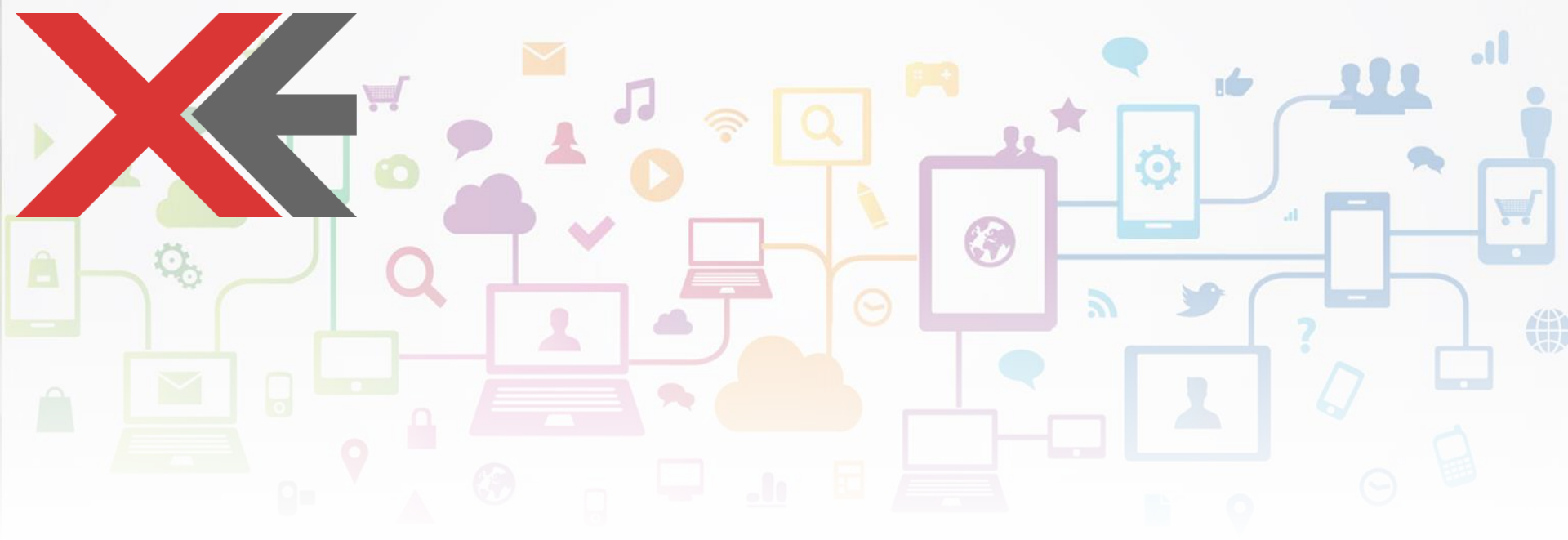


RDBMS: pregi e difetti

Gianluca Hotz
@glhotz | Presidente UGISS.ORG | Data Platform MVP

Chi sono

- Gianluca Hotz | @glhotz | ghotz@ugiss.org
- Fondatore e Mentor SolidQ
  - 20+ anni con SQL Server (dalla 4.21 nel 1996)
  - Modellazione basi di dati, dimensionamento e amministrazione, sviluppo, ottimizzazione
- Interessi
  - Modello relazionale, architettura DBMS, alta disponibilità e Disaster Recovery
- Community
  - 20 anni Microsoft MVP SQL Server (dal 1998)
  - Fondatore e presidente UGISS
    - User Group Italiano SQL Server (PASS Chapter)

Gestione delle Informazioni

- Modello dati
  - Logico? Fisico? Entrambi?
- Data Store specializzati
  - Document? Key-Value? Column-Oriented (Columnar)? Graph?
- Interrogazione
  - Dichiarativa? Imperativa? Mix?
- Funzioni accessorie
  - Gestione transazioni? Persistenza? Concorrenza? Consistenza? Replica? Sharding?
- Terminologia poco chiara e in continuo mutamento…
  - Relational? SQL? NoSQL? NewSQL?
  - DBMS? RDBMS? ORDBMS?

Modello Relazionale

- Modello per la gestione dei dati dichiarativo
  - per specificare i dati (schema e valori)
  - per specificare interrogazioni (query)
  - struttura e linguaggio basati su logica dei predicati
  - puramente logico/concettuale
- Nato per sopperire a limitazioni modelli
  - «Hierarchy»
  - «Network»
- Letteratura storicamente molto formale
  - Prima descrizione di Edgar F. Codd (1969)
  - Molti libri anche recenti Christopher J. Date
  - Moltissimi «paper» accademici (50 anni)

Cos’è un database relazionale?

- Collezione strutturata di fatti
  - sottoinsieme di fatti: del mondo reale, di interesse
- Rappresenta proposizioni considerate vere
  - assiomi
- Permette di derivare nuove proposizioni
  - tramite regole formali di inferenza

https://en.wikipedia.org/wiki/Fabian_Pascal

Proposizioni e predicato

- Esempio: database per dati relativi a prenotazioni
- Fatti (proposizioni)
  - La prenotazione identificata dal numero **990000**, per la stanza **333**, prevede la data di arrivo **29/10/2006** e la data di partenza **01/11/2006**.
  - La prenotazione identificata dal numero **990001**, per la stanza **275**, prevede la data di arrivo **27/10/2006** e la data di partenza **01/11/2006**.
- Forma generalizzata (predicato)
  - La prenotazione identificata dal numero (**IDPrenotazione**), per la stanza (**NumStanza**), prevede la data di arrivo (**DataArrivo**) e la data di partenza (**DataPartenza**).

Basi del modello relazionale

- Concetto di relazione
  - non è la traduzione di relationship del modello E/R!
- Relazione matematica
  - sottoinsieme del prodotto cartesiano di domini
  - insieme di n-uple
    - (v1, v2, ..., vn) con v1 $\in$ D1, v2 $\in$ D2, …, vn $\in$ Dn
  - n-uple ordinate all'interno (stesso ordine dei domini)
- Relazione nel modello relazionale
  - n-uple (tuple) non sono ordinate all'interno
  - riferimento per nome (attributi)

Dati nel modello relazionale

- Consideriamo i domini
  - D1 = costituito da numeri interi
  - D2 = costituito da numeri interi
  - D3 = costituito da date
  - D4 = costituito da date
- Alcune possibili relazioni su D1, D2, D3, D4
  - {(990000, 333, 29/10/2006, 01/11/2006)}
  - {(990000, 333, 29/10/2006, 01/11/2006),
    (990001, 275, 27/10/2006, 30/10/2006)}

Rappresentazione con tabelle

- Una tabella per ogni variabile di tipo relazione
- Nomi degli attributi sono intestazioni delle colonne
- Colonne non sono ordinate
- Righe sono n-uple (tuple) di valori non ordinate
- Righe distinte una dall'altra
  - in un insieme non possono essere presenti due valori uguali
  - una tabella rappresenta una relazione se le righe sono l'una diversa dall'altra

Relazioni e predicato

- Relazione rappresenta un predicato
  - interpretazione di una porzione del mondo reale di interesse
- Istanza del predicato
  - sostituendo le variabili con i valori delle tuple otteniamo una proposizione sempre vera per convenzione

Rappresentazione logica

- Predicato
  - La prenotazione identificata dal numero (**IDPrenotazione**), per la stanza (**NumStanza**), prevede la data di arrivo (**DataArrivo**) e la data di partenza (**DataPartenza**).
- Relazione
  - {(990000, 333, 29/10/2006, 01/11/2006), (990001, 275, 27/10/2006, 30/10/2006)}

IDPrenotazione	NumStanza	DataArrivo	DataPartenza
990000	333	2006-10-29	2006-11-01
990001	275	2006-10-27	2006-10-30

Implementazione reale RDBMS

- 12 regole di Codd
  - https://en.wikipedia.org/wiki/Codd%27s_12_rules
  - non tutti aderiscono a tutte…
  - indipendenza: fisica, logica, integrità, distribuzione
- Linguaggio SQL
  - «approssimazione ingegneristica»
  - basato sul calcolo relazionale

Proprietà ACID

- Non strettamente legate al modello relazionale
- Proprietà gestione transazioni desiderabili
- Atomicità
  - transazione eseguita in modo indivisibile (tutto o niente)
- Consistenza (o coerenza)
  - database deve essere coerente con i vincoli e non contraddittorio prima e dopo l'esecuzione della transazioni
- Isolamento
  - livello di interferenza tra transazioni in esecuzione contestuale
- Durabilità (o persistenza)
  - le modifiche confermate devono essere persistite in modo permanente (nessun guasto successivo potrà determinare la perdita dei cambiamenti)

https://en.wikipedia.org/wiki/ACID

Implementazione fisica SQL Server

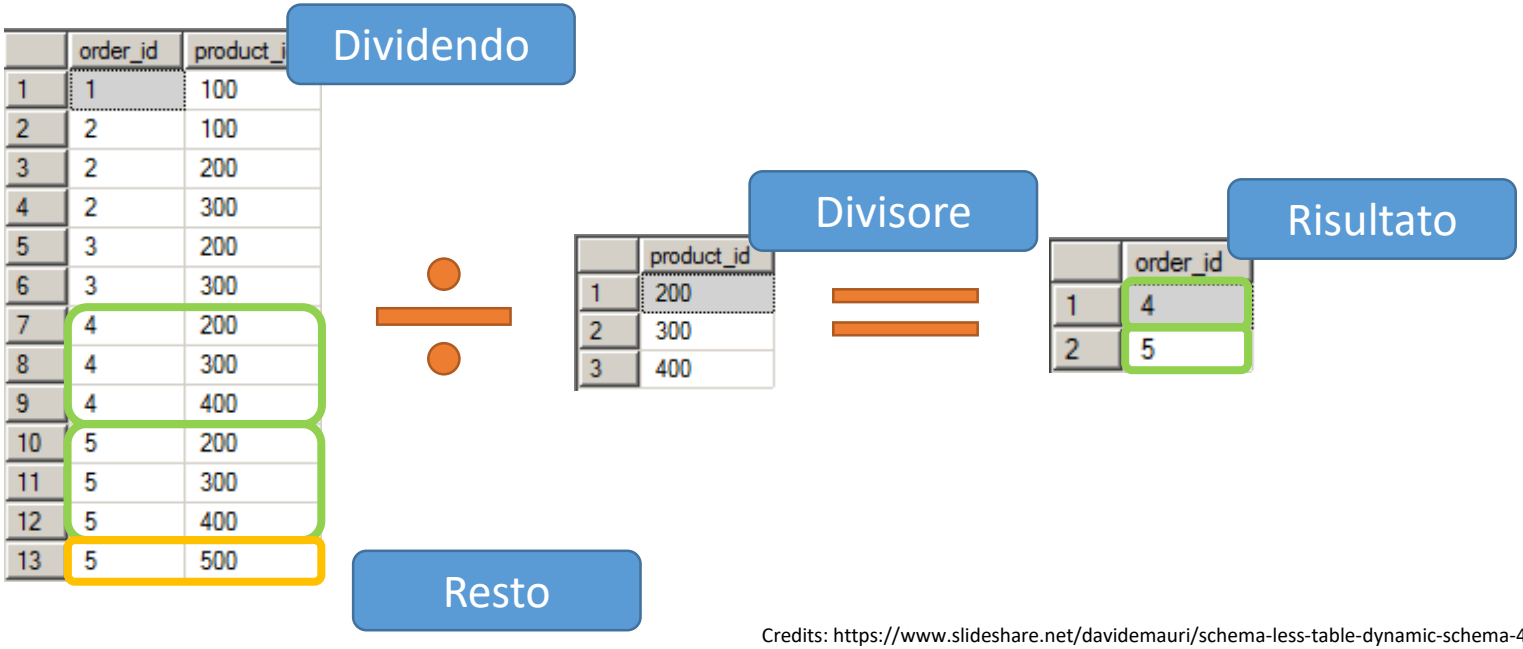
- Tabelle
  - Heap
  - B+Tree
  - Columnstore
  - Hash (In-Memory Tables)
- Indici
  - B+Tree
  - Full-text
  - XML
  - Spatial
  - Columnstore
  - BW-Tree

T-SQL: linguaggio dichiarativo

- Specificando il «cosa» al posto del «come»
  - lascia spazio a continui miglioramenti del «come»
- Ottimizzatore delle Query
  - Interpretazione richieste complesse
    - equivalenze logiche, individuazione contraddizioni, …
  - Ottimizzazione piano di esecuzione
    - algoritmi, parallelismo, utilizzo di indici, …
  - Riutilizzo dei piani di esecuzione
    - anche in modalità adattiva

Esempio: divisione Relazionale

- Trovare sottoinsieme di…
  - competenze richieste dai candidati per un lavoro
  - aree geografiche dove diversi clienti/fornitori operano
  - componenti condivisi da diverse distinte base
  - prodotti venduti insieme in ordini distinti



Credits: https://www.slideshare.net/davidemauri/schema-less-table-dynamic-schema-44295422

DEMO

Ottimizzatore delle Query

«Schema»

- Definizione a priori di una struttura dati
- Modifiche ai dati solo se conformi a schema
- Esempi
  - Tabelle (e vincoli) di un RDBMS
  - XML Schema
  - Class, Struct

«Schema-less»

- Nessuna definizione, caos completo ☺
- Solitamente «Schema» implicito
  - sparso nelle applicazioni e definito «al volo»
  - semplice da estendere ma…
  - …mantenimento vincoli problematico!
- Integrità dei dati è un valore!
  - altrimenti abbiamo dati non informazioni ☺
  - senza integrità estrarre informazioni dai dati diventa
    - difficile
    - dispendioso
    - inaffidabile

