I sistemi informativi aziendali: l'importanza delle informazioni

di Roberta Molinari

Dati e Informazioni

- Il termine **dato** significa letteralmente "*fatto*".
- I dati sono una rappresentazione dei fatti.
- Un dato costituisce un'informazione se fornisce una nuova conoscenza attraverso una chiave di interpretazione
- **L'informazione** è l'incremento di conoscenza che può essere acquisita dai dati.

es. il dato 12, senza chiave di interpretazione, non costituisce informazione. Con la chiave di interpretazione "Quantità disponibile" assume l'informazione di "quantità disponibile per un determinato articolo"

Le Informazioni

- In ogni sistema di vita dell'uomo vengono trattate **informazioni**.
- Costituiscono un grande patrimonio, sono considerate risorse preziosissime grazie alle quali lo stesso sistema umano sopravvive.
- Individuate e raccolte devono essere memorizzate in modo che si possano facilmente eseguire le operazioni **CRUD**:
  1. ***Create - AGGIUNGERE***
  2. ***Read - RECUPERARE***
  3. ***Update - MODIFICARE***
  4. ***Delete - CANCELLARE***

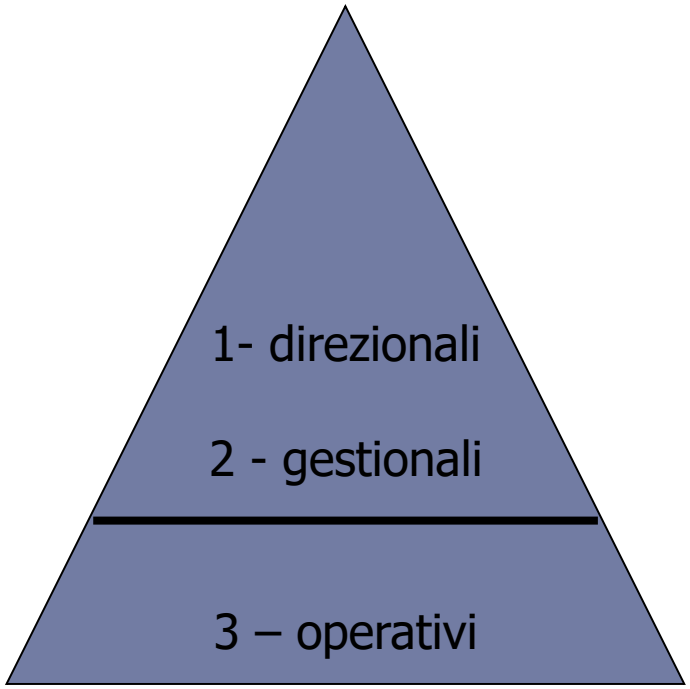
Azienda

- Ogni impresa si organizza in base:
  - alla sua **missione** o **mission** (lo scopo per cui è nata: produrre scarpe)
  - ai suoi **obiettivi generali** o **target** a breve o a lungo termine (aumentare il fatturato)
- Al suo interno si possono individuare:
  - **Risorse**: tutto ciò con cui opera (materiale, immateriale, persone, interne o esterne)
  - **Processi**: insieme di attività (decisioni e azioni) svolte per il raggiungimento di mission e target tramite un risultato definito e misurabile (prodotto)

Classificazioni dei processi Piramide di Anthony

Processi organizzativi

1. concorrono alla definizione degli obiettivi strategici Definiscono i piani a medio e lungo termine, progettano l'organizzazione dell'intera azienda (filiali) (decisioni strategiche)
2. concorrono alla traduzione degli obiettivi in criteri di gestione-programmazione ed effettuano il controllo del raggiungimento di tali obiettivi Controllano il corretto utilizzo delle risorse e definiscono piani a breve e medio termine (raggiungimento di un certo budget) (decisioni tattiche)



Processi operativi

3. concorrono all'attuazione concreta degli obiettivi (produzione delle scarpe)

Dati- Informazioni - Conoscenza

- Un caso molto particolare di risorsa su cui operano tutte le aziende è l'informazione. L'informazione è infatti una risorsa che riguarda tutte le altre risorse.
- I **dati** sono una materia prima in continua crescita
- Le **informazioni** sono il valore aggiunto ai dati
- Possedere la **conoscenza** è un'esigenza fondamentale per poter prendere decisioni

Sistema informativo aziendale (S.I.)

- Il **sistema informativo SI** è l'insieme delle persone, dei mezzi e delle procedure che riguardano la *raccolta*, la *produzione*, *l'archiviazione*, *l'elaborazione*, la *distribuzione* dei dati al fine di ottenere informazioni che servono da supporto alle funzioni operative, ai processi decisionali e al controllo di una organizzazione.

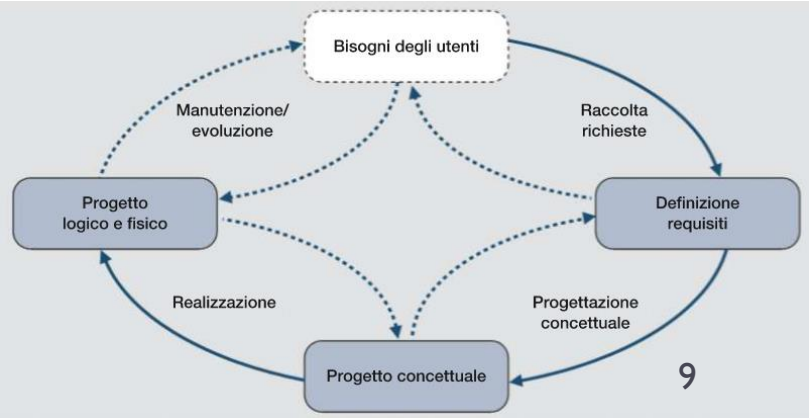
Sistema informatico aziendale

- Quella parte del sistema informativo in cui le informazioni sono raccolte, elaborate, archiviate, scambiate mediante l'uso dell'**ICT** (**I**nformation **&** **C**ommunication **T**echnology, sono le tecnologie della informazione e della comunicazione) costituisce il **sistema informatico,** anche chiamato **EDP (Electronic Data Processing).**

