Domande esame (possibili)

Si descriva brevemente cosa siano indipendenza logica e fisica dai dati e quale/i modello di basi di dati consente/ono di realizzarle

L'architettura a livelli di un DBMS garantisce l'indipendenza dei dati in due livelli:

- **Indipendenza fisica**: consente di interagire con il DBMS in modo indipendente dalla struttura fisica dei dati, senza influire sulle descrizioni e quindi sui programmi che usano i dati
- **Indipendenza logica**: consente di interagire con il livello esterno della base di dati in modo indipendente dal livello logico, per esempio come aggiungere un nuovo schema esterno senza modificare lo schema logico e perciò la sottostante organizzazione fisica dei dati

Il **modello relazionale** risponde perfettamente al requisito dell'indipendenza dei dati perché introduce una netta distinzione tra il livello fisico e quello logico. A differenza dei modelli precedenti (reticolare e gerarchico), che includevano riferimenti espliciti a puntatori fisici, il modello relazionale è basato su valori e non richiede la conoscenza delle strutture fisiche per accedere ai **dati**

"Descrivere brevemente cosa sono le ACID properties, a cosa servono e da dove deriva l'espressione 'ACID'" oppure "Illustrare brevemente cosa siano le proprietà ACID (spiegando anche l'acronimo) ed a quale entità/oggeto sono riferite"

Un DBMS prevede che le interazioni con la base di dati avvengano per mezzo di **transazioni**.
Una **transazione** è una sequenza di azioni di lettura e scrittura del DB e di elaborazioni di dati in memoria temporanea, che il DBMS esegue garantendo le seguenti proprietà (ACID properties):

- **Atomicity**: La transazione è eseguita nella sua interezza oppure non è eseguita affatto (le transazioni che terminano prematuramente sono abortite)
- **Consistency preservation**: una esecuzione corretta della transazione porta il DB da uno stato consistente all'altro (i vincoli di integrità devono essere rispettati).
- **Isolation**: L'esecuzione di una transazione deve essere indipendente da quella di altre transazioni concorrenti. Il risultato deve essere analogo a quello che si otterrebbe eseguendo le transazioni una alla volta (serialmente).
- **Durability** : le modifiche su DB di una transazione terminata normalmente sono permanenti, cioè non sono alterabili da malfunzionamenti successivi alla terminazione

Si definisca brevemente cosa siano i trigger, il paradigma su cui sono basati, vantaggi e svantaggi per essi

I trigger, detti anche regole attive, sono dei costrutti per rendere la base di dati in grado di reagire a eventi definiti dall'amministratore tramite l'esecuzione di opportune azioni.
Seguono il paradigma Evento-Condizione-Azione (ECA), ossia:

- **Evento**: Tipicamente una modifica dello stato del database consiste in una operazione di INSERT, DELETE, UPDATE. Se l'evento accade, il trigger si attiva.
- **Condizione**: Predicato booleano espresso in SQL che identifica se l'azione del trigger deve essere eseguita. Quando il trigger si attiva, viene valutata la condizione, quindi il trigger viene considerato.
- **Azione**: Consiste in una sequenza di update SQL o una procedura. Quando la condizione è verificata, allora l'azione viene eseguita, quindi il trigger viene eseguito.

I trigger sono uno strumento molto potente che permette di gestire vincoli di integrità, calcolare dati derivati, gestire eccezioni e codificare regole aziendali. Un ulteriore vantaggio derivante dall'utilizzo dei trigger consiste nel riuscire a codificare la logica del sistema in maniera centralizzata e condivisa da tutte le applicazioni, con conseguenti vantaggi in fase di lettura e manutenzione del codice, infatti in caso di modifiche al comportamento del sistema è sufficiente intervenire nell'ambito della definizione dei trigger e non in più parti del codice.
Lo svantaggio è che i trigger sono standardizzati solo per SQL-3, per cui potrebbero presentarsi casi (se pur sempre più rari) di non portabilità del codice.

