Azure SQL Database Ledger

Gianluca Hotz

@glhotz

Data Platform MVP - Presidente UGISS.ORG

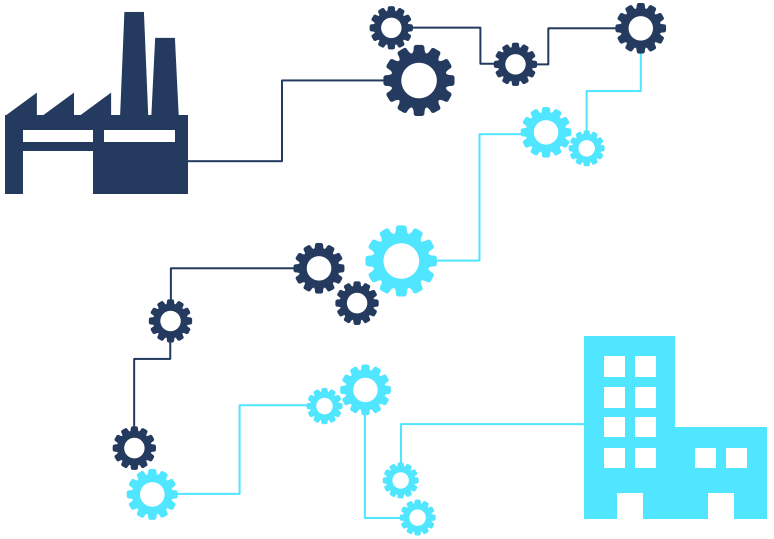
CommunityDays.it

Chi sono?

- Gianluca Hotz | @glhotz | ghotz@ugiss.org
- Consulente indipendente
  - 25 anni su SQL Server (dalla 4.21 nel 1996)
  - Modellazione e sviluppo database, dimensionamento e amministrazione database server, aggiornamenti e migrazioni, performance tuning
- Community
  - 23 anni Microsoft MVP SQL Server/Data Platform (dal 1998)
  - VMware Experts SQL Server
  - Fondatore e presidente UGISS (ex «PASS Chapter»)
  - (Co-organizzatore DAMAG Meetup Community)

Tecnologie «Ledger» e «Digital Trust»

Le aziende si stanno spostando da intermediari e audit manuali che sono **lenti** e **costosi**…



…a tecnologie «Ledger» che **riducono i costi**, fanno **risparmiare tempo** e **riducono i rischi**

CommunityDays.it

Previsioni crescita mercato «blockchain»

1,213 views | May 13, 2020, 10:03am EDT

Will Enterprise Blockchain Survive? Report Puts Blockchain Market At $21 Billion By 2025

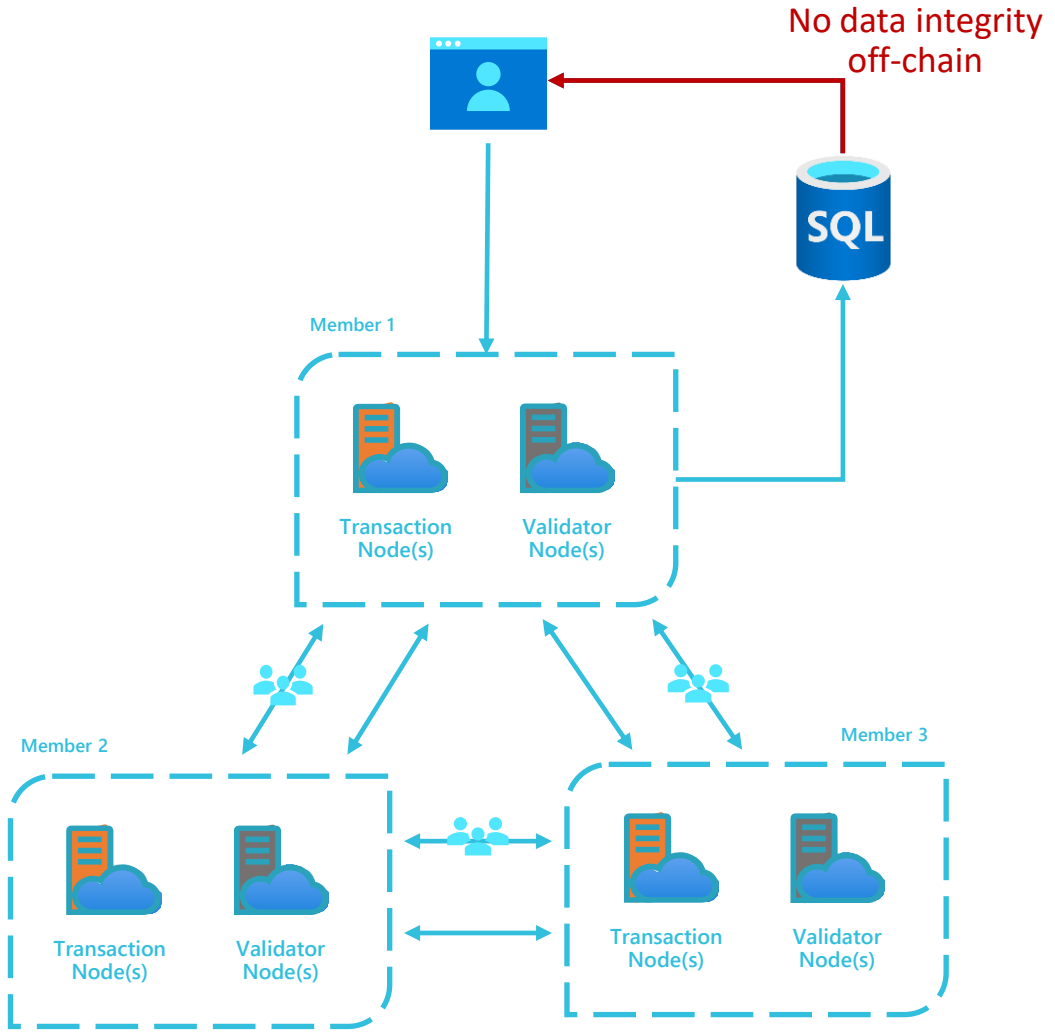
https://www.forbes.com/sites/robertanzalone/2020/05/13/will-enterprise-blockchain-survive-a-new-report-says-that-the-blockchain-technology-market-will-reach-21-billion-by-2025/#7a5f793954b8