- È il sottoinsieme del sistema informativo formato da una componente:
  - **Software**: archivi e programmi di gestione (applicazioni)
  - **Hardware**: supporti fisici, computer, rete, infrastruttura,…

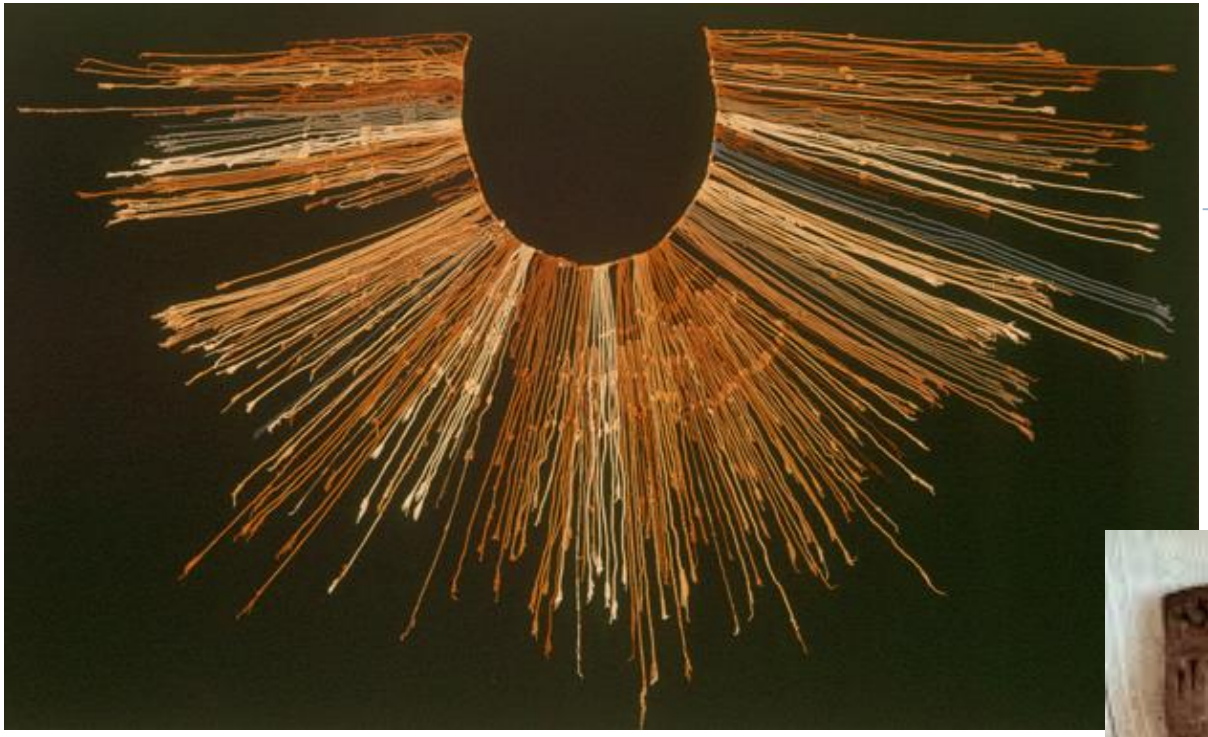
Ciclo di vita del sistema informatico

- L'informatizzazione di un SI deve essere occasione di razionalizzazione delle attività per renderle più efficaci e efficienti.
- Ci sono metodologie differenti, ma in generale è un processo ciclico e sono previste tre fasi
  - Raccolta delle richieste degli utenti
  - Progettazione concettuale
  - Realizzazione (progettazione logica e fisica)



Le basi di dati

di Roberta Molinari



I quipu inca

I **quipu** erano composti da una corda principale cui venivano legate altre secondarie e via così fino a costruzioni anche piuttosto complesse. Il differente colore delle corde aveva un preciso significato, così come la loro posizione e i nodi presenti su di esse.

Uso principale: conteggi per il censo, notazione delle tasse, conteggio degli articoli comprati o venduti e dati numerici di base. Gli amministratori Inca sembra fossero i principali utilizzatori dei quipo, usandoli per tenere traccia di risorse come bestiame e prodotti agricoli.

Mesopotamia

- in Mesopotamia, la "terra tra i due fiumi" Tigri e Eufrate: l'odierno Iraq, l'umanità vede la nascita della prima città (Uruk) e della scrittura nel 3200 a.C.
- I Sumeri scrissero su tavolette d'argilla, usando dapprima pittogrammi e poi caratteri cuneiformi, per oltre tremila anni.
- Utilizzo: documenti contabili o amministrativi, registrazioni di spedizioni e consegne di merci, vendita di terreni, schiavi e altre merci, affitti, salari, multe, prestiti, interessi dovuti, debiti e loro cancellazioni, sanzioni penali, matrimoni, divorzi, donazioni, adozioni, vertenze di ogni genere per i verdetti delle corti, eredità, e così via

Gli archivi

Gli **archivi** servono per conservare i dati in modo permanente, per poter essere reperiti e utilizzati in seguito.

Gli archivi sono un insieme di dati i cui elementi hanno le seguenti caratteristiche:

- sono legati tra loro da un *nesso logico* (si riferiscono ad uno stesso argomento Es. rubrica telefonica)
- sono rappresentati secondo un certo *formato* per permetterne l'interpretazione (Es. nominativo, indirizzo, tel)
- sono registrati in modo permanente su un certo *supporto* su cui è possibile leggere e scrivere (Es. la rubrica cartacea)
- sono *organizzati* per permetterne la consultazione (Es. ci sono le etichette delle lettere)

Se il supporto è di tipo informatico si parla di **Archivi elettronici o File di dati**

Gli archivi

Ogni dato numerico o alfabetico, viene memorizzato tramite una successione di byte

Esempio: stringa "AB"

ASCII("A") = 65 in binario 01000001

ASCII("B") = 66 in binario 01000010

In MM sarà memorizzata la successione di byte

01000001 01000010

Esempio: valore numerico 16706

16706 in binario 0100000101000010

In MM sarà memorizzata la successione di byte

01000001 01000010

Gli archivi
Record logici

I dati, in generale, sono raggruppati in unità logiche ognuna riferita ad un singolo soggetto della "realtà" memorizzata.