Illustrare brevemente cosa sia il conflitto di impedenza e quali soluzioni esistano per gestirlo

Un importante problema che caratterizza l'integrazione tra SQL e i normali linguaggi di programmazione è il cosiddetto **conflitto di impedenza**. I linguaggi di programmazione accedono agli elementi di una tabella scandendone le righe una a una (tuple-oriented). Al contrario SQL è un linguaggio di tipo set-oriented, che opera su intere tabelle e restituisce come risultato di un'interrogazione un'intera tabella. Le soluzioni a questo problema si ottengono con l'utilizzo dei **cursori** e l'utilizzo di linguaggi con costruttori di tipo in grado di gestire una struttura del tipo "insieme di righe" (Call Level Interface).

Si definiscano brevemente le nozioni di sistema organizzativo, sistema informativo e sistema informatico e si delinei la differenza esistente tra di essi

Un **sistema organizzativo** è un insieme di risorse e regole che consentono il funzionamento di una qualunque struttura sociale per il raggiungimento dei suoi obbiettivi

Ogni sistema organizzativo è dotato di **sistema informativo**, ossia l'insieme delle risorse e delle procedure che un'organizzazione utilizza per gestire le informazioni necessarie al

perseguimento dei propri scopi.
Un sistema informativo è molte volte indipendente dalle automatizzazioni che conosciamo.

Un **sistema informatico** non è altro che la parte automatizzata di un sistema informativo, quindi un sistema software orientato alla gestione dei dati, dove l'aspetto prevalente è rappresentato dai dati stessi (memorizzati, ricercati, modificati) che costituiscono il patrimonio informativo di un'**organizzazione**

Si descriva brevemente i/il criteri/o di ottimizzazione delle query implementato dal gestore di ottimizzazione delle query di un DBMS

Il gestore delle interrogazioni è un modulo cruciale dell'architettura di un DBMS, in quanto responsabile dell'esecuzione efficiente di operazioni che sono specificate a livello molto alto. Esso riceve in ingresso un'interrogazione scritta in SQL, controlla che non vi siano errori lessicali, sintattici o semantici, una volta accettata, l'interrogazione viene tradotta in una forma interna di tipo algebrico. A questo punto, l'ottimizzazione vera e propria ha inizio, dividendosi in:

1. **Ottimizzazione algebrica**: effettua trasformazioni sulle operazioni (come l'anticipazione di selezioni e proiezioni verso le foglie dell'albero) che sono sempre convenienti indipendentemente dai costi fisici
2. **Ottimizzazione basata sul modello dei costi**: Sceglie la strategia di esecuzione (es. quale indice usare, quale algoritmo di join tra nested loop, merge scan o hash join) basandosi su un modello di costo che stima il numero di accessi in memoria secondaria e l'uso della CPU, sfruttando i profili statistici delle tabelle memorizzati nel catalogo (come spiegato nei profili delle relazioni);
3. Generazione del codice.

Elencare ed illustrare brevemente le strutture usate da un DBMS per organizzare i file (e le stutture primarie dei file) a livello fisico

La struttura primaria di un file stabilisce il criterio secondo il quale sono disposte le tuple nell'ambito del file. Le strutture possono essere divise in tre categorie principali:

- Sequenziali;
- Ad accesso calcolato (hash);
- Ad albero;

Nelle strutture sequenziali, un file è costituito da vari blocchi di memoria "logicamente" consecutivi, e le tuple vengono inserite nei blocchi rispettando una sequenza:

- **Seriale**: sequenza delle tuple indotta dall'ordine di immissione (organizzazione disordinata).
  Di solito viene chiamata anche **heap**, ossia mucchio

- **Array**: le tuple sono disposte come in un array, e la loro posizione dipende dal valore assunto in ciascuna tupla da un campo di indice.
  Possibile soltanto quando le tuple di una tabella sono di dimensione fissa.
- **Ordinata**: la sequenza delle tuple dipende dal valore assunto in ciascuna tupla da un campo (attributo) del file, ossia la chiave.

Una struttura con accesso ad **hash** garantisce un accesso associativo ai dati, ovvero un tipo di accesso in cui la locazione fisica dei dati dipende dal valore assunto da un campo chiave. Questo avviene tramite specifiche funzioni hash che consentono di trasformare un attributo chiave nell'indice di un array, e quindi associare ad ogni record una posizione specifica in una struttura sequenziale.