Ninety percent of permissioned blockchain projects are misaligned to blockchain technology, because they remain centralized database projects at the core. These projects can be implemented more quickly, more cost-effectively, and with less risk and higher quality by avoiding blockchain altogether.

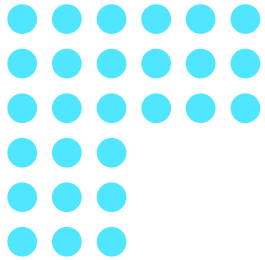
Gartner Predicts 2019: Blockchain Technologies

«Blockchain» esagerate in scenari centralizzati

- Decentramento richiede a tutte le parti di ospitare nodi per partecipare al consenso
- Regole di «governance» devono essere stabilite dal consorzio e distribuite/gestite
- Latenza associata a consenso può influire su velocità effettiva transazioni (<1000 TPS per «Ethereum»
- Archiviazione «off-chain» per interrogazioni è tipica, ma integrità si perde nel processo
- Sistemi personalizzati con tool immaturi rendono **sviluppo e gestione impegnativi**



Azure SQL Database Ledger



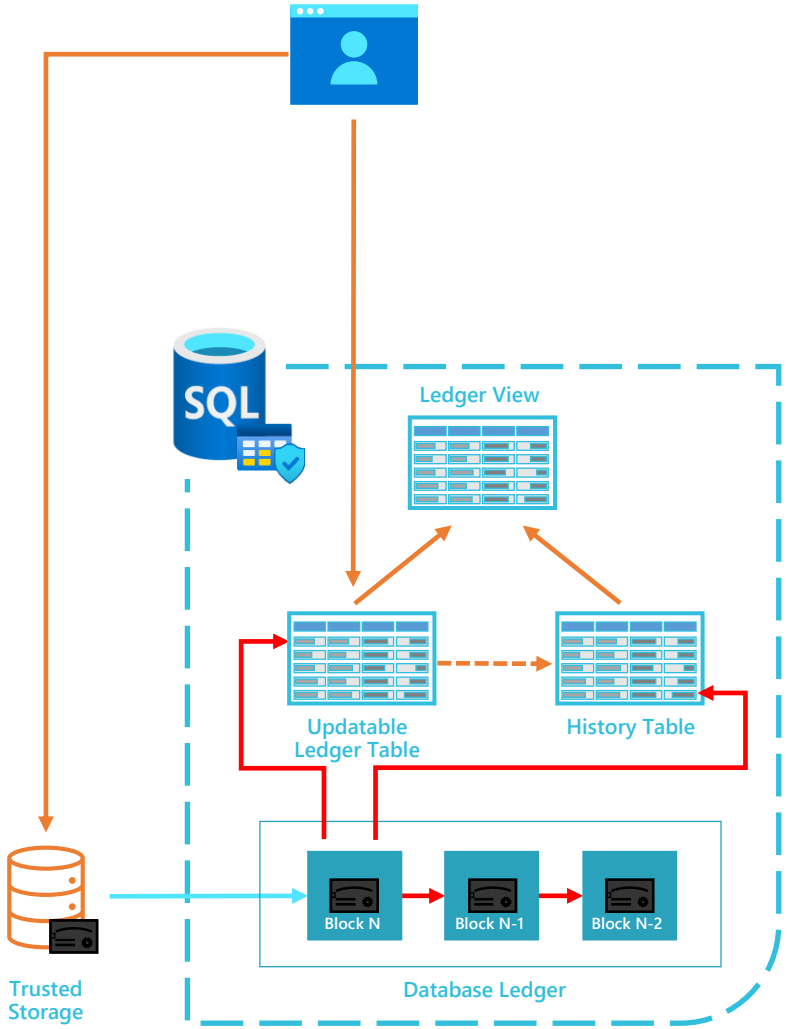
Rende dati in SQL a prova di manomissione tramite crittografia

Fornisce traccia cronologica delle modifiche, verificata tramite prove crittografiche

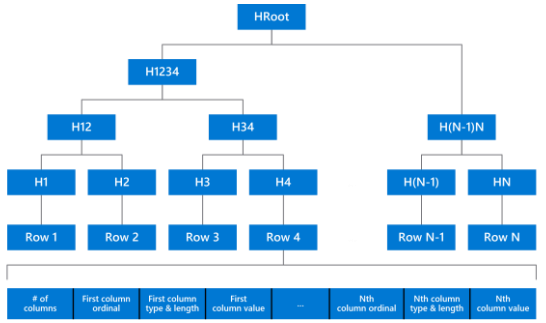
Lo stesso SQL Server già usato in Azure e «on-premises»

Tabelle «Ledger»

- **«Updatable»** permettono «insert/update/delete»
- Cronologia aggiornamento mantenuta tabella storica e nella «Ledger View» di facile consultazione
- Integrità tabelle «aggiornabile» e «storico» mantenuta tramite collegamenti crittografici nel «Database Ledger»
- Ricevute digitali caricate periodicamente in storage attendibile configurato dal cliente
- Cliente può utilizzare ricevute digitali per verificare integrità dei dati
- **«Append-Only»** permesso solo «insert»
  - rimuovono necessità tabella «storico»



