Лекція 6. Введення у сховища даних, *Azure Table Storage*

Тема 1. Основи роботи з Azure Table Storage — поняття сутності (Entity), PartitionKey, RowKey, Timestamp та властивостей (Properties).
Тема 2. Типи запитів до Azure Table Storage — точкові запити (Point Queries), діапазонні запити (Range Queries), фільтрація та сортування.
Тема 3. Основні сценарії використання Azure Table Storage.

Таблиці - Azure Table Storage

Azure Table Storage — це один з сервісів сховища даних в хмарі від Microsoft Azure. Він дозволяє зберігати структуровані дані в таблицях NoSQL. Це хмарний сервіс, що дозволяє зберігати великі обсяги даних із високою доступністю та масштабованістю.

Типи даних

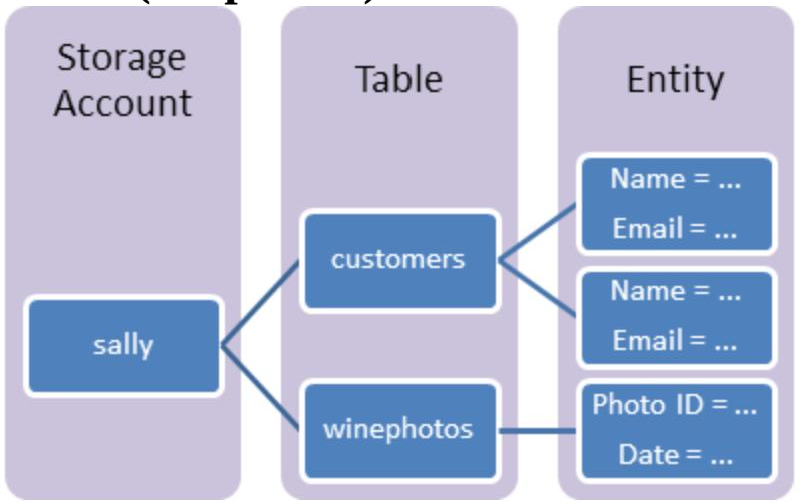
Azure Table Storage зберігає дані у вигляді сутностей (entities), кожна з яких має два основних поля — PartitionKey і RowKey. Вони формують унікальний ідентифікатор сутності.

Основи роботи з Azure Table Storage

Сутність (Entity) є найменшою одиницею даних у таблиці.

Кожна сутність складається з PartitionKey, RowKey, Timestamp та властивостей (Properties).

Формат http://<storage account>.table.core.windows.net/<table>



Загальні обмеження по ресурсах

Ресурс	**Обмеження / Цільовий показник**
Кількість таблиць у сховищі	Не обмежено — лише місткістю облікового запису
Кількість партицій у таблиці	Не обмежено — лише місткістю облікового запису
Кількість сутностей у партиції	Не обмежено — лише місткістю облікового запису
Максимальний розмір однієї таблиці	**500 ТіБ (**1 ТіБ (тебібайт) дорівнює 1024 ГіБ (гібібайтам), 1 ГіБ = 1024 МіБ, 1 МіБ = 1024 КіБ, а 1 КіБ = 1024 байти, що в цілому складає 2^40 байтів
Максимальний розмір сутності (рядка)	**1 МіБ** (включно з усіма властивостями)
Кількість властивостей у сутності	**255** (включно з системними: PartitionKey, RowKey, Timestamp)
Максимальний розмір однієї властивості	Залежить від типу даних (див. нижче)
Довжина PartitionKey	Рядок до **1024 символів**
Довжина RowKey	Рядок до **1024 символів**
Розмір транзакції (entity group transaction)	До **100 сутностей**, **менше 4 МіБ** у сумі
Оновлення однієї сутності в транзакції	Один раз за транзакцію
Кількість збережених політик доступу на таблицю	**До 5**
Максимальна кількість запитів на секунду (на акаунт)	**20 000 транзакцій/с** при розмірі сутності ~1 КіБ
Продуктивність для однієї партиції (1 КіБ-сутності)	До **2 000 сутностей/с**

- ✓ Максимальний розмір сутності (1 МіБ) :
  - ✓ Всі властивості сутності.
  - ✓ Назви властивостей.
  - ✓ Системні метадані

  (Якщо вам потрібно зберегти більший об’єм — зберігайте дані у Blob Storage, а в таблиці — лише метадані.)
- ✓ Транзакції, що охоплюють кілька сутностей, допускаються лише в межах однієї партиції.

