

Basi di Dati Attive

Università degli Studi del Sannio
Facoltà di Ingegneria
Corso di Laurea in Ingegneria Informatica

Corso di Basi di Dati
Anno Accademico 2005/2006
docente: ing. Corrado Aaron Visaggio

email: visaggio@unisannio.it

ricevimento: mercoledì 11.00-13.00.

Corrado Aaron Visaggio

1

Intro...

Una base di dati si dice **attiva** quando dispone di un sottosistema integrato per definire e gestire le **Regole di produzione** o **Regole attive**: Evento->Condizione-> Azione.

L'esecuzione delle regole è controllata dal **Rule Engine** che cattura gli eventi ed esegue le azioni: Base di dati con **comportamento reattivo**.

Indipendenza della conoscenza: parte della logica di business viene estratta dall'applicazione e introdotta nella base di dati.

Quasi tutte le basi di dati possono essere considerate attive, dal momento che mettono a disposizione i **trigger**.

I trigger si basano sul paradigma Evento-Condizione-Azione:

- Gli **eventi** sono primitive per la **manipolazione di dati** in SQL
- La **condizione** è un **predicato booleano** espresso in sql
- L'**azione** è una **sequenza di primitive SQL** generiche.

Corrado Aaron Visaggio

2

...Intro

Un trigger è:

- **Attivato** da un evento
- **Valutato** durante la verifica della condizione
- **Eseguito** quando viene lanciata l'azione.

Due livelli di **granularità**:

- **Row-level**: l'attivazione avviene ad per ogni tupla della relazione
- **Statement -level**: l'attivazione a livello di primitiva SQL.

Due livelli di **esecuzione**:

- **Immediata**: la valutazione avviene **subito dopo l'evento** che li ha attivati
- **Differita**: la valutazione avviene **successivamente al commit-work**.

Caratteristiche delle regole attive

Gli **eventi** possono includere **eventi temporali** (in corrispondenza di un istante temporale preciso) o **applicativi** (generati da programmi utente).

Le **condizioni** possono essere **espressioni booleane di eventi** che utilizzano le precedenze tra eventi e la **congiunzione di eventi**.

L'**azione** può essere eseguita al posto dell'evento (**instead of**)

La **valutazione o l'esecuzione delle regole** può essere **detached**, ovvero avvenire in transazioni differenti da quella che riguarda il trigger.

I **conflitti tra regole** possono essere evitati indicando delle **priorità esplicite**.

Le regole possono essere organizzate in **gruppi** e ciascun gruppo po' essere attivato o disattivato.

Proprietà delle Regole Attive

Un insieme di regole garantisce la **terminazione** quando, per ogni transazione che scatena l'esecuzione delle regole, **tale esecuzione genera lo stato finale**.

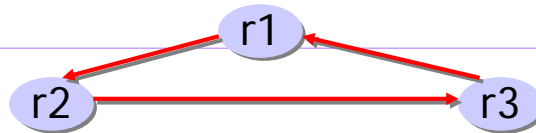
Un insieme di regole garantisce la **confluenza**, quando per ogni transazione che scatena l'esecuzione delle regole, **l'esecuzione termina producendo un unico stato finale**.

Un insieme di regole garantisce il **determinismo** delle osservazioni quando, per ogni transazione che scatena l'esecuzione delle regole, tale esecuzione è **confluente** e **tutte le azioni visibili dalle regole sono identiche e prodotte nello stesso ordine**.

Il processo di analisi delle regole serve a verificare che, a tempo di creazione, le proprietà valgono per un insieme di regole.

Il **grafo di attivazione non deve essere ciclico**:

- Un nodo è una regola.
- La coda dell'arco ha un'azione che coincide con l'evento della testa.



Corrado Aaron Visaggio

5

Basi di dati ad oggetti

Nelle **basi di dati ad oggetti** ogni entità del mondo reale è rappresentata da un **oggetto**

- Componenti Elettronici-CAD
- Componenti Meccanici-CAM
- Specifiche e programmi-CASE
- Dati multimediali
- Dati Spaziali o geografici – GIS

E' necessario fornire una **gestione unitaria** dell'oggetto, che in un RDBM è **disperso** in N tavole. La gestione dei dati come oggetti consente di:

- Definire **relazioni semantiche** tra i dati
- Integrare dati ed operazioni (**tipo di dato astratto**)
- Integrazione stretta con i **linguaggi di programmazione ad oggetti**.

Corrado Aaron Visaggio

6

Tipi...

Gli oggetti hanno :

- **Proprietà statiche:** che descrivono la **struttura** degli oggetti.
 - La parte statica usa dei **costruttori di tipo** e **tipi di dati atomici** (booleano, interi, reali, stringhe, identificatori di oggetto, nil).
- **Proprietà dinamiche:** che descrivono le **operazioni applicabili** agli oggetti.

I tipi di dati complessi definiscono la struttura delle istanze (oggetti complessi):

- Il **costruttore record** consente di costruire tipi le cui istanze sono tuple: $T = \text{record-of}(A_1:T_1, \dots, A_N:T_N)$
- I **costruttori di insieme**, multi-insieme, e tupla, consentono di costruire collezioni di valori complessi dello stesso tipo:
 - **Set**, collezioni non ordinate, prive di duplicati
 - **Bag**, collezioni non ordinate che possono presentare elementi duplicati
 - **List**, collezioni ordinate che possono presentare duplicati.