«Database Ledger»



- Cattura incrementalmente stato database
  - A livello logico: «blockchain» e strutture dati «Merkle Tree»
  - Cattura anche metadati transazione (es. timestamp, utente)
- Blocchi e informazioni transazioni in tabelle di sistema
  - sys.database_ledger_transactions
  - sys.database_ledger_blocks
- Blocchi chiusi
  - ogni 30 secondi
  - oppure esecuzione manuale sys.sp_generate_database_ledger_digest

«Database Digest»

- Hash ultimo blocco «Ledger»
  - Rappresenta stato di tutte le tabelle «Ledger»
- Devono essere mantenuti in uno storage affidabile e immutabile
  - Altrimenti si potrebbero manomettere le informazioni (in teoria)
- Possibilità di generarli manualmente o in automatico
- Generati in automatico possono essere anche salvati in automatico
  - «Immutable Blob Storage»
  - «Azure Confidential Ledger» (ACL)

Storage attendibile

«Immutable Blob Storage»

- Storage «Write Once, Read Many» basato su **policy**
- BLOB possono essere impostati in sola lettura per intervallo specificato
- Dati bloccati solo funzionalmente in base a policy
- Supporto per «audit logging» ma creatore log deve essere considerato attendibile
- Microsoft è la «Trusted Computing Base»

«Azure Confidential Ledger» (ACL)

- Storage «Write Once, Read Many» in **perpetuo**
- BLOB scritti nel «Ledger» non possono essere modificati
- Utilizza «Confidential Enclaves» a prova di manomissione
- Crea ricevute transazioni e file «Ledger» serializzati contenenti informazioni che possono essere verificate dai clienti
- Microsoft è fuori dalla «TCB», codice sorgente è open source (Confidential Consortium Framework)

Verifica del «Ledger»

- Manomissione possibile a meno di modifiche impedite/tracciate
  - Es. modifica diretta file dati, DBCC WRITEPAGE ecc.
- Verifica ricalcola tutti gli hash e li confronta con i «digest»
  - Operazione richiede uso intensivo di risorse
- Verificare
  - quando necessario (es. sospetto manomissione, audit)
  - su base ricorrente (es. giornalmente, ogni ora)
- Verifica tramite procedura di sistema dipende da modalità salvataggio
  - automatico: passando indirizzo storage
  - manuale: passando documento JSON con «Digest»

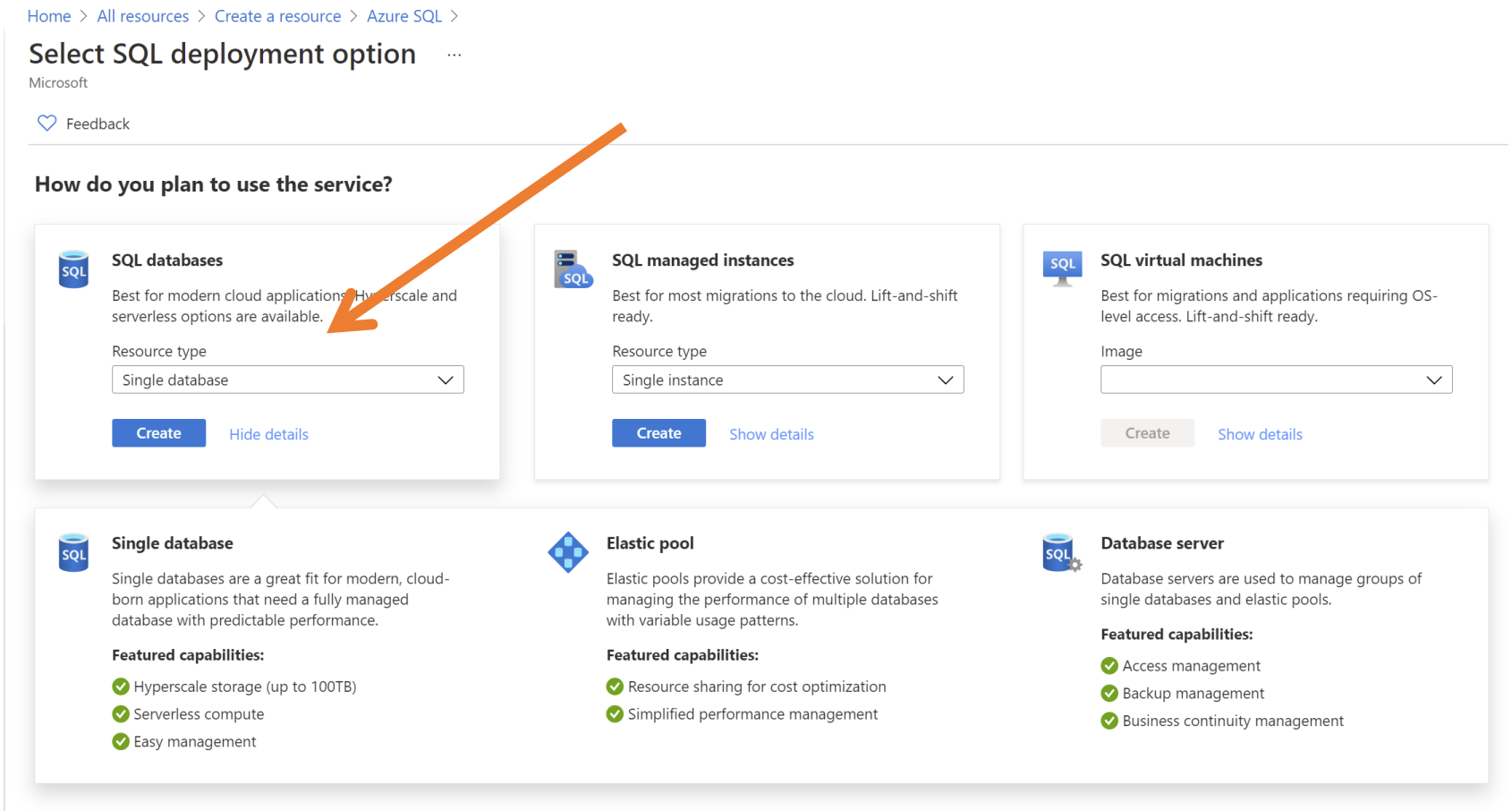
«Ledger auditing»

