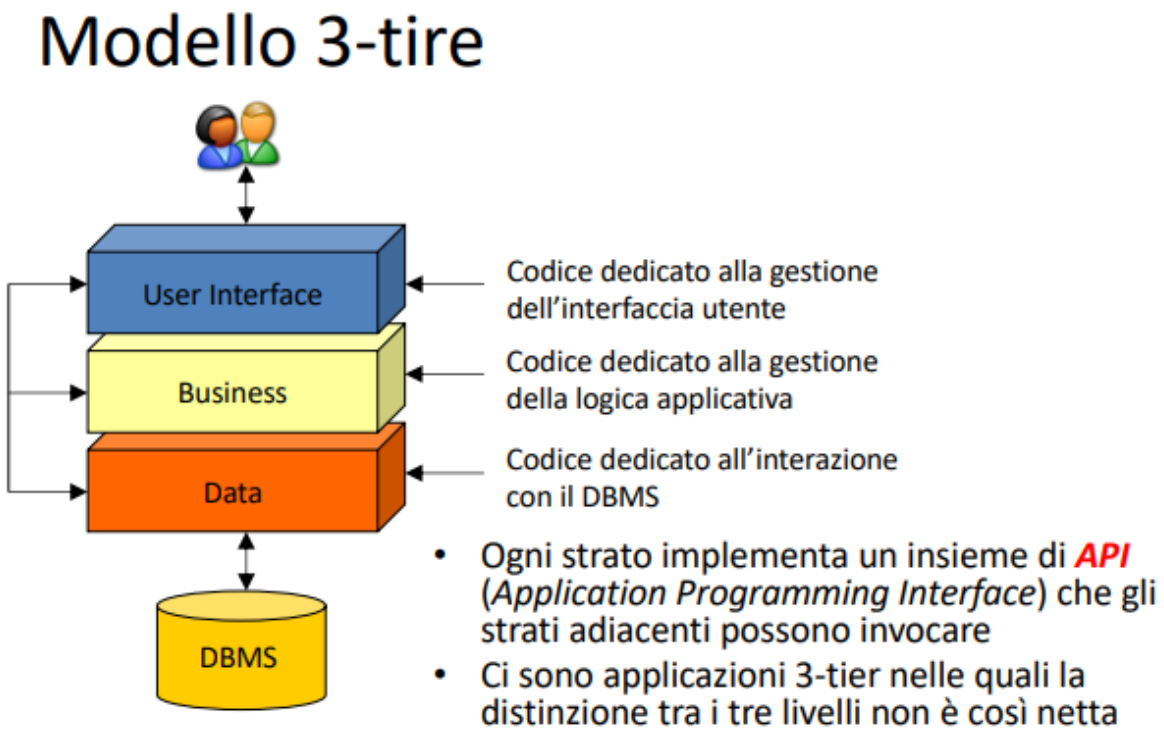
Base dati e sistemi informativi

La base dati sono usati nei modelli **3 tier,** ovvero: **interfaccia**, **logica** e **base dati.**

> La base dati è un insieme di **dati atomici**, **strutturati e permanenti** raggruppati in **insiemi omogenei** in **relazione** tra loro.



Usiamo un DBMS per gestire la base dati, quest'ultimo permette di:

- **Eseguire operazioni sulla base dati**: attraverso DDL e DML.
- **Tolleranza ai guasti:** eseguire backup e recovery dei dati.
- **Multi-utenza:** permette di gestire più utenti senza che si intreccino i dati-

- **Riservatezza dati:** accessi riservati tramite la DCL.