«Schema» e assenza informazioni

- A-Mark
  - valore applicabile ma assente
  - es. no carta identità durante inserimento anagrafica
- I-Mark
  - valore inapplicabile
  - es. data ultima gravidanza paziente maschio
- Gestione tramite il singolo marker NULL
  - problemi logica TRUE, FALSE, UNKNOWN

«Schema» in SQL Server

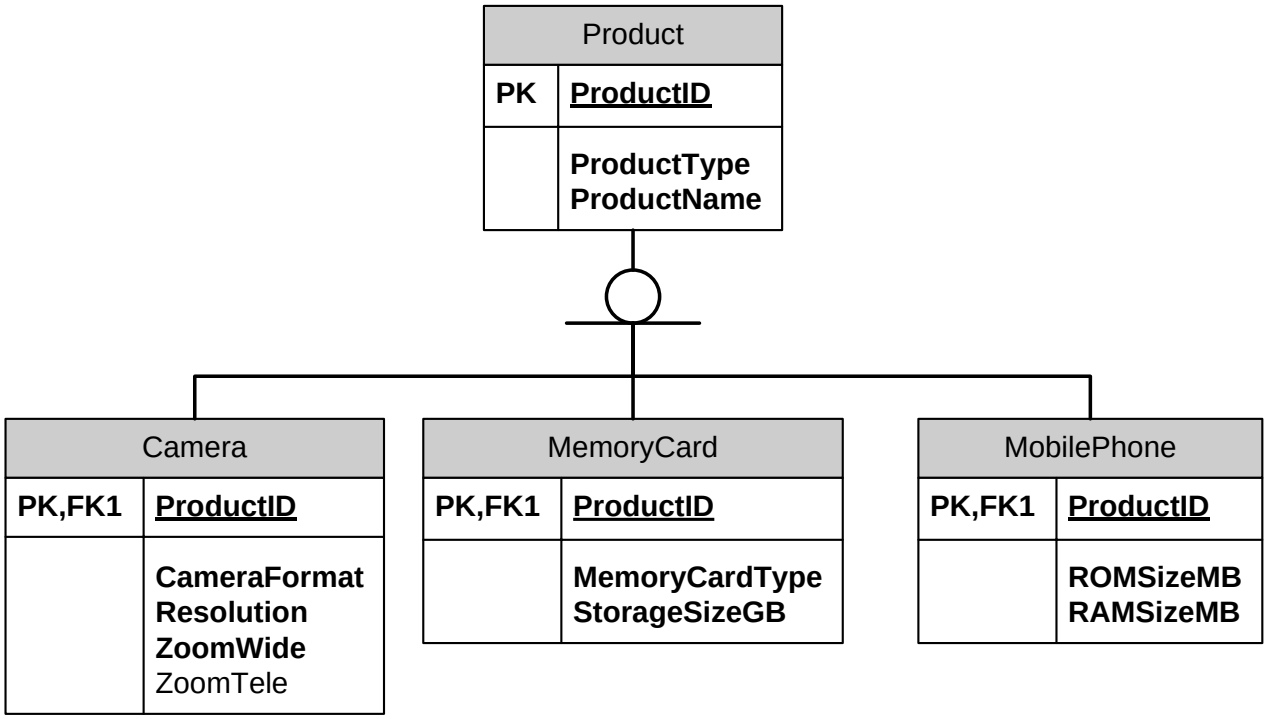
- Ricchezza tipi dato complessi
  - XML, GEOMETRY, GEOGRAPHY, HIERARCHYID, FILESTREAM, FILETABLE, JSON
  - estensibile con integrazione CLR
  - Integrazione SQL (XPATH/XQUERY, funzioni JSON)
- Vincoli dichiarativi
  - NULL/NOT NULL, DEFAULT, CHECK
  - PRIMARY KEY, FOREIGN KEY
  - «XML Schema Collection»
- Vincoli procedurali
  - «After Trigger»
  - «Instead of Trigger»

DEMO

Tipi dato complessi

Valori inapplicabili

- Rimuovibili con modello semanticamente più ricco
- Utilizzo di generalizzazioni/specializzazioni



Rigidità «Schema»

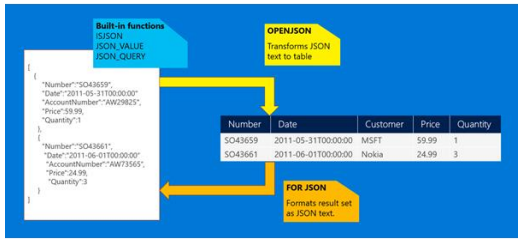
- Spesso problema riguarda estensibilità
  - sottoinsieme noto e stabile di attributi
  - agilità nell'aggiungere ulteriori attributi
  - anche noto come «Open Schema» o «Dynamic Schema»
- Diversi approcci
  - Aggiunta di colonne specifiche
  - Incapsulamento
  - Modello EAV

Aggiunta di colonne

- Richiede una modifica allo schema
  - ALTER TABLE
  - possibile impatto su applicazioni (es. SELECT *)
  - possibile impatto disponibilità/prestazioni
  - consumo spazio senza «Sparse Columns»
- «Sparse Columns»
  - disponibili a partire da SQL Server 2008
  - non consumano spazio se non valorizzate ☺
  - consumano più spazio se valorizzate ☹
  - materializzazione colonna XML con attributi valorizzati

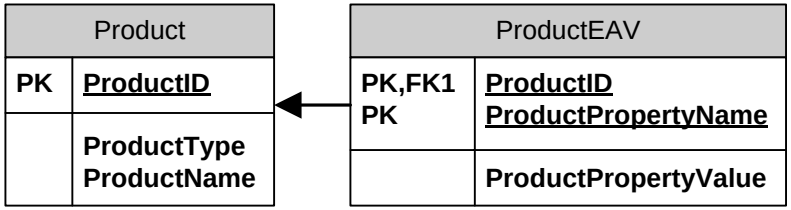
Incapsulamento

- Colonne di tipo strutturato complesso
- BLOB
  - stream binario o tipi implementati con CLR
  - utile principalmente per persistenza, nessun supporto
- XML
  - supporto completo con XML Schema, indici, linguaggio
  - prestazioni buone ma non ottimali
  - utilizzano molto spazio
- JSON
  - materializzazione colonne calcolate e indicizzazione
  - svariate ottimizzazioni con «memory optimized»



Modello EAV

- «Entity Attribute Value»
  - tecnica molto vecchia, funziona con tutti gli RDBMS
  - massima flessibilità, minima ricchezza semantica
  - controllo tipi dato limitato
    - tutte stringhe, un campo per tipo, tipo «SQL Variant» 😐
  - query complesse/poco performanti
- «One True Lookup Table»
  - variante solo per tabelle di «Lookup» (ma cosa sono?)
- Approccio ibrido

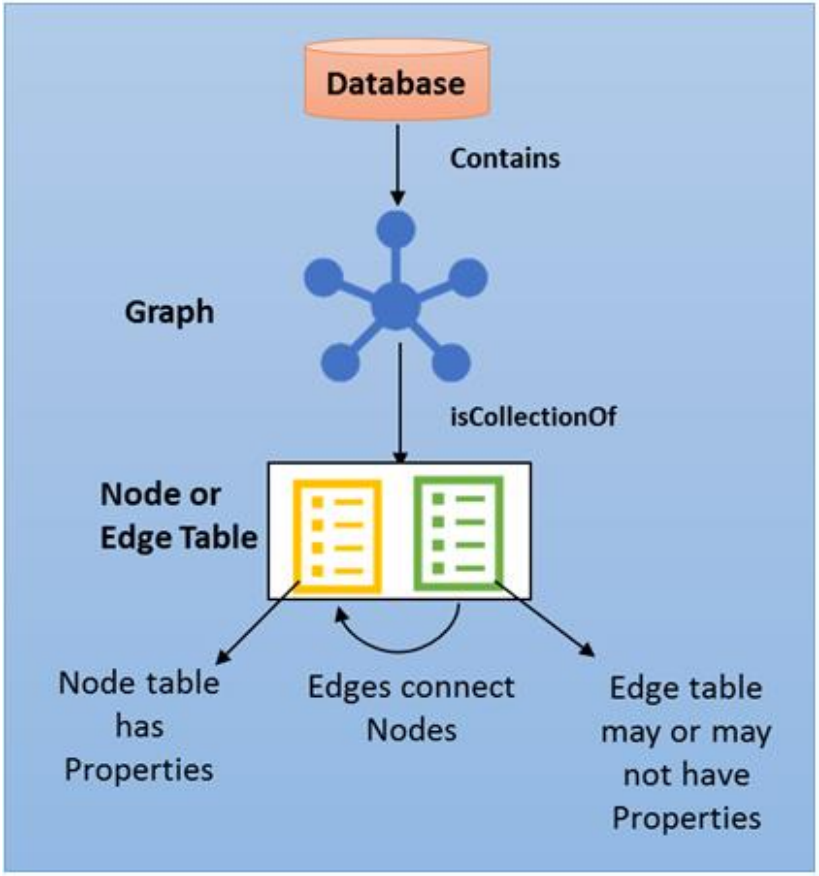


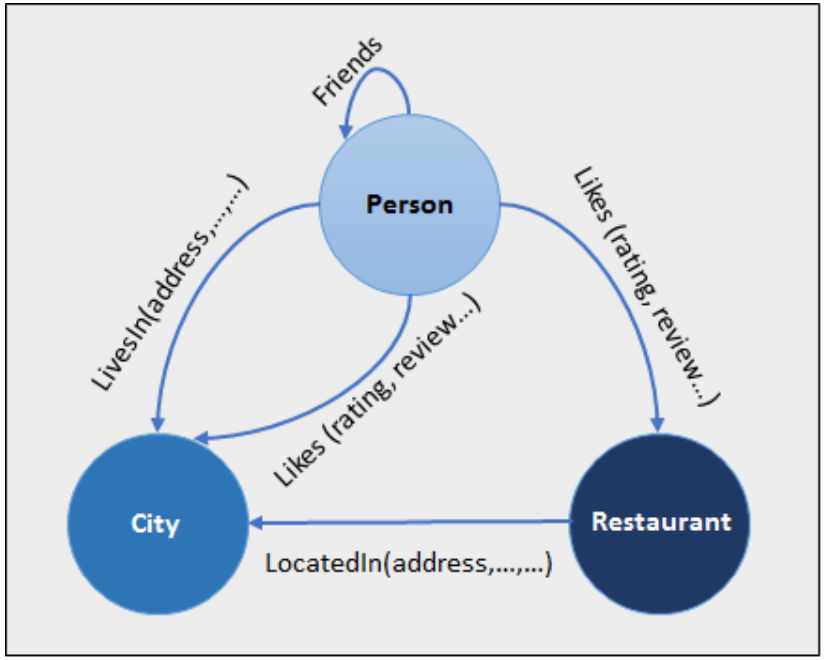
https://en.wikipedia.org/wiki/Entity%E2%80%93attribute%E2%80%93value_model

DEMO

«Sparse Columns»

Graph Processing





Graph Processing Concepts

- Node Table
  - Represent an entity
- Edge Table
  - Represent many-to-many relationship
  - May have properties
  - Is directed, connects two nodes
- MATCH T-SQL Command
  - Search condition for graph objects
  - Pattern matching and traversal

Graph Processing Limitations

- Local/global temporary tables not supported
- Table types/variables not supported
- System-versioned temporal tables not supported
- Memory optimized tables not supported
- Can’t update connected nodes of an edge using UPDATE
  - insert new edge pointing to different nodes
  - delete previous edge
- Cross database queries on graph objects not supported

DEMO

Graph Processing

Transazioni in SQL Server

- Transazioni esplicite
  - comando **BEGIN TRANSACTION**
  - comando **COMMIT/ROLLBACK TRANSACTION**
- Transazioni implicite
  - a livello di singolo comando
  - opzione **SET IMPLICIT_TRANSACTIONS ON**
- Transazioni nidificate
  - Punti di salvataggio
- Controllo immediato
  - rollback dipende dalla criticità dell'errore

