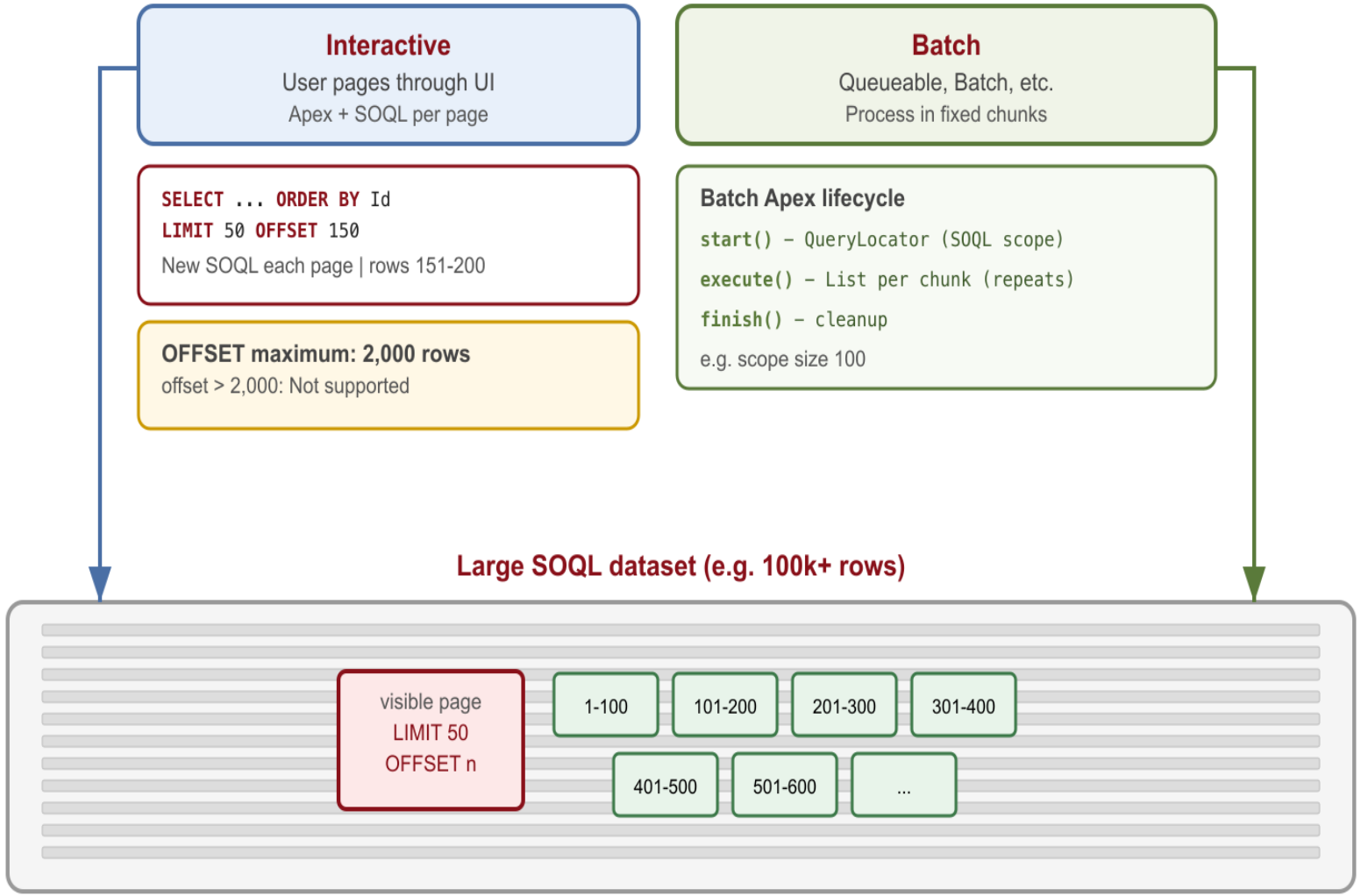
Scaling Apex and LWC with Apex Cursors

#LDNsCall #LC26

Andrew Fawcett, AndyInTheCloud Consulting Ltd
andy@andyinthecloud.com
https://www.linkedin.com/in/andyfawcett/

Accessing and processing datasets today in SOQL

UI paging with OFFSET/LIMIT vs async processing in fixed-size chunks.



