oggetti digitali dovrebbero essere pubblicizzati

Grande quantità di dati: per trovare schemi e dedurre conoscenza, dobbiamo partire da diverse **osservazioni**, fornite sotto forma di esempi di **dati** omogenei (sono necessarie **grandi quantità** di dati per estrarre modelli o conoscenze statisticamente significative):

- Molti esempi dei prezzi delle case e delle relative dimensioni
- Molti esempi di transazione di generi alimentari con l'insieme degli articoli acquistati
- Tanti esempi di sequenze di acquisti nel reparto digitale di un ipermercato.

Estrazione dati: passaggi

1. **Definisci il problema:** l'obiettivo che stiamo cercando di raggiungere e la conoscenza che vorremmo estrarre
2. **Identifica i dati richiesti:** in questo passaggio raccogliamo e comprendiamo i dati di qui abbiamo bisogno per perseguire l'obiettivo.
3. **Prepara e preelabora:** seleziona e pulisci i dati richiesti. Formatta i dati in modo che siano l'input di un algoritmo di apprendimento automatico.
4. **Modella l'ipotesi:** seleziona algoritmi di apprendimento automatico e regola i parametri per costruire un modello "predittivo" accurato.
5. **Addestra e testa:** addestra algoritmi utilizzando un campione di dati, testalo su dati non visti.
6. **Verifica e distribuisci:** verifica il modello finale con le parti interessate (utenti finali), prepara la visualizzazione e distribuisci

Machine Learning (ML)

È al centro del **data mining**, perché è qui che avviene l'**astrazione** dai dati ai **modelli**. Date molte osservazioni, una funzione con un obiettivo specifico (es. indovinare il prezzo di una casa viste le sue caratteristiche) viene "**appresa**" automaticamente dai dati.

- I dati possono essere **molto grandi** (molti record) e altamente dimensionali (molte variabili per record).
- Le relazioni **non lineari** tra le variabili possono essere molto difficili da individuare.
- La codifica manuale di alcuni algoritmi può essere incredibilmente **difficile**

Esempio: abbiamo i dati di 10mila case negli Stati Uniti.

City	Latitude	Longitude	Rooms	SQM	Price
Los Angeles	33°51'37.0"N	118°08'22.2"W	3	130	420000
Los Angeles	33°50'17.7"N	118°09'12.6"W	2	60	380000
...	...	...	...	...	...
Albuquerque	35°14'22.0"N	106°26'26.2"W	2	140	220000
Albuquerque	35°32'23.0"N	106°38'21.2"W	3	150	250000

Obiettivo: apprendere una funzione (modello) che possa dedurre il Prezzo date tutte le altre variabili, per case non presenti nel dataset di 10mila

Probabilmente, Approssimativamente Corretto (P.A.C.)

City	Latitude	Longitude	Rooms	SQM		Price
Los Angeles	33°49'32.3"N	118°08'44.1"W	5	230	→	?

- **Probabilmente**: il modello del mondo reale astratto utilizzando l'apprendimento automatico **dovrebbe essere corretto** con alta probabilità.
- **Approssimativamente**: il modello dovrebbe avere un errore di **generalizzazione** basso, il che significa che dovrebbe **approssimare il caso generale** ed essere accurato con dati non visti.

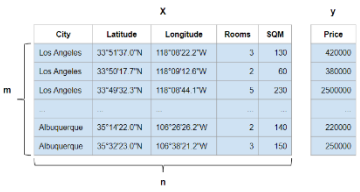
Essendo basati su statistiche, i modelli e le conoscenze estratte nel data mining non sono quasi **mai accurati al 100%:** potrebbero essere coinvolte molte variabili che non sono nei dati (ad esempio un venditore di casa potrebbe voler vendere rapidamente la casa a un prezzo inferiore) e i dati provenienti dal mondo reale sono influenzati da rumore e casualità (ad es. valore del prezzo digitato in modo errato)