Ogni unità è un **record logico**, che a sua volta è suddiviso in **campi**, i cui valori caratterizzano il soggetto. L'elenco dei campi e il relativo tipo costituiscono il **tracciato record**.

Es. di tracciato record per la rubrica telefonica

Cognome-Nome	**Indirizzo**	**Numero Telefono**
Alfabetico	Alfanumerico	Caratteri Numerici

Es. di record

Rossi Mario	Via Roma 1	5556346

Gli archivi
Record logici

Campo: spazio riservato per memorizzare il dato relativo ad un'informazione di senso compiuto (es. indirizzo). Singolo dato

Record: insieme di campi correlati tra loro formanti un'unità informativa più ampia (es. persona).

Insieme di dati di uno stesso soggetto

Nome	Indirizzo	Telefono
Mario Rossi	Via Roma 1	0171-66666
Bianchi Luisa	P.zza Italia 14	011-999999
Verdi Ugo	Via Po 12	011-444444

Archivio: insieme di record omogenei (es. rubrica).

Insieme di dati di un archivio di più soggetti

Gli archivi
Operazioni

- **Creazione dell'archivio**: si definisce il supporto, il tracciato record, il nome e l'organizzazione
- **Manipolazione dei dati**:
  - *Inserimento*
  - *Modifica o aggiornamento*
  - *Cancellazione*
- **Interrogazione o consultazione dei dati**: reperimento di informazioni
- **Distruzione**

Operazioni **CRUD**:
1. ***Create - AGGIUNGERE***
2. ***Read - RECUPERARE***
3. ***Update - MODIFICARE***
4. ***Delete - CANCELLARE***

Archivi
Limiti archivi tradizionali file-based

- I dati sono legati al programma realizzato: modifiche della struttura record comportano modifiche ai programmi che la utilizzano
- L'accesso ai dati è determinato dal tipo di organizzazione degli archivi (sequenziale, diretto) e si devono comunque leggere record per record
- Alcuni dati si presentano più volte nello stesso file o in file differenti
- L'accesso ai dati avviene solo tramite l'applicazione. Nuove interrogazioni richiedono la modifica delle applicazioni

22

Archivi
Limiti archivi tradizionali file-based

- In una gestione tradizionale ogni applicazione opera sui suoi dati la cui struttura è definita all'interno del programma stesso. È complicata la condivisione di dati da parte di più applicazioni.
- Se i dati sono utilizzati da più applicazioni, spesso vengono duplicati, con spreco di memoria e rischio di inconsistenza dei dati

23

Archivi
Limiti archivi tradizionali file-based

Ordini							
CodOrd	**Codacc**	**Descr**	**Prezzo**	**Qt**	**Codcli**	**Nome**	**Indirizzo**
01	M03	Batteria	100,00	3	010	Rossi Mario	Via Roma 1
01	M12	Antenna	25,00	1	010	Rossi Mario	Via Roma 1
02	M03	Batteria	100,00	2	020	Verdi Luca	C.so Italia 10

24

Archivi
Limiti archivi tradizionali file-based

Ridondanza

- Gli stessi dati appaiono in più punti

Incongruenza

- Conseguenza della ridondanza in caso di modifiche parziali delle occorrenze dei dati ripetuti

Inconsistenza

- Conseguenza della incongruenza: i dati non sono più affidabili

25

Database
Definizione

I **database** nascono per superare i limiti degli archivi tradizionali.

Sono una raccolta di dati (archivi) organizzati per essere usati in modo efficiente da differenti applicazioni e da utenti diversi. (è sicuramente su supporto informatico)

Nei database è presente sia la definizione della struttura dei dati (tracciato record: numero nome e tipo) che i dati stessi. La struttura non fa più parte dell'applicazione, è ora indipendente da essa.

26

Database
Teoria delle basi di dati

- Nell'informatica la **teoria delle basi di dati** studia come organizzare al meglio grandi quantità di informazioni, per poterle gestire in modo:
  - **Semplice**: per utenti e applicazioni
  - **Efficiente**: in tempo e spazio
  - **Efficace**: rappresentano realmente la realtà che si vuole gestire
  - **Sicuro**: da utenti non autorizzati
  - **Solido**: resistente a guasti o errori accidentali degli operatori
  - **Condiviso**: permettere l'accesso simultaneo

27

Database
Caratteristiche

- **Eliminazione della ridondanza**: non si devono duplicare dati perché gli archivi sono integrati
- **Sicurezza e protezione dei dati**: protezione da accessi non autorizzati e guasti, si effettua tramite autenticazione, autorizzazione e controllo integrità
- **Integrità e recupero dei dati**: vi sono controlli per recuperare anomalie causate da programmi o utenti autorizzati
- **Garantita la consistenza dei dati**: contro il pericolo dovuto ad accessi concorrenti di lettura/scrittura
- **Facilità di accesso**: accesso semplice e veloce
- **Interrogazioni**: richieste di dati che verifichino un certo criterio di ricerca
- **Garantita l'integrità dei dati** a 3 livelli:
  - di campo (tipo e vincoli espliciti)
  - di tabella (integrità sull'entità: non duplicati e pk non nulla)
  - di associazione (integrità referenziale)

Garantire l'**integrità** significa garantire la consistenza, validità dei dati. Un **vincolo di integrità** è una proprietà che deve essere soddisfatta dalle istanze di un database