Обмеження: До 100 сутностей. Загальний об’єм — менше 4 МіБ

Одна сутність не може оновлюватися двічі в межах однієї транзакції

- ✓ Продуктивність (Throughput)

Рівень	Продуктивність
Storage account	До **20 000 транзакцій/сек** (при розмірі сутності ~1 КіБ)
Одна партиція	До **2 000 транзакцій/сек** (1 КіБ-сутності)

Визначте розмірність ТіБ (тебібайта): За стандартом, тебібайт (ТіБ) — це двійкова одиниця вимірювання, що дорівнює 1024 гігабайтам (ГіБ).

Визначте розмірність ГіБ: Гібібайт (ГіБ) дорівнює 1024 мегабайтам (МіБ).

Визначте розмірність МіБ: Мебібайт (МіБ) дорівнює 1024 кілобайтам (КіБ).

Визначте розмірність КіБ: Кібібайт (КіБ) дорівнює 1024 байтам.

Розрахуйте кількість байтів в 1 ТіБ: 1 ТіБ = 1024 * 1024 * 1024 * 1024 байт = 1 099 511 627 776 байт.

Розрахуйте кількість байтів в 500 ТіБ: 500 * 1 099 511 627 776 байт = 549 755 813 888 000 байт.

Завдання

1. Розрахуйте скільки сутностей розміром 500 КіБ поміститься в одну таблицю розміром 500 ТіБ (тебібайт) ?
2. Сформулюйте структуру сутності так, щоб не перевищити 255 властивостей і 1 МіБ загального розміру?

Відповіді

Завдання 1. Вихідні дані:

Розмір таблиці: 500 ТіБ = 500 × 1024^4 байт = 549755813888000 байт

Розмір однієї сутності: 500 КіБ = 500 × 1024^2 байт = 512000 байт

Кількість сутностей = Розмір таблиці / Розмір сутності

= 549755813888000 / 512000

= 1 073 741 824 сутностей

Відповіді

Завдання 2. Сформулюйте структуру сутності так, щоб не перевищити 255 властивостей і 1 МіБ загального розміру.

Обмеження:

Максимум 255 властивостей на сутність (включаючи PartitionKey, RowKey, Timestamp)

Загальний розмір сутності — до 1 МіБ = 1 × 2^{20} = 1 048 576 байт

```
{
  "PartitionKey": "user_001",        // ~10 байт
  "RowKey": "order_123",             // ~10 байт
  "Timestamp": "2023-10-01T12:00Z",   // ~20 байт
  "Name": "Ivan Petrenko",           // ~30 байт
  "Email": "ivan.petrenko@email.com", // ~35 байт
  "Age": 30,                         // 8 байт (Int64)
  "IsVerified": true,                // 1 байт (Boolean)
  ...
  // Далі ще до 248 користувацьких властивостей
}
```

Разом: (PartitionKey, RowKey) формують **унікальний ключ** для кожного запису (entity). Таблиця масштабується за PartitionKey.

Компонент	Опис
PartitionKey	Групує дані в логічні "розділи". Всі записи з однаковим PartitionKey зберігаються разом.
RowKey	Унікальний ідентифікатор рядка в межах PartitionKey.

Обирати PartitionKey так, щоб:

Данні логічно групувалися (наприклад, всі замовлення одного клієнта).

Кількість записів в одній партиції була збалансованою (не 90% в одній, а 10% в решті).

RowKey має бути:

Унікальним в межах PartitionKey

Стабільним (не змінюється після створення)

Приклади визначення PartitionKey, RowKey

Сценарій	PartitionKey	RowKey
Замовлення клієнтів	customer_id	order_id
Дані сенсорів	device_id	timestamp
Логи користувача	user_id	datetime_log_type
Каталог товарів	category	product_id

Можна комбінувати дані у ключах для досягнення ефективності:

PartitionKey = "customer123"

RowKey = "2025-10-02T09:15:00_order456"

Припустимо, ви зберігаєте дані про покупки в інтернет-магазині, і кожна покупка має таку інформацію:

•**ID покупця** (customer_id)

•**ID замовлення** (order_id)

•**Дата покупки** (purchase_date)

•**Сума покупки** (purchase_amount)

Ваше завдання — ефективно зберігати ці дані в Azure Table Storage . ***Які поля треба визначити – як PartitionKey? Як RowKey?***

1. Проєктування для ефективного читання (Read-efficient design).