- Nuovi eventi SQL Audit
  - ENABLE LEDGER
    - Creazione tabelle (o conversione, non ancora supportata)
  - ALTER LEDGER
    - Eliminazione/cambio nome tabelle (non ancora supportati)
  - GENERATE LEDGER DIGEST
  - VERIFY LEDGER
  - LEDGER_OPERATION_GROUP

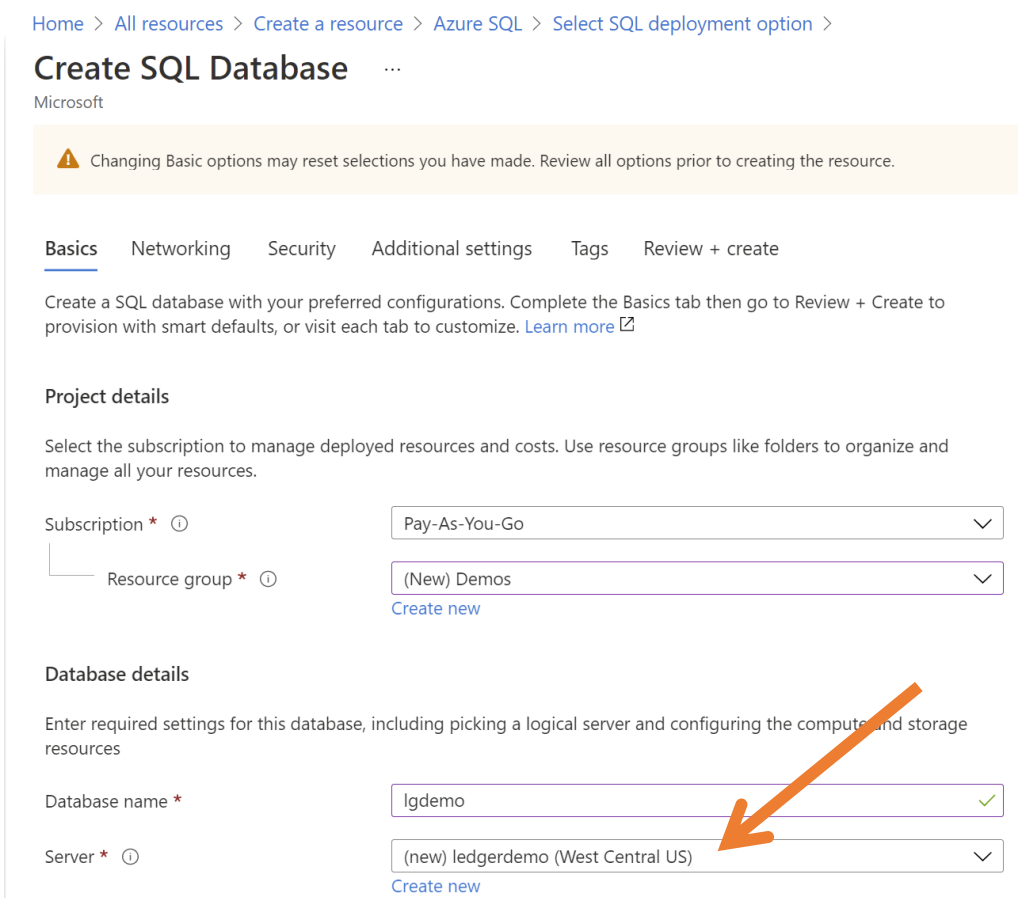
Demo

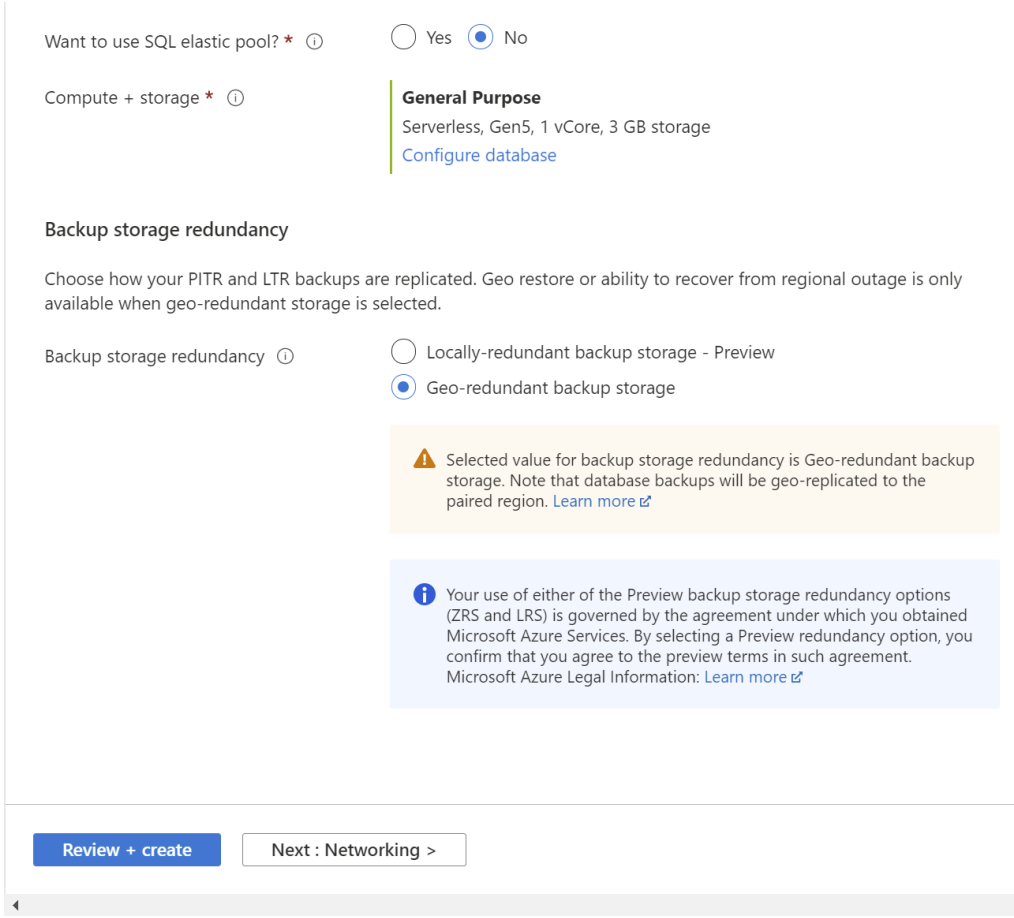
Abilitazione Database Ledger

Selezione opzione «deployment»