Le strutture ad albero, denominate anche **indici**, favoriscono l'accesso in base al valore di uno o più campi, consentendo sia accessi puntuali che corrispondenti a valori con complessità logaritmica (sulla base della profondità dell’albero).

Definire cosa siano RDF ed RDFS e le differenze tra di essi" oppure "Descrivere brevemente RDF e quale sia il suo linguaggio di interrogazione

RDF (Resource Description Framework): È un modello per rappresentare informazioni sul Web sotto forma di **triple** (soggetto, predicato, oggetto), che formano un grafo orientato. Permette di descrivere risorse in modo semplice ma non impone uno schema rigido

RDFS (RDF Schema): È un'estensione di RDF che permette di definire **metadati** (uno schema). Introduce concetti come **classi**, **proprietà** e relazioni di ereditarietà (*subClassOf*), consentendo di descrivere la struttura e derivare nuova conoscenza (inferenza). Il linguaggio di interrogazione standard per RDF è **SPARQL**

Illustrare brevemente cosa siano B-tree e B+-Tree e le differenze tra essi" oppure "Illustrare brevemente cosa sia un B+-Tree..." o "Determinare quali azioni possono essere necessarie su una struttura dati di tipo B-Tree.

Le strutture ad albero, denominate anche **indici**, favoriscono l'accesso in base al valore di uno o più campi, consentendo sia accessi puntuali che corrispondenti a valori con complessità logaritmica (sulla base della profondità dell’albero).
Queste strutture sono usate per l'indicizzazione. Garantiscono che le foglie siano tutte alla stessa distanza dalla radice, offrendo tempi di accesso logaritmici

Le differenze principali sono:

- **B-Tree:** I nodi intermedi possono contenere i dati veri e propri (o i puntatori ai record).
- **B+-Tree:** I dati (o i puntatori ai record) sono contenuti **solo nelle foglie**. I nodi interni servono solo da instradamento. Inoltre, le foglie sono collegate in una catena

sequenziale, rendendo questa struttura molto efficiente anche per interrogazioni su intervalli di valori

Descrivere brevemente cosa è la normalizzazione e quali problemi risolve

Esistono alcune proprietà, dette **forme normali**, che certificano la qualità dello schema di una base di dati relazionale tramite l'assenza di determinati difetti.
Quando una relazione non è normalizzata presenta ridondanze e si presta a comportamenti indesiderabili o anomali durante gli aggiornamenti.

Dunque, per **normalizzazione** si intende la procedura che permette di trasformare schemi non normalizzati in schemi che soddisfano una forma normale. È bene sottolineare, però, che la normalizzazione va utilizzata come tecnica di verifica dei risultati della progettazione di una base di dati, infatti una corretta applicazione di una metodologia di progettazione porta generalmente a schemi già normalizzati.

Si definisca cosa è una decomposizione senza perdita di informazione e quale condizione è possibile utilizzare per verificare tale proprietà

Data una relazione r su un insieme di attributi X, con X_1 e X_2 sottoinsiemi di X la cui unione sia pari a X stesso, si può decomporre senza perdita di dati sugli insiemi X_1 e X_2 se il join delle due proiezioni è uguale a r stessa (ossia non contiene **spurie**).
Quindi si decompone senza perdita su due sottoschemi se l'attributo comune ai due è chiave per almeno uno dei due

Si definisca cosa è un vincolo di integrità per una base di dati relazionale

Il **vincolo di integrità** è una proprietà che deve essere soddisfatta dalle istanze che rappresentano informazioni corrette per l'applicazione.
Ogni vincolo può essere visto come un predicato che assegna valori vero o falso se queste soddisfano le condizioni o no.

Si definisca cosa sia una superchiave ed una superchiave minimale (o chiave).

un insieme K di attributi è *superchiave* di una relazione r se r non contiene due tuple distinte t_1 e t_2 con $t_1[K] = t_2[K]$;
K è *chiave* di r se è una superchiave minimale di r (cioè non esiste un'altra superchiave K' di r che sia contenuta in K come sottoinsieme proprio).