ML: nozioni comuni

- **Esempio**: un'osservazione, come i dati di una singola casa.
- **Variabili di input**: le variabili che vengono fornite in input per ogni esempio (es. i metri quadrati).
- **Variabile di output**: la variabile che vogliamo dedurre in base alle variabili di input (es. il prezzo della casa).
- **Ipotesi** (alias modello): la funzione dalle variabili di input alla variabile di output appresa da un algoritmo di apprendimento automatico.
- **Etichetta**: nei task supervisionati, è il valore della variabile di output per una determinata istanza di input. È dato dai dati ed è considerato il vero valore.
- **Previsione**: è il valore della variabile di output "indovinata" dalla funzione ipotesi, dovrebbe essere corretto con alta probabilità (il tempo potrebbe non essere affatto coinvolto).

Notazione

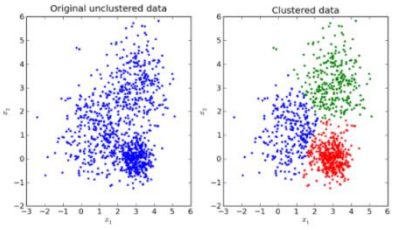
- Lettera maiuscola **X:** matrice dei dati di input
- Lettera minuscola **y:** vettore della variabile di uscita
- Lettera minuscola **m:** numero di esempi
- Lettera minuscola **n:** numero di variabili degli esempi



Tecniche supervisionate: viene fornito un "training **set" di dati** all'algoritmo di apprendimento automatico (con un'etichetta per ogni esempio) e viene presa una **variabile** come verità fondamentale, che rappresenta **l'output per una serie di esempi** (**esempio**: viene fornito il prezzo effettivo, insieme alle variabili di input). L'apprendimento supervisionato ha due compiti a seconda del tipo di variabile di output:

- **Regressione: la variabile di output è un valore numerico**. Dato un set di dati di test, etichettato con il prezzo effettivo invisibile per l'algoritmo di apprendimento, è facile misurare **l'accuratezza** o **l'errore**. Ad esempio, nel problema della previsione del prezzo della casa.
- **Classificazione: la variabile di output è una classe**.
  - **Classificazione binaria**: l'output può essere 0 o 1 (esempio: la casa è di Los Angeles o no).
  - **Classificazione multi-classe:** quando le classi sono più di due e **l'output è un elemento di un insieme finito** (esempio: da quale città proviene la casa, tra queste 20 città?)

Tecniche non supervisionate: non c'è un valore della variabile di output, o non c'è **nessuna variabile di output**. A volte **non ci sono obiettivi prefissati** a priori e si vuole solo scoprire regolarità e relazioni all'interno dei dati.



- **Clustering**: divide gli esempi in diversi gruppi (cluster). Gli esempi nello stesso gruppo dovrebbero essere simili, mentre gli esempi in gruppi diversi dovrebbero essere dissimili (attività più comune).
    - Ad esempio, la nostra osservazione ha due variabili: x1 e x2. **Non c'è nessuna verità fondamentale**, né etichette sui dati. La variabile di output è "la classe a cui appartiene il punto dati" ma non è fornita nell'addestramento. **Non è facile misurare la precisione**: senza una verità fondamentale la correttezza è “negli occhi di chi guarda”.
- **Regole di associazione**: relazioni tra variabili. Per esempio. {cipolla, patate} => {hamburger} nelle vendite dei supermercati
- **Rilevamento anomalie:** trova gli "outlier" in una serie di esempi. Ad esempio, da un registro delle transazioni bancarie trova quelli sospetti.
- **Modelli generativi:** dati molti esempi generano una nuova istanza "sintetica" con caratteristiche simili. Per esempio: generazione di melodie, scrittura automatica di poesie/libri.
- **Estrazione delle caratteristiche:** riduce il numero di variabili generando nuove caratteristiche caratterizzanti.

