@STUDIOMADO 05/11/2019

OLTRE IL MODELLO RELAZIONALE

IL MODELLO RELAZIONALE

IL MODELLO RELAZIONALE

- Definito da Edgar Codd nel 1970
- Definisce come un insieme di dati deve essere presentato all'utente e non come deve essere rappresentato in memoria
- Permette diversi livelli di adesione attraverso le forme normali
- Non definisce il modo in cui debbano essere gestite richieste concorrenti

ELEMENTI DELLA TEORIA RELAZIONALE

- **Tuple:** insieme disordinato di attributi (righe e colonne)
- **Relazioni:** collezione di tuple distinte (tabelle)
- **Vincoli:**
  - Necessari per mantenere la consistenza del database
  - Utilizzati per identificare le tuple e le loro relazioni
- **Operazioni:** vengono effettuate sulle relazioni e restituiscono sempre una relazione (union, proiezioni, join, etc.)

IL MODELLO TRANSAZIONALE

- Definito da Jim Gray nel 1970
- in particolare definisce che

Una transazione é una trasformazione di stato, la quale deve essere atomica, permanente e consistente

IL MODELLO TRANSAZIONALE (ACID)

- **Atomica:** una transazione é indivisibile; tutte le istruzioni devono essere applicate con successo altrimenti deve essere ripristinato lo stato iniziale
- **Consistente:** il database deve rimanere in uno stato consistente prima e dopo la transazione
- **Isolata:** se due transazioni vengono eseguite contemporaneamente, l'una non deve vedere gli effetti dell'altra
- **Permanente:** gli effetti di una transazione avvenuta con successo devono persistere in memoria

DATABASE WARS

ORACLE®

SYBASE®

Microsoft® SQL Server®

MySQL®

IBM DB2

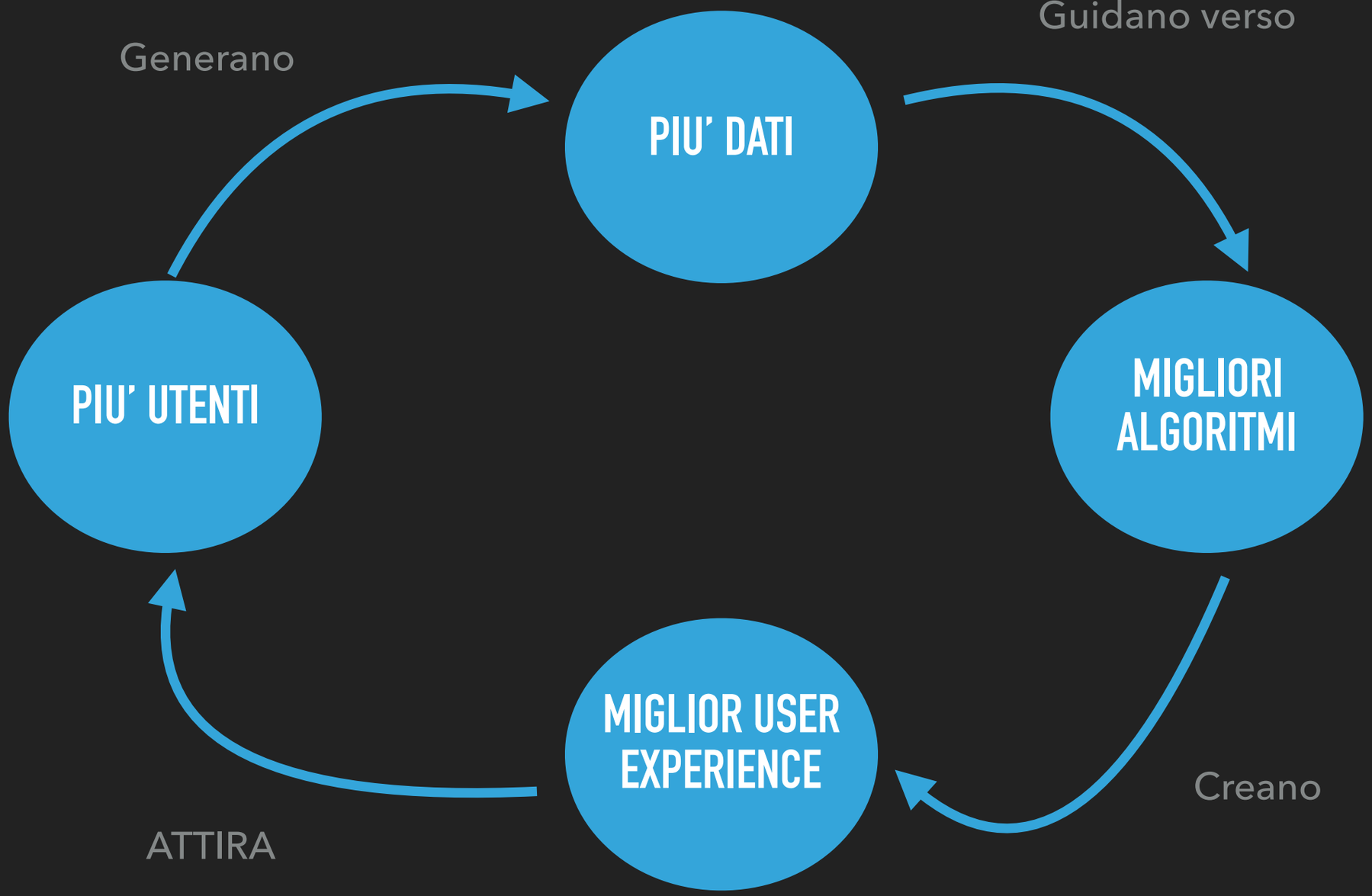
Informix®

GOOGLE, BIG DATA E HADOOP

GOOGLE, BIG DATA E HADOOP

- Crescita esponenziale dei dati a disposizione (multimedia, social network, dati transazionali, etc.)
- Capacitá di generare ulteriore valore aggiunto dai dati a disposizione grazie ai passi in avanti fatti in Machine Learning, analisi predittiva, data mining