Elencare le proprietà fondamentali di una transazione e descrivere brevemente ognuna di esse

Una **transazione** è una sequenza di azioni di lettura e scrittura del DB e di elaborazioni di dati in memoria temporanea, che il DBMS esegue garantendo le seguenti proprietà (ACID properties):

- **Atomicity**: La transazione è eseguita nella sua interezza oppure non è eseguita affatto (le transazioni che terminano prematuramente sono abortite)
- **Consistency preservation**: una esecuzione corretta della transazione porta il DB da uno stato consistente all'altro (i vincoli di integrità devono essere rispettati).
- **Isolation**: L'esecuzione di una transazione deve essere indipendente da quella di altre transazioni concorrenti. Il risultato deve essere analogo a quello che si otterrebbe eseguendo le transazioni una alla volta (serialmente).
- **Durability** : le modifiche su DB di una transazione terminata normalmente sono permanenti, cioè non sono alterabili da malfunzionamenti successivi alla terminazione

Si descrivano brevemente i diversi livelli di isolamento di una transazione ed il motivo per cui sono stati introdotti.

1. **Read uncommitted (degree of isolation 0):** consente transazioni che fanno solo operazioni di lettura (quelle di modifica sono proibite) che vengono eseguite dal sistema senza bloccare in lettura i dati. Si rende il sistema molto più veloce, ma può accadere che una transazione legga dati modificati da un’altra transazione non ancora terminata (dati sporchi) oppure abortita in seguito, motivo per cui questo livello di isolamento può applicarsi esclusivamente su porzioni di DB utilizzate sempre e solo in lettura.
2. **Read committed (degree of isolation 1):** a differenza del livello precedente in cui sui dati in lettura non vi era un bloccaggio, questo livello prevede che i dati in lettura siano bloccati esclusivamente per il tempo di lettura e subito rilasciati, mentre i dati in scrittura siano rilasciati alla terminazione della transazione. Questo comporta letture non ripetibili, ovvero letture successive sugli stessi dati possono dare risultati diversi perché i dati sono stati modificati da altre transazioni terminate nell’intervallo tra la prima e la seconda lettura.
3. **Repeatable read (degree of isolation 2)**: prevede che i blocchi in lettura e scrittura non sia applicati sull’intera tabella, ma siano assegnati solo su sottoinsiemi di tuple e vengano rilasciati alla terminazione della transazione. Questa soluzione evita il problema delle letture non ripetibili, ma non quello delle letture fantasma.
4. **Serializable (degree of isolation 4)**: le transazioni vengono serializzate in maniera sicura.

Illustrare brevemente quali sono i dati gestiti dal buffer manager e come avviene la sua gestione delle richieste

Il gestore del buffer gestisce, oltre al buffer appunto, un direttorio che per ogni pagina mantiene:

- File fisico e numero blocco corrispondete alla pagina

- Due variabili di stato: un contatore che indica quanti programmi utilizzano la pagina, un bit che indica se la pagina è “sporca”, cioè se è stata modificata

La conoscenza di questi dati è fondamentale nel momento in cui diventa necessario introdurre nuove pagine in un buffer saturo, per capire quali pagine andare a sostituire. Il buffer comunica con il sistema mediante delle operazioni primitive:

- fix: consiste nella richiesta di accesso ad una pagina, restituisce il riferimento alla pagina richiesta, richiede una lettura se la pagina non è nel buffer, incrementa il contatore per l’utilizzo della pagina
- $\mathrm{setDirty}$: comunica al buffer manager che la pagina è stata modificata
- unfix: indica che la transazione ha concluso l’utilizzo della pagina, quindi decrementa il contatore di utilizzo di pagina
- force: trasferisce in modo sincrono una pagina in memoria secondaria su richiesta del gestore dell’affidabilità.

E' possibile creare domini complessi in SQL? Quali sono i domini che SQL mette a disposizione?