Log delle transazioni

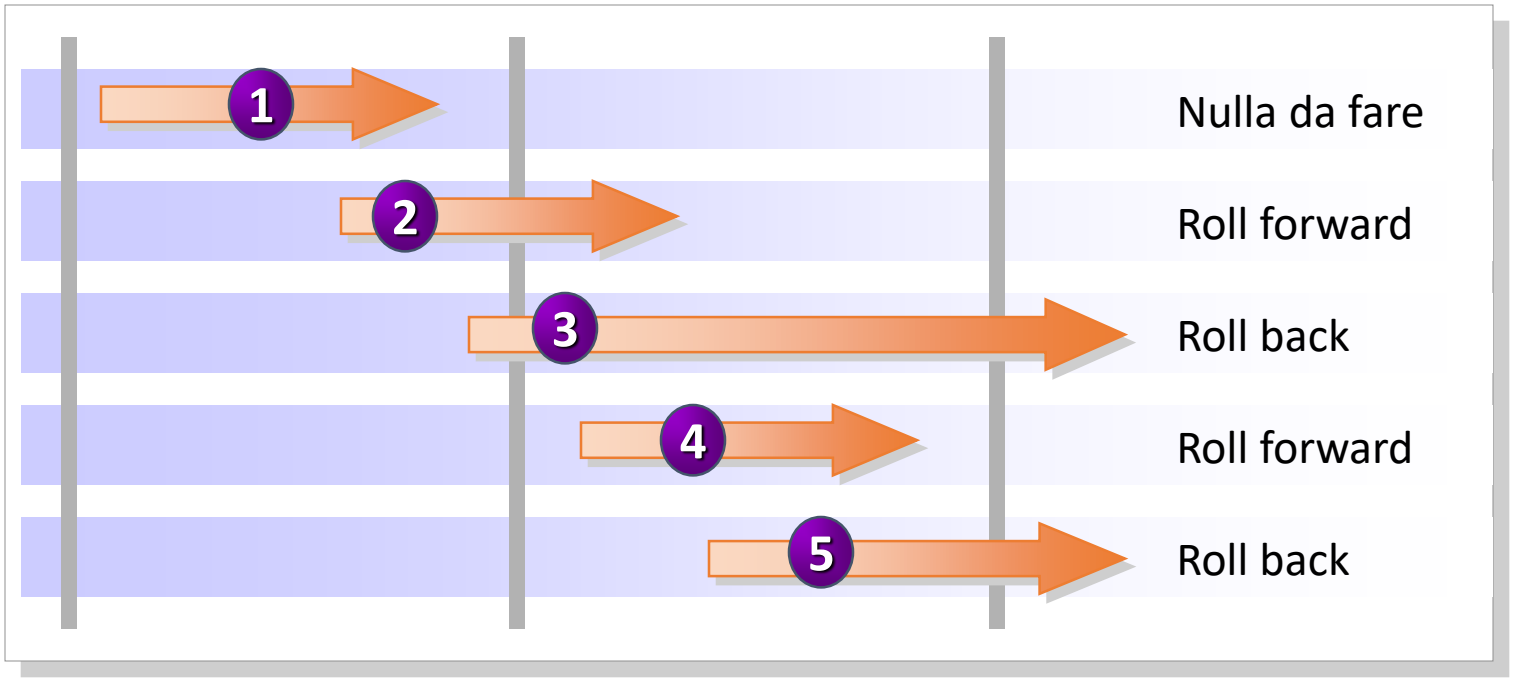
- Garantisce le proprietà di
  - Atomicità
  - Durabilità (persistenza)
- WAL («Write Ahead Logging»)
  - Scrive le modifiche prima nel log e poi in memoria
  - Meccanismi asincroni consolidano modifiche nei file dati
    - «Checkpoint», «Lazy Writer», manutenzione della cache
  - Basato su ARIES
    - «Algorithms for Recovery and Isolation Exploiting Semantics»
    - Comprese ottimizzazioni
      - «No force on commit» (full recovery model)
      - «Force on commit» (bulk logged recovery model)
    - http://www.sai.msu.su/~megera/postgres/gist/papers/concurrency/p94-mohan.pdf

Log transazioni e «Recovery»

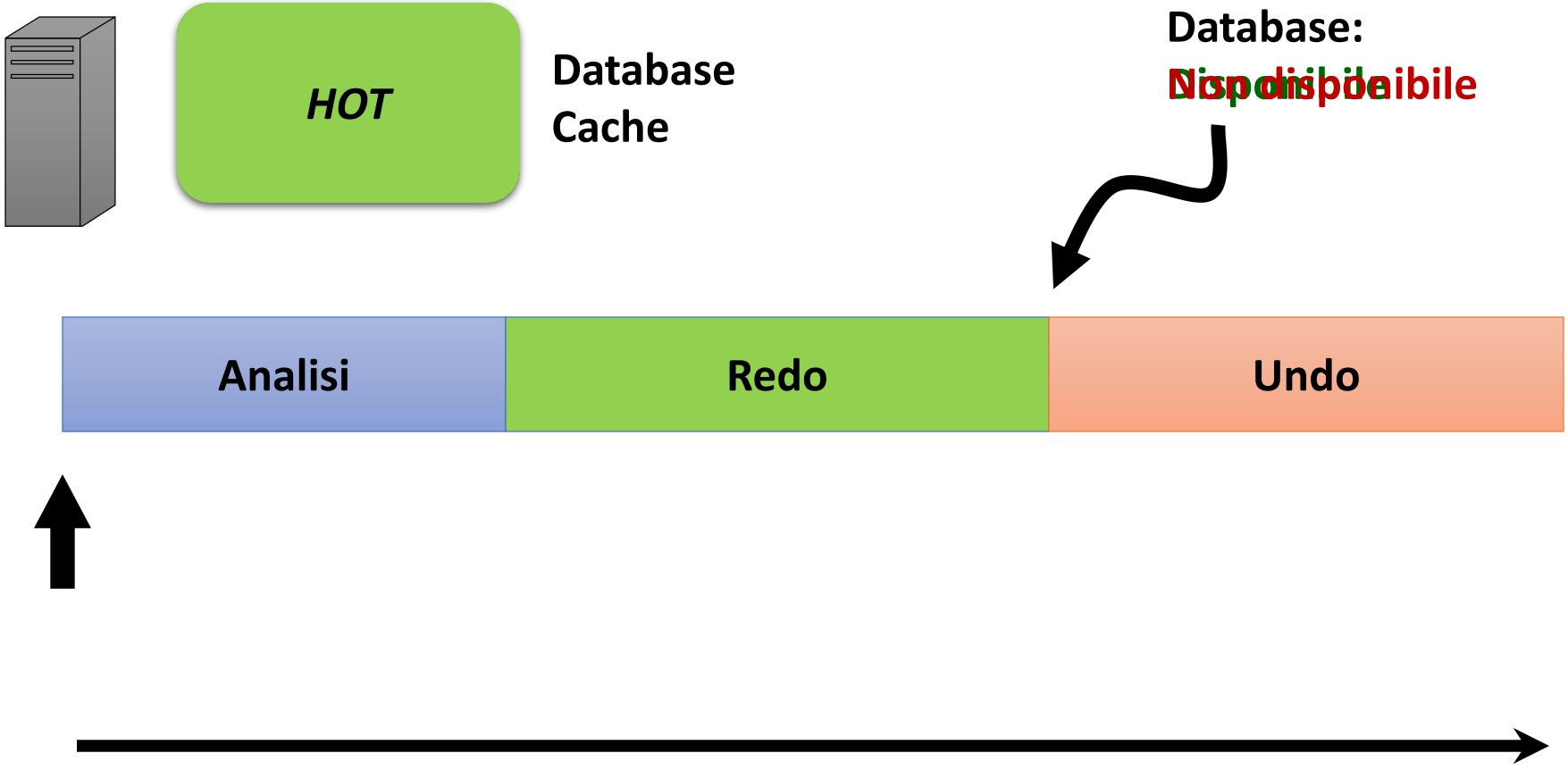
- «Crash Recovery» (ripresa a caldo)
  - alla ripartenza del servizio, per ogni database
    - ripristino transazioni confermate
    - annullamento transazioni non confermate
- «Media Recovery» (ripristino a freddo)
  - al termine del «restore»
    - ripristino anche del log delle transazioni
    - Esecuzione della ripresa a caldo

Ripresa a caldo e «Checkpoint»

Ripristino transazioni



Ripresa a caldo ottimizzata



«Delayed Durability»

- Transazione può essere
  - «Fully Durable»
    - protocollo WAL tradizionale, commit sincrono
  - «Delayed Durable»
    - AKA «lazy commit» o commit asincrono
- «Eventual Durability» ☺
  - record Transaction Log mantenuti nel buffer
  - buffer scritto quando pieno o evento di «Flush»
  - possibile perdita della transazione!
    - in modo consistente
  - Applicazione deve tollerare la perdita di dati!

«Delayed Durability Flush»

- Buffer Transaction Log pieno
- Commit transazione «fully durable»
  - stesso database
- Esecuzione manuale procedura sp_flush_log
- Attenzione! Shutdown SQL Server
  - potrebbe non scrivere il buffer
  - va trattato come un possibile evento di perdita dati

Controllo «Delayed Durability»

- Livello Database

```
ALTER DATABASE … SET DELAYED_DURABILITY = { DISABLED | ALLOWED | FORCED }
```

- Livello «Atomic block» («Native Compilation»)

```
CREATE PROCEDURE <procedureName> …
WITH NATIVE_COMPILATION, SCHEMABINDING, EXECUTE AS OWNER
AS BEGIN ATOMIC WITH
(   DELAYED_DURABILITY = ON,
    TRANSACTION ISOLATION LEVEL = SNAPSHOT,
    LANGUAGE = N'English'
    … )
END
```

- Livello «Commit»

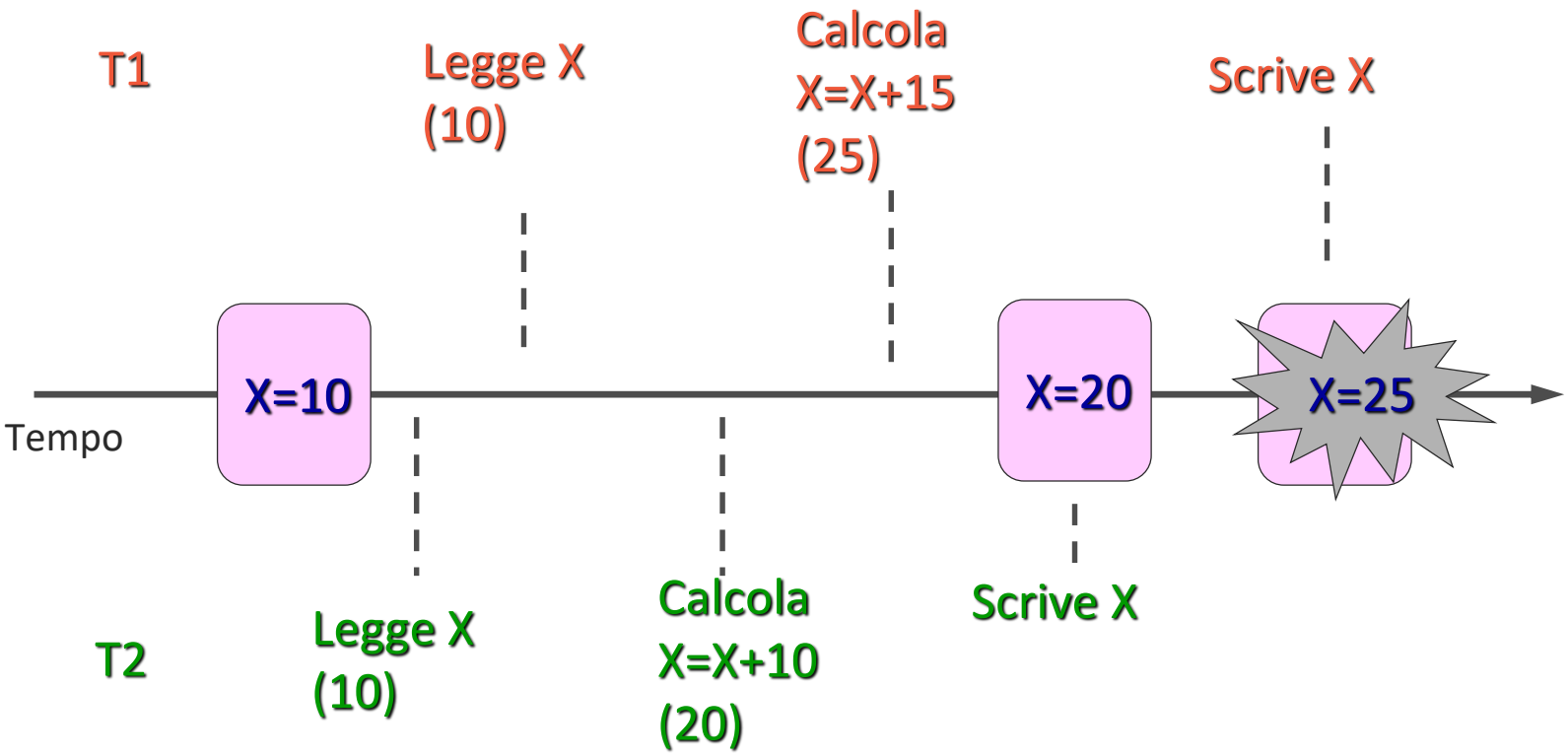
```
COMMIT [ { TRAN | TRANSACTION } ] [ transaction_name | @tran_name_variable ] ]
[ WITH ( DELAYED_DURABILITY = { OFF | ON } ) ]
```

Isolamento delle transazioni

- Idealmente: serializzazione delle transazioni
  - massima consistenza
  - minor concorrenza
- Se le transazioni non sono coordinate
  - perdita di aggiornamenti (lost updates)
- Se transazioni sono parzialmente isolate
  - letture inconsistenti («dirty reads»)
  - letture non ripetibili («non-repeatable reads»)
  - letture fantasma («phantoms»)
  - conflitto in aggiornamento («update conflict»)