Interdisciplinarità

Uno dei motivi del successo del data mining e del machine learning sono i campi di **applicazione** virtualmente illimitati: traduzione automatica, diagnosi medica, sistemi di raccomandazione (ad es. Amazon che suggerisce libri), riconoscimento vocale, recupero delle informazioni (imparare a classificare), computer vision (es. auto a guida autonoma), sicurezza informatica (sistemi antintrusione), commercio, previsioni del tempo, ecc.

Pretrattamento dei dati

Indipendentemente dall'obiettivo specifico, la fase di preelaborazione è **comune** a tutti i progetti di data mining. I **dati del mondo reale** sono generalmente:

- **Incompleti**: mancano i valori degli attributi, mancano alcuni attributi di interesse o contengono solo dati aggregati (ad esempio, occupazione = "")
- **Rumorosi**: contengono errori o outlier (ad esempio, stipendio = -100)
- **Incoerenti**: contengono discrepanze nel codice o nei nomi (ad esempio, Compleanno = 42 o sesso = maschio, incinta = sì)

La frase "**garbage in, garbage out**" è particolarmente applicabile nel data mining.

Fasi di preelaborazione

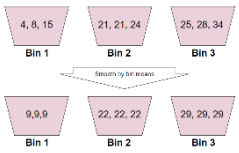
1. **Pulizia dei dati:** compila i valori mancanti, leviga i dati rumorosi, identifica o rimuove i valori anomali, risolve le incongruenze.
2. **Integrazione dei dati:** integrazione di più database, cubi di dati o file.
3. **Trasformazione dei dati:** testo e categorie devono essere "codificati" in vettori di numeri reali (es. bag of word, one-hot encoding), valori molto grandi o molto piccoli devono essere normalizzati, altrimenti gli algoritmi di ottimizzazione nei modelli di machine learning non convergeranno in tempi ragionevoli.

Pulizia dei dati: è il problema numero uno nel **DW** e nel **data mining**. Le **attività** comuni di pulizia dei dati nel data mining sono:

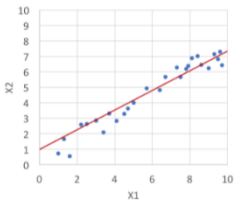
1. **Deduzione:** inserire i valori dei **dati mancanti**
   - Possono **provenire** dal valore dei dati "**non applicabile**" al momento della raccolta, da **considerazioni diverse** tra il momento in cui i dati sono stati raccolti e quando vengono analizzati o da **problemi umani, hardware o software.** Ad esempio, molte tuple non hanno alcun valore registrato per diversi attributi, come il reddito del cliente nei dati di vendita
   - I dati mancanti potrebbero essere dovuti a **malfunzionamenti** dell'apparecchiatura, **incoerenza** con altri dati registrati e quindi cancellati, a dati non inseriti per incomprensione o alcuni dati potrebbero non essere stati considerati importanti al momento dell'inserimento oppure non sono state registrare le modifiche alla cronologia dei dati
   - **Gestire i dati mancanti**
     1. **Ignora** la tupla: di solito eseguita quando manca l'etichetta della classe, non è efficace quando la percentuale di valori mancanti per attributo varia considerevolmente.
     2. Inserisci **manualmente** il valore mancante: noioso, ma potrebbe essere addirittura impossibile se il valore è totalmente sconosciuto.
     3. Compilalo **automaticamente** con:
        - **Una costante globale** (ad es. "sconosciuto")
        - La **media** dell'attributo per un campione o per tutti i campioni appartenenti alla **stessa classe** (ad es. sostituire il prezzo della casa mancante con il prezzo medio di tutte le case)
        - Il **valore più probabile**: basato sull'inferenza (regressione)
2. **Identificare** i valori **anomali** (errati), **livellare** i **dati rumorosi** (errori casuali o varianze in una variabile misurata) e rimuoverli
   - Possono **provenire** da strumenti di raccolta dati difettosi, errori umani o informatici durante l'immissione o la trasmissione dei dati, incoerenza nella convenzione di denominazione
   - **Gestire dati rumorosi**: i computer rilevano i valori sospetti e l'uomo li controlla
     1. **Binning**: liscia un valore di dati ordinato consultando il suo "vicinato" (i valori intorno ad esso). Prima ordina i dati e li partiziona in contenitori di uguale frequenza e poi liscia con mezzi bin.
     2. **Regressione**: uniforma inserendo i dati nelle funzioni di regressione
     3. **Raggruppamento**: rileva e rimuovi gli outlier
3. Correlare i **dati incoerenti:** possono provenire da diverse fonti di dati o da violazioni della dipendenza funzionale (esempio: modifica di dati collegati)
4. Risolvere la **ridondanza** causata dall'integrazione dei dati (**record duplicati**)