У багатьох застосунках (аналітика, звіти, API) навантаження на читання сутностей значно перевищує кількість записів. У таких випадках важливо проєктувати рішення "від запитів".

PartitionKey	RowKey	Timestamp	purchasedate
customer_1	order_1	2025-10-02T16:51:34.4633...	9/14/2023
customer_1	order_10	2025-10-02T16:51:34.8944...	9/23/2023
customer_1	order_2	2025-10-02T16:51:34.5111...	9/15/2023
customer_1	order_3	2025-10-02T16:51:34.5599...	9/16/2023
customer_1	order_4	2025-10-02T16:51:34.6057...	9/17/2023
customer_1	order_5	2025-10-02T16:51:34.6525...	9/18/2023

Складені ключі (Compound Keys) - Гнучкість у фільтрації

Для запитів типу:
"Усі замовлення після певної дати"
"Усі замовлення за датою і номером замовлення"
"Усі замовлення, що були в певному місяці"

Showing all 99 items

PartitionKey	RowKey	Timestamp	purchasedate
customer_1	20230914_order_1	2025-10-02T17:06:27.2672...	9/14/2023
customer_1	20230914_order_10	2025-10-02T17:06:27.6993...	9/23/2023
customer_1	20230914_order_2	2025-10-02T17:06:27.318Z	9/15/2023
customer_1	20230914_order_3	2025-10-02T17:06:27.3647...	9/16/2023
customer_1	20230914_order_4	2025-10-02T17:06:27.4125...	9/17/2023
customer_1	20230914_order_5	2025-10-02T17:06:27.4603...	9/18/2023

Дублювання сутностей (Entity Duplication)

Дозволяє підтримувати кілька способів доступу до однієї і тієї ж інформації, таким чином обійти обмеження використання лише однією парою ключів: PartitionKey + RowKey.
Швидко знаходити всі замовлення конкретного клієнта → потрібен PartitionKey = customer_id
Але також треба знаходити замовлення за номером замовлення (order_id) → потрібен PartitionKey = order_id

	PartitionKey	RowKey	Timestamp	purchasedate
◯	order_1	customer_1	2025-10-02T17:16:57.1636...	9/14/2023

Властивість PartitionKey

Таблиці розподіляються на партиції для підтримки балансування навантаження між вузлами зберігання. Сутності таблиці організовані за партиціями. Партиція — це послідовний діапазон сутностей, що мають однакове значення PartitionKey. PartitionKey — це унікальний ідентифікатор партиції в межах конкретної таблиці, що задається властивістю PartitionKey. Партиційний ключ є першою частиною первинного ключа сутності. Значення партиційного ключа може бути рядком до 1024 символів.

Властивість PartitionKey повинна бути включена в кожну операцію вставки, оновлення та видалення.

Властивість RowKey

Друга частина первинного ключа — це рядковий ключ, що визначається властивістю RowKey. RowKey — це унікальний ідентифікатор сутності в межах певної партиції. Разом PartitionKey і RowKey унікально ідентифікують кожну сутність у таблиці.

RowKey — це рядкове значення, що може бути до 1024 символів.

Властивість RowKey повинна бути включена в кожну операцію вставки, оновлення та видалення.

Властивість Timestamp

Властивість Timestamp — це значення типу DateTime, яке підтримується на сервері для реєстрації часу останнього редагування сутності. Служба таблиць використовує властивість Timestamp внутрішньо для підтримки оптимістичної одночасності. Значення властивості Timestamp для сутності збільшується щоразу, коли сутність змінюється. Цю властивість не слід встановлювати при операціях вставки або оновлення (значення буде проігноровано).

Властивість Timestamp повинна бути виражена в одному з прийнятих форматів ISO 8601 UTC. Для отримання додаткової інформації про прийняті формати UTC, див. розділ «Форматування значень DateTime».

Створити таблицю на Azure порталі

Імена таблиць повинні відповідати таким правилам:

Імена таблиць повинні бути унікальними в межах облікового запису.

Імена таблиць можуть містити лише алфавітно-цифрові символи.

Імена таблиць не можуть починатися з числових символів.

Імена таблиць не залежать від регістру.

Імена таблиць повинні містити від 3 до 63 символів.

Деякі імена таблиць зарезервовані, зокрема "tables". Спроба створити таблицю з зарезервованим ім'ям таблиці повертає код помилки 404 (Невірний запит).