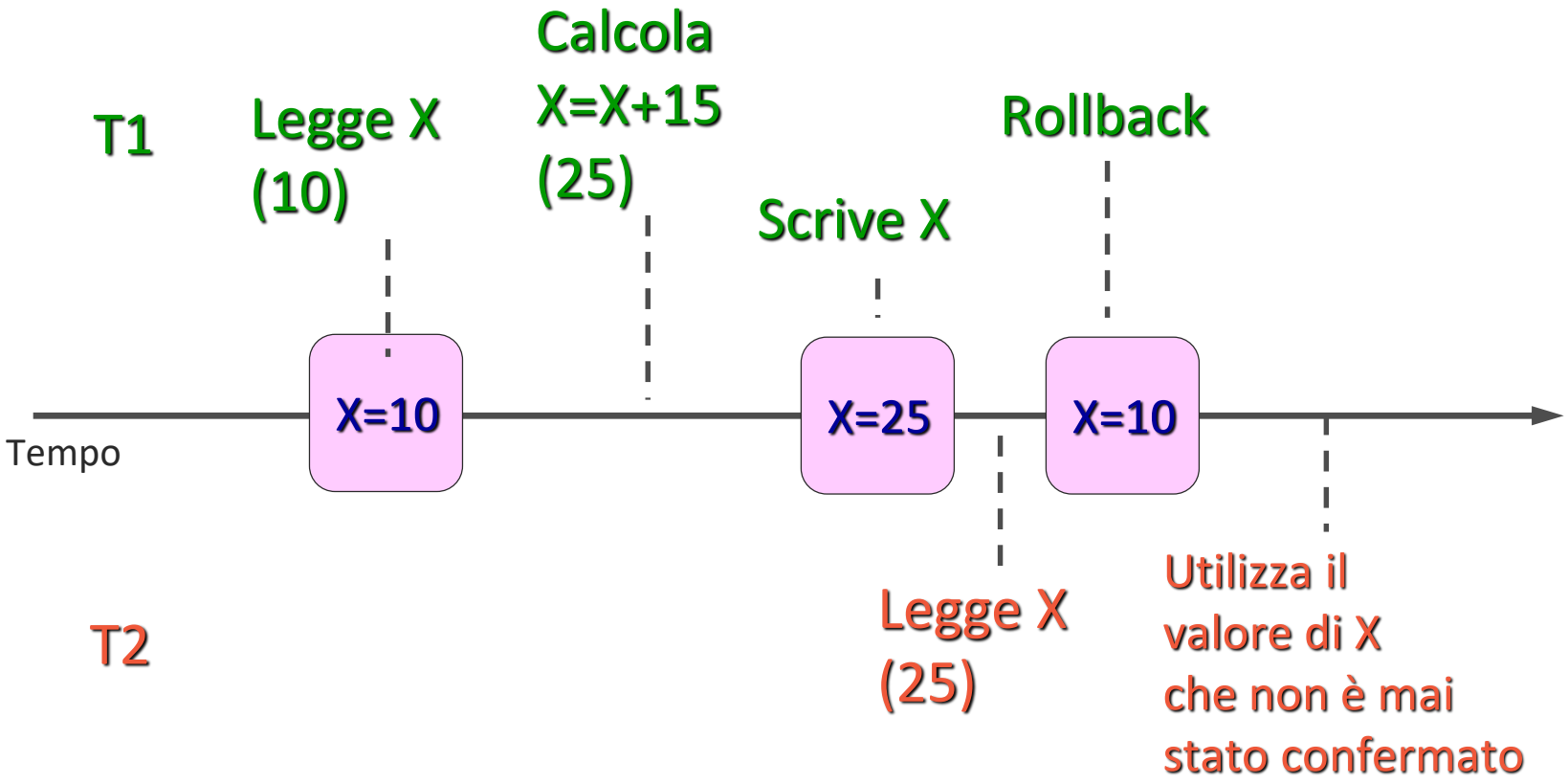
Perdita di aggiornamenti

- Aggiornamento non sincronizzato



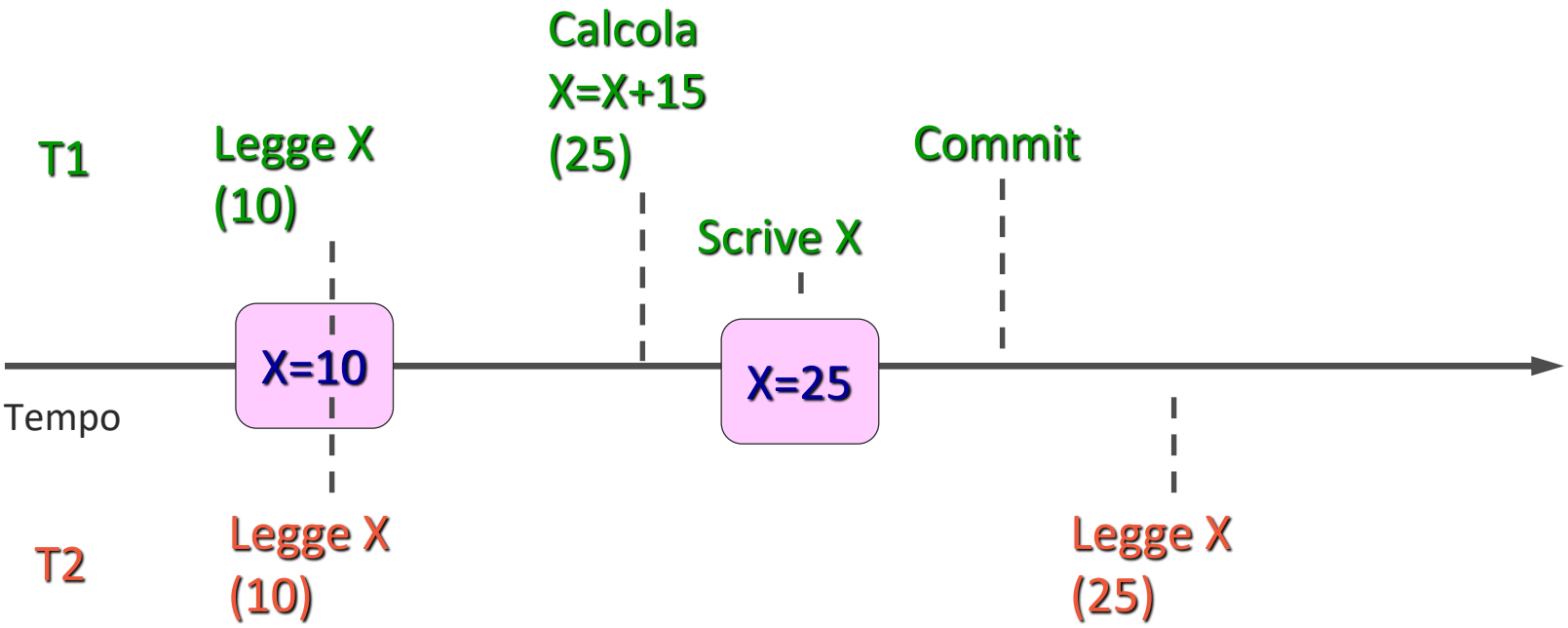
Letture inconsistenti

- T2 vede le modifiche non confermate di T1



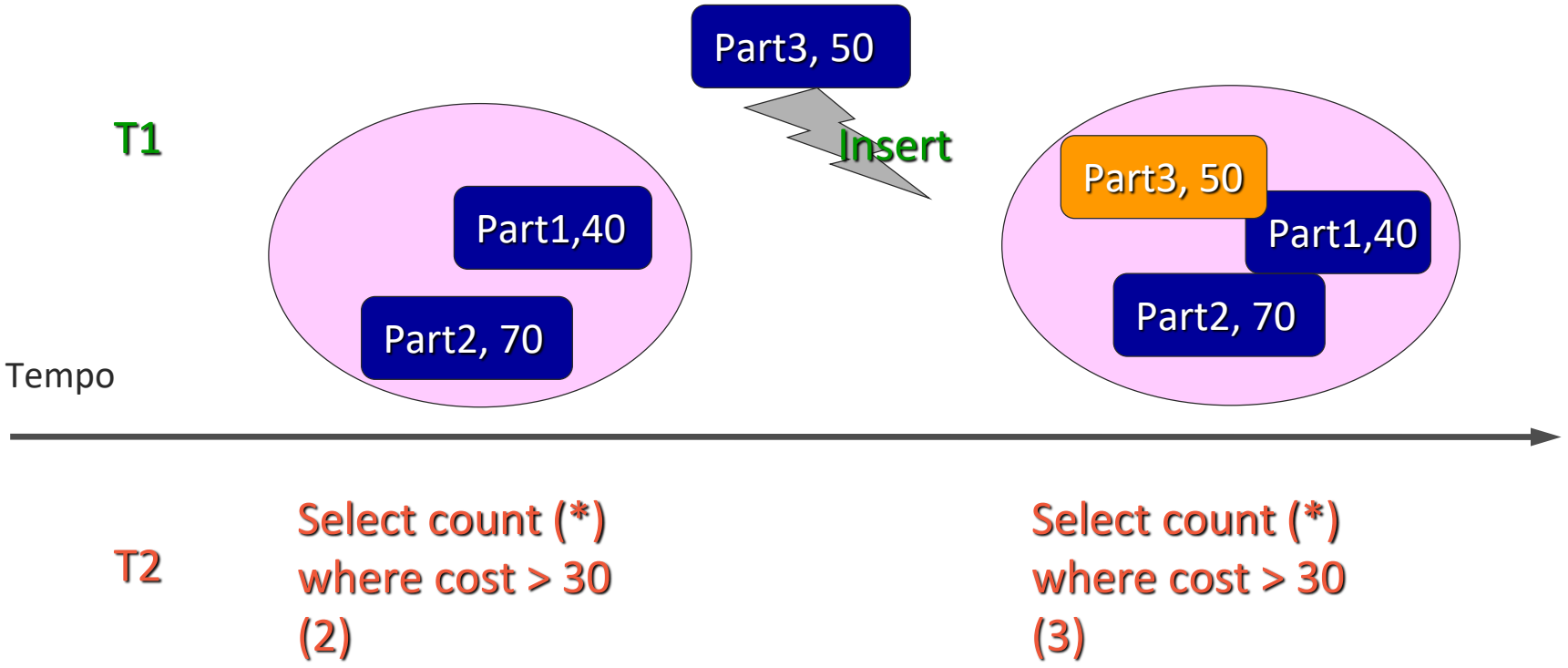
Letture non ripetibili

- Lettura successiva inconsistente



Letture fantasma

- Lettura intervallo successiva inconsistente

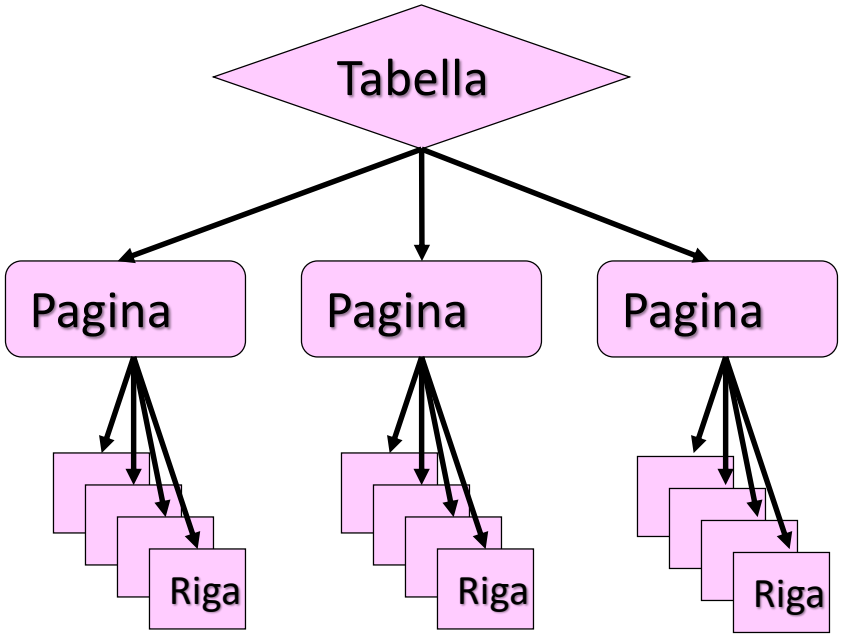


Livelli di isolamento in SQL Server

- 4 livelli supportati per standard ANSI
  - «Read Uncommitted», «Read Committed», «Repeatable Read», «Serializable»
- Predefinito «Read Committed» in due modalità
  - pessimistico, usa protocollo di «Locking»
  - ottimistico, usa «Row Versioning»
- Aggiunta di un ulteriore livello «Snapshot»
  - simile a «Serializable» ma ottimistico in scrittura
  - genera errore alla «Commit» in caso di conflitto

Gestione dei «Lock»

- Tipologie di lock
  - Condivisi (S)
  - Esclusivi (X)
  - Aggiornamento (U)
- Intenzione (I)
- Granularità
  - Riga/chiave
  - Intervallo tra chiavi
  - Pagina
  - Tabella
  - «Extent»
  - Database



Compatibilità «Lock»

- Sottoinsieme della matrice di compatibilità

	Lock già rilasciato							
Lock richiesto	IS	S	U	IX	X	SchS	SchM	BU
IS	Si	Si	Si	Si	No	Si	No	No
S	Si	Si	Si	No	No	Si	No	No
U	Si	Si	No	No	No	Si	No	No
IX	Si	No	No	Si	No	Si	No	No
X	No	No	No	No	No	Si	No	No
SchS	Si	Si	Si	Si	Si	Si	No	Si
SchM	No	No	No	No	No	No	No	No
BU	No	No	No	No	No	Si	No	Si

«Read Committed» (pessimistico)

```
CREATE TABLE t1 ( c1 int unique, c2 int)
INSERT INTO t1 VALUES (1, 5)
```

Transazione 1

```
BEGIN TRAN

UPDATE t1
SET    c2 = 9
WHERE  c1 = 1

COMMIT TRAN
```

Transaction 2
(Read Committed)

```
BEGIN TRAN

SELECT c2 FROM t1
WHERE  c1 = 1

-- SQL Server risponde 9

COMMIT TRAN
```

Bloccato!