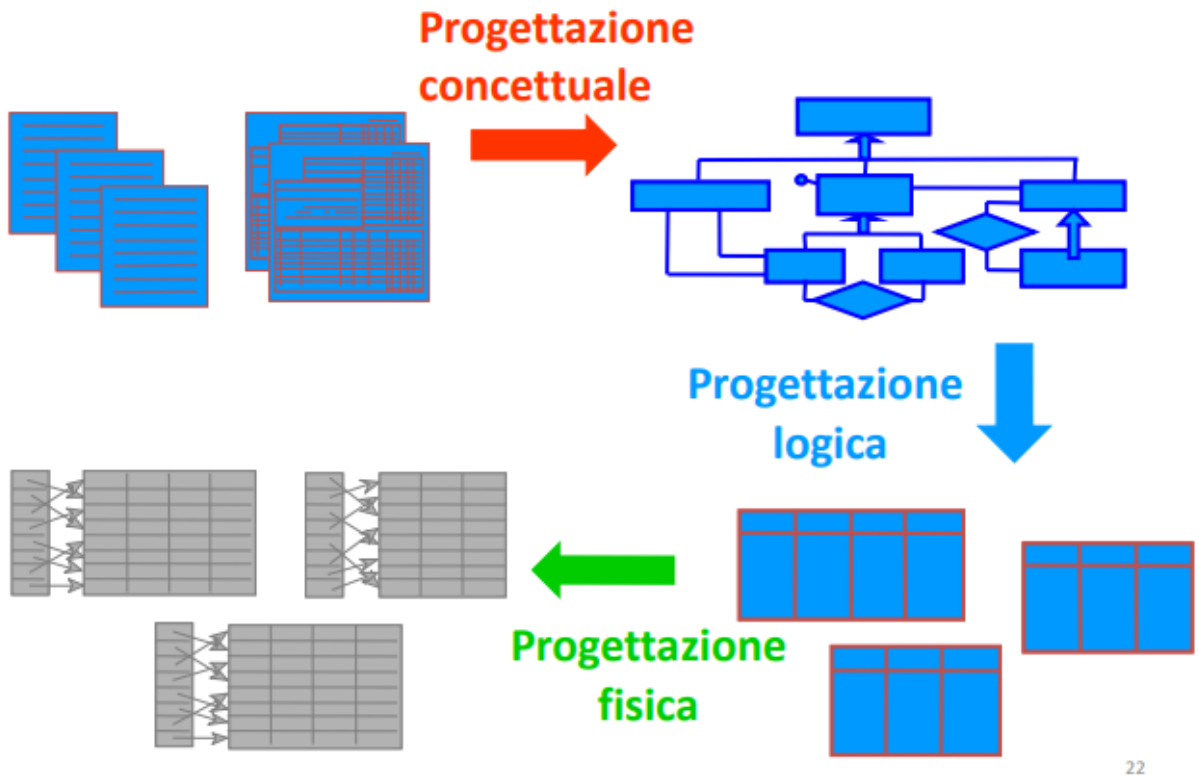
Un DBMS racchiude i tre linguaggi principali ovvero **DDL (Definizione)**, **DCL (Controllo)** e **DML (Manipolazione)** i quali svolgono operazioni sui dati.

Fasi di un sistema informativo

1. **Studio fattibilità**: si stabiliscono le possibili soluzioni con i relativi costi
2. **Raccolta requisiti**: si raccolgono i requisiti hardware e software.
3. **Progettazione**: si crea il modello del progetto.
4. **Implementazione**: si costruisce il progetto.
5. **Validazione e collaudo**: si verifica il funzionamento.
6. **Funzionamento**: si mette in produzione con manutenzioni e revisioni.

Le variabili su cui si gioca sono: tempo, costi e qualità.

Fasi di progettazione della base dati



Abbiamo tre fasi distinte:

1. **Progettazione concettuale**: ci fornisce lo schema concettuale (ER)
2. **Progettazione logica**: ci fornisce lo schema relazionale (ER ristrutturato + relazionale)
3. **Progettazione fisica**: ci fornisce lo schema fisico (SQL + dati)

Progettazione concettuale

Questa è la prima fase ovvero dove iniziamo a raccogliere le informazioni **uniformandole**, **rimuovendo le ambiguità** e costruendo un **glossario dei termini comuni** e individuando le **operazioni più comuni** per poterle successivamente disporre nel nostro progetto.