CICLO VIRTUOSO DEI BIG DATA



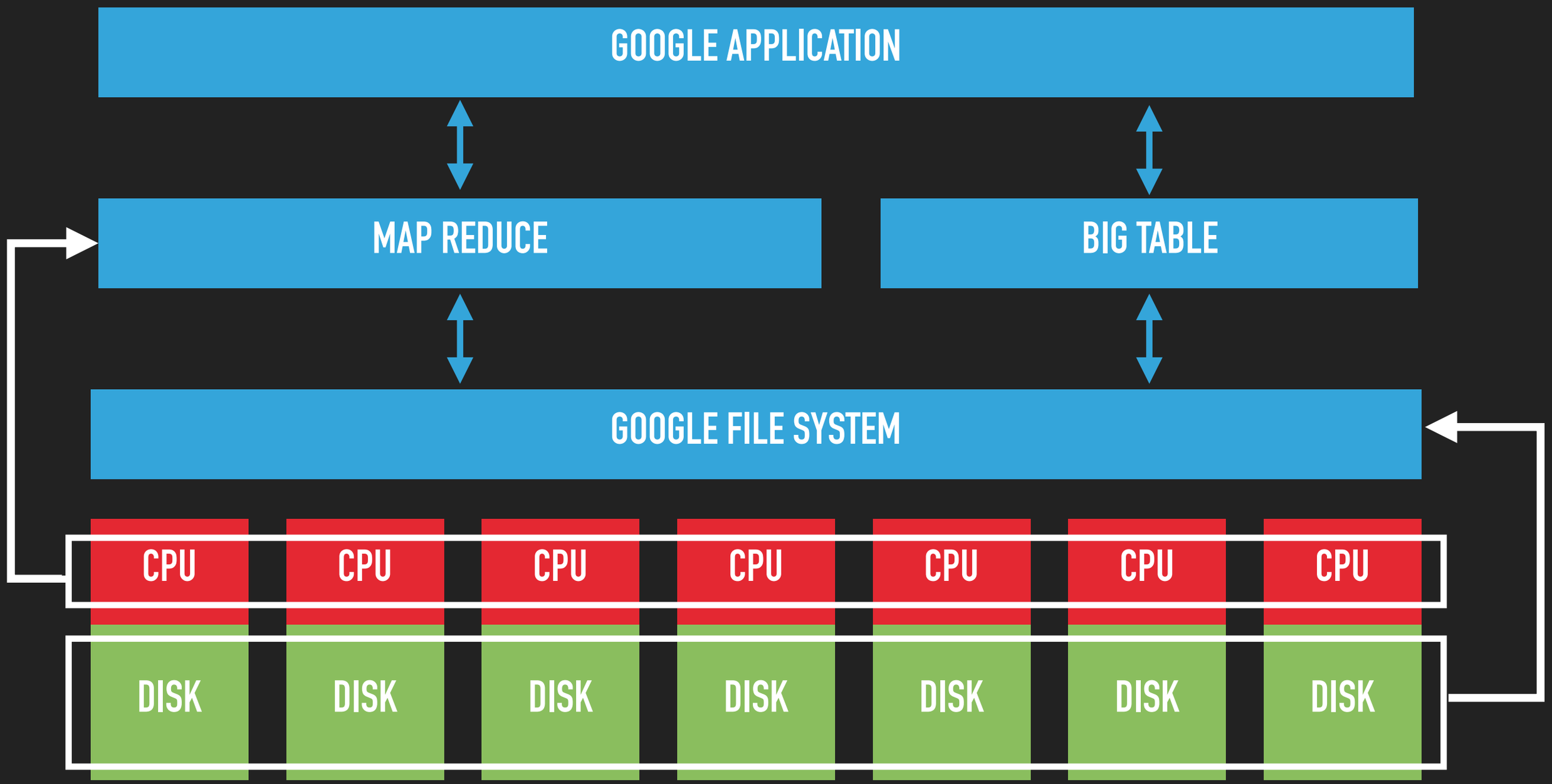
GOOGLE E I BIG DATA

- I database relazionali erano inadeguati per gestire le grandi moli di dati prodotte da Google attraverso l'indicizzazione del web
- Google utilizzata una propria architettura hardware e software per immagazzinare e processare la sempre più crescente quantità di dati

STACK SOFTWARE GOOGLE

- **GFS (Google File System)**
  - File system in grado di astrarre gli storage dei data center, permettendo di accedervi come ad un unico massivo e ridondante file System
- **MapReduce**
  - modello di programmazione per l'esecuzione di thread in parallelo su più server in grado di gestire grandi quantità di dati
- **BigTable:**
  - Sistema database non relazionale che utilizza GFS come storage