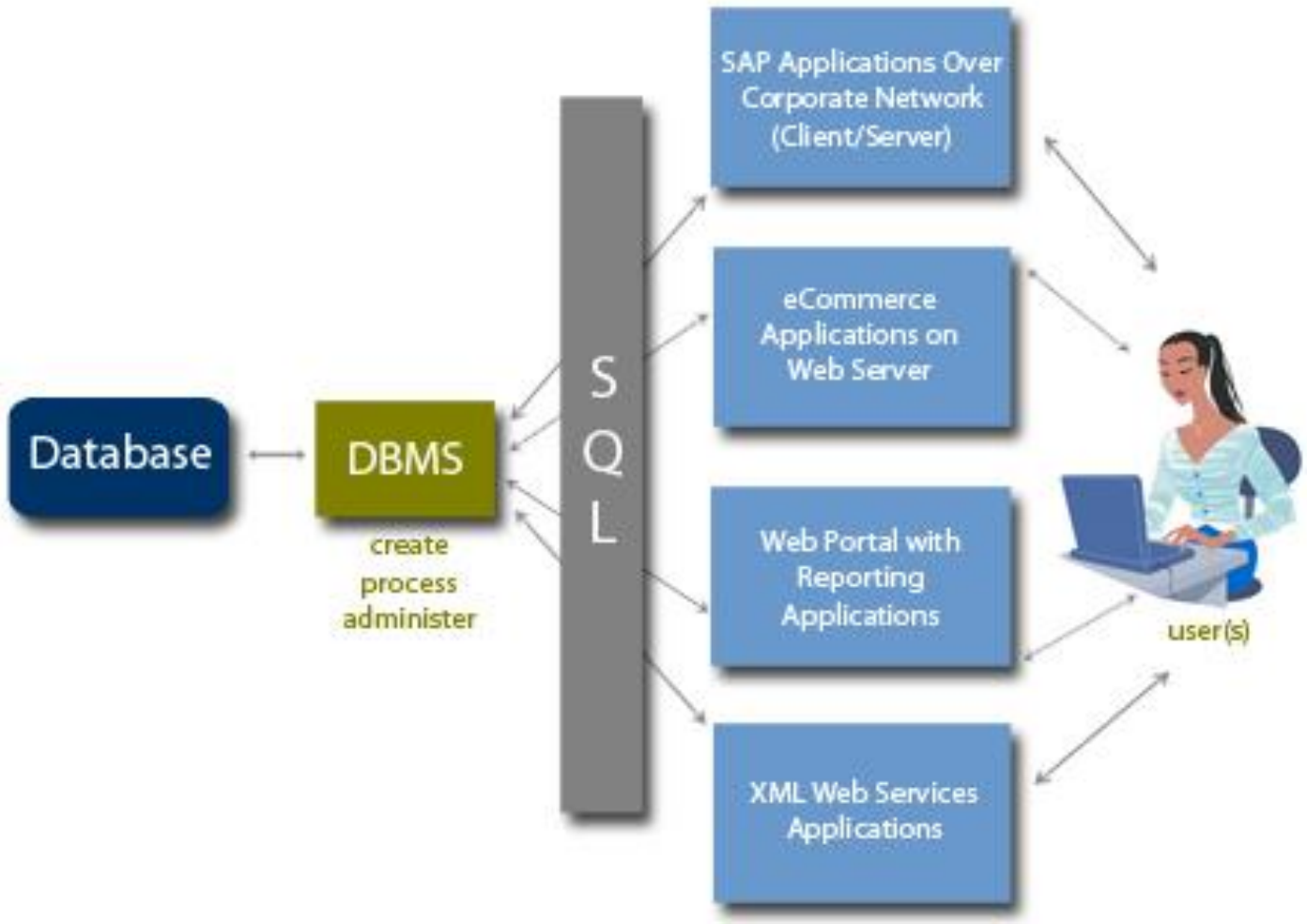
DataBase Management System DBMS

I prodotti software per la gestione dei DB si chiamano **DBMS** - ***DataBase Management System***. Costituiscono un interfaccia utente/applicazione-DB. Permettono di:

- Definire la struttura dei dati (modello logico)
- Manipolare ed interrogare il DB
- Controllare l'integrità dei dati
- Permettere la condivisione
- Garantire sicurezza e protezione e persistenza
- Gestire il modo in cui fisicamente sono archiviati i dati

30

DataBase Management System DBMS



DBMS: modifica e interrogazione

- I linguaggi di interrogazione del database mediante *query* (**interrogazioni**) e i generatori di *report* permettono agli utenti di interrogare in maniera interattiva il database e di analizzarne i dati.

- Il DBMS fornisce un modo per **aggiornare e immettere** nuovi dati nel database, oltre che per interrogarlo, questa capacità permette di gestire database personali.

DBMS: integrità

Il DBMS può mantenere **l'integrità del database**:

- non consentendo a più utenti di modificare lo stesso record contemporaneamente (blocco del record).
- Il database può impedire l'immissione di due record duplicati; per esempio può essere impedita l'immissione nel database di due clienti con lo stesso numero identificativo ("campi chiave").
- L'integrità è garantita dalla proprietà "ACID" delle transizioni.
- Implementano l'integrità dei dati ai tre livelli: campo, tabella, associazione.

DBMS: gestione autorizzazioni

Il sistema di sicurezza dei dati impedisce agli utenti non autorizzati di visualizzare o aggiornare il database. Mediante l'uso di *password* (parole d'ordine) agli utenti è permesso l'accesso all'intero database o a un suo sottoinsieme: in questo secondo caso si parla di ***subschema o vista***. Per esempio un database di impiegati può contenere tutti i dati riguardanti un singolo soggetto e un gruppo di utenti può essere autorizzato a vedere solamente i dati riguardanti lo stipendio, mentre altri utenti possono essere autorizzati a vedere solamente le informazioni che riguardano la sua storia lavorativa e la situazione sanitaria.