Nella definizione delle tabelle si può fare riferimento ai domini predefiniti del linguaggio o a domini definiti dall'utente a partire da quelli predefiniti, infatti proprio da questi è possibile definirli in questa maniera:

$$\begin{aligned}&\text{CREATE DOMAIN } NomeDominio\ as\ TipoDiDato\\&[ValoreDiDefault]\\&[Vincolo]\end{aligned}$$

I domini elementari di SQL sono:

- Caratteri
- Tipi numerici esatti
- Tipi numerici approssimativi
- Istanti temporali
- Intervalli temporali

Descrivere brevemente in cosa consiste SQL Embedded e per quale motivo viene introdotto

SQL Embedded prevede di introdurre direttamente nel programma sorgente scritto nel linguaggio di alto livello le istruzioni SQL, distinguendole dalle normali istruzioni tramite un opportuno separatore

Si descriva brevemente cosa sia un cursore e per risolvere quale problema viene introdotto

Un cursore è una variabile speciale che permette ad un programma di accedere alle righe di una tabella una alla volta e permette di risolvere il cosiddetto **conflitto di impedenza**, infatti linguaggi di programmazione accedono agli elementi di una tabella scandendone le righe una a una (tuple-oriented), al contrario SQL è un linguaggio di tipo set-oriented, che opera su intere tabelle e restituisce come risultato di un'interrogazione un'intera tabella.

Descrivere le caratteristiche principali di una procedura definita in SQL-2 standard e le eventuali differenze rispetto a SQL-3

Lo standard SQL-2 prevede la definizione di **Procedure**, ovvero dei brevi sottoprogrammi memorizzati nel database come parte dello schema (motivo per cui vengono dette anche **stored procedures**). Esse permettono di assegnare un nome a un'istruzione SQL ed eventuali parametri. Una volta definita, la procedura è utilizzabile come un qualunque comando SQL.

È bene sapere che lo standard SQL-2 non tratta la scrittura di procedure complesse, ma solo quelle composte da un singolo comando SQL. Questo è invece permesso in SQL-3, dove viene fornita una ricca sintassi per la definizione di procedure, integrando anche le strutture di controllo

Illustrare quando è possibile utilizzare SQL statico e quando invece è necessario utilizzare SQL dinamico.

Nel caso di SQL statico, si usa quando sono noti a tempo di compilazione e vengono gestiti dal preprocessore, venendo ottimizzati solo una volta, e non ogni volta che il comando deve essere eseguito. Questo comporta grossi vantaggi in termini di prestazioni. L'SQL dinamico non può avvalersi della fase di preprocessamento, non essendo noti a priori i comandi da ottimizzare, quindi la costruisce a tempo di esecuzione, quindi viene utilizzato in questi casi necessari.

Illustrare brevemente cosa siano: algebra relazionale, calcolo relazionale ed SQL, le loro peculiarità e la relazione che intercorre tra di essi

L'algebra relazionale si configura come un linguaggio di tipo procedurale che si basa su una collezione di operatori definiti su relazioni, i quali producono a loro volta nuove relazioni come risultato. La sua peculiarità fondamentale risiede nella necessità di specificare esplicitamente il procedimento da seguire, ovvero la sequenza di operazioni quali selezione σ, proiezione π, ridenominazione ρ e diverse forme di join $\bowtie$, per giungere alla costruzione del risultato desiderato.

Il calcolo relazionale costituisce una famiglia di linguaggi di interrogazione di natura dichiarativa, basati sui principi del calcolo dei predicati del primo ordine. La caratteristica distintiva del calcolo è la sua capacità di descrivere esclusivamente le proprietà del risultato cercato attraverso formule logiche, senza indicare minimamente la procedura necessaria per ottenerlo .

Il linguaggio SQL, acronimo di Structured Query Language, rappresenta lo standard universale per l'interazione con le basi di dati relazionali, integrando in un'unica soluzione le funzionalità di definizione dei dati attraverso il Data Definition Language e di manipolazione e interrogazione tramite il Data Manipulation Language. La peculiarità dell'SQL è la sua impostazione dichiarativa che solleva l'utente dal compito di definire il piano di accesso fisico, delegando al sistema la ricerca della strategia ottimale