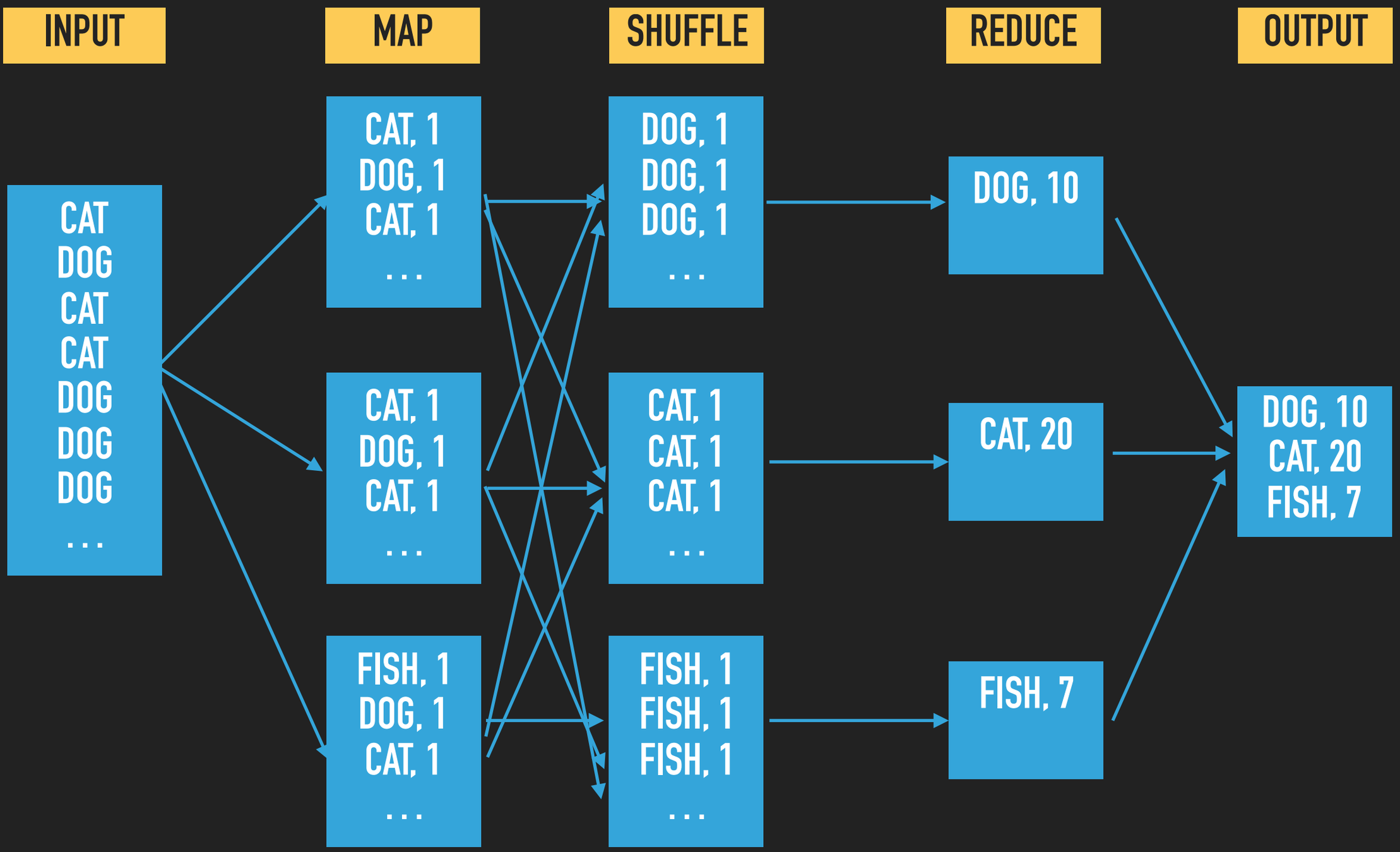
GOOGLE SOFTWARE STACK



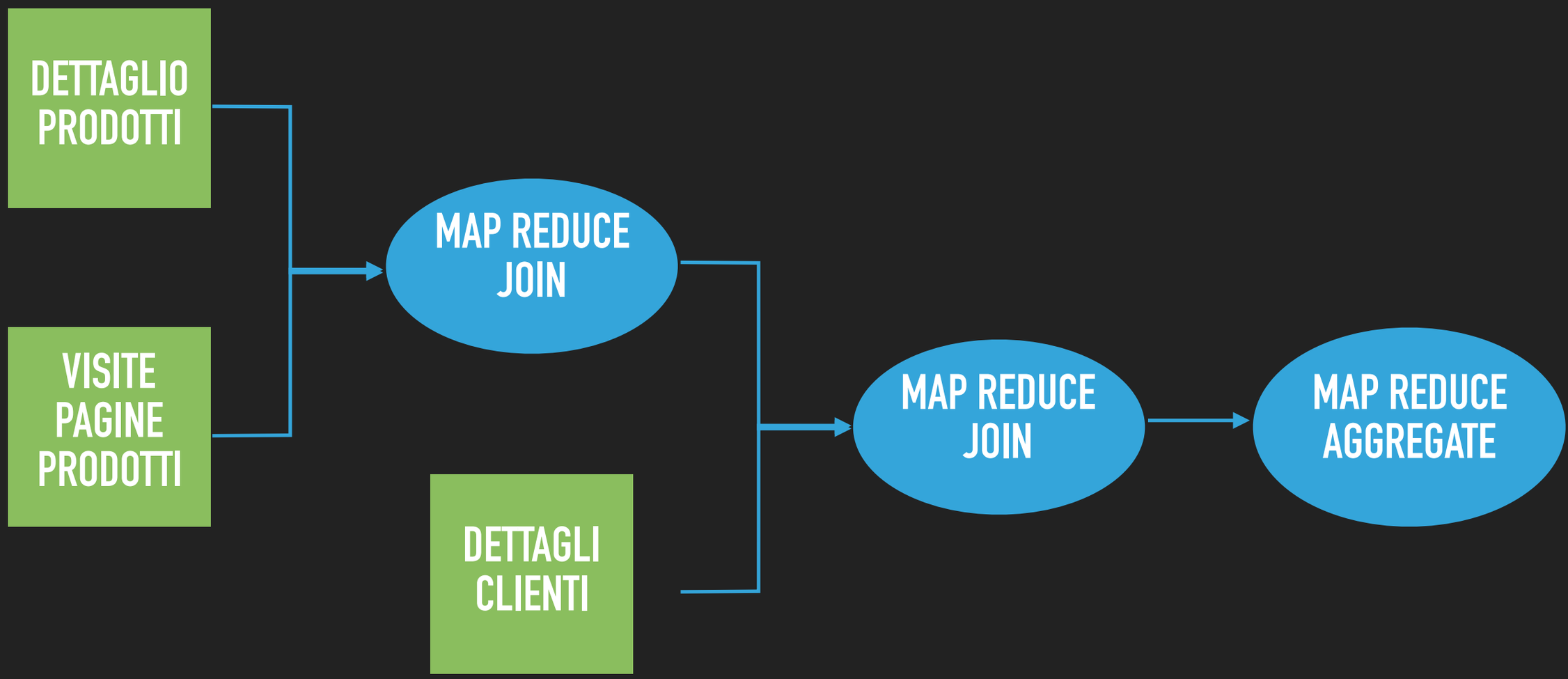
MAP/REDUCE

- Si divide in 2 fasi
  - **Mapping**: l'input viene diviso in chunk di uguali dimensioni e processati da thread che vengono eseguiti in parallelo su server differenti
  - **Reduce**: l'output dei thread viene raggruppato e aggregato per ottenere l'output finale

CONTEGGIO PAROLE



UN ESEMPIO UN PO' PIU' COMPLESSO



HADOOP

- **2003**: Google pubblica i dettagli di GFS
- **2004**: Google pubblica i dettagli di MapReduce
- **2006**: Google pubblica i dettagli di BigTable
- **2007**: prima versione del progetto Open Source Hadoop

COMPONENTI DELL'ECOSISTEMA HADOOP

- **HDFS**: versione open source di GFS
- **HBASE**: versione open source di Big Table
- **HADOOP**: piattaforma di calcolo
  - Resource Manager
  - Node Manager
  - Application Manager

HADOOP

- Motivi del successo di Hadoop
  - Scalabilitá
  - Ridondanza dei dati
  - Schema on read

HBASE

- Versione open source di delle Google Big Table
- Utilizza HDFS allo stesso modo in cui un RDBMS utilizza il file system di un sistema operativo
- Alla base ci sono gli stessi componenti di un normale RDBMS (colonne, tabelle, record, chiavi)
- Per ciascun valore (colonna di un record) é possibile effettuare il versionamento (tramite timestamp)
- Ciascun record può avere il proprio set di colonne

HBASE

NOME	SITO	VISITE
Marco	Google	500
Marco	Ebay	100
Pietro	Facebook	250
Gianni	Facebook	560
Pietro	Google	450
Marco	Amazon	300
Pietro	Amazon	400

HBASE

ID	NOME
1	Marco
2	Pietro
3	Gianni

ID	SITO
1	Google
2	Ebay
3	Facebook
4	Amazon

UTENTE_ID	SITO_ID	VISITE
1	1	500
1	2	250
2	3	300
2	4	100
3	1	300
3	2	430

HBASE

ID	UTENTE	GOOGLE	EBAY	AMAZON
1	Marco	500	300	250

ID	UTENTE	GOOGLE	AMAZON
1	Gianni	500	250

ID	UTENTE	EBAY	AMAZON
1	Pietro	100	300

HIVE

- Infrastruttura datawarehouse costruita su Hadoop per fornire un riepilogo dei dati, interrogazioni e analisi
- E' in grado di gestire i metadati dei dati registrati in HDFS
- Definisce un proprio linguaggio di interrogazione (HQL) che viene tradotto in jobs Hadoop
- Non si tratta comunque di uno strumenti in grado di fornire dati real-time

SHARDING, AMAZON E LA NASCITA DI NOSQL

IL PROBLEMA DELLA SCALABILITA'

- **Web 1.0**: pagine HTML statiche
- **Web 2.0**: pagine HTML generate dinamicamente con capacità di fare richieste transazionali
- Nascita del pattern web server/database server
- **Web server**: facile da scalare
- **Database server**: difficile da scalare

SOLUZIONE OPEN SOURCE

- Utilizzo di **memcached**: possibilità di costruire una cache distribuita di dati rappresentati come oggetti
- **Repliche di database**: le modifiche di un database (master) vengono replicate su altri database (read only)
- Queste tecniche hanno aumentato la capacità di lettura dei database, ma non hanno risolto il problema di bottleneck durante le operazioni di scrittura

SHARDING

- Permette di partizionare un database in più server (**shard**)
- Una tabella applicativa può essere distribuita in più shard
- L'applicazione deve individuare in quale shard si trovano i dati necessari ed inviare la query al server appropriato
- Facile in teoria, ma complesso in pratica

SHARDING

- L'applicazione deve contenere anche la logica per capire in che shard si trova un particolare dato
- Solo il programmatore può interrogare l'intero database
- Perdita di integrità transazionale
- Difficile gestione del load balancing

IL TEOREMA 'CAP'

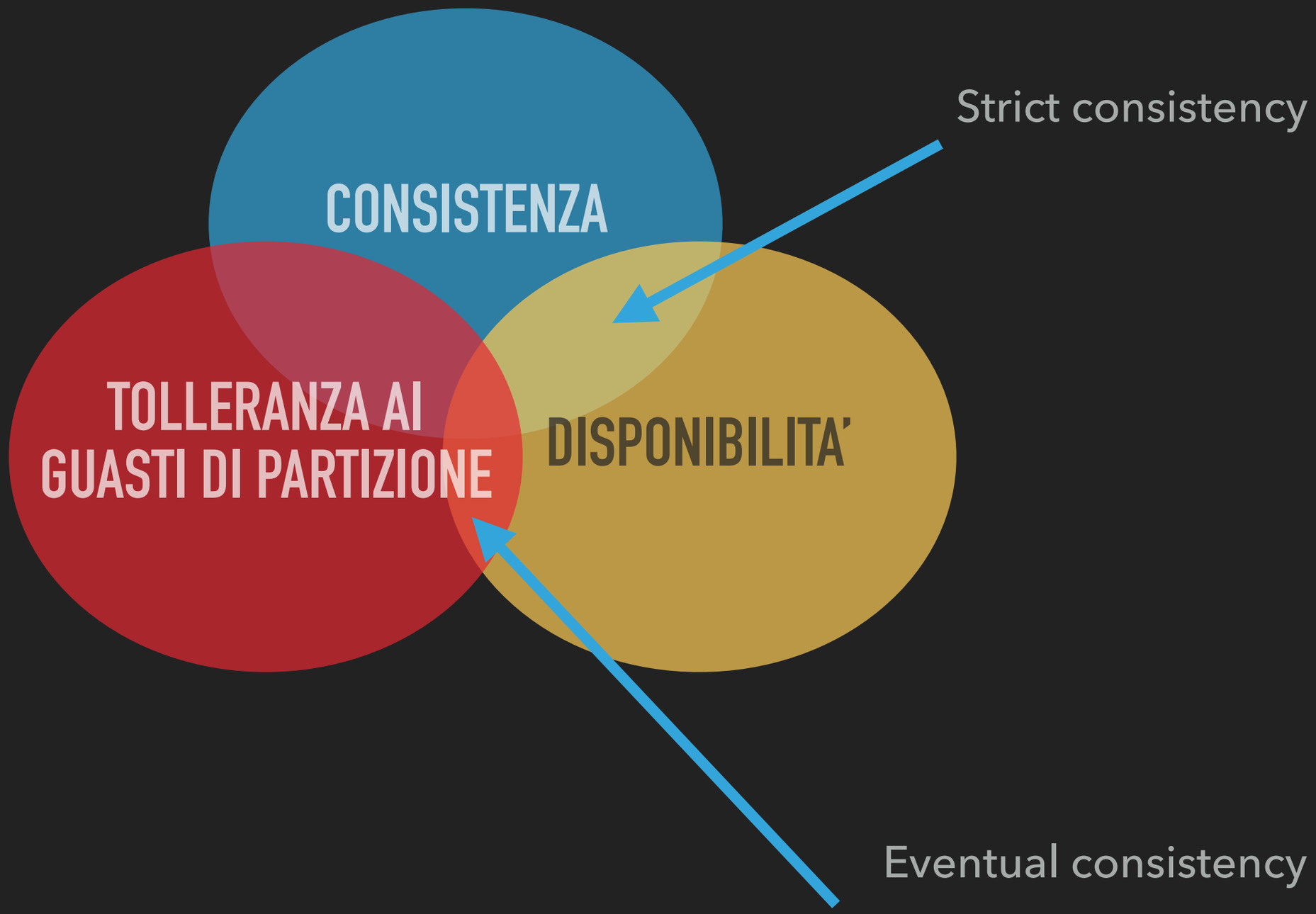
- 2000: Eric Brewer enuncia un teorema tale per cui