DataBase Management System
I linguaggi di un DBMS

I DBMS permettono tramite linguaggi (comandi) specifici di:

- Definire la struttura dati logica, definire le maschere video e i prospetti (Data Definition Language-**DDL**)
- Definire la struttura fisica relativamente ad una specifica MM (Data Media Control Language-**DMCL o** Storage Definition Language-**SDL**)
- Manipolazione dei dati (Data Manipulation Language-**DML**)
- Definire i vincoli di sicurezza, le autorizzazioni agli accessi e tipi di operazioni consentite agli utenti (Data Control Language-**DCL**)
- Interrogazione del DB (**Data Query Language –** DQL)
- TCL (**Transaction Control Language**): consente di avviare, concludere e gestire le transazioni.

DataBase Management System
I linguaggi di un DBMS in SQL

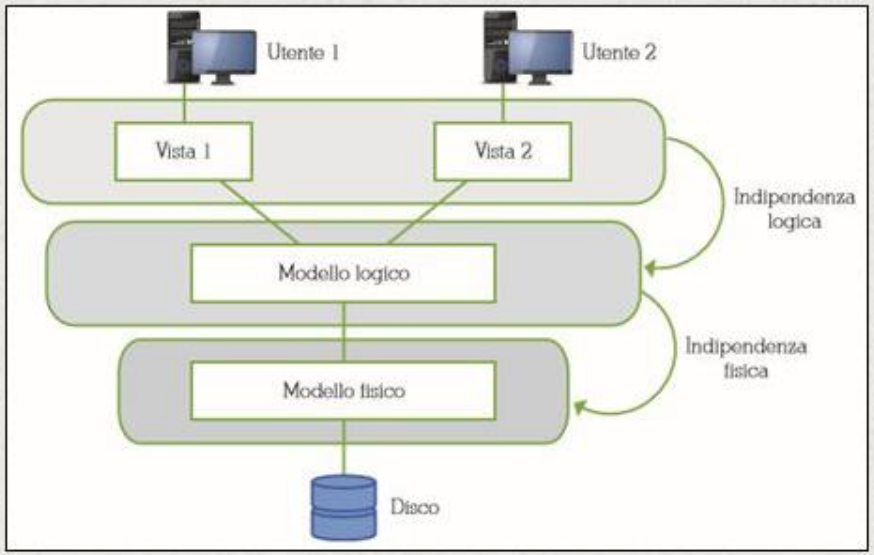
DataBase Management System Architettura a 3 livelli

I DBMS permettono di interagire con il DB su 3 livelli:

- **Livello esterno**: usato dagli utenti del DB. Il **DBA** (*DB Administrator*) ha realizzato per essi delle **viste** differenti in base a permessi definiti con il **DCL** e compiti degli utenti. A questo livello è possibile modificare i dati con il **DML** o fare interrogazioni con il **DQL.**
- **Livello concettuale o logico**: viene definito lo schema dei dati indipendentemente dalla implementazione fisica. Si usa il **DDL.**
- **Livello interno o fisico**: a questo livello il DBA decide i supporti di memorizzazione, l'organizzazione, i metodi di accesso per il DB,... Si usa il **DMCL.**

DataBase Management System
Architettura a 3 livelli

- il **livello fisico** gestisce i *file* che dovranno essere memorizzati sul disco,
- il **livello logico** gestisce le tabelle relazionali,
- il **livello di interfaccia verso l'esterno** si occupa di quali dati far vedere ("vista") agli utenti e in che modalità.



DataBase Management System Architettura a 3 livelli

L'architettura a tre livelli dei db, descrive i dati secondo 3 differenti livelli di astrazione, mediante opportuni schemi. Il DBMS realizza questi meccanismi di astrazione dei dati e assicura l'**indipendenza dei dati**: i livelli superiori non sono influenzati , entro certi limiti, dai cambiamenti che avvengono in quelli inferiori:

- **Indipendenza dalla struttura fisica dei dati**: si possono modificare i supporti, spostare tabelle, ecc. senza modificare il livello logico e quindi quello esterno realizzato dalle applicazioni
- **Indipendenza dalla struttura logica dei dati**: si possono modificare le definizioni delle strutture fatte a livello logico, senza modificare le viste esterne utilizzate dalle applicazioni

DataBase Management System
Tipi di architettura: stand alone

- Architettura in cui un singolo sistema o un'applicazione utilizza un database locale, spesso gestito da un sistema di gestione dei database (DBMS) come SQLite o Microsoft Access.
- Non è coinvolto un server centrale o una connessione di rete per accedere al database.
- Il database è ospitato localmente su una macchina singola e non dipende da altri sistemi per funzionare.
- Adatto per applicazioni su sistemi embedded o desktop o soluzioni standalone che non richiedono la condivisione dei dati tra più utenti o sistemi.

DataBase Management System
Tipi di architettura: Terminal Server

- Coinvolge un server centrale che offre servizi desktop o applicazioni virtuali ai client tramite una connessione di rete.
- I client sono dei terminali, dispositivi che non hanno capacità di elaborazione propria, ma si limitano a inviare le richieste e a ricevere le risposte dal server.
- Il server è un computer potente che esegue tutte le operazioni richieste dai client, compresa la gestione del database che si trova sul server.
- Il server centralizza il controllo e la sicurezza dei dati, ma deve anche sopportare un carico elevato di lavoro e di traffico di rete.

DataBase Management System
Tipi di architettura: Client-Server

- Architettura in cui i client sono dei computer o dei programmi che hanno una certa capacità di elaborazione propria, e che richiedono al server solo i servizi o le risorse di cui hanno bisogno.
- Il server, in questo caso, si occupa solo di fornire i servizi richiesti dai client, senza eseguire le operazioni al posto loro.
- Il database può essere gestito dal server o da un altro computer dedicato.
- Il server riduce il carico di lavoro e il traffico di rete, ma deve anche garantire la condivisione dei dati tra più utenti e sistemi.