Se **T** è un **tipo complesso**, un oggetto che ha per tipo **T** si dice **istanza di T**.

... Tipi

Una **definizione di tipo** inizia, di solito, con un costruttore di tipo record.

I costruttori consentono di rappresentare la **complessità strutturale** di un oggetto.

La notazione punto consente di accedere alle singole parti del tipo.

V1.Targa= "Mi678RE"

V1.Costruttore.Stabilimento="FIAT"

V1.PartiMeccaniche.Motore="Mercedes"

```
Automobile: record-of (
  Targa: string,
  Modello: string,
  Costruttore: record-of(
    Nome: String,
    Presidente: String,
    Stabilimento: set-of (
      Record-of (
        Nome:String,
        Città: String,
        Addetti: Integer)
    Colore: String,
    Prezzo: Integer,
    PartiMeccaniche: record-of(
      Motore: String,
      Ammortizzatore: String))
  )
)
```



Oggetti e Valori

I riferimenti fra oggetti (OID) consentono di evitare la ripetizione delle definizioni all'interno dei tipi complessi.

La parte strutturale di un oggetto diventa un insieme di coppie (OID, valore), ove il valore è un'istanza dell'oggetto.

Se una proprietà di un oggetto ha tipo *T si dice che essa ha per valore un oggetto – è object-valued.

Automobile: record-of (
 Targa: string,
 Modello: string,
 Costruttore: *Costruttore,
 Colore: String,
 Prezzo: Integer,
 PartiMeccaniche: record-of(
 Motore: String,
 Ammortizzatore: String))

Costruttore: record-of(
 Nome: String,
 Presidente: String,
 Stabilimento: set-of
 (*Stabilimento)

Stabilimento: Record-of (
 Nome:String,
 Città: String,
 Addetti: Integer)



Identità e uguaglianza...

L'uso di OID consente anche di garantire che due oggetti distinti abbiano lo stesso stato e differiscano solo per l'OID.

Due oggetti sono identici quando hanno lo stesso identificatore e lo stesso stato.

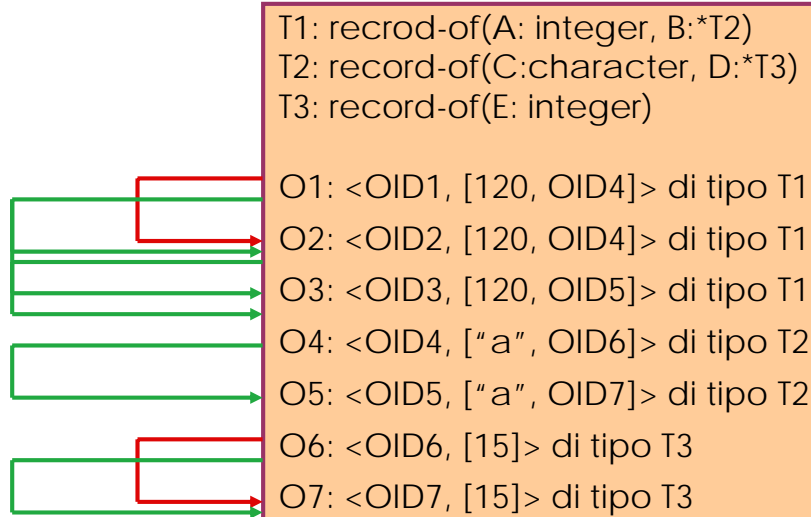
Uguaglianza superficiale: due oggetti hanno lo stesso stato [esistono gli operatori che verificano tale uguaglianza]

Uguaglianza profonda: due oggetti hanno stessi OID ma non necessariamente lo stesso stato [si verifica con opportuni programmi].

...Identità e uguaglianza

Uguaglianza
Superficiale

Uguaglianza
Profonda



Classi

Una classe raccoglie oggetti dello stesso tipo, che possono essere dinamicamente aggiunti o tolti alla classe. Gli oggetti che appartengono alla stessa classe sono omogenei, cioè dotati dello stesso tipo. Una classe comprende:

- L'interfaccia
- L'implementazione.

Le basi di dati espongono ai loro utenti la struttura dello stato degli oggetti



L'estensione consente di inserire oggetti dello stesso tipo in collezioni differenti e dare nomi distinti a queste collezioni.

I metodi sono usati per manipolare gli oggetti di un OODBMS: questi sono la principale innovazione di un OODBMS, rispetto ad un RDBMS. E' composto da:

- Signature
- Implementation.

Metodi...

Ciascun metodo è associato con una specifica classe di oggetti (**target**) o con più classi (**multitarget**).

Ogni metodo può presentare più **parametri di ingresso**, ma un solo **parametro di uscita**.

Quattro categorie di metodi:

- **Costruttori**, costruiscono gli oggetti a partire dai parametri di ingresso.
- **Distruttori**, cancellano gli oggetti.
- **Accessori**, restituiscono informazioni sul contenuto degli oggetti.
- **Trasformatori**, cambiano il contenuto degli oggetti.

...Metodi...