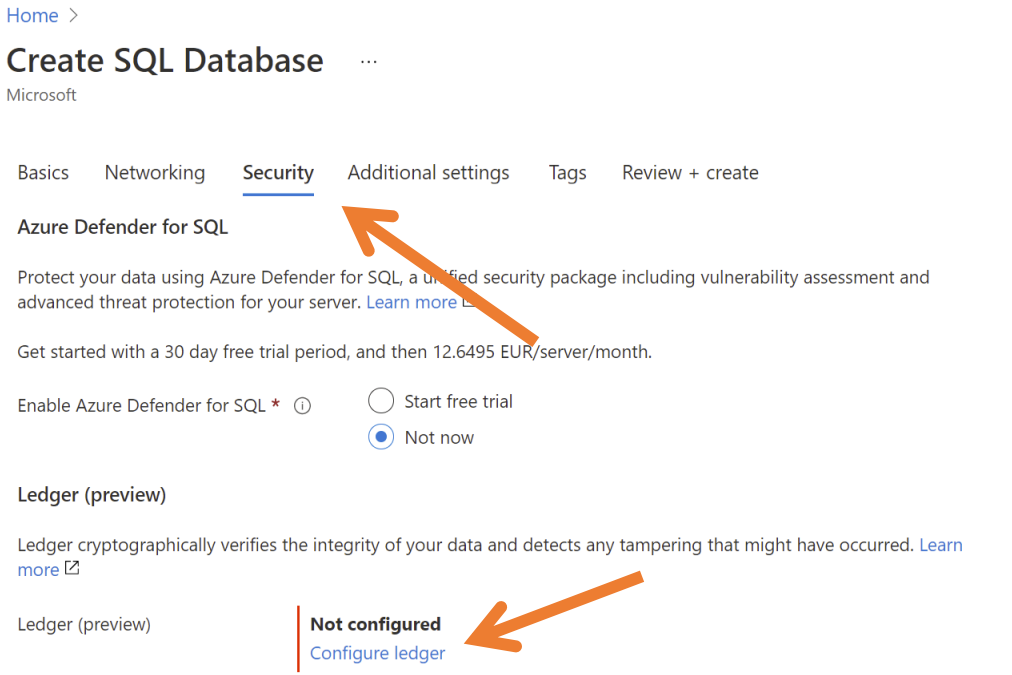


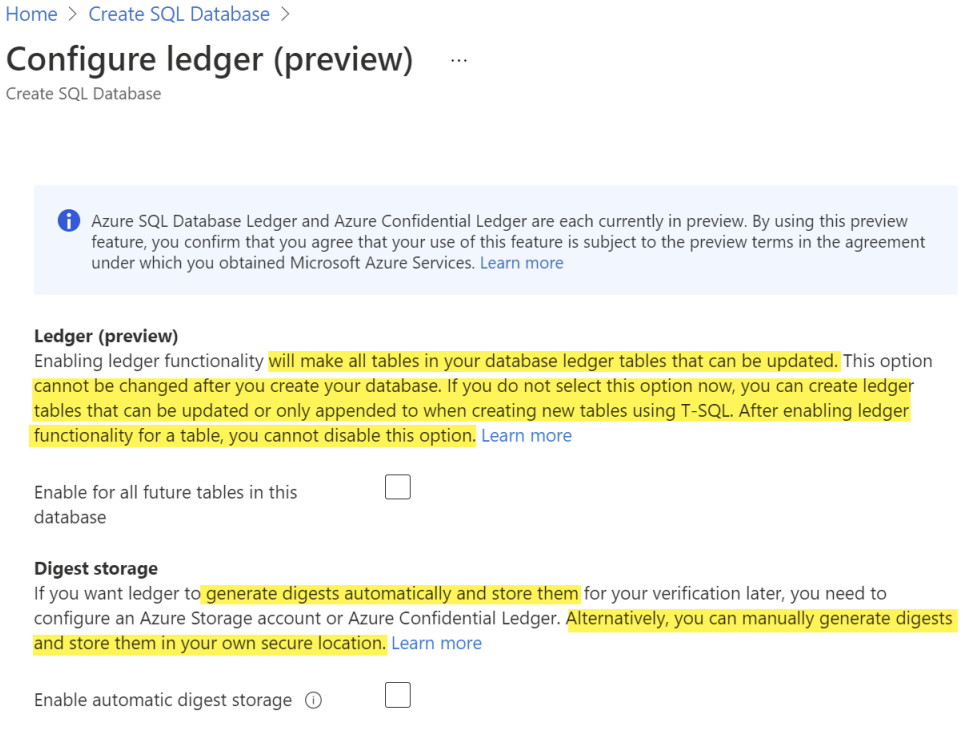
Creazione Database