Costruzione dello schema concettuale

Per poter costruire lo schema di partenza facciamo affidamento a: entità, attributi, associazioni e generalizzazioni.

Ci sono costrutti che **non posso essere rappresentati** nello schema ER:

- **Glossario dei termini:** le parole più usate
- **Business rules:** le regole da seguire per determinati campi
    - **Vincoli di integrità**: regola che garantisce coerenza del dato
    - **Derivazioni**: come determinati dati debbano essere ricavati partendo da altri

Pattern di progettazione

Esistono diverse tipologie di progettazione, le più usate sono:

- **Instance-of:** è una relazione in cui una entità (debole) deriva da un'altra (forte).
- **Part-of:** una entità (debole) fa parte di un'altra (forte).
- **Storicizzazione di un'Entità**: quando vogliamo avere i cambiamenti, introduciamo la data.
- **Storicizzazione di un'Associazione**: quando vogliamo avere i cambiamenti, introduciamo la data.

Strategie di costruzione

Queste sono le strategie che esistono per creare un progetto:

- **Top-down**: partire dal problema generale e creare uno scheletro che si va a raffinare, questo approccio è quello più comodo ma raramente usato dato che bisogna avere un'idea generale di tutte le componenti e nei progetti di grandi dimensioni non è possibile averla.
- **Bottom-up**: creare le unità più elementari e unirle fino a creare lo schema finale, questa strategia è ottima per quanto riguarda la progettazione di gruppo ma è difficile integrare i diversi schemi che si creano.
- **Inside-out**: questa strategia è una variazione della bottom-up infatti individuiamo prima concetti importanti e poi li uniamo a "macchia d'olio" quindi ci espandiamo dal punto in cui abbiamo iniziato, il problema è che richiede continue revisione per mantenere una coerenza dello schema.

Progettazione logica

In questa seconda fase dobbiamo **trasformare** lo **schema concettuale** in uno **schema relazionale (detto anche logico)** ovvero uno schema che rappresenti i dati in maniera efficiente. Viene suddiviso in due fasi:

- **Ristrutturazione dello schema concettuale:** in questa fase andiamo ad evidenziare alcune inefficienze dello schema concettuale che sono presenti dato che la prima fase serve per costruire lo scheletro.
- **Traduzione verso lo schema relazionale e ottimizzazioni**: fase in cui lo schema logico viene tradotto in schema relazionale.

Analisi delle operazioni

Per poterle migliorare, dobbiamo prima analizzare le operazioni, in tempo e spazio, per questo costruiamo:

- **Tavola dei volumi**: contiene dati con stime di dimensioni.
- **Tavola delle operazioni**: contiene la frequenza con cui le operazioni sono svolte.

Nella **tavola dei volumi** abbiamo due tipi: **E** (entità), **R** (relazioni) mentre nella **tavola delle operazioni** abbiamo **B** (batch, ovvero non richiede parametri), **I** (interattiva, ovvero richiede parametri). Ogni operazione ha un costrutto ad esso legato:

- **Tavola degli accessi**: contiene tutte le entità e relazioni che si vanno ad interrogare per quella operazione. Questa **tabella serve nella rimozione delle ridondanze**.

Ristrutturazione dello schema concettuale

1. Rimozione delle ridondanze

Dobbiamo vedere se rimuovendo la ridondanza ovvero un attributo che possiamo ricavare anche in altri modi, conviene mantenerlo o meno. Ci basiamo quindi su due calcoli ovvero tempo e spazio.

2. Rimozione delle gerarchie

Non sono direttamente rappresentabili e quindi abbiamo diverse strade per rimuoverle:

- **Accorpamento dei figli nel genitore**
- **Accorpamento del genitore nei figli**
- **Sostituzione con delle associazioni**

3. Partizionamento/accorpamento di attributi

Gli accessi di riducono in due casi: **separando attributi di uno stesso concetto** ai quali si accede da **operazioni diverse**, **accorpando attributi di concetti diversi** ai quali si accede da **operazioni uguali**.