Scritto in Co2

Definizione metodo

Self è l'oggetto ricevente

Corpo dell'implementazione

```
Add method init (
    Targa_par:string,
    Modello_par: String,
    Colore_par: string,
    Prezzo_par: integer): automobile
In class Automobile is public

Body init(Targa_par:string, Modello_par:
String, Colore_par: string,
Prezzo_par: integer): Automobile in
class Automobile

Co2{ self -> Targa=Targa_par;
Self -> Modello = Modello_par;
Self->Prezzo= Prezzo_Par;
Self->Colore=Colore_par;
Return (self);} $
```



...Metodi

Mismatch di impedenza. E' il problema di far dialogare linguaggi di programmazione che contiene variabili scalari con il linguaggio SQL che elabora tuple.

Negli OODBMS il passaggio persistenza-memoria temporanea può essere gestito in modo trasparente [ortogonale]. Attualmente gli OODBMS hanno **accesso associativo** ai dati.

Criteri di progettazione dei metodi.

- Brevi
- Coerenti
- Separare politiche con implementazioni
- Generalizzare
- Information hiding
- ereditarietà



Gerarchie di Generalizzazione...

Le gerarchie di generalizzazione tra classi equivalgono a quelle tra oggetti; garantiscono la **complessità semantica**:

- Tutti gli oggetti della sotto-classe appartengono alla super-classe
- Tutte le proprietà ed i metodi della super-classe sono ereditati dalle sotto-classes. Si può modificare l'implementazione di un metodo senza modificare l'interfaccia.
- Le sotto-classes possono essere estese con nuove proprietà e metodi

Le generalizzazioni godono della proprietà transitiva.

```
Add class AutoSportiva
inherits Automobile
type tuple(MaxVelocità:integer
Pilota:persona)
```

```
Add class AutoStorica
inherits Automobile
type tuple(AnnoCostr:integer)
```

...Gerarchie di Generalizzazione...

```
Execute co2{
  o2 Autostorica X;
  X = new(AutoStorica);
  [X init ("Mi5653", "Ferrari", "rossa", 150.000];
  X -> AnnoCostr=1957;}%
```

Migrazione tra classi. Un oggetto può modificare il suo stato da una super-classe ad una sotto-classe [generalizzazione <- specializzazione ->].

Un oggetto è **istanza** di una classe solo se è la classe più specializzata della gerarchia. Invece è **membro** di tutte le sue superclassi.

In alcuni OODBMS un oggetto può essere istanza di più classi, in altre solo di una.

...Gerarchie di Generalizzazione...

Ereditarietà Multipla. Una classe può, in alcuni sistemi, ereditare da più super-classi.

```
Add class AutoStoricaSportiva
  inherits AutoSportiva, Autostorica
  type tuple(GareVinte:integer)
```

Le istanze della classe AutoStoricaSportiva sono membri di Automobile, AutoStorica e di AutoSportiva.

Le istanze della classe AutoSportiva o AutoStorica che non appartengono ad AutoStoricaSportiva, sono membri della classe Automobile.

Esistono istanze della classe Automobile che non appartengono né ad AutoSportiva né ad AutoStorica.

...Gerarchie di Generalizzazione...

Conflitti. La ereditarietà multipla potrebbe dare origine al problema dei conflitti di nome.

Politiche per la risoluzione del conflitto:

- Rilevare il conflitto all'atto della definizione delle classi e non accettare come corrette le definizioni [problema: ridisegno in fasi avanzate]
- Definire meccanismi per togliere ambiguità alla scelta [problema: aumenta la complessità dell'ODL]
- Ridefinire le proprietà ed i metodi localmente [problema: aggravio di sforzo sul progetto]

...Gerarchie di Generalizzazione...

Persistenza. Gli oggetti persistenti vengono scritti nell'OODB, mentre quelli temporanei cessano di esistere con il programma. Una classe diviene persistente tramite:

- La definizione di **tipo persistente**
- La raggiungibilità a partire da un altro oggetto persistente [garantisce l'integrità referenziale ma viene cancellato solo quando non è più raggiungibile.]
- La **denominazione** per ritrovarlo ad una successiva invocazione del programma.

Non tutti questi meccanismi sono, però, supportati

...Gerarchie di Generalizzazione...

Ridefinizione di metodi. E' possibile ridefinire i metodi all'interno di una gerarchia di classi (**override**). La scelta del metodo da invocare dipende dal tipo di oggetto [**late binding**, scelta del codice a tempo di esecuzione].

E' possibile, anche avere l'**overloading** dei metodi, ovvero metodi che mantengono lo stesso nome, ma modificano i parametri insieme al comportamento.

Ridefinizioni con raffinamenti di tipo. E' possibile modificare l'interfaccia, introducendo meccanismi di sotto-tipazione: T1 è sottotipo di T2 se ammette tipi più specifici [a **livello elementare**].

Dato un tipo di record $T_x = [A_1:T_1, \dots, A_m:T_m]$ T_y è sottotipo di T_x se $T_y = [A_1:T'_1, \dots, A_m:T'_m, \underline{A_{m+1}:T'_{m+1}}, \dots, A_n:T'_n]$, con T'_i sottotipo di T_m [a **livello di record**]

...Gerarchie di Generalizzazione

- Sia C2 una classe che eredita da C1 e una generica proprietà A:T di C1. La **co-varianza delle proprietà** consente di attribuire alla proprietà A, ridefinita in C2, un sottotipo T' del tipo T.
- Sia C2 una classe che eredita da C1 e un generico metodo m con un certo numero di parametri di ingresso di tipo T_i ed un parametro di uscita T. La **co-varianza del parametro di uscita** si ottiene dando al metodo m' ridefinito in C2 un sotto-tipo T' di T.
 - Se per ogni parametro di ingresso di m, T'_i è un sotto-tipo di T_i si ha la **co-varianza dei parametri di ingresso**.
 - Se per ogni argomento di ingresso di m, T_i è un sottotipo di T'_i si ha la **contro-varianza dei parametri di ingresso**.