DataBase Management System
Tipi di architettura: confronto

- Un'architettura stand alone per database offre vantaggi in termini di semplicità, autonomia e sicurezza, ma anche svantaggi in termini di scalabilità, condivisione e aggiornamento dei dati.
- Un'architettura terminal server è più centralizzata e semplice da gestire, ma anche più costosa e vulnerabile ai guasti.
- Un'architettura client-server tradizionale è più distribuita e flessibile, ma anche più complessa e disomogenea.

DataBase Management System
Transazioni

- **Transazione**: insieme di operazioni che devono essere eseguite in modo atomico, come un unico blocco: l'**atomicità** delle transazioni implica che o vengono eseguite completamente o non vengono eseguite. Se una transazione è annullata, per ripristinare la situazione iniziale si devono annullare tutte le operazioni eseguite fino a quel momento.
- Si classificano in
  - **Implicite**, create in automatico dal DBMS quando esegue operazioni di aggiornamento, inserimento e cancellazione
  - **Esplicite**, dichiarate dal programmatore

Es.

- Trasferimento di denaro tra conti correnti.
- Prenotazione di un posto in aereo

DataBase Management System
Transazioni – Esempio

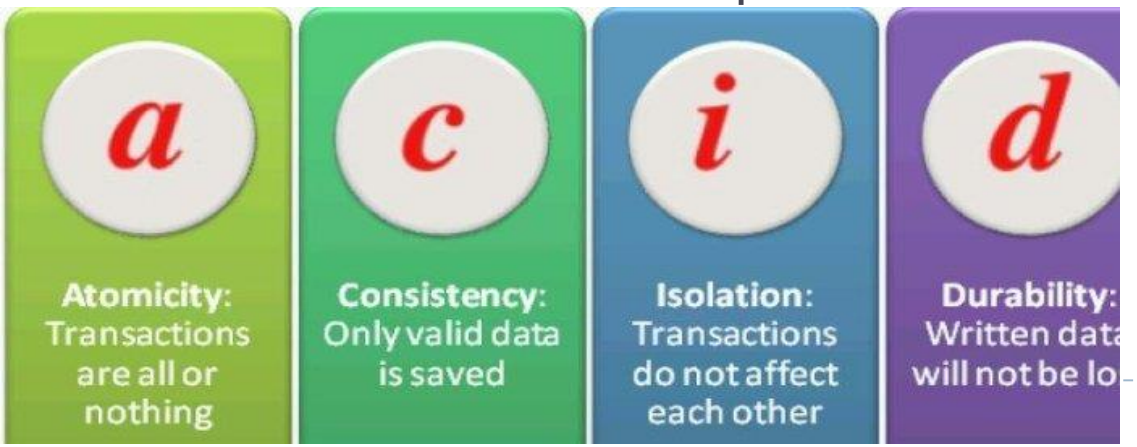
```
begin_transaction;
read (saldoX);
saldoX = saldoX – importo;
write (saldoX);
read (saldoy);
saldoy = saldoY + importo;
write (saldoY);
if(saldoX < fido) then
   rollback;
else
   commit;
end_if;
end_transaction;
```

Annulla tutte le modifiche dall'inizio della transazione

Conferma tutte le modifiche apportate nella transazione

DataBase Management System
Transazioni - Proprietà ACID

- Il DBMS deve garantire alle transazioni le seguenti proprietà
  - **Atomicity (atomicità)**: tutte le istruzioni sono eseguite come un'unica unità (o tutto o niente)
  - **Consistency (consistenza)**: il DB si deve trovare in uno stato consistente sia all'inizio che alla fine di una transazione, non deve violare eventuali vincoli di integrità
  - **Isolation (isolamento)**: in caso di transazioni concorrenti solo una può modificare i dati, se una fallisce le altre non devono fallire
  - **Durability (persistenza)**: una volta giunta a termine (commit), le modifiche effettuate dalla transazione sono permanenti nel DB



DataBase Management System
Transazioni – Transaction Log

- Per garantire l'atomicità il DBMS non esegue gli aggiornamenti direttamente sul DB, ma registra le operazioni da effettuare in un file di sistema chiamato **transaction log**. Ad ogni operazione sul DB si aggiunge un record nel file contenente:
  - Before image: dato prima della modifica
  - After image: dato dopo aggiornamento
  - User: chi ha richiesto la modifica
- Se la transazione è portata a termine (**commited**), allora viene effettivamente eseguita sul DB e si segna il punto in cui è arrivato nell'esecuzione delle operazione presenti nel log, con un check point
- Se la transazione è abortita (**aborted**), non si aggiornano i file e il record corrispondente nel file log viene cancellato

DataBase Management System
Transazioni concorrenti: anomalie

Le transazioni concorrenti possono portare a diverse anomalie:

- **perdita di aggiornamento (lost update)**
- **lettura sporca (dirty read)**
- **letture inconsistenti (unrepeatable read)**
- **inserimento fantasma (phantom insert o phantom read)**
- **aggiornamento fantasma (phantom update o inconsistent analysis)**

DataBase Management System
Anomalie: perdita di aggiornamento

Per esempio, il conto di Rossi, che ha un saldo di 3500 euro, viene aggiornato contemporaneamente da due differenti processi per effetto di due versamenti da 300 e 200 euro. L'esecuzione concorrente dei processi può portare, casualmente, alla seguente sequenza di operazioni:

- il primo processo accede al disco e legge il saldo, quindi ne aggiorna (in memoria centrale) il valore a 3800 euro.
- Nel frattempo anche il secondo processo accede al disco per conoscere il saldo, leggendo sempre il valore di 3500 euro.
- Il primo processo, subito dopo, aggiorna l'archivio scrivendo il valore 3800 su disco.
- A questo punto il secondo processo calcola il nuovo valore del saldo e lo scrive sul disco.