> ***In un sistema database distribuito si possono avere al massimo due tra consistenza (Consistency), disponibilità (Availability) e tolleranza ai guasti di partizione (Partition tolerance)***

IL TEOREMA 'CAP'

- **Consistenza**: ogni utente del database ha la stessa vista sui dati in un dato istante
- **Disponibilitá**: in caso di errore di uno o più nodi, il database rimane operativo
- **Tolleranza ai guasti di partizione**: il database rimane operativo in caso di errore di comunicazione tra aree di rete

IL TEOREMA 'CAP'



IL MODELLO DYNAMO

- Amazon é stato il pioniere nell'utilizzo delle tecnologie nate con il Web 2.0
- Inizialmente ha utilizzato Oracle come repository dei propri prodotti, utenti e ordini
- Ha suddiviso il sito in aree funzionali, ognuna delle quali aveva un database dedicato
- E' stato uno dei primi ad adottare un'**architettura orientata ai servizi**
- Nel 2007 ha pubblicato i dettagli di un sistema database non relazionale sviluppato internamente, chiamato **Dynamo**

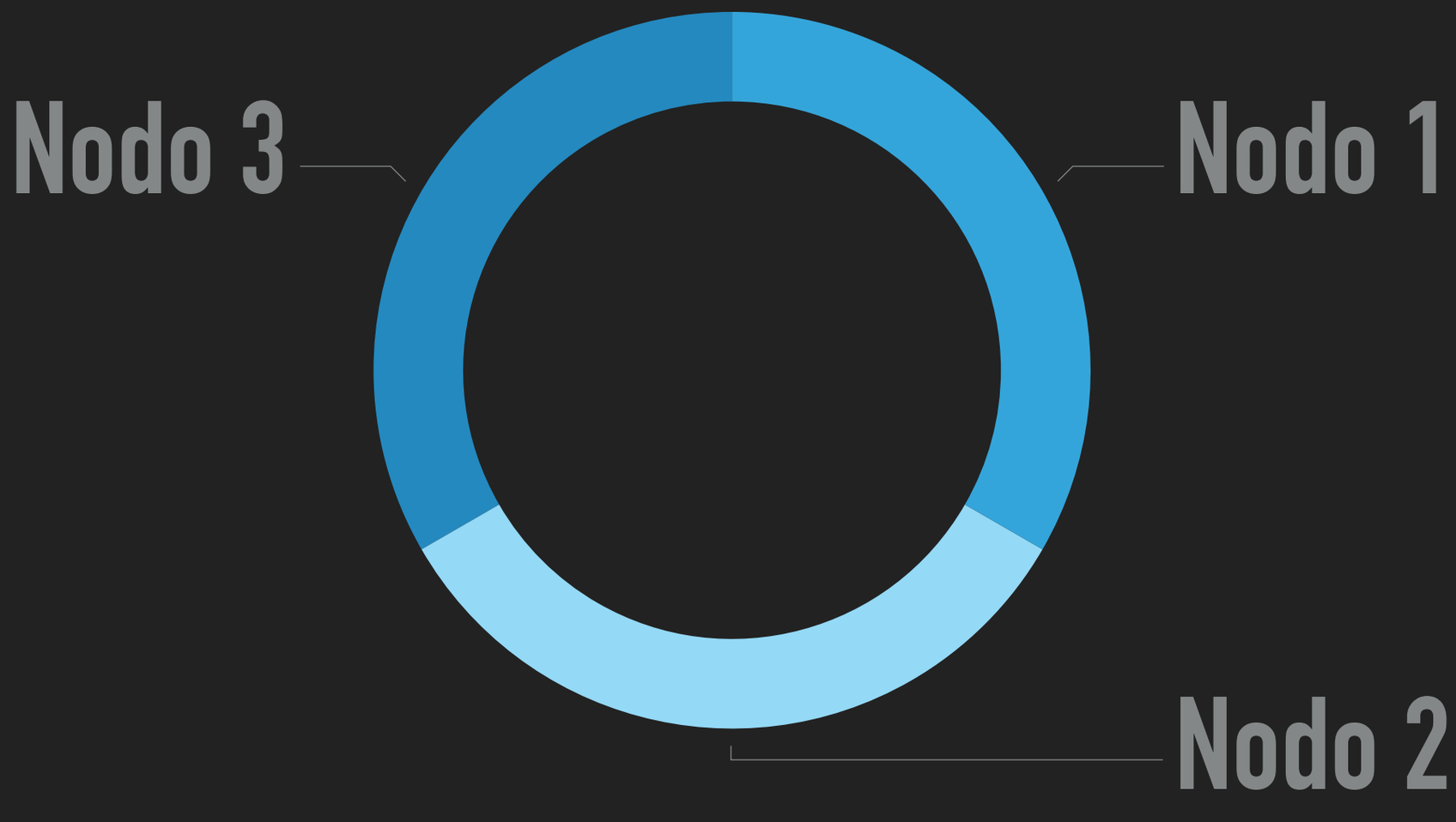
REQUISITI MODELLO DYNAMO

- **Disponibilitá continua**: i dati devono essere sempre disponibili
- **Tollerente ai guasti di rete**
- **Efficiente:** i dati vengono salvati come oggetti non strutturati e accessibili direttamente tramite chiave; la dimensione di questi oggetti deve essere limitata (1 MB)
- **Economico**
- **Scalabile**