La covarianza di proprietà e di parametri di ingresso non garantisce la corretta sostituzione dei valori a tempo di compilazione



Manifesto delle Basi di dati ad Oggetti

Proprietà obbligatorie:

- Complessità strutturale
- Identità degli oggetti
- Incapsulamento
- Tipizzazione
- Definizione di gerarchie
- Over-loading/riding dei metodi
- Completezza computazionale del linguaggio
- Estensibilità
- Persistenza
- Efficienza
- Accessi concorrenti
- Robustezza
- Linguaggio di interrogazione

ODL

ODL: linguaggio per la **definizione di schemi ad oggetti**. Descrive **tipi** e non classi ed è **indipendente dal linguaggio di programmazione** prescelto per la implementazione delle classi.

I riferimenti tra tipi vengono chiamate relazioni e sono bidirezionali



Estensione del tipo



In ODL si può **definire l'interfaccia** di un metodo nella definizione di tipo:

L'**implementazione** è realizzata tramite linguaggio di programmazione

```
Interface Automobile (
  Extent Automobili
  Key targa)
Attribute String Targa;
  Attribute String modello;
  Attribute String colore;
  Attribute Integer prezzo;
  Attribute Structure PartiMeccaniche
  {String motore;
  String Ammortizzatore};
  Relationship <Costruttore> Costruttore;
  Inverse Costruttore: :costruisce;}
Interface Costruttore {
  Attribute String Nome;
  Attribute String Presidente;
  Relationship Set<Automobile> Costruisce
  Inverse Automobile::Costruttore;
  Relationship set
  <Stabilimento>Stabilimenti
  Inverse Stabilimento::Costruttore;}
Interface Stabilimento {
  Attribute String Nome;
  Attribute String Città;
  Attribute Integer Addetti;
  Relationship <Costruttore> Costruttore
  Inverse Costruttore::Stabilimenti;}
```

Corrado Aaron Visaggio

25

Linguaggio di interrogazione...



RESEARCH CENTRE ON SOFTWARE TECHNOLOGY

Il linguaggio OQL è stato adeguato all'interrogazione di OODBMS.

- È un'estensione di SQL
- computazionalmente incompleto [non ha strutture di controllo]
- Consente l'invocazione di metodi
- Non consente la modifica dello stato degli oggetti



```
Select distinct x.Targa
From x in Automobile
Where x.colore="Rosso";
```

```
Select distinct x.Targa
From x in AutoSportivaStorica
Where x.colore="Rosso"
And "MilleMiglia34" in x.GareStoricheVinte
```

Corrado Aaron Visaggio

26

...Linguaggio di interrogazione...

L'OQL, a differenza di SQL, consente l'uso di **espressioni complesse** [path expression].

```
Select distinct x.Targa
From x in AutoSportivaStorica
Where x.Pilota.Nome="Fangio"
And "Maranello" in x.Costruttore.Stabilimenti.Nome
```

```
Select distinct x.Targa
From x in AutoSportivaStorica
Where x.Pilota=x.Costruttore.Presidente
```

```
Select distinct x.Pilota.Nome
From x in AutoSportivaStorica
Where x.Pilota.Nome=x.Costruttore.Presidente.Nome
```

...Linguaggio di interrogazione...

Come si sostituisce il join in OQL...

```
Select x.Targa
From x in AutoSportiva and c in Costruttore and s in Stabilimento
Where c=a.Costruttore and s=c.Stabilimenti
And s.Città<>"Maranello" and c.Nome="Ferrari" and
a.MaxVelocità>250
```

```
Select distinct Struct (Mod: x.Modello, col: x.Colore)
From x in AutoSportiva
Where "MilleMiglia39" in x.GareStoricheVinte
```

```
Select count (select x.Modello
From x in
(select y
From y in Automobile, z in Costruttore
Where z=y.Costruttore
And sum(z.Stabilimenti.Addetti)>4500)
```



...Linguaggio di interrogazione...

Esempi di raggruppamenti

Group a in Automobile
 By (Costr: a.Costruttore)
 With (NmrAuto: count(select x from x in partition))

Group a in Automobile
 By (veloci: a.MaxVel<200,
 Rapide:a.MaxVel>=200 and as.maxVel<250,
 Super : a.MaxVel>250)



ORDBMS...

Le basi di dati relazionali ad oggetti ORDBMS si pongono come obiettivo la realizzazione di estensioni compatibili della nozione classica di tabella presente in SQL-2.

Il modello di dati SQL-3 è compatibile con il modello dei dati relazionali del DDL SQL-2. Per poter definire una tabella, è necessario prima creare un **tipo di tupla** [definiscono la struttura da inserire nelle tabelle]

Creo il tipo tupla



```
Create row type PerType (
Nome varChar (30) not Null,
Residenza varChar(40),
CodFisc char(16) primary Key)
```

Creo le tavole



```
Create Table Persona of PerType
Create Table Pilota of PerType
```

...ORDBMS

In SQL-3 è possibile definire **gerarchie di tipo e di tabelle**, estendendo quelle già esistenti con nuove proprietà.

```
Create row type AutoStoricaType (
  AnnoCostr Integer)
Under AutoType
```

In SQL-3 è possibile definire anche **tipi di tupla astratti** che possono essere usati nella costruzione dei tipi di tupla.