Ці правила також описуються регулярним виразом "^[A-Za-z][A-Za-z0-9]{2,62}$"

3. Розробіть PartitionKey-стратегію, яка дозволяє масштабування до 20 000 транзакцій/сек?

Завдання 3. Вихідні дані:

Розробіть PartitionKey-стратегію, яка дозволяє масштабування до 20 000 транзакцій/сек.

Вихідні дані:

Продуктивність однієї партиції: до 2 000 транзакцій/сек

Для 20 000 транзакцій/сек потрібно принаймні 10 партицій

```
PartitionKey = f"{datetime.utcnow():%Y%m%d}_part_{random.randint(0, 9)}"
```

Хешування або суфікс UUID: Створює велику кількість унікальних партицій. Забезпечує високу паралельність, але ускладнює пошук.

```
PartitionKey = "order_" + str(uuid.uuid4())[:4]
```

Дає **100 партицій**, що добре масштабується

```
PartitionKey = f"user_{int(user_id) % 100}"
```

Запити до таблиц

Служба таблиць підтримує два типи запитів:

✓ Query Tables Operation

Повертає список таблиць у вказаному обліковому записі зберігання.

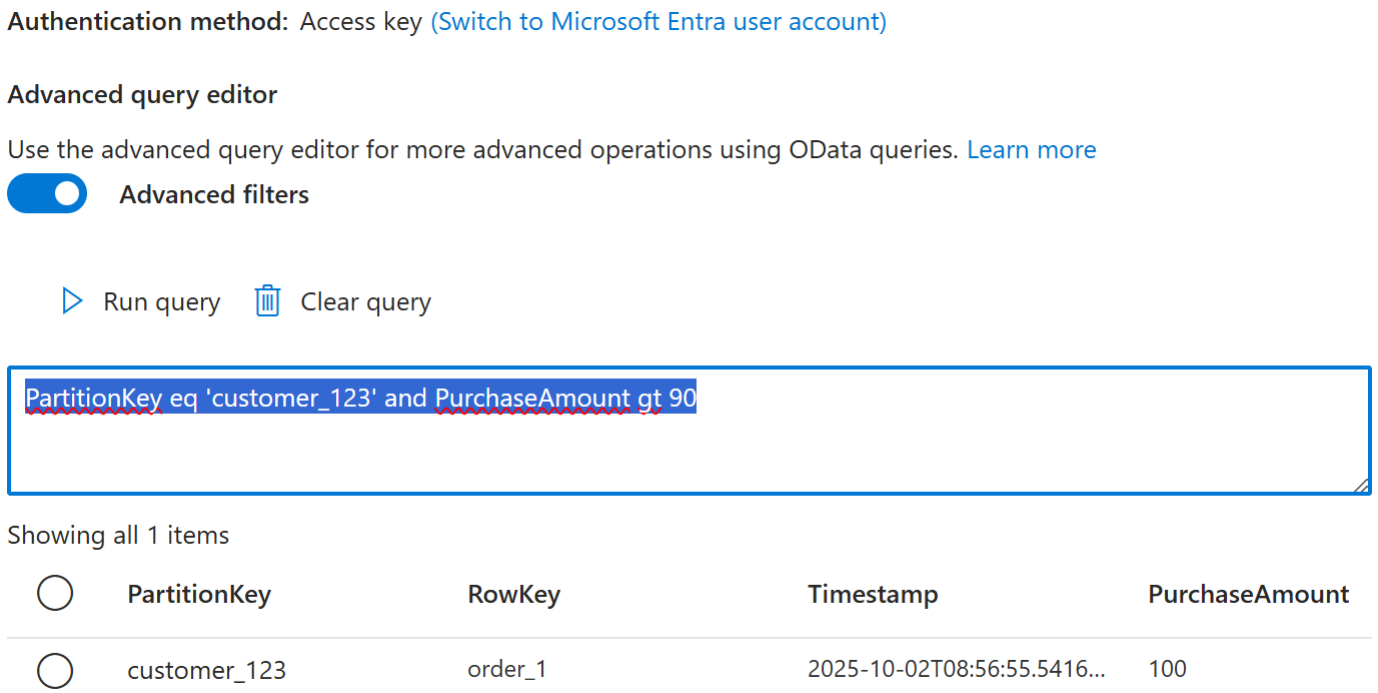
Можна застосовувати фільтри (наприклад, за префіксом назви таблиці).