Tempo

«Read Committed Snapshot» (ottimistico)

```
CREATE TABLE t1 ( c1 int unique, c2 int)
INSERT INTO t1 VALUES (1, 5)
```

Transazione 1

```
BEGIN TRAN

UPDATE t1
SET    c2 = 9
WHERE  c1 = 1

COMMIT TRAN
```

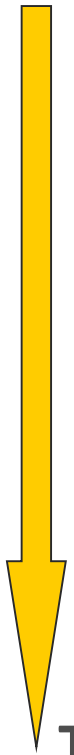
Transaction 2
(Read Committed Snapshot)

```
BEGIN TRAN

SELECT c2 FROM t1
WHERE  c1 = 1

  -- SQL Server risponde 5

SELECT c2 FROM t1
WHERE  c1 = 1
  -- SQL Server risponde 9

COMMIT TRAN
```

Tempo

«Snapshot» (in lettura)

```
CREATE TABLE t1 (c1 int unique, c2 int)
INSERT INTO t1 VALUES (1, 5)
```



Transazione 1

```
BEGIN TRAN
UPDATE t1
SET    c2 = 9
WHERE  c1 = 1


COMMIT TRAN
```

Transazione 2 (Snapshot Isolation)

```
SET TRANSACTION ISOLATION LEVEL
SNAPSHOT

BEGIN TRAN

SELECT c2 FROM t1 WHERE c1 = 1
  -- SQL Server risponde 5

SELECT c2 FROM t1 WHERE c1 = 1
  -- SQL Server risponde 5

COMMIT TRAN

SELECT c2 FROM t1 WHERE c1 = 1
  -- SQL Server risponde 9
```

«Snapshot» (in aggiornamento)

```
CREATE TABLE t1 ( c1 int unique, c2 int)
INSERT INTO t1 VALUES (1,5)
```

Transazione 2

```
BEGIN TRAN

UPDATE t1
SET    c2 = 9
WHERE  c1 = 1

COMMIT TRAN
```

Tempo

transazione 1 (Snapshot Isolation)

```
SET TRANSACTION ISOLATION LEVEL
SNAPSHOT

BEGIN TRAN

SELECT c2 FROM t1 WHERE c1 = 1

  -- SQL Server risponde 5

UPDATE t1
SET    c2 = 15
WHERE  c1 = 1
```

Bloccato!

Rollback perché conflitto tra aggiornamenti

Livelli, fenomeni, concorrenza

	Fenomeni permessi				
Livelli di isolamento	**Dirty Read**	**Non-Repeatable Read**	**Phantoms**	**Update Conflict**	**Modello concorrenza**
Read Uncommitted	Si	Si	Si	No	
Read Committed 1 Locking *2 Snapshot*	No No	Si Si	Si Si	No No	**Pessimistico** **Ottimistico**
Repeatable Read	No	No	Si	No	**Pessimistico**
Snapshot	No	No	No	Si	**Ottimistico**
Serializable	No	No	No	No	**Pessimistico**

Scelta livello di isolamento

- Mediazione
  - livello alto: più consistenza, meno prestazioni
  - livello basso: meno consistenza, più prestazioni
- Modello pessimistico o ottimistico?
  - dipende… quasi sempre va bene ottimistico
  - è bene censire gli altri casi
- Scelta in base alle necessità applicative!
  - non alle prestazioni!

Altri problemi «Read Committed»

- «Shared Lock» rilasciati durante «scan»
- Righe possono muoversi fisicamente
  - Es. aggiornamento indice «Clustered»
  - Fenomeni problematici
    - Rilettura stessa riga
    - Mancata lettura di una riga

«Deadlock» (blocco critico)

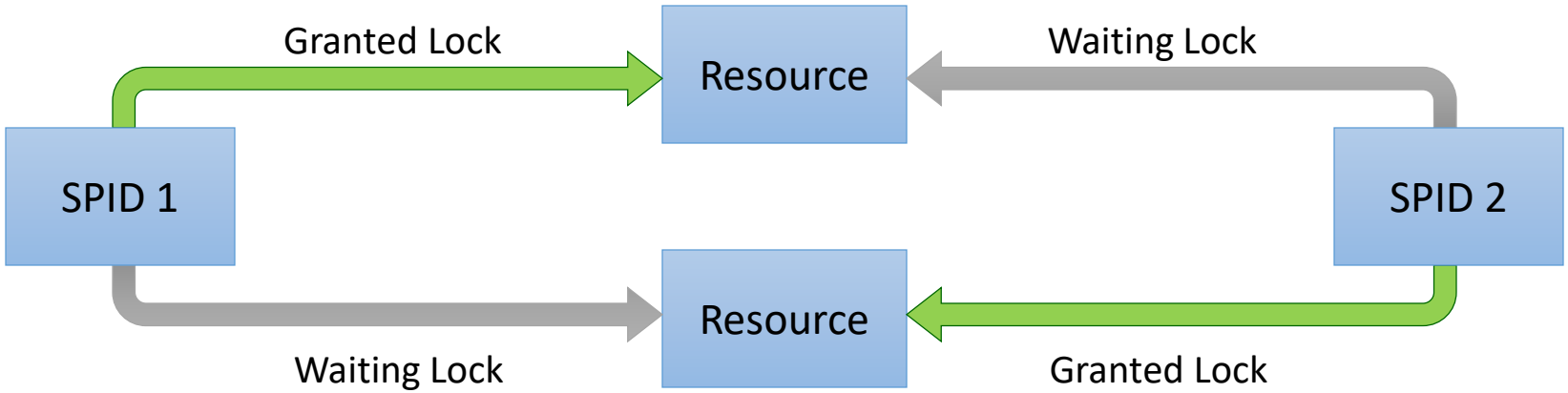
- Evento normale
  - tecniche per mitigarne la frequenza
  - non sempre si possono eliminare completamente
- Risoluzione automatica
  - controllo cicli ad intervalli regolari
  - transazione meno costosa per annullamento viene scelta come vittima («rollback»)
  - controllo più frequente quando si verificano «Deadlock»
- Applicazioni
  - ricevono errore 1205
  - devono gestire il problema, es. riprovando

Minimizzare i «Deadlock»

- Usare livello di isolamento appropriato
- Verificare possibilità di usare livelli «Snapshot»
- Controllare gli indici
- Controllare normalizzazione della base dati
- Le transazioni dovrebbero
  - durare poco
  - non richiedere interazione con utente
  - accedere alle risorse nello stesso ordine

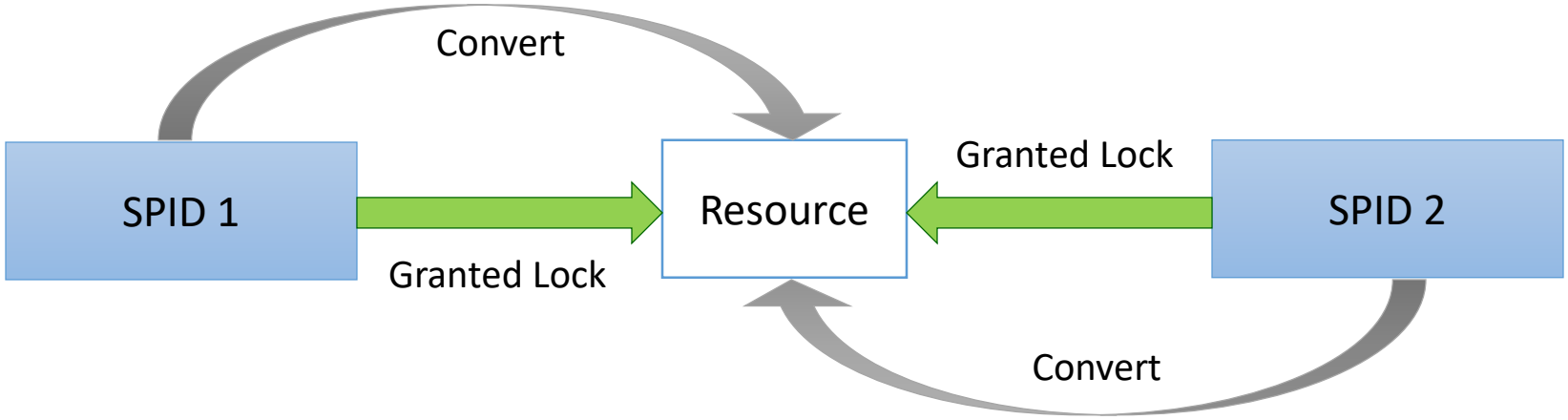
«Cycle Deadlock»

- Coinvolge più risorse
- Processi si bloccano a vicenda in modo ciclico

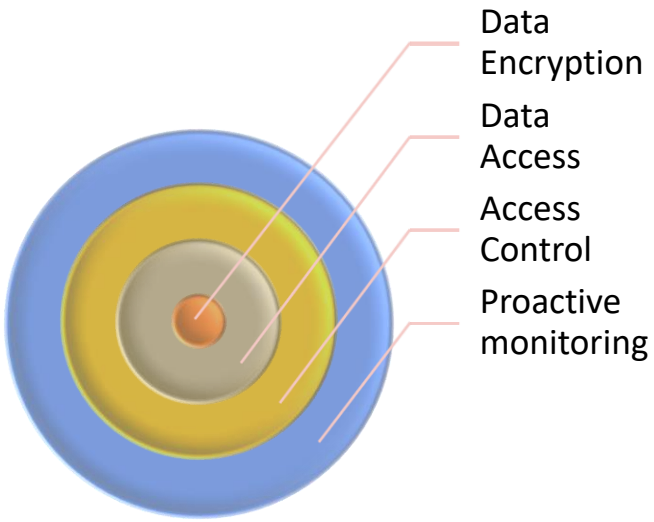


«Conversion Deadlock»

- Coinvolge una risorsa
- Entrambi i processi hanno un «Lock» sulla risorsa
  - tipicamente «Shared»
  - con livello di isolamento più alto di «Read Committed»
- Entrambi cercano di convertire il «Lock»
  - con uno incompatibile



«Security Layering»



Data Encryption

- Transport Layer Security (in transit)
- Transparent Data Encryption (at rest)
- Cell-Level Encryption (at rest)
- Always Encrypted (at rest and in transit)

Data Access

- Dynamic Data Masking
- Row-Level Security

Access Control

- Encrypted Authentication
- SQL Firewall*

Proactive monitoring

- Auditing
- Threat Detection*

Autenticazione

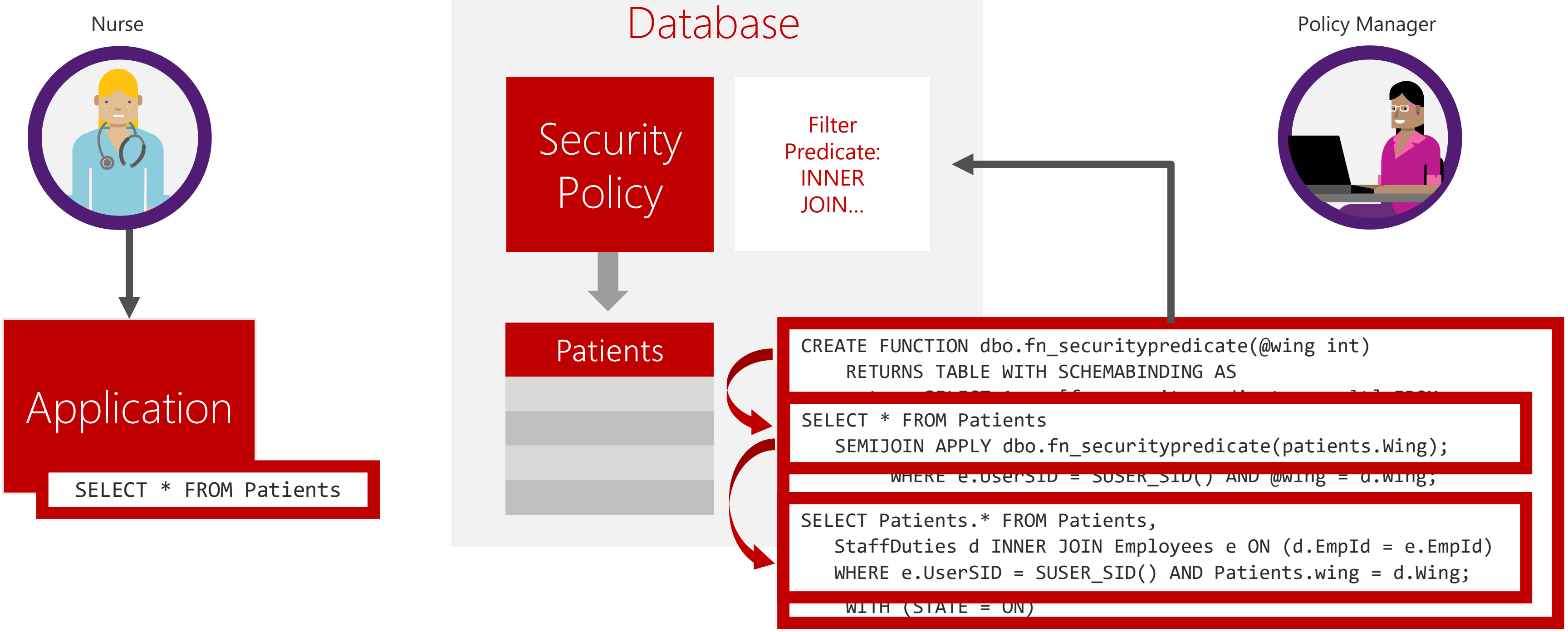
- Due livelli: Server login e Database User
- Nativa SQL Server
- Integrata con Active Directory
  - Utilizza protocollo Kerberos
  - Supporto Azure Active Directory
  - Supporto Active Directory Universal Authentication
    - Multi-factor Authentication (es. via telefono)
    - Supportata solo con SSMS a partire dalla versione 17

Autorizzazioni

- Permessi granulari
  - **GRANT** cosa può fare **ON** su che oggetto **TO** chi
  - Ereditarietà gerarchia di oggetti
  - Assegnazione (GRANT), negazione (DENY), revocare (REVOKE)
  - Row Level Security
- Ruoli a livello di server e database
  - Es. sysadmin, dbcreator, db_datareader, db_datawriter
  - Specifici per alcune funzionalità (database msdb)
- Principio guida: minor privilegio!
  - Possibile elevare temporaneamente
    - Impersonation
    - Stored Procedure (in generale moduli T-SQL firmati)

«Row-Level Security»

[illegible] utente e crea «security policy» che vincola tale predicato alla tabella dei pazienti.