What is a SQL Cursor?

Traverse a query result incrementally instead of returning the full set at once.

1. One declared result set - position moves along rows

2. Lifecycle pattern (SQL statements)

1. `DECLARE c CURSOR FOR <select>;`
   Defines query scope
2. `OPEN c;`
   Before first row
3. `FETCH ... FROM c INTO ...;`
   NEXT, PRIOR, ABSOLUTE...
4. `CLOSE c;`
   Release resources
5. `DEALLOCATE c;` (some products)
   Drop definition

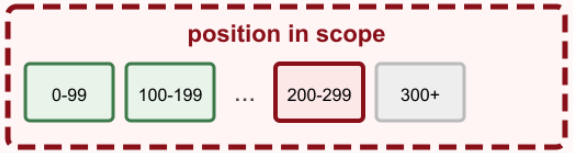
Example

```
DECLARE account_cur CURSOR FOR
  SELECT id, name FROM accounts
  WHERE active ORDER BY id;
OPEN account_cur;
FETCH NEXT FROM account_cur ...
CLOSE account_cur;
```

3. Access patterns (how you read rows)

Forward-only cursor (typical default)

`FETCH NEXT` – one row forward
Most common in ETL / streaming loops



Scrollable cursor (when supported)

`FETCH NEXT | PRIOR | FIRST | LAST`
`FETCH ABSOLUTE n | FETCH RELATIVE n`
Jump relative to start or current position (product-dependent)

What is a SOQL Cursor?

SQL cursor concepts on Salesforce: Database.getCursor(SOQL) and incremental fetch in Apex.

1. One SOQL result set - position moves along rows

2. Lifecycle pattern (Apex API)

1. `Database.getCursor(soql);`
   Or PaginationCursor
2. `cursor.getNumRecords();`
   Optional row count
3. `cursor.fetch(offset, count);`
   Repeat until done
4. Hold cursor: LWC or Cache
   Available across requests
5. Governed by Limits.*ApexCursor*
   Apex cursor limits

Example (this repo)

```
Database.Cursor c =
  Database.getCursor(
    'SELECT Id, Name FROM Account
    WHERE Name LIKE 'TEST%'
    ORDER BY Name',
  AccessLevel.USER_MODE);
List<Account> batch =
  c.fetch(offset, pageSize);
```

3. Access patterns (how you read rows)

Database.Cursor

`fetch(offset, count)`
Forward by offset into scope (infinite scroll)

Database.PaginationCursor

`fetchPage(start, pageSize)`
Page-oriented access; CursorFetchResult

position in scope

fetch(200, 100) = rows 200-299

Usecases for Apex Cursors

Optimial handling of larger data volumes

Interactive

- Data Analyst tools and views
- End user list views, reporting and drilldown
- Preview job records to process

```
apex>
Database.PaginationCursor c =
  Database.getPaginationCursor(
    soql, AccessLevel.USER_MODE);
Database.CursorFetchResult r =
  c.fetchPage(start, pageSize);
```