Al termine dei due aggiornamenti il saldo registrato in archivio è di 3700 euro anziché di 4000. Per effetto dell'esecuzione concorrente, uno dei due aggiornamenti non è stato registrato e si è avuta una **perdita di aggiornamento.**

DataBase Management System
Anomalie: perdita di aggiornamento

t1: r(x), x = x -1000, w(x)

t2: r(x), x = x -1000, w(x)

- Inizialmente x=4000; dopo un'esecuzione seriale x=2000
- Un'esecuzione concorrente:

t_1	t_2
$r_1(x)$	
$x = x - 1000$	
	$r_2(x)$
	$x = x - 1000$
$w_1(x)$	
commit	
	$w_2(x)$
	commit

- Quando due transazioni modificano un dato e un aggiornamento viene ignorato: alla fine x=3000

Esempio: incasso di assegni

DataBase Management System
Anomalie: lettura sporca

- Se x=100

t_1	t_2
$r_1(x)$	
$x = x + 1000000$	
$w_1(x)$	
	$r_2(x)$
abort	

- t2 ha letto uno stato intermedio ("sporco") e lo può comunicare all'esterno
- Accade quando si legge un dato modificato da un'altra transazione e quest'ultima fallisce.

Esempio: lettura del saldo su un aggiornamento sbagliato

DataBase Management System
Anomalie: letture inconsistenti

- Se x = 100

t_1	t_2
$r_1(x)$	
	$r_2(x)$
	$x = x + 1$
	$w_2(x)$
	commit
$r_1(x)$	

- t1 legge due volte lo stesso dato e ha due valori diversi.
- Quando una seconda transazione modifica i dati che la prima sta leggendo ripetutamente. Ad ogni transazione bisogna garantire di leggere lo stesso valore della stessa risorsa per tutta la sua durata.

Esempio: "quanti posti disponibili" ?

DataBase Management System
Anomalie: inserimento fantasma

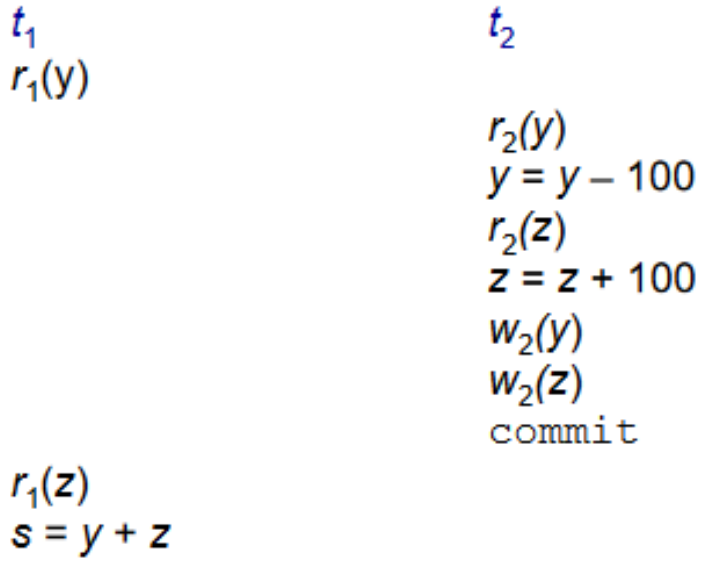
t_1	t_2
"conta i posti disponibili"	
	"aggiungi nuovi posti" (aereo più grande)
	`commit`
"conta i posti disponibili"	

- Questa anomalia si crea quando una transazione effettua delle operazioni con dati aggregati (come media, somma, ecc.) e durante la sua esecuzione un'altra transazione effettua una inserimento.

Esempio: sistema di prenotazioni

DataBase Management System
Anomalie: aggiornamento fantasma

- Supponiamo che ci sia il vincolo y + z = 1000;
  y = 400 e z = 600



- s= 1100: ma la somma y + z non è mai stata "davvero" 1100 e t1 vede uno stato non consistente che non esiste
- Questa anomalia porta il database (visto da una transazione) in uno stato inconsistente (che viola delle condizioni imposte) non reale.

Esempio: trasferimento da un conto ad un altro

DataBase Management System
Anomalie: analisi inconsistente

Transazione1

```
begin_transaction;
read (saldoX);
saldoX = saldoX – importo;
write (saldoX);
read (saldoy);
saldoY = saldoY + importo;
write (saldoY);
commit;
end_transaction;
```

Transazione2

```
begin_transaction
Totale=0
read(SaldoX)
Totale=Totale+SaldoX
read(SaldoY)
Totale=Totale+SaldoY
commit
end_transaction
```

La somma dei due saldi non deve differire nelle due transazioni

DataBase Management System
analisi inconsistente

Tempo	T1	T2	SaldoX	SaldoY	Totale
1	begin_transaction	begin_transaction	3500	5000	
2	read(SaldoX)	Totale=0	3500	5000	0
3	SaldoX=SaldoX-100	read(SaldoX)	3500	5000	0
4	write(SaldoX)	Totale=Totale+SaldoX	3400	5000	3500
5	read(SaldoY)		3400	5000	3500
6	SaldoY=SaldoY+100		3400	5000	3500
7	write(SaldoY)		3400	5100	3500
8	commit	read(SaldoY)	3400	5100	3500
9	end_transaction	Totale=Totale+SaldoY	3400	5100	8600
10		commit	3400	5100	8600
11		end_transaction	3400	5100	8600

DataBase Management System
Transazioni – Lock

- La tecnica usata per prevenire queste e le altre anomalie che si possono presentare per effetto dell'esecuzione concorrente delle transazioni consiste nell'uso dei **lock (blocchi)**: ogni transazione prima di accedere ai dati invia al controllore della concorrenza una richiesta di lock. Il controllore accetta o meno la richiesta in base ai lock precedentemente concessi su quel dato. Se la richiesta viene esaudita la transazione procede, diversamente si blocca in attesa che il lock le venga concesso.
- Questi blocchi possono essere impostati a livello di tabella, di riga e di campo.