✓ Query Entities Operation

Повертає набір сутностей (рядків) з вказаної таблиці.

Дає можливість фільтрувати результати за умовами (OData syntax):

Наприклад:



Тпи запитів

Точкові запити (Point Queries):
Точкові запити — це запити за значеннями PartitionKey та RowKey, що дозволяють отримати одну сутність.
PartitionKey eq 'user_123' and RowKey eq 'order_567'

Діапазонові запити (Range Queries):
Запити, що вибирають сутності на основі діапазону значень RowKey в межах одного розділу. Це дозволяє отримувати кілька сутностей за один запит.

Фільтрація та сортування:
Azure Table Storage дозволяє застосовувати фільтри до запитів, але потрібно пам'ятати, що сортування за іншими властивостями, окрім RowKey, може бути неефективним.

Фільтрація у запитах до Azure Table Storage

$filter — це спеціальний параметр у запитах до Azure Table Storage (OData-протокол), який дозволяє відфільтрувати сутності таблиці за певними умовами.
Порівняльні оператори (Comparison Operators)

Назва оператора	Вираз в URI	Приклад
Дорівнює	eq	purchaseamount eq 100
Не дорівнює	ne	status ne 'cancelled'
Більше	gt	age gt 21
Менше	lt	score lt 50
Більше або рівне	ge	balance ge 0
Менше або рівне	le	rating le 5

PartitionKey eq 'customer_123' and PurchaseAmount gt 90

Кодування символів у рядках запиту (Query String Encoding)

Під час формування $filter запиту, деякі символи потрібно кодувати, оскільки вони мають спеціальне значення в URL.

Символ	Значення	Приклад кодування
/	Слеш	%2F
?	Знак питання	%3F
:	Двокрапка	%3A
@	At	%40
&	Амперсанд	%26
=	Знак рівності	%3D
+	Плюс	%2B
,	Кома	%2C
$	Долар	%24
'	Одинарна лапка	'' (подвійна лапка!)

Завдання

1. Спроєктуйте структуру таблиці для системи логування, яка підтримує 10 000 записів/сек з мінімальною затримкою при пошуку за типом події.
2. Складіть план дублювання сутностей для системи, де запити можливі як за userId, так і за orderId.

Завдання 1.

Потрібно забезпечити високу швидкість запису: 10 000 записів/сек. І водночас — швидкий доступ до подій за типом (eventType). Потрібно уникнути "гарячих партицій"

Рішення:

PartitionKey:

Комбінувати тип події + час → це дозволяє: Розподілити події між різними типами. Уникнути концентрації на одному ключі

PartitionKey = f"{event_type}_{datetime.utcnow():%Y%m%d%H%M}_{random.randint(0, 99)}"

RowKey: Використати випадковий або монотонно зростаючий унікальний ключ

RowKey = str(uuid.uuid4())

```
{
  "PartitionKey": "error_202510021215_43",
  "RowKey": "d9f1c5e2-...",
  "eventType": "error",
  "timestamp": "2025-10-02T12:15:30Z",
  "message": "NullReferenceException at line 44",
  "source": "ServiceA"
}
```

Завдання 2:

Складіть план дублювання сутностей для системи, де запити можливі як за userId, так і за orderId.

Аналіз завдання:

Потрібно забезпечити дві моделі доступу:

Отримати всі замовлення користувача (userId)

Знайти замовлення за його ідентифікатором (orderId)

Рішення:

Дублюємо кожну сутність у 2 таблицях або 2 партиціях однієї таблиці:

Варіант 1: Одна таблиця, 2 копії сутності:

```
// Запис 1
{
 "PartitionKey": "user_101",
 "RowKey": "ORD123",
 "amount": 120.50,
 "status": "confirmed"
}

// Запис 2
{
 "PartitionKey": "order_ORD123",
 "RowKey": "101",
 "amount": 120.50,
 "status": "confirmed"
}
```

TableServiceClient — це головний клас, який дозволяє взаємодіяти з Azure Table Storage на рівні облікового запису (storage account).
З його допомогою можна:

- ✓ Підключатися до облікового запису
- ✓ Створювати, видаляти або перераховувати таблиці
- ✓ Отримувати об'єкти TableClient для доступу до конкретних таблиць