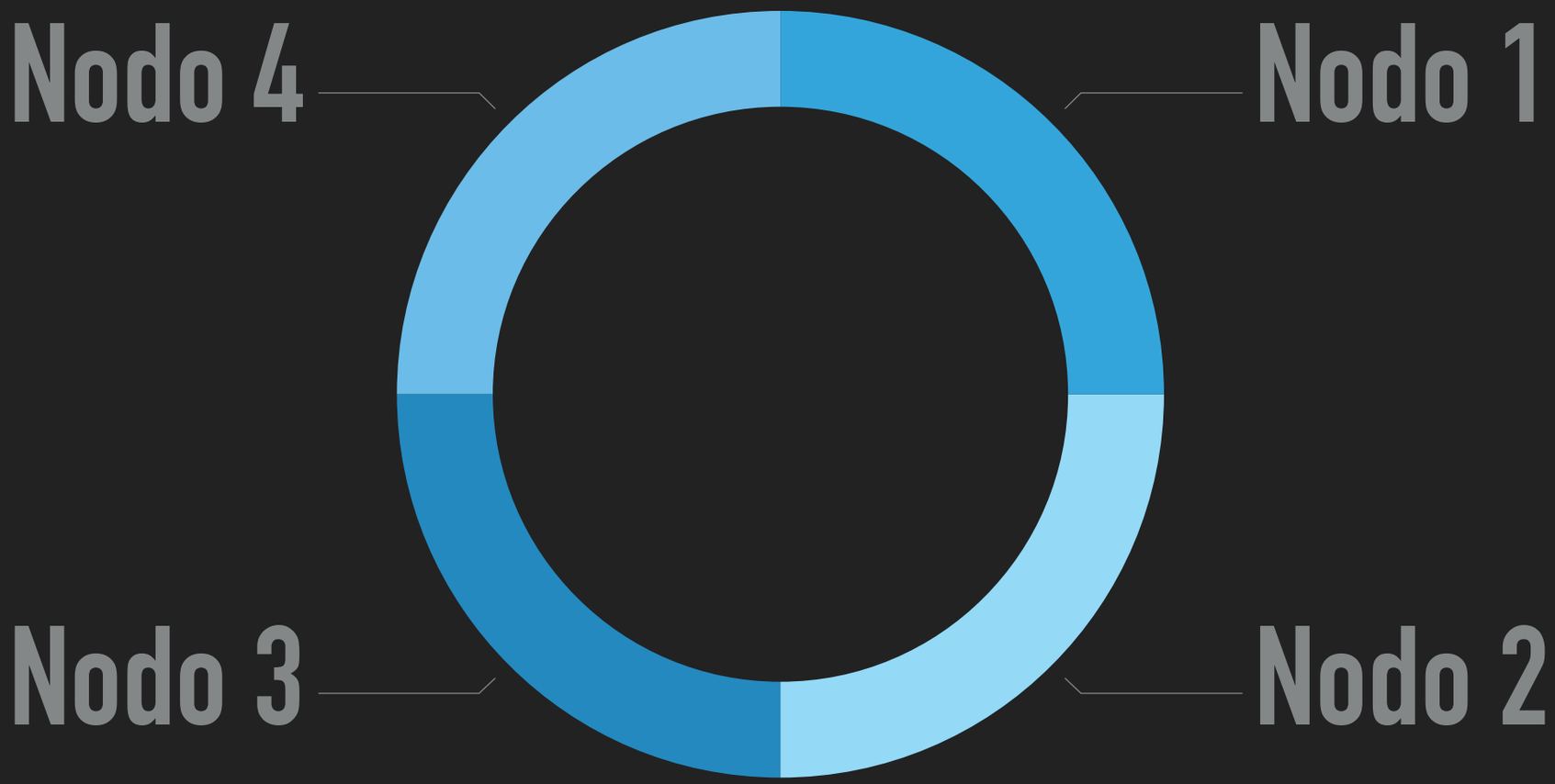
HASHING

- E’ una funzione matematica che viene eseguita su un valore di chiave
- Il risultato della funzione serve per determinare in che nodo deve essere memorizzato il dato
- Una buona funzione di hashing distribuisce in modo omogeneo i dati tra tutti i nodi disponibili
- Difficile da gestire quando si aggiungono o si rimuovono nodi: i dati devono essere ridistribuiti