«SQL Server Audit»

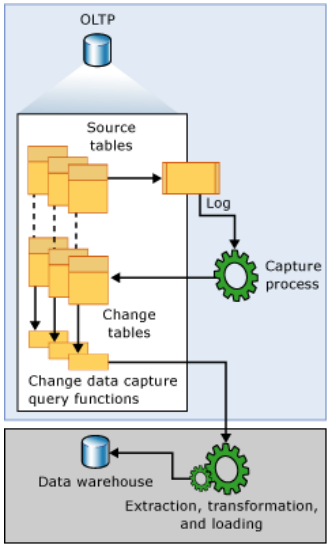
- Definizione gruppi di azioni da tracciare
  - Server
  - Database
- Target
  - File
  - Windows Log
- Disponibile con tutte le edizioni
  - a partire da SQL Server 2016 Service Pack 1
  - Implementazione diversa per Azure SQL Database

«Policy Based Management»

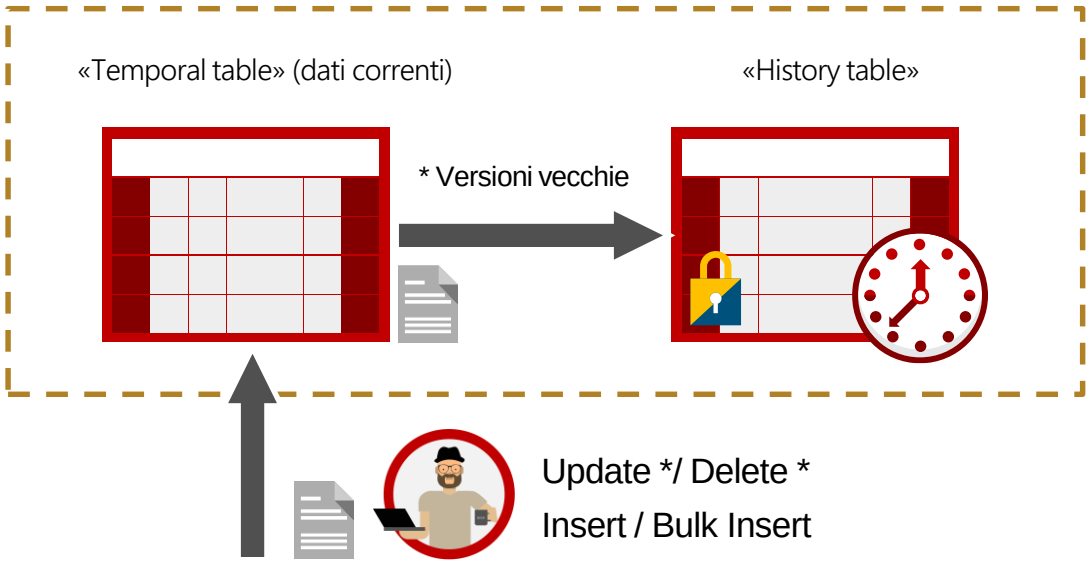
- Infrastruttura per
  - Definizione «policy»
  - Controllo conformità installazione con «policy»
- Serie di «policy» conformi a Best Practice
  - Categoria di policy per sicurezza!
- Enterprise Policy Management Framework
  - http://aka.ms/epmframework

Change Data Capture

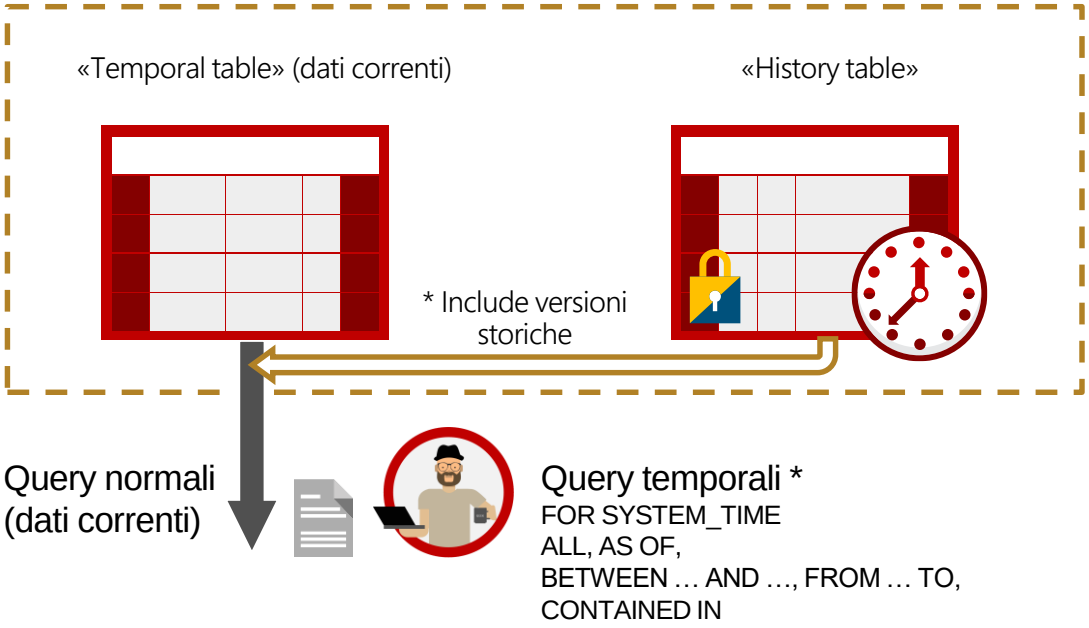
- Soluzione tradizionale Audit modifiche dati?
  - Trigger! Ma…
  - …complessità, prestazioni… ☹
- CDC cattura automaticamente modifiche
  - Pensato principalmente per caricare DWH
  - Genericamente va bene per tenere storico modifiche



Modifiche Database Temporali



Interrogazioni Database Temporali



Introduzione alta disponibilità

- Alta disponibilità è una combinazione di
  - architettura (design)
  - persone
  - processi
  - tecnologia
- Alta disponibilità non è
  - solamente una soluzione tecnologica
  - sinonimo di scalabilità

Significato di indisponibilità

- Esempi di indisponibilità completa
  - manutenzione del server
  - problemi non programmati
    - attacchi DOS, malfunzionamenti hardware
  - tempo per implementare una tecnologia
  - «restore» di uno i più database
- Esempi di indisponibilità percepita
  - lentezza degli applicativi
  - DBMS disponibile, mid-tier non disponibile
  - problemi di rete
  - rilascio nuove versioni degli applicativi

Valutazione requisiti HA

- Rilevazione guasti automatica o manuale
- Failover automatico o manuale
- Tempo di «failover» e di «failback»
- Numero guasti a cui può sopravvivere il sistema
- Granularità
  - istanza, database, tabella, riga

Valutazione impatti HA

- Perdita di dati
  - tutti dicono perdita zero, ma in caso di disastro reale solitamente ci sono dei margini
- Consistenza dei dati
- Complessità
  - implementazione
  - gestione
- Costo di sistemi ridondati
  - hardware, gestione
- Impatto sugli applicativi

Opzioni di «Standby»

- «Hot standby»
  - nodo secondario mantiene una copia dei dati
  - dati consistenti (transazioni sincrone)
  - rilevazione guasti e «failover» automatici
- «Warm standby»
  - nodo secondario mantiene una copia dei dati
  - dati non consistenti (transazioni asincrone)
  - rilevazione guasti e «failover» automatici o manuali
- «Cold standby»
  - nodo secondario configurato ma nessuna copia
  - rilevazione guasti e «failover» manuali

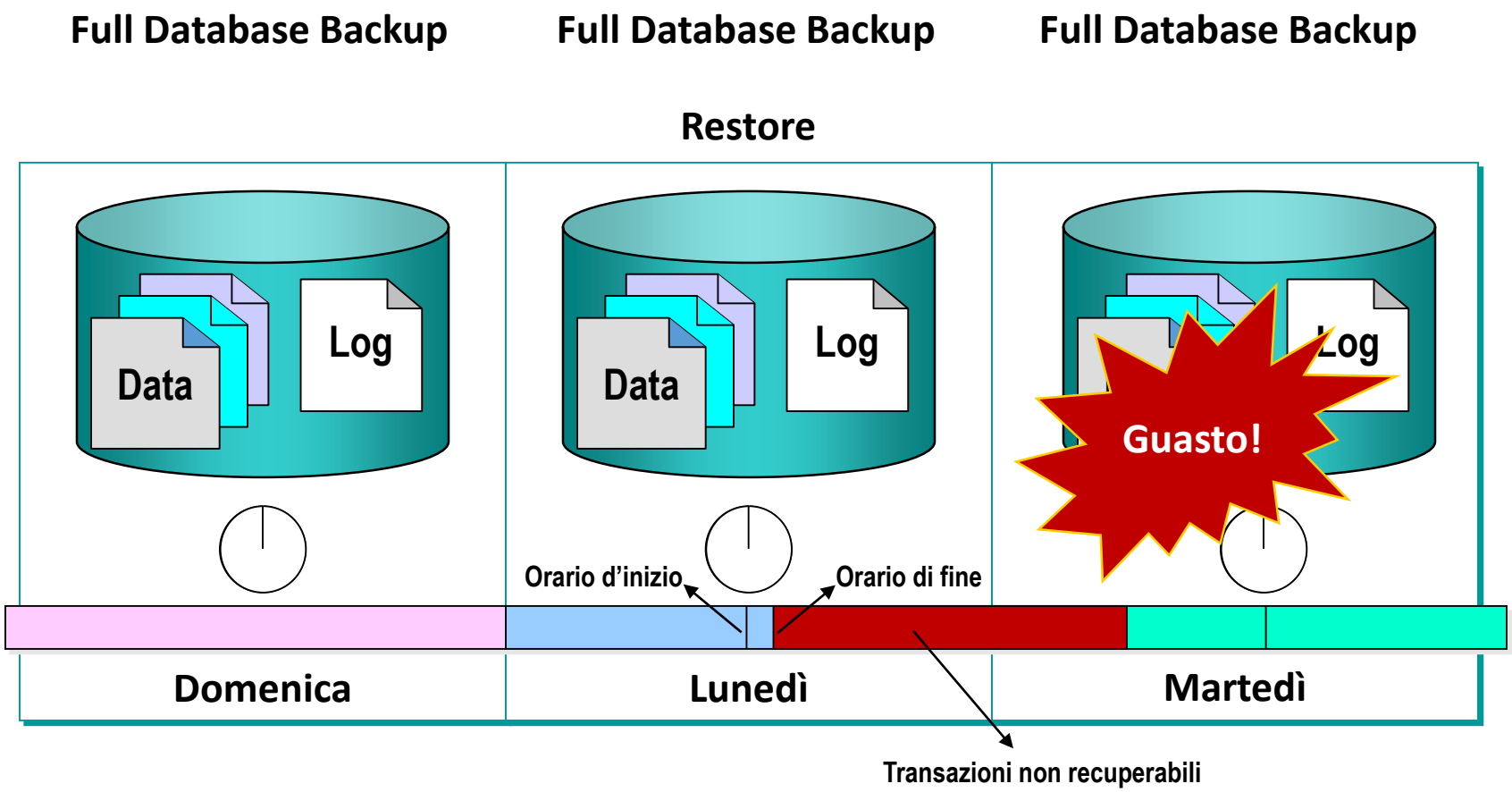
«Backup & Restore»

- Diverse tipologie di Backup
  - Completo
  - Differenziale
  - Del log delle transazioni (incrementale)
- Granularità
  - Database
  - «Filegroup»
  - «Data File»
- Ripristino
  - Completo
  - «Point in time»

«Full Backup»

- Salvataggio completo di tutti i dati
  - tutte le pagine di tutti i file
    - catalogo di sistema (tabelle primary filegroup)
  - include il log delle transazioni
    - backup consistente con l'orario in cui termina
  - baseline di partenza per la ripresa a freddo (restore)

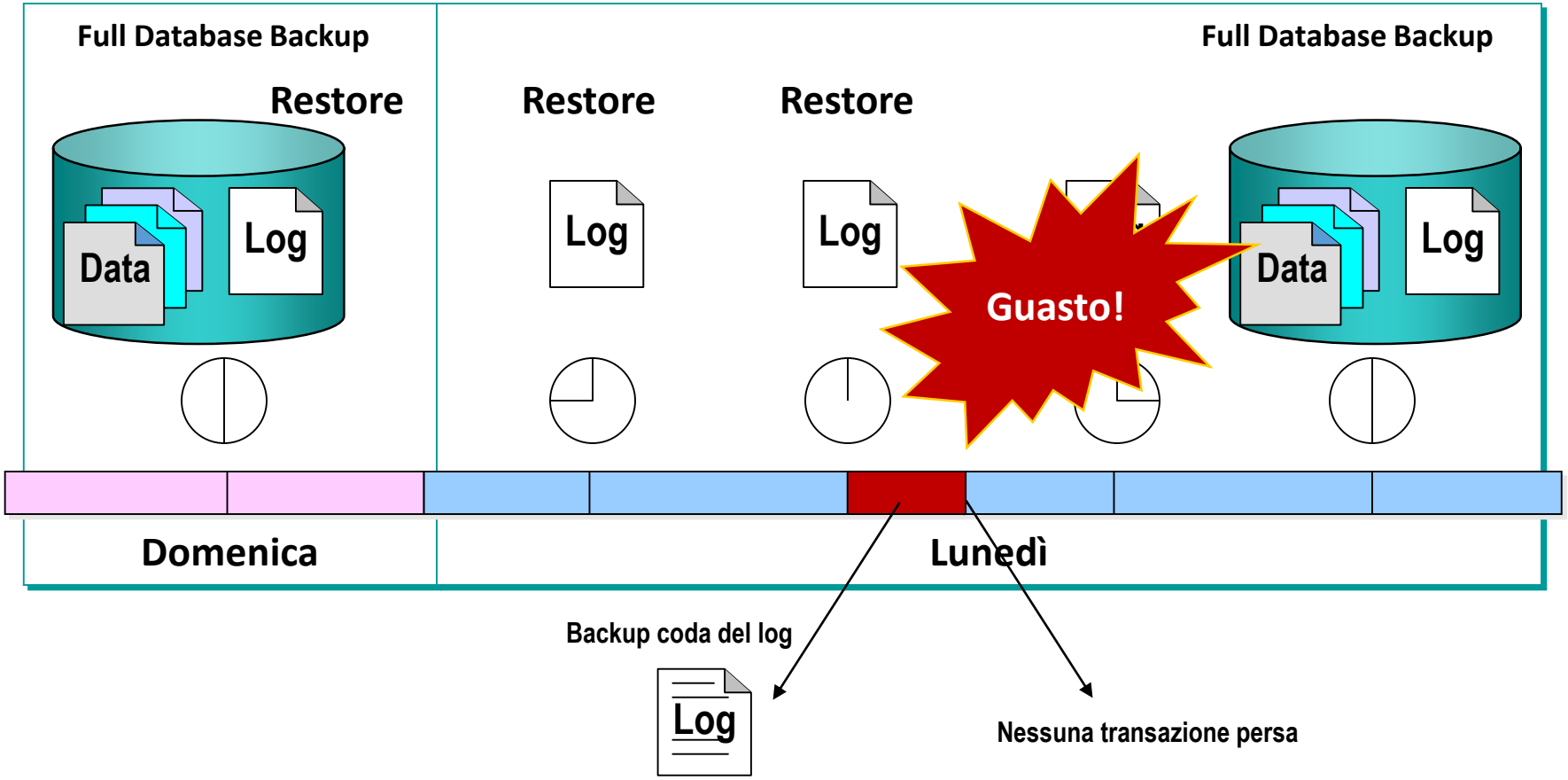
Strategia «Full Backup»