Denoising dei dati (rimuovere il rumore)

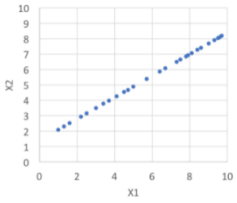
Levigatura per contenitori (binning): una volta popolati i bin, il valore viene sostituito con valori aggregati. Poiché i metodi di raggruppamento considerano solo i valori nello stesso bin, eseguono il livellamento locale.



Regressione lineare: ricerca della **linea** "migliore" per adattare due attributi in modo che un attributo possa essere utilizzato per prevedere l'altro. La regressione potrebbe essere **l'obiettivo finale** di un progetto di data mining, ma può anche essere utilizzata **a priori** nel progetto per **lisciare** i dati.

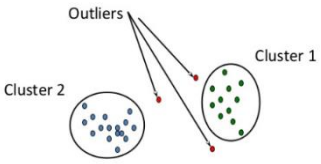


1) Regression



2) Smoothing, moving data points to the regression line

Clustering: i valori simili sono organizzati in gruppi (cluster). I **valori che cadono al di fuori** dei cluster possono essere considerati outlier (anomalie). Come per il metodo di regressione lineare, il clustering può essere anche il compito ultimo di un progetto di data mining, ma potrebbe essere applicato nel compito di preelaborazione per migliorare la qualità dei dati.



Quando si pensa al **rumore dei dati** dobbiamo tenere a mente i compiti finali e le loro differenze:

- **Data Warehousing**: il prossimo passo è il caricamento nel **DW**. La gestione dei dati rumorosi nella fase di preelaborazione è imperativo perché il decisore non avrà mezzi per eliminare il rumore dei dati.
- **Data Mining**: il prossimo passo è il **machine learning**. I modelli di machine learning (anche i più semplici come le regressioni logistiche) hanno mezzi per il denoising dei dati attraverso tecniche di "regolarizzazione".

Suggerimento pratico: costruire un primo modello senza denoising, quindi ripetere i passaggi di formazione con passaggi di preelaborazione.

Integrazione dei dati: abbina entità e attributi tra diverse fonti di dati.

Problema di identificazione dell'entità: nella tabella A il codice identificativo del cliente è denominato A.customer_id, mentre nella tabella B il codice identificativo del cliente è denominato B.cust_numb. Come può un analista di dati o un computer essere sicuro che customer_id su un database e cust_number in un altro facciano riferimento allo stesso attributo?

Schema A

customer_id	birth	city
109238	07/12/87	Rome
113125	23/08/89	London
159483	28/11/90	Paris
198828	22/12/92	Bristol

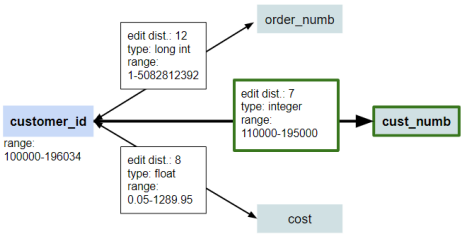
Schema B

order_numb	cust_numb	cost
4982812389	113125	34.50
4982812390	113125	110.02
4982812391	151514	98.49
4982812392	129827	32.40

Soluzione per l'identificazione dell'entità

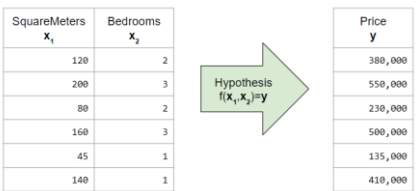
Utilizziamo i **metadati** degli attributi, ad esempio:

- **Nome dell'attributo**: possiamo usare le distanze sulla stringa come la distanza di modifica per misurare quanto due nomi sono simili
- **Tipo di dati**: gli attributi sono entrambe stringhe?
- **Intervallo di valori**: se un attributo contiene valori in migliaia e gli altri valori compresi tra - 1 e 1, probabilmente stiamo osservando due cose diverse.
- Selezioniamo cust_numb perché ha la **distanza di modifica minima**, lo **stesso tipo di dati** e **l'intervallo di valori più simile**.
- **Non è possibile** determinare in modo completamente automatico le corrispondenze (con una precisione del 100%).



Trasformazione dei dati: codifica i dati categorici e di testo in vettori numerici e ridimensiona i valori in intervalli fissi.

Recall: l'ipotesi è la funzione modellata da algoritmi di apprendimento automatico, dalle variabili di input alla variabile di output (ad esempio una linea di regressione che si adatta a punti dati 2D). Questa funzione è una funzione matematica che accetta **valori numerici in input** e restituisce uno o più valori numerici.



Perché abbiamo bisogno di trasformazione (problema di codifica): cosa succede se alcune variabili dei nostri dati **non** sono **numeriche**? Per esempio, potrebbero essere valori **categorici** (città), valore **booleani** (sì/no) come o valore di **testo libero**

(Appartamento molto carino e accogliente vicino al centro commerciale Centrum, quartiere amichevole).

Problema di scala: valori numerici molto grandi non sono buoni in ML. I modelli di apprendimento automatico si basano su tecniche di **ottimizzazione numerica** che riducono al minimo l'errore tra l'ipotesi ei dati di addestramento. Gli algoritmi di ottimizzazione **convergono più velocemente** con i valori normalizzati (ad es. tra 0 e 1 o tra - 1 e 1).

Soluzione di codifica per il testo

Vettorizzazione: come abbiamo visto in Information Retrieval, il **testo libero** può essere rappresentato (codificato) come un **vettore di variabili numeriche** (valori reali)

- **Bag-Of-Words** (BOW): ogni variabile del vettore è per una parola specifica e il valore di una variabile x è 1 se la **parola i-esima** dell'insieme di tutte le parole è presente nel testo, 0 altrimenti. Questa è una rappresentazione "**scarsa**": il vettore risultante per un testo ha la maggior parte delle variabili impostate a 0, mentre solo le parole presenti nel testo sono impostate a 1.

"Appartamento molto carino con vista meravigliosa".

...	apartment	close	cozy	flat	floors	friendly	great	mall	view	...
...	0	0	0	1	0	0	0	0	1	...

Soluzione di codifica per categorie

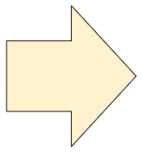
Codifica a caldo: simile alla codifica BOW, ma con la differenza che vale solo **una categoria** alla volta (es. una casa non può essere sia a N.Y. che a L.A.) e che non c'è bisogno di tokenizzare (dividere le stringhe in parole) o rimuovere le stop-word, perchè ogni valore univoco avrà la sua variabile (una variabile per ogni città). **Solo una** delle variabili generate è attiva (**calda**), mentre tutte le altre sono impostate su 0.

Città: New York

...	Albuquerque	Boston	New York	Los Angeles	Miami	Las Vegas	San Francisco	...
...	0	0	1	0	0	0	0	...

$$\text{scale(value)} = \frac{value - \min(values)}{\max(values) - \min(values)}$$

SQM	Price
130	420000
60	80000
180	500000
140	220000
150	250000



SQM	Price
0.583	0.80952
0	0
1	1
0.666	0.333
0.75	0.404

Ridimensionamento (normalizzazione): migliora prestazioni e precisione della maggior parte dei modelli di machine learning. L'intervallo di scala più comune è [0, 1], può essere ottenuto con il **ridimensionamento lineare**