CONSISTENT HASHING



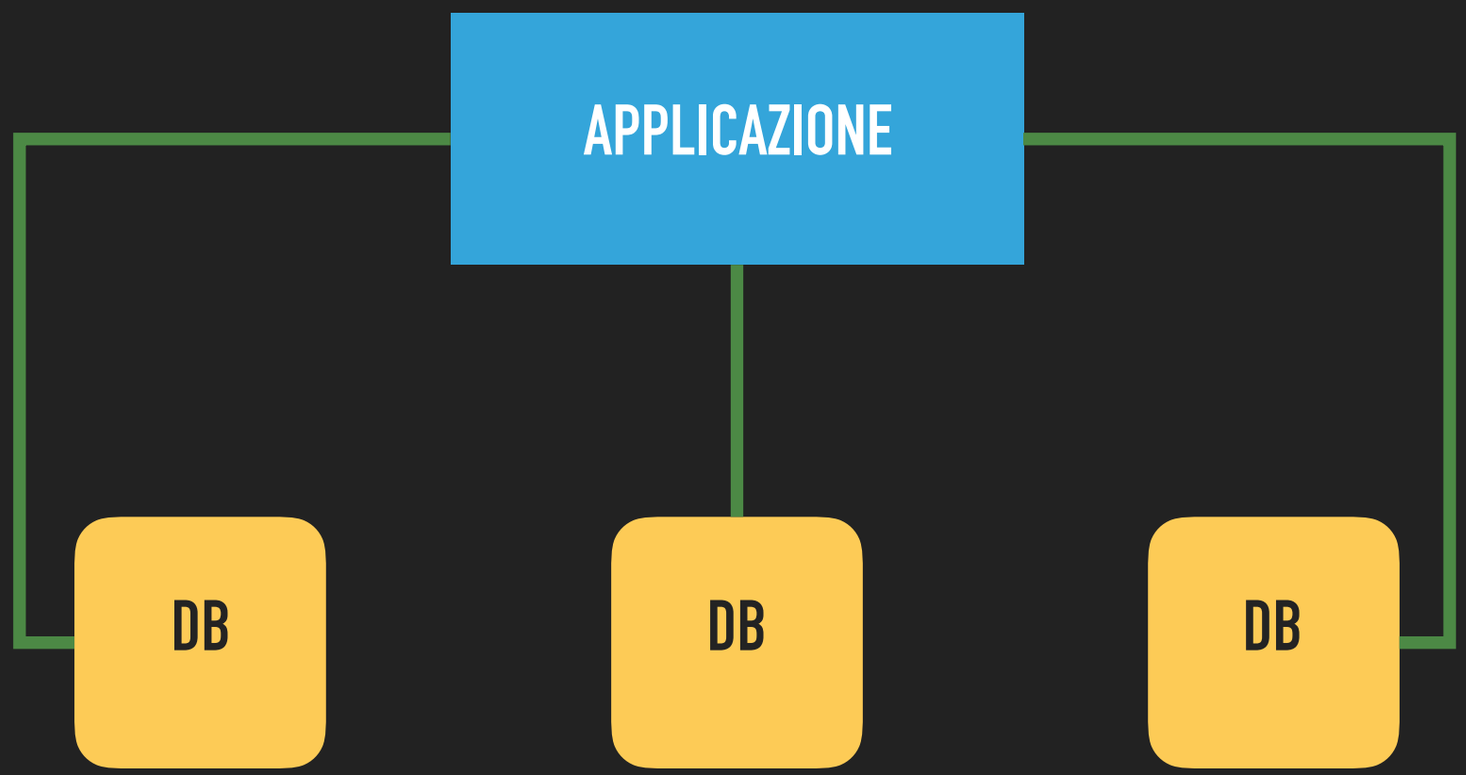
CONSISTENT HASHING



TUNABLE CONSISTENCY

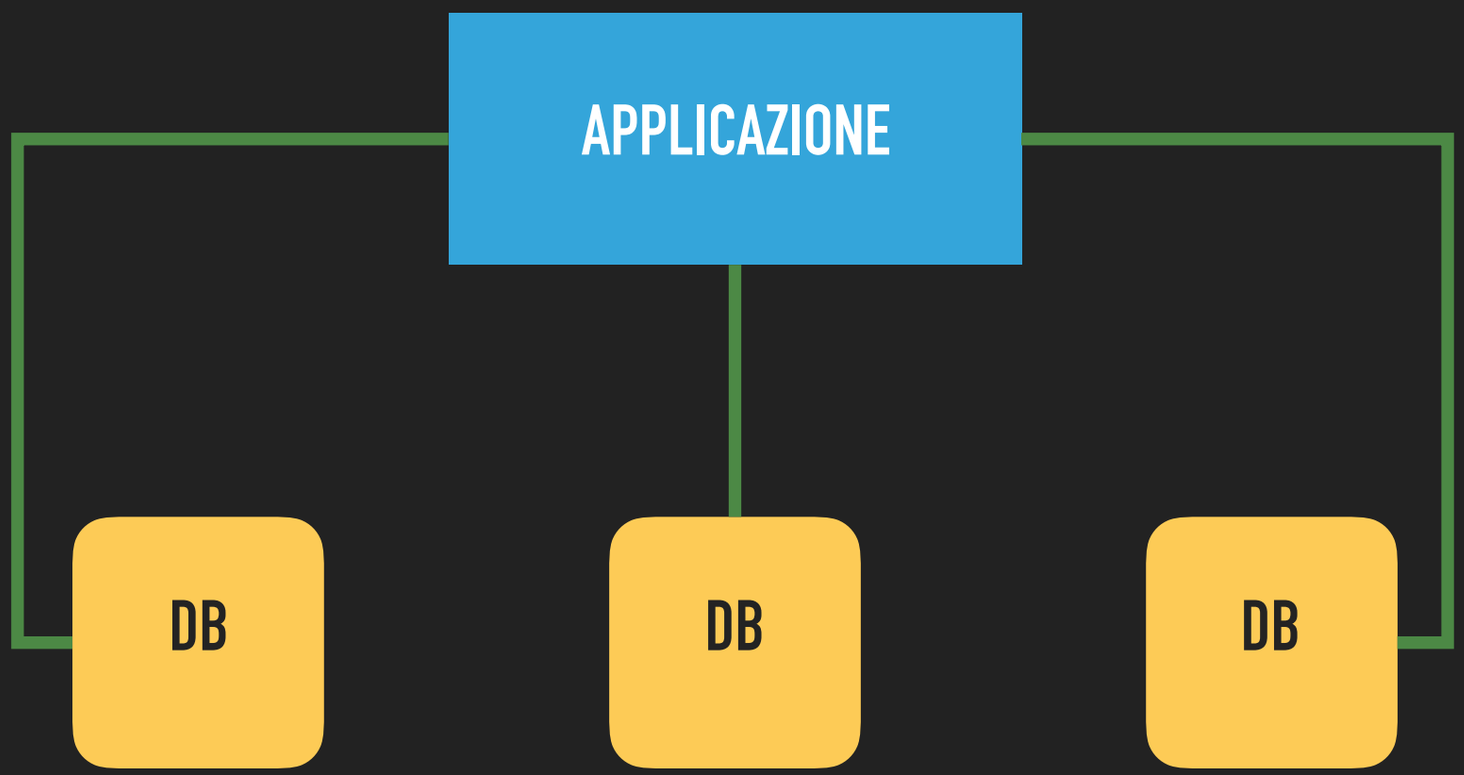
- Dynamo permette di regolare il livello di consistenza dei dati, performance in lettura e performance in scrittura attraverso i parametri:
  - **N**: numero di copie di un singolo dato che il database deve mantenere
  - **W**: numero di copie di un singolo dato che il database deve scrivere prima che una scrittura possa considerarsi conclusa
  - **R**: numero di copie alle quali l'applicazione deve accedere quando un dato deve essere letto

TUNABLE CONSISTENCY



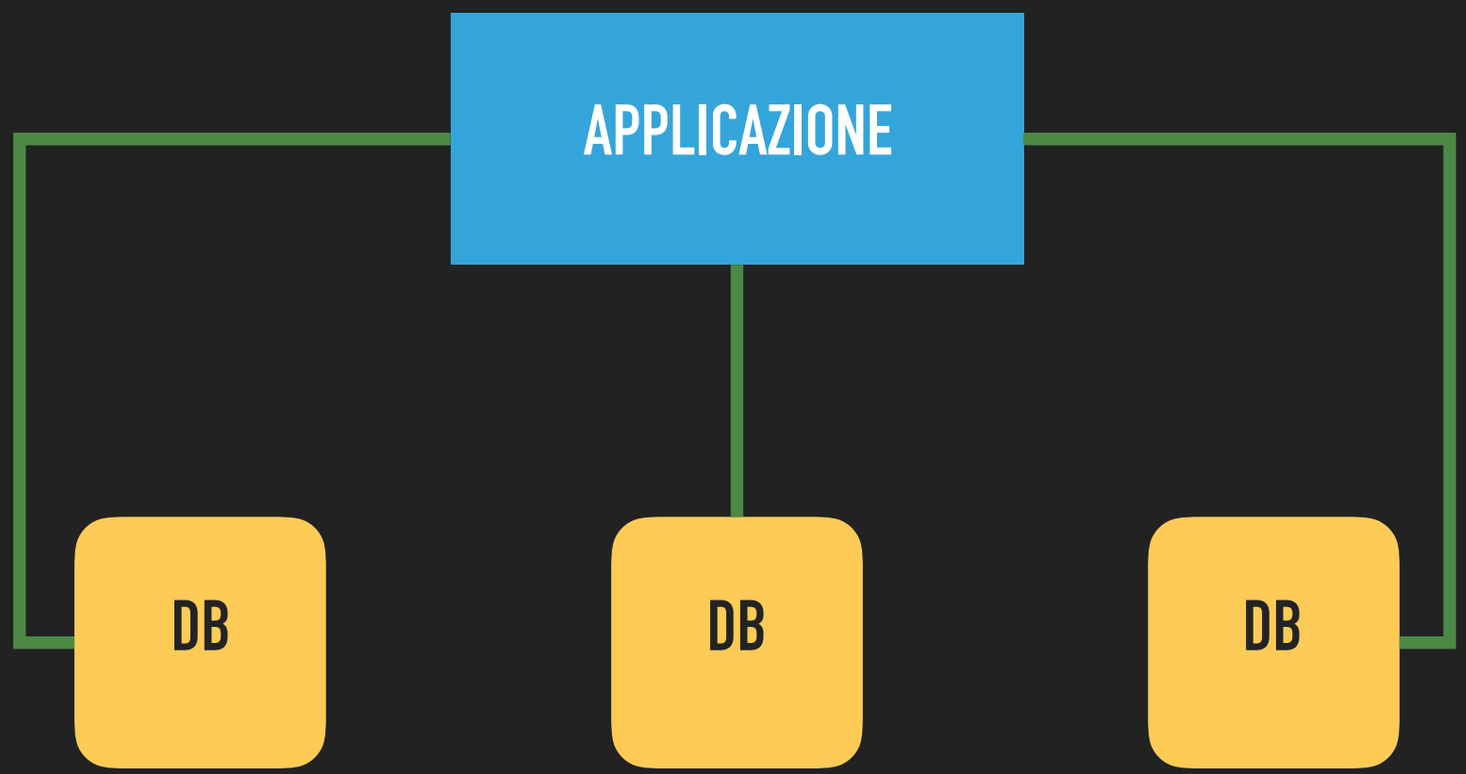
N=3 W=3 R=1
SCRITTURE LENTE
LETTURE VELOCI
CONSISTENTE

TUNABLE CONSISTENCY



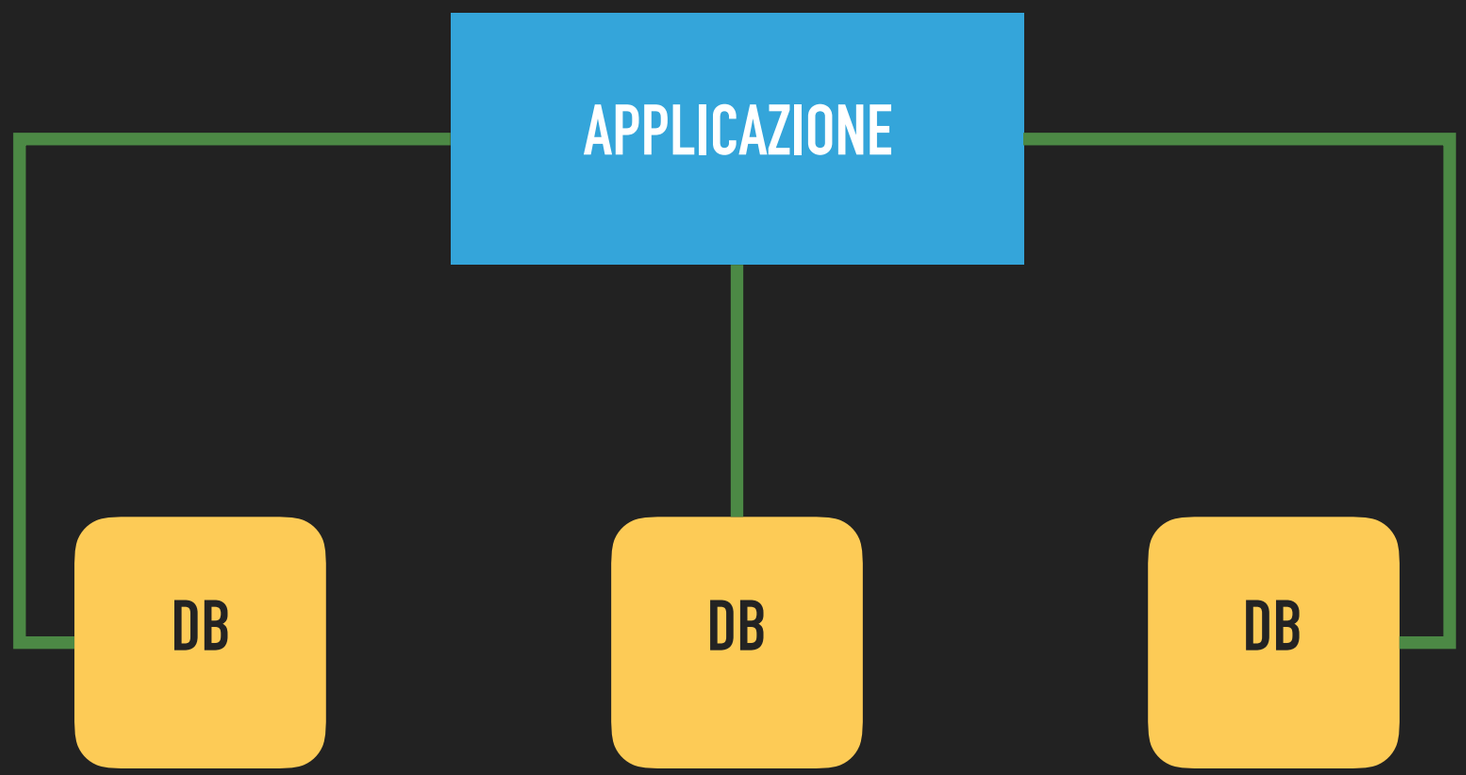
N=3 W=2 R=2
SCRITTURE VELOCI
CONSISTENTE