La relazione che intercorre tra questi linguaggi è di natura formale e tecnologica, in quanto l'algebra relazionale e il sottoinsieme del calcolo relazionale indipendente dal dominio sono considerati linguaggi equivalenti, poiché per ogni espressione formulata nell'uno è possibile trovarne una corrispondente nell'altro che produca il medesimo insieme di dati.
Nella pratica dei sistemi di gestione di basi di dati, l'SQL funge da interfaccia di alto livello basata sul calcolo su tuple, ma il gestore delle interrogazioni del DBMS traduce internamente ogni istruzione SQL in una rappresentazione algebrica equivalente

Descrivere brevemente i difetti del calcolo relazionale su domini per i quali è stato introdotto il calcolo relazionale su tuple

Il calcolo relazionale su domini presenta alcuni limiti strutturali e teorici piuttosto significativi, tra i quali il principale è rappresentato dalla dipendenza dal dominio, una proprietà per cui il risultato di un'interrogazione può variare in base all'universo dei valori considerato per gli attributi. Questo difetto implica che alcune espressioni possano produrre risultati privi di senso pratico o addirittura infiniti qualora il dominio di riferimento sia illimitato, come accade ad esempio in una negazione che includa tutti i valori non presenti in una determinata relazione

Per correggere queste problematiche è stato introdotto il calcolo relazionale su tuple con dichiarazioni di range, il quale apporta un cambiamento fondamentale definendo variabili che denotano intere tuple invece di singoli valori atomici

Si definisca brevemente cosa sia un DBMS e le sue principali caratteristiche

Il DBMS (sistemi di gestioni di basi di dati) è un sistema software in grado di gestire collezioni di dati che siano grandi, condivise e persistenti, garantendo affidabilità, privatezza, efficienza ed efficacia.

Un DBMS deve garantire:

- **Affidabilità**, quindi una resistenza a malfunzionamenti HW e SW in modo da mantenere intatto il contenuto o permetterne la ricostruzione (backup e recovery).
- **Privatezza dei dati**, un sistema deve poter definire dei meccanismi di autorizzazione per utente (opportunamente riconosciuto)

Data una base di dati si dica quale sia la sua parte invariante e la sua parte variabile e perché

Nella base di dati esiste una parte invariata nel tempo, detta **schema della base di dati**, costituita dalle caratteristiche dei dati, e una parte variabile, chiamata **istanza** della base di dati, costituita dai valori effettivi.

Descrivere brevemente cosa sia un modello dei dati ed elencare i modelli dei dati conosciuti

Un **modello di dati** è un insieme di concetti (o costrutti) per organizzare i dati di interesse e descriverne la struttura in modo comprensibile ad un elaboratore.
Ogni modello di dati fornisce meccanismi di astrazione per definire nuovi tipi sulla base di tipi (elementari) predefiniti e costruttori di tipo.

Il modello relazione dei dati (più diffuso tra tutti) permette di definire tipi per mezzo del costrutto della **relazione**, che consente di organizzare i dati in insiemi di record a struttura fissa. Oltre quello esistono diversi

Modello	Struttura usata	Anni	DBMS
Modello gerarchico	basato sull'uso di strutture ad albero	'60	IMS System 2000
Modello reticolare	basato sull'uso di strutture a grafo	Inizio '70	IDMS, IDS II, DM IV
Modello relazionale	basato sull'uso di relazioni: insiemi di record a struttura fissa con campi di tipo primitivo	'80	System R, Ingres, Oracle, DB2
Modello ad oggetti	basato sull'uso di classi di oggetti e istanze	Fine '80 - '90	O2 ObjectStore
Modello XML	rivisitazione del modello gerarchico: dati presentati insieme alla loro descrizione e non devono sottostare rigidamente ad un'unica struttura logica	'90	BaseX
Modelli semi-strutturati e flessibili	non hanno rigidità nell'organizzazione dei dati ed hanno alte prestazioni	'00	Sistemi NoSQL