Batch

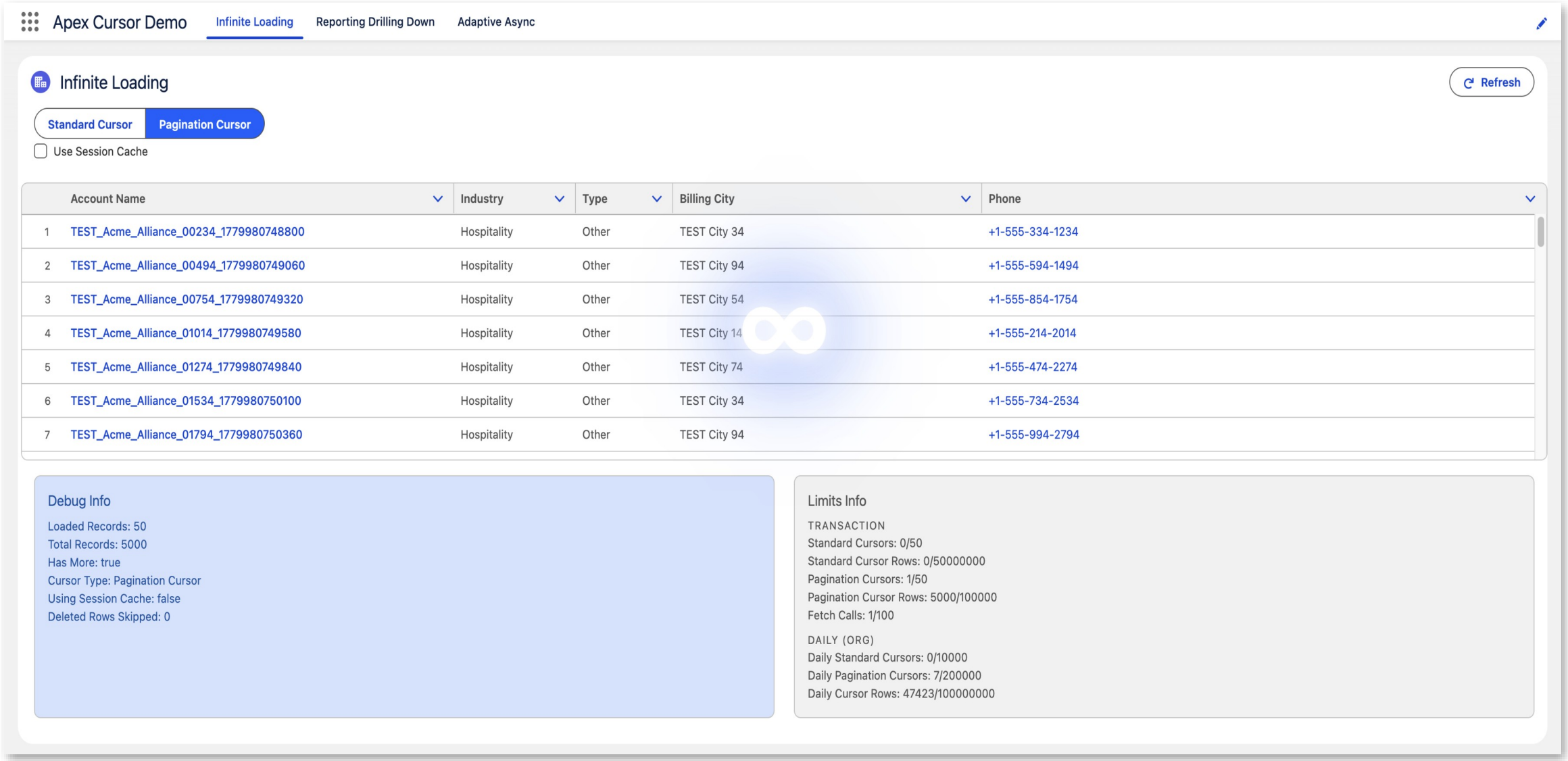
- Orchestrate process execution
- Look ahead to optimize within limits
- Safer immutable record set

```
apex>
Database.Cursor c =
  Database.getCursor(
    soql, AccessLevel.USER_MODE);
List<Account> batch =
  c.fetch(offset, pageSize);
```

Architecture

24hr cached large record sets | Scalable replacement to SOQL OFFSET | Bidirectional navigation

Demo 1: Virtual Data Table



Demo 1: Virtual Data Table Summary

1

```
apex>
cursor = Database.getCursor(
    ACCOUNT_SOQL,
    AccessLevel.USER_MODE);
pageCursor =
  Database.getPaginationCursor(
    ACCOUNT_SOQL,
    AccessLevel.USER_MODE);
```