4. Scelta degli identificatori primari

La scelta di una chiave primaria è fondamentale per poter eseguire le operazioni, la chiave deve essere composta dal minor numero possibile di attributi.

Traduzione verso lo schema relazionale e ottimizzazioni

Per tradurre quello che abbiamo in uno schema logico dobbiamo impostare le cardinalità alle relazioni che richiede un'analisi, esistono diversi tipi di relazioni e sono:

- **Uno a uno**: la relazione diretta per cui a una tupla di un'entità corrisponde una sola tupla di un altra entità.
- **Uno a molti**: una relazione in cui una tupla può essere collegata a molteplici tuple
- **Molti a molti**: una relazione che richiede di creare una relazione a parte che collega le due entità.

Traduzione in schema relazionale

A questo punto possiamo impostare lo schema relazionale prendendo le entità e avremo tutti gli attributi con relative chiavi esterne, dalle quali partirà la freccia verso altre chiavi primarie. Le relazioni molti a molti saranno descritte mentre le altre no.

Data Definition Language (DDL)

Con la DDL possiamo creare tabelle, domini e vincoli. I domini già impostati in SQL sono “Timestamp” o “Date”, i domini sono un insieme di valori con un vincolo.

```
CREATE DOMAIN Nome AS Tipo CHECK(Clausola);
```

Un **dominio** serve per mantenere l’integrità della base di dati, così da evitare l’inserimento di dati non congruenti.

Un **vincolo** serve per imporre una condizione ad esempio un valore non nullo o unico all’interno della tabella. Le chiavi primarie sono un esempio di vincolo le quali sono uniche e non nulle.

Concetti evoluti

Le business rules sono regole da rispettare all’interno della base dati per garantire l’integrità, possiamo controllarle con l’uso di CHECK il quale permette di impostare un dominio.

Data Manipulation Language (DML)

Con la DML possiamo manipolare i dati delle tabelle. La manipolazione può essere fatta con funzioni matematiche oppure anche su stringhe con dei confronti totali o parziali (costrutto LIKE tra stringhe parziale con “%”).

Possiamo unire due tabelle con il costrutto “JOIN” il quale sulla base di un valore andrà a fare un’unione. Il risultato **senza clausole** sarà l’operazione di **prodotto cartesiano**!

Utilizzo di alias, possiamo rinominare la colonna di una tabella con l’uso di “AS” così facendo possiamo rinominare solo per quella query.

```
SELECT * FROM Tabella WHERE clausola;
```

Per eliminare i duplicati possiamo usare la parola chiave “DISTINCT” così facendo potremo unire su un valore i risultati.