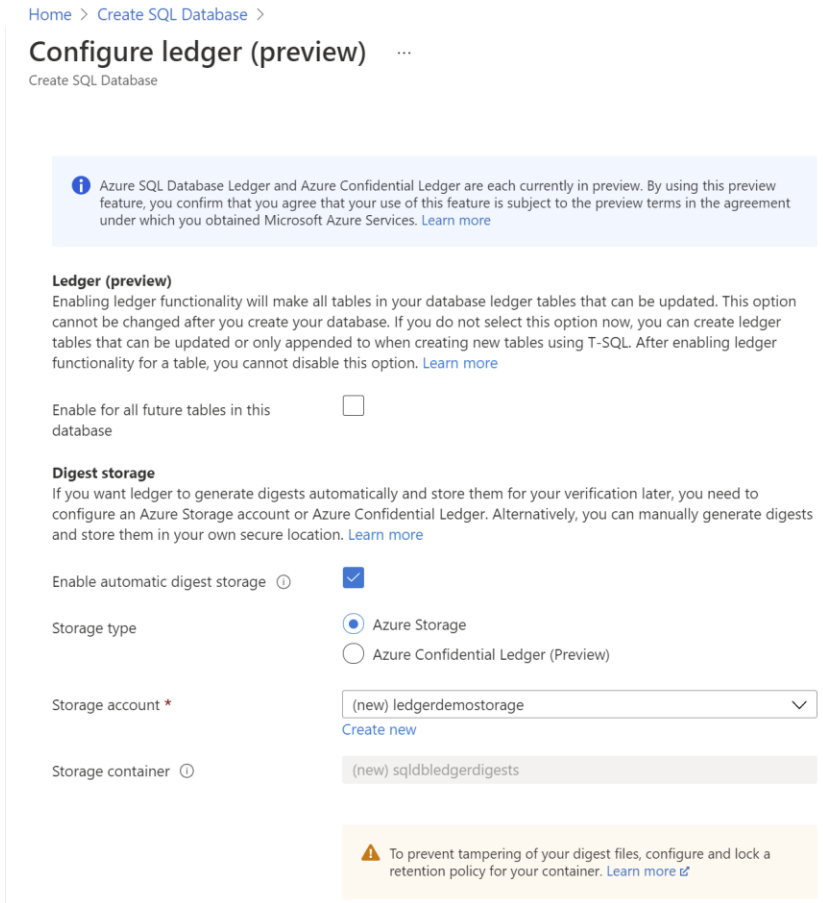


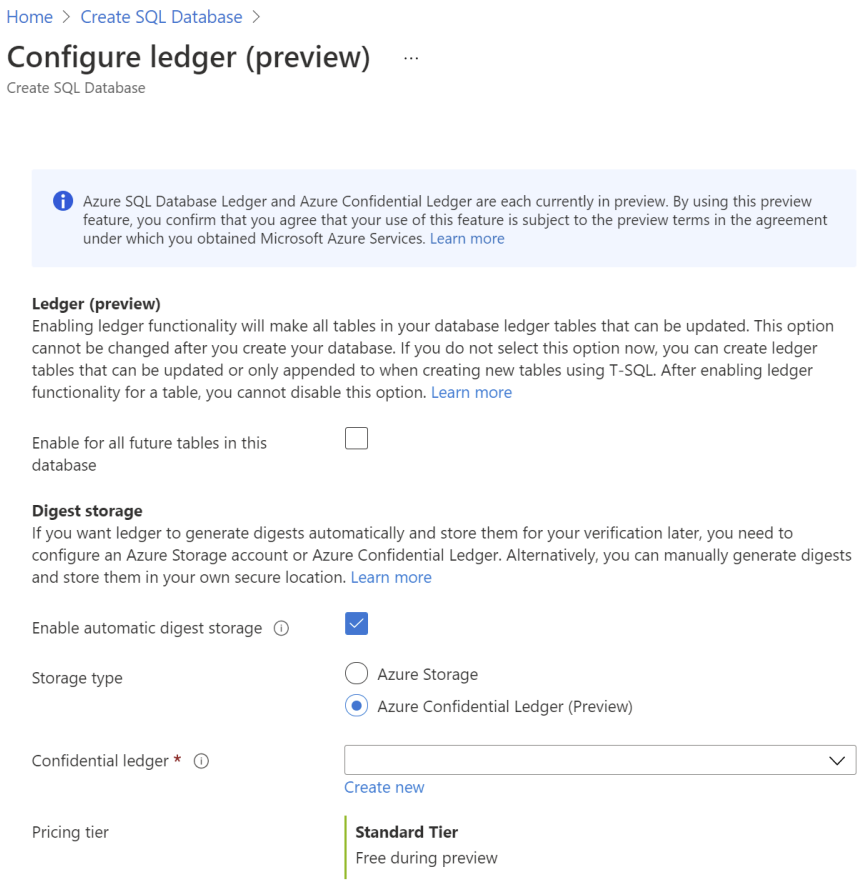
Configurazione «Ledger»





«Digest Storage»