2

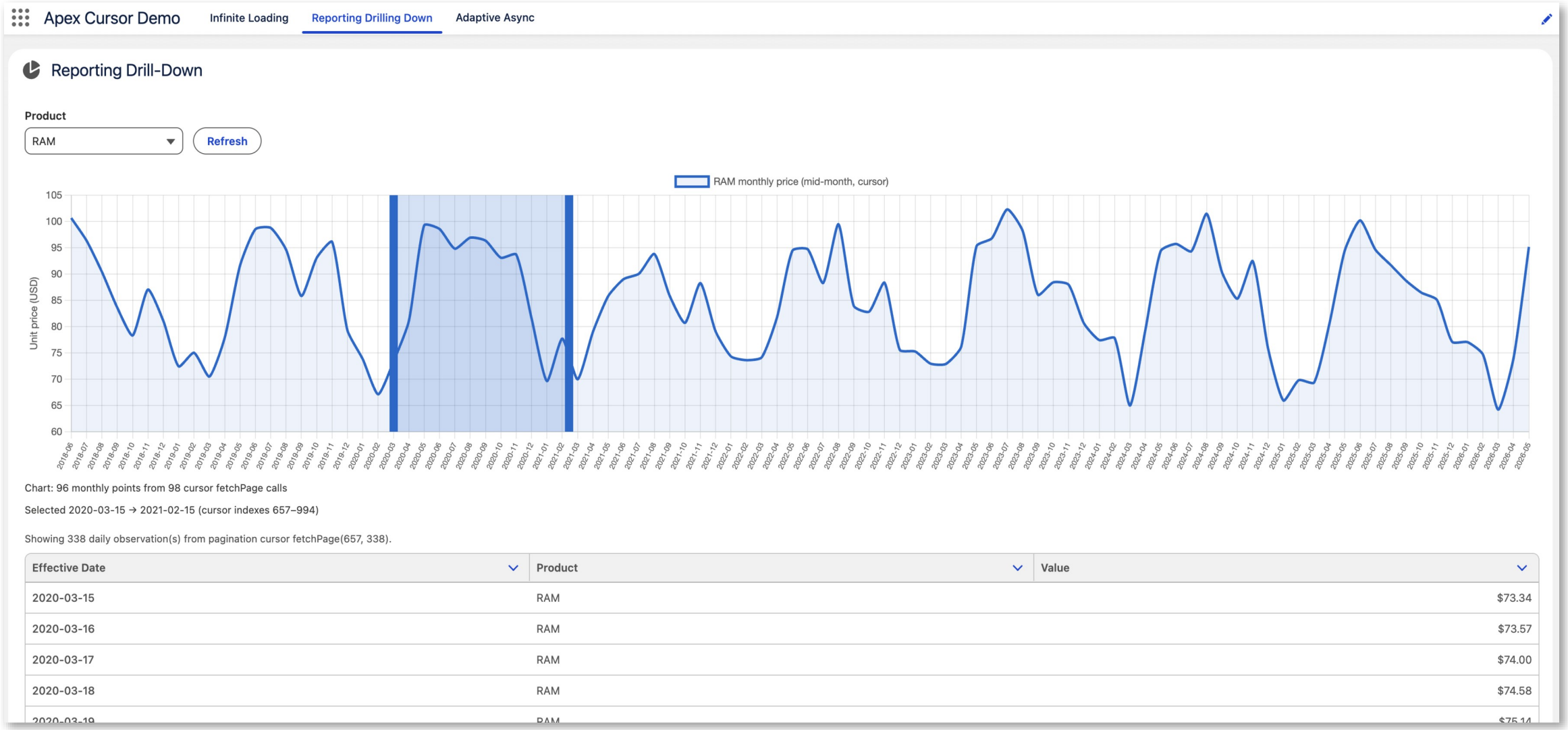
```
apex>
if (pageCursor == null) {
  pageCursor = Database.getPaginationCurs
    ACCOUNT_SOQL, AccessLevel.USER_M
  Cache.Session.put(
    SESSION_PAGINATION_CURSOR_KEY,
    pageCursor);
}
```

3

```
apex>
Integer remaining = totalRecords - start;
Integer fetchSize = remaining > 0
    ? Math.min(pageSize, remaining) : 0;
fetchResult = fetchSize > 0
    ? pageCursor.fetchPage(start, fetchSize)
    : null;
```

- Standard Cursor or PaginationCursor — same ACCOUNT_SOQL, different paging APIs
- Cache.Session keeps the pagination cursor between requests; session cache and cursors are per user
- Page size is capped to remaining rows in the result set
- Limits panel shows cursor, fetch, and daily governor usage

Demo 2: Interactive Reporting and Drilldown



Demo 2: Interactive Reporting and Drilldown Summary

1

```
soql>
SELECT Id, EffectiveDate__c,
  Value__c, Product__c
FROM UnitPriceObservation__c
WHERE Product__c = :product
ORDER BY EffectiveDate__c ASC
```