TUNABLE CONSISTENCY



N=3 W=1 R=N
SCRITTURE VELOCI
LETTURE LENTE
CONSISTENTE

TUNABLE CONSISTENCY



N=3 W=1 R=1
SCRITTURE VELOCI
LETTURE VELOCI
NON CONSISTENTE

DOCUMENT DATABASE

DOCUMENT DATABASE

- Una database documentale é un database che memorizza i dati come documenti strutturati, tipicamente in formato JSON o XML
- L'emergere di questo tipo di database si deve al fatto che risolvono il conflitto tra la programmazione ad oggetti ed il modello relazionale
- Il formato autodescrittivo dei documenti rende il dato indipendente dalla piattaforma su cui é stato creato
- Di default non implementano nessun modello transazionale o infrastrutturale

XML VS JSON

- I primi database documentali memorizzavano i dati come documenti XML
- XML richiede però molto spazio per essere memorizzato ed il costo computazione di parsing é abbastanza elevato
- Durante i primi anni del 2000, grazie all'esplosione delle tecnologie Web 2.0, viene alla ribalta un nuovo formato, ***JSON (Javascript Object Notation)***

DOCUMENT DATABASE

- **Documento**
  - Rappresenta l’unitá base di memorizzazione di un dato
  - Possiamo paragonarlo approssimativamente ad un record di un database relazionale
    - E’ composta da una o più coppie chiave/valore
    - Ciascun valore può essere a suo volta un altro documento o un array di documenti
- **Collezione**
  - Insieme di documenti
  - Possiamo paragonarla approssimativamente ad una tabella di una database relazionale

DATA MODEL IN UN DATABASE DOCUMENTALE

- Normalmente, i database documentali non permettono di effettuare operazioni di join
- La struttura di un documento dovrebbe già assomigliare alla struttura degli oggetti utilizzati nel codice
- I documenti innestati hanno il vantaggio di poter ottenere tutti i dati con una sola operazione evitando operazioni di join
- Dall'altra parte si ha
  - Duplicazione di dati
  - Possibile inconsistenza di dati
  - Difficile gestione delle modifiche

DATA MODEL IN UN DATABASE DOCUMENTALE

- In un database relazionale, la modellazione dei dati é guidata dalla natura stessa dei dati
- In un database documentale, la modellazione dei dati é guidata dalla natura dell’utilizzo finale

DATABASE A GRAFO

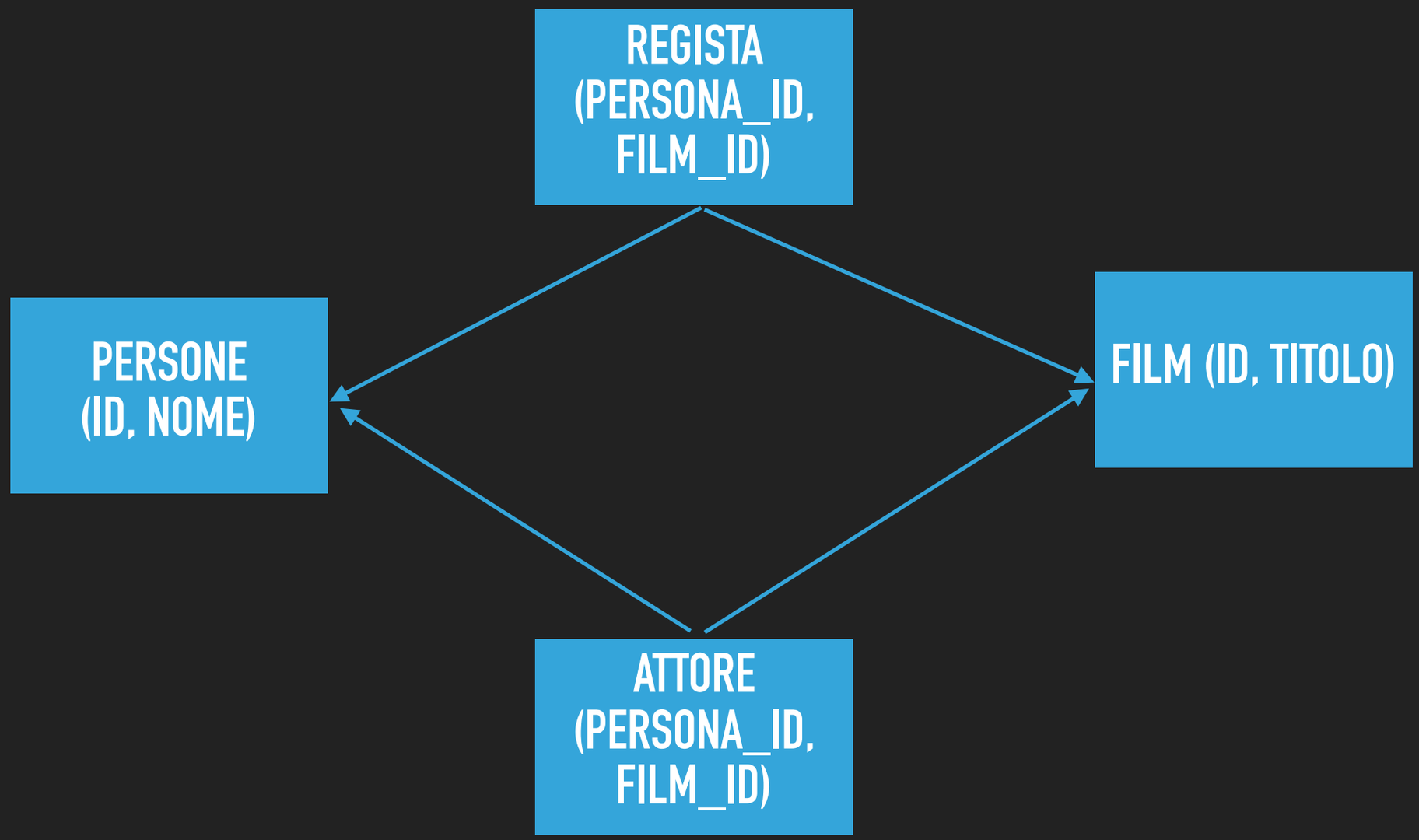
L'IMPORTANZA DELLE RELAZIONI

- Database relazionali organizzano dati in tabelle
- Database documentali organizzano dati in collezioni
- In certi contesti sono più rilevanti le relazioni tra i dati che i dati stessi (ad esempio all'interno dei social network)
- Il modello relazionale é in grado di gestire queste informazioni, ma non in maniera efficiente

TEORIA DEI GRAFI

- Un grafo é composto da:
  - **Nodi**: rappresentano gli oggetti distinti all'interno del grafo
  - **Archi**: rappresentano i collegamenti tra 2 nodi
- Sia i nodi che gli archi possono avere delle proprietà
- Un grafo può essere percorso attraverso delle operazioni di attraversamento

PROGETTARE UN GRAFO IN UN RDBMS



TEORIA DEI GRAFI

- La sintassi ed SQL stesso non sono adatti per l'attraversamento in profondità di un grafo, soprattutto se questa é sconosciuta
- Le performance degradano velocemente mano a mano che la profondità aumenta

NEO4J

- E’ il database a grafo più diffuso
- Scritto in Java
- In grado di gestire milioni di nodi
- Implementa il modello ACID
- Implementa il linguaggio dichiarativo **Cypher**, in grado di interrogare il database in maniera simile ad SQL, ma in maniera molto più espressiva e performante

NEO4J

Qualche esempio...