DataBase Management System
Transazioni – Lock

Si distinguono

- *Blocchi ottimistici*: in cui le informazioni anche se bloccate da un'altra transazione, possono essere letti da altre transazioni
- *Blocchi pessimistici*: in cui le informazioni bloccate non possono neanche essere letti da altre transazioni

DataBase Management System
Transazioni – Lock

Perché le diverse anomalie non si presentino, è necessario che la politica di concessione avvenga con la tecnica del **locking a due fasi** (2 Phase Locking):

- le transazioni devono prima acquisire tutti i lock di cui hanno bisogno (*fase crescente*) per effettuare le diverse operazioni di lettura e/o di scrittura.
- Effettuate tali operazioni, le transazioni iniziano a rilasciare i lock (*fase calante*).

Una transazione non può acquisire uno o più lock, rilasciare uno dei lock acquisiti e poi prenderne altri.

DataBase Management System
Transazioni – deadlock

Un altro possibile problema è il **deadlock** (situazione di **stallo**): accade quando un processo è in attesa di una risorsa che è utilizzata da un altro processo, e questo richiede una risorsa utilizzata dal primo.

Lo stallo è difficile da identificare. Una tecnica molto diffusa nei database commerciali consiste nell'uso dei **timeout** nella concessione dei lock: una transazione può rimanere in attesa di un lock solo per un tempo predefinito. Il gestore della concorrenza, se non riesce a concedere il lock nel tempo prefissato, assume che la transazione sia in stallo, la interrompe, e la fa ripartire da capo.

DataBase Management System Backup

Uno dei compiti fondamentali di un DBMS è quello del salvataggio periodico dei dati (**backup**) e dell'eventuale loro ripristino (**restore**). Per poter ripristinare il DB, oltre ai file di backup, utilizza anche il **transaction log**.

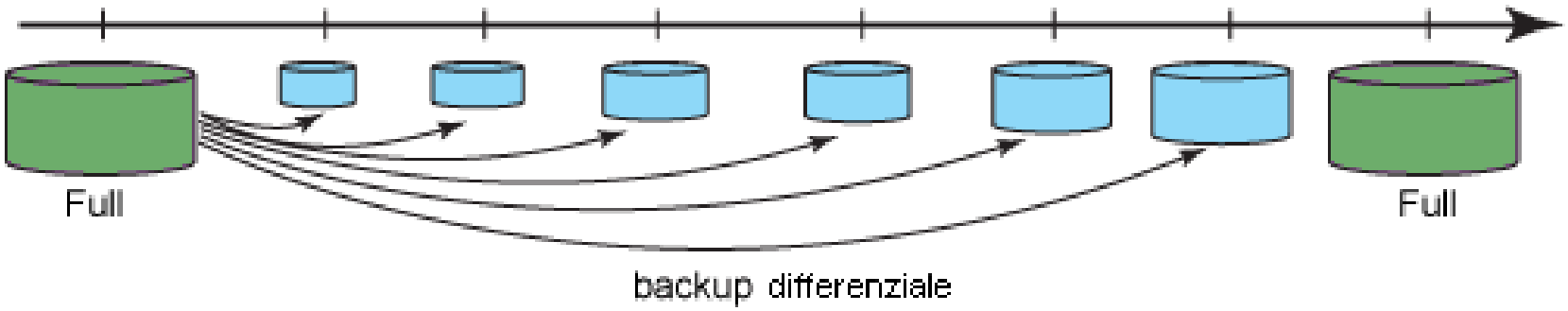
Si parla di backup a **caldo** (o Hot backup) quando è effettuato mentre il database è in linea. I dati possono quindi essere modificati mentre il backup è in corso.

I backup si classificano in:

1. **Completo** (o Full backup): backup di tutti i files sul sistema. A differenza della disk image, un full backup non include le tavole di allocazione, le partizioni ed i settori di boot.

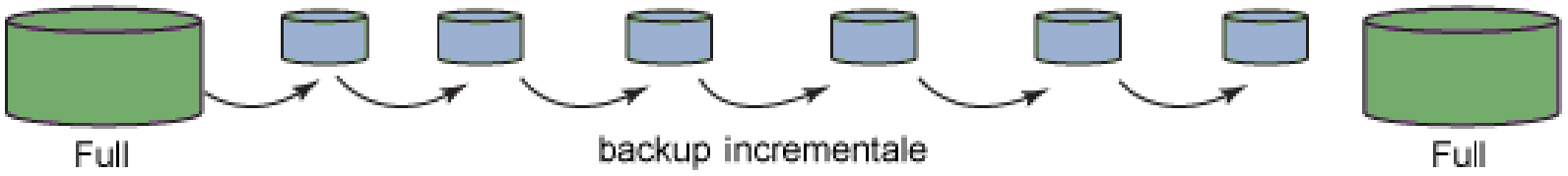
DataBase Management System
Backup

2. **Differenziale:** Backup cumulativo di tutti i cambiamenti effettuati a partire dall'ultimo backup completo effettuato. Il vantaggio è il minor tempo necessario rispetto ad un backup completo. Lo svantaggio è che i dati da salvare aumentano per ogni giorno trascorso dall'ultimo backup.



DataBase Management System
Backup

3. **Incrementale:** Backup che contiene tutti i files cambiati dal precedente backup (completo o incrementale). Il backup incrementale è più rapido di quello differenziale, ma richiede tempi di restore più lunghi poiché è necessario partire dall'ultimo backup completo e poi aggiungere in sequenza tutti i backup incrementali.



DataBase Management System
Backup: modalità

- Se un DB è di dimensioni limitate si farà
  - Un backup completo ogni sera
  - Ogni giorno si crea un nuovo transaction log
- Se un DB è di grandi dimensioni si farà
  - Un backup completo ogni fine settimana
  - Un backup incrementale ogni sera
  - Ogni giorno si crea un nuovo transaction log

In caso di malfunzionamento si ripristinerà il DB all'ultimo full backup, quindi all'eventuale backup differenziale della sera precedente e tramite le transaction log si ricostruiranno le operazioni eseguite nella giornata