2

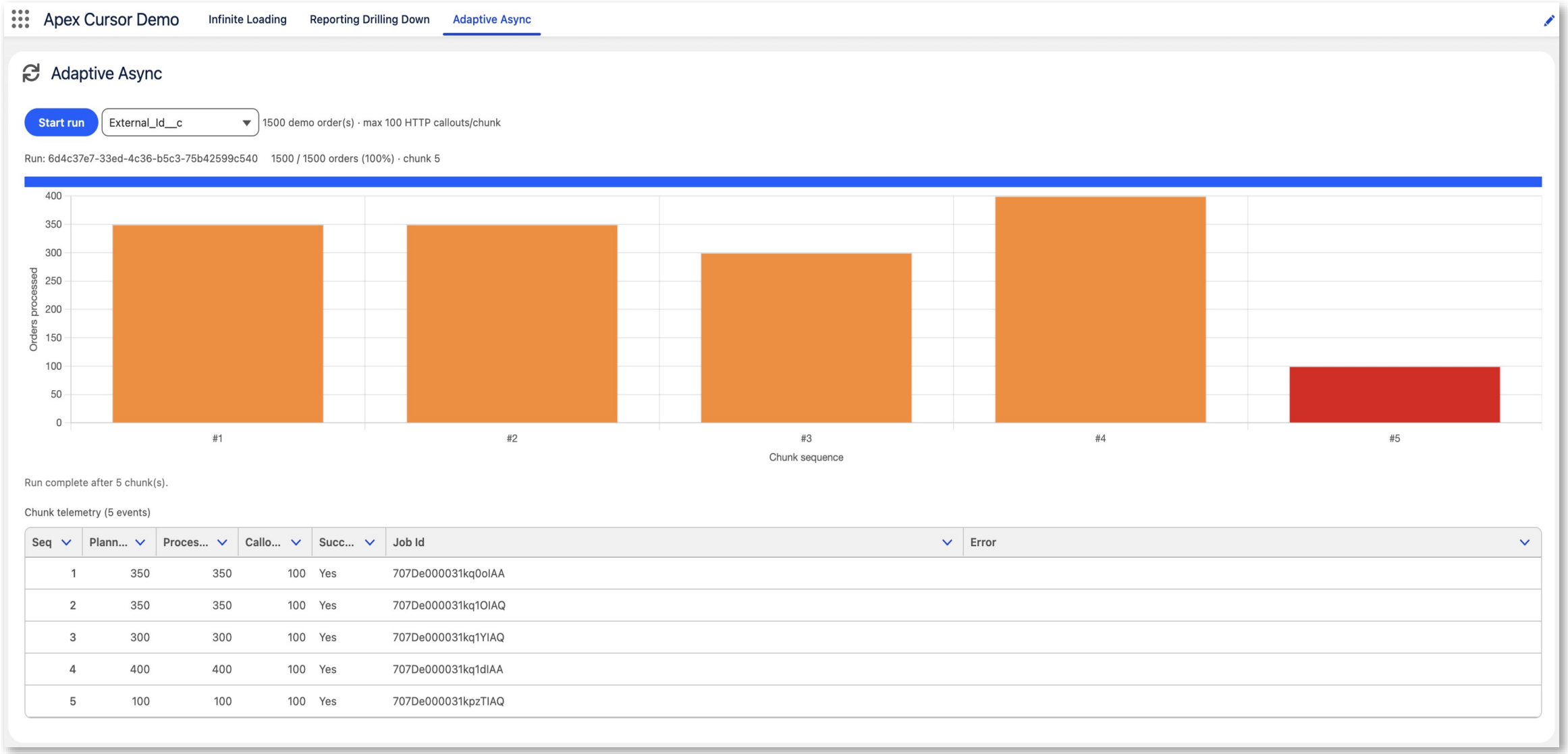
```
apex>
Database.CursorFetchResult
   midChunk =
      pageCursor.fetchPage(dayIndex, 1);
```

3

```
apex>
Database.CursorFetchResult
    drilldownRecords =
      pageCursor.fetchPage(
        drillStart, drillSize);
```