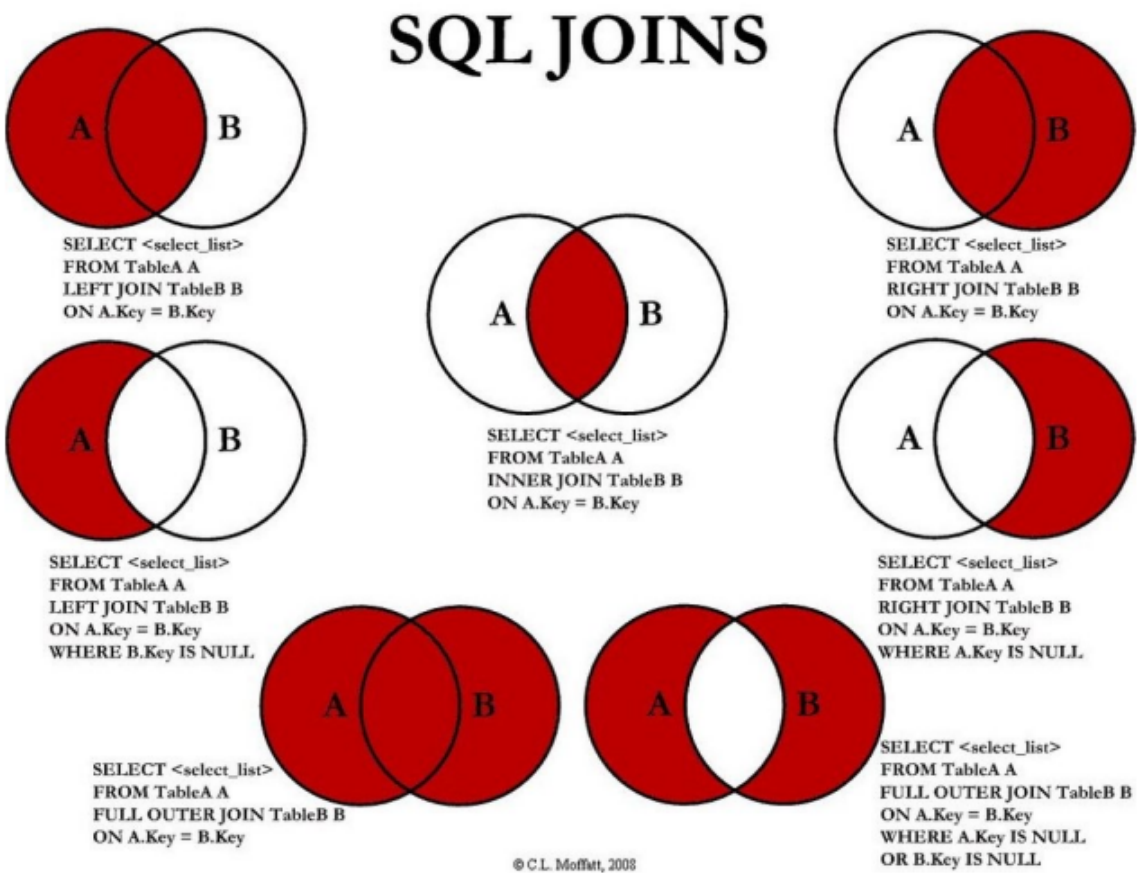
Possiamo ordinare i risultati con la parola chiave “ORDER BY” su un attributi in maniera crescente o decrescente.

Tipologie di JOIN

Ci sono tre tipologie di JOIN ovvero:

- **Inner** JOIN: quando abbiamo una corrispondenza in entrambe le tabelle
- **Natural** JOIN: si basa su colonne con lo stesso nome
- **Outer** JOIN: restituiscono tutte le righe di una tabella e le righe corrispondenti dell'altra tabella e ci sono tre tipi:
  - **Right outer join**: restituisce tutte le righe della tabella di destra
  - **Left outer join**: restituisce tutte le righe della tabella di sinistra
  - **Full outer join**: entrambe

```
SELECT * FROM Tabella1 JOIN Tabella2 ON clausolaJoin WHERE ...;
```



Operatori aggregati

Fino ad ora abbiamo valutato i risultati delle tabelle come singole tuple, ma possiamo unire i risultati secondo dei calcoli.

Gli operatori di aggregazione prendono in considerazione **più tuple** e **restituiscono** un **valore singolo, ad esempio COUNT, MAX, MIN, SUM**.

Per impostare una condizione sul costrutto "**HAVING**" dobbiamo usarne un altro chiamato "GROUP BY" infatti unendo le tuple su un valore è possibile poi lavorarci con delle condizioni. Non possiamo usare gli operatori di aggregazione nel WHERE ma dobbiamo usare HAVING.