1. Progettazione Concettuale: La finalità di questa fase è rappresentare le specifiche informali della realtà di interesse in una **descrizione formale e completa**, ma totalmente **indipendente** dai criteri di rappresentazione del DBMS che verrà utilizzato.
    - **Peculiarità:** Si caratterizza per un **alto livello di astrazione**. Il progettista si concentra esclusivamente sul contenuto informativo della base di dati, senza

preoccuparsi delle modalità di codifica in un sistema reale né dell'efficienza dei programmi. In questa fase si utilizzano principalmente le specifiche sui dati per definire gli elementi dello schema.
    - **Prodotto finale:** Il risultato è lo **schema concettuale**, tipicamente realizzato attraverso il **modello Entità-Relazione (E-R)**, che fornisce una visione unificata dei dati utile anche a scopo documentativo.
2. Progettazione Logica: La finalità della progettazione logica è la **traduzione dello schema concettuale** nel modello di rappresentazione dei dati adottato dal DBMS a disposizione (nel caso specifico, il modello relazionale).
    - **Peculiarità:** Sebbene sia ancora indipendente dai dettagli fisici, la rappresentazione è **concreta** poiché deve rispettare i vincoli del modello logico scelto. Non si tratta di una semplice traduzione meccanica: lo schema deve essere **ristrutturato** per semplificare la traduzione stessa (ad esempio eliminando le generalizzazioni) e per **ottimizzare le prestazioni** sulla base del carico applicativo previsto (volumi di dati e frequenza delle operazioni). In questa fase si utilizzano tecniche formali come la **normalizzazione** per verificare la qualità del risultato
    - **Prodotto finale:** Il risultato è lo **schema logico** (una collezione di tabelle nel modello relazionale) e la relativa documentazione dei vincoli di integrità.
3. Progettazione Fisica: La finalità della progettazione fisica è il completamento dello schema logico con la specifica dei **parametri fisici di memorizzazione** per massimizzare l'efficienza del sistema.
    - **Peculiarità:** Questa fase è **fortemente dipendente dal DBMS specifico** scelto. Il progettista deve conoscere le caratteristiche tecnologiche del sistema per definire l'organizzazione dei file e la creazione di strutture ausiliarie per l'accesso ai dati. Qui si prendono decisioni su quali indici creare per ottimizzare i tempi di risposta delle query e sulla contiguità di allocazione dei dati.
    - **Prodotto finale:** Il risultato è lo **schema fisico**, costituito dalle definizioni effettive delle relazioni e delle strutture fisiche utilizzate con i relativi parametri di implementazione.

Si descriva brevemente quali e quante sono le forme di ridondanza individuabili all'interno di uno modello E-R

All'interno di uno schema Entità-Relazione, la ridondanza è definita come la presenza di informazioni che possono essere derivate, ovvero ottenute attraverso una serie di operazioni, da altri dati già esistenti nel medesimo schema. Si possono individuare sostanzialmente tre forme principali di ridondanza:

1. La prima categoria riguarda gli attributi derivabili, occorrenza per occorrenza, da altri attributi appartenenti alla stessa entità o alla stessa associazione.
2. La seconda forma di ridondanza è costituita dagli attributi derivabili da attributi di altre entità o associazioni, operazione che avviene solitamente attraverso l'impiego di funzioni aggregate;

3. La terza forma riguarda le associazioni derivabili dalla composizione di altre associazioni in presenza di cicli nello schema.

Illustrare brevemente i motivi per cui è necessario effettuare l'analisi della ridondanza

La presenza di una ridondanza ha effetti positivi, semplificare le interrogazioni, ed effetti negativi, appesantisce gli aggiornamenti e comporta l'occupazione di più memoria. Per questo motivo, la decisione di mantenere o eliminare una ridondanza va presa in seguito ad una **analisi quantitativa** che confronti il costo di esecuzione delle operazioni che coinvolgono il dato ridondante e l’occupazione di memoria, sia in presenza e che in assenza di ridondanza.

Si descriva brevemente quando un join naturale si dice completo

Si parla di **join completo** se ogni tupla di ciascun operando contribuisce ad almeno una tupla del risultato.

Descrivere brevemente perché il modello relazionale è anche detto modello 'basato su valori'

Il modello relazionale viene definito basato su valori poiché, a differenza dei modelli logici precedenti come quello gerarchico o reticolare, le corrispondenze e i riferimenti tra i dati contenuti in relazioni diverse vengono rappresentati esclusivamente attraverso valori comuni che compaiono nelle tuple, senza l'impiego di puntatori espliciti o riferimenti a indirizzi fisici .