DATABASE A COLONNE E DATAWAREHOUSE

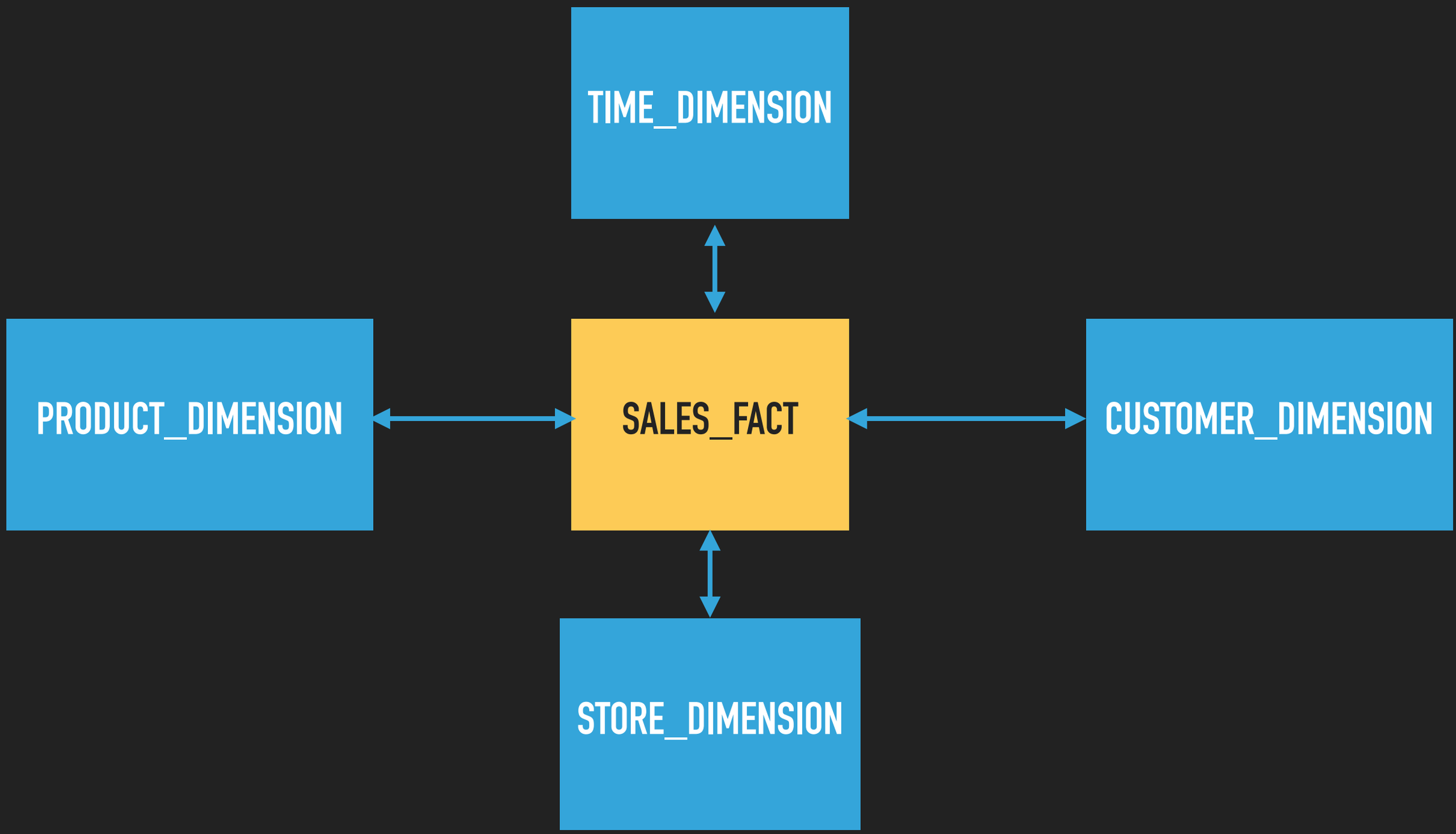
I DATAWAREHOUSE

- Necessità di produrre report dai dati presenti all'interno dei database relazionali applicativi
- Non possono essere utilizzati i database applicativi di produzione a causa della grande mole di dati da elaborare
- La schema stesso dei database relazionali applicativi non é adatto per l'esecuzione di questi complesse
  - I database applicativi sono progettati per soddisfare requisiti CRUD

I DATAWAREHOUSE

- Necessitá di uno schema ottimizzato per la sola lettura dei dati
  - Utilizzo di indici
  - Normalizzazione rilassata e ridondanza dei dati accettabile
  - Schema progettato sulle esigenze di interrogazione
- Al crescere dei dati, si continua ad avere un peggioramento delle prestazioni

I DATAWAREHOUSE



DATABASE A COLONNA

BLOCK	NAME	BIRTHDAY
1	James	22/07/1987
2	Frank	01/01/1990
3	Bob	03/03/2005
4	Robert	04/05/1998
5	Dan	12/05/1999
6	Steven	13/02/2000

DATABASE A COLONNA

BLOCK				
1	James	Frank	Bob	Robert
2	01/01/2000	23/02/1997	12/09/2008	13/04/2000

I DATABASE A COLONNA

- Le query che aggregano valori su una specifica colonna sono ottimizzate perché riducono drasticamente il numero di accessi al disco
- Possibilità di comprimere i dati:
  - Compressione di dati ripetuti nella stessa porzione di disco
  - Se ordinati, possibilità di rappresentare ciascun dato come delta rispetto al dato precedente

IN MEMORY DATABASE

LIMITI DEI SUPPORTI FISICI

- I/O verso dischi magnetici sono da sempre diversi ordini di grandezza inferiori agli accessi verso la memoria o la CPU
- Performance migliorare grazie alla nascita di memorie SSD, ma le prestazioni rimangono limitate, specialmente in scrittura
  - Per sfruttare al massimo le caratteristiche SSD é necessario sviluppare database in grado di sfruttarle

IN MEMORY DATABASE

- Alcuni database sfruttano una cache in memoria per migliorare le prestazioni
- I database tradizionale però prevedono l'accesso al disco per operazioni frequenti
  - Commit
  - Transaction Log
  - Checkpoint
- E' necessaria un'architettura che preveda e consenta la completa gestione dei dati in memoria e senza perderli in caso di errore

IN MEMORY DATABASE

- Rispetto ai database tradizionali:
  - Non hanno cache (stiamo già gestendo i dati in memoria)
- Differente modello di persistenza attraverso:
  - Replica dei dati su altri cluster
  - Scrittura di un'immagine dell'intero database su disco

TIMES-TEN

- E’ un database relazionale creato nel 1995 e acquisito da Oracle nel 2005
- La persistenza é realizzata attraverso scritture periodiche dei dati su disco (le scritture sono asincrone)
- Su se verifica un errore durante la scrittura su disco, alcuni dati possono andare persi

REDIS

- E’ un database chiave/valore sviluppato nel 2009
- Se la dimensione dei dati supera la memoria a disposizione é in grado di gestire un’area di swap su disco
- La persistenza é realizzata attraverso la scrittura dell’intero database su disco:
  - La scrittura può avvenire:
    - On demand
    - Schedulazione
    - Superamento di una quota di dati da scrivere

SPARK

- Sviluppato nel 2011, rappresenta la versione in memory del modello MapReduce di Hadoop
- Astrae il modello di MapReduce, semplificandone lo sviluppo e aumentandone la produttività
- Trattandosi di un modello di programmazione non necessita di alcun metodo di persistenza