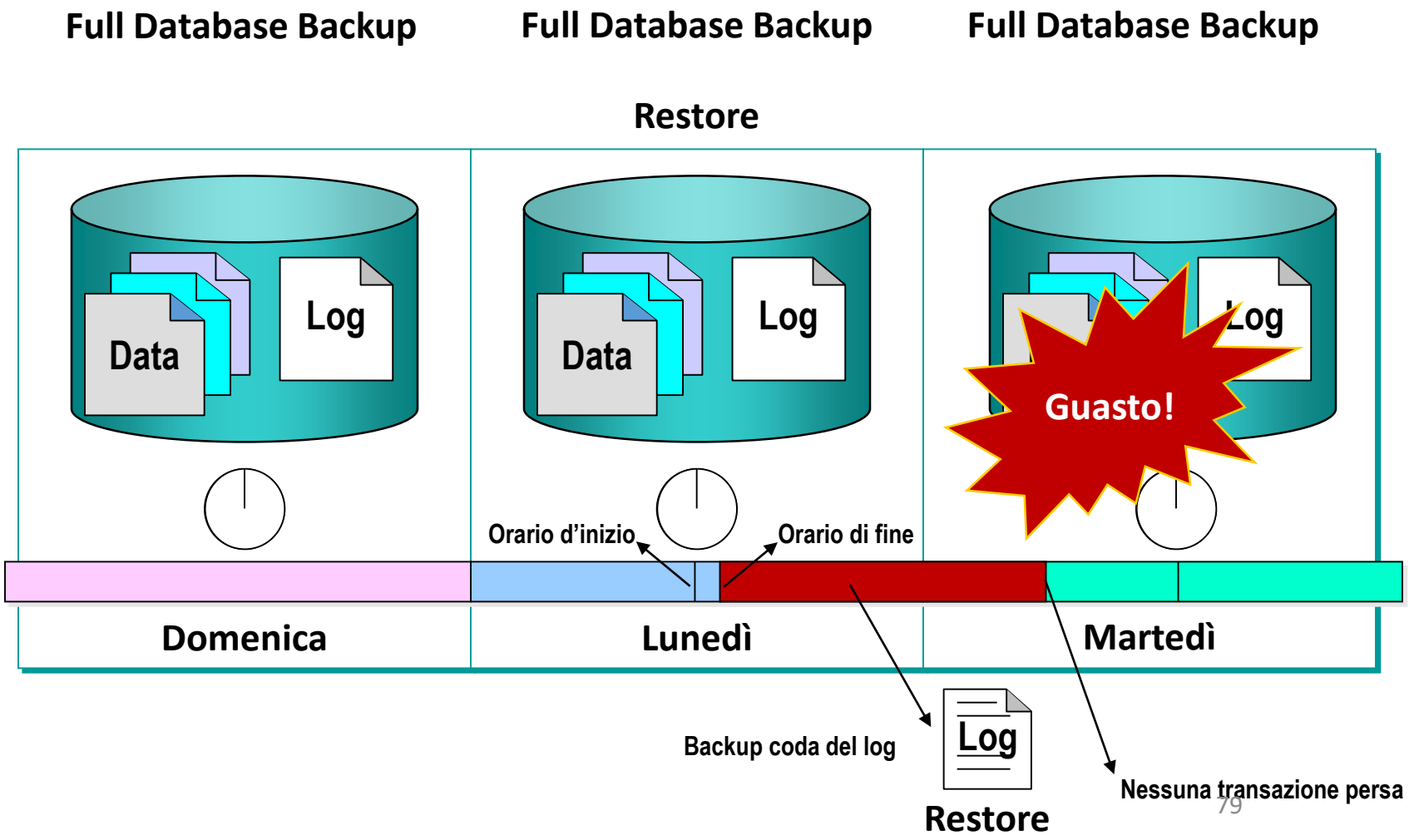
«Log Backup»

- Salvataggio log delle transazioni
  - contiene tutte le transazioni dall'ultimo backup del log
  - backup del log costituiscono una catena di backup
    - applicare in fase di ripristino per riportare a uno stato consistente
  - catena si estende da
    - backup completo
    - backup parziale
    - set completo backup di file
  - svuota il log fino alla porzione attiva
    - fino al virtual log che contiene la transazione attiva più vecchia
- Supportato per i recovery model full o bulk-logged
    - in modalità bulk, backup può essere più voluminoso

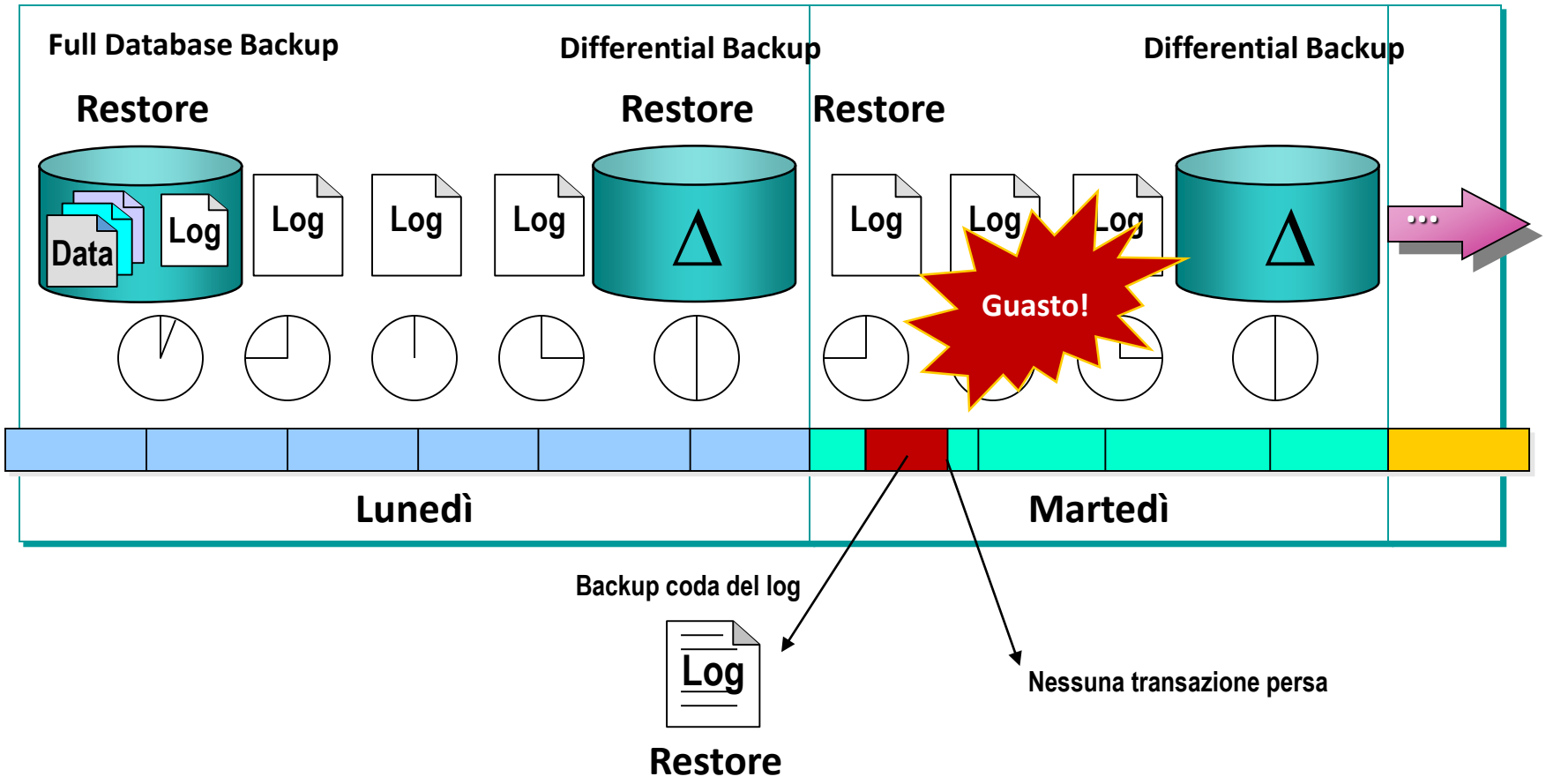
Strategia «Log Backup»



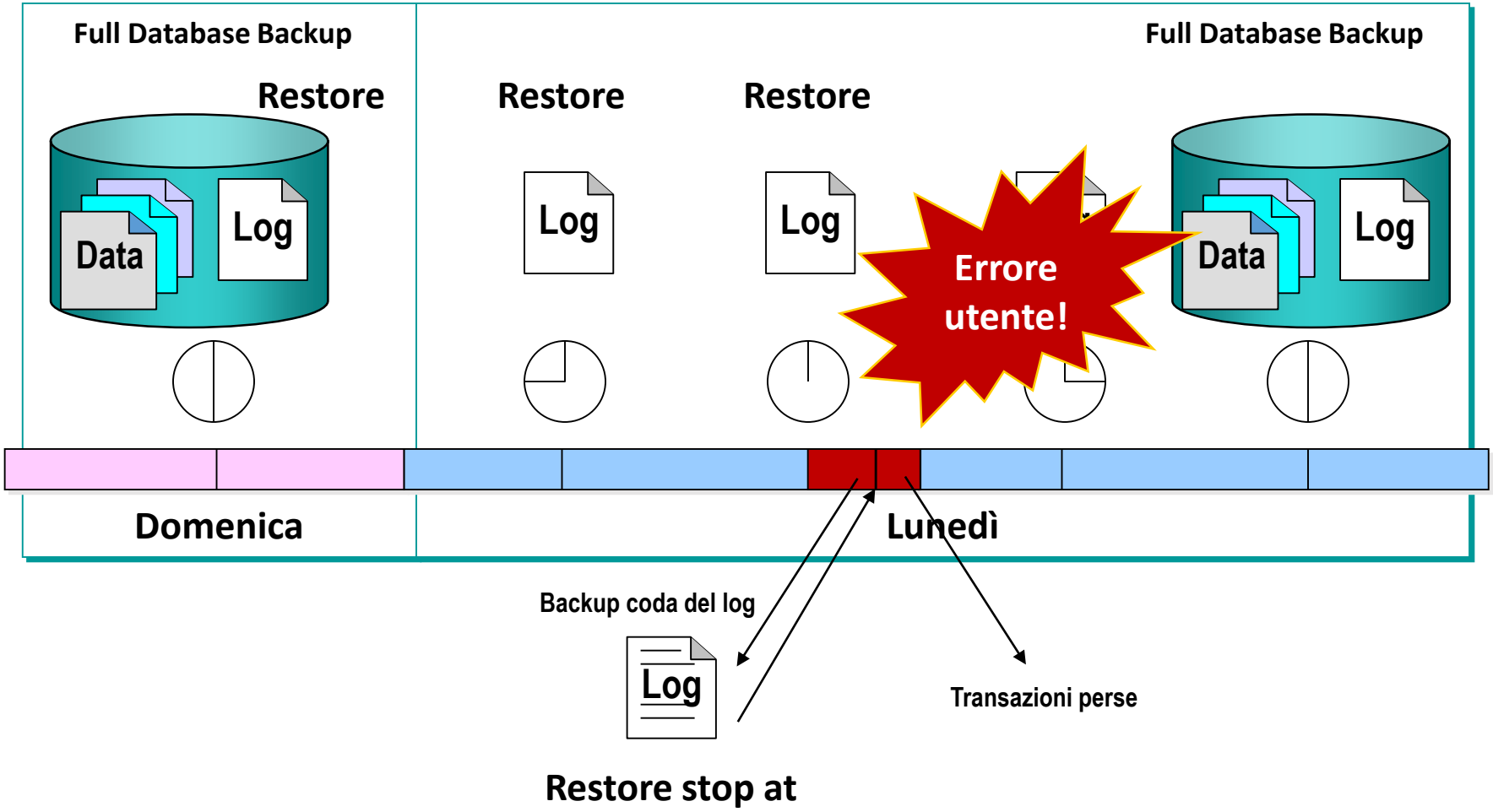
«Full Backup» e «Recovery Model» non Simple