- PaginationCursor builds the monthly chart with one fetchPage per month
- Session cache keeps the cursor for brush-selection drill-down
- Index-based fetchPage maps chart range to daily observation rows
- fetchPage reports deleted rows when underlying data changes

Demo 3: Adaptive Async



Demo 3: Adaptive Async Summary

1

```
apex>
Database.Cursor cursor =
  Database.getCursor(
    soql,
    AccessLevel.USER_MODE);
System.enqueueJob(
  new AdaptiveOrderWorkerQueueable(
    runId, cursor, 1), options);
```

2

```
apex>
Integer rowCalloutsRequired =
   rowCalloutsFor(row);
Boolean canProcessRow =
   rowCalloutsRequired == 0 ||
       scope.predictedCallouts +
      rowCalloutsRequired <= calloutBudget;
```

3

```
apex>
if (Limits.getFetchCallsOnApexCursor() >=
    Limits.getLimitFetchCallsOnApexCursor()) {
     return scope;
}
```

- One cursor opened in Apex and passed through chained queueable jobs
- Chunk size adapts to remaining HTTP callout budget per transaction
- Track fetch limits dynamically via Limits.getFetchCallsOnApexCursor
- Cursor position travels on the job state

Usage Best Practices

Stable ORDER BY

```
soql>
'SELECT ... FROM Account
  WHERE Name LIKE 'TEST%'
  ORDER BY Name',
  AccessLevel.USER_MODE);
```

Sort on a unique, stable key.
Keeps offsets and page indexes predictable.

Guard number of records

```
apex>
if (cursor.getNumRecords() == 0) {
  return result;
}
```

Exit early when the result set is empty.
fetch(0, 0) can throw a GACK.

Cap fetchPage size

```
apex>
pageSize = Math.min(
  PAGE_SIZE,
  pc.getNumRecords()
    - startIndex);
```

Do not request rows past the result set end.
Critical for partial last pages.

Retry transient errors

```
apex>
} catch (System.
  TransientCursorException e) {
  if (++retries <= MAX)
    return retry();
```

TransientCursorException may be retried.
Do not retry FatalCursorException.

Cursor returns to LWC

```
apex>
@AuraEnabled
public static Result load(...) {
  result.cursor = cursor;
  return result;
```

Cursors are Aura-serializable.
Return on @AuraEnabled result.

Check for latest security changes

```
apex>
[SELECT Id FROM Account
  WHERE Id IN :ids
  WITH USER_MODE];
cursor = Database.getCursor(
```

Fetch does not re-evaluate security, per Batch Apex. If critical requery changes.

Limits

Limit	Pagination	Standard
Per transaction (each Apex request)		
Cursor instances open	50	50
Rows across all cursors in txn	100,000	50,000,000
Fetch calls (shared)	100	100
Per pagination fetchPage call		
Max rows per page	2,000	—
Per org (rolling 24 hours)		
Cursor instances opened	200,000	10,000
New rows fetched (shared)	100,000,000	100,000,000

- Transaction limits apply to each Apex request; daily limits roll per org over 24 hours.
- Fetch calls and daily row counts are shared across standard and pagination cursors.
- Cursors expire like API query cursors — see API Query Cursor Limits in Salesforce Help.
- Aggregate, subselect and virtual object queries are not supported.

Other Use Cases

Cursor patterns beyond the three demos

Records to Process Adjustment

Present records to process and let users deselect rows. Pass the cursor and row indexes to ignore into the job.

Repeatable Datasets

Retry processing against the exact original record set.

Point in Time Reporting

Snapshot a record-set selection for the user session.

Simple Queue

Create task records and delete when complete — until cursor rows reach zero.

LONDON'S CALLING

Q&A

Andrew Fawcett
andy@andyinthecloud.com | LinkedIn

#LDNsCall #LC26

View Session Resources

LONDON'S CALLING

SmarterPay

PDF BUTLER flexpricer provar Gearset Nutrient SoClo INVENTORY CLOUD PERSPECTIVE

WITH secure avonni TITAN Elements .cloud Super mums Vinton From Conversations to Data

testsigma autorabit ClickLearn COPADO FormAssembly

Thank You

#LDNsCall #LC26

Andrew Fawcett

andy@andyinthecloud.com | LinkedIn