I tipi astratti possono includere **insiemi di funzioni** [definite in SQL-3 o in un linguaggio di programmazione esterno].

```
Create type TipoPartiAuto (
  Motore char(10)
  Potenza integer
  Cilindrata integer
  Equlas Potenza,
  Greater than MaggiorPotenza)
```

31

Linguaggio di interrogazione in SQL-3

Il SQL-3 è compatibile con il SQL-2, per cui è possibile definire query relazionali standard su tabelle standard.

```
Select Nome
From Persona
Where Codfisc = '1234567890'
```

Deferenziamento. Tale operatore consente di accedere da un oggetto x ad un attributo A di un oggetto y referenziato in x.

```
Select Presidente->Nome
From Costruttore
Where Nome = 'Fiat'
```

Doppio Punto. Per l'accesso a componenti interne viene utilizzato il doppio punto.

```
Select Costruttore->Presidente->Nome
From Automobile
Where PartiMeccaniche..Motore = 'XY-wow!!!'
```



Attenzione: in OQL si usa solo la **notazione puntata**.

Linguaggio di interrogazione in SQL-3

Appiattimento. Si realizza trasferendo in strutture relazionali, cioè 'piatte', dati strutturati tramite costruttori di tipo, quali setof.

```
Select C.Nome, S.Città
From Costruttore as C, C.Stabilimento
as S
```

Annidamento (Nesting). Si realizza tramite l'operatore di group by e costruisce partizioni a pari valore dell'attributo di raggruppamento

```
Select Città, set(Nome)
From Costruttore
Group by Città
```

Basi di dati della terza generazione

Nasce da un articolo che definisce le **basi di dati di terza generazione** come naturale evoluzione delle basi di dati relazionali, che avevano sostituito le basi di dati gerarchiche e reticolari

Assunzioni di fondo sui **sistemi di terza generazione** :

- saranno in grado di gestire sistemi di **oggetti e regole** complessi
- dovranno essere **compatibili** con i sistemi di **seconda generazione**
- **Aperti all'interazione** con altri sistemi
- Sistema di **tipi** ricco
- Gerarchie di **generalizzazione tra tipi**
- Supporto di **funzioni** e **incapsulamento**
- Ricorrere agli **OID** solo in **assenza di chiavi** definite dall'utente
- Regole attive (**trigger**) e passive (**vincoli di integrità**)
- SQL è il linguaggio di riferimento

Basi di Dati Multimediali

Tipi di dati

Immagini. Il problema principale è il numero di bit utilizzati per la loro memorizzazione in formato binario: diversi paradigmi di compressione (GIF, JPEG, TIFF, PNG).

Audio. Un segnale audio è segmentato in piccole finestre temporali, al cui interno il segnale ha caratteristiche (ampiezza e frequenza) uniformi. Per 10 min fino a 100.000 finestre.

Video. UN video di 60 minu può contenere fino a 100.000 frame. MPEG-1 (non indicato alle riproduzioni televisive), MPEG-2 (richiede un numero elevato di bit), MPEG-4 (migliora ulteriormente la qualità).

Documenti. Testi+Immagini; si usano tipicamente linguaggi di mark-up.

Annotazioni. Testi liberi, talvolta scritti a mano, aggiunti ad altri dati multimediali.

Basi di Dati Multimediali

Interrogazioni

Questo è un problema specifico della disciplina di DB.

Un'interrogazione ad un database di immagini può richiedere l'estrazioni di immagini con caratteristiche definite.

Ancora più complessa è l'estrazioni di suoni che abbiano **pattern particolari**.

Il rationale che si segue, in questi casi, è l'**estrazione di sottoinsiemi** di oggetti candidati sui quali si eseguono **analisi di simiglianza**.

Tecniche adattative, pattern recognition, interpolazioni, motori inferenziali, information retrieval.

Basi di Dati Multimediali

Ricerca di documenti...

L' **information retrieval** è la tecnica che consente di estrarre informazioni dai documenti di testo [applicazioni esemplificative: motori di ricerca, filtri anti-spam, code cloning].

Questa tecnica si basa sul **calcolo dell'occorrenza** delle parole chiavi nel testo.

- Escludere le parole poco significative
- Ridurre le parole simili ad un'unica parola chiave
- Associare ad ogni parola chiave la propria frequenza

La **precisione** (precision) indica la percentuale di **documenti rilevanti nei documenti estratti**.

Il **richiamo** (recall) indica la percentuale di **documenti rilevanti estratti rispetto al totale dei documenti rilevanti** nella base di dati.

...Basi di Dati Multimediali

Ricerca di documenti

I documenti sono rappresentati da **matrici**.

Le strutture ad **indici invertiti** contengono come foglie delle parole chiave gli identificatori del documento (più comodo per ricerche su più parole chiave)

La **segnatura** è la lista ordinata delle parole chiave più frequenti nel documento.

	Parola chiave 1	Parola chiave 3	Parola chiave 4
id1	Freq1	Freq2	Freq3
id2	Freq1_2	Freq2_2	Freq3_2



Basi di Dati Multimediali Rappresentazione di dati Spaziali...

I dati spaziali descrivono l'informazione presente in uno spazio ad n dimensioni. Ha acquisito rilevanza specifica con i Geographical Information System (GIS).