Demo

Ledger Tables

Tabella Aggiornabile

```
CREATE SCHEMA [Account];
GO
CREATE TABLE [Account].[Balance]
(
        [CustomerID]    int             NOT NULL PRIMARY KEY CLUSTERED
,       [LastName]      varchar(50)     NOT NULL
,       [FirstName]     varchar(50)     NOT NULL
,       [Balance]       decimal(10,2)   NOT NULL
)
WITH (
        SYSTEM_VERSIONING = ON --(HISTORY_TABLE = [Account].[BalanceHistory])
,       LEDGER = ON --(LEDGER_VIEW = [Account].[BalanceLedgerView])
);
GO
```

Transazioni di inserimento

```
-- Prima transazione
INSERT INTO [Account].[Balance]
VALUES
        (1, 'Jones', 'Nick', 50);
GO

-- Seconda transazione
INSERT INTO [Account].[Balance]
VALUES
        (2, 'Smith', 'John', 500)
,       (3, 'Smith', 'Joe', 30)
,       (4, 'Michaels', 'Mary', 200);
GO
```


Selezione da tabella

```
-- Di default le colonne con le informazioni relative alle
-- transazioni non vengono tornate (trasparente applicazioni)
SELECT *
FROM [Account].[Balance];
GO
```

Results Messages

	CustomerID	LastName	FirstName	Balance
1	1	Jones	Nick	50.00
2	2	Smith	John	500.00
3	3	Smith	Joe	30.00
4	4	Michaels	Mary	200.00

Selezione da tabella campi aggiuntivi

```
-- Devono essere selezionate esplicitamente
SELECT *
,       [ledger_start_transaction_id]
,       [ledger_end_transaction_id]
,       [ledger_start_sequence_number]
,       [ledger_end_sequence_number]
FROM [Account].[Balance];
GO
```

Results Messages

	CustomerID	LastName	FirstName	Balance	ledger_start_transaction_id	ledger_end_transaction_id	ledger_start_sequence_number	ledger_end_sequence_number
1	1	Jones	Nick	50.00	1420	NULL	0	NULL
2	2	Smith	John	500.00	1423	NULL	0	NULL
3	3	Smith	Joe	30.00	1423	NULL	1	NULL
4	4	Michaels	Mary	200.00	1423	NULL	2	NULL

Aggiornamento

```
UPDATE[Account].[Balance]
SET[Balance] = 100
WHERE[CustomerID] = 1;
GO
```

Interrogazione dopo aggiornamento

```
-- Interroghiamo la tabella aggiornabile, quella di storico e la vista
SELECT *
,[ledger_start_transaction_id]
,[ledger_end_transaction_id]
,[ledger_start_sequence_number]
,[ledger_end_sequence_number]
FROM [Account].[Balance];

SELECT * FROM [Account].[MSSQL_LedgerHistoryFor_1525580473];

SELECT * FROM [Account].[Balance_Ledger] ORDER BY [ledger_transaction_id];
GO
```

Risultato interrogazione

Results Messages

	CustomerID	LastName	FirstName	Balance	ledger_start_transaction_id	ledger_end_transaction_id	ledger_start_sequence_number	ledger_end_sequence_number
1	1	Jones	Nick	100.00	1432	NULL	0	NULL
2	2	Smith	John	500.00	1423	NULL	0	NULL
3	3	Smith	Joe	30.00	1423	NULL	1	NULL
4	4	Michaels	Mary	200.00	1423	NULL	2	NULL

	CustomerID	LastName	FirstName	Balance	ledger_start_transaction_id	ledger_end_transaction_id	ledger_start_sequence_number	ledger_end_sequence_number
1	1	Jones	Nick	50.00	1420	1432	0	1

	CustomerID	LastName	FirstName	Balance	ledger_transaction_id	ledger_sequence_number	ledger_operation_type	ledger_operation_type_desc
1	1	Jones	Nick	50.00	1420	0	1	INSERT
2	2	Smith	John	500.00	1423	0	1	INSERT
3	3	Smith	Joe	30.00	1423	1	1	INSERT
4	4	Michaels	Mary	200.00	1423	2	1	INSERT
5	1	Jones	Nick	50.00	1432	1	2	DELETE
6	1	Jones	Nick	100.00	1432	0	1	INSERT

Tabella «Append-Only»

```
CREATE SCHEMA [AccessControl];
GO
CREATE TABLE [AccessControl].[KeyCardEvents]
(
      [EmployeeID]                int           NOT NULL PRIMARY KEY CLUSTERED
,     [AccessOperationDescription] nvarchar(MAX) NOT NULL
,     [Timestamp]                 datetime2     NOT NULL
)
WITH (
      LEDGER = ON (APPEND_ONLY = ON)
);
GO
```

Inserimento e aggiornamento

```
-- Inseriamo una prima riga
INSERT INTO [AccessControl].[KeyCardEvents]
VALUES ('43869', 'Building42', '2020-05-02T19:58:47.1234567');
GO
-- Se proviamo a fare un'aggiornamento, da errore
UPDATE [AccessControl].[KeyCardEvents]
SET[EmployeeID] = 34184
WHERE[EmployeeID] = 43869;
GO
```