Strategia «Full + Differential + Log»



«Restore to a point»



Tecnologie di protezione

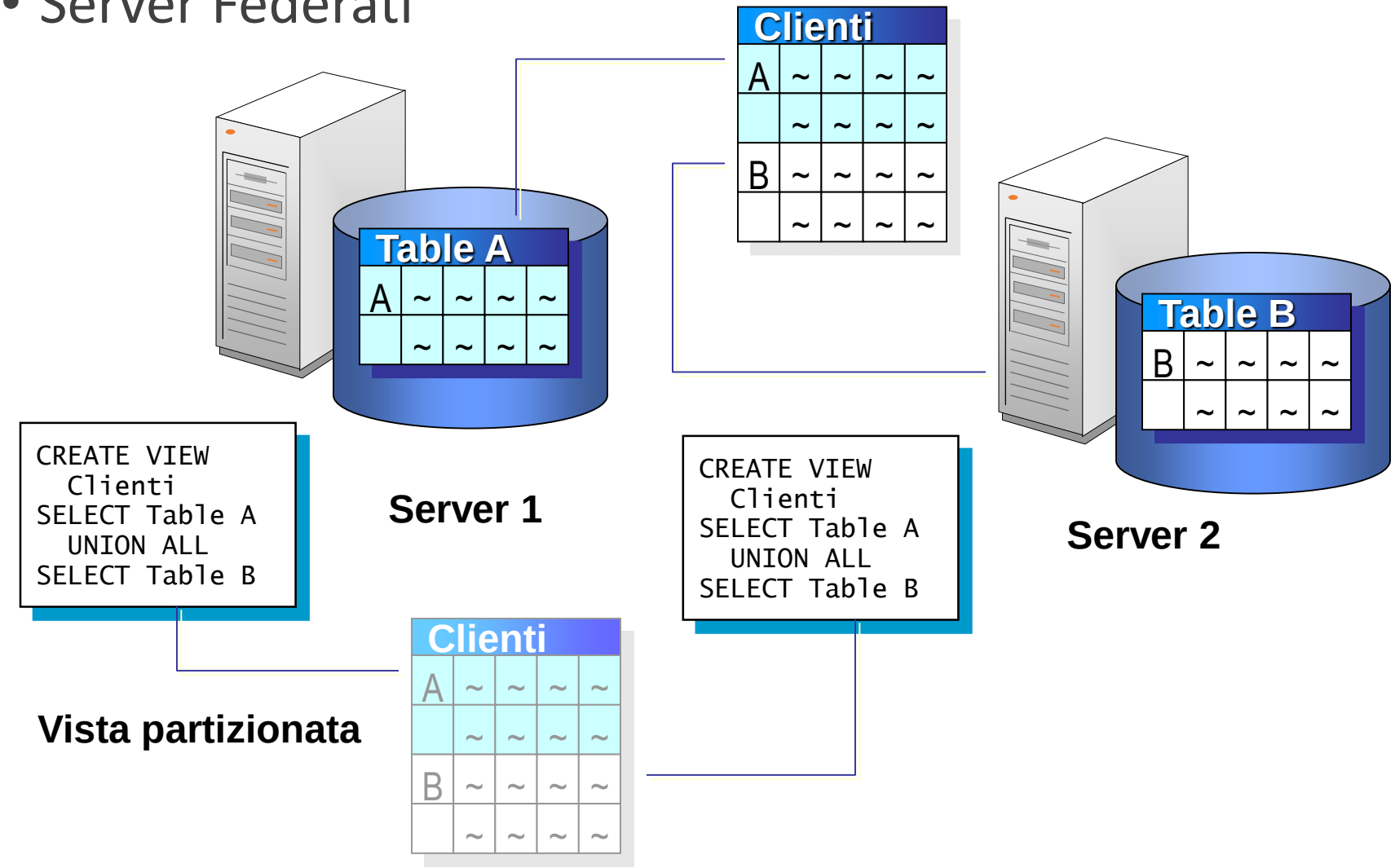
- AlwaysOn
  - FCI («Failover Cluster Instances»)
  - AG («Availability Groups»)
- Log Shipping
- Database Mirroring (deprecato -> AlwaysOn AG)
- Database Snapshot
- Database Replication

Controllo delle prestazioni

- «Resource Governor»
- «Query Store»
- Telemetria
  - DMV
  - Extended Events
  - Performance Counters

«Shared Nothing Clustering»

- Server Federati



«Sharding»

- Molteplici database condivisi da più «tenant»?
- Tecnica «Scale out» distribuzione dati
  - Strutturati in maniera identica
  - In più database indipendenti
  - In base a «Sharding Key»
  - Mappature per intervallo di valori o lista

«Elastic Database client library»

- «Shard Map Management»
  - Mappatura «Shard Keys» e database
  - «Shard Keys» liste o intervalli di valori
- «Data Dependent Routing»
  - Supporto apertura connessione in base a «Shard Key»
- «Multi-Shard Queries»
  - Supporto Query che coinvolge più «Shard»
  - Fusione unico «Result Set» con Semantica UNION ALL

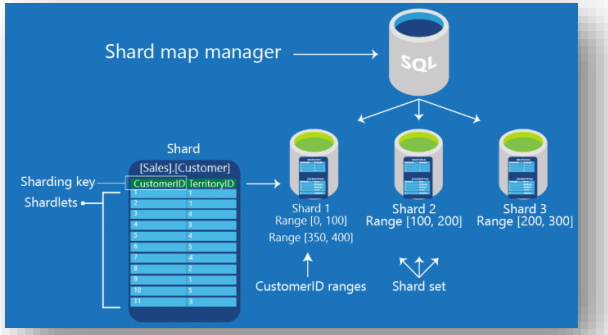


Image source: https://docs.microsoft.com/en-us/azure/sql-database/sql-database-elastic-scale-shard-map-management

«Elastic Database split-merge tool»

- «Split Range» distribuisce una «Shard» su due «Shard»
- «Merge Range» accorpa due «Shard» in una «Shard»
- «Move Shardlet»

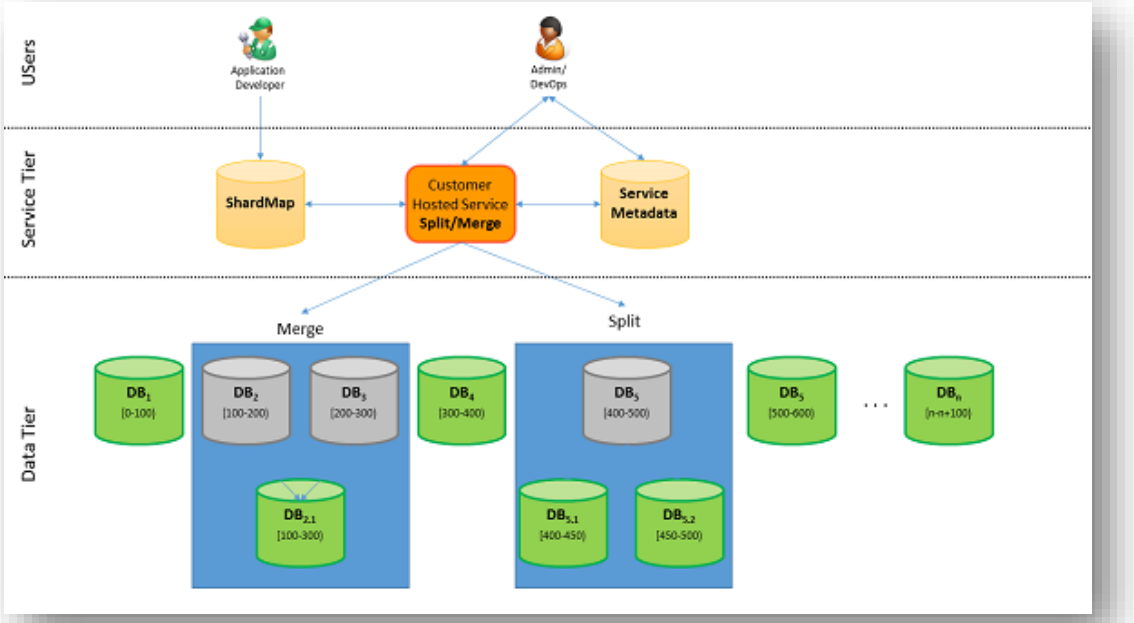


Image source: https://azure.microsoft.com/en-us/documentation/articles/sql-database-elastic-scale-overview-split-and-merge

«Elastic Database Jobs»

- Esecuzione Script T-SQL o applicazione DACPAC
  - Collezione database personalizzata
  - Tutti i database di «Elastic Database Pool»
  - «Shard Set»
- Ritenta in caso di fallimento
- Schedulazione

Scenari «Elastic Database Jobs»

- Attività amministrative
  - es. «deploy» nuovo schema
- Ricostruzione degli indici
- Aggiornamento di dati in comune a più database
- Raccolta dati di telemetria in tabella comune

«Elastic Database Queries»

- Supporto T-SQL per query distribuite
  - CREATE EXTERNAL DATA SOURCE
  - CREATE EXTERNAL TABLE
- Partizionamento verticale «cross-database queries»
  - TYPE = RDBMS
- Partizionamento orizzontale «Sharding»
  - TYPE = SHARD_MAP_MANAGER

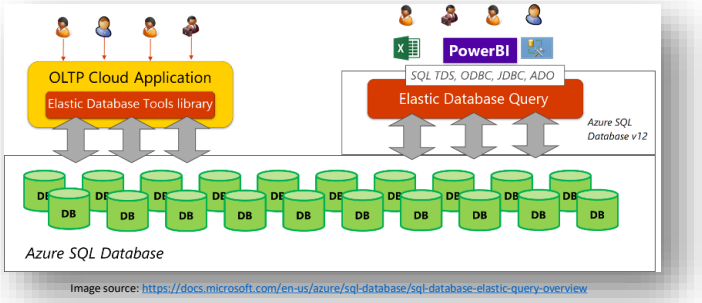


Image source: https://docs.microsoft.com/en-us/azure/sql-database/sql-database-elastic-query-overview

«Eventual Consistency»

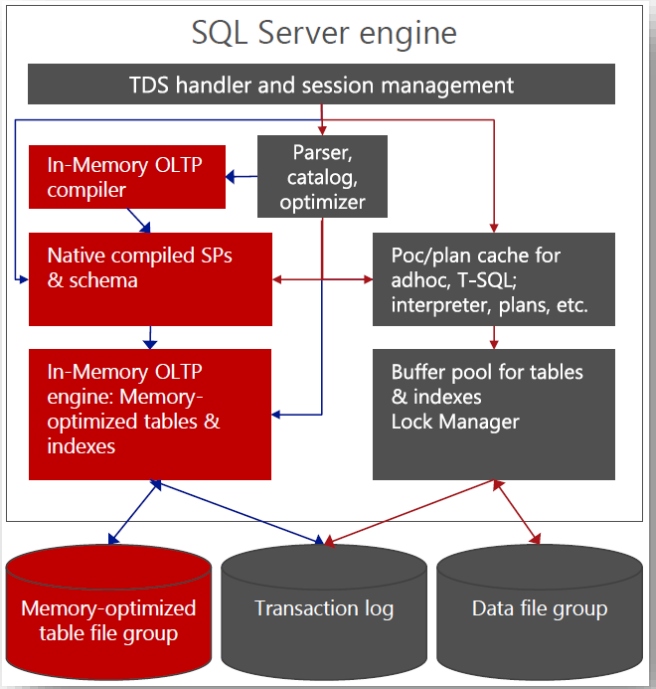
- Caratteristiche dei «Data Store» distribuiti
- Teorema CAP
  - In caso di «Partitioning»…
  - Compromesso tra «Consistency» e «Availability»
  - Spesso sacrificata la prima
- Estensione PACELC
  - Anche senza «Partitioning», «Availability» implica replica
  - Compromesso tra «Consistency» e «Latency»
  - Anche in questo caso spesso sacrificata la prima

DDBS	P+A	P+C	E+L	E+C
DynamoDB	Yes		Yes[a]	
Cassandra	Yes		Yes[a]	
Cosmos DB	Yes		Yes	
Riak	Yes		Yes[a]	
VoltDB/H-Store		Yes		Yes
Megastore		Yes		Yes
BigTable/HBase		Yes		Yes
MongoDB	Yes			Yes
PNUTS		Yes	Yes	
Hazelcast IMDG[6]	Yes		Yes	Yes

https://en.wikipedia.org/wiki/PACELC_theorem

«In-Memory OLTP»

- Tabelle «Memory Optimized»
- Stored Procedure compilate
- Strutture dati lock e latch free
- Completamente integrate



«Memory Optimized Tables»

- Struttura
  - Completamente in memoria
  - Indici di tipo hash e range (BW-tree)
  - Compilazione schema e metodi accesso
- Persistenza (durability)
  - Solo Schema o Schema e dati
- Storage
  - Data file e delta file usano tecnologia FILESTREAM
  - Transaction-log

Compiled SQL Modules

- Compilazione codice nativo
  - «Stored Procedures»
  - «After Triggers»
  - «Inline TVFs»
  - «Scalar UDFs»
- Massime prestazioni ma…
  - svariate limitazioni
  - rimosse in ogni nuova versione

Concorrenza «In-Memory OLTP»

- Nuovo protocollo per transazioni MVCC
  - Conforme proprietà ACID
  - Molteplici versioni delle righe
  - Completamente lock-free (timestamp based)
  - Garbage Collector versioni non più utili
- Metodi di accesso «latch/spinlock-free»
  - Operazioni CAS (Compare & Swap)
  - Logica di «Thread assist/abort»

Effetti collaterali MVCC

- Protocollo ottimistico
  - Possibili conflitti tra scritture
  - Validazione prima della commit potrebbe fallire
  - Gestione e logica di «retry» necessaria