E' necessario scegliere una struttura dati che rappresenti la posizione degli oggetti nello spazio.

Oltre alla posizione di un punto nello spazio, tali sistemi informatici devono essere in grado di descrivere altre informazioni relative a tali punti.

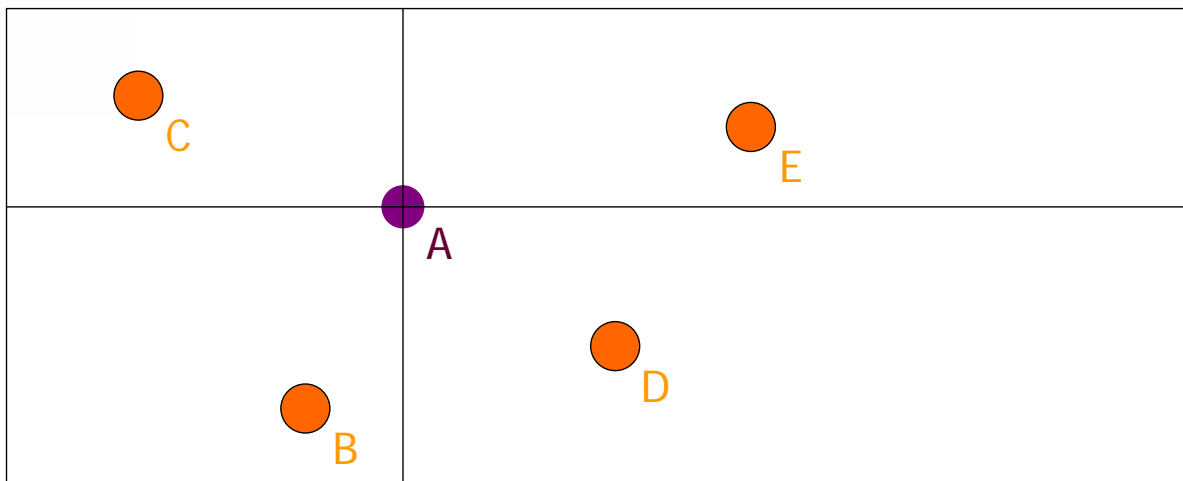
Bisogna esprimere anche proprietà spaziali, quali la contiguità e la distanza (in questo caso si utilizzano strutture ad albero).

- Nell'organizzazione ad albero bidimensionale (2d tree) ogni nodo ha due successori. I nodi figli sono a distanza pari (destra) o dispari (sinistra) dalla radice.



...Basi di Dati Multimediali Rappresentazione di dati Spaziali...

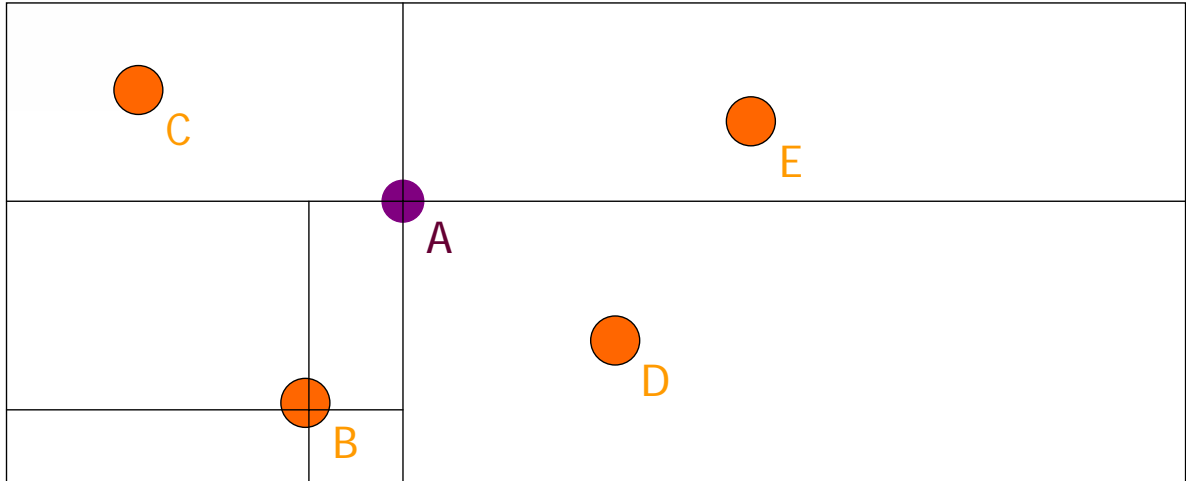
Nell'organizzazione a quad tree ogni nodo ha quattro successori.