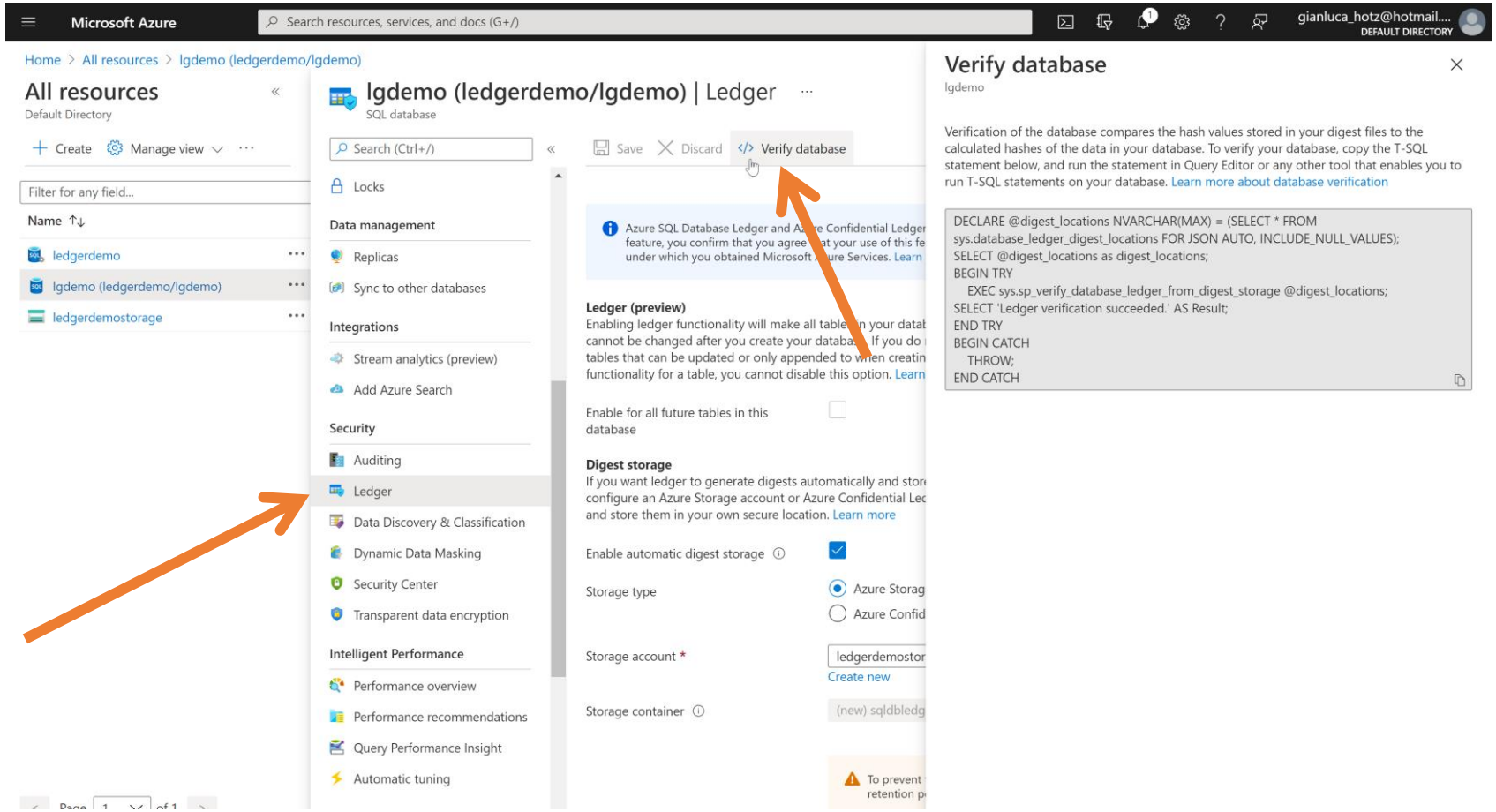
Messages

```
Msg 37359, Level 16, State 1, Line 141
Updates are not allowed for the append only Ledger table 'AccessControl.KeyCardEvents'.
```

Demo

Verifica Database

Verifica Database



CommunityDays.it

Limitazioni generali

- Opzione a livello di database non può essere disabilitata
- No conversione tabelle esistenti
- No cambio di nome o spostamento di «schema»
- No eliminazione dati tabelle di storico (tabelle aggiornabili)
- Transazione può aggiornare solo (!) 200 tabelle
- «Long-term backups» (LTR) non supportati
- «Ledger» aggiornabili ereditano limitazioni tabelle temporali

Limitazioni di interoperabilità

- Tabelle «In-memory» non supportate
- No operazioni di SWITCH IN/OUT
- No indici di tipo «Full-Text»
- No indice «non-clustered rowstore» con «clustered columnstore»
- No «Change Tracking»
- No tabelle FILETABLE
- No utilizzo API UPDATETEXT e WRITETEXT

Limitazioni schema

- Numero massimo di colonne (sempre 1024)
  - Tabelle aggiornabili +4 colonne
  - Tabelle solo accodamento +2 colonne
- Aggiunta solo di colonne «nullable» (senza WITH VALUES)
- No eliminazione colonne, modifica limitata:
  - NULL/NOT NULL, lunghezza tipi a lunghezza variabile, SPARSE
  - «Collation» per tipi Unicode, se non cambia «code page» per gli altri
- No tipi dato XML, FILESTREAM, SqlVariant e «user-defined»
- Colonne «computed» solo deterministiche
- No «Sparse Column Set»

Scenari per tabelle «Ledger»

- In generale: quelli che necessitano solo della «Forward Integrity»
  - Sistema attendibile al processamento della transazione, protetto contro manomissioni future
- Alcuni esempi
  - Semplificazione scenari di audit
    - Verifica crittografica manomissione dati verso terze parti (interne o esterne)
  - Processi aziendali tra più parti
    - Alternativa a «blockchain» per sistemi intrinsecamente centralizzati in ottica «trust, but verify»
  - Storage «off-chain» affidabile per interrogazioni dati «blockchain»
- «Choosing an Azure ledger technology»
  - https://techcommunity.microsoft.com/t5/azure-sql/choosing-an-azure-ledger-technology/ba-p/2450502

Risorse

- Announcement blog
  - https://aka.ms/sql-ledger-blog
- Azure SQL Database ledger Documentation
  - https://aka.ms/sql-ledger-docs
- Whitepaper
  - https://aka.ms/sql-ledger-whitepaper

Grazie!

- Il materiale sarà online nei prossimi giorni su http://www.communitydays.it