Основні методи TableServiceClient

Метод	Опис
create_table(table_name)	Створює нову таблицю
create_table_if_not_exists(table_name)	Створює таблицю, якщо вона ще не існує
delete_table(table_name)	Видаляє таблицю
list_tables()	Повертає всі таблиці в обліковому записі
get_table_client(table_name)	Повертає TableClient — об'єкт для доступу до конкретної таблиці

Взаємодія з TableClient

TableServiceClient повертає об'єкти TableClient, які вже дозволяють:

- ✓ Додавати сутності
- ✓ Читати сутності
- ✓ Фільтрувати дані
- ✓ Оновлювати або видаляти сутності

Розглянемо приклад CRUD з методами класа ***TableClient***

Обмеження на виконання запитів

Запити до служби таблиць мають обмеження:

Обмеження	Значення
Максимум результатів за один запит	**1 000** сутностей
Максимальний час виконання запиту	**5 секунд**
Загальний час запиту (включно з чергою та обробкою)	**30 секунд**

Continuation Tokens — для поділу запиту на частини

Якщо:

результат містить понад **1 000 сутностей,**

або запит триває довше **5 секунд**,

або запит охоплює кілька партицій (PartitionKey),

— то служба повертає заголовки з токенами продовження (continuation tokens), щоб ви могли отримати решту результатів у наступному запиті.

Навіть якщо результат порожній, continuation token може бути повернутий — отже, варто перевіряти наявність токена.

Continuation Token — це спеціальний маркер, який вказує, з якого місця продовжити читання даних у наступному запиті.

Клієнт повинен:

Перевірити наявність continuation token у відповіді.

Якщо він є — зробити наступний запит, передавши цей токен.

```
# Псевдокод на Python
entities = []
token = None

do:
    result = table_client.query_entities(filter="PartitionKey eq 'A'", continuation_token=token)
    entities.extend(result.items)
    token = result.continuation_token
while token is not None
```

Table Storage (NoSQL сховище ключ-значення)

Це сервіс NoSQL бази даних. Не плутайте його з традиційними SQL базами даних, такими як SQL Server або MySQL. Тут немає жорстких схем або зв'язків."

Що це?

Сховище ключ-значення без схеми. Воно зберігає величезні колекції структурованих, але нереляційних даних. Воно розроблене для дуже швидких запитів, коли ви знаєте ключ даних, які хочете отримати.

Основний сценарій використання:

•Зберігання даних профілю користувача для веб-додатка.
•Дані з пристроїв IoT сенсорів.
•Адресні книги, налаштування додатків або будь-який набір даних, які можна запитати за допомогою унікального ключа.

Ключові поняття:

•**Table (таблиця):** Колекція сутностей (не SQL таблиця).
•**Entity (сутність):** Один запис, як рядок у таблиці. Може містити до 255 властивостей.
•**Properties (властивості):** Пара ключ-значення, як стовпець у таблиці.
•**PartitionKey та RowKey:** Це складний первинний ключ. Разом вони формують унікальну адресу для сутності, що дозволяє здійснювати миттєвий пошук. Всі сутності з однаковим PartitionKey зберігаються разом.

Питання

1. Що таке Entity в Azure Table Storage і з яких елементів вона складається?
2. Для чого використовується PartitionKey і як він впливає на продуктивність запитів?
3. Яку роль виконує RowKey, та чому разом із PartitionKey він формує унікальний ключ сутності?
4. Що таке Timestamp і хто відповідає за його оновлення — клієнт чи система?
5. Які обмеження та правила існують для Properties у сутності Azure Table Storage?
6. Що таке точковий запит (Point Query) і чому він вважається найшвидшим?
7. У яких випадках використовують діапазонний запит (Range Query) та які поля можуть у ньому фігурувати?
8. Як працює фільтрація в Azure Table Storage і які оператори порівняння доступні?
9. Чому сортування в Azure Table Storage обмежене та у яких випадках можна отримати дані в певному порядку?
10. Чому складні комбіновані запити (AND/OR) можуть бути менш ефективними в порівнянні з простими точковими?
11. У яких випадках Azure Table Storage є кращим вибором порівняно з реляційними базами даних?
12. Як Azure Table Storage підходить для зберігання даних телеметрії або логів?
13. Які переваги має Table Storage для сценаріїв з високою масштабованістю та великим обсягом даних?
14. Які типові обмеження треба враховувати при проєктуванні моделі даних в Azure Table Storage?
15. Як Table Storage використовується у зв’язці з іншими Azure сервісами, наприклад Azure Functions або Logic Apps?