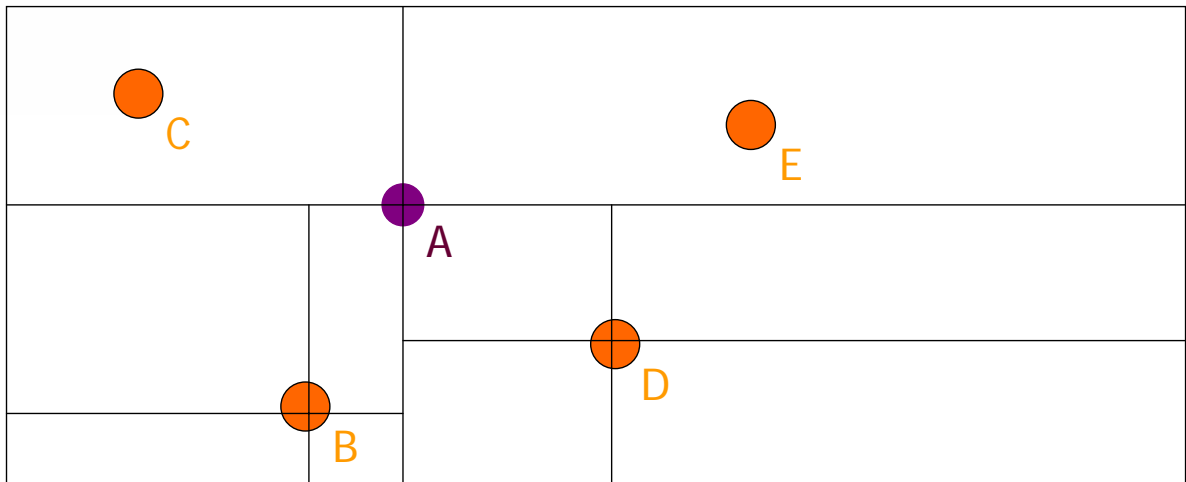
...Basi di Dati Multimediali Rappresentazione di dati Spaziali...

Nell'organizzazione a **quad tree** ogni nodo ha quattro successori.



...Basi di Dati Multimediali Rappresentazione di dati Spaziali...

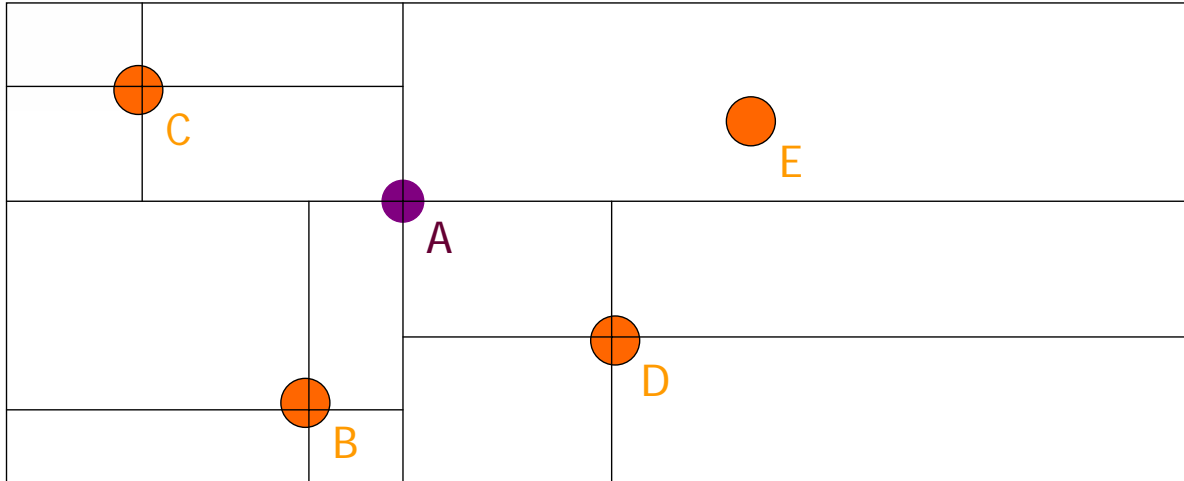
Nell'organizzazione a **quad tree** ogni nodo ha quattro successori.





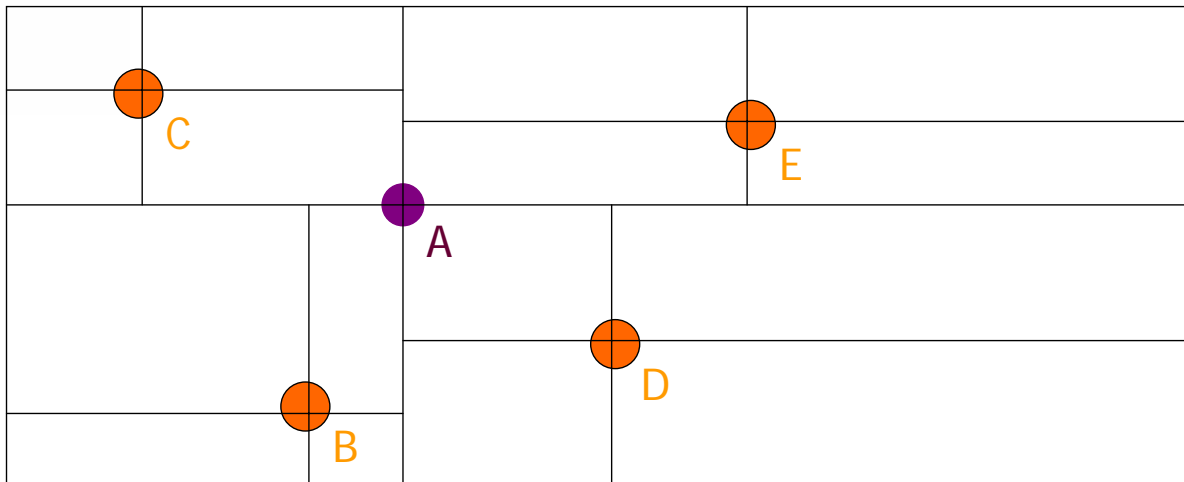
...Basi di Dati Multimediali Rappresentazione di dati Spaziali...