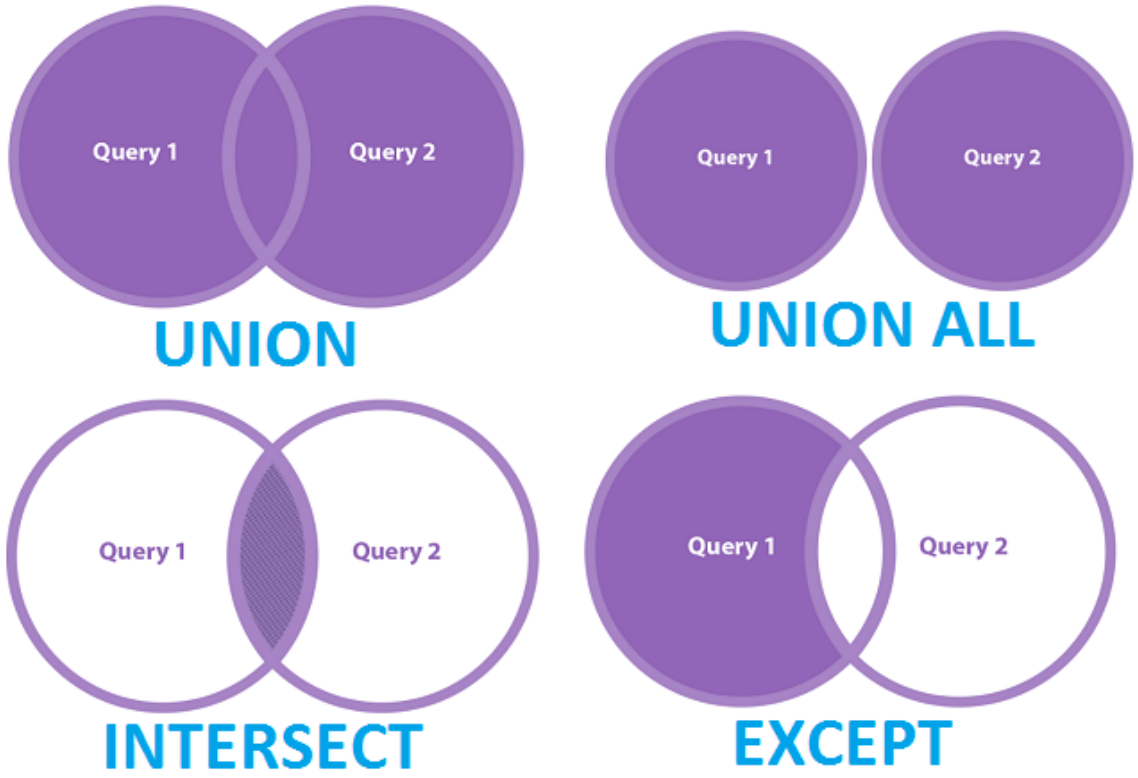
```
SELECT * FROM Tabella GROUP BY Colonna HAVING Count(Colonna) cla
```

Operatori insiemistici

Abbiamo diversi operatori insiemistici come:

- **Union**: serve per unire due risultati di query
- **Intersect**: serve per restituire le righe comuni
- **Except**: restituisce le righe della prima query che non sono presenti nella seconda

L'operatore union elimina i duplicati in automatico a meno che non venga richiesto di mantenerli con l'aggiunta di ALL.



Query nidificate

Possiamo creare delle sottoquery all'interno della nostra query principale esistono due tipi:

- **Semplici**: viene prima valutata quella interna
- **Correlate**: l'interrogazione interna fa fede a un risultato di una tabella esterna

Possiamo usarle nel WHERE e nel FROM. Per Valutare se un valore è in un insieme possiamo usare **IN** o **NOT IN**, gli insiemi sono solitamente le sottoquery per determinare su un valore è o meno nell'insieme.

```
WITH nome_subquery (SELECT ...) // Successivamente si imposta la
```

Viste

È possibile aggiungere allo schema del database rappresentazioni diverse dello stesso insieme di dati definendo tabelle derivate da tabelle di base. Ne esistono di due tipi: **virtuali** e **materializzate**.

```
CREATE VIEW nomeVista AS (sottoquery);
```

Se serve solo alcuni attributi possiamo selezionarli in questo modo: nomeVista(listaAttributi). Quando eseguiamo un cambio alla vista possiamo farlo e si ripercuote sulla tabella di partenza solo se andiamo a selezionare esattamente una riga.

Possiamo impostare un controllo sui check con "with check option" alla fine della creazione della vista.