Nell'organizzazione a **quad tree** ogni nodo ha quattro successori.



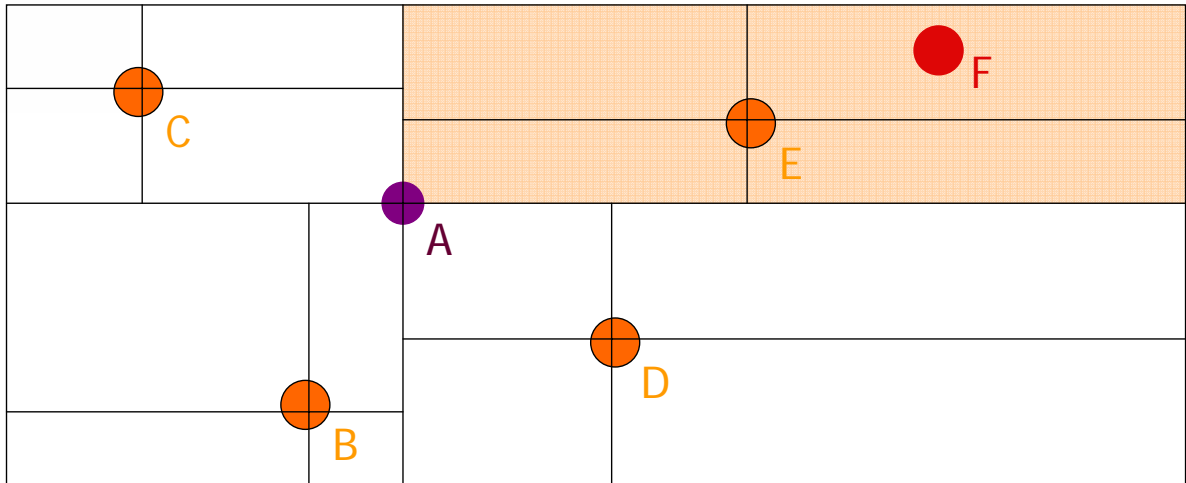
...Basi di Dati Multimediali Rappresentazione di dati Spaziali...

Nell'organizzazione a **quad tree** ogni nodo ha quattro successori.



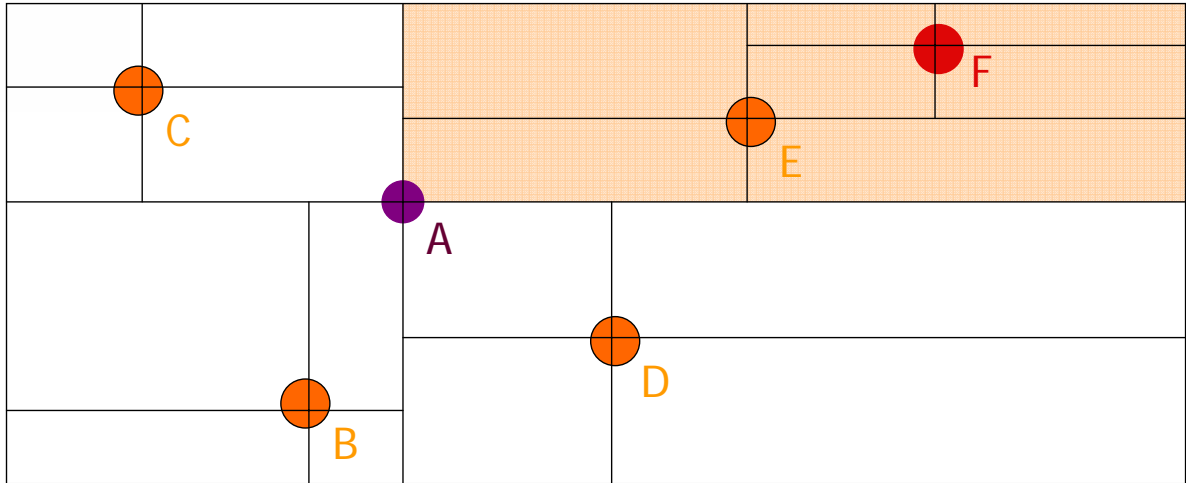
...Basi di Dati Multimediali Rappresentazione di dati Spaziali...

Nell'organizzazione a **quad tree** ogni nodo ha quattro successori.



...Basi di Dati Multimediali Rappresentazione di dati Spaziali...

Nell'organizzazione a **quad tree** ogni nodo ha quattro successori.



Transazioni

Sul piano transazionale, gli OODBMS offrono, oltre alle proprietà acide, meccanismi meno restrittivi:

- Nel **controllo di concorrenza**, sono definite operazioni di **check out** degli oggetti che consentono di caricare oggetti da un server alla workstation dell'utente, consentendo ad altri utenti di accedere allo stesso oggetto. La **check in** consente di caricare l'oggetto sul server.
- Un altro modo di garantire processi concorrenti cooperativi è basato sull'uso di **versioni**, ovvero varie copie dello stesso oggetto a istanti progressivi.
- In alcune applicazioni possono essere previste **transazioni di lunga durata** (quali il check in ed il check out di un oggetto).
- Le transazioni possono essere composte da **transazioni complesse**. Per esempio un client che deve colloquiare con differenti server; uno dei problemi più annosi è la gestione dell'atomicità.